COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

Bootcamp

Inteligencia Artificial

Nivel (Intermedio)

Docente: Víctor Viera Balanta

Fecha 29/07/2024

UT TALENTOTECH

cymetria | tecnalia colombia

Tabla de contenidos

Profesor: Víctor Viera Balanta

Random Forest Classifier

El clasificador de bosque aleatorio crea un conjunto de árboles de decisión a partir de un subconjunto del conjunto de entrenamiento seleccionado aleatoriamente. Es un conjunto de árboles de decisión (DT) de un subconjunto seleccionado aleatoriamente del conjunto de entrenamiento y luego recopila los votos de diferentes árboles de decisión para decidir la predicción final.

Profesor: Víctor Viera Balanta

UT TALENTOTECH

Cronograma

Cronograma tentativo del Proyecto

	Julio				Agosto				Septiembre			
Actividades	1	2	3	4	1	2	3	4	1	2	3	4
Introducción												
Problemática												
Descripción Objetivos												
General												
Específicos												
Alance y Limitadones												
Tecnologías Abordadas												
Cronograma del Proyecto												
Desarrollo del proyecto												
Herramientas y Frameworks												
Descripción de datos												
Proceso de Tratamiento de datos												
Modelo o estrategia abordada												
Métricas de entrenamiento												
Métricas de prueba												
Resultados obtenidos												
Clonclusiones												
Referencias y Bibliografía												

Profesor: Víctor Viera Balanta

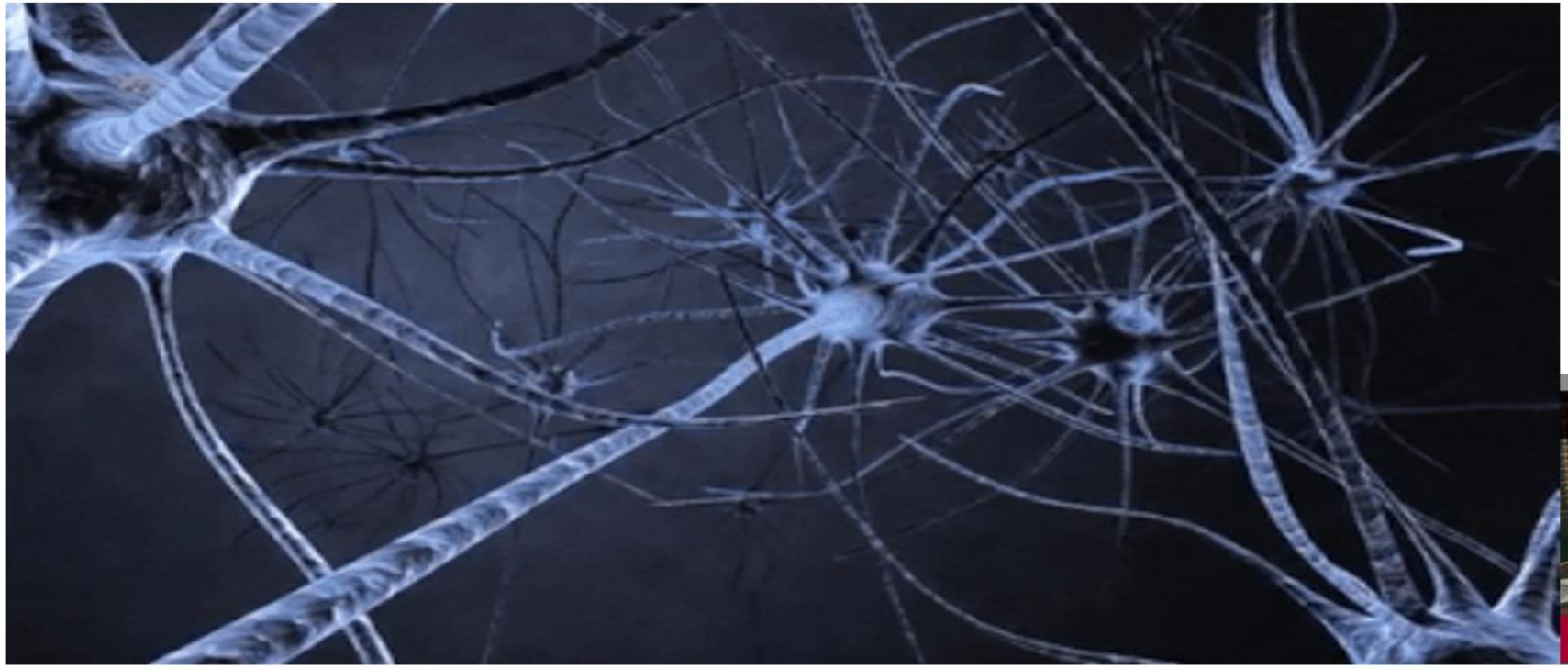
COLOMBIA POTENCIA DE LA VIDA

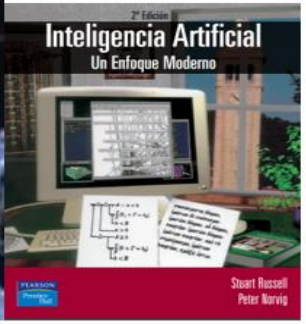
TALENTO TECH

TIC

Redes Neuronales

Representación de una Neurona



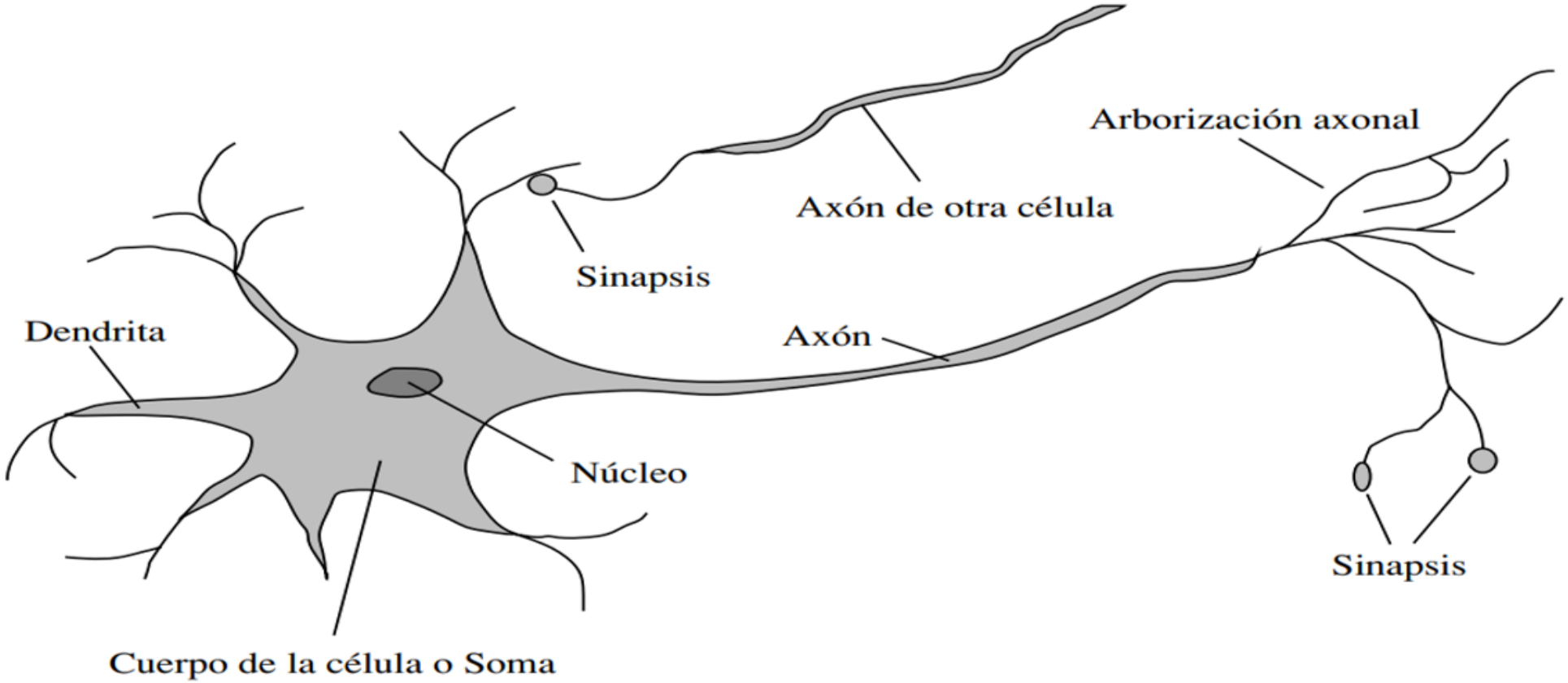


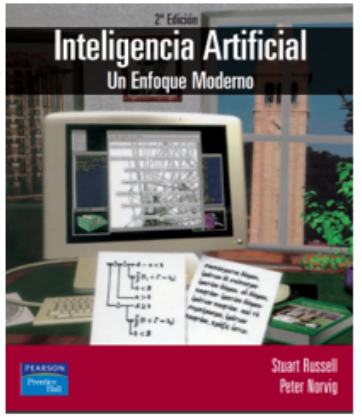
UT TALENTOTECH

cymetria | tecnal:a colombia

Profesor: Víctor Viera Balanta

Representación de una Neurona





Profesor: Víctor Viera Balanta

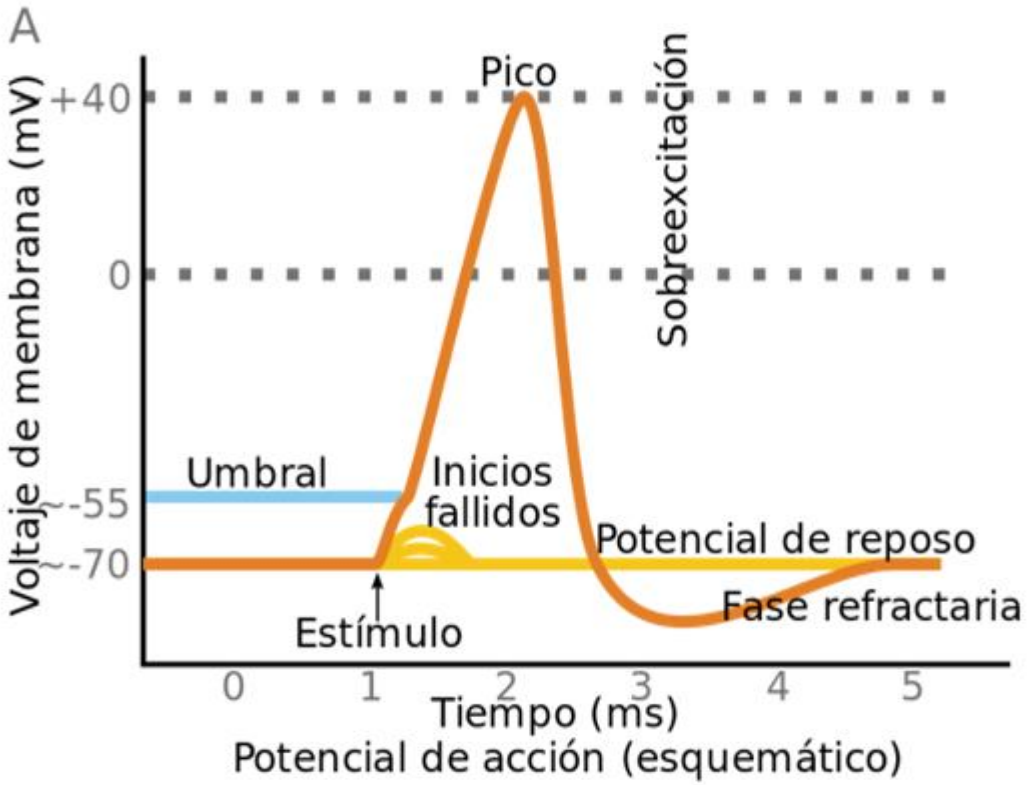
El proceso

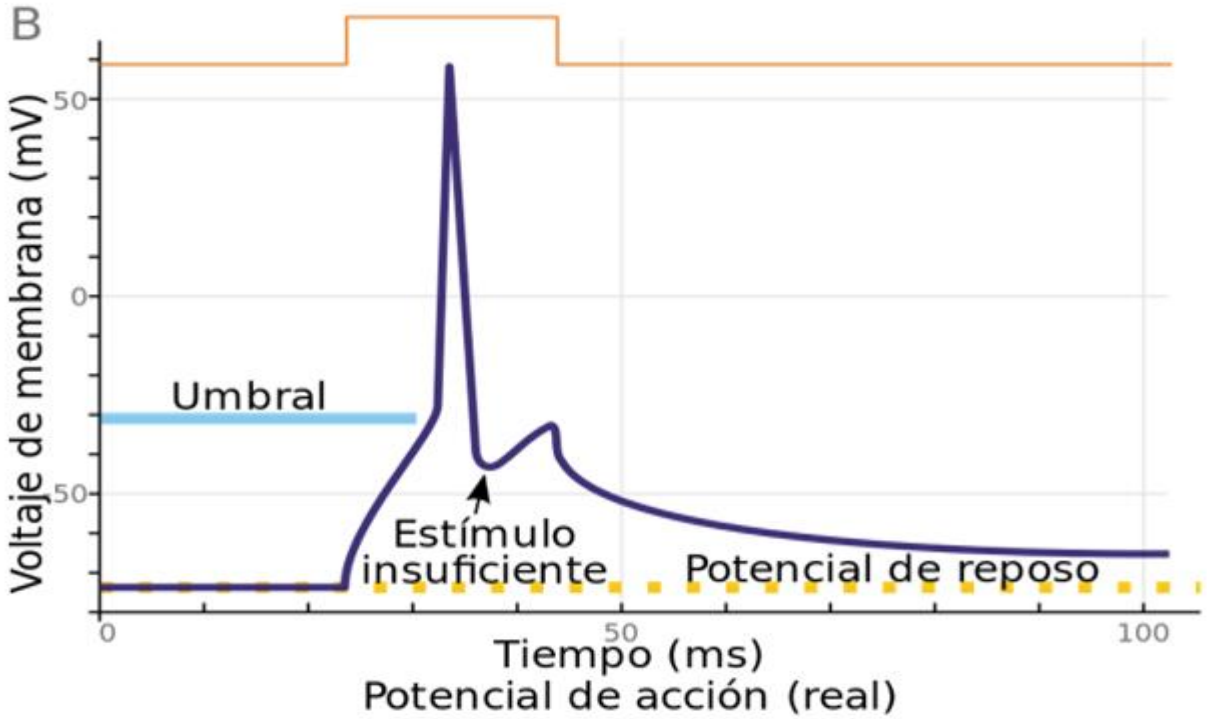
https://youtu.be/TM7LYvYckiE?si=js10QJejazTQla3X

