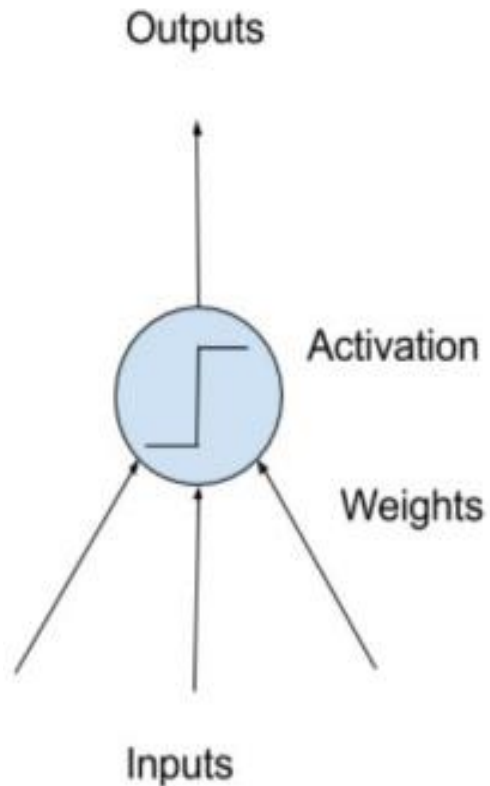
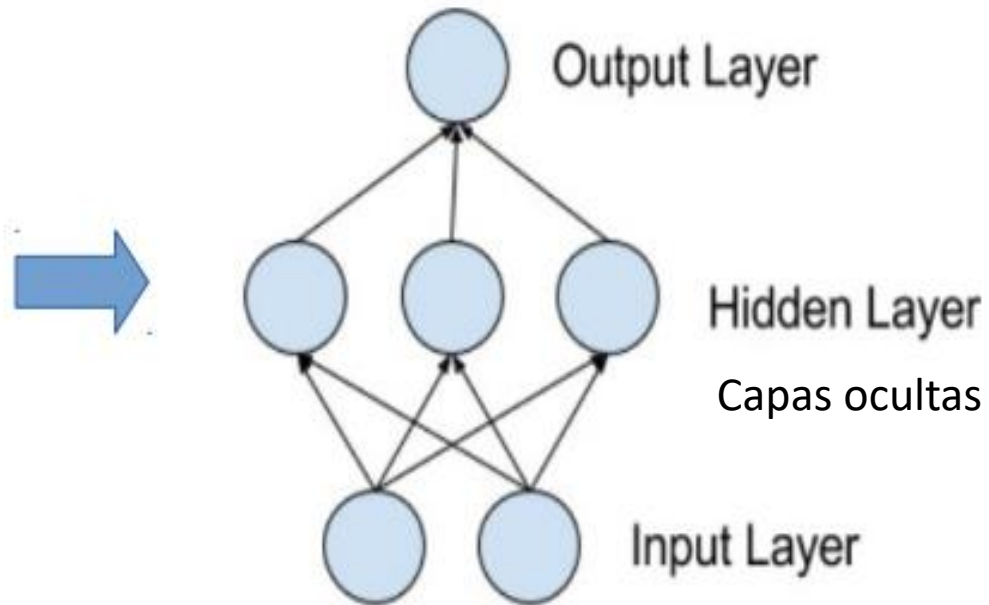


# Como funciona una neurona

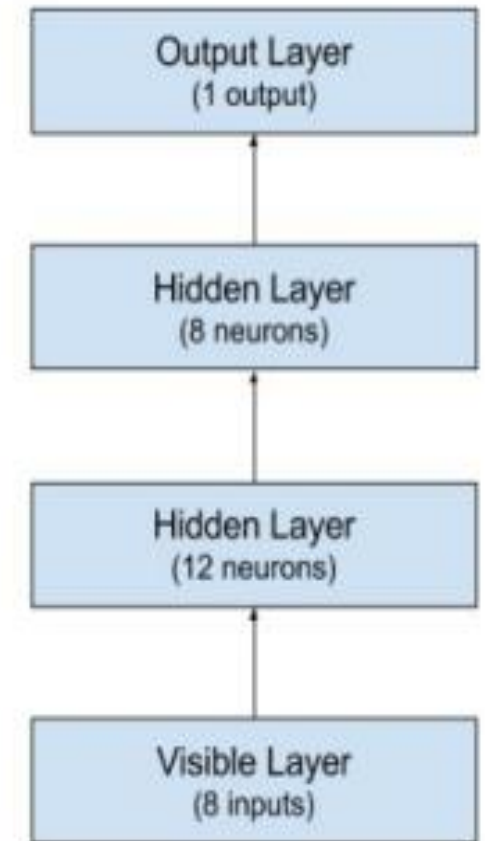
Background



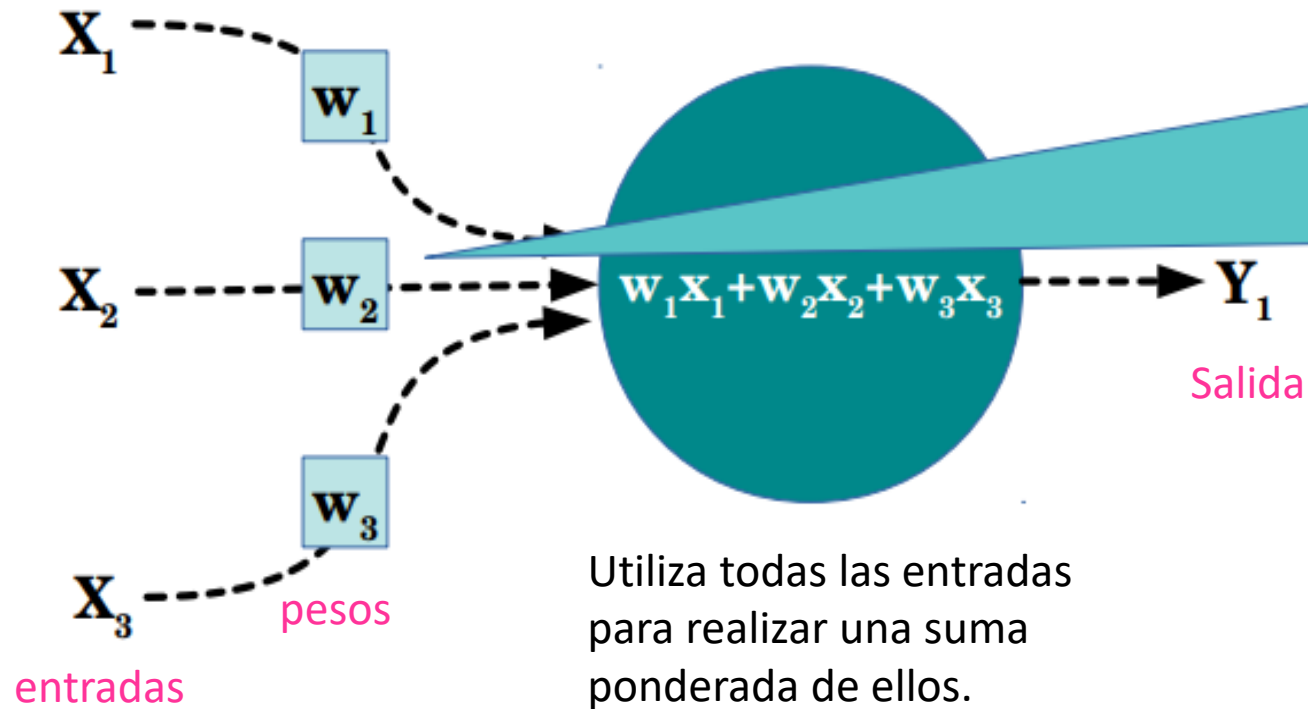
Si juntamos varias neuronas tenemos un perceptrón multicapa



Por ejemplo, podríamos tener



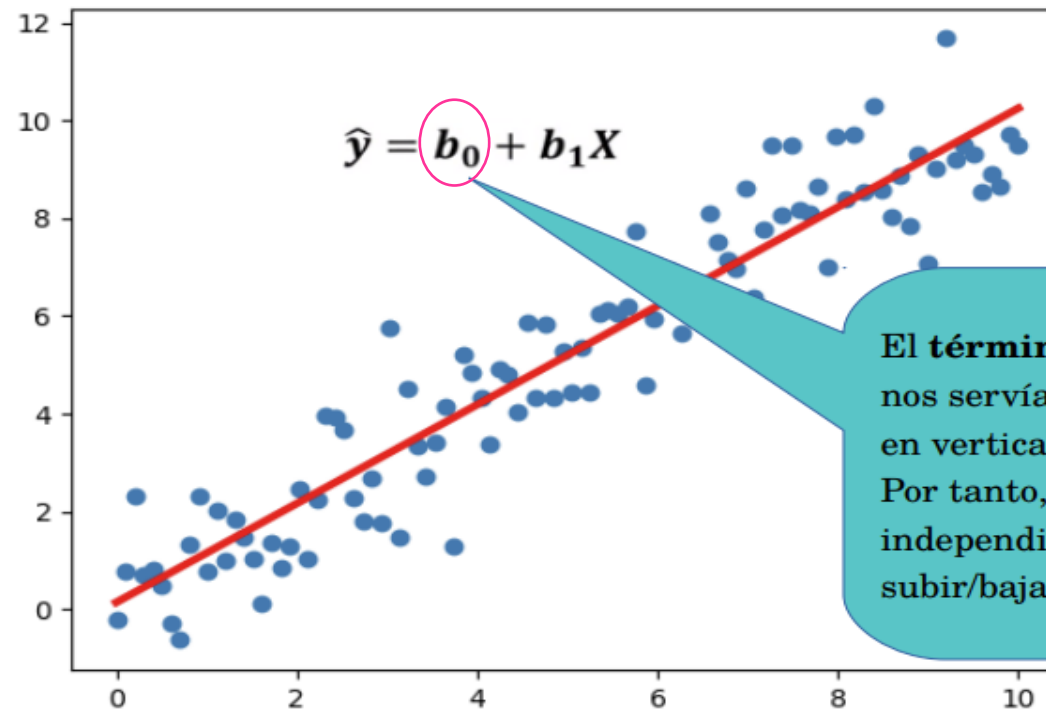
## Neurona



Hay que tener en cuenta que la suma viene dada por el **peso** ( $w$ ) que se le asigna a cada una de las entradas

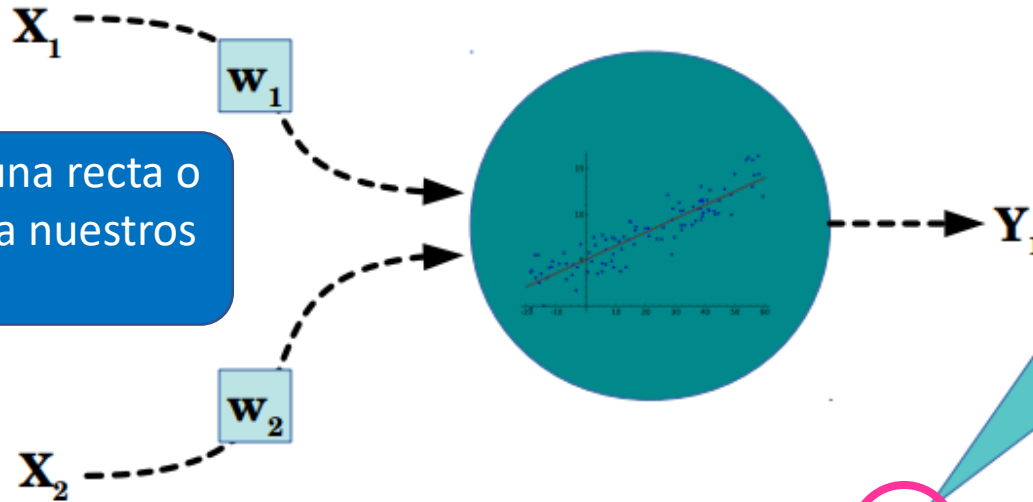
Utiliza todas las entradas para realizar una suma ponderada de ellos.

A que suena esto?  
Suena a Regresión lineal



El **término independiente** nos servía para mover la recta en vertical, i.e., en el eje y. Por tanto, el término independiente nos servía para subir/bajar la recta

## Función de la NN



Podemos ver como una recta o hiperplano se ajusta a nuestros datos

El término independiente, llamado **bias** o **sesgo**, nos permite el desplazamiento de la recta.

$$y = w_1x_1 + w_2x_2 + \mathbf{b}$$

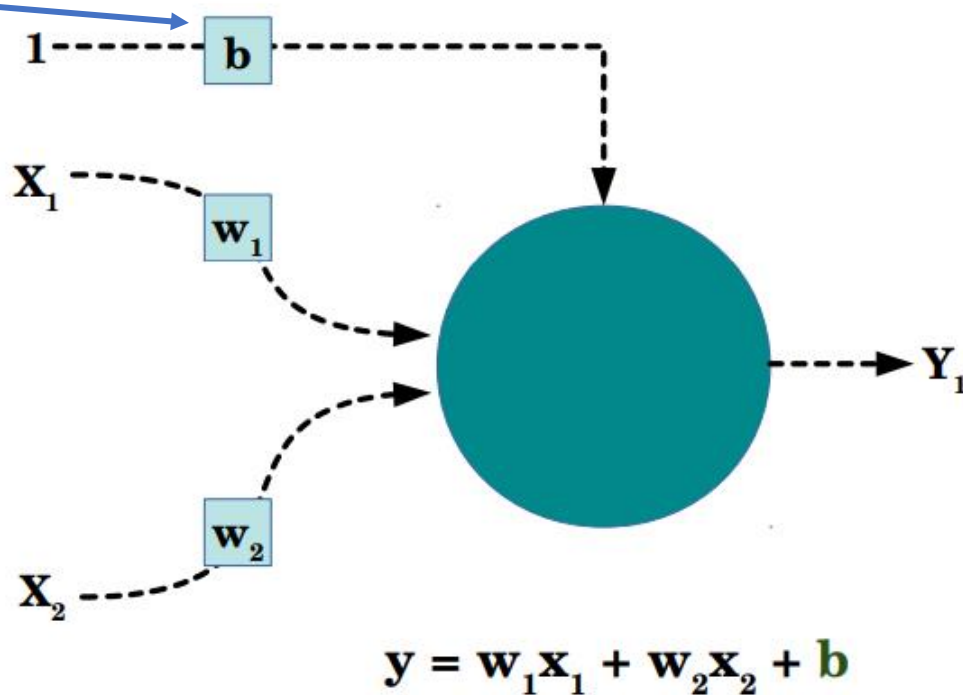
## Ejemplo:

1: estudio  
0: no estudio

1: estoy en casa  
0: no estoy en casa

(X1) (X2) **Estudiar + Estar en casa = Aprobar el examen**

Asignamos a nuestra neurona el termino independiente



valor



Estudiamos + estamos en casa	= Aprobamos el examen (1,1)
Estudiar + no estamos en casa	= No Aprobamos el examen (1,0)
No Estudiar + estamos en casa	= No Aprobamos el examen (0,1)
No Estudiar + no estamos en casa	= No Aprobamos el examen (0,0)

# Umbral

- Si un peso asignado a esa característica es menor o igual al umbral,  $Y=0$   $WX \leq \text{UMBRAL} \rightarrow Y = 0$
- Si un peso asignado a esa característica es mayor al umbral,  $Y=1$   $WX > \text{UMBRAL} \rightarrow Y = 1$

---

$$\text{BIAS} = -\text{UMBRAL}$$

El sesgo está supeditado al umbral en sentido negativo

## Esto implica

$$\text{BIAS} = -\text{UMBRAL}$$

---

$$WX + b \leq 0 \rightarrow Y = 0$$











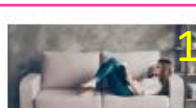

- Si el peso multiplicado por la característica + el sesgo es menor o igual a 0, entonces,  $Y=0$

$$WX + b > 0 \rightarrow Y = 1$$

- Si el peso multiplicado por la característica + el sesgo es mayor a 0, entonces,  $Y=1$

# Estudiar + Estar en casa = Aprobar el examen

$$WX + b \leq 0 \rightarrow Y = 0$$

$X_1$	$X_2$	Target	Y
 0	 0		?
 1	 0		?
 0	 1		?
 1	 1		?

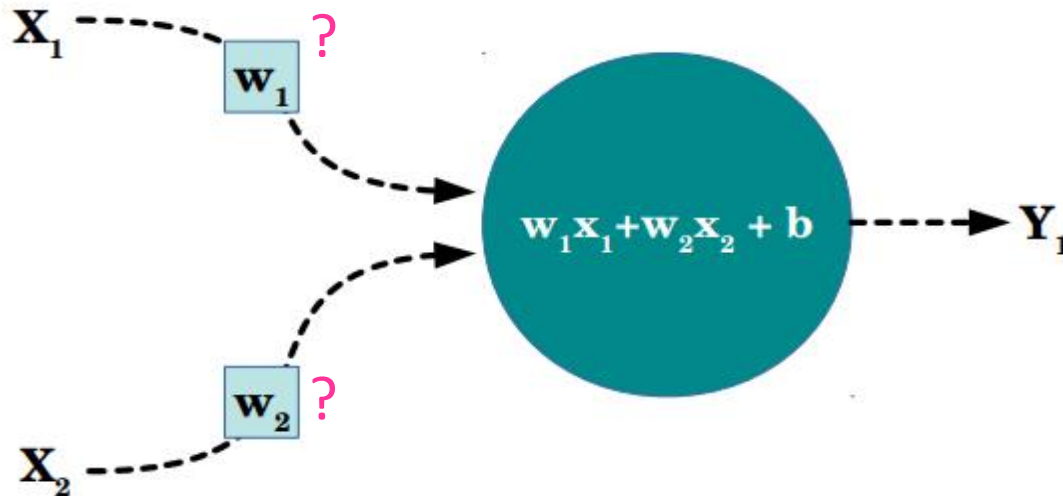
$$WX + b > 0 \rightarrow Y = 1$$

El resultado de la NN estará condicionado por el valor de los **parámetros**. Por tanto, tenemos que buscar esos pesos y bias.

