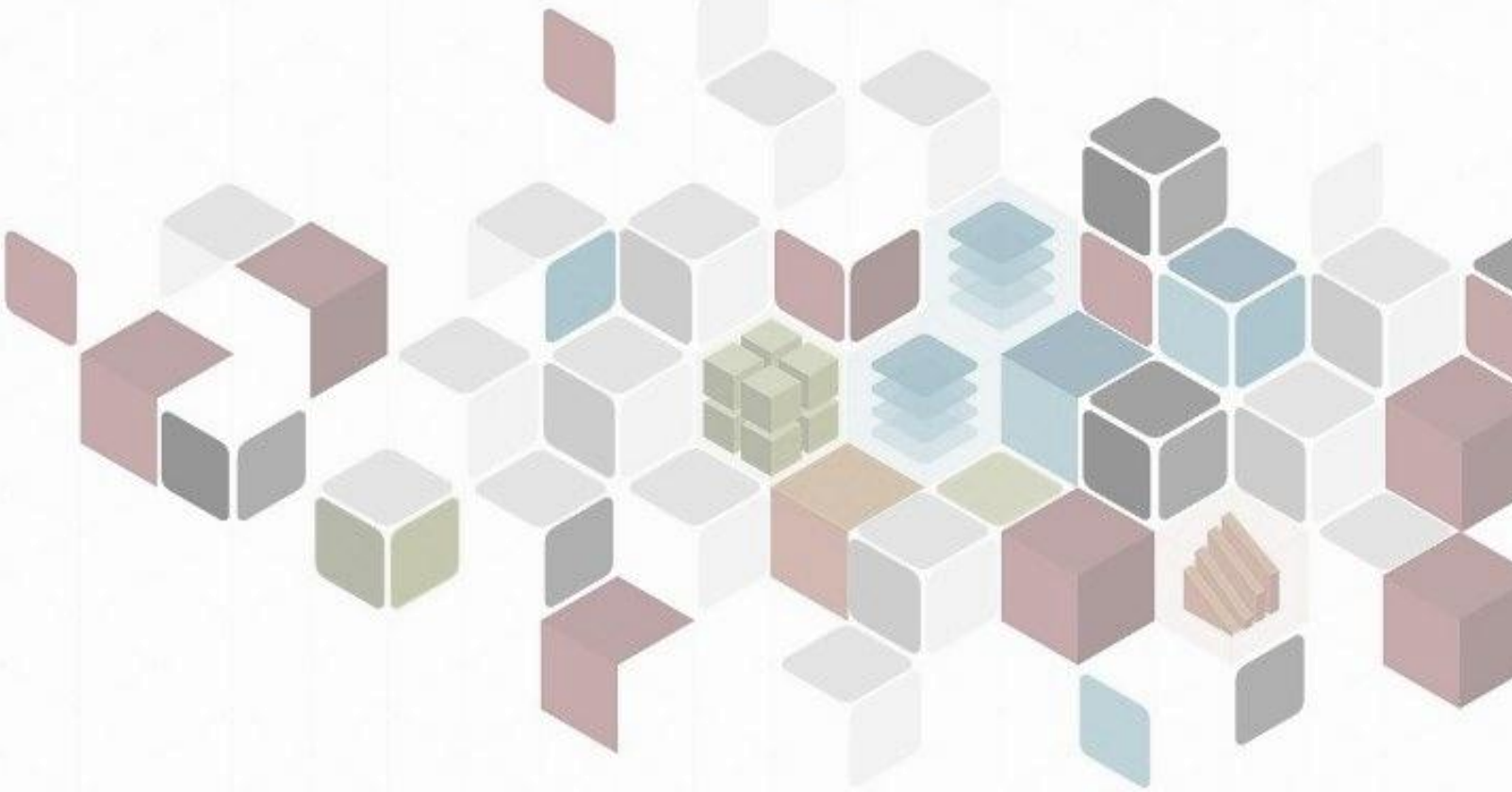
Grandes de Bases de Datos

Modelos de clasificación y predicción

Redes Neuronales



Redes neuronales

- Conforman un método de procesamiento automático inspirado en el sistema nervioso central.

- Se basan en la interconexión de elementos (neuronas) en una red para producir un estímulo de salida.

Redes neuronales

- El conjunto de unidades de E/S tiene conexiones con un peso asociado.

- Durante el proceso de aprendizaje las RNA´s ajustan estos pesos para poder predecir la clase correcta a la que pertenecen las tuplas.

Neurona artificial

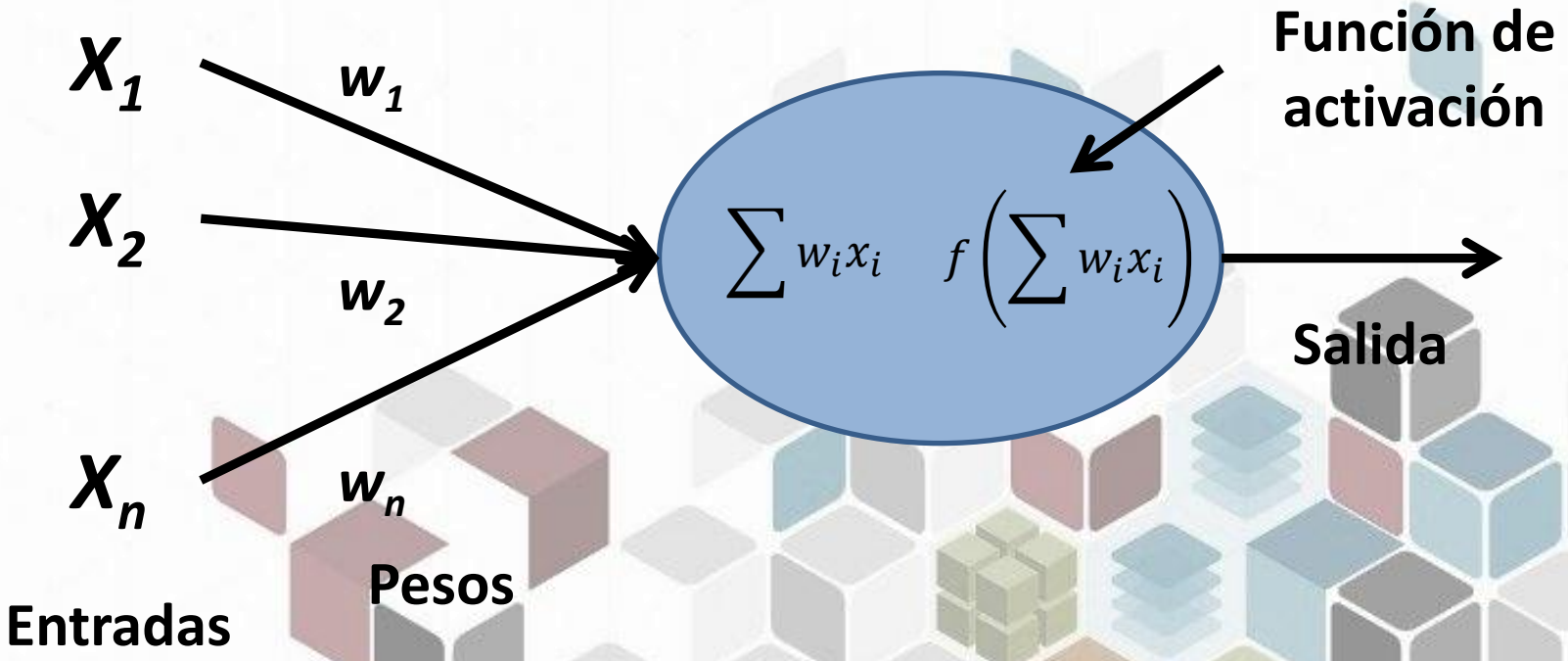
- Es la unidad básica de procesamiento de información sobre la que se fundamenta la operación de una red neuronal artificial (RNA).