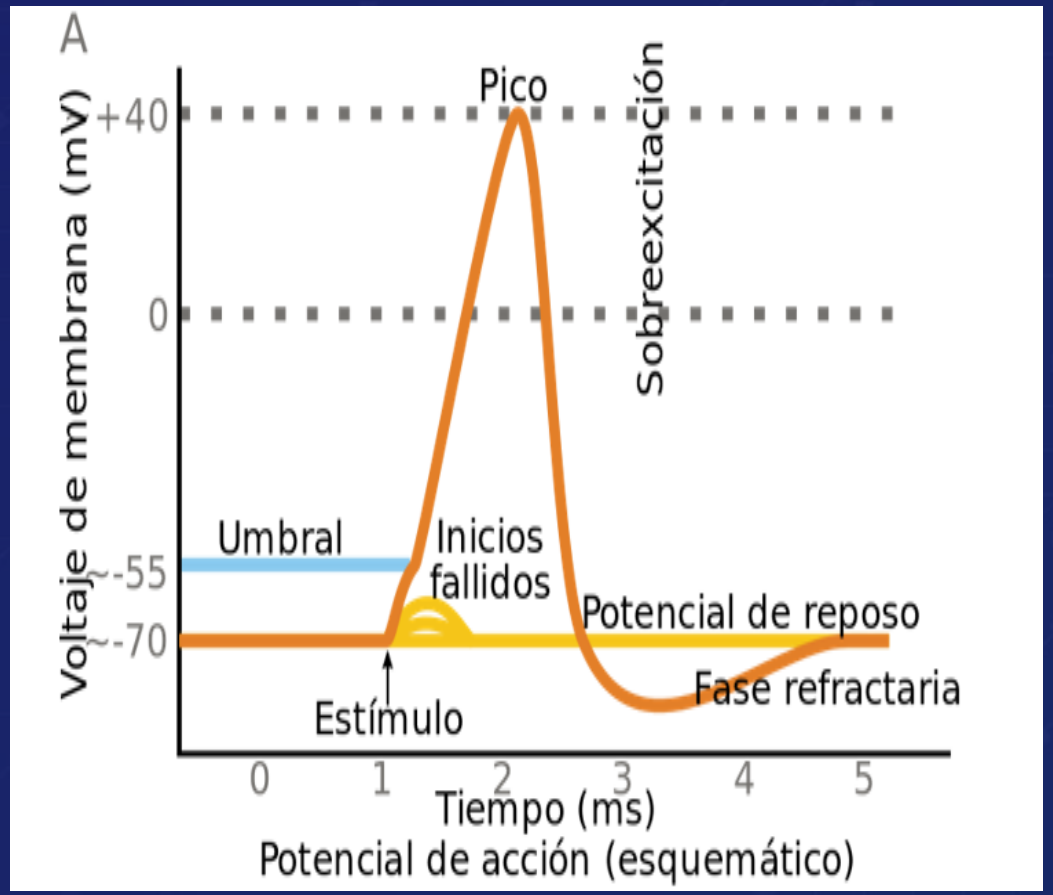
Profesor: Víctor Viera Balanta

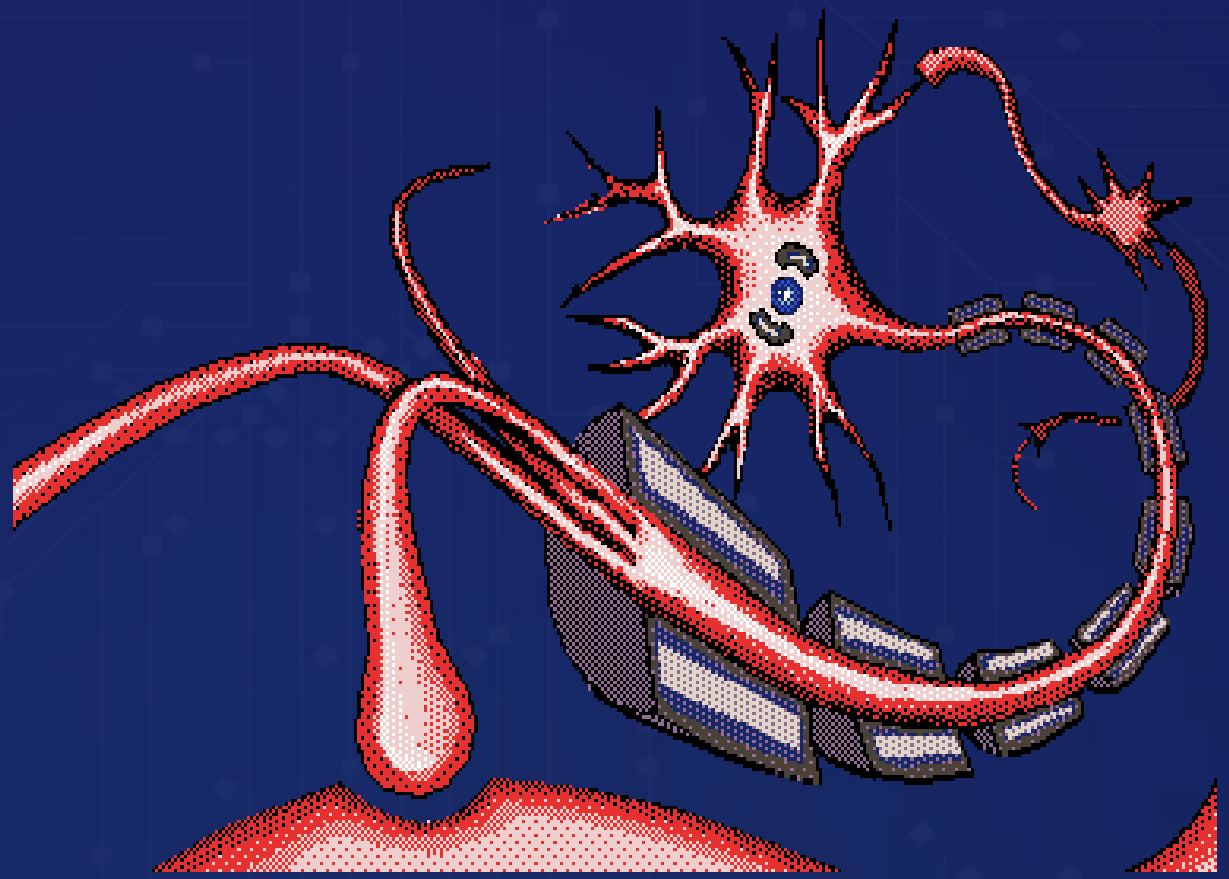
UT TALENTOTECH

Activación de la neurona-potencial de acción



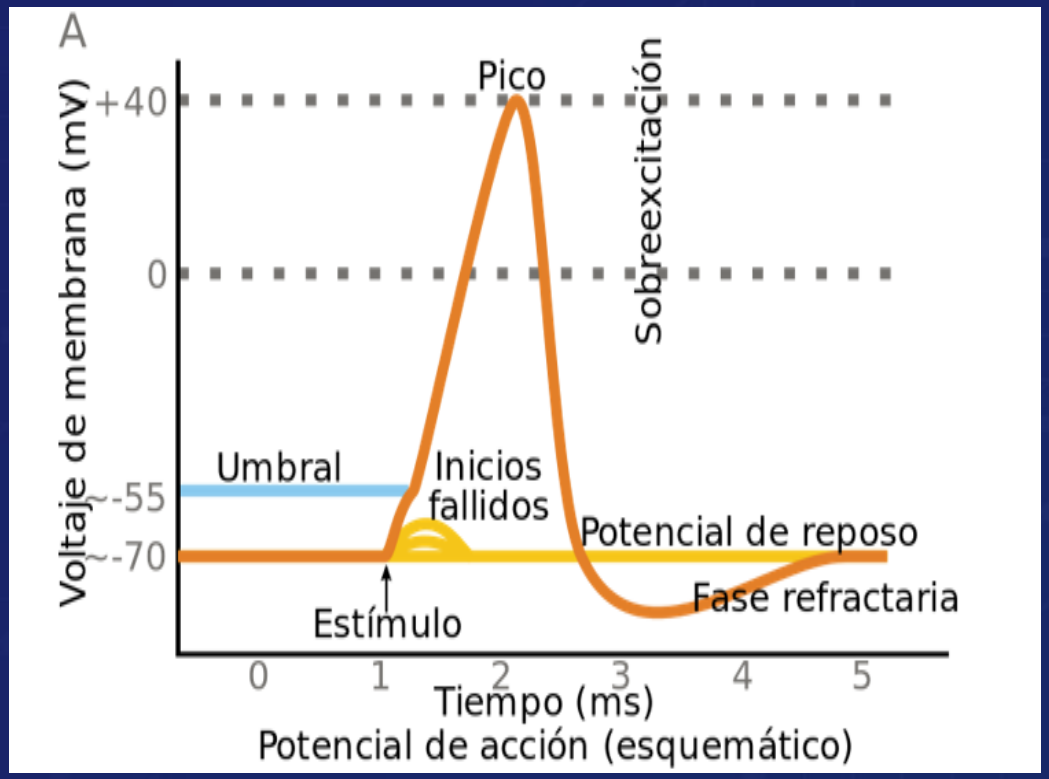


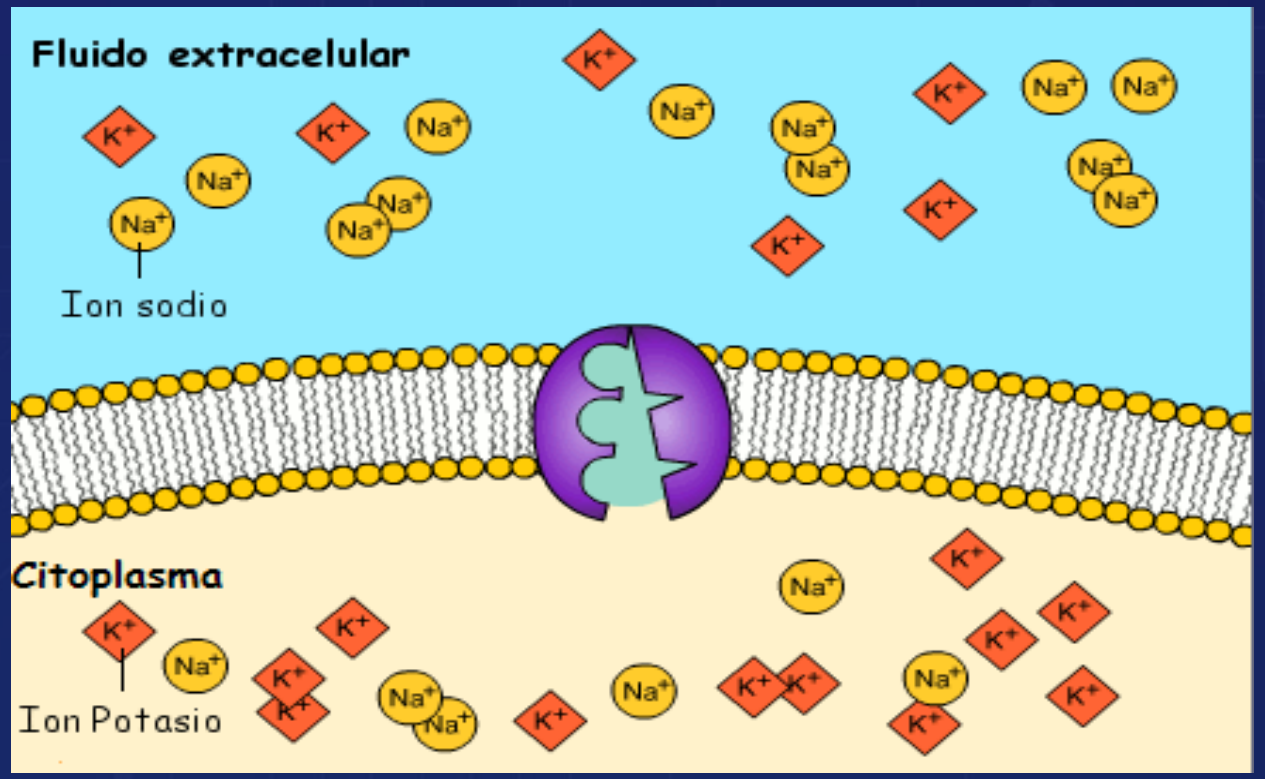




UT TALENTOTECH

Activación de la neurona-potencial de acción





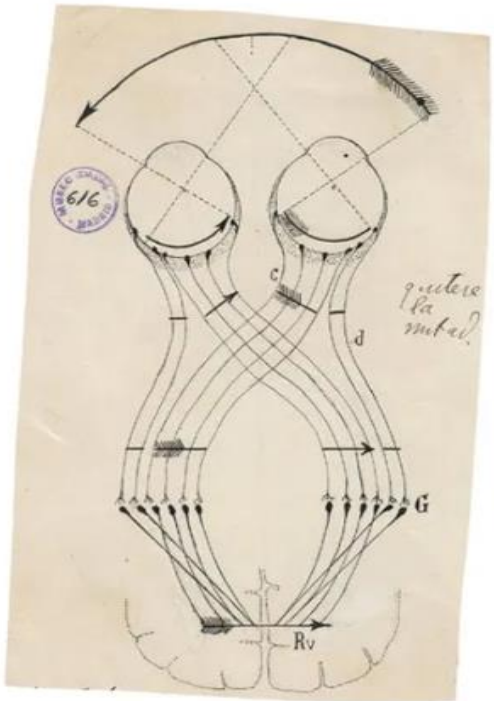
Bomba de sodio potasio

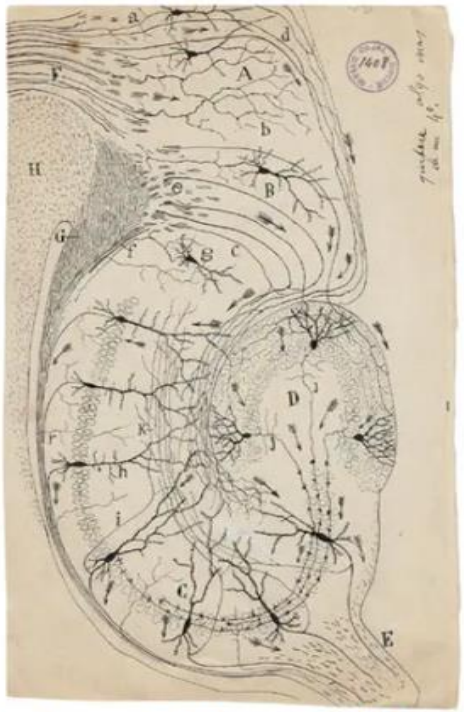
Profesor: Víctor Viera Balanta

Redes Neuronales

Santiago Ramón y Cajal

El Premio Nobel 1906





https://www.nytimes.com/es/2017/02/21/espanol/cultura/santiago-ramon-y-cajal-el-hombre-que-dibujo-los-secretos-del-cerebro.html

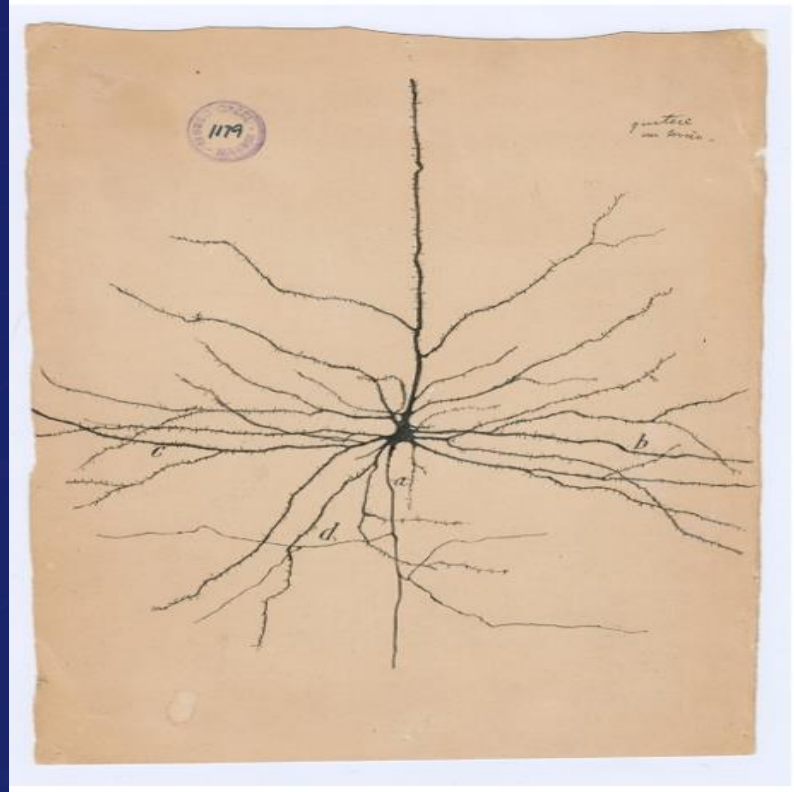
COLOMBIA POTENCIA DE LA VIDA

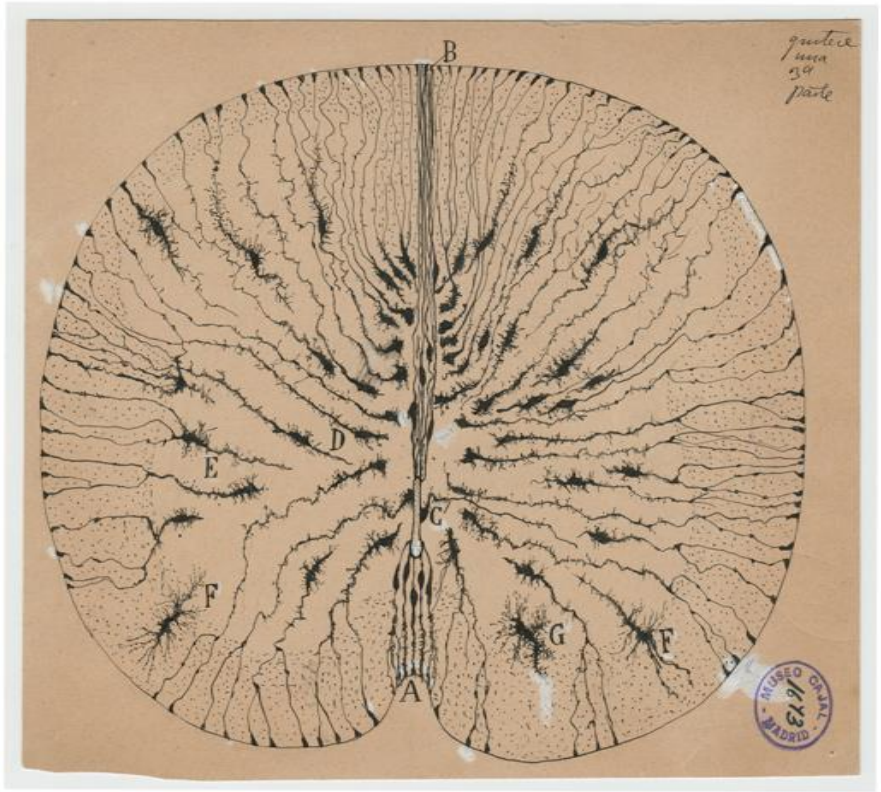
TALENTO TECH

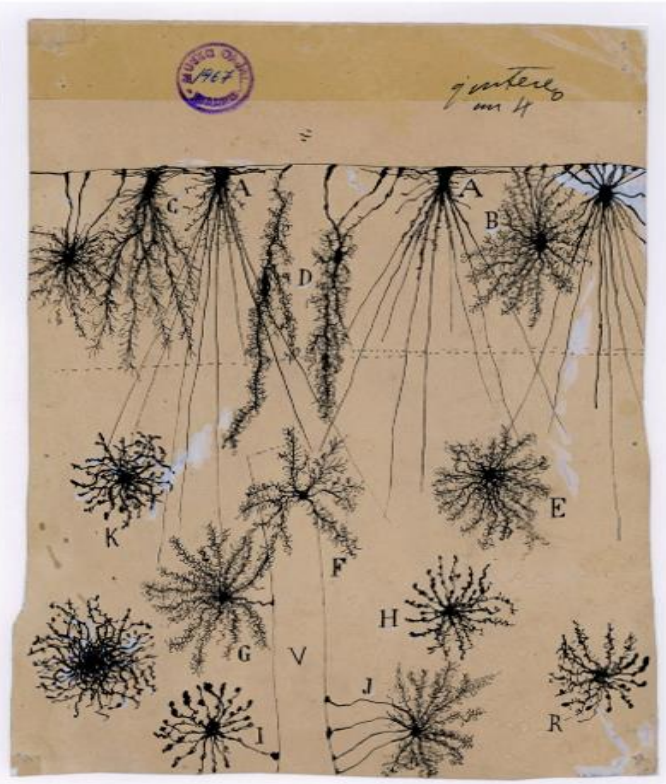
TIC

Redes Neuronales

Dibujos, Santiago Ramón y Cajal





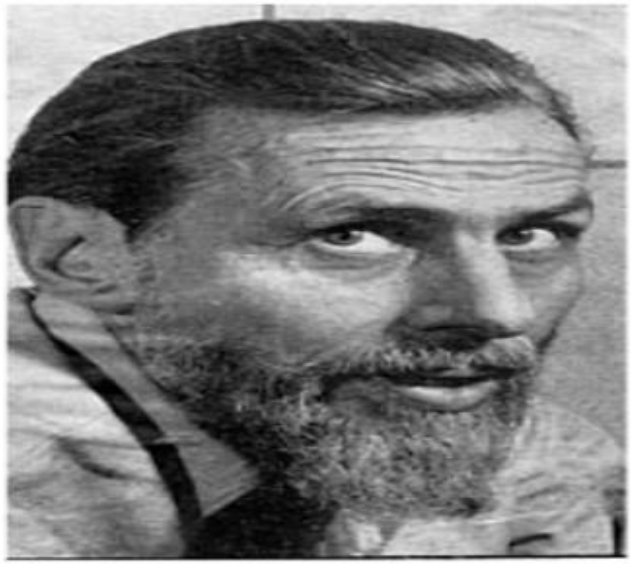


https://www.nationalgeographic.com.es/ciencia/actualidad/una-muestra-con-dibujos-santiago-ramon-cajal-recorre-estados-unidos-canada_11187

Profesor: Víctor Viera Balanta

cymetria | tecnal:a colombia

Warren McCulloch Walter Pitts



1943

Se propone uno de los primeros modelos matemáticos de la neurona

Warren McCulloch Walter Pitts

Bulletin of Mathematical Biology Vol. 52, No. 1/2, pp. 99–115, 1990.
Printed in Great Britain.

0092–8240/90$3.00 + 0.00
Pergamon Press plc
Society for Mathematical Biology

1943

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*

WARREN S. MCCULLOCH AND WALTER PITTS
University of Illinois, College of Medicine,
Department of Psychiatry at the Illinois Neuropsychiatric Institute,
University of Chicago, Chicago, U.S.A.

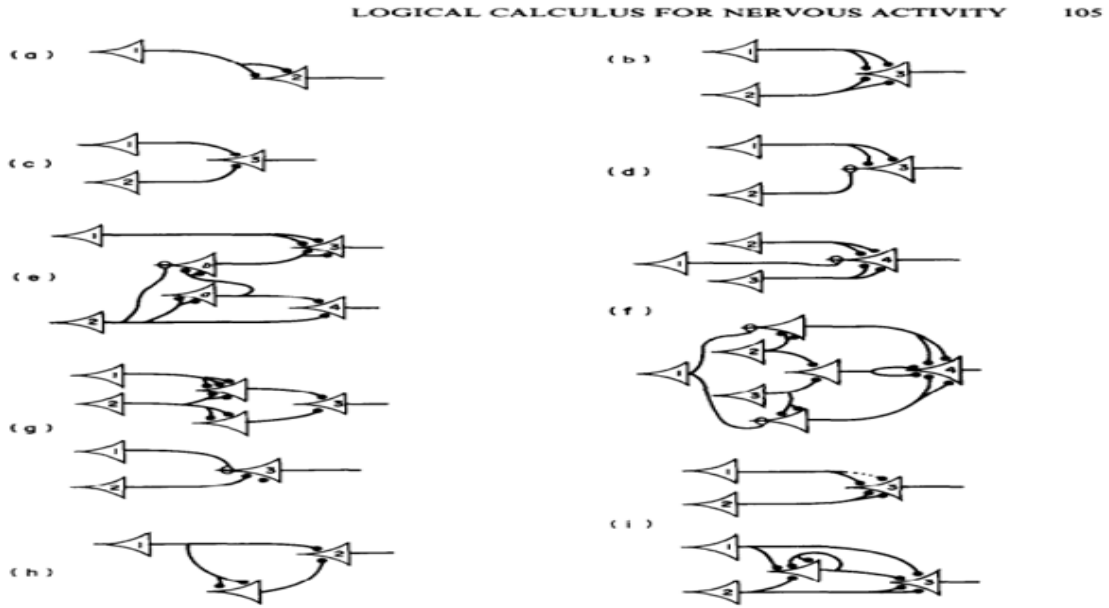
https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf

Se propone uno de los primeros modelos matemáticos de la neurona

Redes Neuronales

Warren McCulloch Walter Pitts

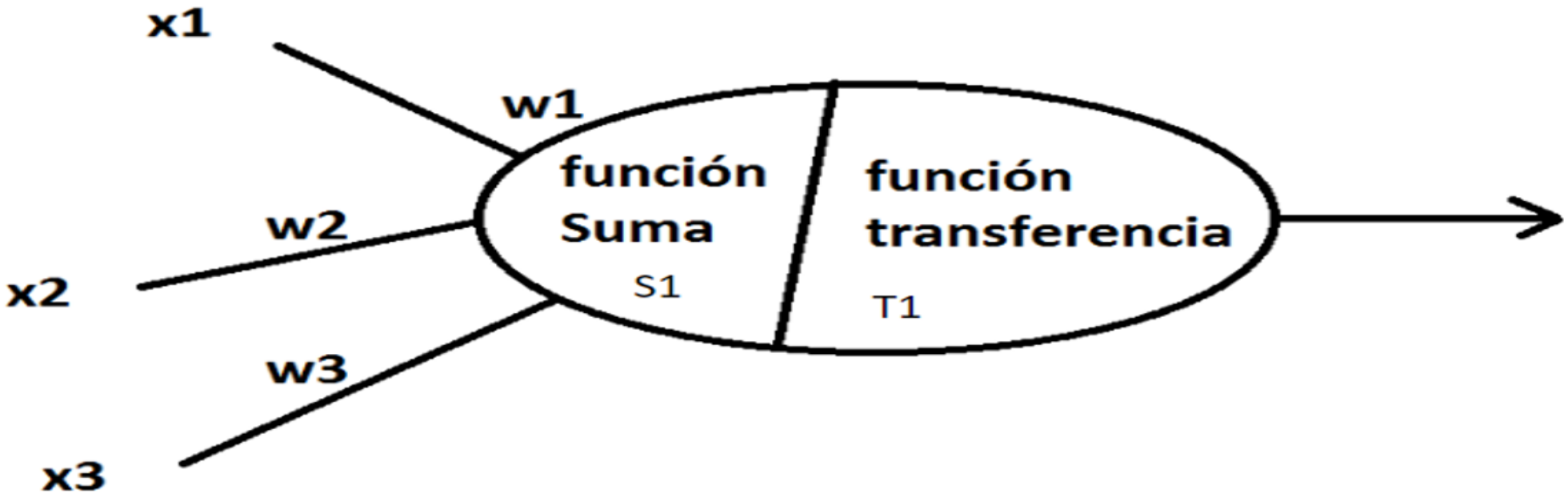
1943



Se propone uno de los primeros modelos matemáticos de la neurona

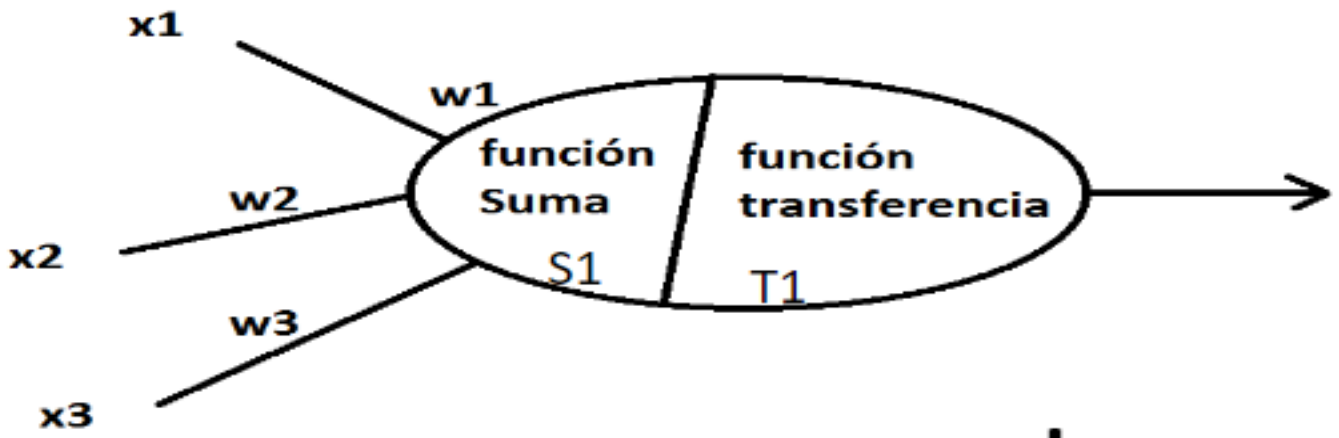
https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf

Redes Neuronales



Se propone uno de los primeros modelos matemáticos de la neurona

Redes Neuronales



h es el Umbral

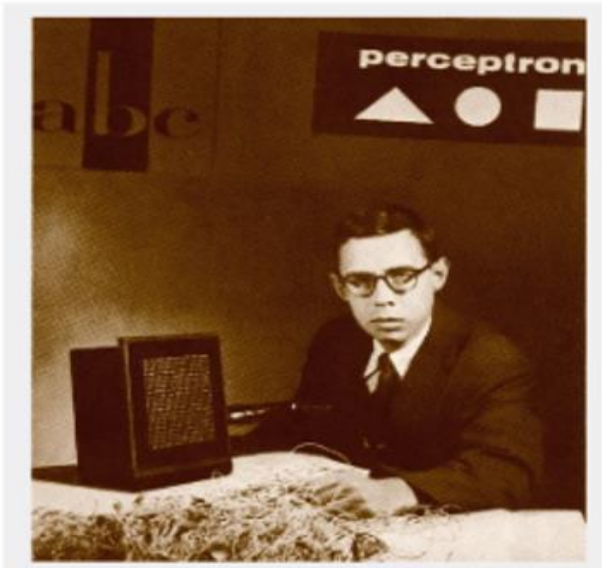
Salida 0 si T1 < h

Salida 1 si T1 > h

Profesor: Víctor Viera Balanta

Redes Neuronales

A finales de 1950 Frank Rosenblatt



Crean los Perceptrones

Doctor en Filosofía a Universidad Cornell,

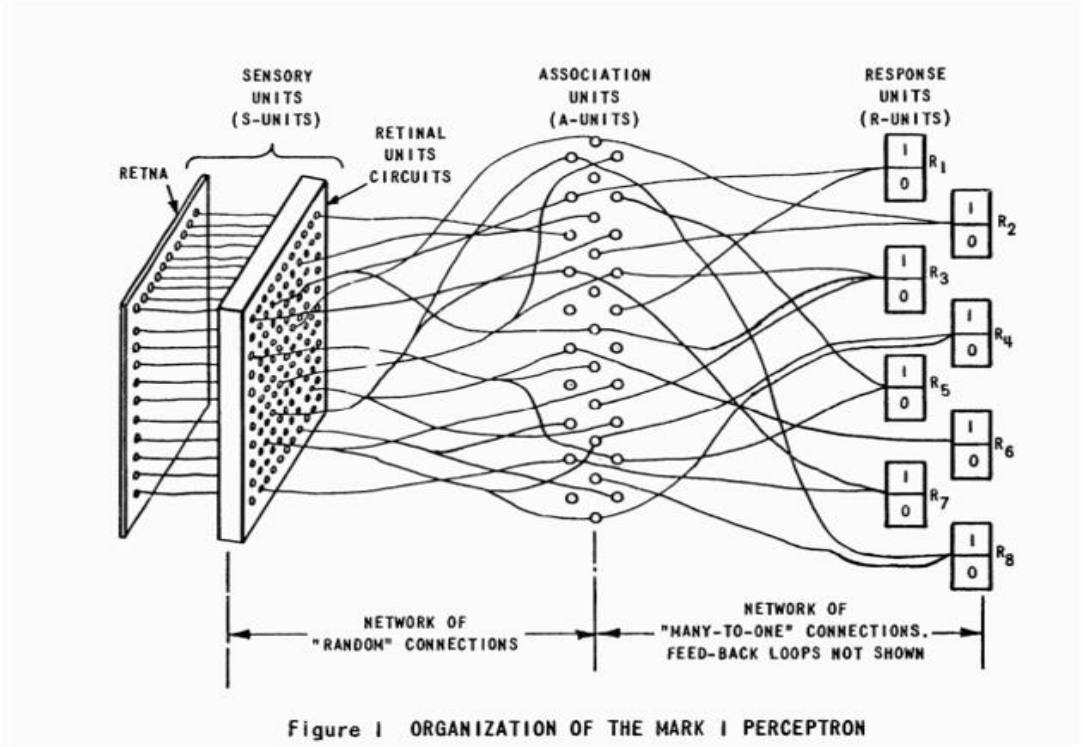
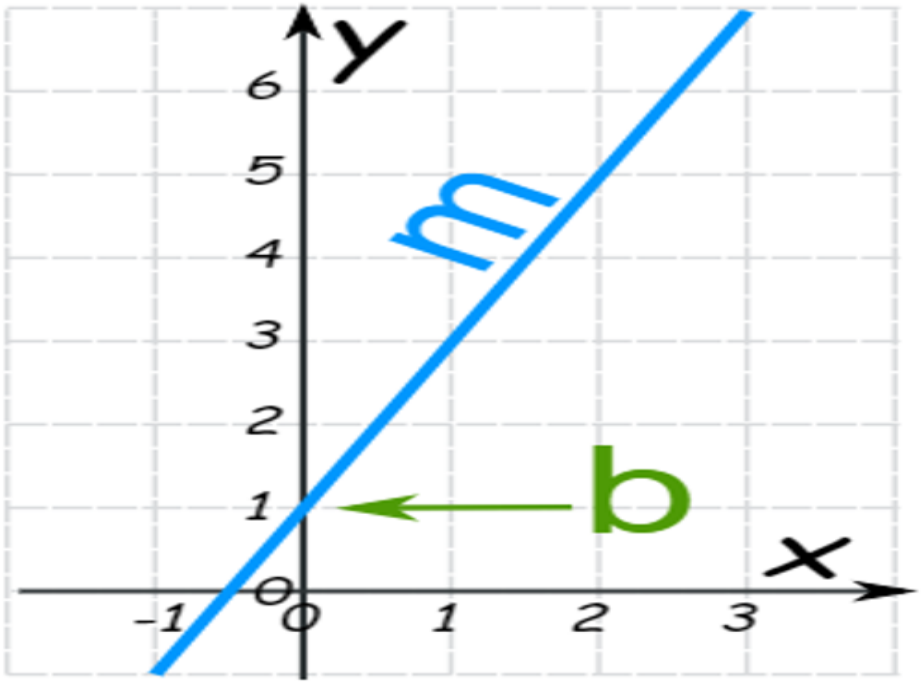


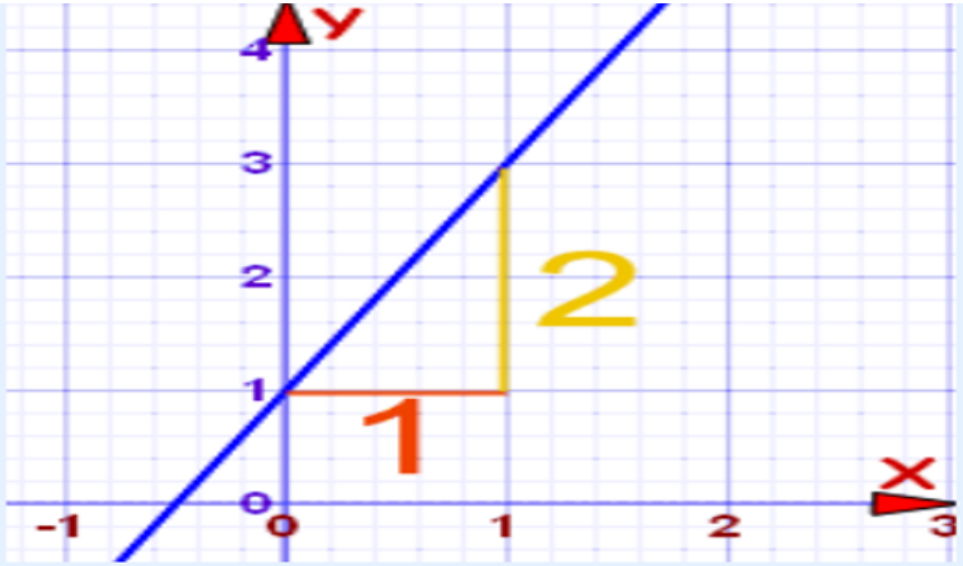
Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf

Redes Neuronales

El sesgo permite darle flexibilidad al modelo, para mover la línea a un corte diferente al origen de coordenadas. "Mecanismo" Matemático del Perceptrón. Por ahora..