Buscar combinaciones que cumplan esta característica

$$WX + b \leq 0 \rightarrow Y = 0$$

$$WX + b > 0 \rightarrow Y = 1$$















El objetivo es:

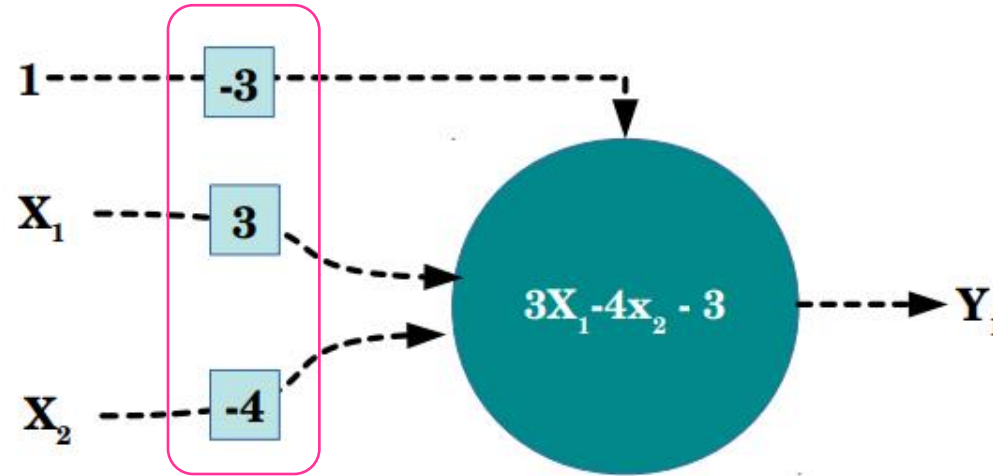
- $X_1=0$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=0$  y  $X_2=1$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=1$  tiene que ser mayor a 0



Entramos a un proceso de asignación de pesos buscando el objetivo.

Por ejemplo:

$X_1$	$X_2$	Target	Y
			-3
			0
			-7
			-4



Caso 1: si  $X_1=0$  y  $X_2=0$ : la ecuación da: -3 (cumple)

Caso 2: si  $X_1=1$  y  $X_2=0$ : la ecuación da: 0 (cumple)

Caso 3: si  $X_1=0$  y  $X_2=1$ : la ecuación da: -7 (Cumple)

Caso 4: si  $X_1=1$  y  $X_2=1$ : la ecuación da: -4 (no cumple)
















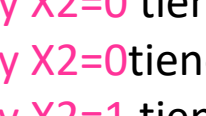
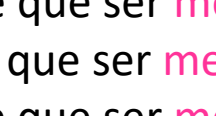

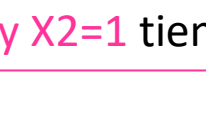
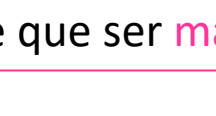




patrón

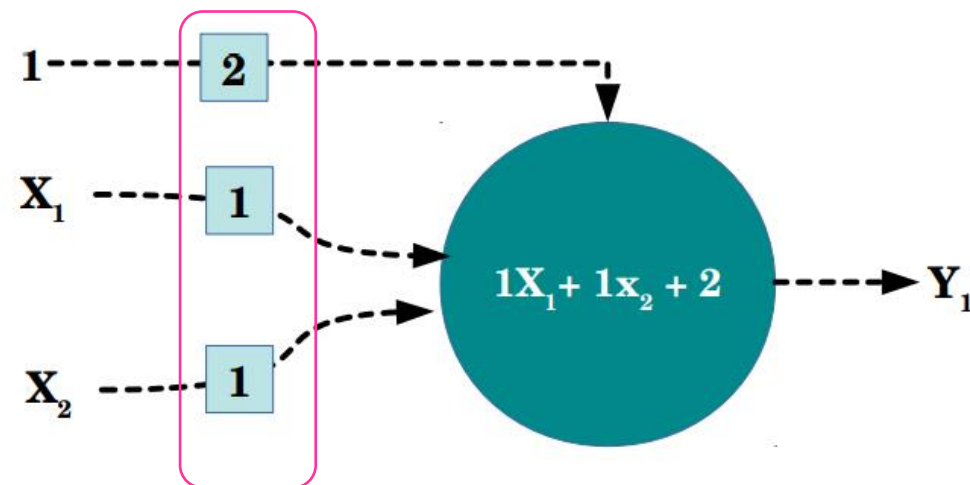
- $X_1=0$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=0$  y  $X_2=1$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=1$  tiene que ser mayor a 0

La configuración de parámetros no cumple el objetivo, tengo que configurar otros pesos.

Pasamos a otra iteración, y le llamaremos EPOCA, entonces pasamos a otra época y corregimos los pesos (arreglo multilayer perceptrón )

Con nuevos pesos, pasando a otra EPOCA

$X_1$	$X_2$	Target	$Y$
			2
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3
			3















Caso 1: si  $X_1=0$  y  $X_2=0$ : la ecuación da: 2 (No cumple)  
 Caso 2: si  $X_1=1$  y  $X_2=0$ : la ecuación da: 3 (No cumple)  
 Caso 3: si  $X_1=0$  y  $X_2=1$ : la ecuación da: 3 (No Cumple)  
 Caso 4: si  $X_1=1$  y  $X_2=1$ : la ecuación da: 4 (cumple)

patrón

- $X_1=0$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=0$  y  $X_2=1$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=1$  tiene que ser mayor a 0

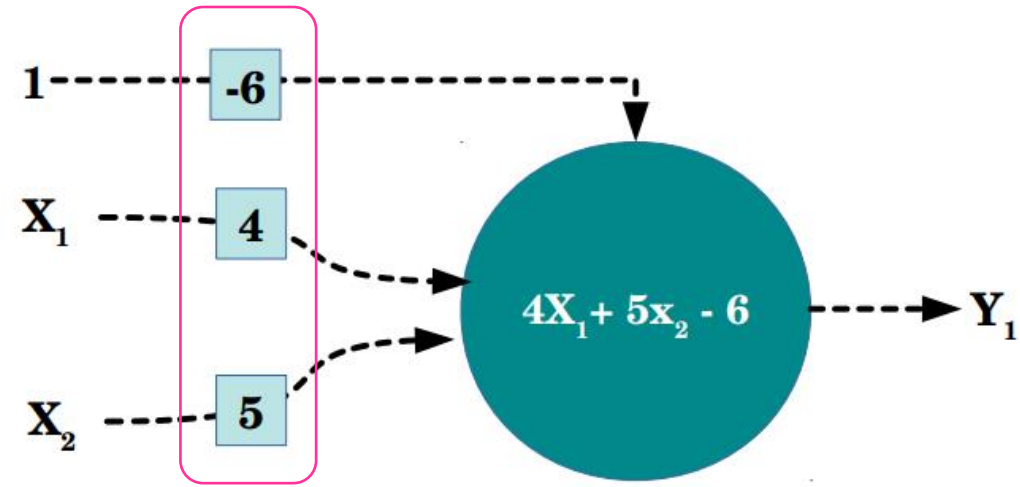


Con nuevos pesos, pasando a otra EPOCA

$X_1$	$X_2$	Target	Y
			-6
			-2
			-1
			3

patrón

- $X_1=0$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=0$  y  $X_2=1$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=1$  tiene que ser mayor a 0



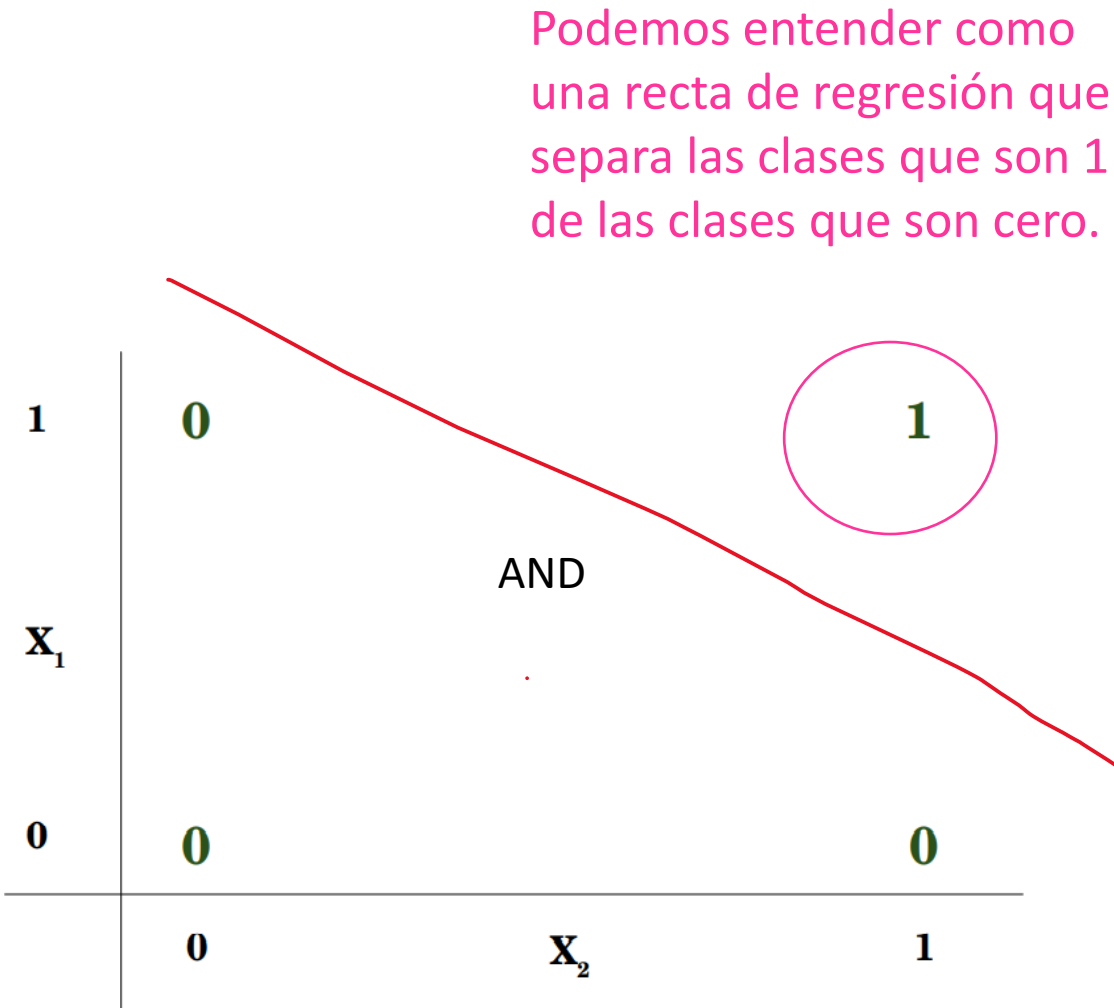
- Caso 1: si  $X_1=0$  y  $X_2=0$ : la ecuación da: -6 (Cumple)  
Caso 2: si  $X_1=1$  y  $X_2=0$ : la ecuación da: -2 (Cumple)  
Caso 3: si  $X_1=0$  y  $X_2=1$ : la ecuación da: -1 (Cumple)  
Caso 4: si  $X_1=1$  y  $X_2=1$ : la ecuación da: 3 (Cumple)

Tenemos una búsqueda optimizada

Para no estar en este problema, usaremos el Algoritmo back propagation

# En forma bidimensional

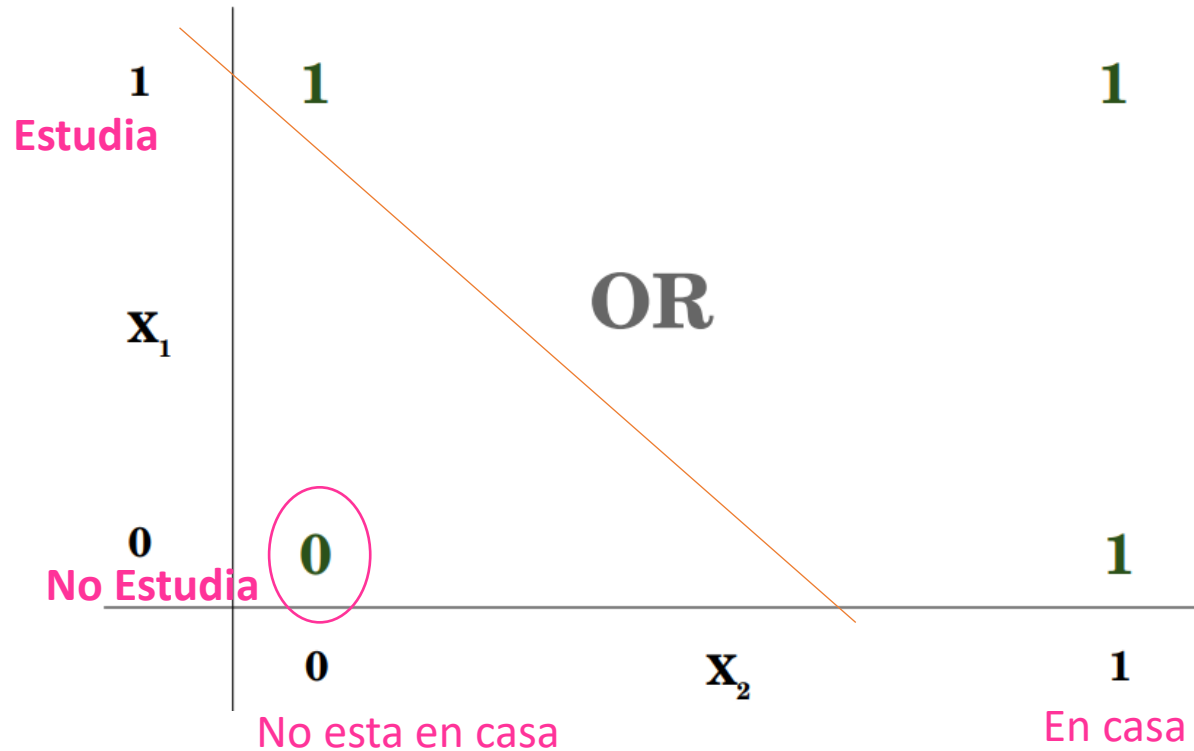
## Entender el procedimiento



Observamos que tenemos una **puerta lógica AND**

# Puerta lógica OR

Suponiendo



Ajustar los parámetros hasta que la red neuronal de estas opciones

$$WX + b \leq 0 \rightarrow Y = 0$$













$$WX + b > 0 \rightarrow Y = 1$$

patrón

- $x_1=0$  y  $x_2=0$  tiene que ser menor o igual a 0
- $x_1=1$  y  $x_2=0$  tiene que ser mayor a 0
- $x_1=0$  y  $x_2=1$  tiene que ser mayor a 0
- $x_1=1$  y  $x_2=1$  tiene que ser mayor a 0

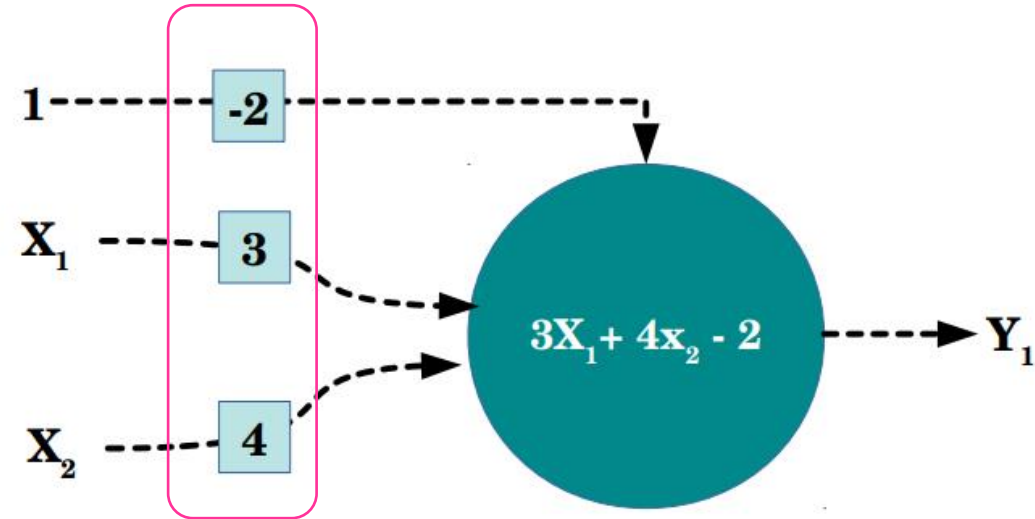
# Solamente no va a aprobar cuando no estudia y no este en casa