- Es un dispositivo simple de cálculo que a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta o salida.

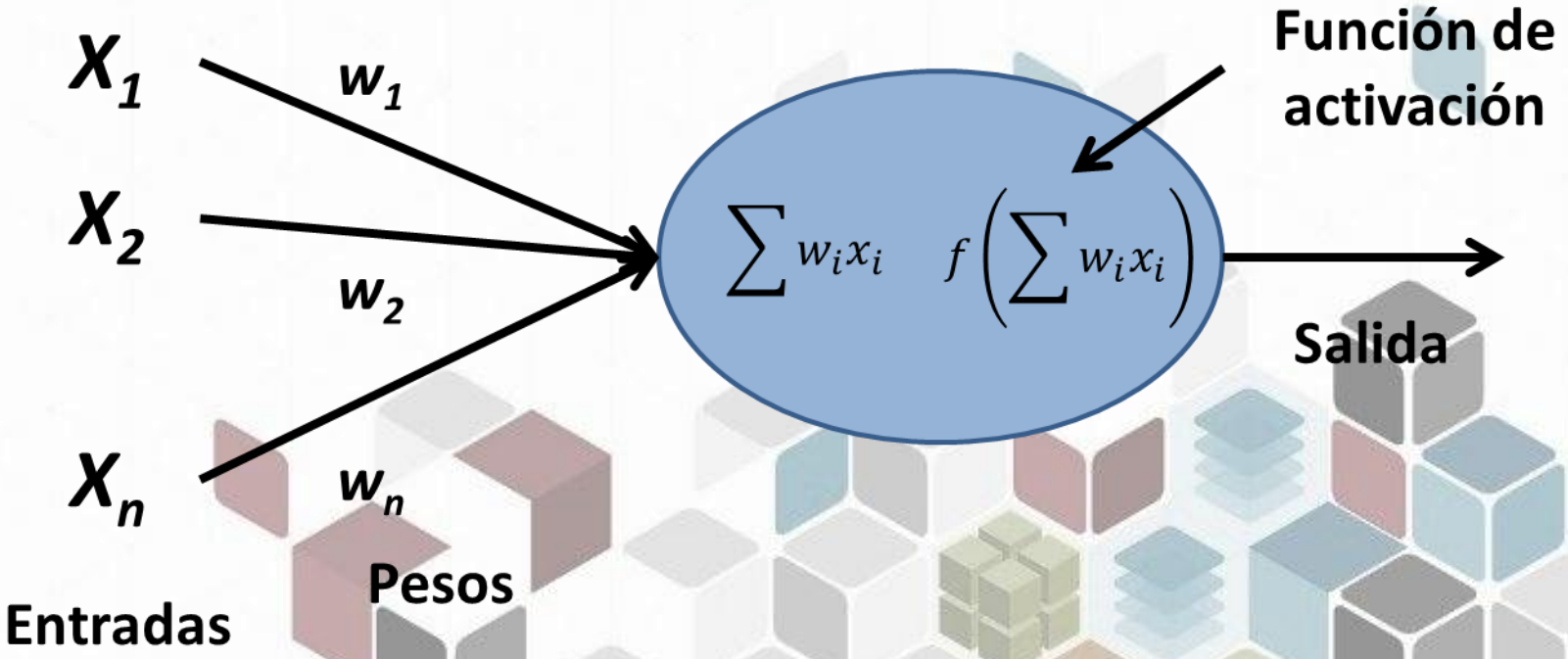
Neurona artificial

- El algoritmo más utilizado se conoce como "*Backpropagation*". Apropiado para problemas con las siguientes características:
  - La entrada de múltiples dimensiones
  - Valores reales
  - Manejo de valores ruidosos
  - Tiempos altos de aprendizaje aceptables
- Ejemplos:
  - Reconocimiento de voz
  - Clasificación de imágenes
  - Predicciones financieras