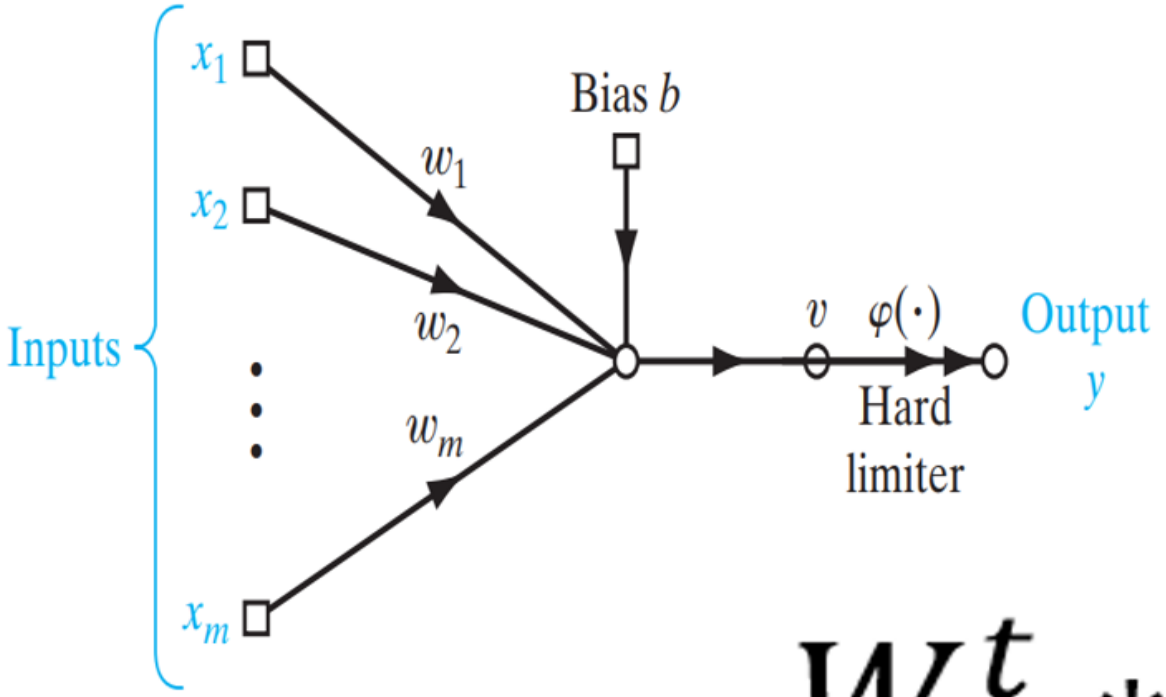




$$y = 2x + 1$$

Profesor: Víctor Viera Balanta

Redes Neuronales

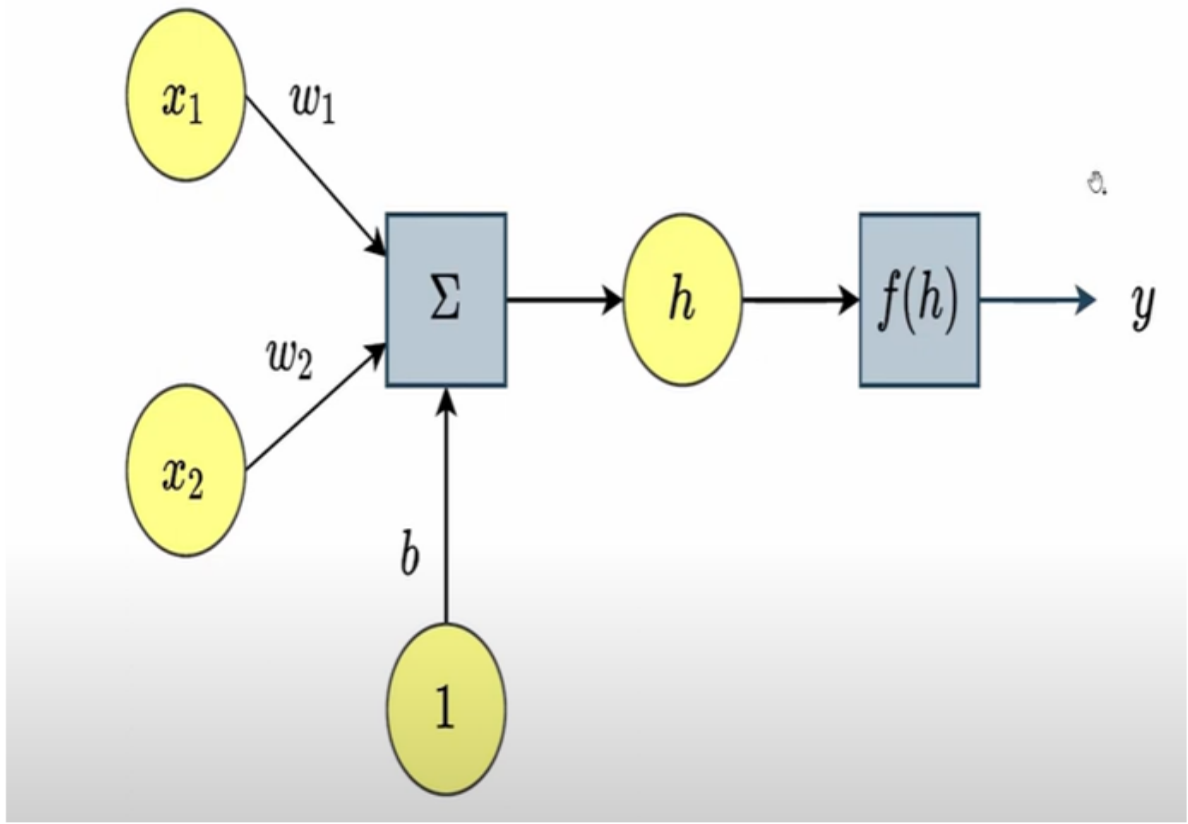


Sesgo o Bias(Inglés)

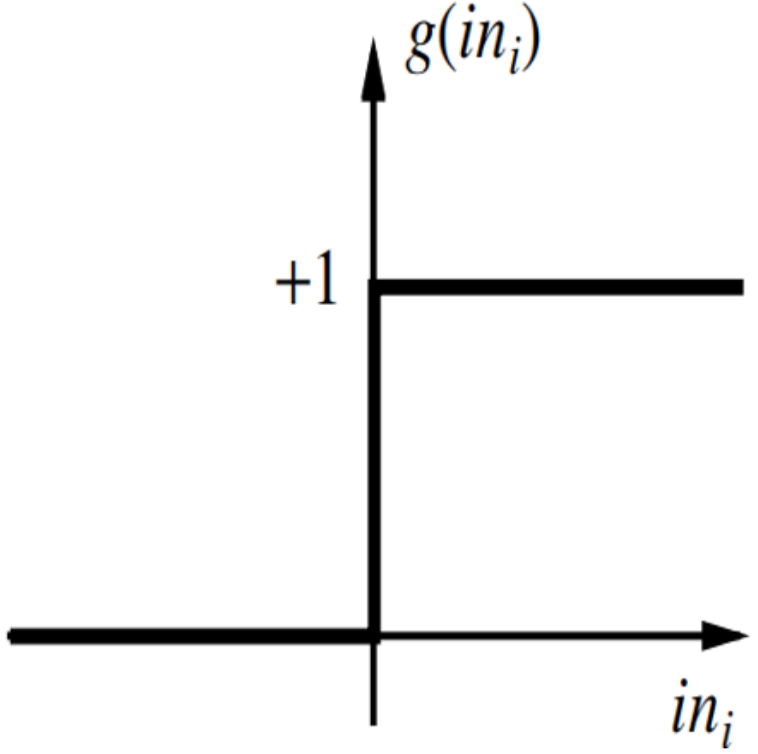
$$v = \sum_{i=1}^{m} w_i x_i + b$$

$$W^t * x + b$$

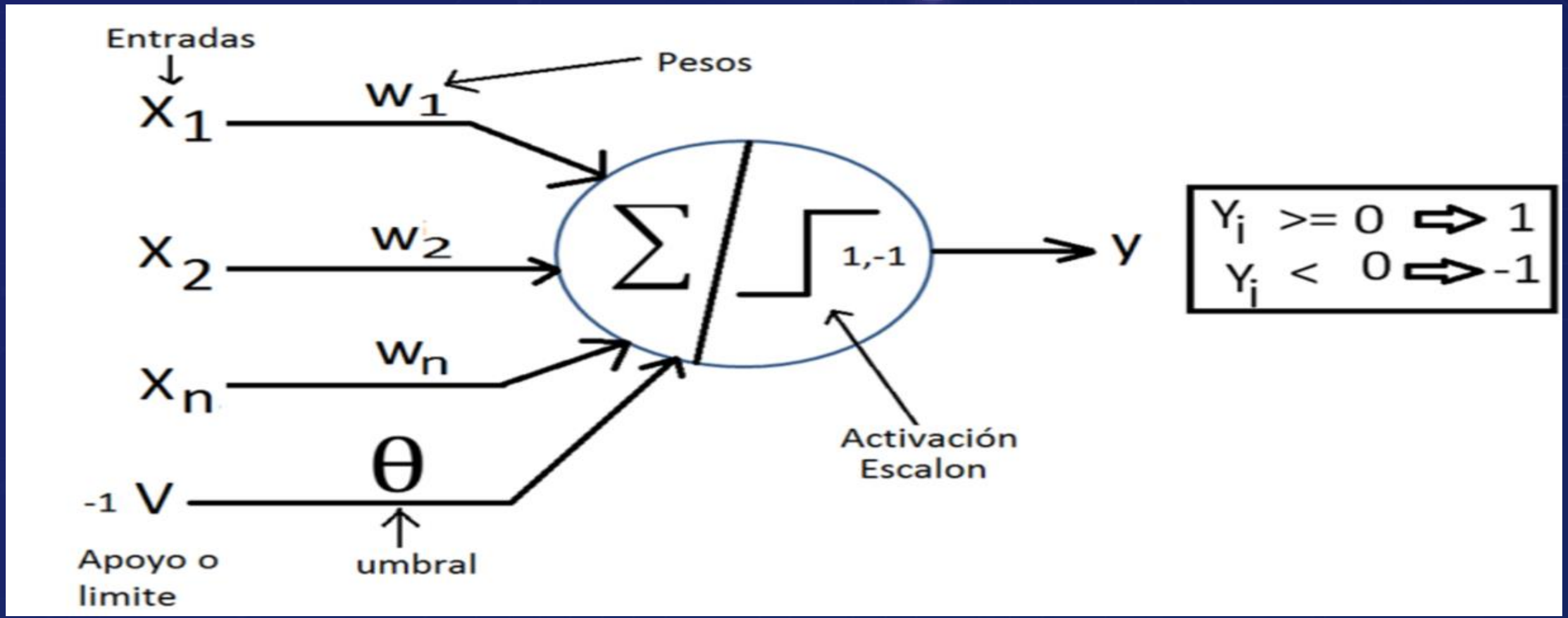
Redes Neuronales



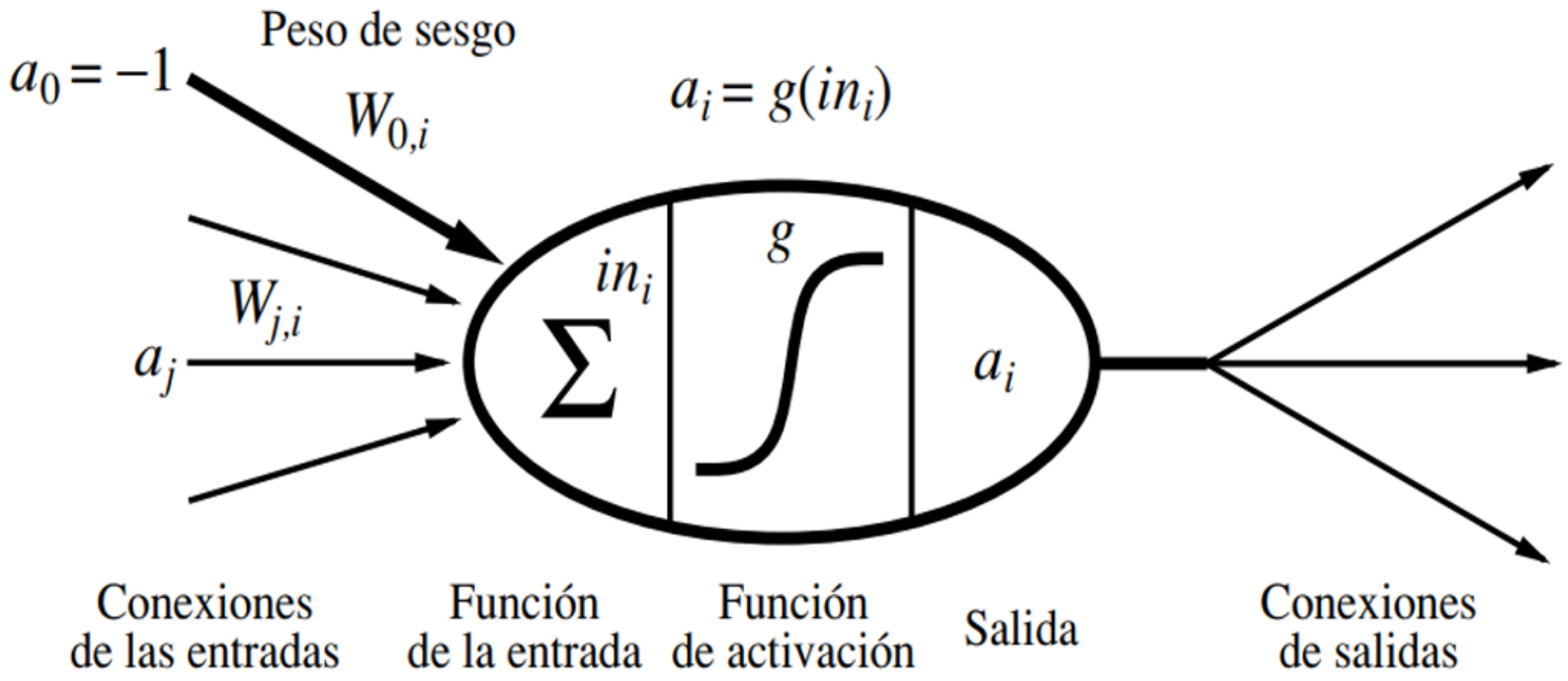
Función Escalón

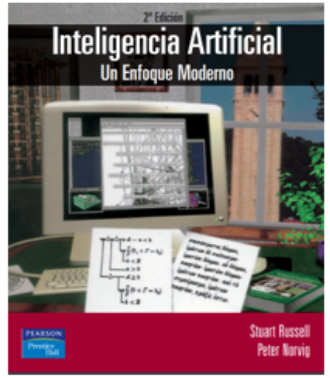


Profesor: Víctor Viera Balanta


Entradas
Pesos
x_1
w_1
x_2
w_2
x_n
w_n
-1 V
θ
Apoyo o limite
umbral
Σ
1,-1
Activación Escalon
y
$Y_i >= 0 \Rightarrow 1$
$Y_i < 0 \Rightarrow -1$

Redes Neuronales

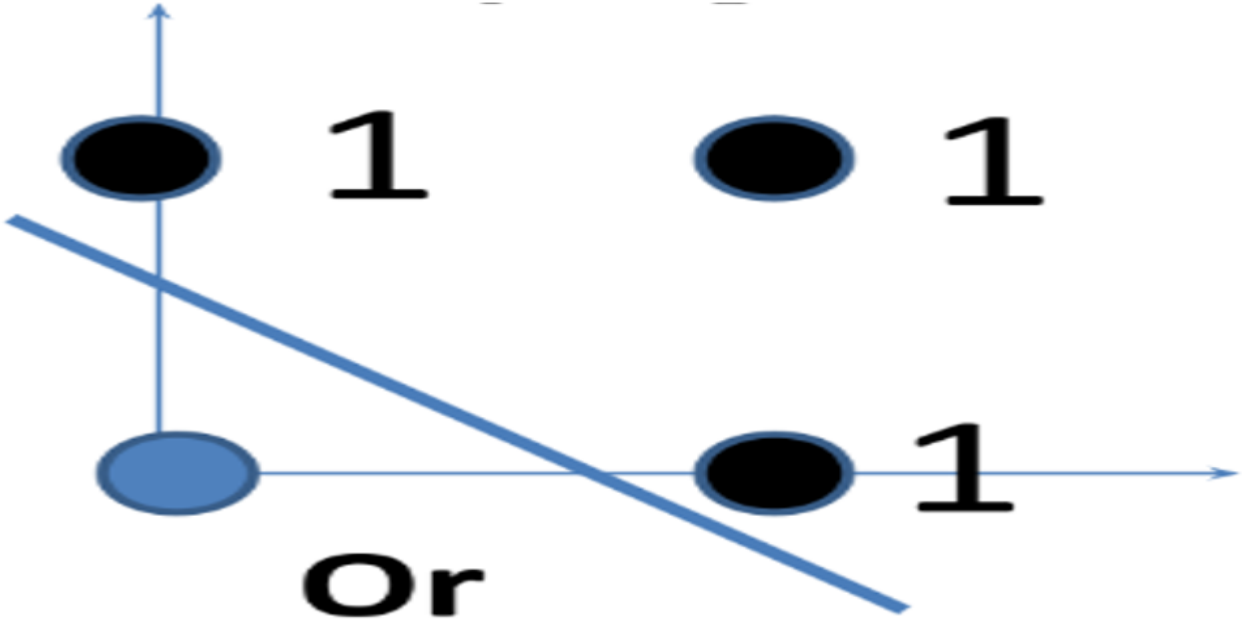




Redes Neuronales

Función OR es linealmente separable

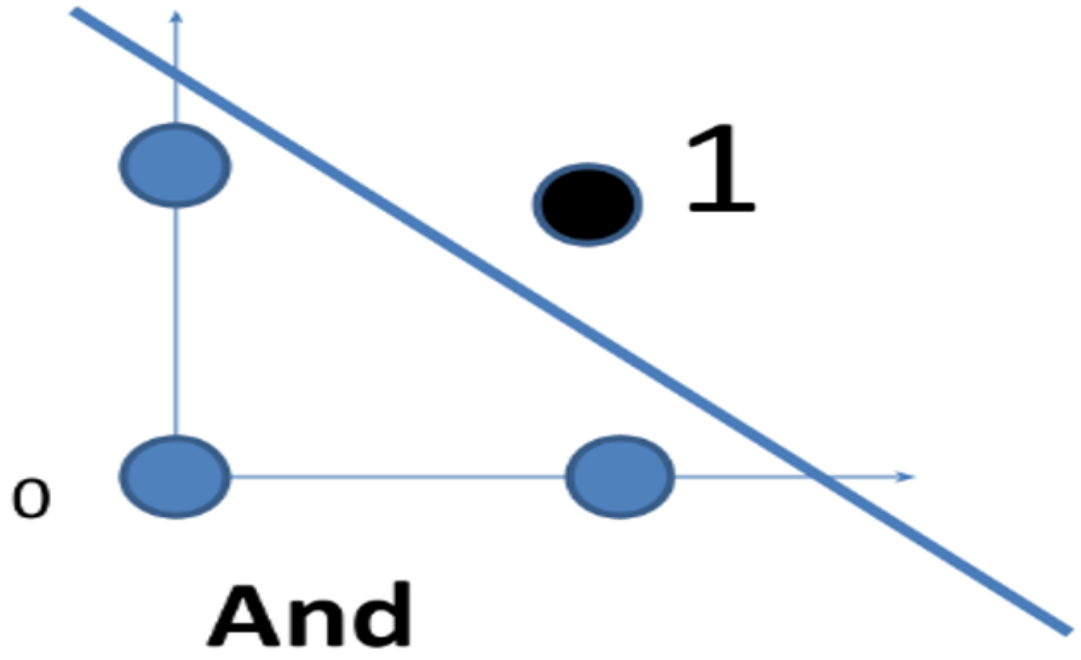
INPUT		OUTPUT
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



Redes Neuronales

Función AND es linealmente separable

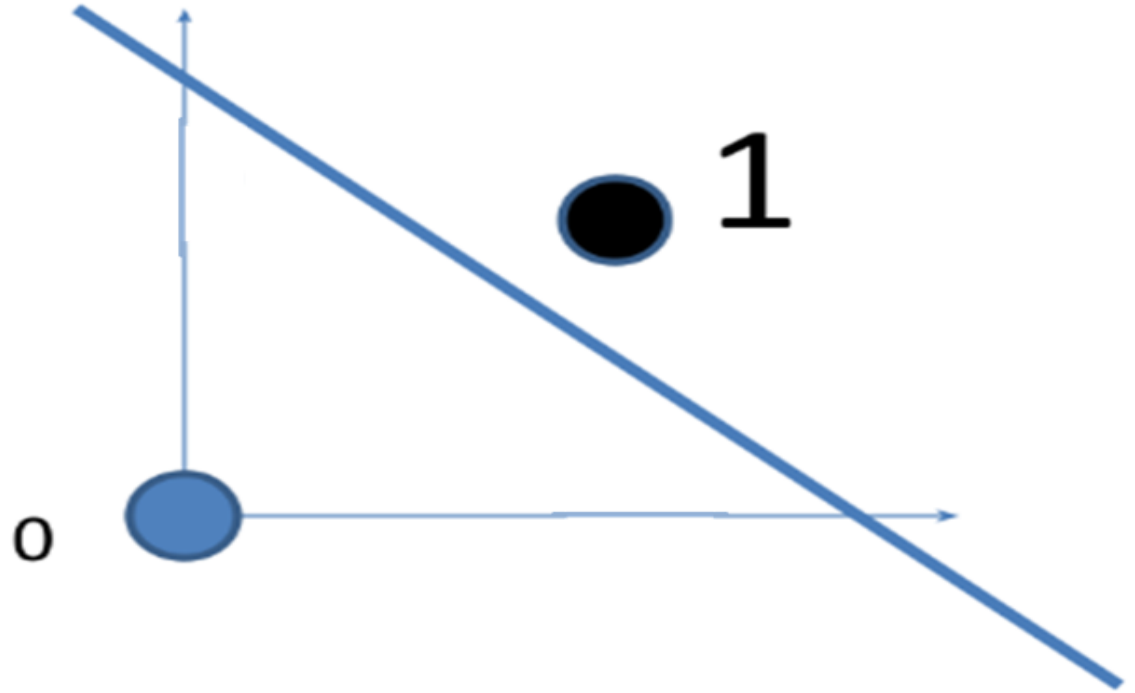
INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