Probando con, -2,3 y 4 ponderaciones de entrada

$X_1$	$X_2$	Target	Y
			-2
			1
			2
			5

patrón

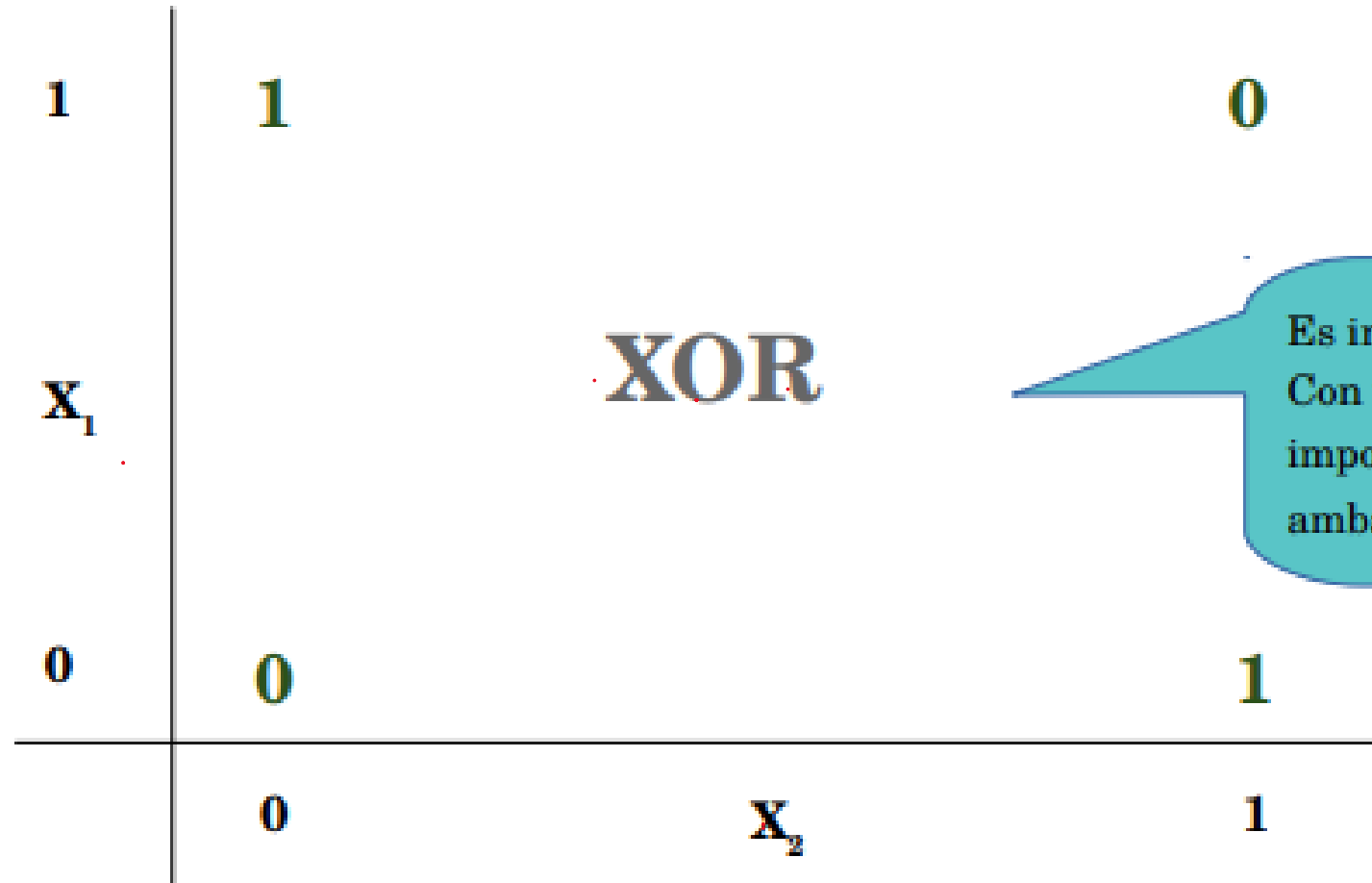
- $X_1=0$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=0$  tiene que ser mayor a 0
- $X_1=0$  y  $X_2=1$  tiene que ser mayor a 0
- $X_1=1$  y  $X_2=1$  tiene que ser mayor a 0



Tiene que salir positivo cuando va a aprobar y negativo cuando no va a aprobar

- Caso 1: si  $X_1=0$  y  $X_2=0$ : la ecuación da: -2 (Cumple)  
Caso 2: si  $X_1=1$  y  $X_2=0$ : la ecuación da: 1 (Cumple)  
Caso 3: si  $X_1=0$  y  $X_2=1$ : la ecuación da: 2 (Cumple)  
Caso 4: si  $X_1=1$  y  $X_2=1$ : la ecuación da: 5 (Cumple)

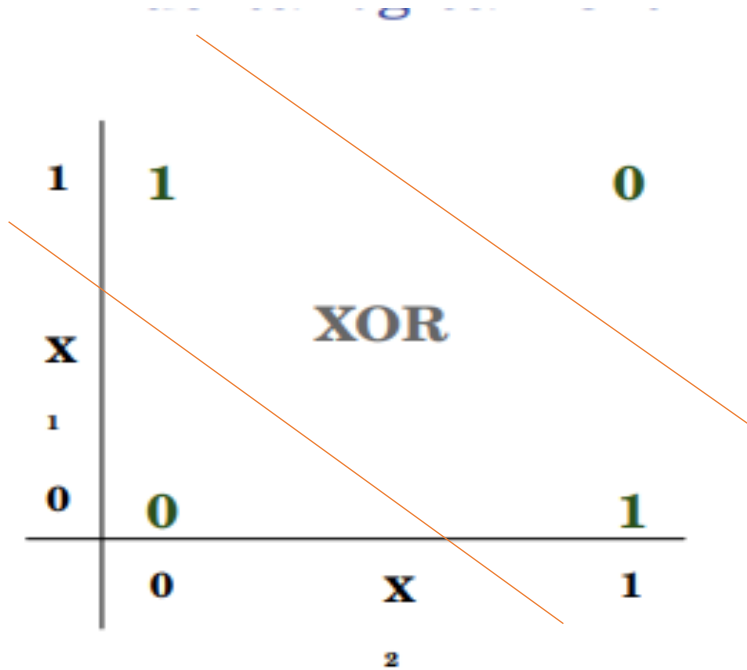
## Puerta lógica XOR



Es imposible modelar el problema  
Con una única neurona, i.e., es  
imposible **separar linealmente**  
ambas clases

Entonces usaremos dos neuronas

## Asignación de pesos buscando el objetivo.

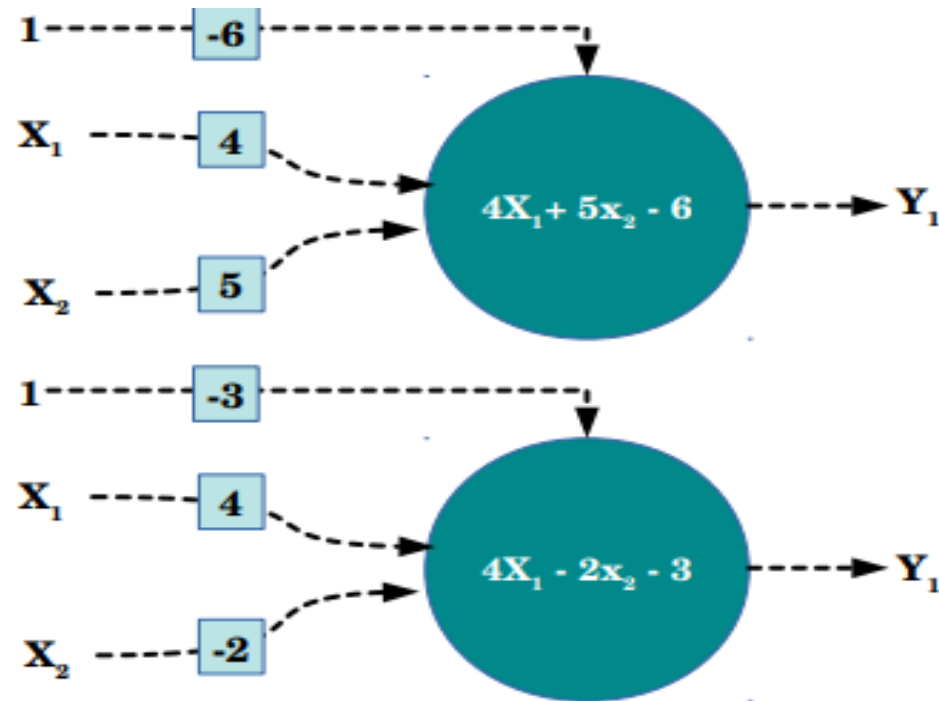


patrón

- $X_1=0$  y  $X_2=0$  tiene que ser menor o igual a 0
- $X_1=1$  y  $X_2=0$  tiene que ser mayor a 0
- $X_1=0$  y  $X_2=1$  tiene que ser mayor a 0
- $X_1=1$  y  $X_2=1$  tiene que ser menor o igual a 0

$$WX + b \leq 0 \rightarrow Y = 0$$

$$WX + b > 0 \rightarrow Y = 1$$



Tenemos  
que  
combinar  
varias  
neuronas

Caso 1: si  $X_1=0$  y  $X_2=0$ : la ecuación da: -6 (Cumple)

Caso 2: si  $X_1=1$  y  $X_2=0$ : la ecuación da: -2 (no Cumple)

Caso 3: si  $X_1=0$  y  $X_2=1$ : la ecuación da: -1 (no Cumple)

Caso 4: si  $X_1=1$  y  $X_2=1$ : la ecuación da: 3 (no Cumple)

Caso 1: si  $X_1=0$  y  $X_2=0$ : la ecuación da: -3 (Cumple)

Caso 2: si  $X_1=1$  y  $X_2=0$ : la ecuación da: 1 (Cumple)

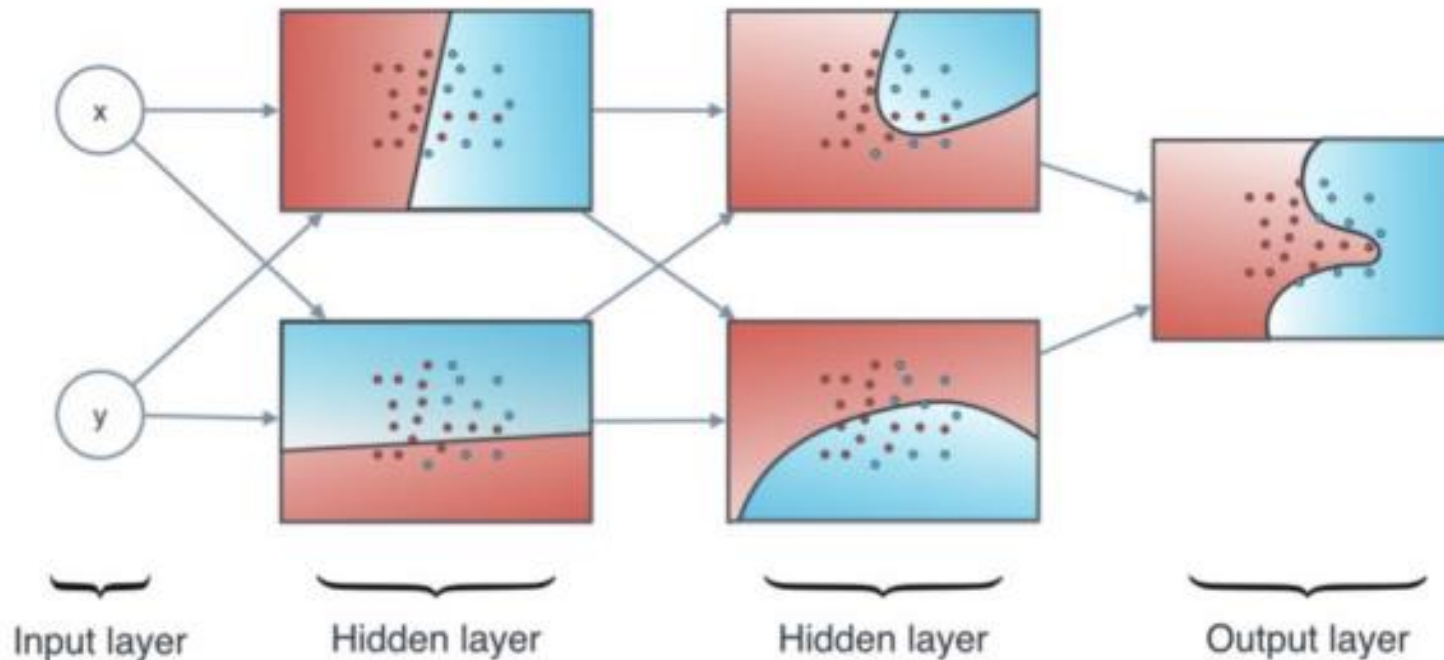
Caso 3: si  $X_1=0$  y  $X_2=1$ : la ecuación da: -5 (no Cumple)

Caso 4: si  $X_1=1$  y  $X_2=1$ : la ecuación da: -1 (no Cumple)



## Arquitectura de redes neuronales

- Son la base de las **redes profundas**, actualmente muy utilizadas con mucho éxito.
- Cada **capa extrae características** cada vez más complejas

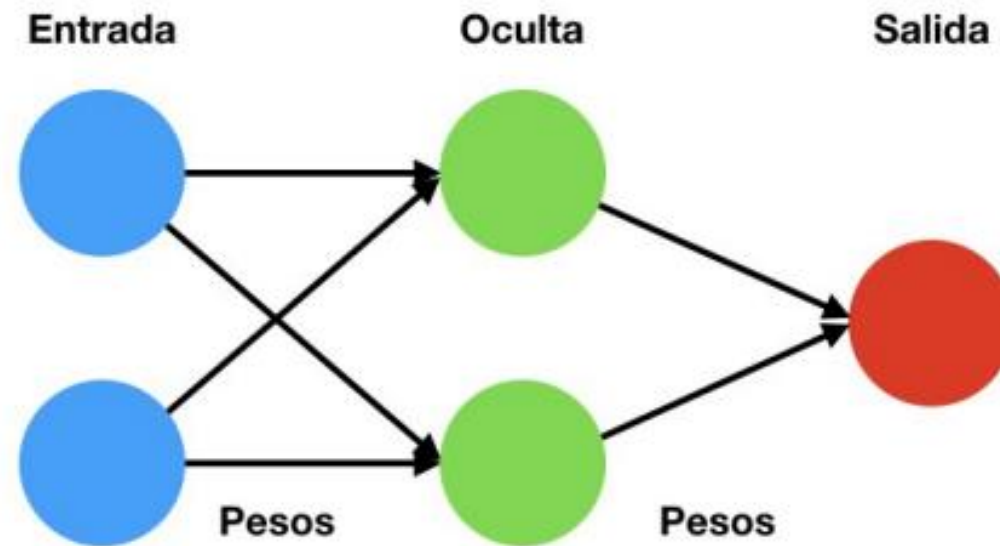


Mas capas y mas neuronas para resolver problemas complejos, a esto se llama perceptrón multicapa

# Perceptrón multicapa

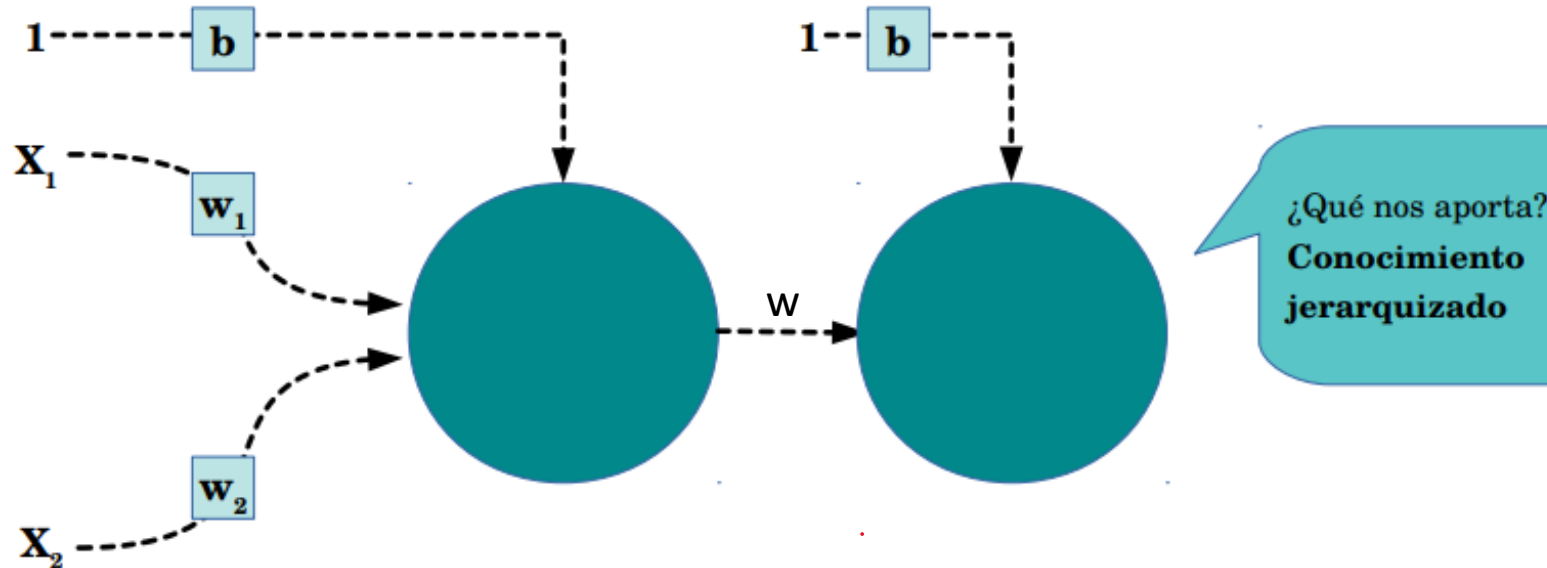
## Definición

- Una neurona no puede representar toda la información
  - Problema de linealidad como XOR
- Necesitamos una agrupación de neuronas



## Secuencialidad de neuronas

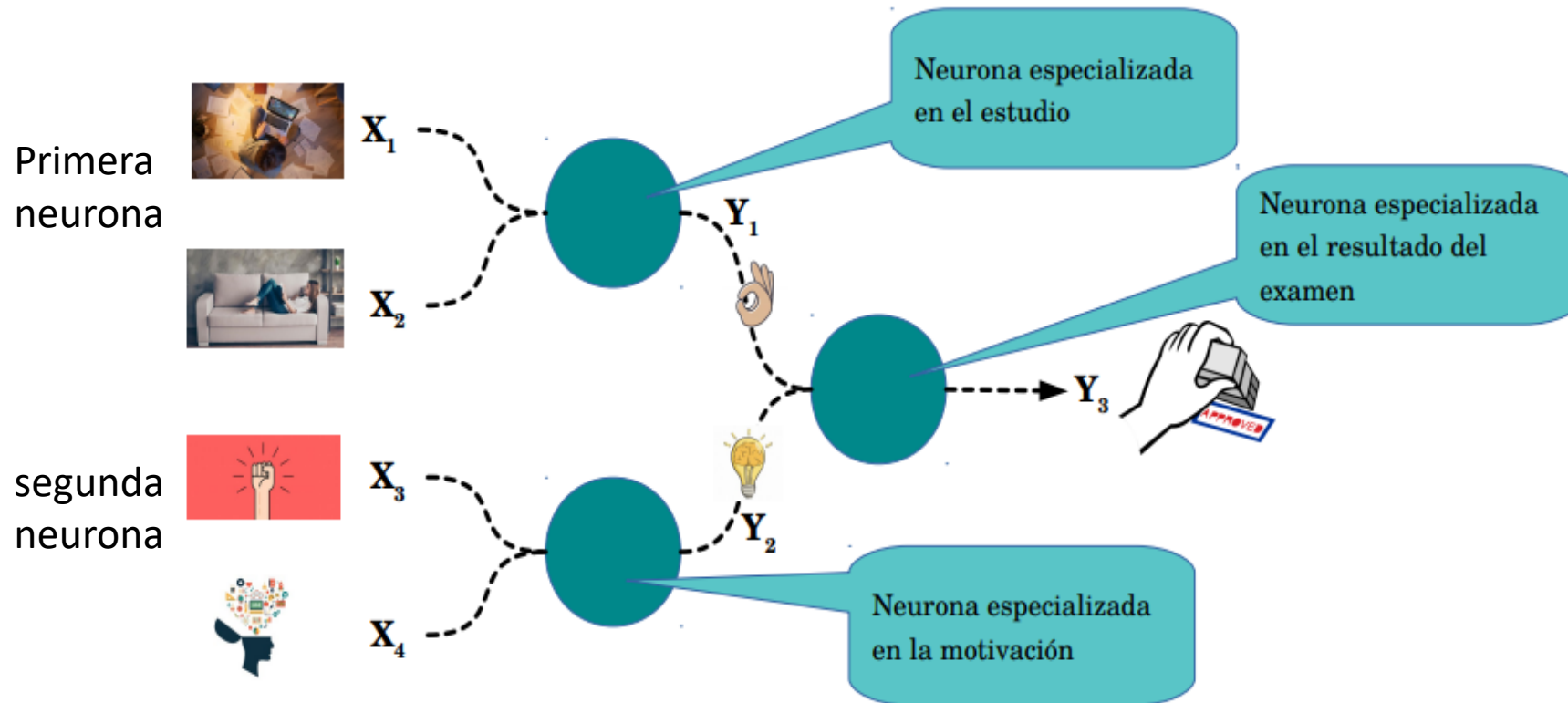
Lo que tenemos sería una secuencialidad de neuronas.