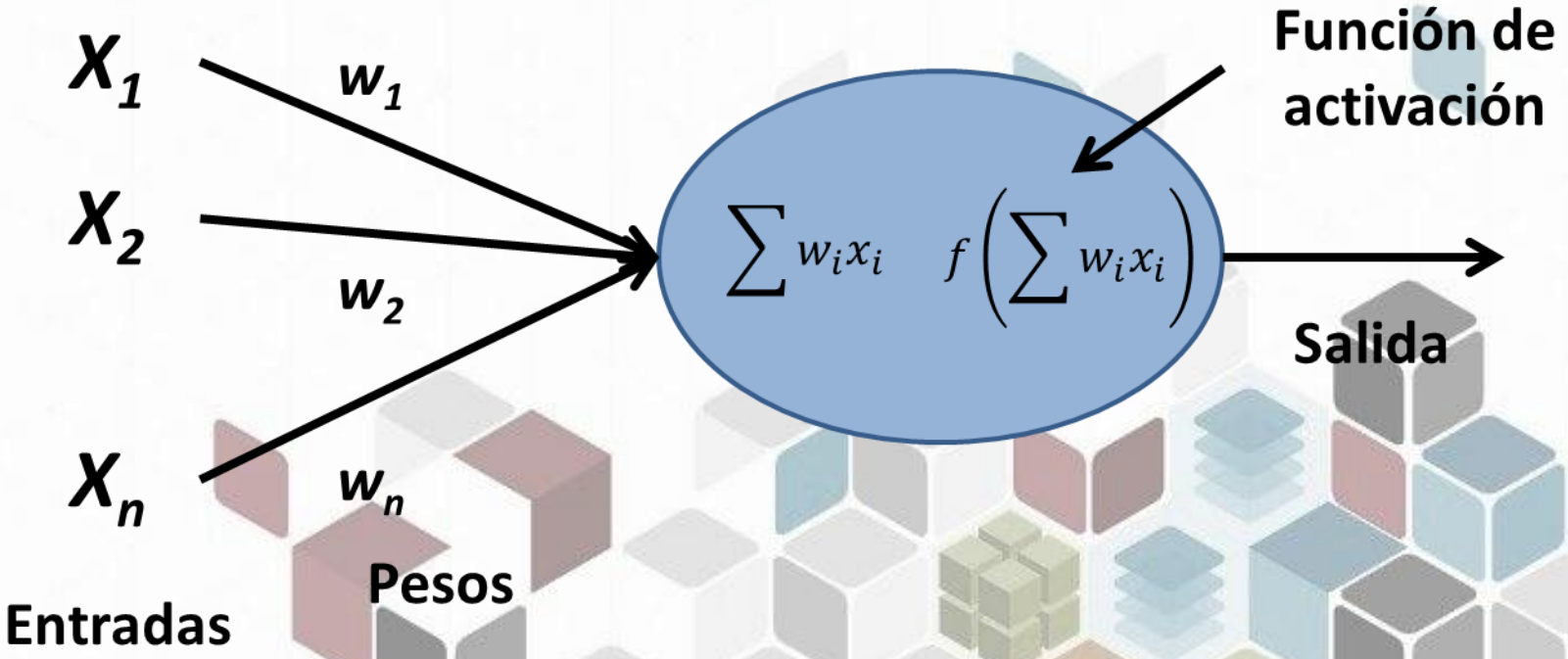
Neurona artificial



Neurona artificial

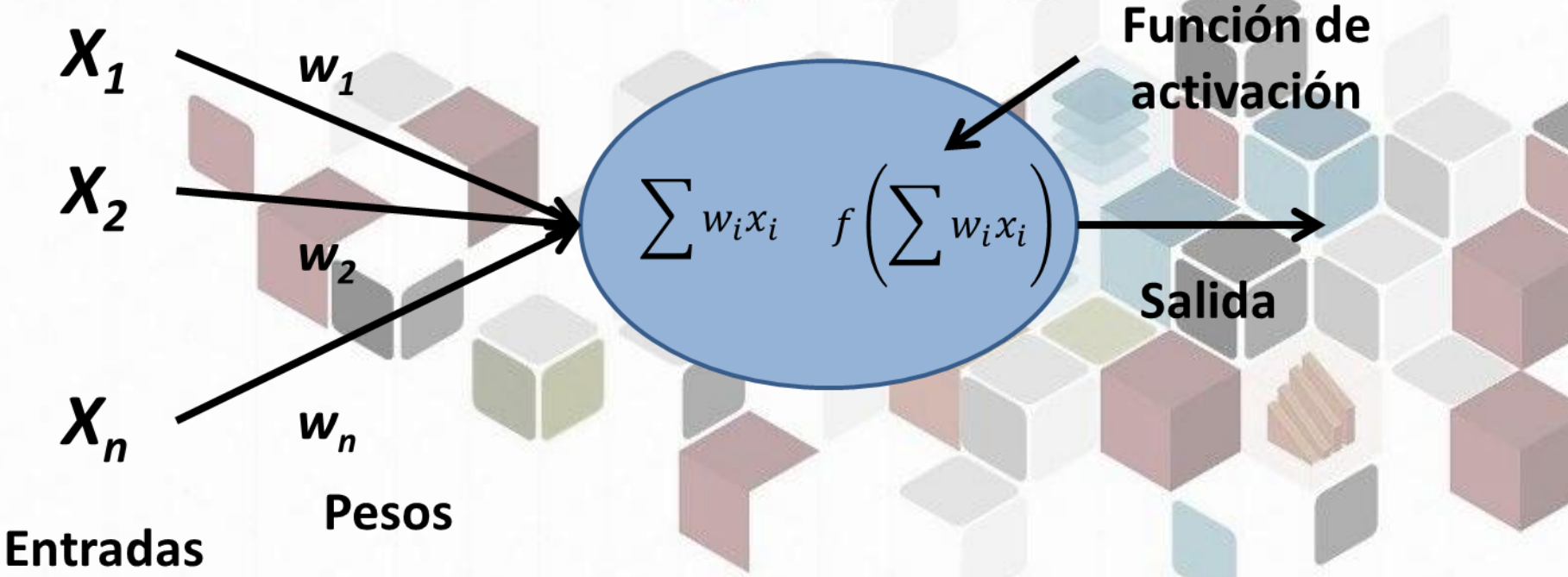


Neurona artificial



Neurona artificial

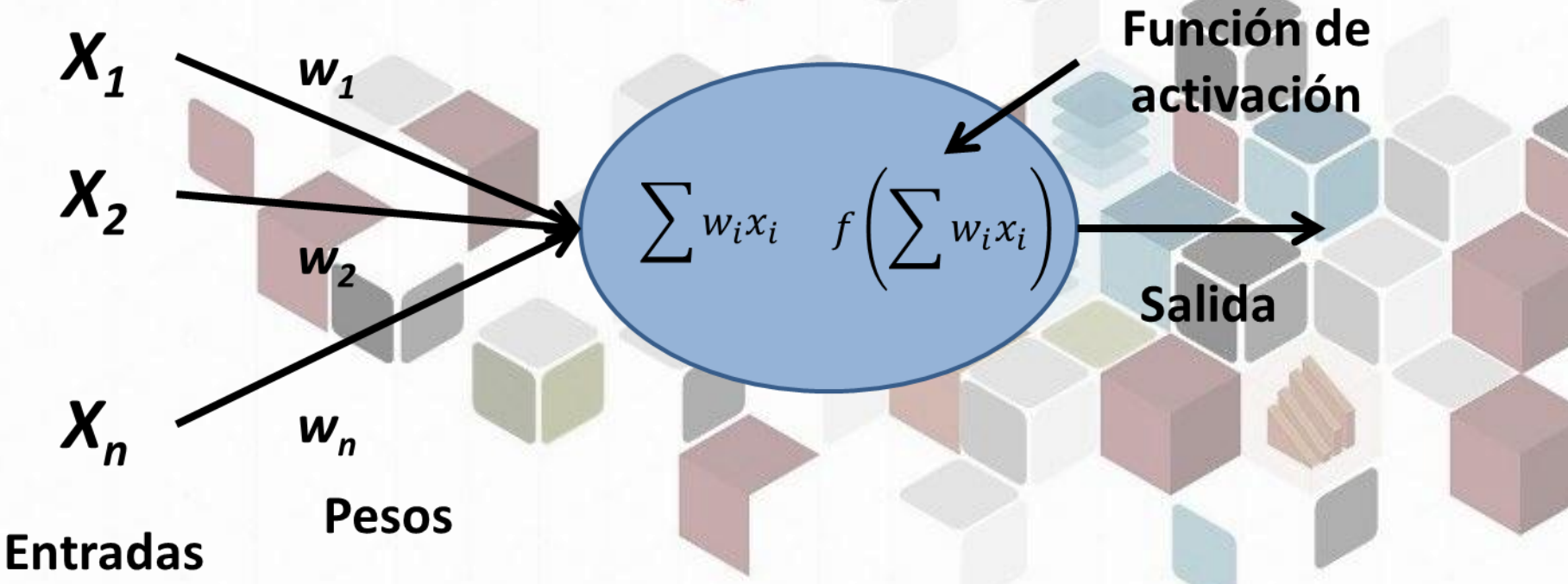
- X_n = Conjunto de entrada
- W_n = Pesos sinápticos, que representan la intensidad de interacción entre dos o más neuronas (entre -1 y 1).
- $\sum w_j x_i$ = Regla de propagación o suma pesada.
- $f\left(\sum w_i x_i\right)$= Función de activación