Redes Neuronales

Función NOT es linealmente separable

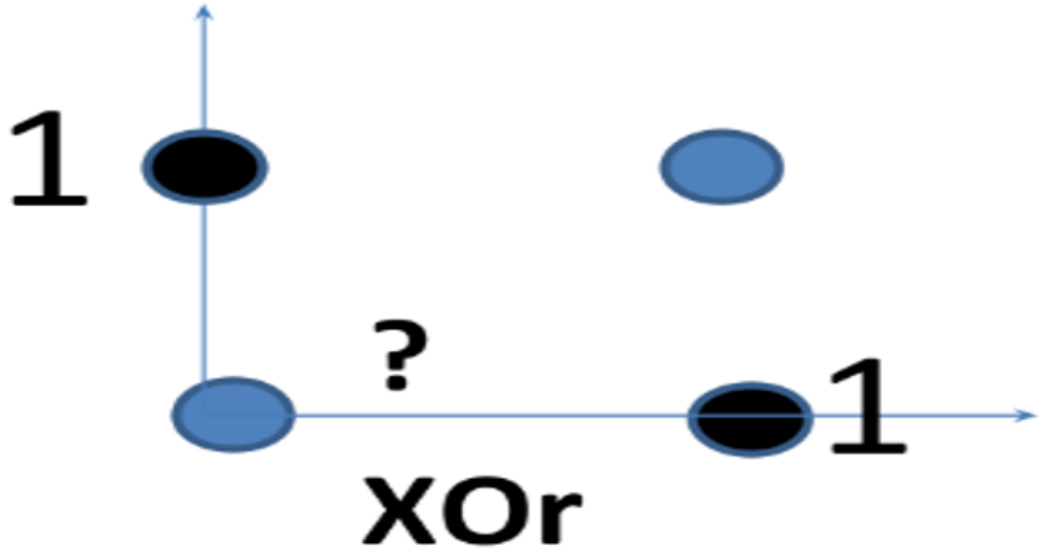
INPUT	OUTPUT
A	NOT A
0	1
1	0



Redes Neuronales

Función XOR **NO** es linealmente separable

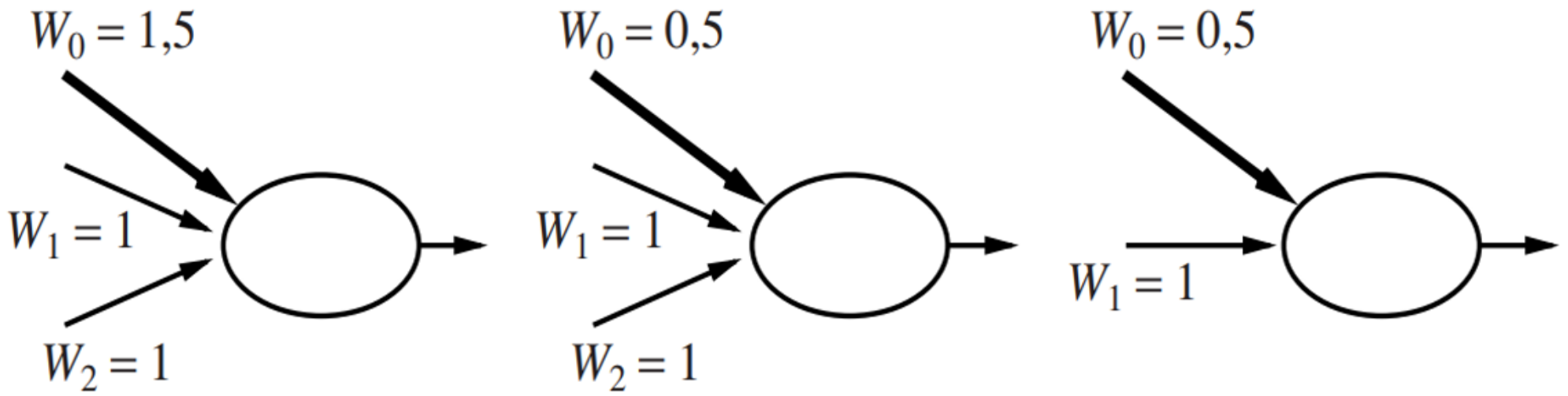
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0



El perceptrón solo no la puede aprender

Redes Neuronales

Con los pesos adecuados, el perceptrón aprende estas funciones



Redes Neuronales

¿Cómo se crean y actualizan los pesos?

$$v = \sum_{i=1}^{m} w_i x_i + b$$

Inicialmente, se pueden colocar de forma aleatoria.
Luego ir actualizando los pesos

Redes Neuronales

Actualización de Pesos, aprendizaje

$$W_j = W_j + e(t_i) * X_j$$

- W_j: Peso Nuevo
- W_j: Peso Actual
- e: Factor de Aprendizaje
- t_i: Valor que debe aprender
- X_j: Entrada

Profesor: Víctor Viera Balanta

Redes Neuronales

función APRENDIZAJE-PERCEPTRÓN(*ejemplos*, *red*) **devuelve** perceptrón como hipótesis

entrada: *ejemplos*, un conjunto de ejemplos, cada uno con entrada $\mathbf{x} = x_1, ..., x_n$ y salida y

red, un perceptrón con pesos $W_j, j = 0 \ldots n$, y función de activación g

repetir

para cada e **en** *ejemplos* **hacer**

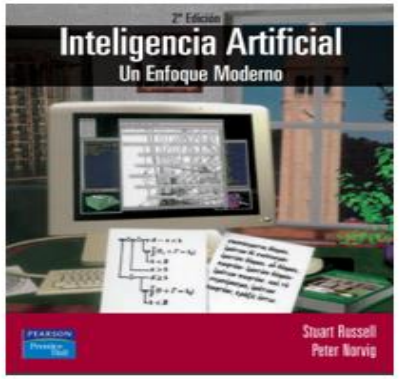
$in \leftarrow (\Sigma_{j=0}^{n} W_j x_j[e]$

$Err \leftarrow y[e] - g(in)$

$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j[e]$

hasta que se satisfaga algún criterio de parada

devolver HIPÓTESIS-RED-NEURONAS(*red*)

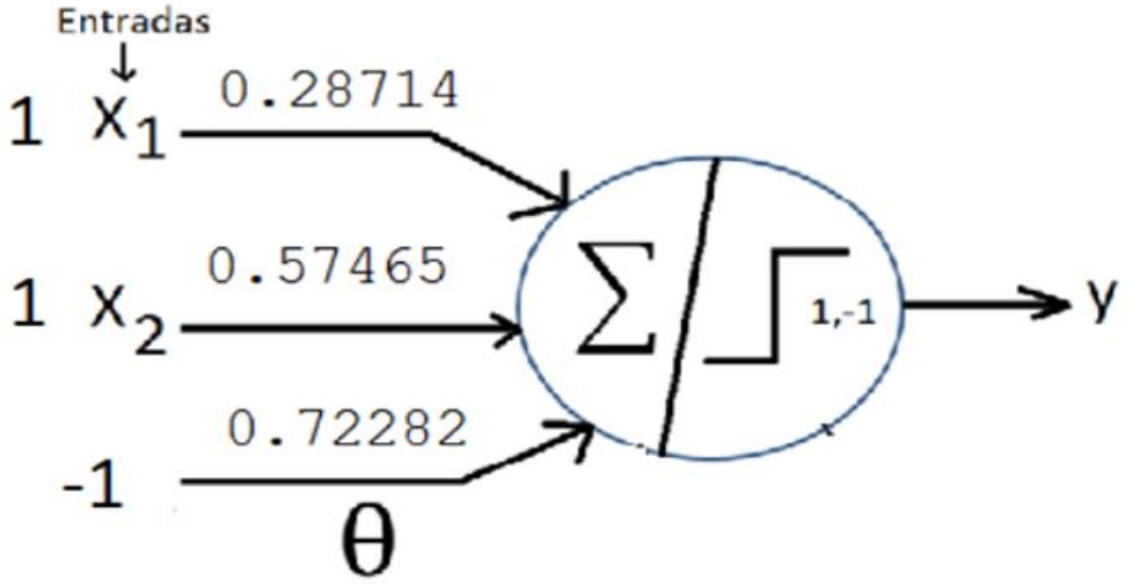


Redes Neuronales

Ejemplo: aprender la función or.

X_1	X_2	Y
1	1	1
1	0	1
0	1	1
0	0	0

X_1	X_2	Y
1	1	1
1	-1	1
-1	1	1
-1	-1	-1



Redes Neuronales

Entrada (1,1) Salida 1

$$Y_1 = 1*0{,}28714 + 1*0.57465 - 1*0.72282 = 0{,}13 \Rightarrow 1$$

Entrada (1,-1) Salida 1

$$Y2 = 1*0{,}28714 + -1*0.57465 - 1*0.72282 = -1{,}0103 \Rightarrow -1$$

Se recalculan los Pesos con :

$$W_j = W_j + e(t_i) * X_j$$

- W_j (izquierda): Peso Nuevo
- W_j (derecha): Peso Actual
- e: Factor de Aprendizaje
- t_i: Valor que debe aprender
- X_j: Entrada

Redes Neuronales

Se recalculan los Pesos con :

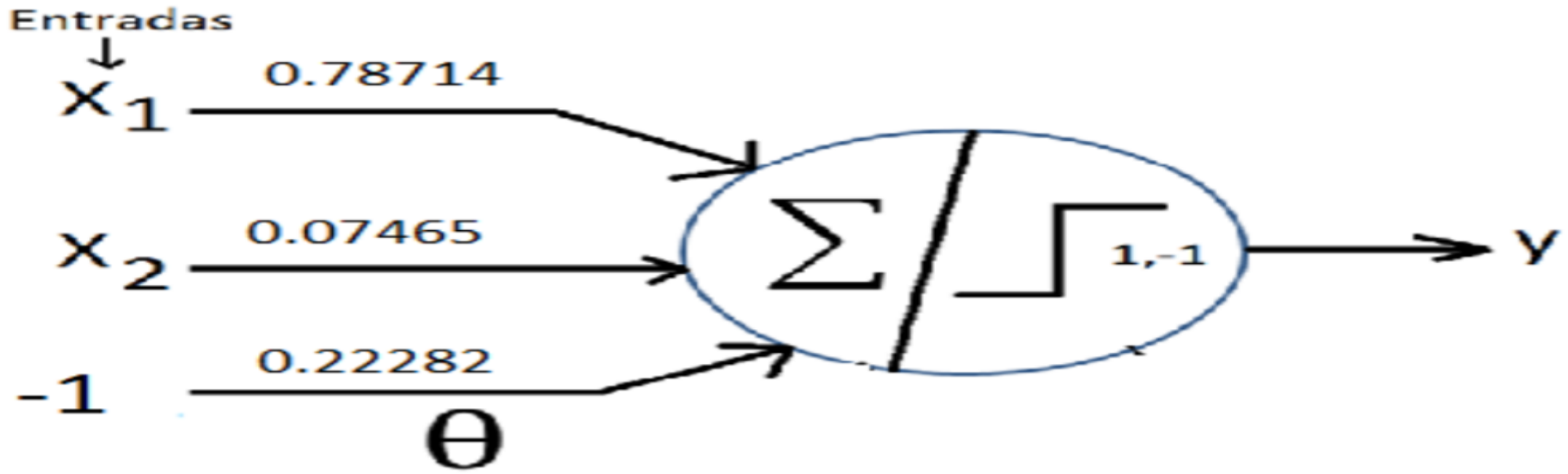
$$W_j = W_j + e(t_i) * X_j$$

- W_j: Peso Nuevo
- W_j: Peso Actual
- e: Factor de Aprendizaje
- t_i: Valor que debe aprender
- X_j: Entrada

peso	Factor	Por Aprender	Entrada Actual	Nuevo Peso
W_1 =0,28714	+ (0,5) *	(1)	(-1)	= -0,21286
W_2 = 0.57465	+ (0,5) *	(1)	(1)	= 1,07465
θ = 0.72282	+ (0,5) *	(1)	(-1)	= 0.22282

Redes Neuronales

Perceptrón con nuevos Pesos



Redes Neuronales

Programa en Java Entrenamiento, función (Or)

Entrada (1,1) Salida 1

Y_1 =1*0,78714+ 1*0.07465 - 1*0.2228= 0,6389 => 1

Entrada (1,-1) Salida 1

Y2 =1*0,78714+ -1*0.07465 - 1*0.2228=0,4896=>1

Entrada (-1,1) Salida 1

Y3 =-1*0,78714+ 1*0.07465 - 1*0.2228=-0,93529=>-1

Redes Neuronales

Se recalculan los Pesos con :

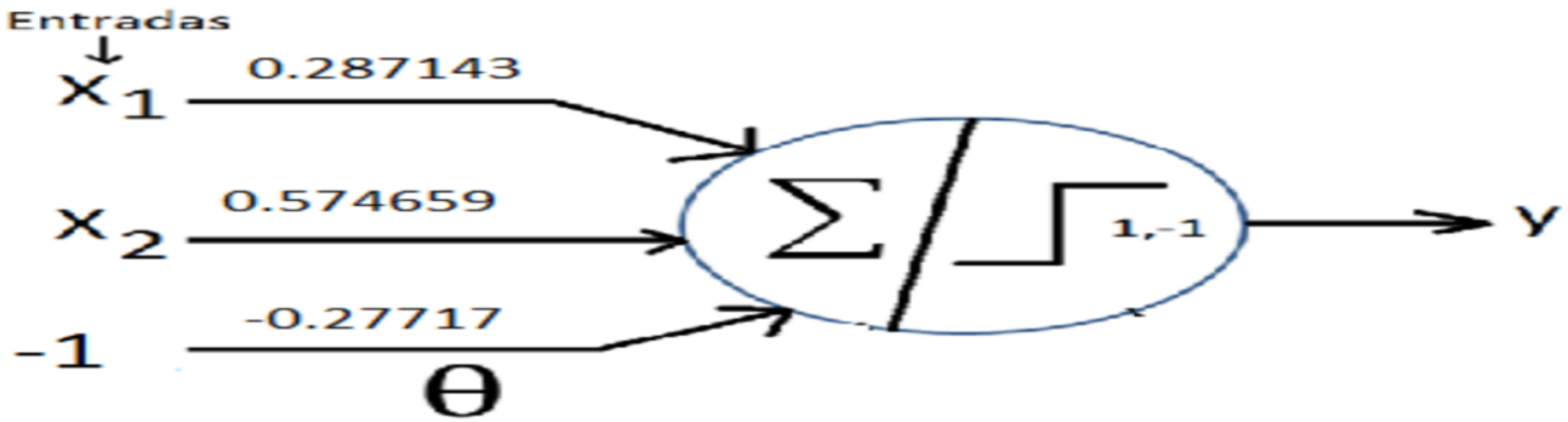
$$W_j = W_j + e(t_i) * X_j$$

- W_j: Peso Nuevo
- W_j: Peso Actual
- e: Factor de Aprendizaje
- t_i: Valor que debe aprender
- X_j: Entrada

peso	Factor	Por Aprender	Entrada Actual	Nuevo Peso
W_1 = 0,287143	+ (0,5) *	(1)	(1)	= 0.787143
W_2 = 0.574659	+ (0,5)*	(1)	(-1)	= 0.074659
θ = -0.27717	+ (0,5)*	(1)	(-1)	= -0.77717

Redes Neuronales

Perceptrón con nuevos Pesos



Redes Neuronales

Entrada (1,1) Salida 1

$Y_1 = 1*0.78714 + 1*0.07465 - 1*-0.77717 = 0,08462 \Rightarrow 1$

Entrada (1,-1) Salida 1

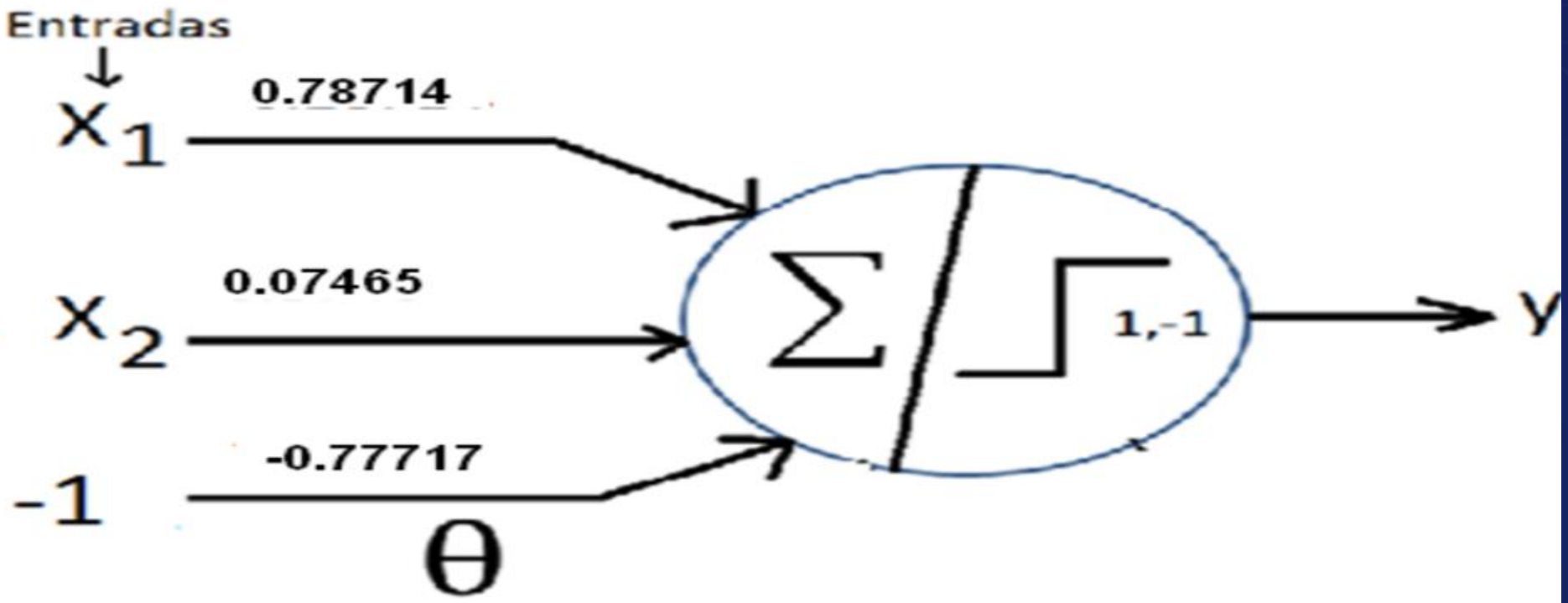
$Y2 = 1*0.78714 + -1*0.07465 - 1*0.77717 = -0,06468 \Rightarrow -1$

Se recalculan los Pesos con :

$$W_j = W_j + e(t_i) * X_j$$

- W_j (izquierda): Peso Nuevo
- W_j (derecha): Peso Actual
- e: Factor de Aprendizaje
- t_i: Valor que debe aprender
- X_j: Entrada

Redes Neuronales



Redes Neuronales

Programa en Java Entrenamiento, función (Or)