Las primeras capas solucionan estructuras mas simples  
Y las otras solucionan estructuras cada vez mas complejas

# Conocimiento jerarquizado

Ejemplo:

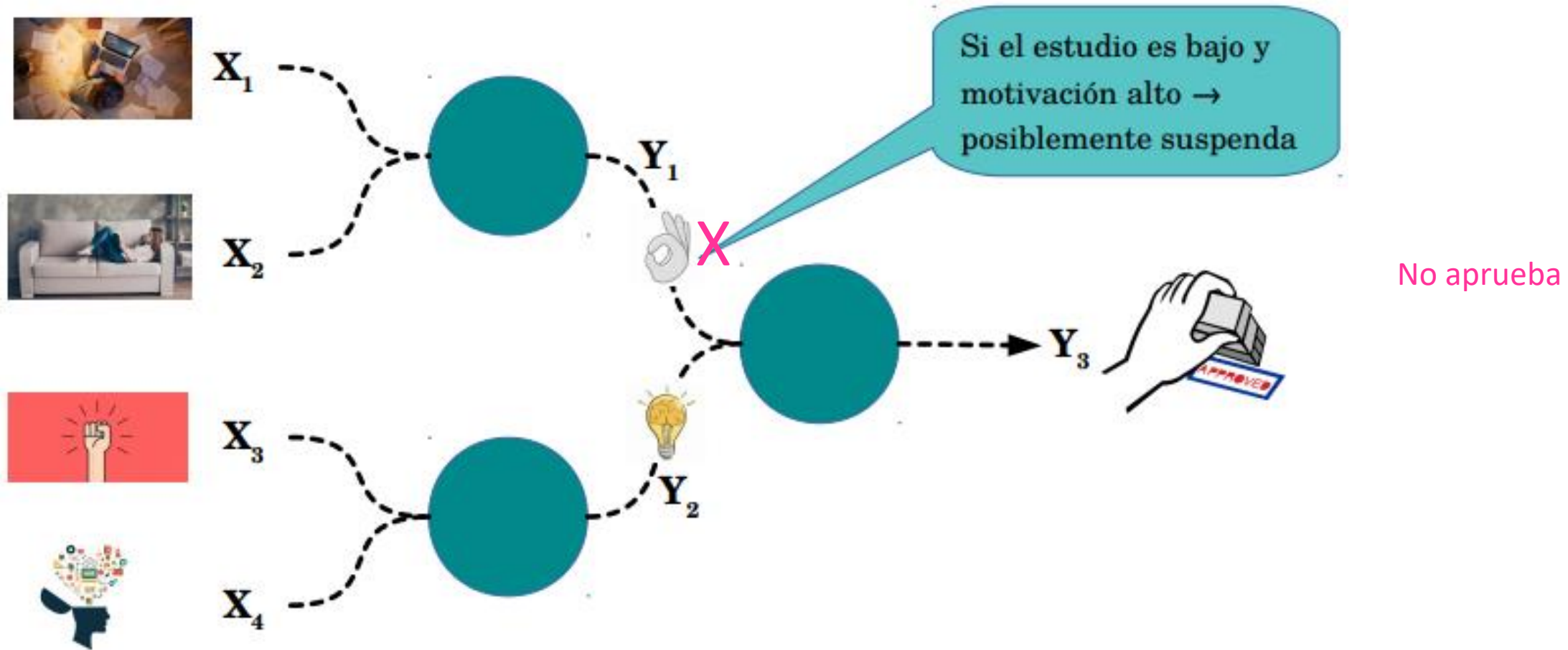


1) Si estudia y esta en casa = aprueba.

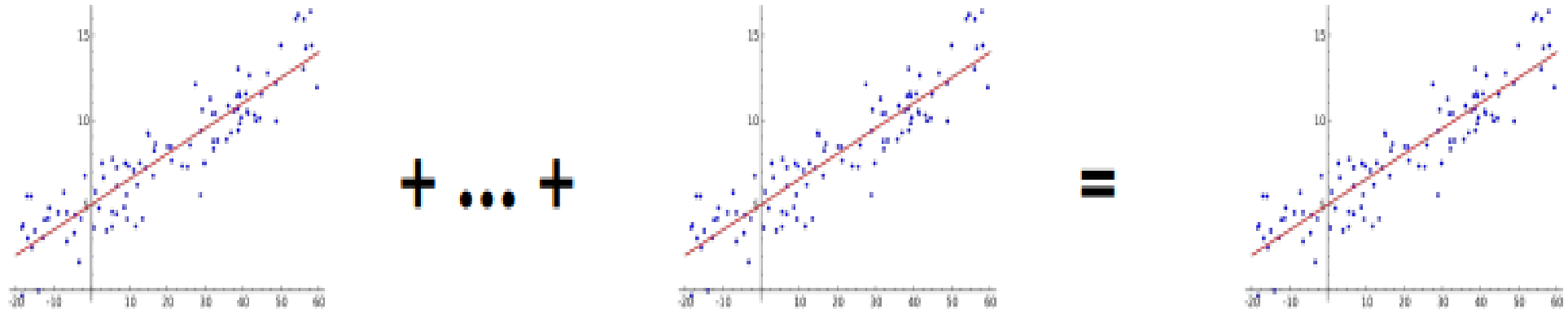
2) Si esta motivado y le gusta la asignatura = aprueba

3) Si 1 y 2 = aprueba

## Ejemplo:



## Problema



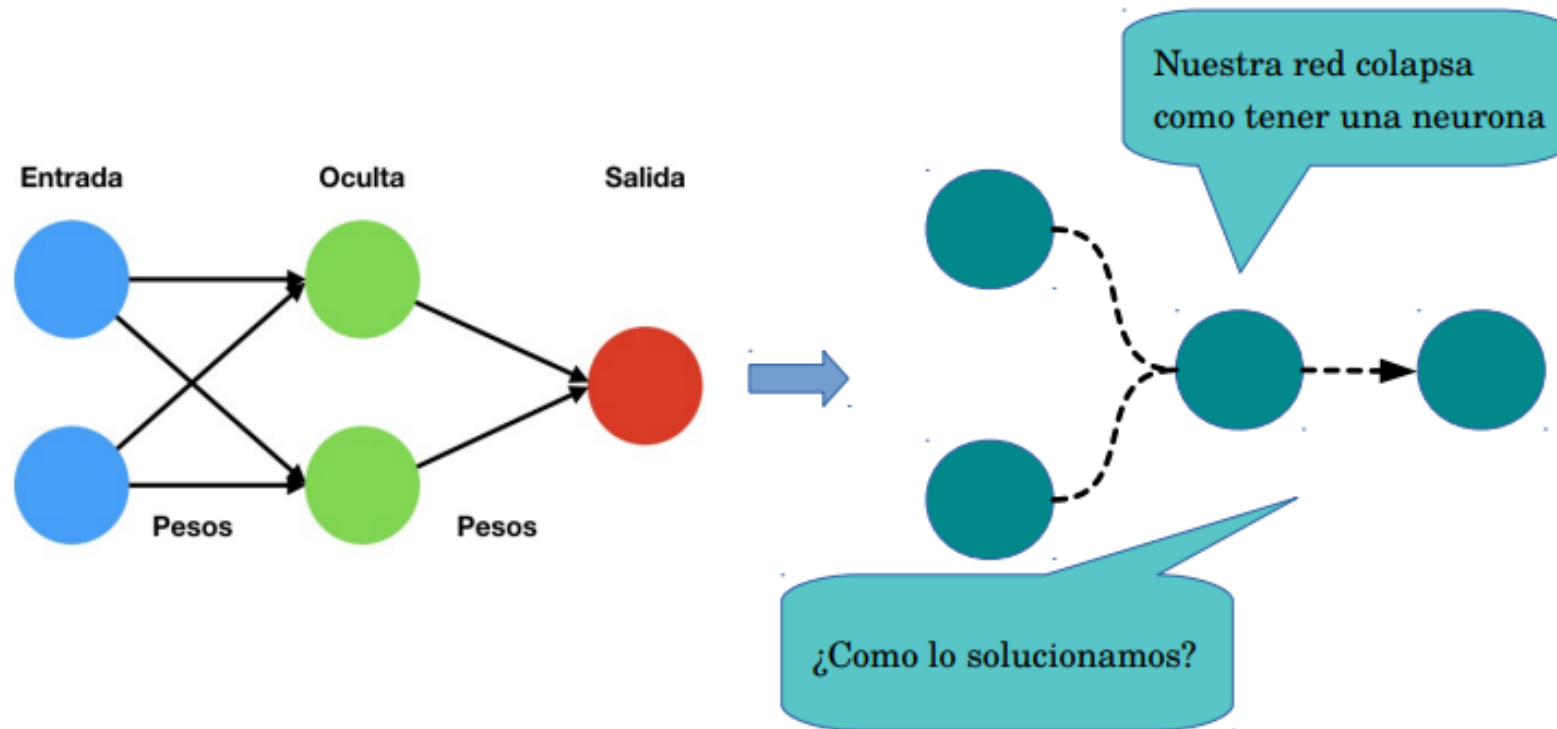
Si concatenamos regresiones lineales esto es = a otra regresión lineal  
(Problema de linealidad)

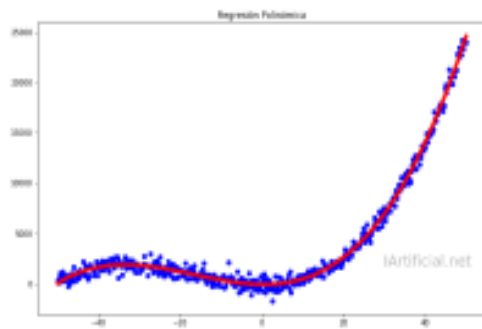
Como representamos como una red neuronal



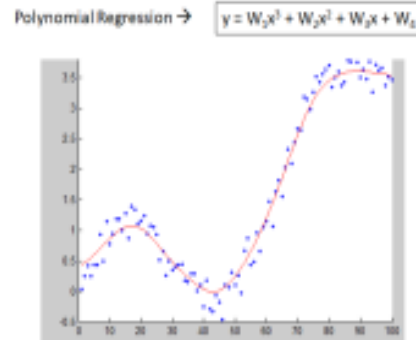
Si cada neurona esta representada por una regresión lineal, y la solución es otra función lineal, entonces en este ejemplo el sistema colapsa aunque pongamos 1000 capas. Se comporta como si tuviéramos una sola neurona.

La solución seria distorsionar las funciones que están en las neuronas

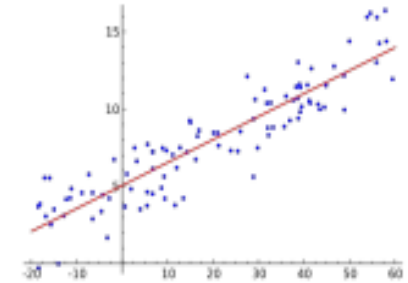




+ ... +



$\neq$



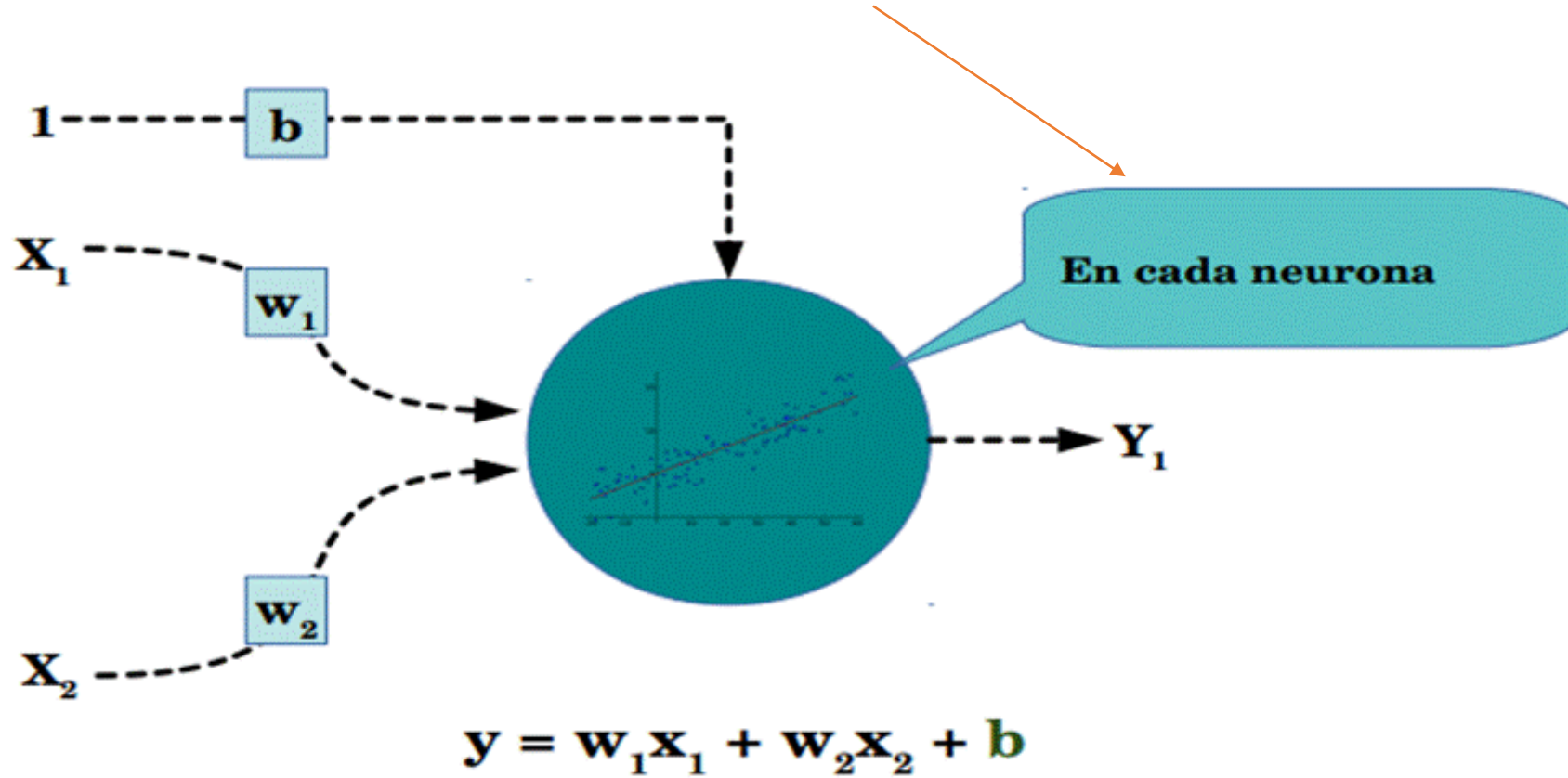
Así el resultado **no es Lineal**

Se distorsiona la linealidad..... Así ya no tenemos una salida lineal  
Entonces..

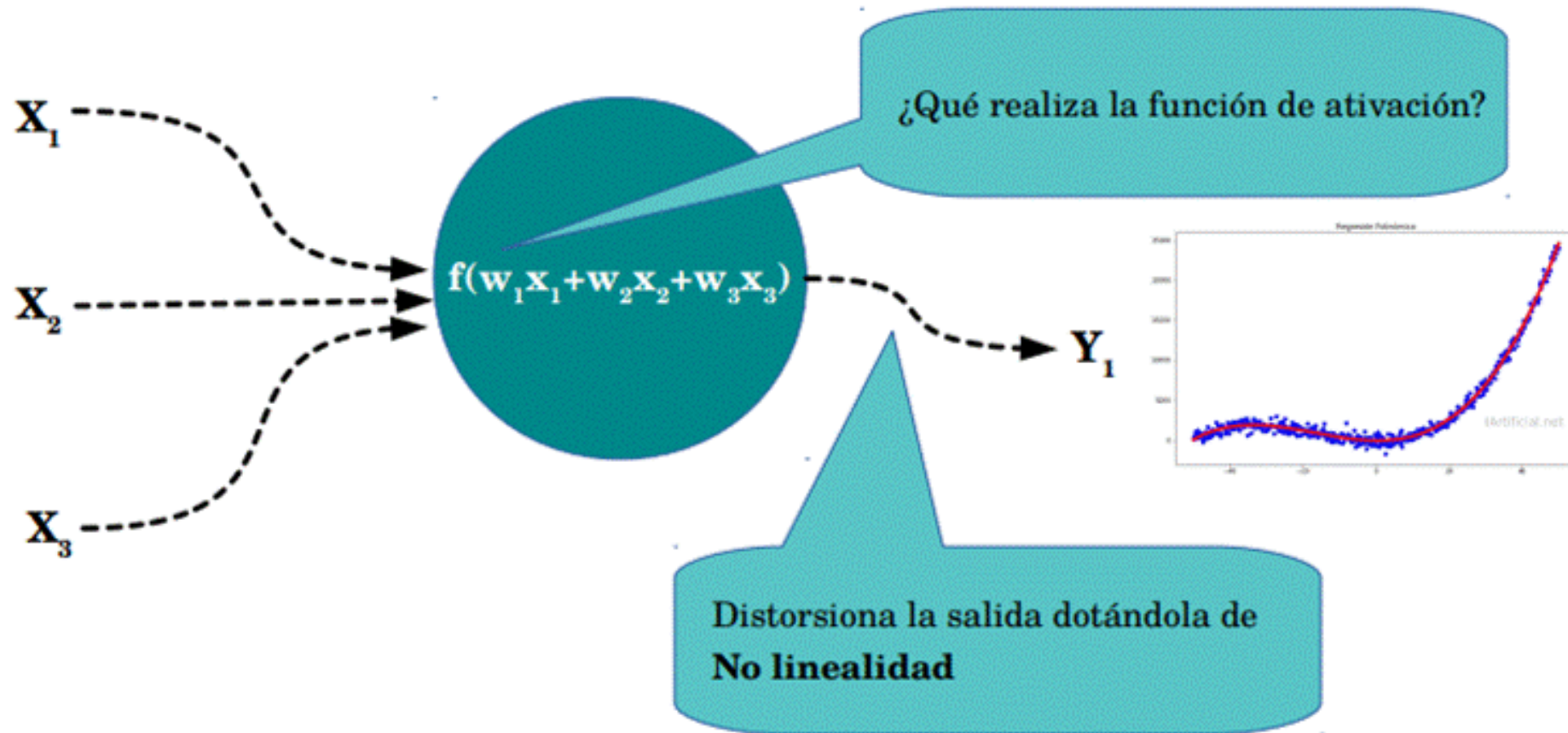
Para distorsionar NECESITAMOS **funciones de activación**

## FUNCION DE ACTIVACION

¿Donde se encuentra la función de activación?