Neurona artificial

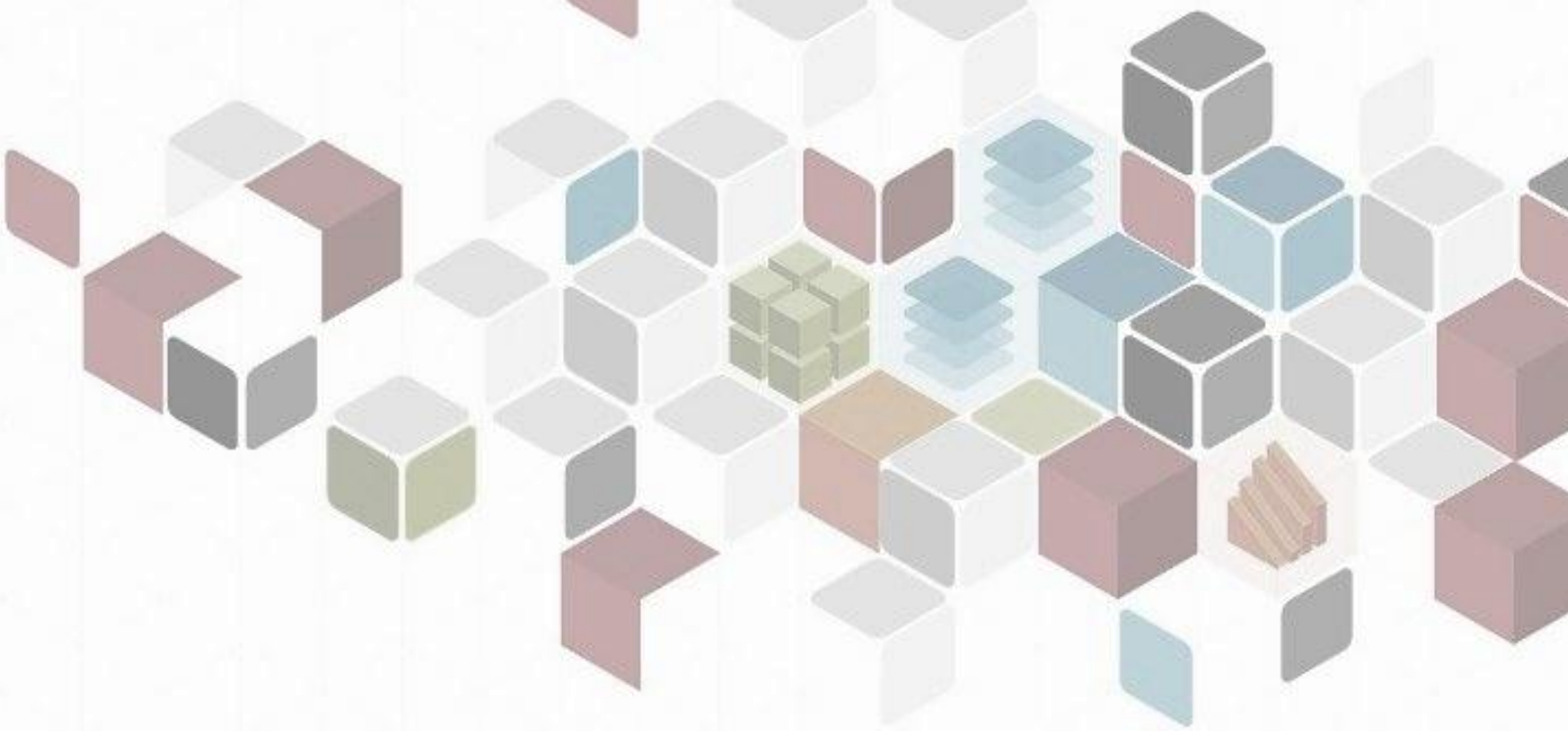
Una neurona puede excitar o inhibir los valores de entrada.

Los valores provenientes de otras neuronas determinan si una neurona se activa o no.

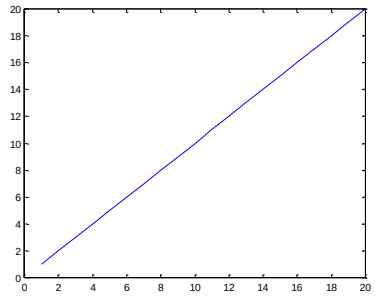


Función de activación

- Es una función acotada y derivable
- Puede o no existir, su salida es dependiente de la función de propagación

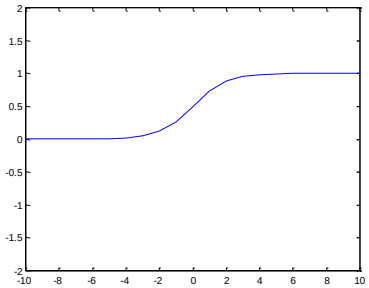


Función de activación



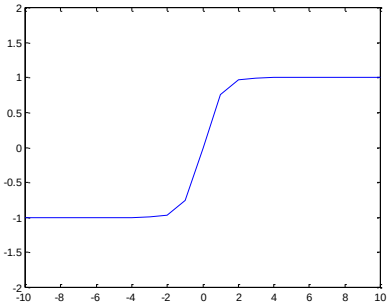
Linear

$$y = x$$



Logistica

$$y = \frac{1}{1+e^{-n}}$$

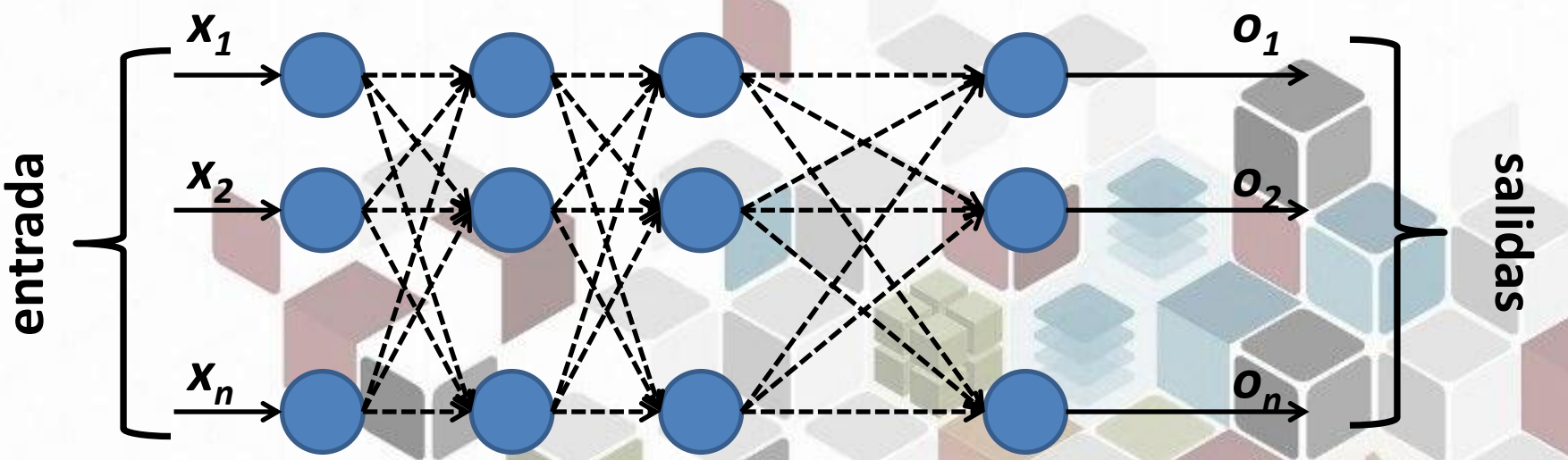


Tangente hiperbolica

$$y = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$$

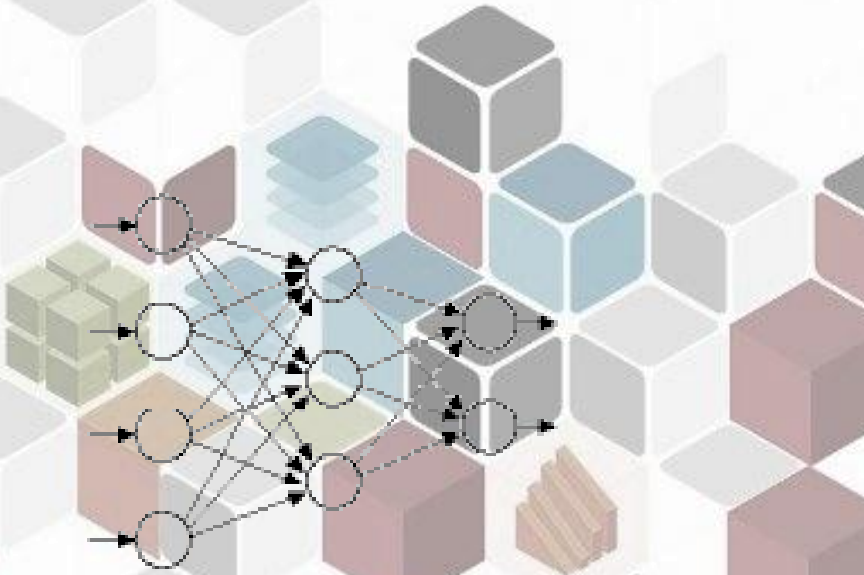
Red neuronal artificial

- Es un conjunto de unidades (neuronas) de procesamiento altamente interconectadas.