Entrada (1,1) Salida 1

Y_1 =1*0.78714+ 1*0.07465 - 1*-0.7717= 1,63349 => 1

Entrada (1,-1) Salida 1

Y2 =1*0.78714+ -1*0.07465 - 1* -0.7717 =1,4841=>1

Entrada (-1,1) Salida 1

Y2 =-1*0.78714+ 1*0.07465 - 1* -0.7717 =0,05921=>1

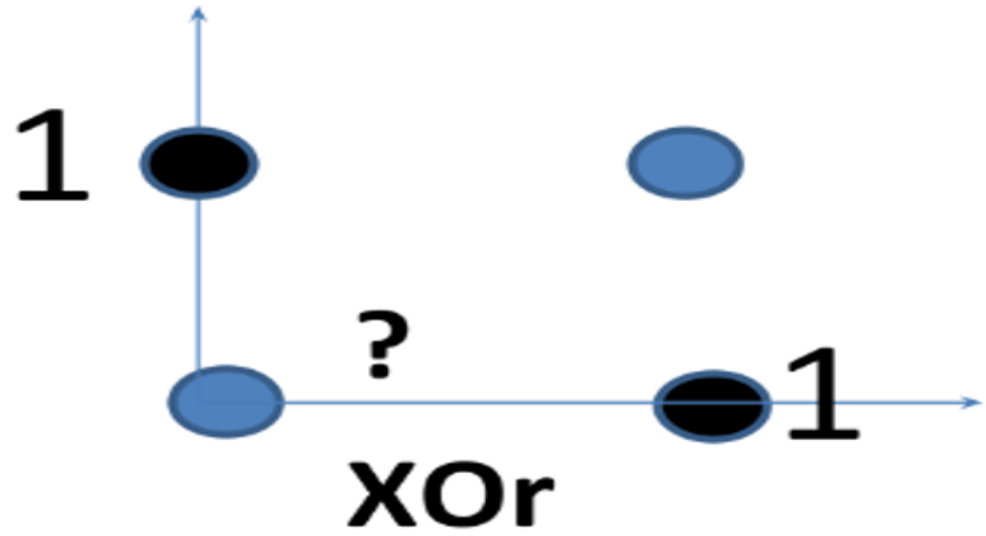
Entrada (-1,-1) Salida -1

Y2 =-1*0.78714+ -1*0.07465 - 1* -0.7717 =-0,09009=>-1

Redes Neuronales

Función XOR **NO** es linealmente separable

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0



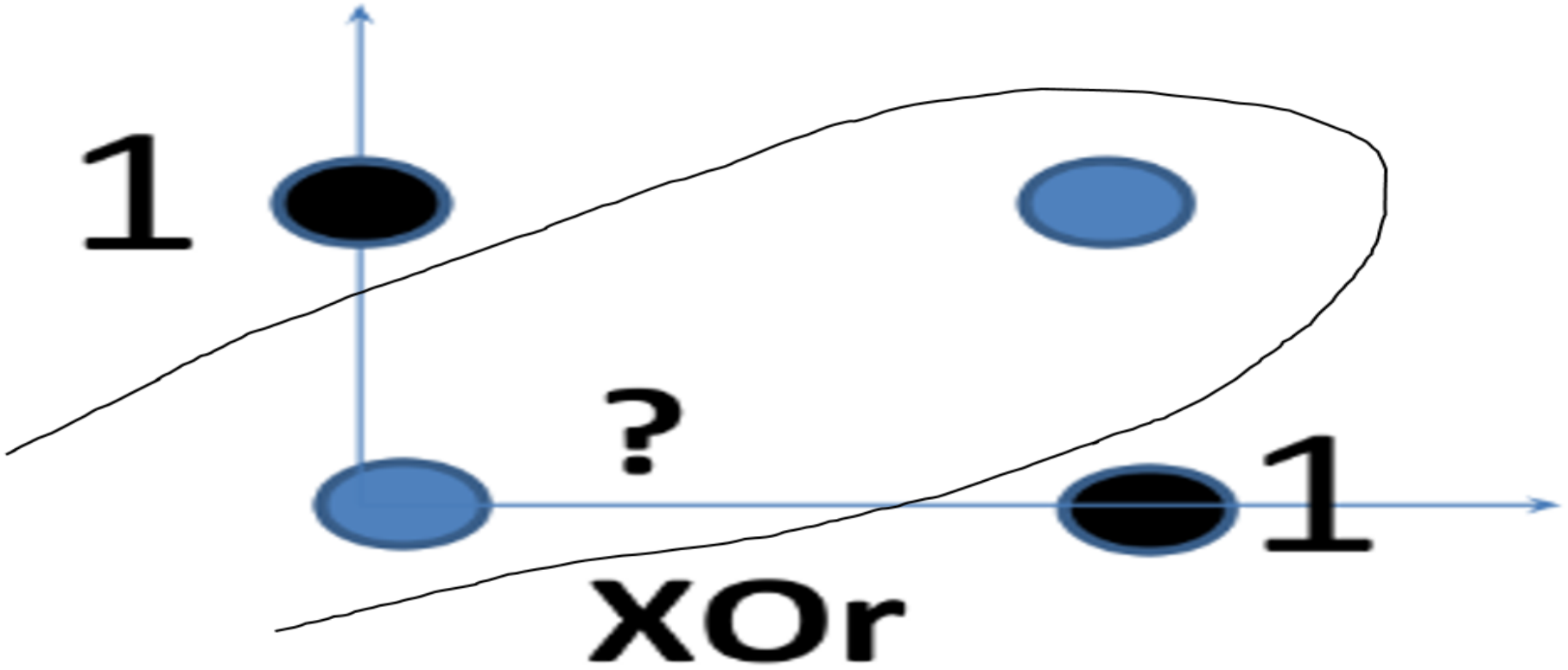
El perceptrón solo no la puede aprender

COLOMBIA POTENCIA DE LA VIDA

TALENTO TECH

TIC

Redes Neuronales



Profesor: Víctor Viera Balanta

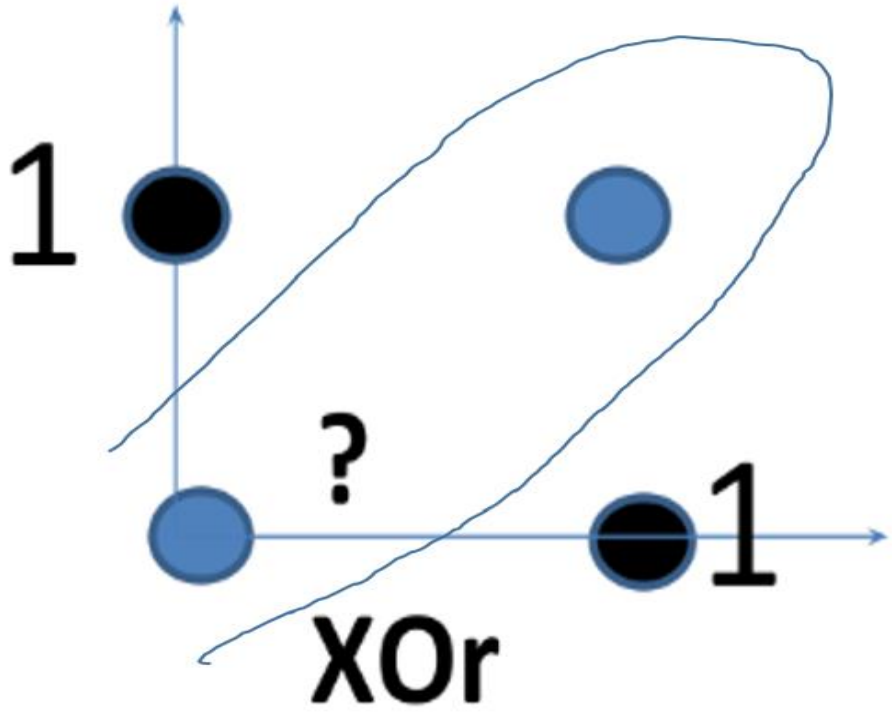
cymetria | tecnalia colombia

COLOMBIA POTENCIA DE LA VIDA

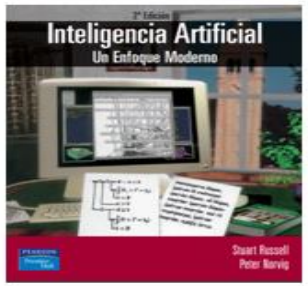
TALENTO TECH

TIC

Redes Neuronales



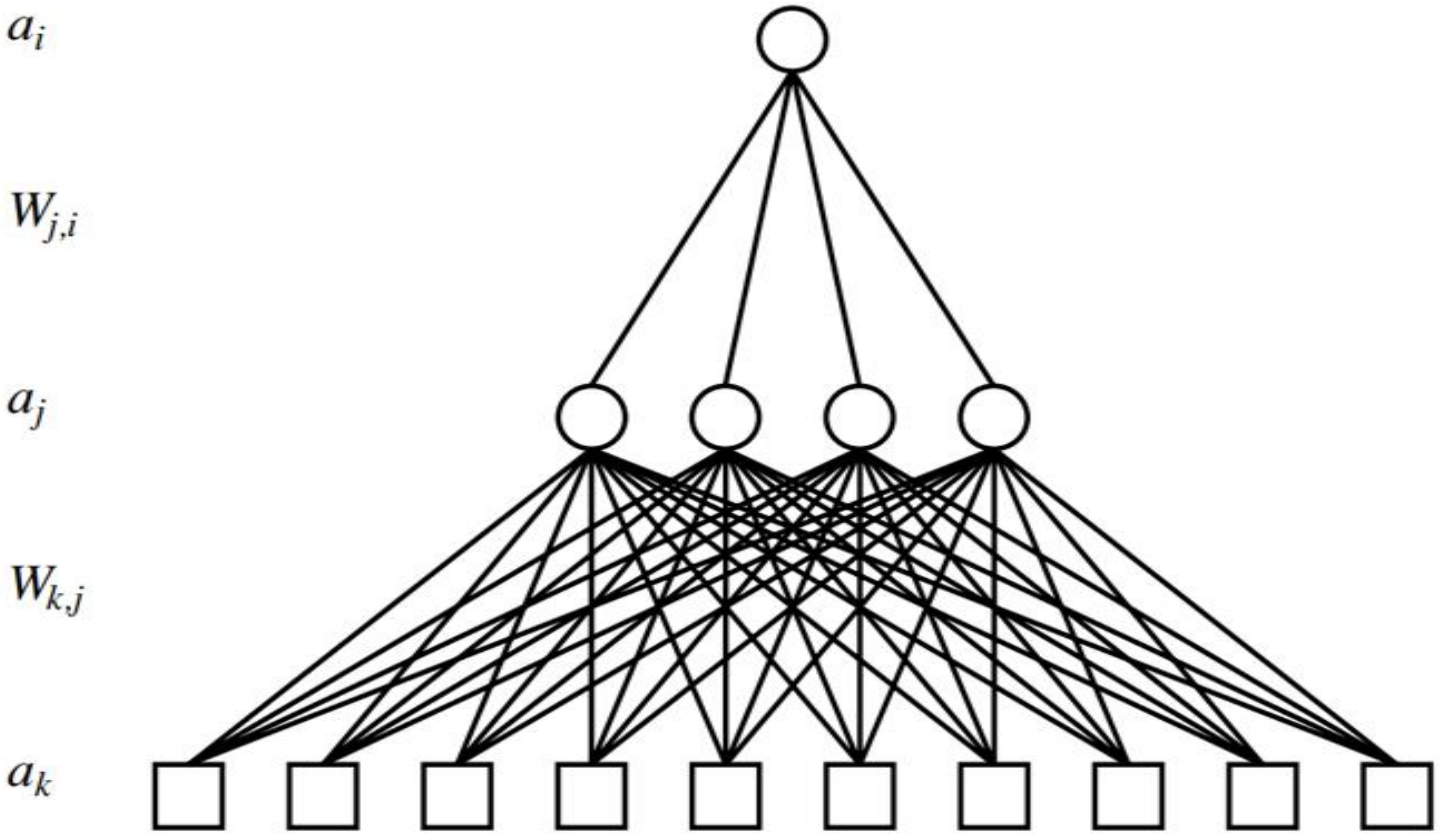
Perceptrón Multicapa

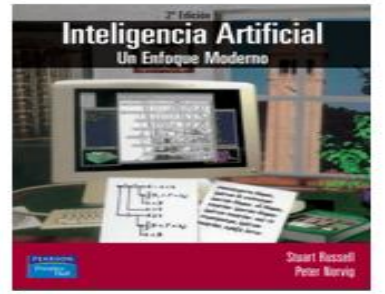


Profesor: Víctor Viera Balanta

cymetria | tecnal:a colombia

Redes Neuronales



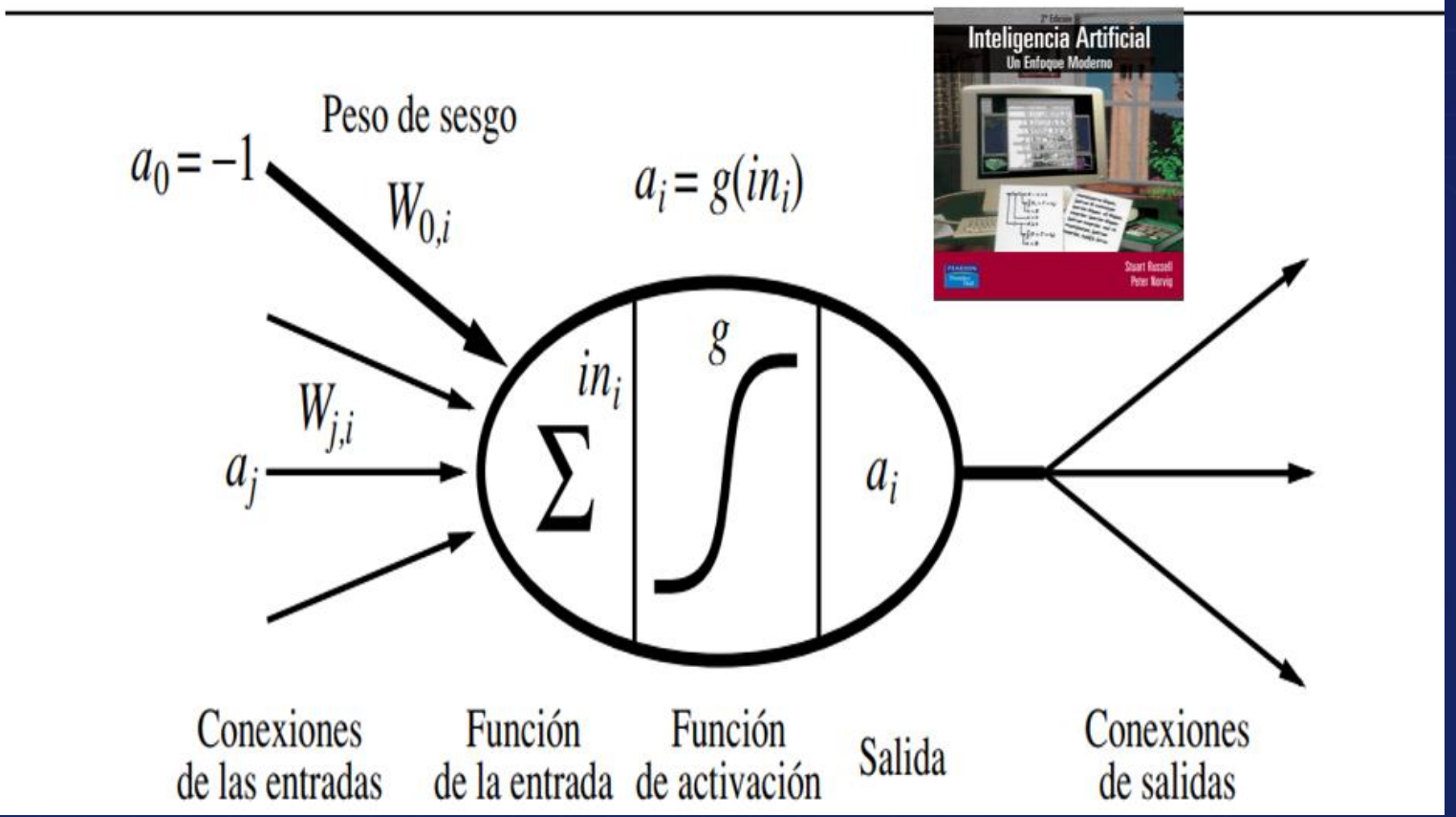


Modelo

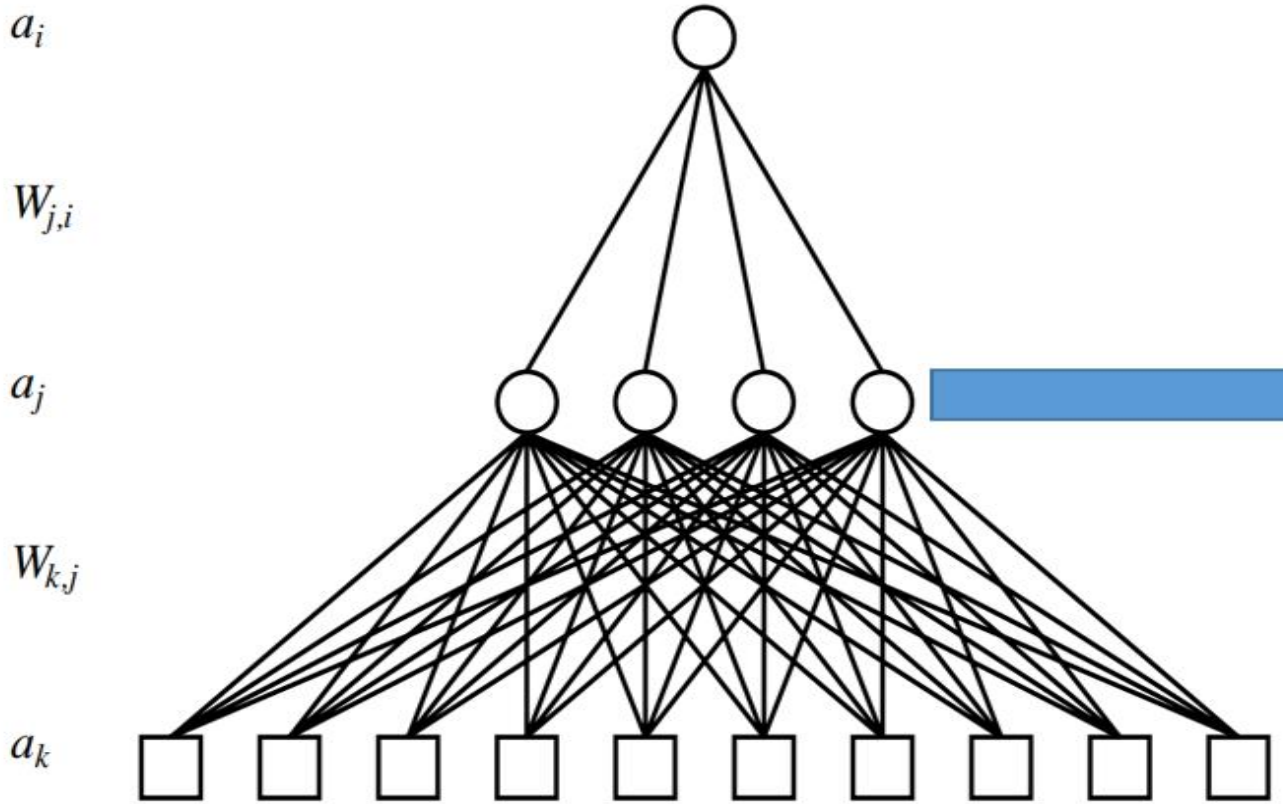
Profesor: Víctor Viera Balanta

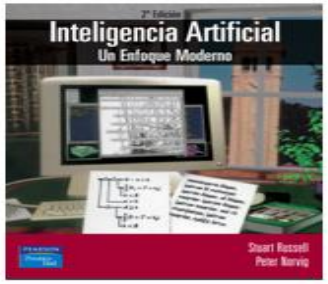
Redes Neuronales

$$a_i = g(in_i)$$



Redes Neuronales



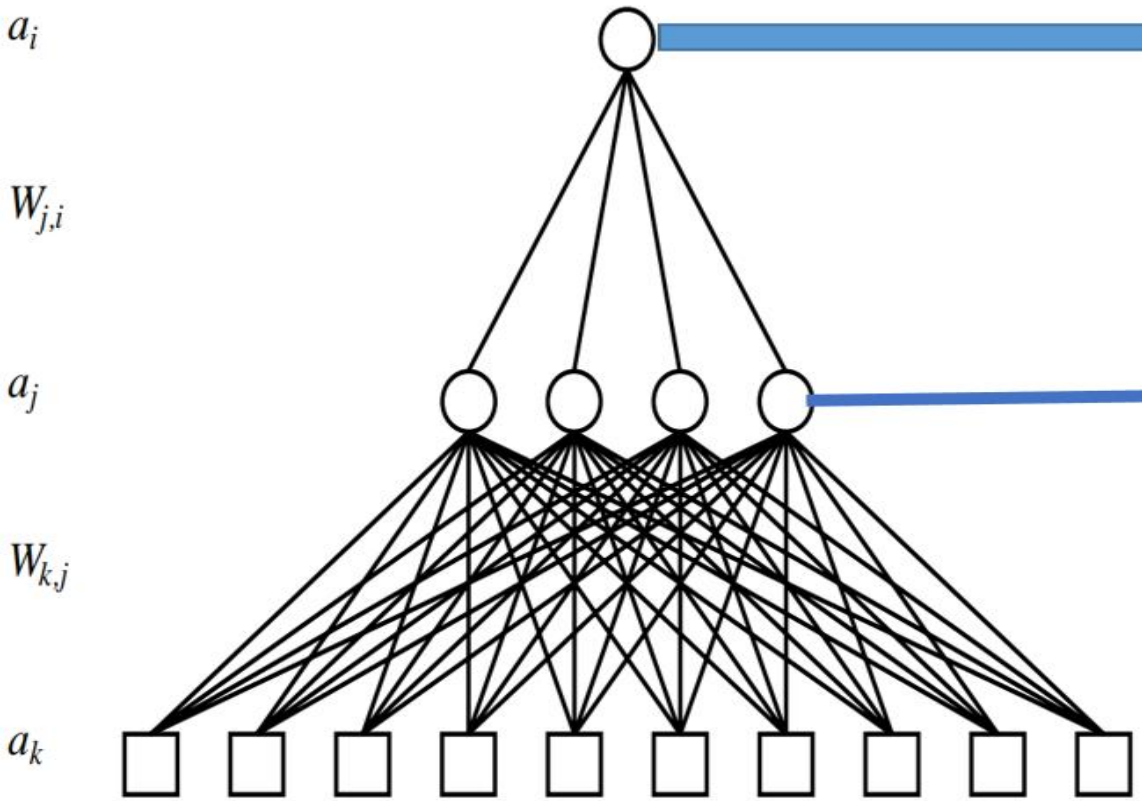


$$a_j = g(in_i)$$

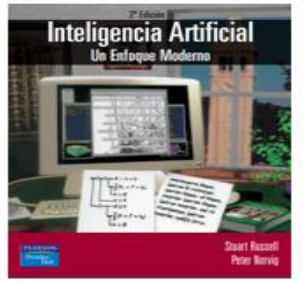
$$\sum_{i=1}^{m} w_i x_i$$

Profesor: Víctor Viera Balanta

Redes Neuronales



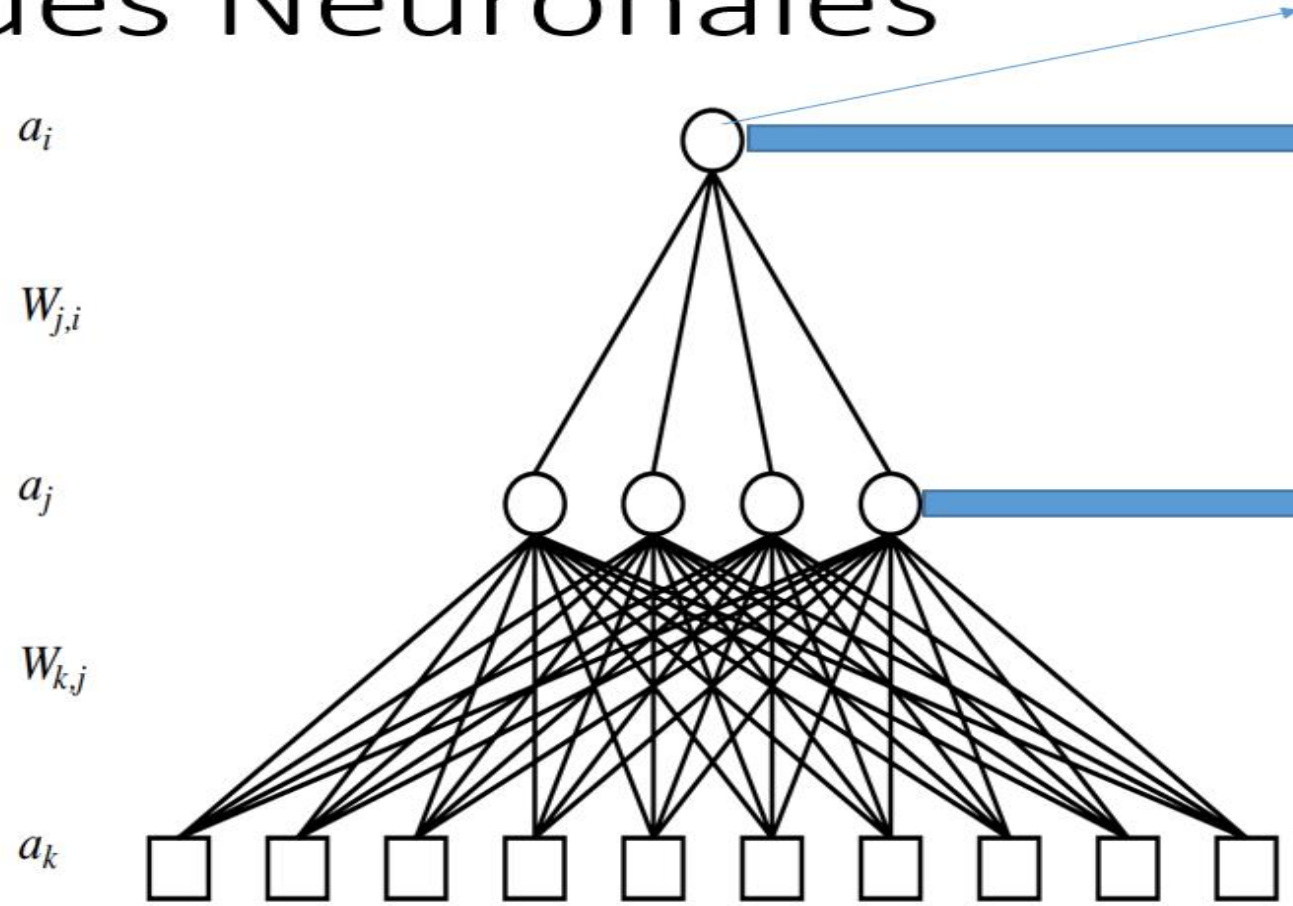
$$in_i = \sum_{j=0}^{n} W_{j,i} a_j$$



$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i} a_j \right)$$

Profesor: Víctor Viera Balanta

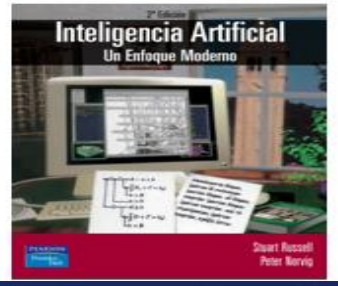
Redes Neuronales



Valor que debe Aprender

$$a_i = g(in_i)$$

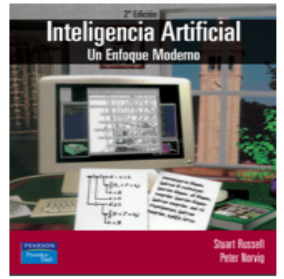
$$a_j = g(in_j)$$

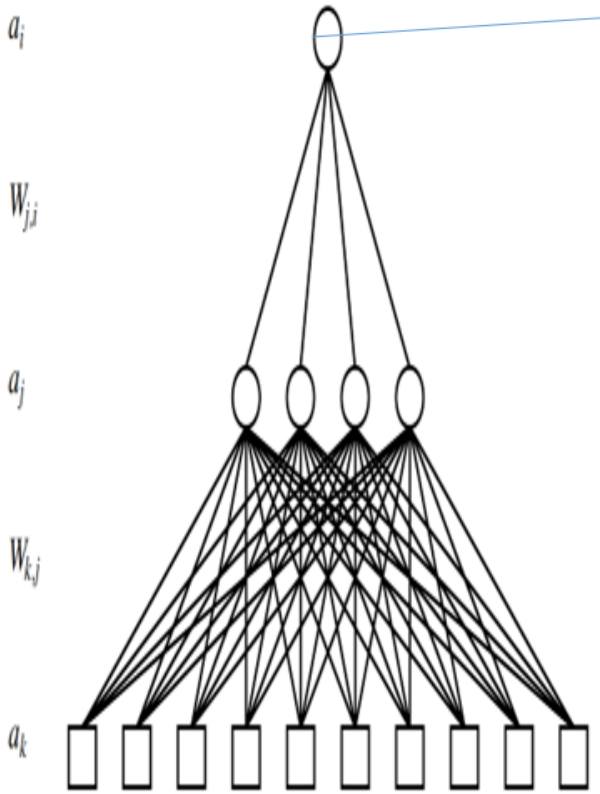


Profesor: Víctor Viera Balanta

Redes Neuronales

Valor que debe Aprender





ti Valor que debe Aprender

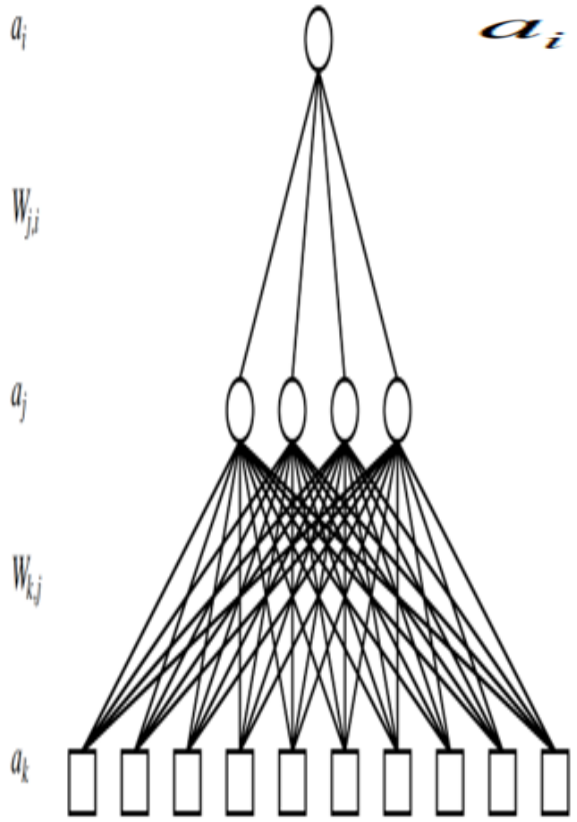
La medida «clásica» del error es la suma de los errores cuadrados