## ¿Como la representamos la función de activación?



## EN EL EJEMPLO:







**Estudiar + Estar en casa = Aprobar el examen**

$$WX \leq \text{UMBRAL} \rightarrow Y = 0$$

$$WX > \text{UMBRAL} \rightarrow Y = 1$$

---

$$\text{BIAS} = -\text{UMBRAL}$$

	1	0
$X_1$		
$X_2$		
$Y_1$		

Teníamos una función de activación binaria

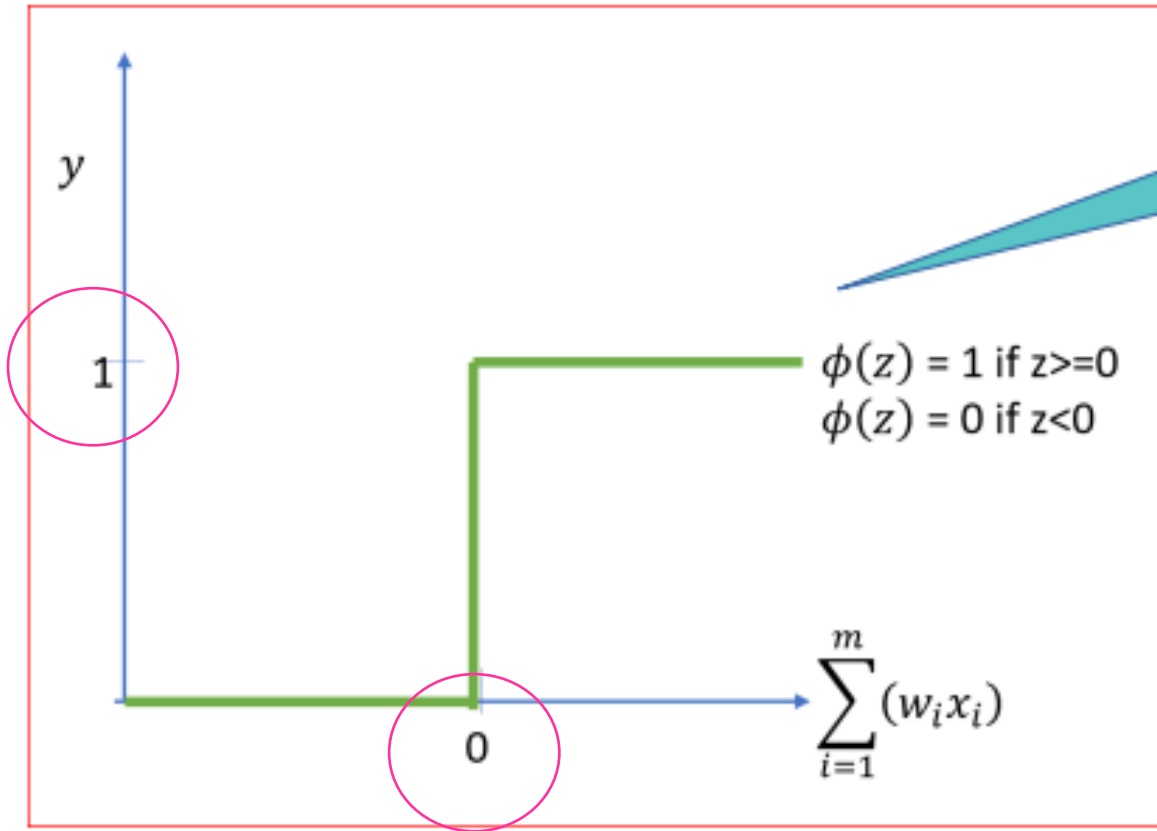
$$WX \leq \text{UMBRAL} \rightarrow Y = 0$$

$$WX > \text{UMBRAL} \rightarrow Y = 1 \quad \longrightarrow \quad f(WX) \rightarrow Y = \{0, 1\}$$

---

$$\text{BIAS} = -\text{UMBRAL}$$

## Función de activación Binaria

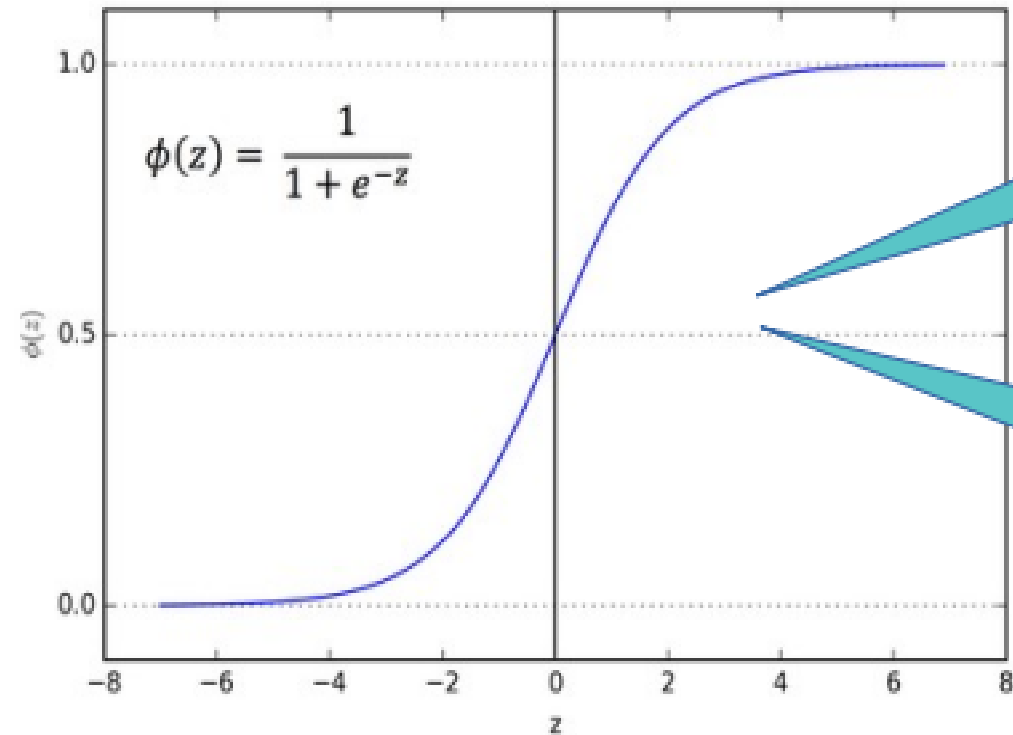


Dependiendo del umbral es 0 ó 1  
Problema lineal  $\rightarrow$  no nos interesa

$\leftarrow \mathbf{f(WX)} \rightarrow \mathbf{Y = \{0, 1\}}$



# Función de activación Sigmoide

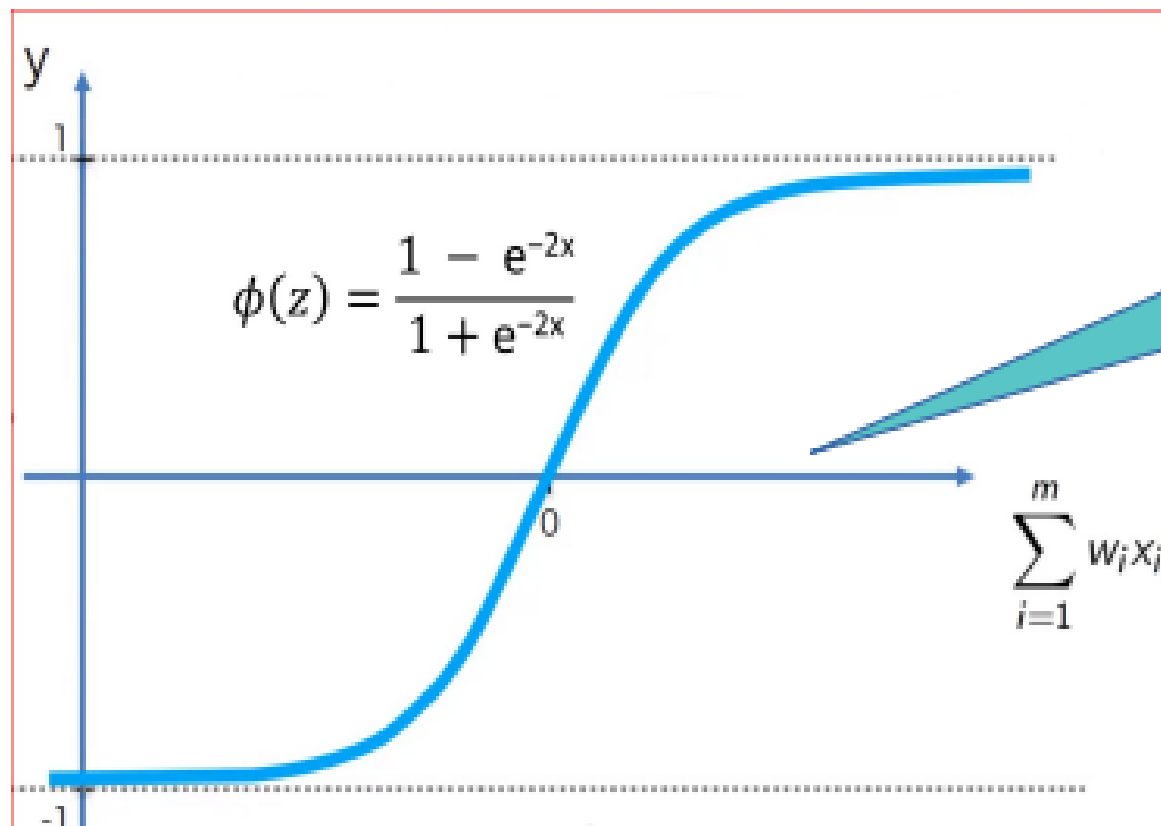


Los valores muy **grandes** se saturan en 1  
Los valores muy **pequeños** se saturan en 0

Por tanto, nos permite:

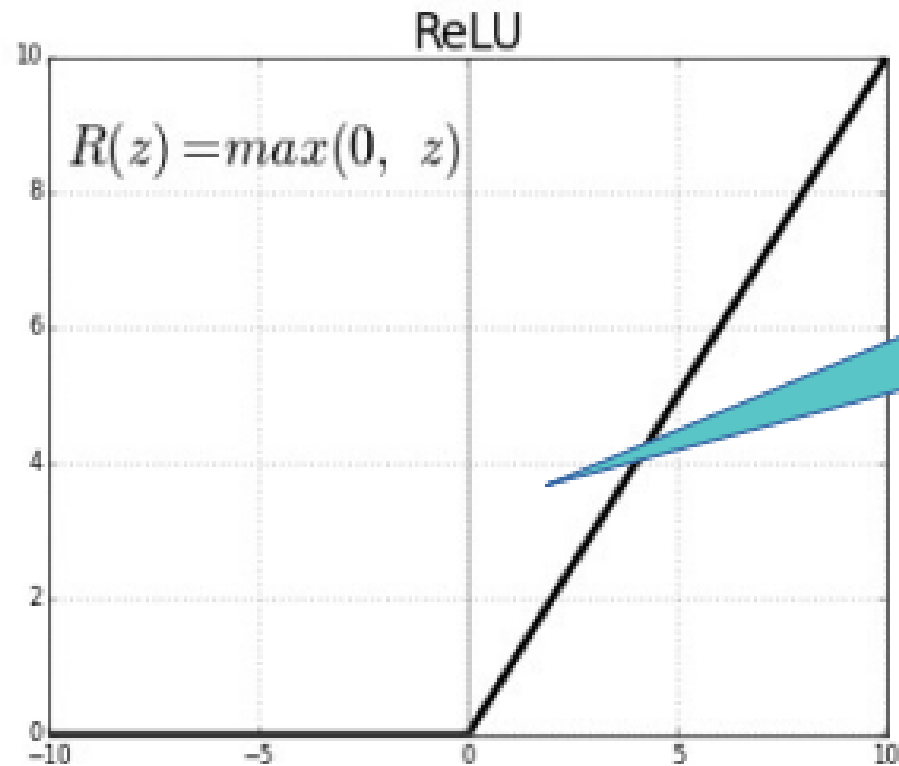
- 1) **Distorsionar** la señal
- 2) Representar **probabilidades** entre 0 y 1

## Función de activación Tangente hiperbólica (TANH)

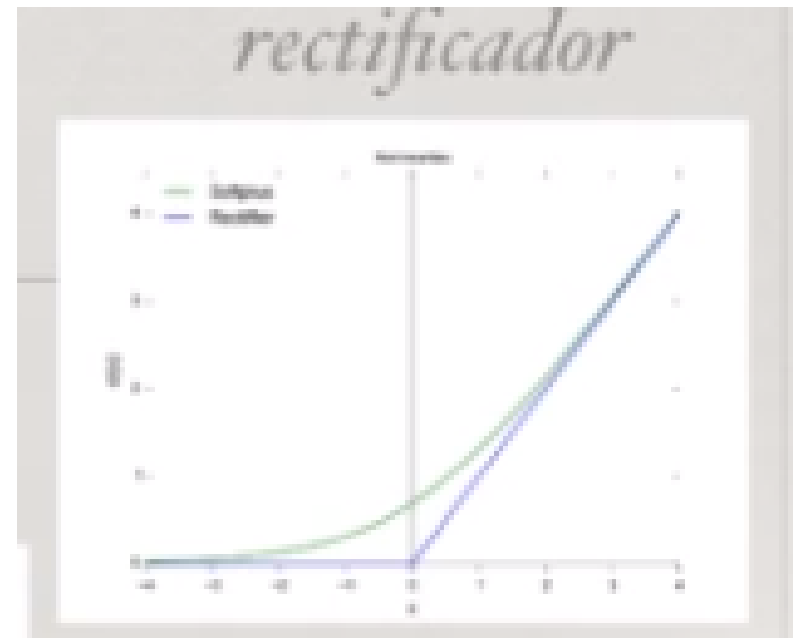


Parecida a la Sigmoide pero con un rango de -1 a 1

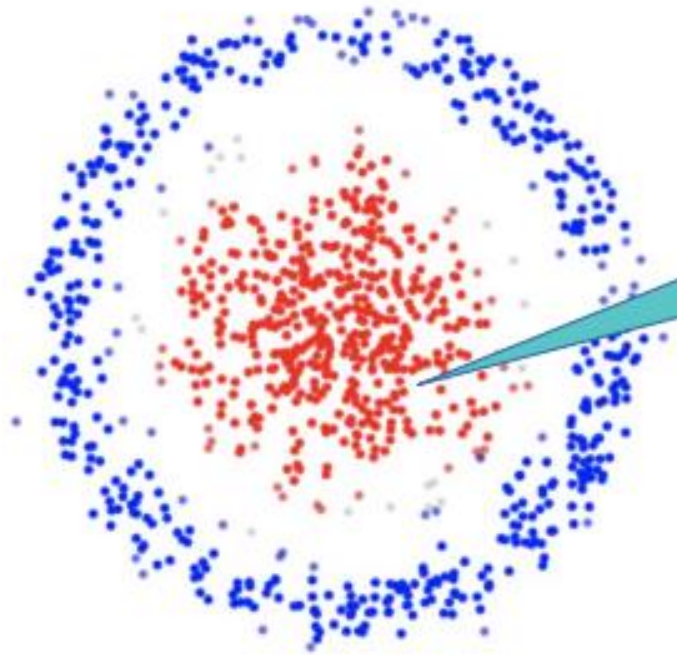
# Función de activación Rectificada Lineal (ReLU)