Topología o arquitectura de una RNA

- Consiste en la organización de las neuronas en la red formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red.
- Unicapa
- Multicapa
  - Con tres capas basta

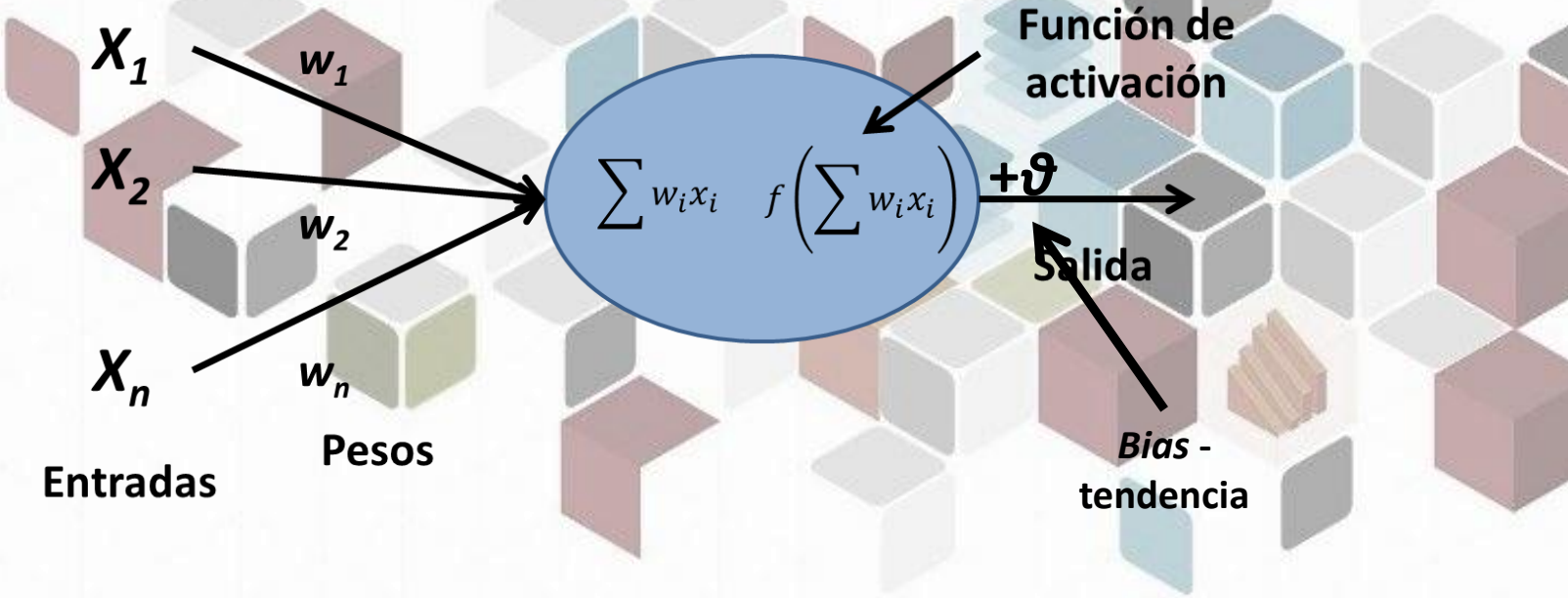


Regla de aprendizaje

- Una regla de aprendizaje define la manera en que los pesos deben de ser modificados en la red neuronal.
- *Regla de corrección por error*. Define el error o diferencia entre el valor correcto y el conseguido por la red.
- Se utiliza para modificar los pesos y así disminuir gradualmente el error.

Perceptrón

El perceptrón simple fue inicialmente investigado por Rosenblatt en 1962. El perceptrón simple tiene una estructura de varias neuronas de entrada y una de salida. El perceptrón no tiene capa oculta.



Perceptrón

Procedimiento:

1.-La red comienza en un estado aleatorio. Los pesos de las neuronas poseen valores pequeños y aleatorios entre (-1 y 1).

2.-Seleccionar un vector de entrada, **X**, a partir del conjunto de ejemplos de entrenamiento.

Perceptrón

Procedimiento:

3.-Se propaga la activación hacia adelante a través de los pesos en la red para calcular la salida.

4.-Si la salida es correcta se vuelve al paso 2.

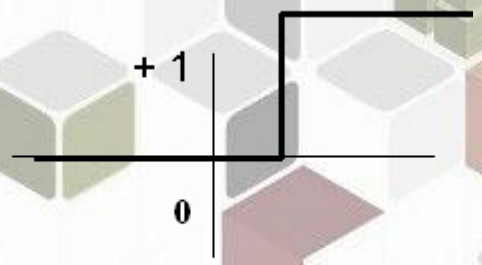
5.-En caso contrario se realiza un ajuste en los pesos de tal forma que la salida de la red se asemeje a los valores esperados.

Perceptrón

Calcula una función lineal para cada Y_j (salida):

$$Y_j = f\left(\sum_{i=1}^{n} W_{i,\,j} \cdot x_i - \theta\right)$$

Aquí, la función de activación (*f*) es la función escalón.



Actualización de los pesos

Ley de aprendizaje:

$$w(t + 1) = w(t) + J_k x_k$$

J_k = Valor real – valor devuelto por la red

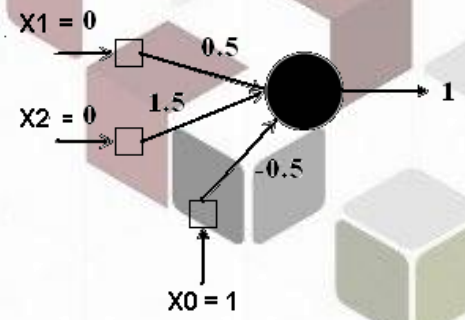
$J_k = 0$ patrón bien clasificado, no hay error

$J_k \neq 0$ error al clasificar

Proceso de aprendizaje

- Ejemplo

X1	X2	OR
0	0	0
0	1	1
1	0	1
1	1	1



Pesos iniciales:

Wo = -0.5

W1 = 0.5

W2 = 1.5

Ejemplo cont…

W_1=0.5 W_2=1.5 W_0=-0.5

X_1=0 X_2=0

Y=0(.5)+0(1.5)+0.5=0.5 //FA// =1

Pero la salida debería de ser 0 entonces pasamos a actualizar los pesos.