$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$

Valor que debe Aprender

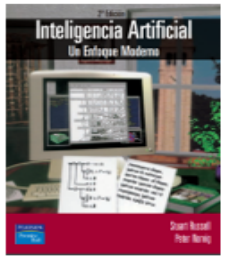
Cálculo de la Red

Redes Neuronales



$$a_i = g(in_i)$$

Valor que debe Aprender

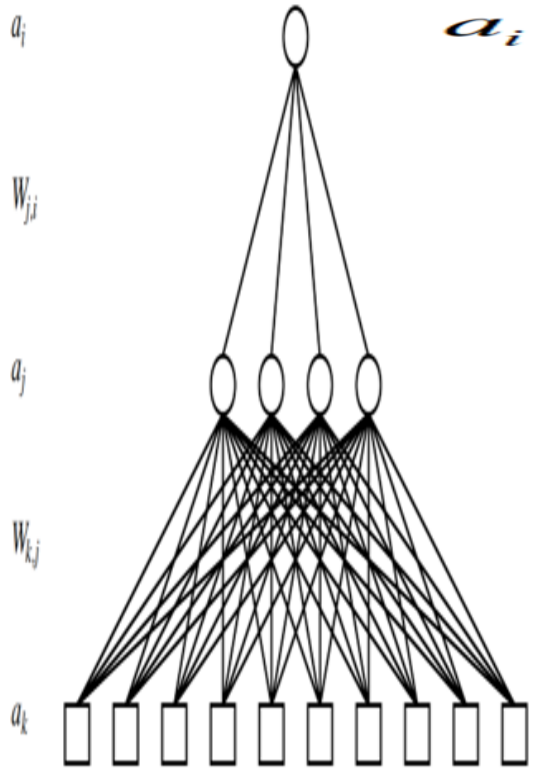


$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$

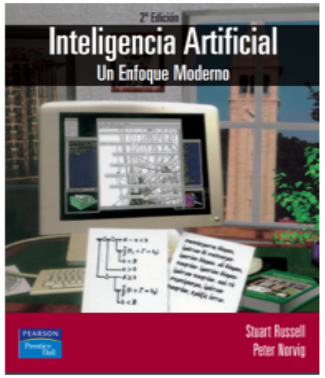
$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - a_i)^2$$

Redes Neuronales

Valor que debe Aprender



$$a_i = g(in_i)$$



$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$

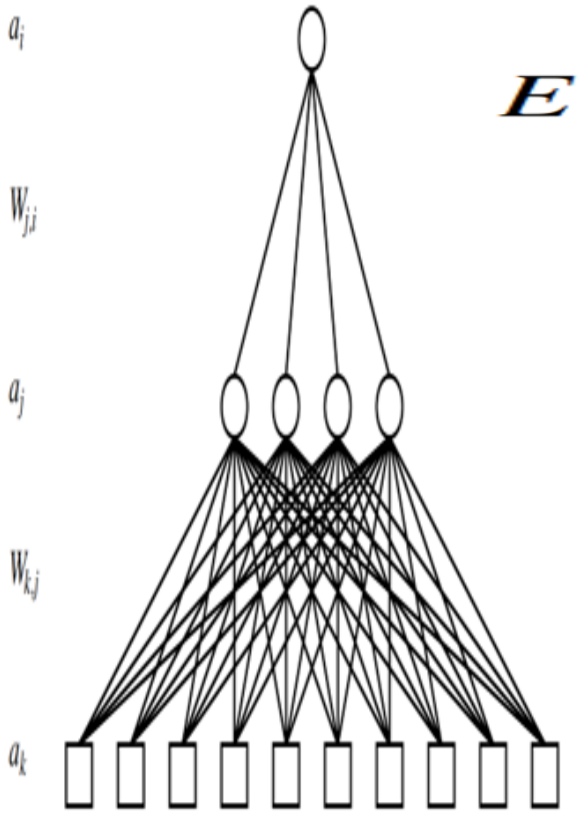
$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - a_i)^2$$

Se involucran los pesos

$$g\left(\sum_{j=0}^{n} W_{j,i} a_j\right)$$

Redes Neuronales

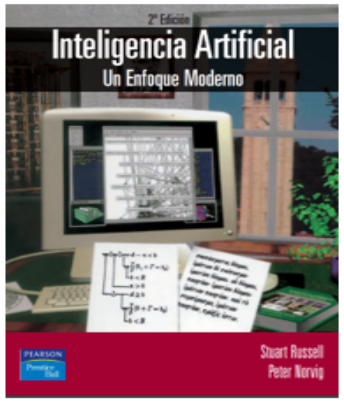
Valor que debe Aprender



$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$

$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - a_i)^2$$

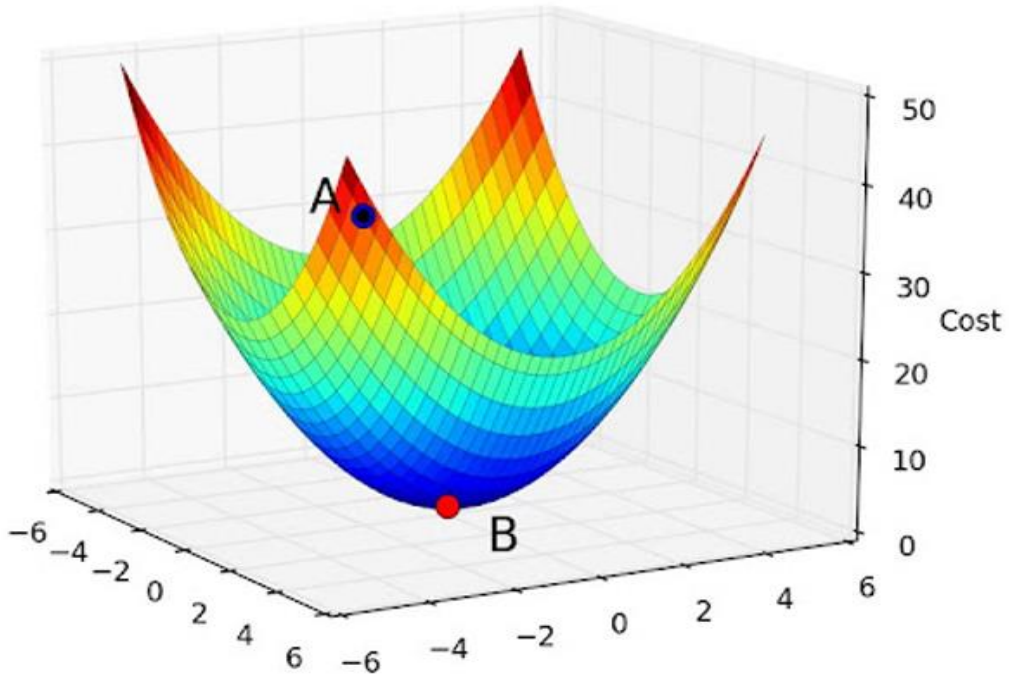
$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - g\left(\sum W_{i,j}\, a_i\right))^2$$



Profesor: Víctor Viera Balanta

Redes Neuronales

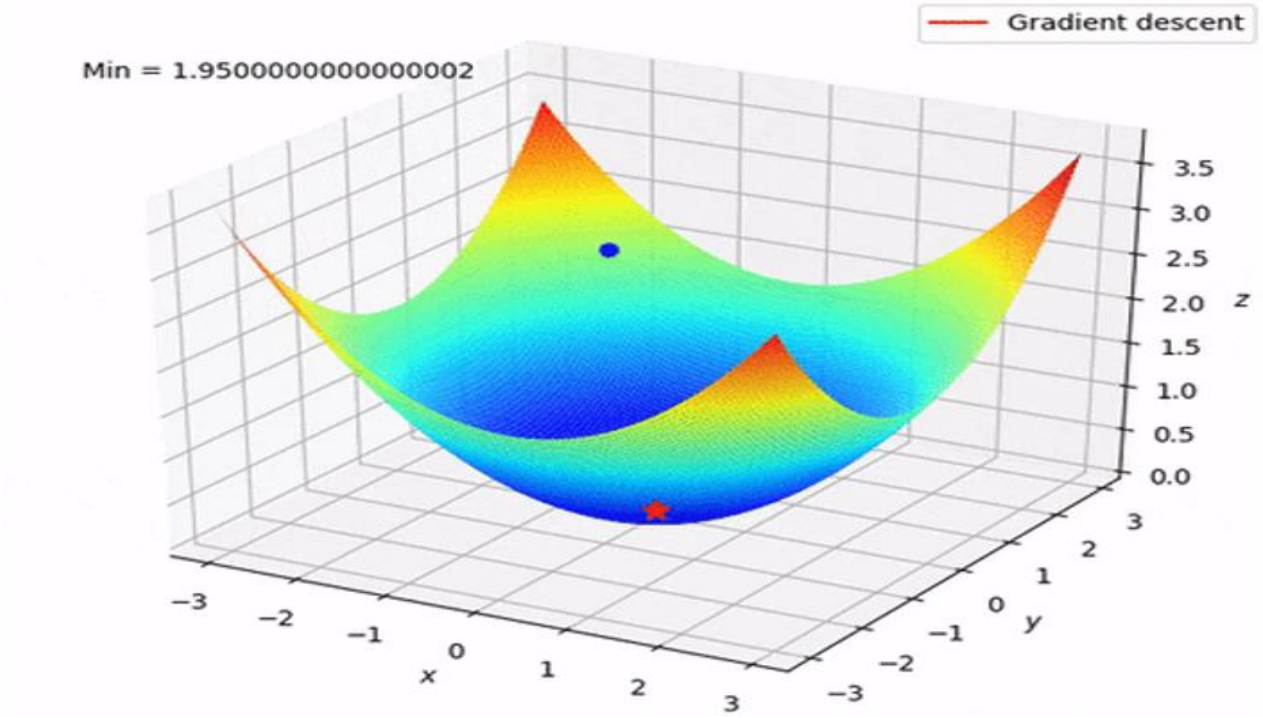
Se usa el método del **Descenso del Gradiente** para reducir el error cuadrado calculando la derivada parcial de E con respecto a cada peso.



https://www.ellaberintodefalken.com/2019/03/regresion-lineal-descenso-de-gradiente.html

Redes Neuronales

Se usa el método del **Descenso del Gradiente** para reducir el error cuadrado calculando la derivada parcial de E con respecto a cada peso.



https://www.ellaberintodefalken.com/2019/03/regresion-lineal-descenso-de-gradiente.html

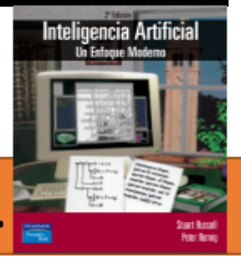
Redes Neuronales

para reducir el error cuadrado.

$$E = \frac{1}{2} Err^2 \equiv \frac{1}{2}(y - g(\sum W_{i,j}\, a_i))^2$$

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j}$$

Los pesos son lo únicos que se pueden cambiar, por eso la derivada parcial, respecto a ellos.



SE calculando la derivada parcial de E con respecto a cada peso.

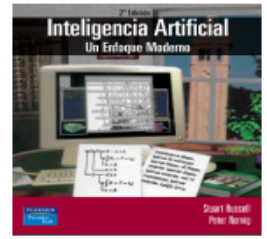
Profesor: Víctor Viera Balanta

Redes Neuronales

Derivada interna

$$\frac{\partial\left(\frac{1}{2}Err^2\right)}{\partial(w_1)} \Rightarrow \frac{\frac{1}{2}\cdot 2 \times Err \times \partial(Err)}{\partial(w_1)}$$

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j}$$



Redes Neuronales

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j}$$

$$\frac{1}{2}(y - g\left(\sum W_{i,j}\, a_i\right))^2$$

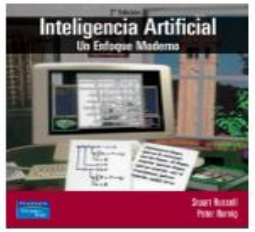
$$= Err \times \frac{\partial}{\partial W_j}\left(y - g\left(\sum_{j=0}^{n} W_j x_j\right)\right)$$

Redes Neuronales

Se deriva la función, la derivada interna. Si se deriva parcialmente respecto a W, se elimina el resto

$$= Err \times \frac{\partial}{\partial W_j}\left(y - g\left(\sum_{j=0}^{n} W_j x_j\right)\right)$$

$$= -Err \times g'(in) \times x_j$$



donde g' es la derivada de la función de activación

Redes Neuronales

Ahora, para hacer el descenso de gradiente, se actualizan los pesos

$$= -Err \times g'(in) \times x_j$$

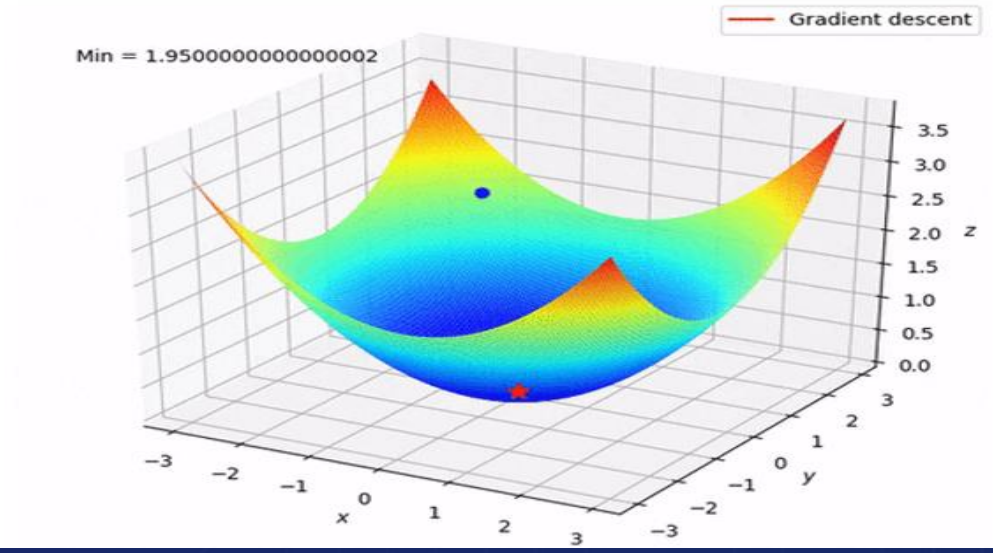
$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

donde α es la **tasa de aprendizaje**

Factor de Aprendizaje

Redes Neuronales

Intuitivamente, esto tiene mucho sentido. Si el error $Err = y - h_{\mathbf{W}}(\mathbf{x})$ es positivo, la salida de la red es demasiado pequeña y por ello los pesos se *incrementan* para las entradas positivas y se *decrementan* para las entradas negativas. Cuando el error es negativo[10], ocurre lo contrario.



repetir

para cada e **en** *ejemplos* **hacer**

$$in \leftarrow (\Sigma_{j=0}^{n} W_j x_j[e]$$

$$Err \leftarrow y[e] - g(in)$$

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j[e]$$

hasta que se satisfaga algún criterio de parada

devolver HIPÓTESIS-RED-NEURONAS(*red*)

Cómo Aprende el Perceptrón?

https://youtu.be/-cJTJqR8nTQ?si=eTOu4xhi_t0nBUrC

Profesor: Víctor Viera Balanta

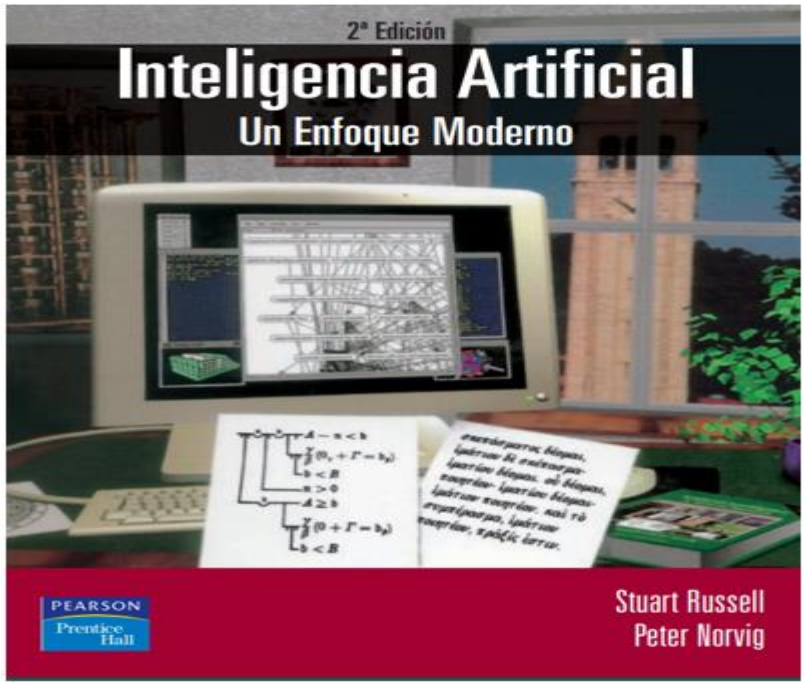
COLOMBIA POTENCIA DE LA VIDA

TALENTO TECH

TIC

Redes Neuronales

Para profundizar en los cálculos . Página 848-850

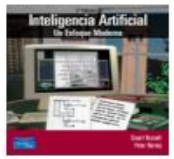


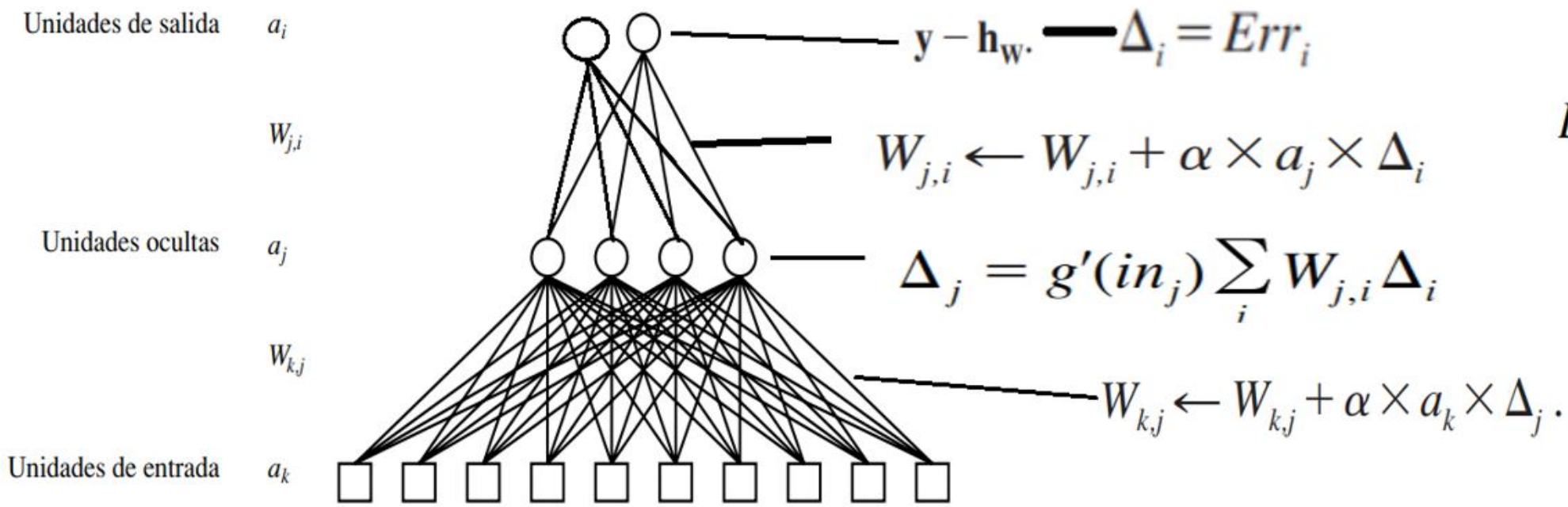
Profesor: Víctor Viera Balanta

cymetria | tecnal:a colombia

Redes Neuronales

Para profundizar en los cálculos . Página 848-850

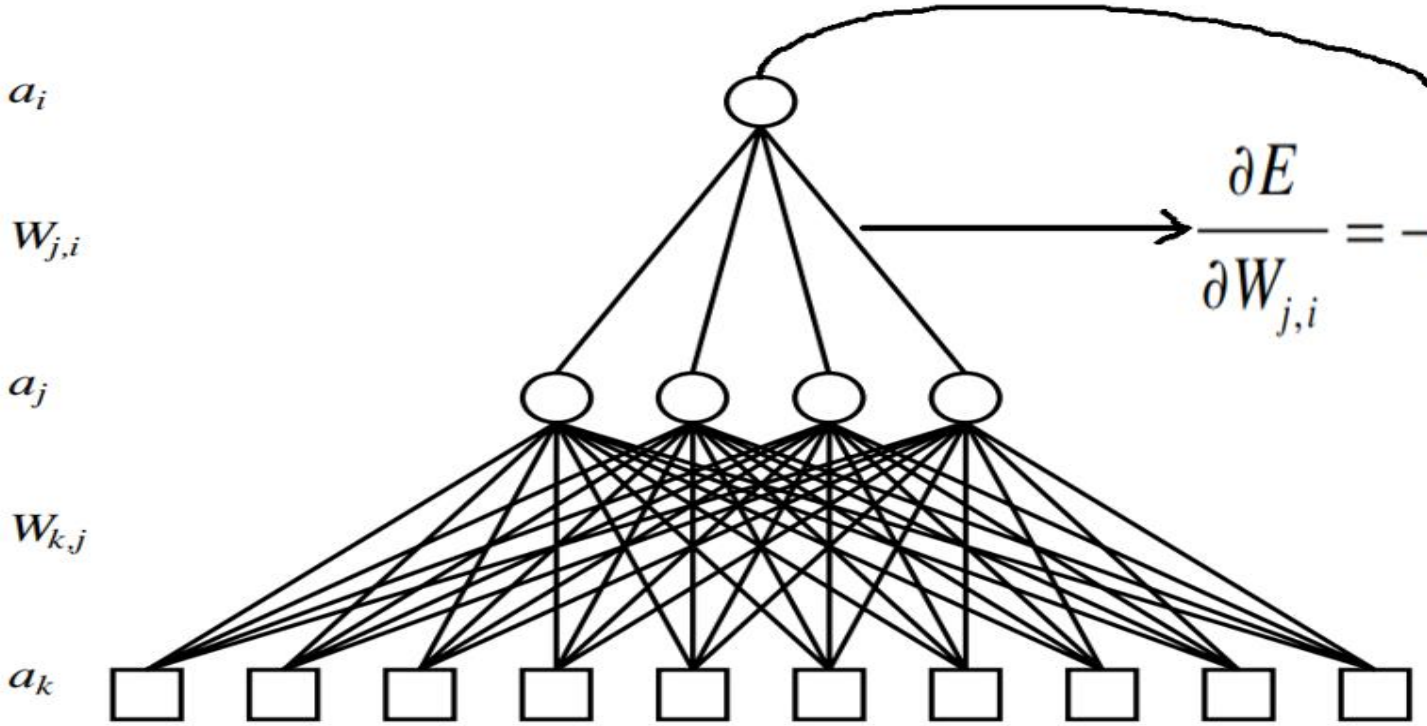




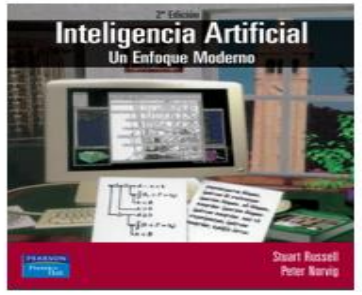
$$E = \frac{1}{2} \sum_i (y_i - a_i)^2$$