**Función lineal** cuando es positiva  
**Constante a 0** cuando el valor de entrada es negativo



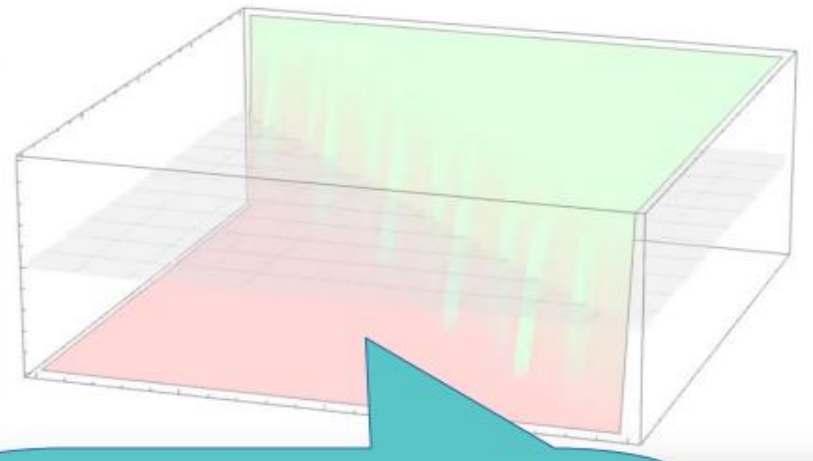
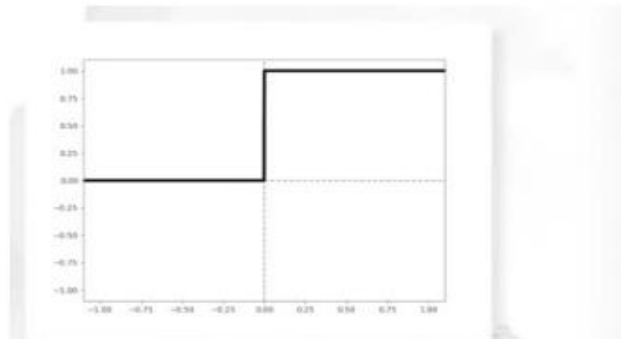
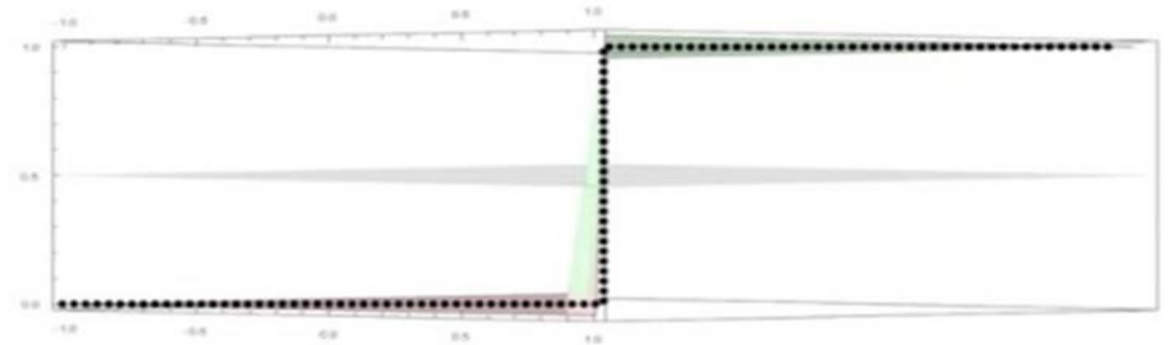
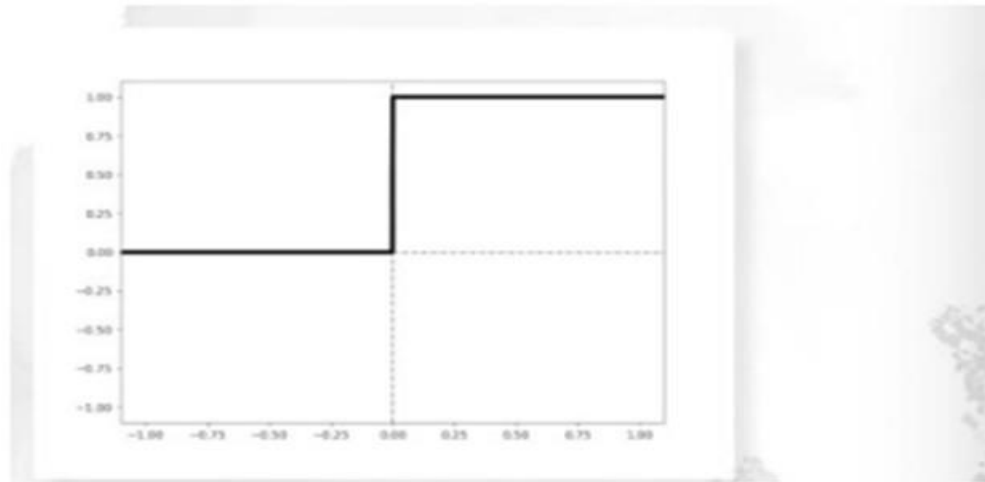
## Cómo opera el multi layer perceptrón (MLP)



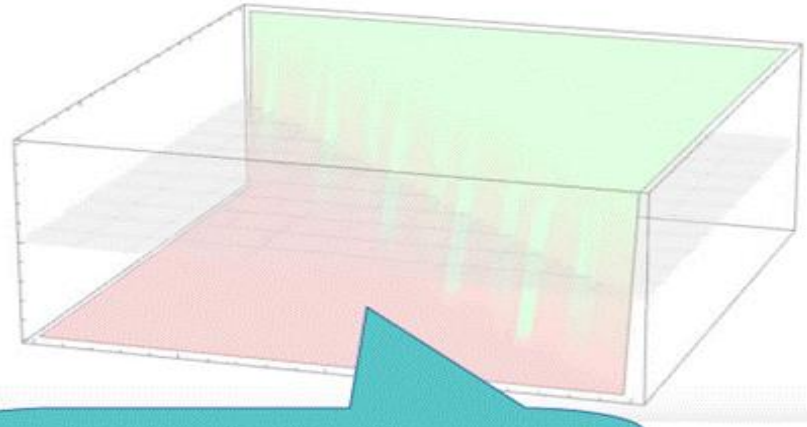
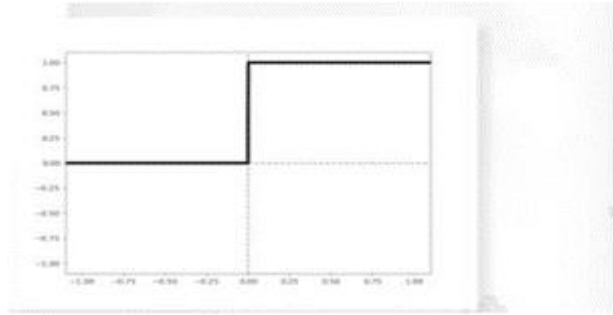
¿Cómo ver geoméricamente el efecto de una función de activación?

No es separable linealmente

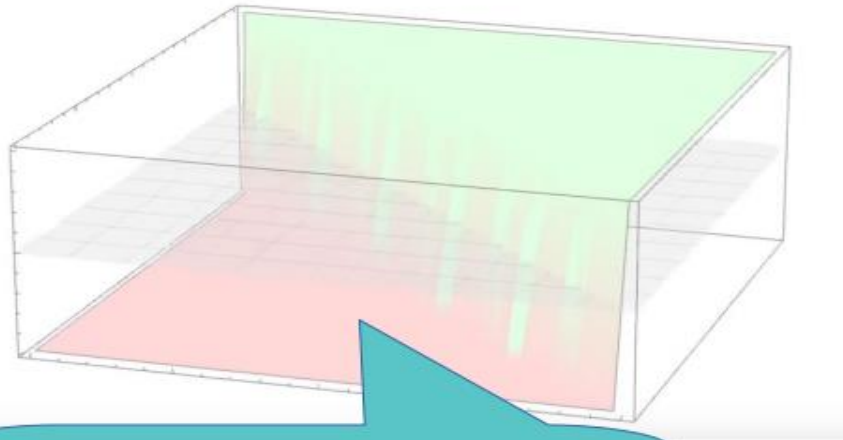
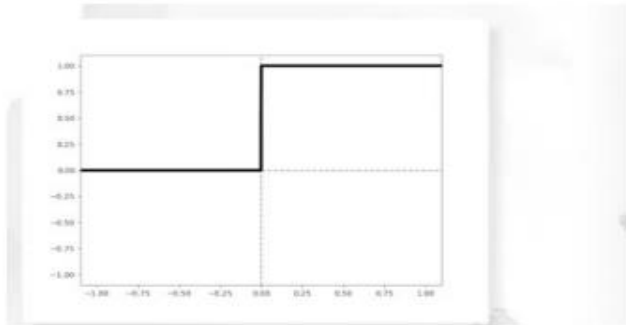
# Función de activación binaria



Todo lo que está encima del hiperplano será una clase y lo que está debajo otra



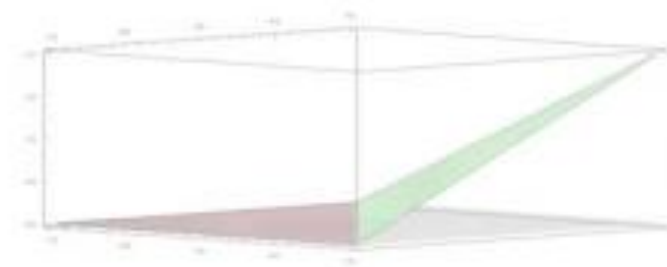
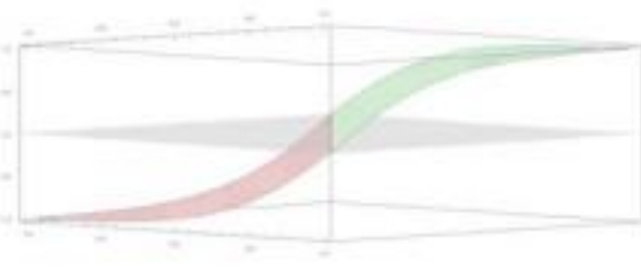
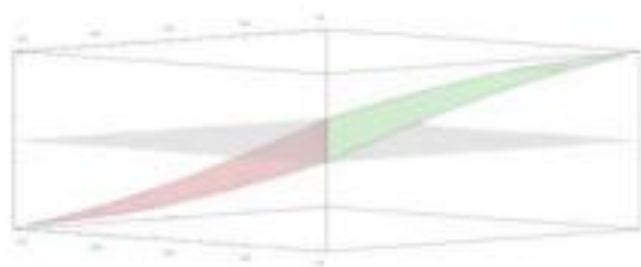
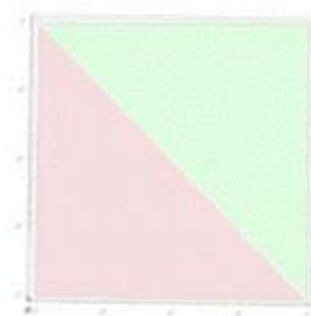
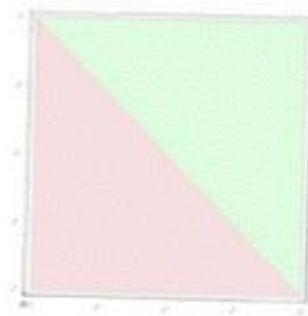
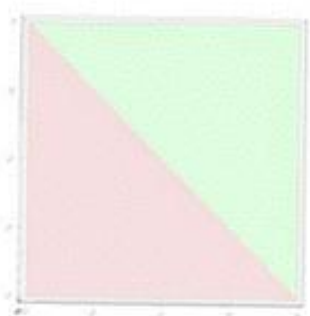
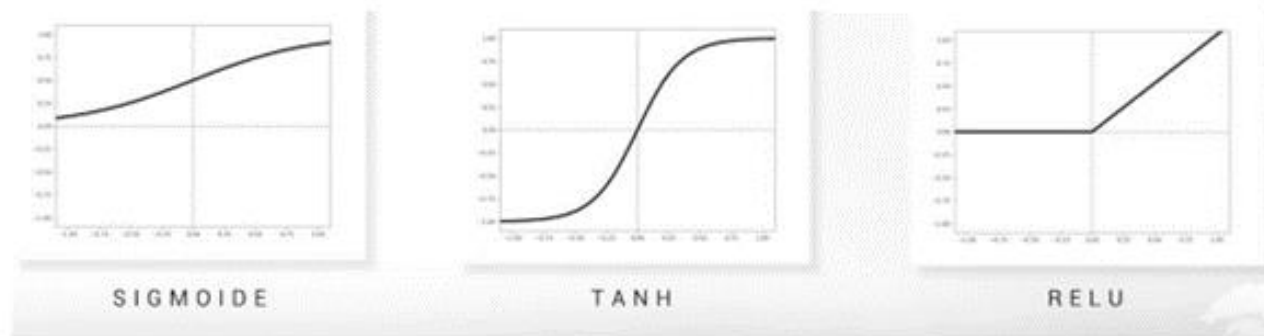
Todo lo que está encima del hiperplano será una clase y lo que está debajo otra



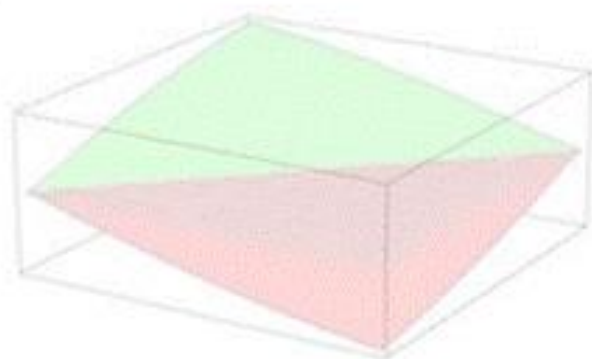
Necesitamos distorsionar la señal para acomodar el problema inicial



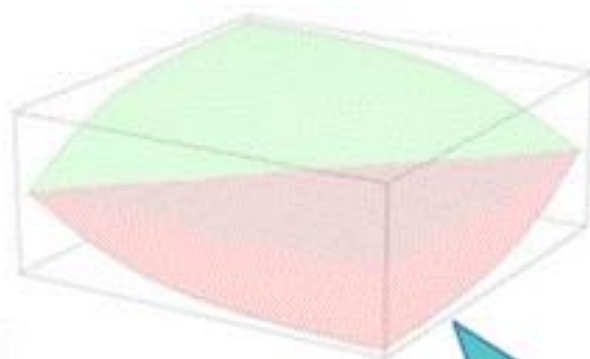
## Representar Geométricamente otras funciones de activación



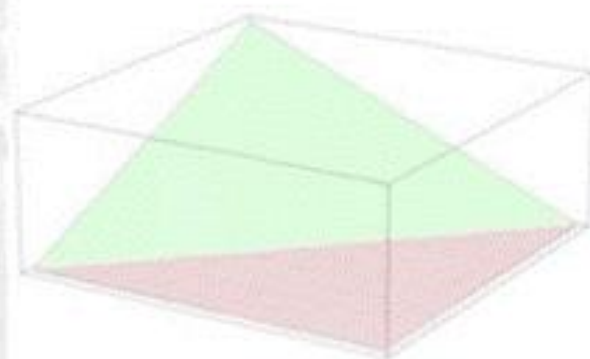
SIGMOIDE



TANH

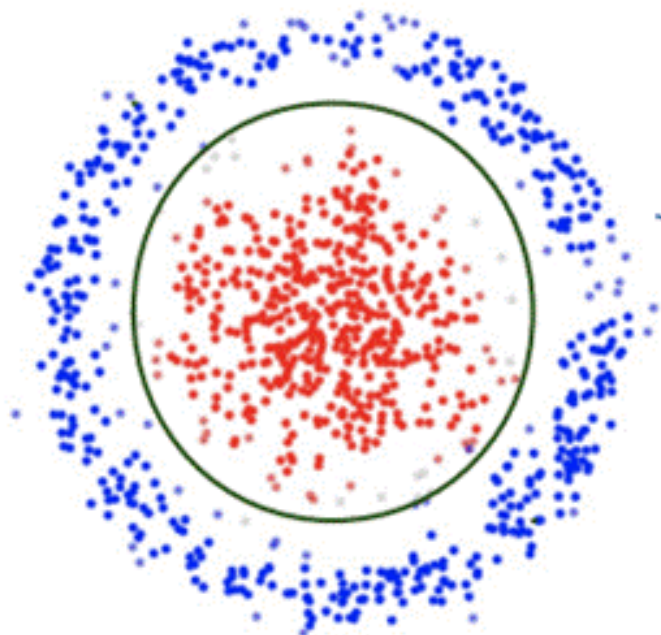


RELU

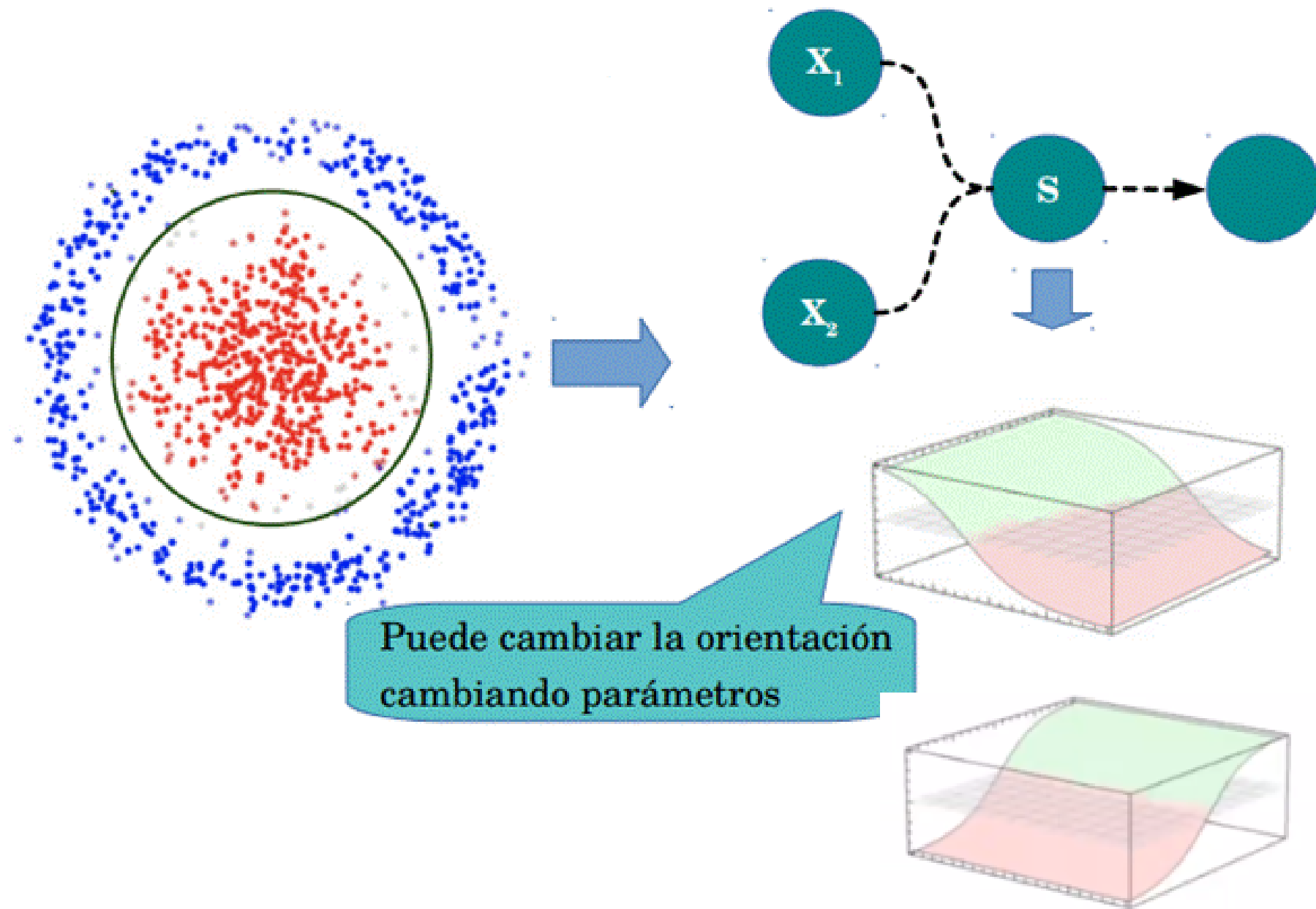


Pero la frontera no deja de ser una  
Línea recta, ¿Cómo resolvemos el problema?