D-Y= 0-1=-1

$W_1(t+1)$=0.5+(-1)(0)=0.5

$W_2(t+1)$=1.5+(-1)(0)=1.5

$W_0(t+1)$=-0.5+(-1)1=0.5

Ejemplo cont…

W_1=0.5 W_2=1.5 W_0=0.5

X_1=0 X_2=1

Y=0(.5)+(1)(1.5)-.5=1 //FA// =1

X_1=1 X_2=0

Y=1(.5)+(0)(1.5)-.5=0 //FA// =1

X_1=1 X_2=1

Y=1(.5)+(1)(1.5)-.5=1.5 //FA// =1

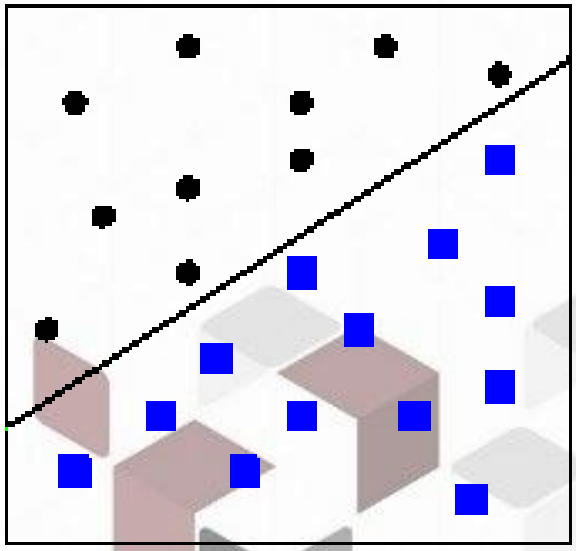
X_1=0 X_2=0

Y=0(.5)+(0)(1.5)-.5=-5 //FA// =0

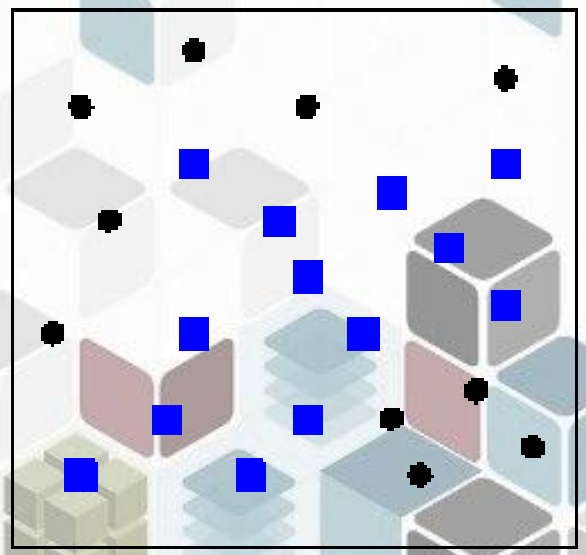
¿Qué problemas se resuelven con el Perceptrón?

Los problemas que podemos resolver con un perceptrón son aquellos que se denominan linealmente separables

Perceptrón



Linealmente separables

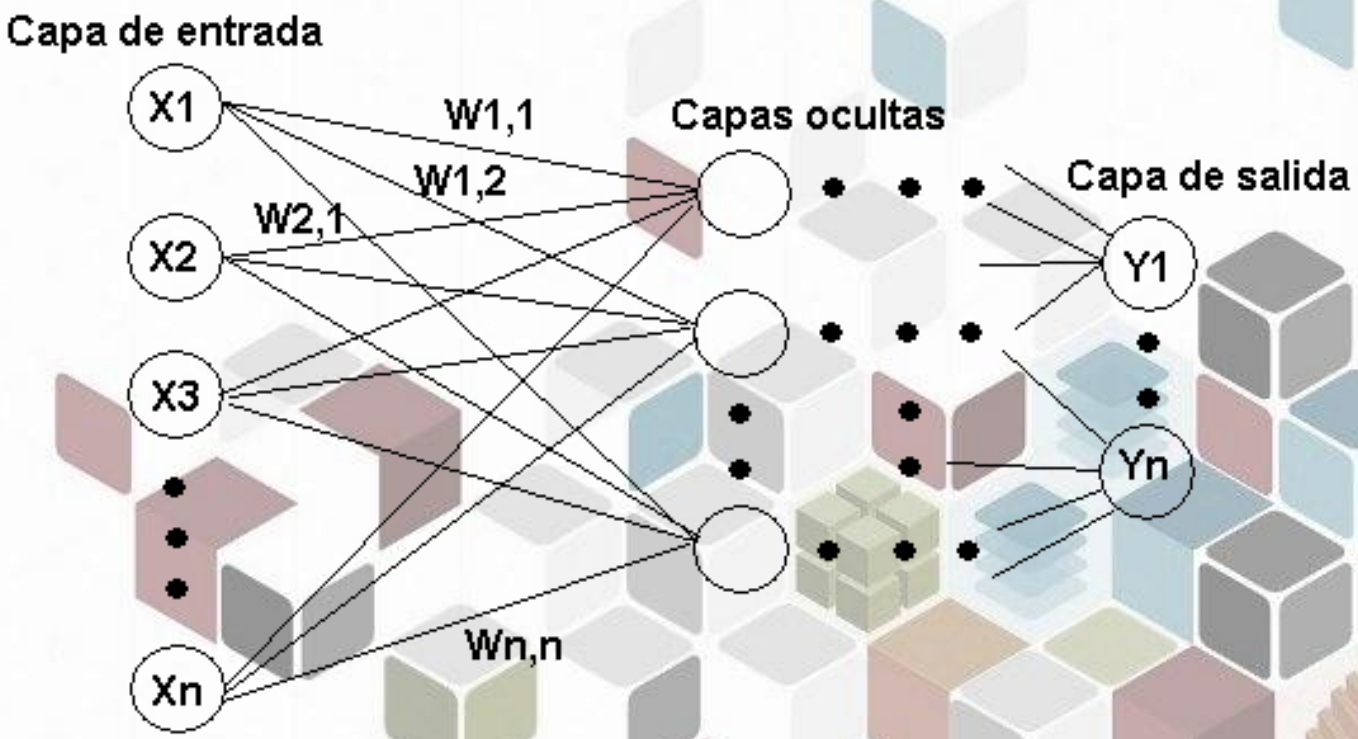


no separables linealmente

Perceptrón Multicapa (MLP)

El perceptrón no puede resolver problemas que no son linealmente separables, pero pueden ser modelados mediante el empleo del Perceptrón Multicapa, es decir una red neuronal en forma de cascada, que tiene una o más capas ocultas.

MLP


Capa de entrada
X1
X2
X3
Xn
W1,1
W1,2
W2,1
Wn,n
Capas ocultas
Capa de salida
Y1
Yn

Aprendizaje MLP

Hay que actualizar dos conjuntos de pesos: aquellos entre la capa oculta o intermedia y la de salida, y aquellos entre la capa de entrada y la capa intermedia.

Se propaga el error hacia atrás debido a los errores que tienen lugar en el segunda capa de pesos y se asigna el error proporcional a los pesos que lo causan.

Red backpropagation

- Esta red suele entrenarse con el algoritmo denominado *backpropagation* de errores, motivo por el cual a una red multicapa (con tres capas) con regla de aprendizaje *backpropagation* suele denominarse *Red backpropagation*.

Red backpropagation

- El nombre *backpropagation* surge del cálculo que se hace en el sentido inverso de la red, propagándose desde los nodos de salida hacia los nodos de entrada

Algoritmo de backpropagation

1. Inicializar los pesos a valores pequeños aleatorios.
2. Escoger un vector de entrada y presentarlo a la capa de entrada.
3. Propagar la activación hacia delante a través de los pesos hasta que la activación alcance las neuronas de la capa de salida
4. Calcular los valores del error para la capa de salida.
5. Calcular los valores del error para la capa oculta.
6. Actualizar los pesos.
7. Repetir del paso 2 al 6 para todas las tuplas de entrada.