Redes Neuronales

Para profundizar en los cálculos . Página 848-850



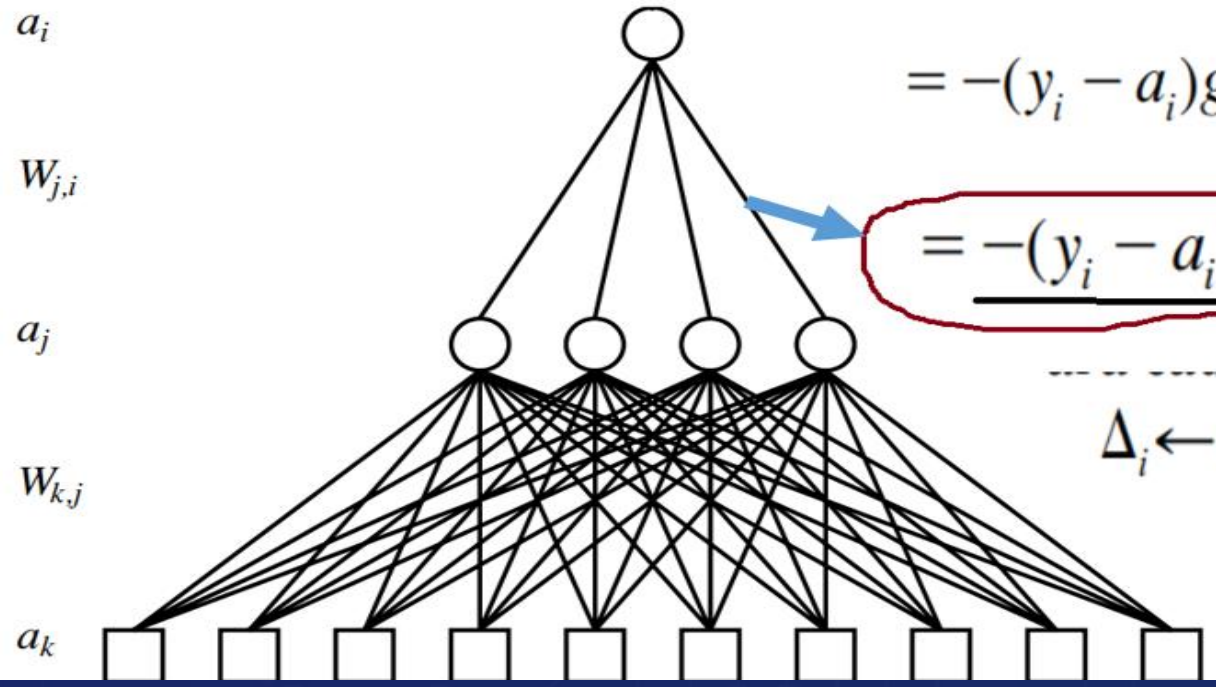
$$\frac{\partial E}{\partial W_{j,i}} = -(y_i - a_i)\frac{\partial a_i}{\partial W_{j,i}} = -(y_i - a_i)\frac{\partial g(in_i)}{\partial W_{j,i}}$$



Profesor: Víctor Viera Balanta

Redes Neuronales

Para profundizar en los cálculos . Página 848-850



derivada interna

$$= -(y_i - a_i)g'(in_i)\frac{\partial in_i}{\partial W_{j,i}} = -(y_i - a_i)g'(in_i)\frac{\partial}{\partial W_{j,i}}\left(\sum_j W_{j,i}a_j\right)$$

$$= -(y_i - a_i)g'(in_i)a_j = -a_j\Delta_i$$

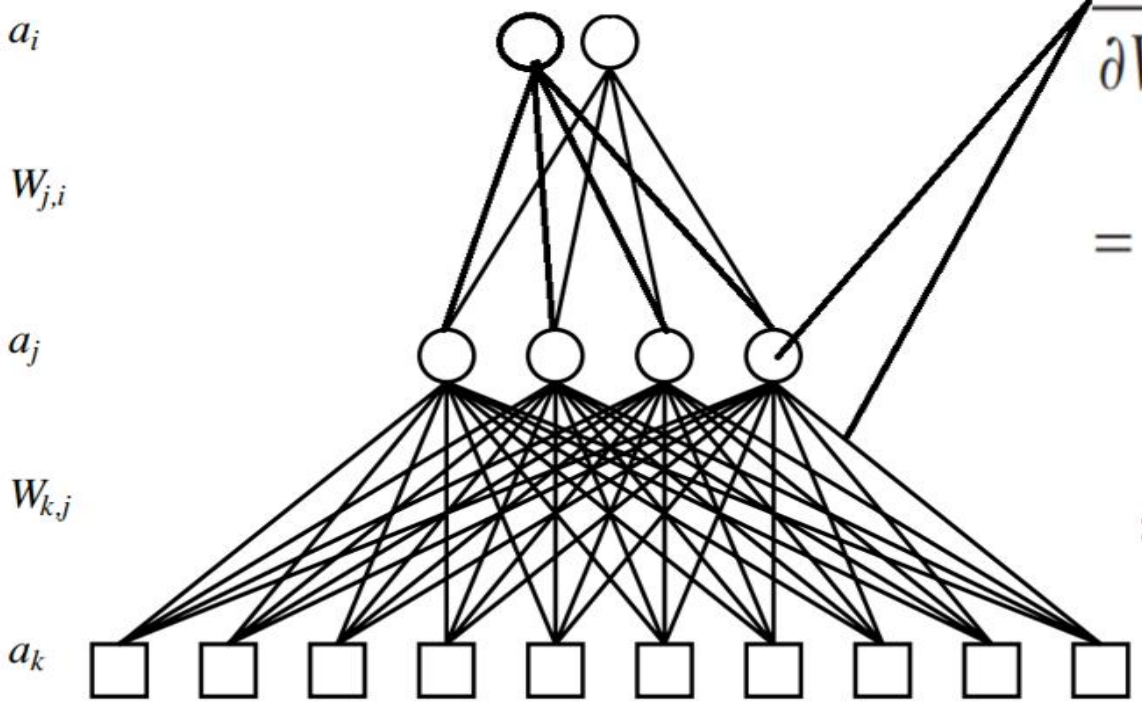
$$\Delta_i \leftarrow g'(in_i) \times (y_i[e] - a_i)$$

Derivada Interna

Profesor: Víctor Viera Balanta

Redes Neuronales

Para profundizar en los cálculos . Página 848-850



$$\frac{\partial E}{\partial W_{k,j}} = -\sum_i (y_i - a_i)\frac{\partial a_i}{\partial W_{k,j}} = -\sum_i (y_i - a_i)\frac{\partial g(in_i)}{\partial W_{k,j}}$$

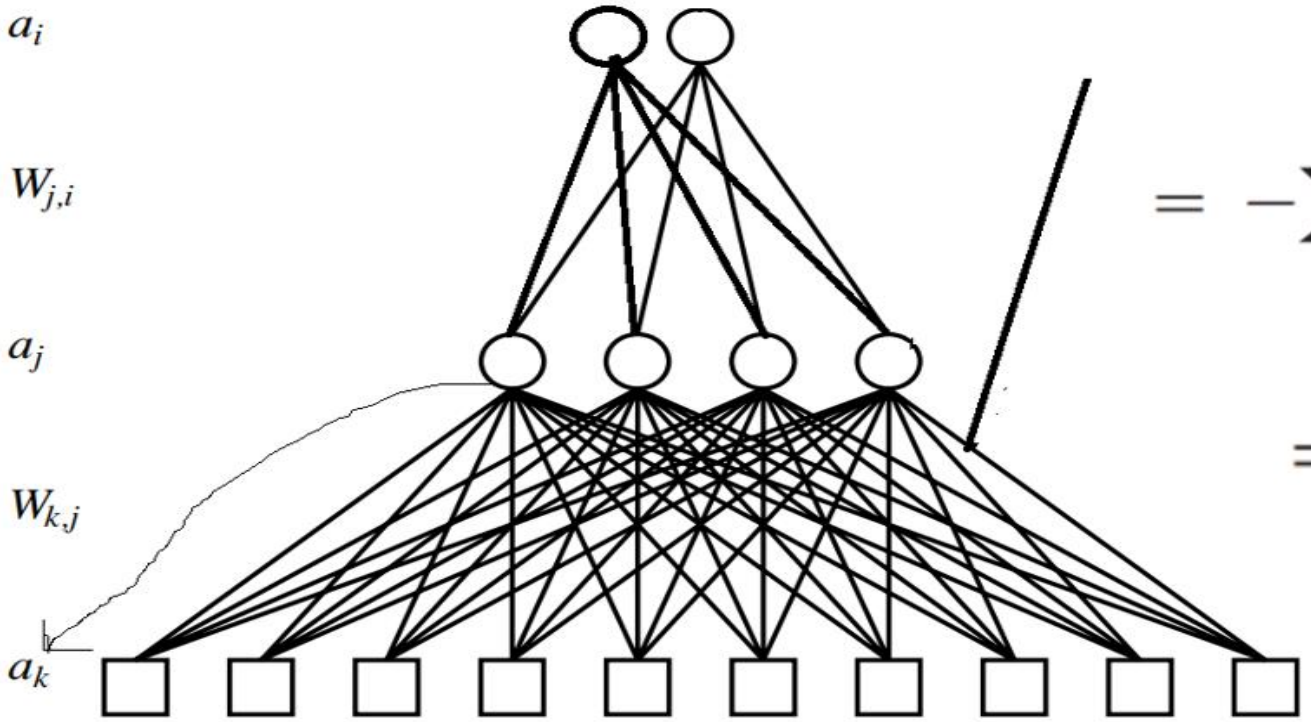
$$= -\sum_i (y_i - a_i) g'(in_i)\frac{\partial in_i}{\partial W_{k,j}} = -\sum_i \Delta_i \frac{\partial}{\partial W_{k,j}}\left(\sum_j W_{j,i} a_j\right)$$

$$= -\sum_i \Delta_i W_{j,i} \frac{\partial a_j}{\partial W_{k,j}} = -\sum_i \Delta_i W_{j,i} \frac{\partial g(in_i)}{\partial W_{k,j}}$$

Profesor: Víctor Viera Balanta

Redes Neuronales

Para profundizar en los cálculos . Página 848-850



$$= -\sum_i \Delta_i W_{j,i}\, g'(in_j)\, \frac{\partial in_j}{\partial W_{k,j}}$$

$$= -\sum_i \Delta_i W_{j,i}\, g'(in_j) \frac{\partial}{\partial W_{k,j}} \left(\sum_k W_{k,j} a_k \right)$$

$$= -\sum_i \Delta_i W_{j,i}\, g'(in_j) a_k = -a_k \Delta_j$$

Profesor: Víctor Viera Balanta

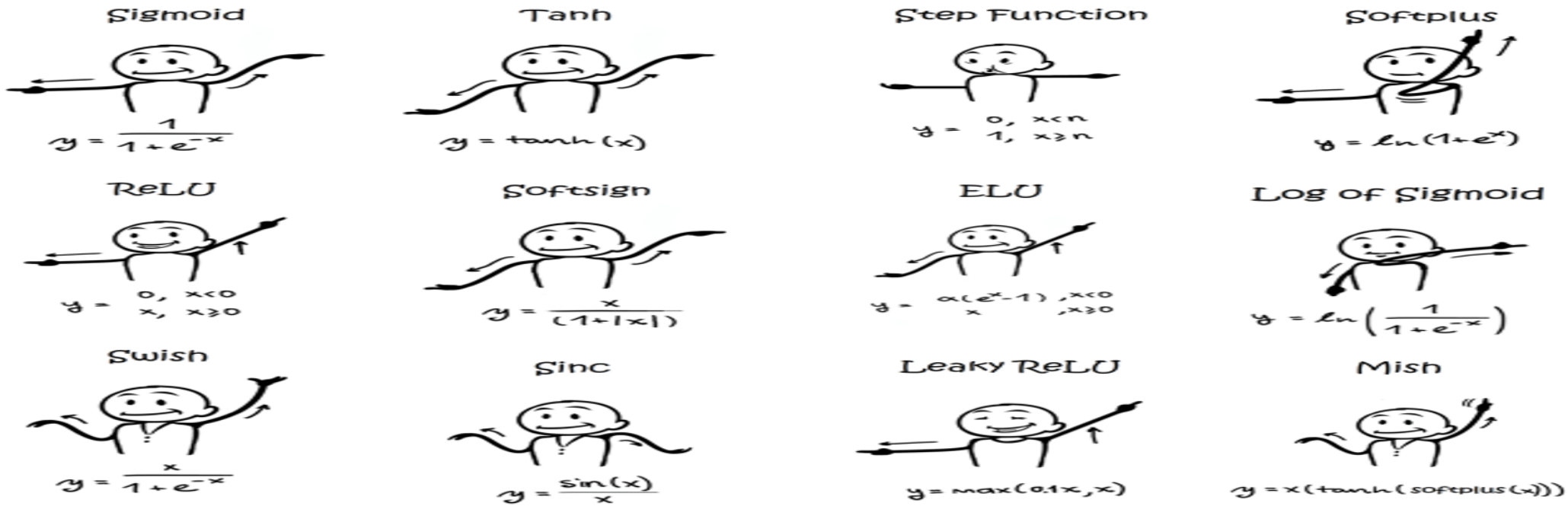
Funciones de Activación

Profesor: Víctor Viera Balanta

UT TALENTOTECH

Redes Neuronales

Funciones de Activación

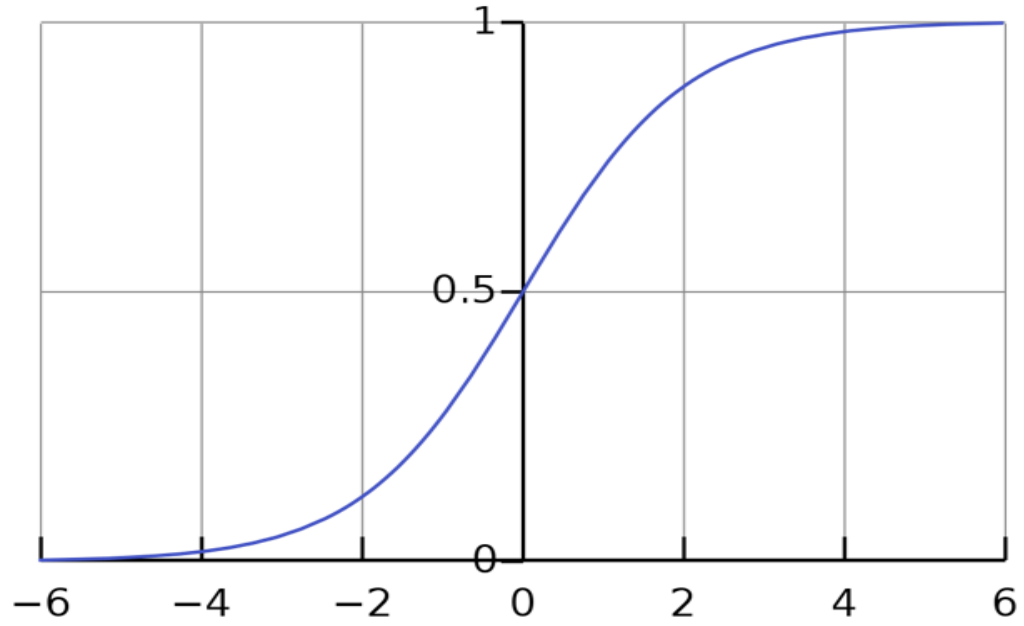


https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow/

JAHAZIEL PONCE

Redes Neuronales

Explicación algorítmica



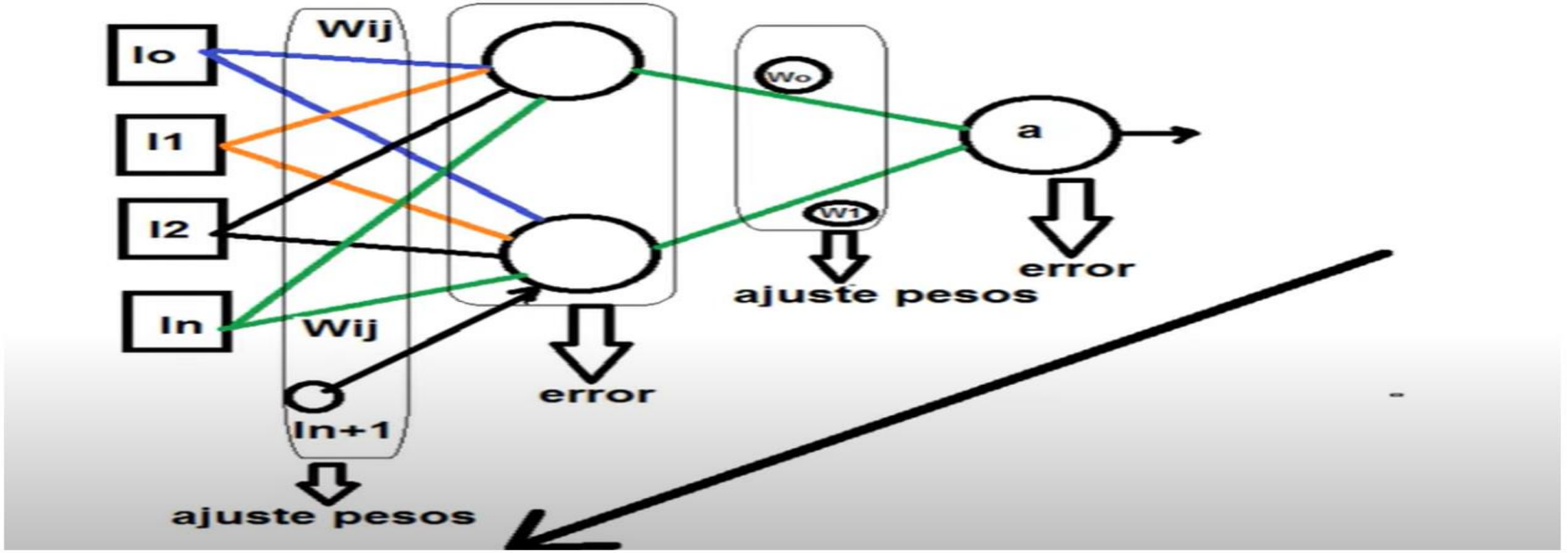
Función Sigmoide

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

Derivada

$$\sigma'(x) = \sigma(x).(1 - \sigma(x))$$

Redes Neuronales



COLOMBIA POTENCIA DE LA VIDA

TALENTO TECH

TIC

Redes Neuronales

Back Propagation

Paul Werbos, International Joint Conference on Neural Networks (IJCNN), 8. July 1991, Seattle.

Backpropagation Through Time: What It Does and How to Do It 1974 **Tesis doctoral**

PAUL J. WERBOS

Backpropagation is now the most widely used tool in the field of artificial neural networks. At the core of backpropagation is a method for calculating derivatives exactly and efficiently in any large system made up of elementary subsystems or calculations which are represented by known, differentiable functions; thus, backpropagation has many applications which do not involve neural networks as such.

This paper first reviews basic backpropagation, a simple method which is now being widely used in areas like pattern recognition and fault diagnosis. Next, it presents the basic equations for backpropagation through time, and discusses applications to areas like pattern recognition involving dynamic systems, systems identification, and control. Finally, it describes further extensions of this method, to deal with systems other than neural networks, systems involving simultaneous equations or true recurrent networks, and other practical issues which arise with this method. Pseudocode is provided to clarify the algorithms. The chain rule for ordered derivatives—the theorem which underlies backpropagation—is briefly discussed.

https://ieeexplore.ieee.org/document/58337

propagation." The concepts here will already be familiar to those who have read the paper by Rumelhart, Hinton, and Williams [2] in the seminal book *Parallel Distributed Processing*, which played a pivotal role in the development of the field. (That book also acknowledged the prior work of Parker [3] and Le Cun [4], and the pivotal role of Charles Smith of the Systems Development Foundation.) This section will use new notation which adds a bit of generality and makes it easier to go on to complex applications in a rigorous manner. (The need for new notation may seem unnecessary to some, but for those who have to *apply* backpropagation to complex systems, it is essential.)