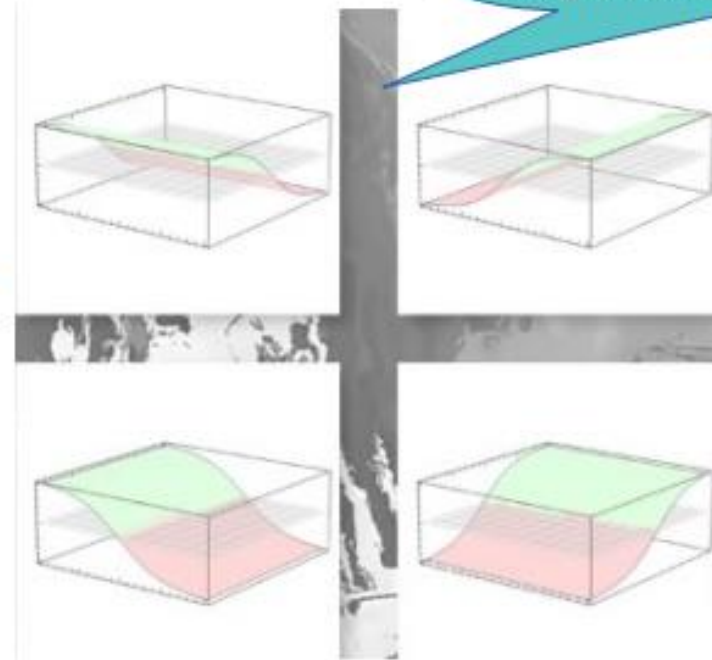
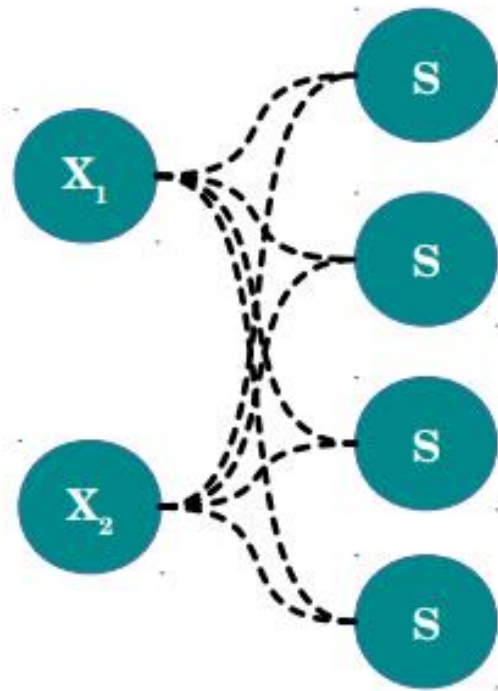
## Encontrar el hiperplano



¿Cómo trazar esa frontera?

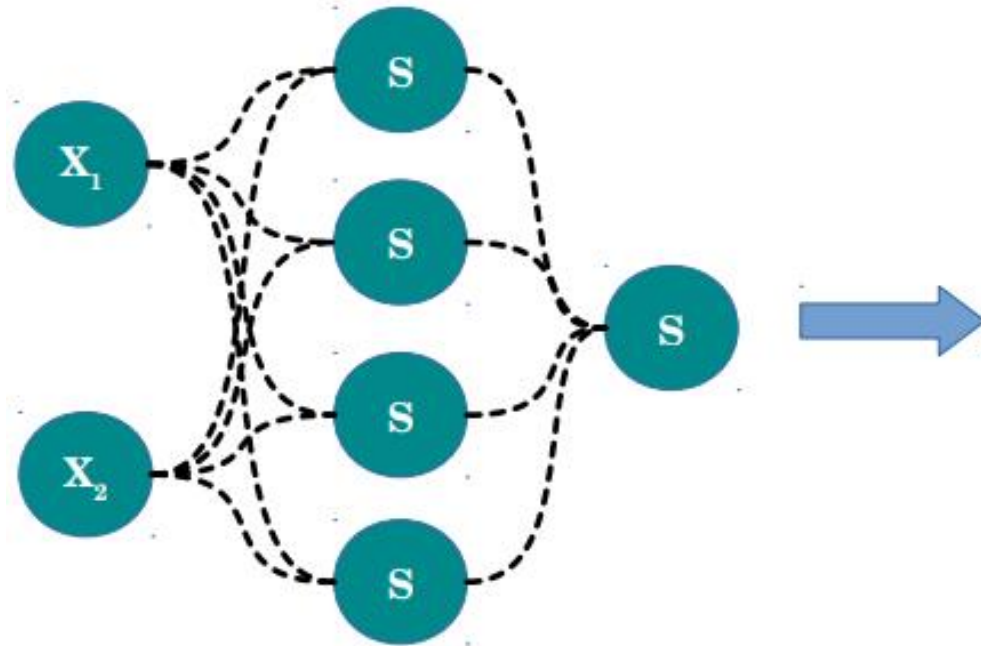


## Aumentamos las neuronas

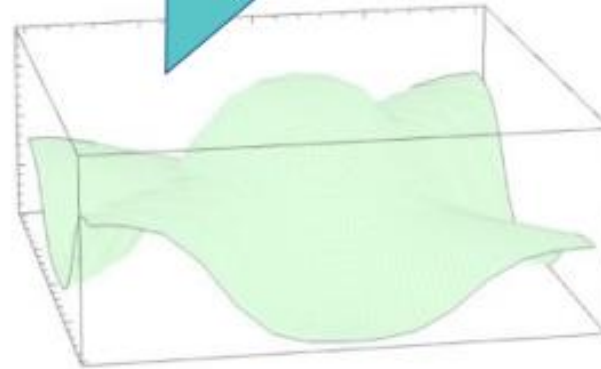


Pero cada una tiene una  
orientación diferente  
Las juntamos...

## Segunda capa oculta

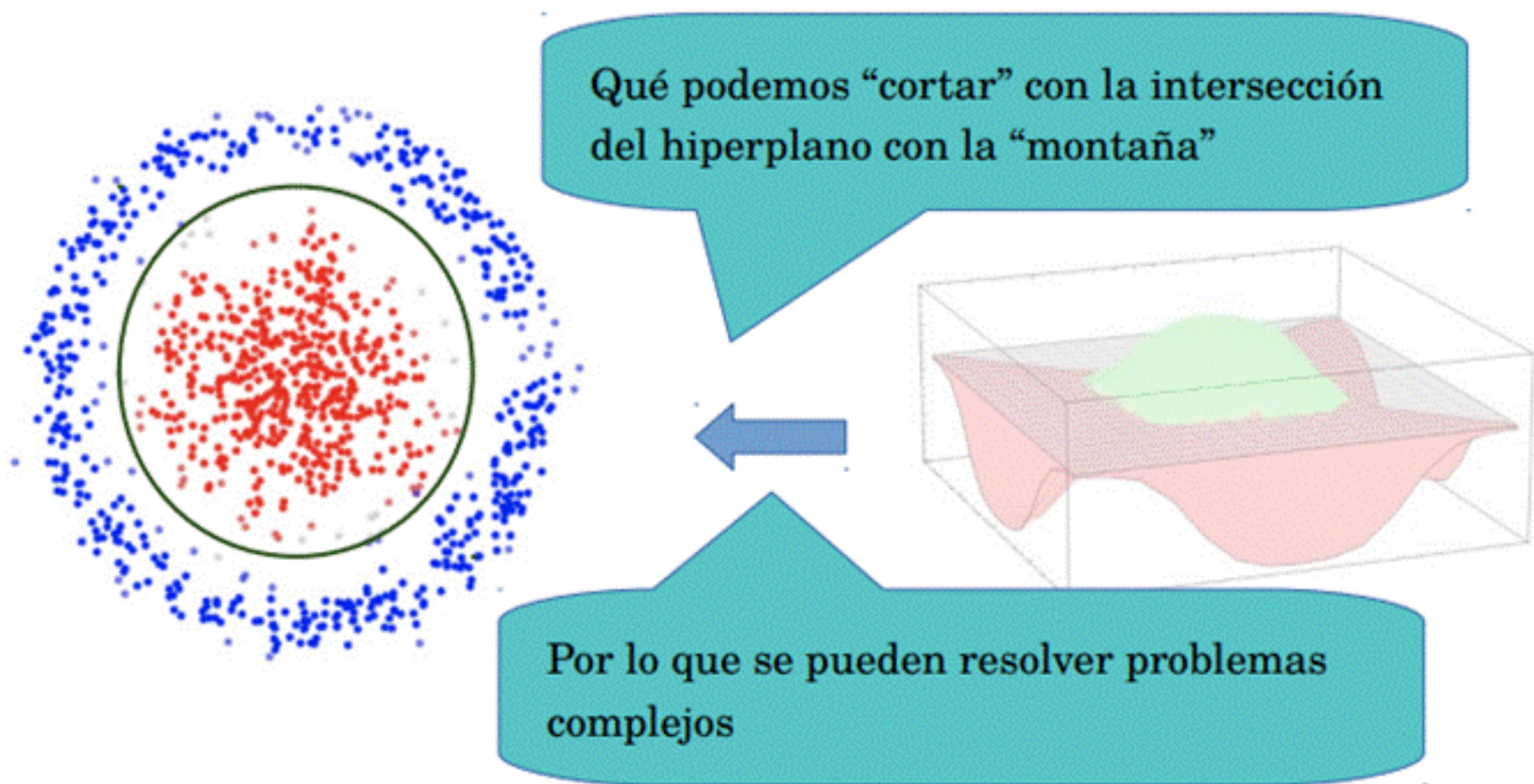


Obtenemos la superposición  
geométrica con un “bulto”  
al medio





## Segunda capa oculta



## Recursos:

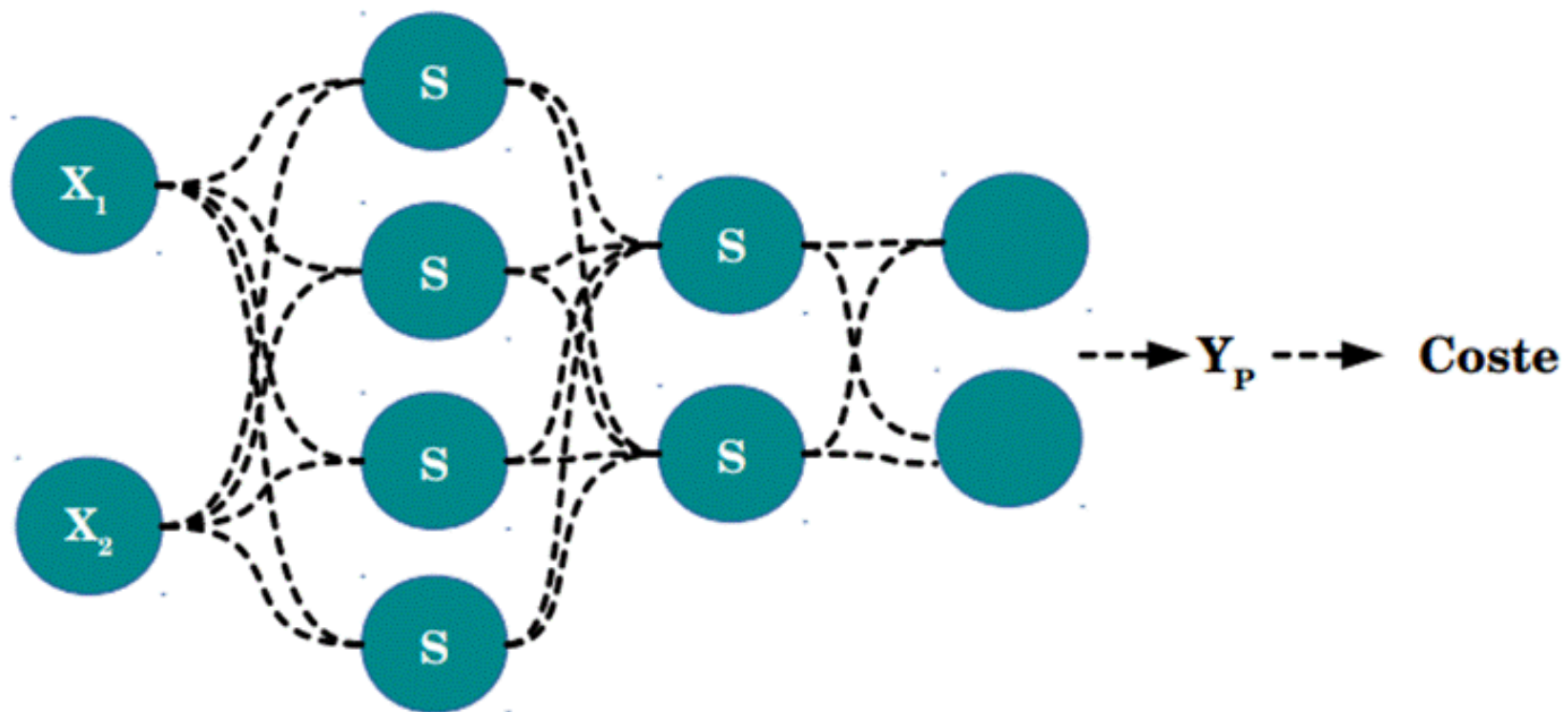
Documentación oficial de KERAS	<a href="https://keras.io/">https://keras.io/</a>
Documentación oficial de TensorFlow	<a href="https://www.tensorflow.org/?hl=es-419">https://www.tensorflow.org/?hl=es-419</a>
Introducción a bias y varianza	<a href="https://aprendeia.com/bias-y-varianza-en-machine-learning/">https://aprendeia.com/bias-y-varianza-en-machine-learning/</a>
An Interactive Tutorial on Numerical Optimization	<a href="http://www.benfrederickson.com/numerical-optimization/">http://www.benfrederickson.com/numerical-optimization/</a>
Simulador TensorFlow	<a href="https://playground.tensorflow.org/#activation=tanh&amp;batchSize=10&amp;dataset=circle&amp;regDataset=reg-plane&amp;learningRate=0.03&amp;regularizationRate=0&amp;noise=0&amp;networkShape=4,2&amp;seed=0.50939&amp;showTestData=false&amp;discretize=false&amp;percTrainData=50&amp;x=true&amp;y=true&amp;xTimesY=false&amp;xSquared=false&amp;ySquared=false&amp;cosX=false&amp;sinX=false&amp;cosY=false&amp;sinY=false&amp;collectStats=false&amp;problem=classification&amp;initZero=false&amp;hideText=false">https://playground.tensorflow.org/#activation=tanh&amp;batchSize=10&amp;dataset=circle&amp;regDataset=reg-plane&amp;learningRate=0.03&amp;regularizationRate=0&amp;noise=0&amp;networkShape=4,2&amp;seed=0.50939&amp;showTestData=false&amp;discretize=false&amp;percTrainData=50&amp;x=true&amp;y=true&amp;xTimesY=false&amp;xSquared=false&amp;ySquared=false&amp;cosX=false&amp;sinX=false&amp;cosY=false&amp;sinY=false&amp;collectStats=false&amp;problem=classification&amp;initZero=false&amp;hideText=false</a>
<b>Multivariate Linear Regression</b>	<a href="https://medium.com/@lope.ai/multivariate-linear-regression-from-scratch-in-python-5c4f219be6a">https://medium.com/@lope.ai/multivariate-linear-regression-from-scratch-in-python-5c4f219be6a</a>