Algoritmo de backpropagation

El error para la capa de salida se calcula de la siguiente forma:

$$\mathbf{Errj} = \mathit{Oj}(1 - \mathit{Oj})(\mathit{Tj} - \mathit{Oj})$$

Donde *Oj* es la salida actual de la red para la neurona *j* y *Tj* es el valor esperado (real).

Algoritmo backpropagation

El error para la(s) capa(s) ocultas se calcula de la siguiente forma:

$$Err_i = O_i(1 - O_i) \sum_k Err_k W_{jk}$$

- Err_k es el error de la neurona k.
- W_{jk} es el peso de la conexión de la neurona j a la neurona k.

Algoritmo backpropagation

La actualización de los pesos y la neurona de bias (umbral) se calculan como sigue:

$$\triangle W_{jk} = (n)\, Err_k O_j$$

$$W_{jk} = W_{jk} + \triangle W_{jk}$$

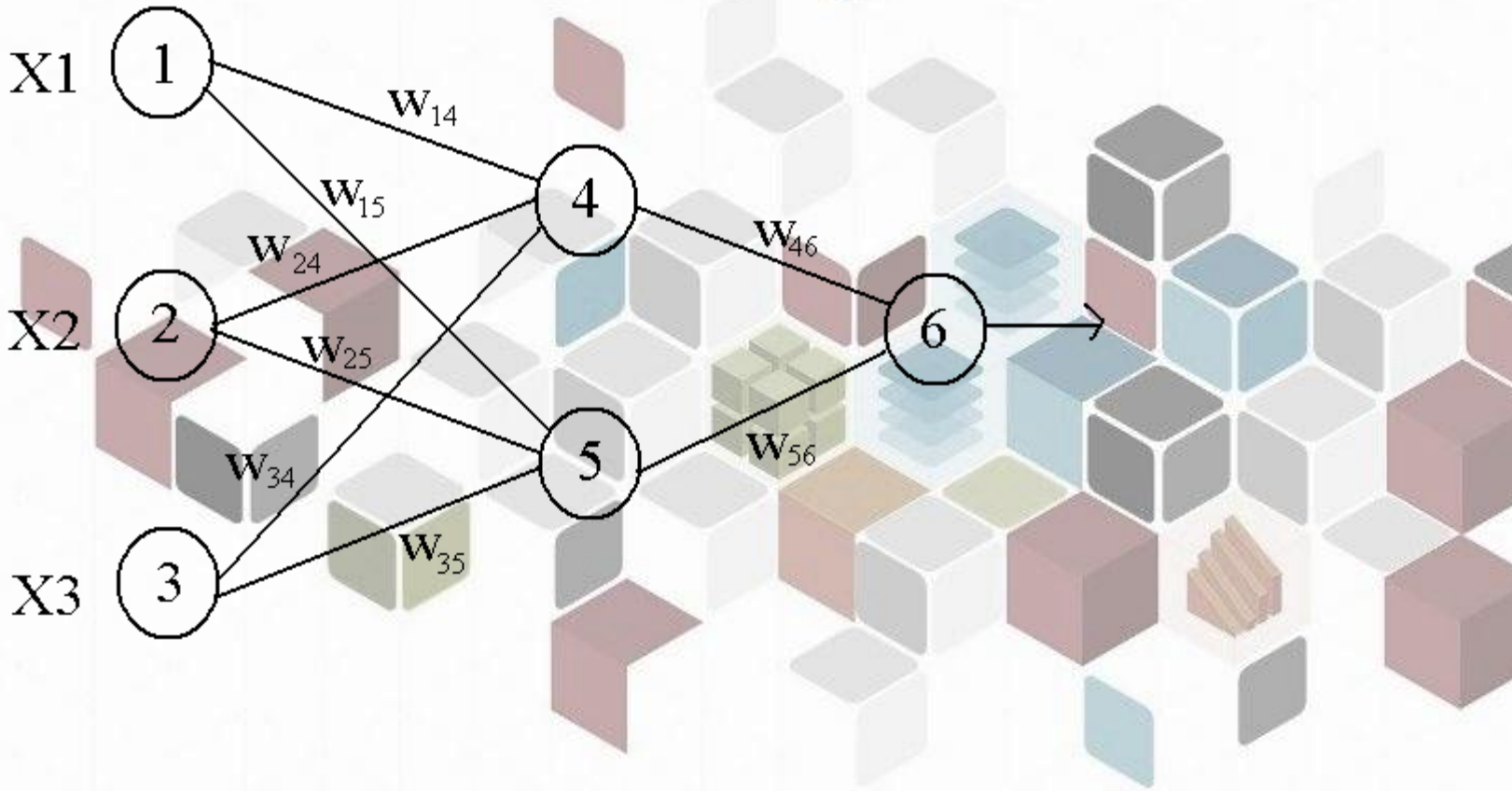
Donde n es la tasa de aprendizaje.

$$\triangle \Theta_k = (n)\, Err_k$$

$$\Theta_k = \Theta_k + \triangle \Theta_k$$

Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

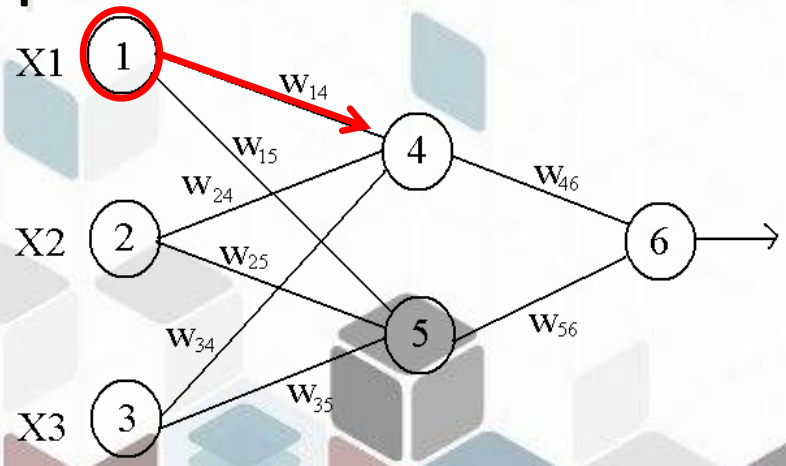


Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

X1	X2	X3	W_{14}	W_{15}	W_{24}	W_{25}
1	0	1	0.2	-0.3	0.4	0.1

W_{34}	W_{35}	W_{46}	W_{56}	B_4	B_5	B_6
-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1