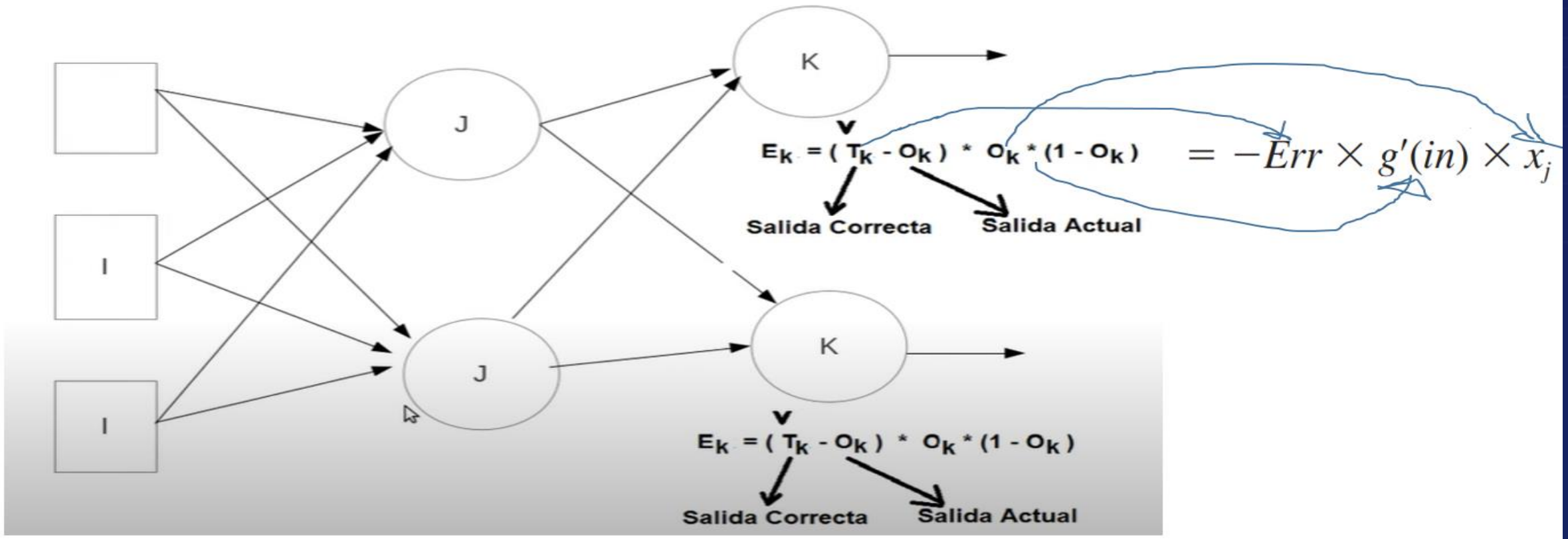
Section III will use the same notation to describe backpropagation through time. Backpropagation through time has been applied to concrete problems by a number of authors, including, at least, Watrous and Shastri [5], Sawai

$$F\_z_i = \frac{\partial E}{\partial z_i} + \sum_{i>i} F\_z_i * \frac{\partial z_i}{\partial z_i}.$$

Profesor: Víctor Viera Balanta

cymetria | tecnalia colombia

Redes Neuronales

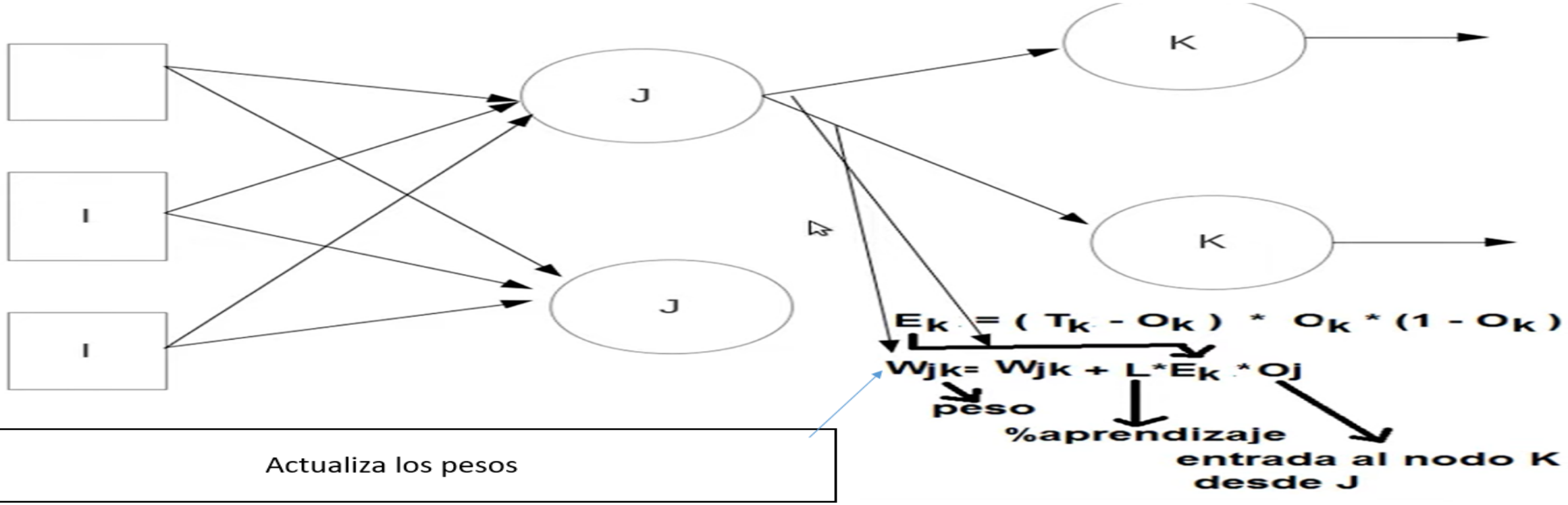


Profesor: Víctor Viera Balanta

Redes Neuronales

Actualiza los pesos

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$



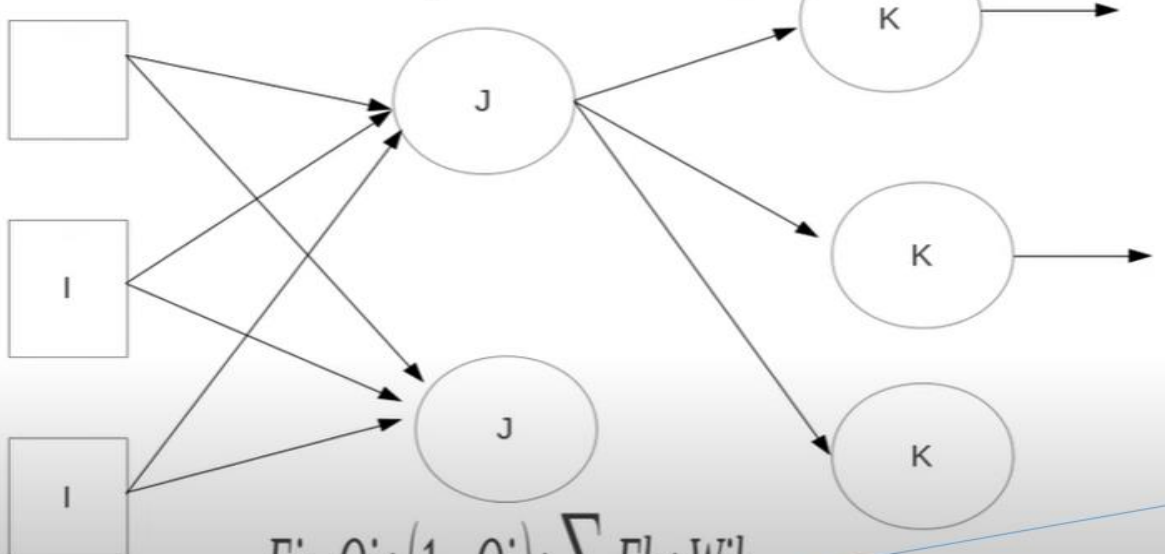
Actualiza los pesos

Profesor: Víctor Viera Balanta

Redes Neuronales

Derivada de la función Sigmoide

$$Ej = Oj * (1 - Oj) * \sum Ek * Wjk$$



$$Ej = Oj * (1 - Oj) * \sum Ek * Wjk$$

Se puede observar, que se multiplican todos los pesos de jk por sus errores..
esto da el error en el neuron J

$$\text{where } \Delta_i = Err_i \times g'(in_i)$$

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i$$

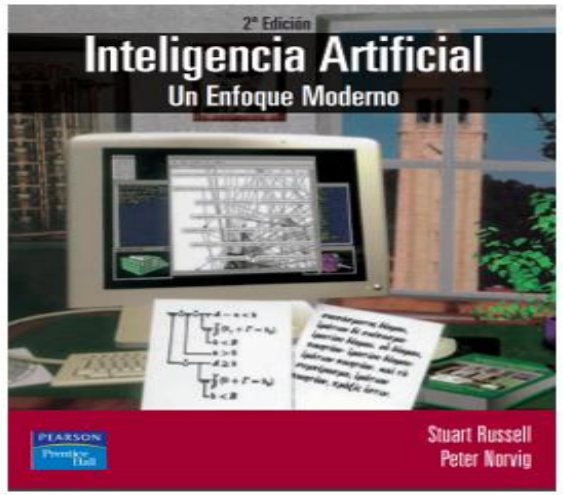
nptelhrd

Redes Neuronales

Algoritmo de BackPropagation

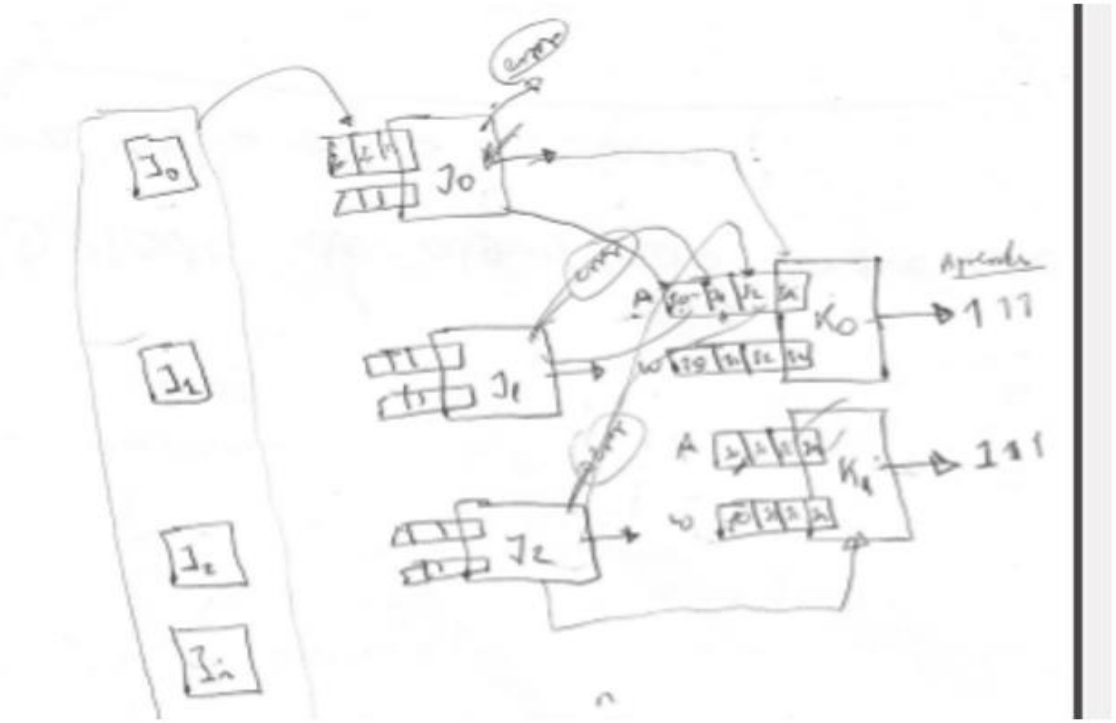
función APRENDIZAJE-PROP-ATRÁS(*ejemplos*, *red*) **devuelve** una red neuronal
entrada: *ejemplos*, un conjunto de ejemplos, cada uno con vector de entrada **x** y vector de salida **y** *red*, una red multicapa con L capas, pesos $W_{j,i}$, función de activación g

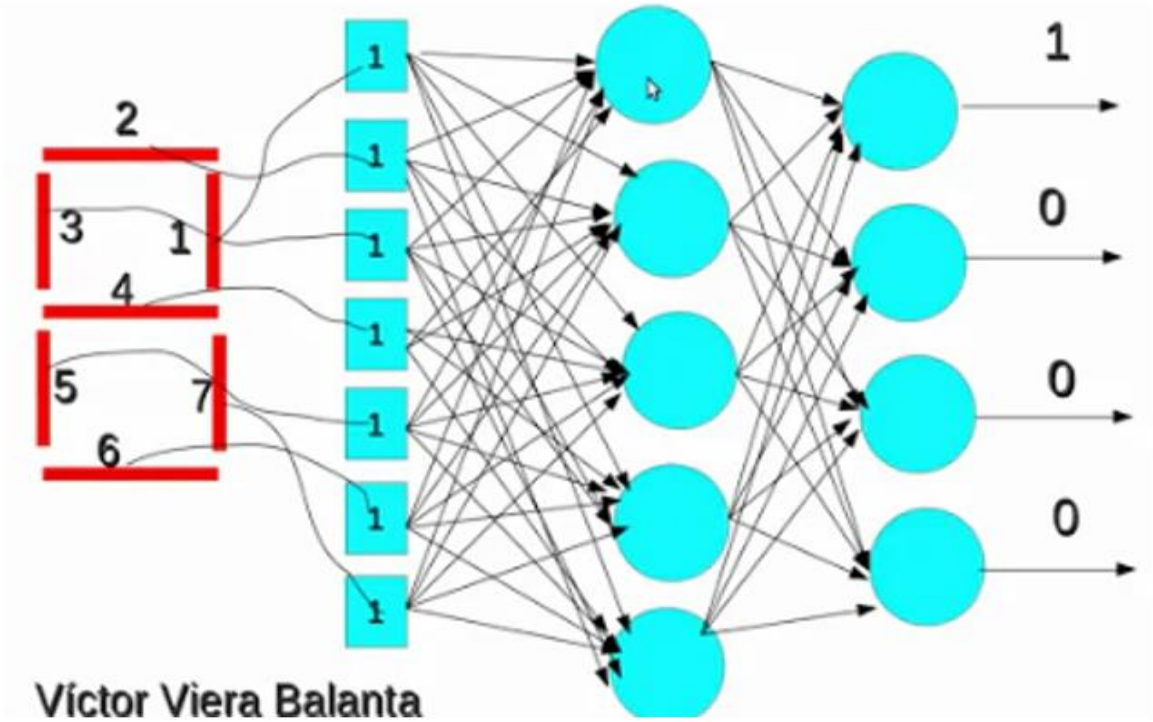
repetir
  para cada e **en** *ejemplos* **hacer**
    para cada nodo j en la capa de entrada **hacer** $a_j \leftarrow x_j[e]$
    para $\ell = 2$ **a** M **hacer**
      $in_i \leftarrow \Sigma_j W_{j,i} a_j$
      $a_i \leftarrow g(in_i)$
    para cada nodo i en la capa de salida **hacer**
      $\Delta_i \leftarrow g'(in_i) \times (y_i[e] - a_i)$
    para $\ell = M - 1$ **a** 1 **hacer**
      para cada nodo j en la capa ℓ **hacer**
        $\Delta_j \leftarrow g'(in_j) \Sigma_i W_{j,i} \Delta_i$
        para cada nodo i en la capa $\ell + 1$ **hacer**
          $W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$
hasta que se satisfaga algún criterio de parada
devolver HIPÓTESIS-RED-NEURONA(*red*)



Redes Neuronales

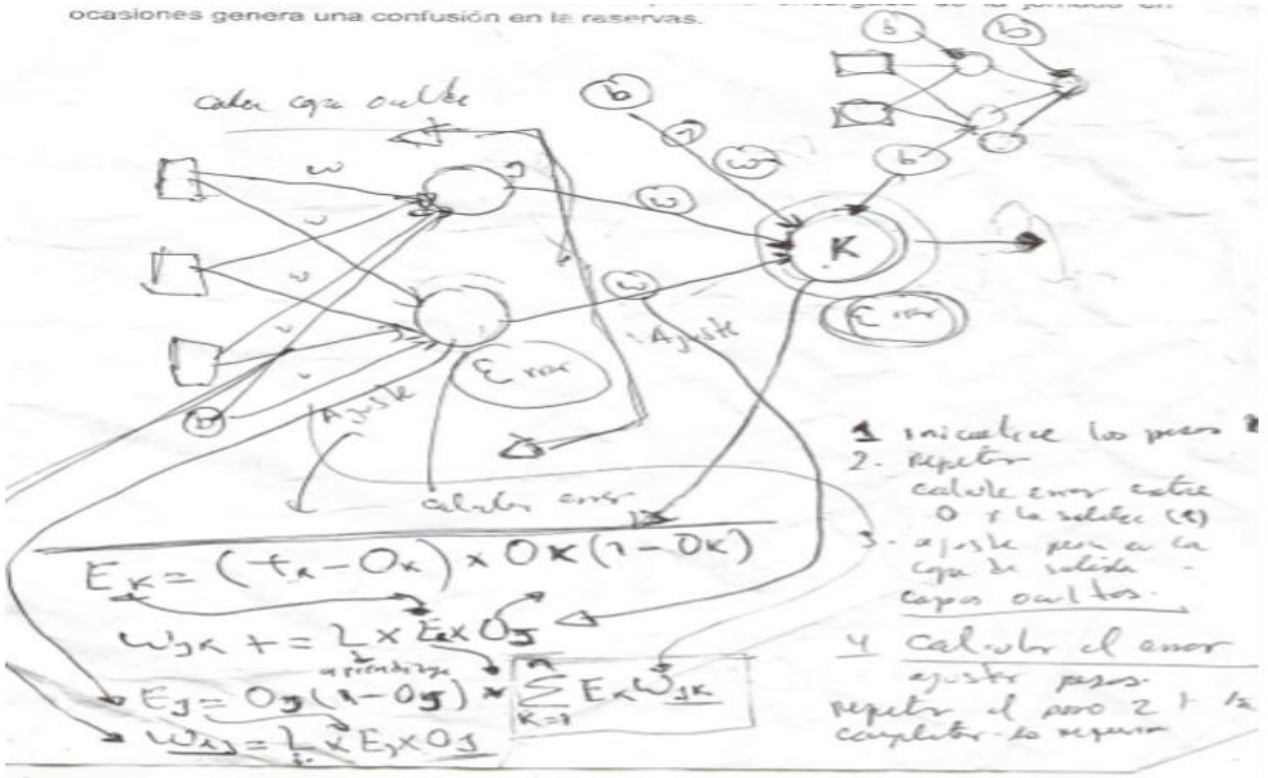
Redes Neuronales Propias

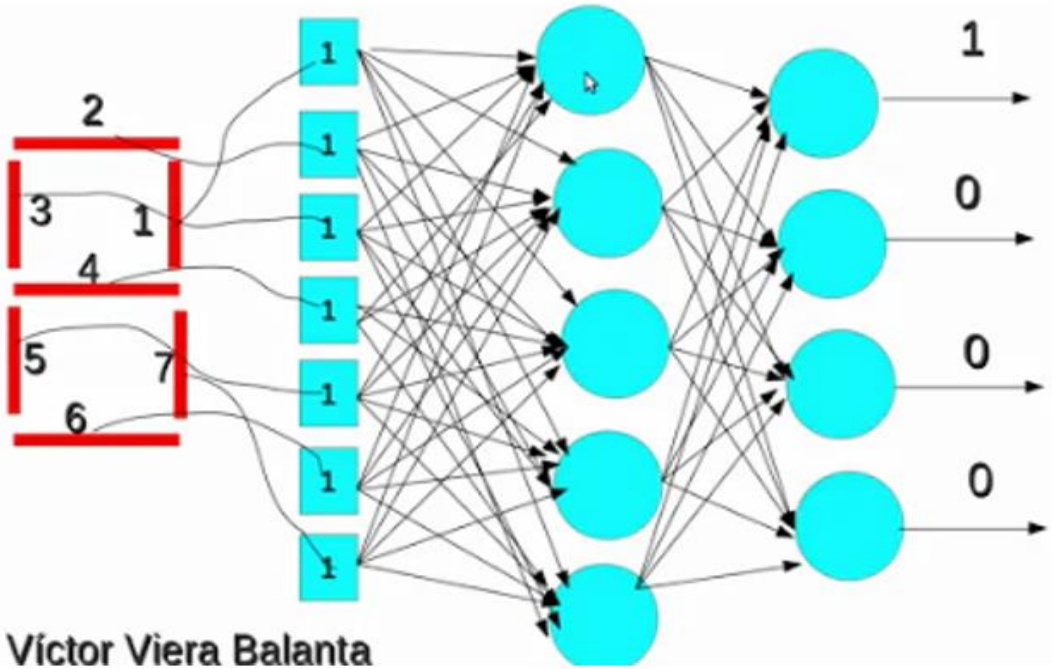




Redes Neuronales

Redes Neuronales Propias





Víctor Viera Balanta

Profesor: Víctor Viera Balanta

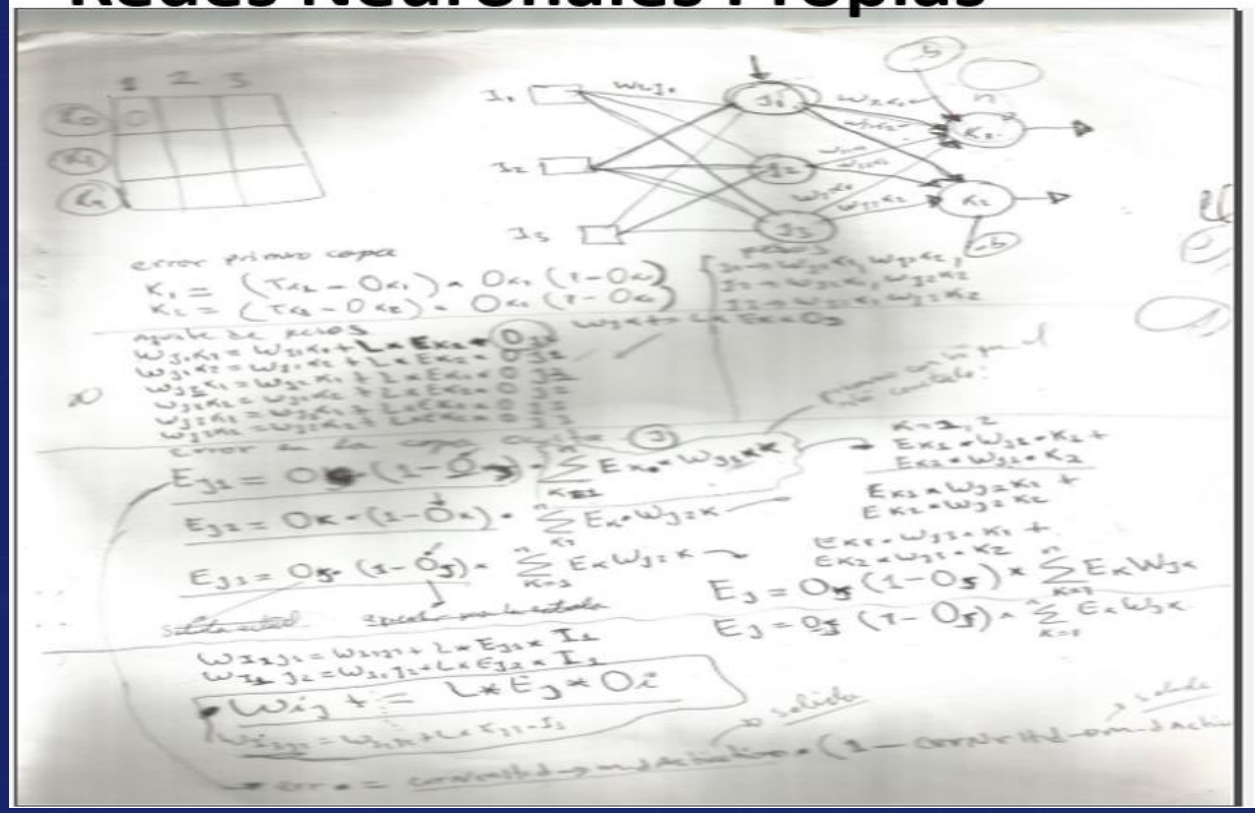
COLOMBIA POTENCIA DE LA VIDA

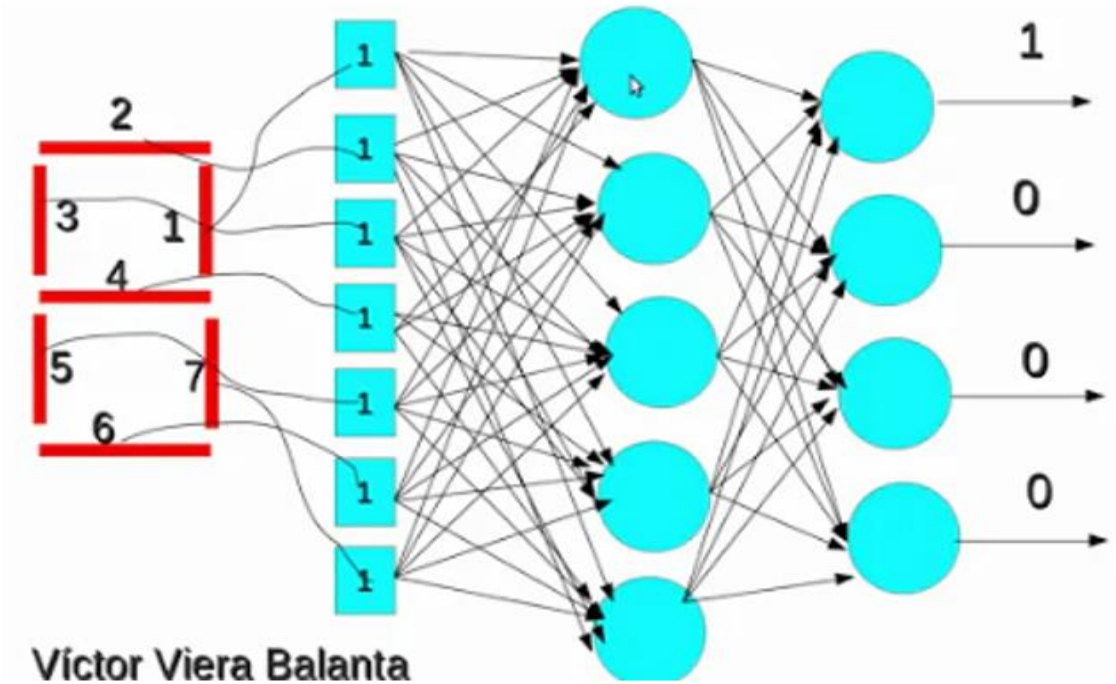
TALENTO TECH

TIC

Redes Neuronales

Redes Neuronales Propias





Profesor: Víctor Viera Balanta

cymetria | tecnal:a colombia

COLOMBIA POTENCIA DE LA VIDA

TALENTO TECH

TIC

UT TALENTOTECH

cymetria | tecnalia colombia

Profesor: Víctor Viera Balanta

COLOMBIA POTENCIA DE LA VIDA

TALENTO TECH

TIC

¡Gracias!

UT TALENTOTECH

cymetria | tecnalia colombia