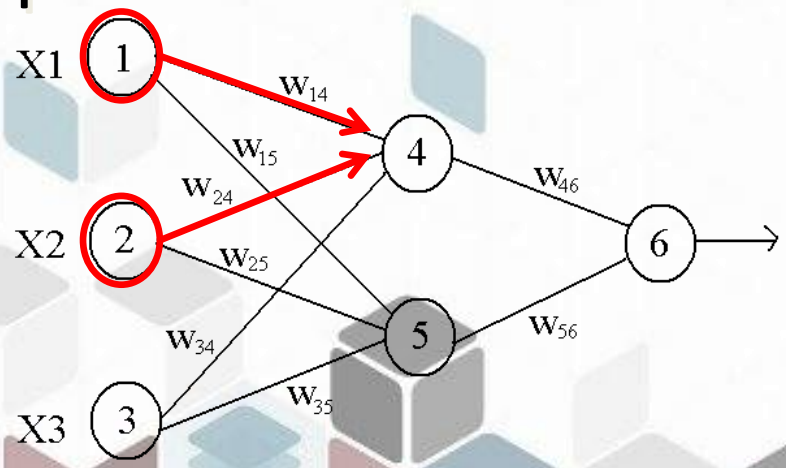
Neurona	Entrada a la red	Salida O_j
4	$1 * 0.2$	$1/(1+e^{-0.3}) = 0.332$
5	-0.3 + 0 + 0.2 + 0.2 = 0.1	$1/(1+e^{-0.1}) = 0.525$
6	(-0.3)(0.332)-(0.2)(0.525) + 0.1 = -0.105	$1/(1+e^{0.105}) = 0.474$

Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

X1	X2	X3	W_{14}	W_{15}	W_{24}	W_{25}
1	0	1	0.2	-0.3	0.4	0.1

W_{34}	W_{35}	W_{46}	W_{56}	B_4	B_5	B_6
-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1



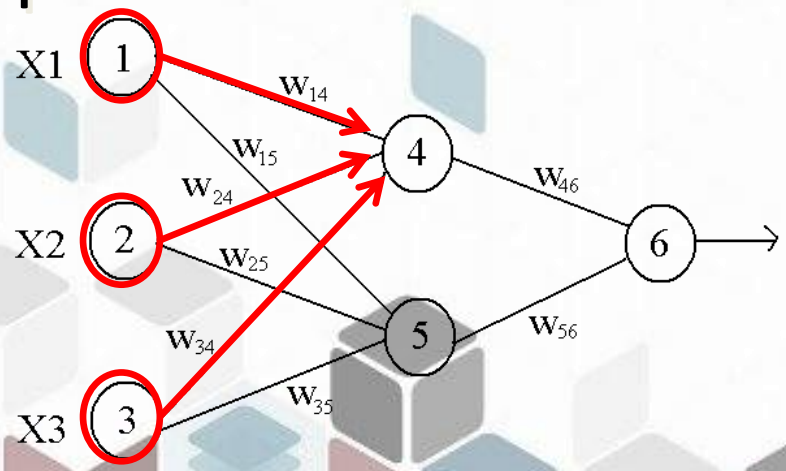
Neurona	Entrada a la red	Salida O_j
4	$1 * 0.2 + 0 * -0.4$	$1/(1+e^{-0.3}) = 0.332$
5	-0.3 + 0 + 0.2 + 0.2 = 0.1	$1/(1+e^{-0.1}) = 0.525$
6	(-0.3)(0.332)-(0.2)(0.525) + 0.1 = -0.105	$1/(1+e^{0.105}) = 0.474$

Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

X1	X2	X3	W_{14}	W_{15}	W_{24}	W_{25}
1	0	1	0.2	-0.3	0.4	0.1

W_{34}	W_{35}	W_{46}	W_{56}	B_4	B_5	B_6
-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1



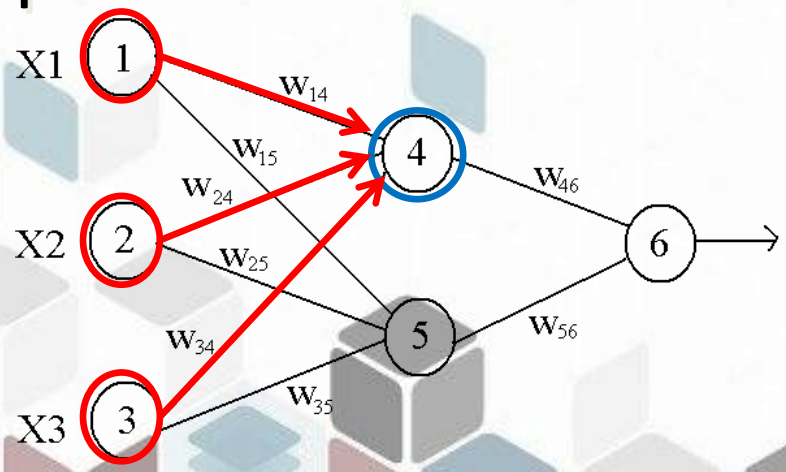
Neurona	Entrada a la red	Salida O_j
4	$1*0.2+0*-0.4+1*-0.5$	$1/(1+e^{-0.3}) = 0.332$
5	-0.3 + 0 + 0.2 + 0.2 = 0.1	$1/(1+e^{-0.1}) = 0.525$
6	(-0.3)(0.332)-(0.2)(0.525) + 0.1 = -0.105	$1/(1+e^{0.105}) = 0.474$

Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

X1	X2	X3	W_{14}	W_{15}	W_{24}	W_{25}
1	0	1	0.2	-0.3	0.4	0.1

W_{34}	W_{35}	W_{46}	W_{56}	B_4	B_5	B_6
-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1



Neurona	Entrada a la red	Salida O_j
4	$1*0.2+0*-0.4+1*-0.5 = -0.3$	$1/(1+e^{-0.3}) = 0.332$
5	-0.3 + 0 + 0.2 + 0.2 = 0.1	$1/(1+e^{-0.1}) = 0.525$
6	(-0.3)(0.332)-(0.2)(0.525) + 0.1 = -0.105	$1/(1+e^{0.105}) = 0.474$

Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

Neurona j	Err_j
6	(0.474)(1 – 0.474)(1 – 0.474) = 0.1311
5	(0.525)(1-0.525)(0.1311)(-0.2) = -0.0065
4	(0.332)(1-0.332)(0.1311)(-0.3) = -0.0087

Algoritmo backpropagation

Ejemplo: X=(1,0,1) y la clase esperada es 1

Peso o Bias	Nuevo valor
W_{46}	-0.3 + 0.9(0.1311)(0.332) = -0.261
W_{56}	-0.2 + 0.9(0.1311)(0.525) = -0.138
W_{14}	0.2 + 0.9(-0.0087)(1) = 0.192
W_{15}	- 0.3 + 0.9(-0.0065)(1) = -0.306
W_{24}	0.4 + 0.9(-0.0087)(0) = 0.4
W_{25}	0.1 + 0.9(-0.0065)(0) = 0.1
W_{34}	-0.5 + 0.9(-0.0087)(1) = -0.508
W_{35}	0.2 + 0.9(-0.0065)(1) = 0.194
B_6	0.1 + 0.9(0.1311) = 0.218
B_5	0.2 + 0.9(-0.0065) = 0.194
B_4	-0.4 + 0.9(-0.0087) = -0.408

Consideraciones para clasificación

- El numero de neuronas de entrada será igual al número de atributos que tengamos.
- Las neuronas de la capa intermedia son a prueba y error, existen heurísticas sobre el número de neuronas en esta capa.
- El número de neurona de salida dependerá del número de clases existentes, si solo tiene dos clases el atributo clasificados (yes, no) sólo será necesaria una neurona, en otro caso, será igual al número de clases.
- Los valores categóricos deben de ser transformados a valores numéricos.

Función de error

- Nos sirve para saber la magnitud del error

$$E_p = \frac{1}{2}\sum_{k=1}^{M}(d_{pk} - Z_{pk})^2$$

- Este función refleja la capacidad de adaptación de la red, debe de disminuir conforme la red vaya aprendiendo hasta una cota deseada.