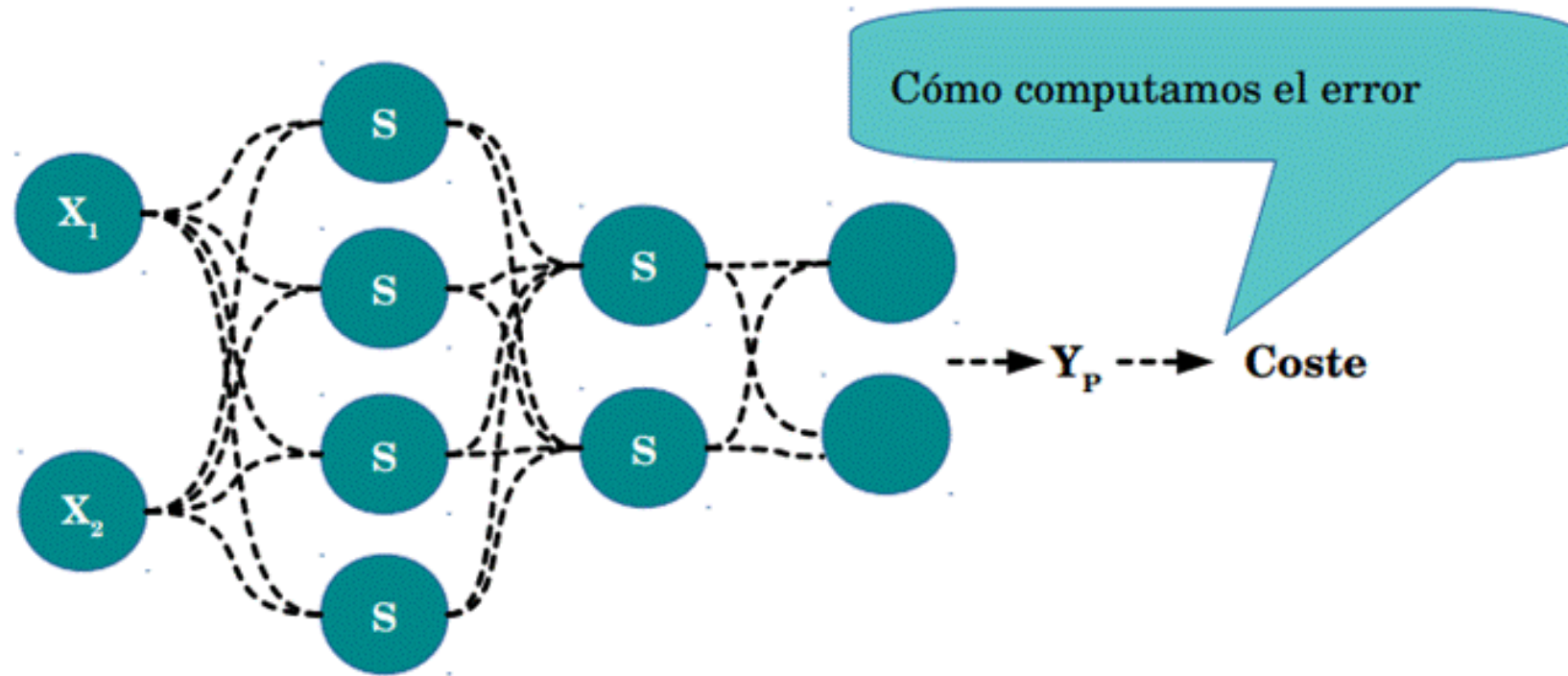
# Backpropagation

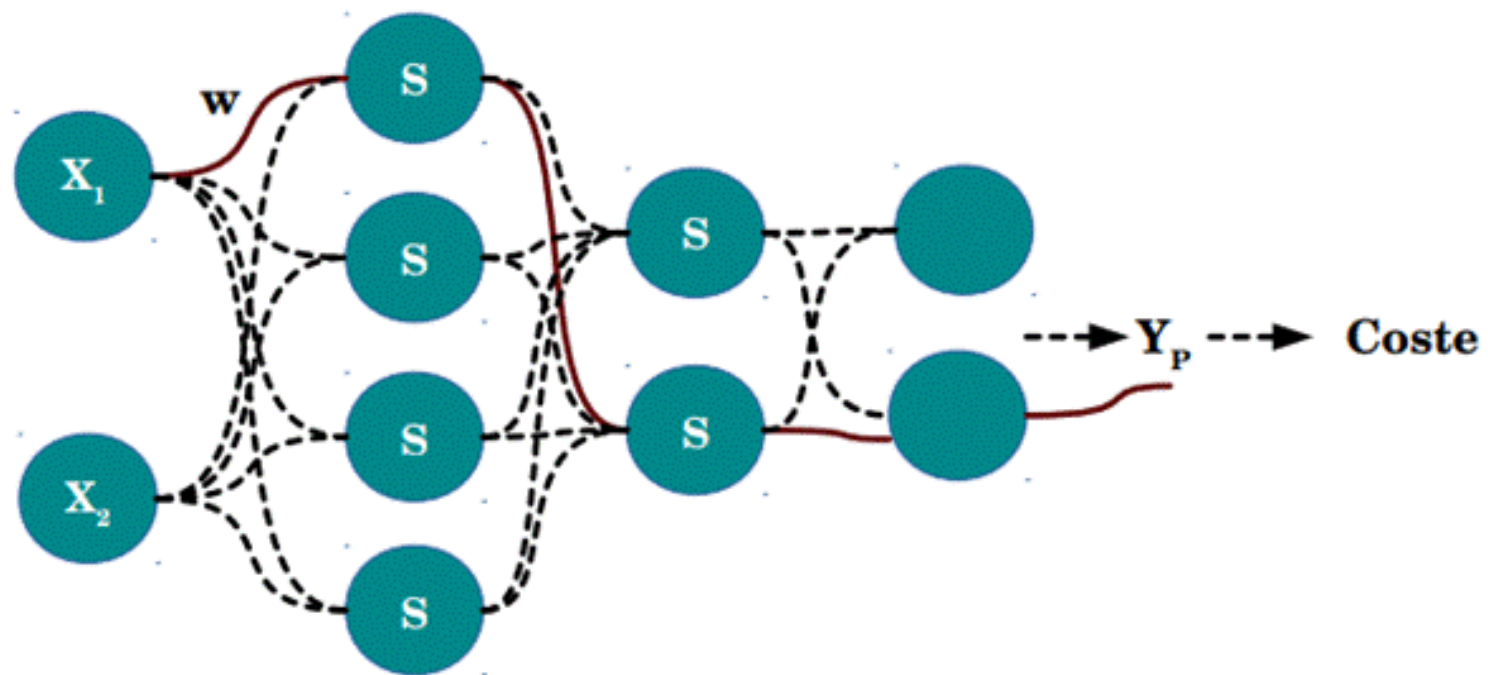
- En 1958 se publica el artículo de Perceptron, i.e., desarrollo de una única neurona. – Pero al poco tiempo muestran diferentes estudios con todas las limitaciones como hemos visto.
  - Se llega al periodo de “Invierno de la Inteligencia Artificial” que dura 15 años de inactividad en desarrollo.
  - Básicamente las limitaciones son muchas y se corta la financiación.
- Pero volvemos a la “Primavera de la IA” en 1986 con la llegada del artículo que desarrolla algoritmo Backpropagation de David E. Rumelhart et al.
  - Básicamente exponen un algoritmo el cual una red neuronal ajusta “automáticamente” sus parámetros.
  - Entra en acción el “Gradiente Descendiente” pero lo veremos en la siguiente sesión.

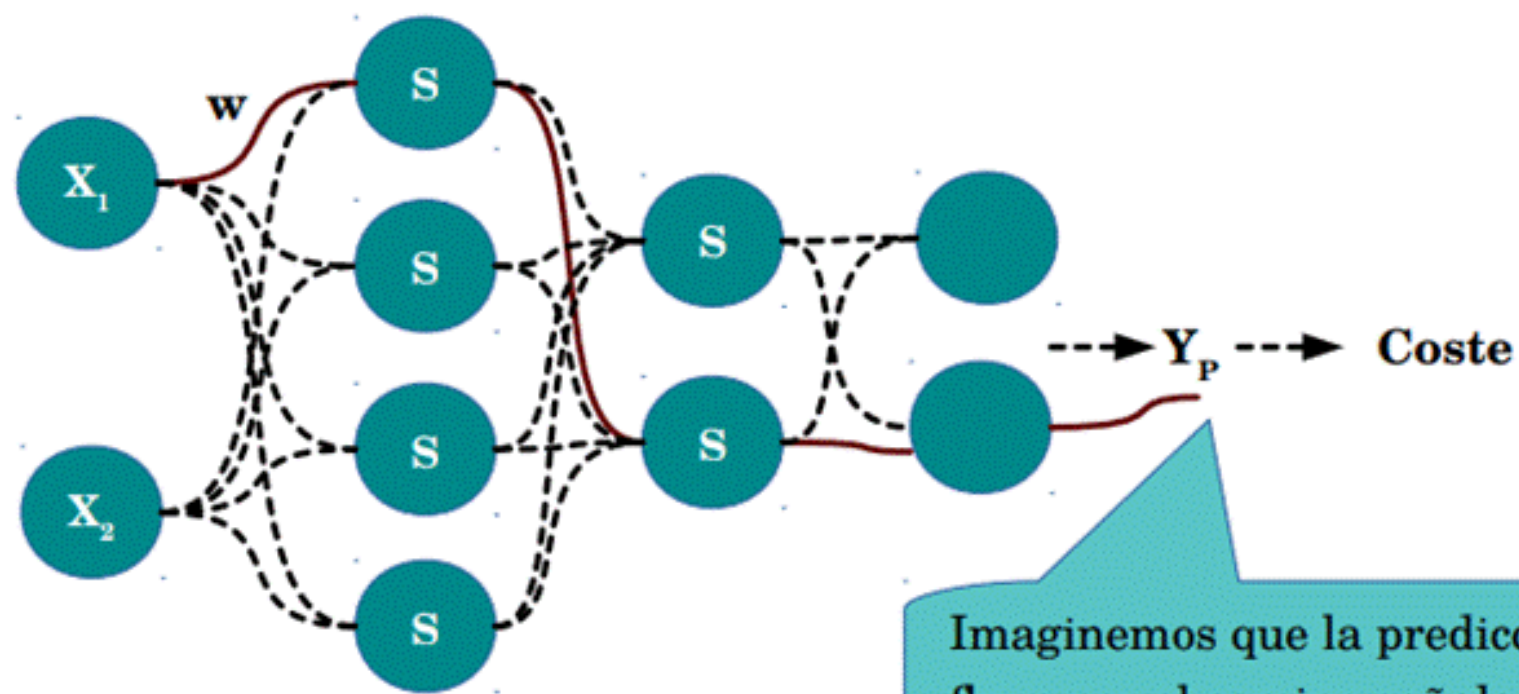
## Contexto



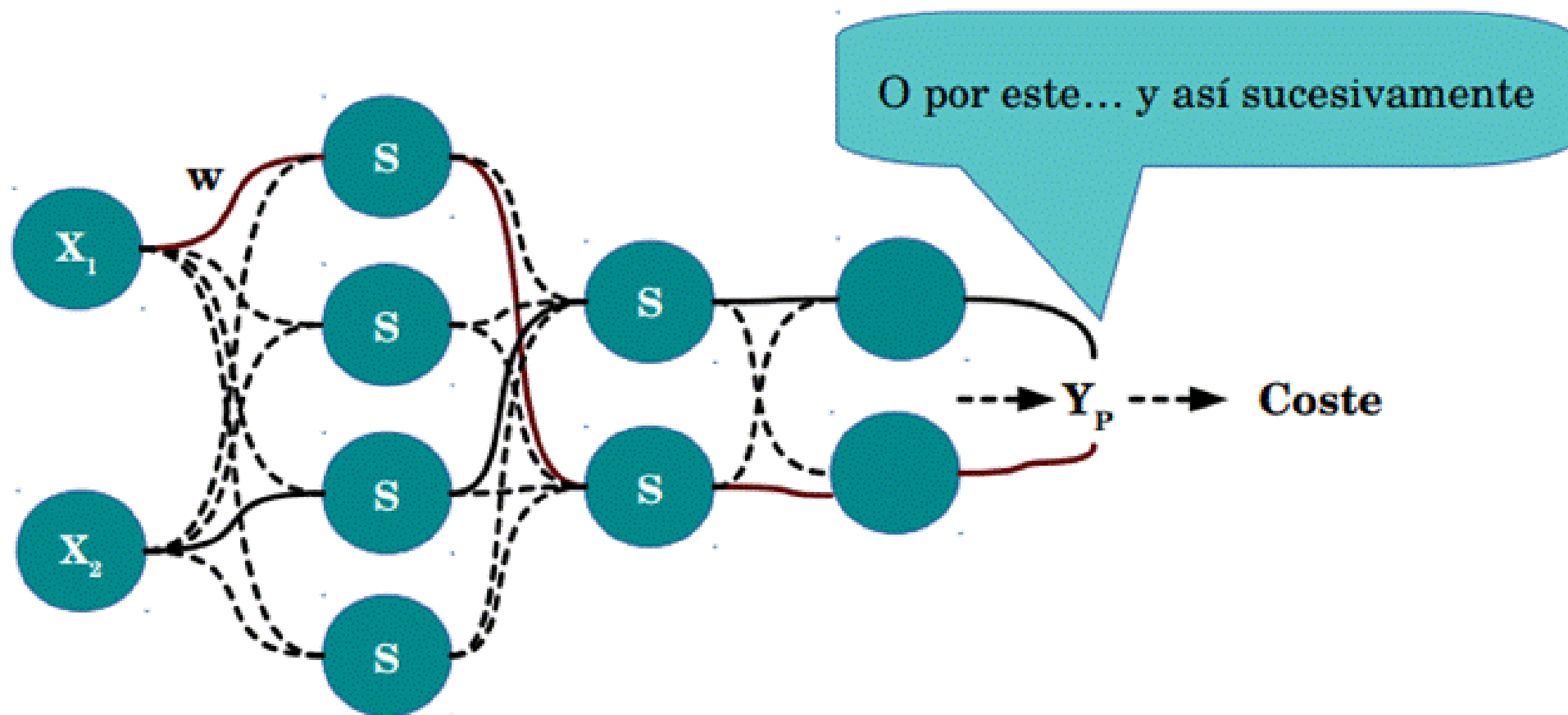
## Conexiones e iteraciones

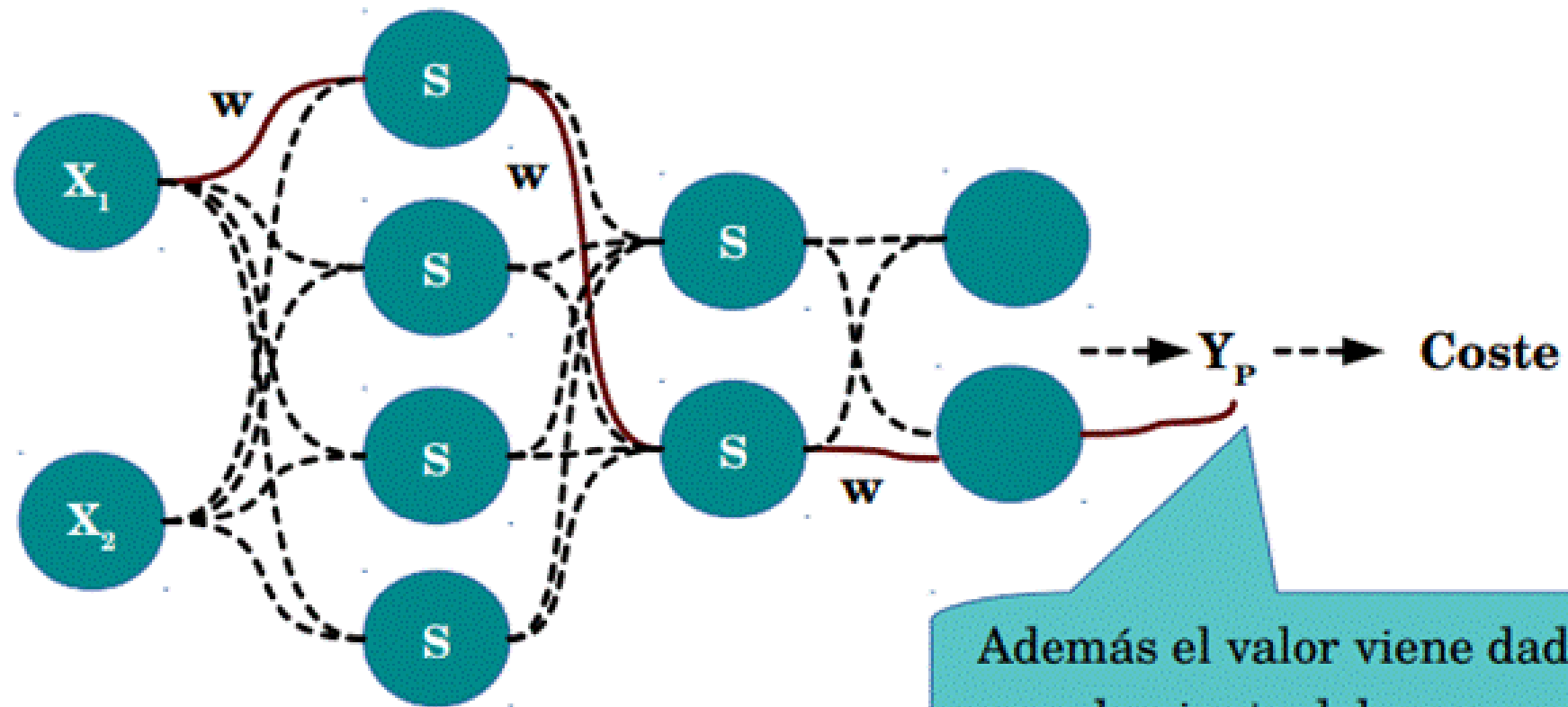




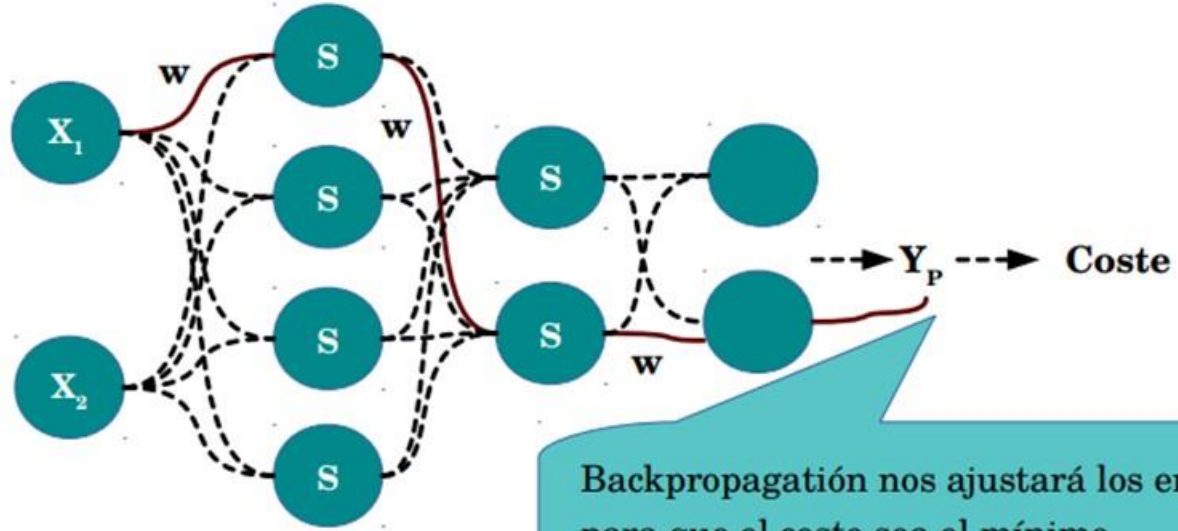


Imaginemos que la predicción fluye por el camino señalado





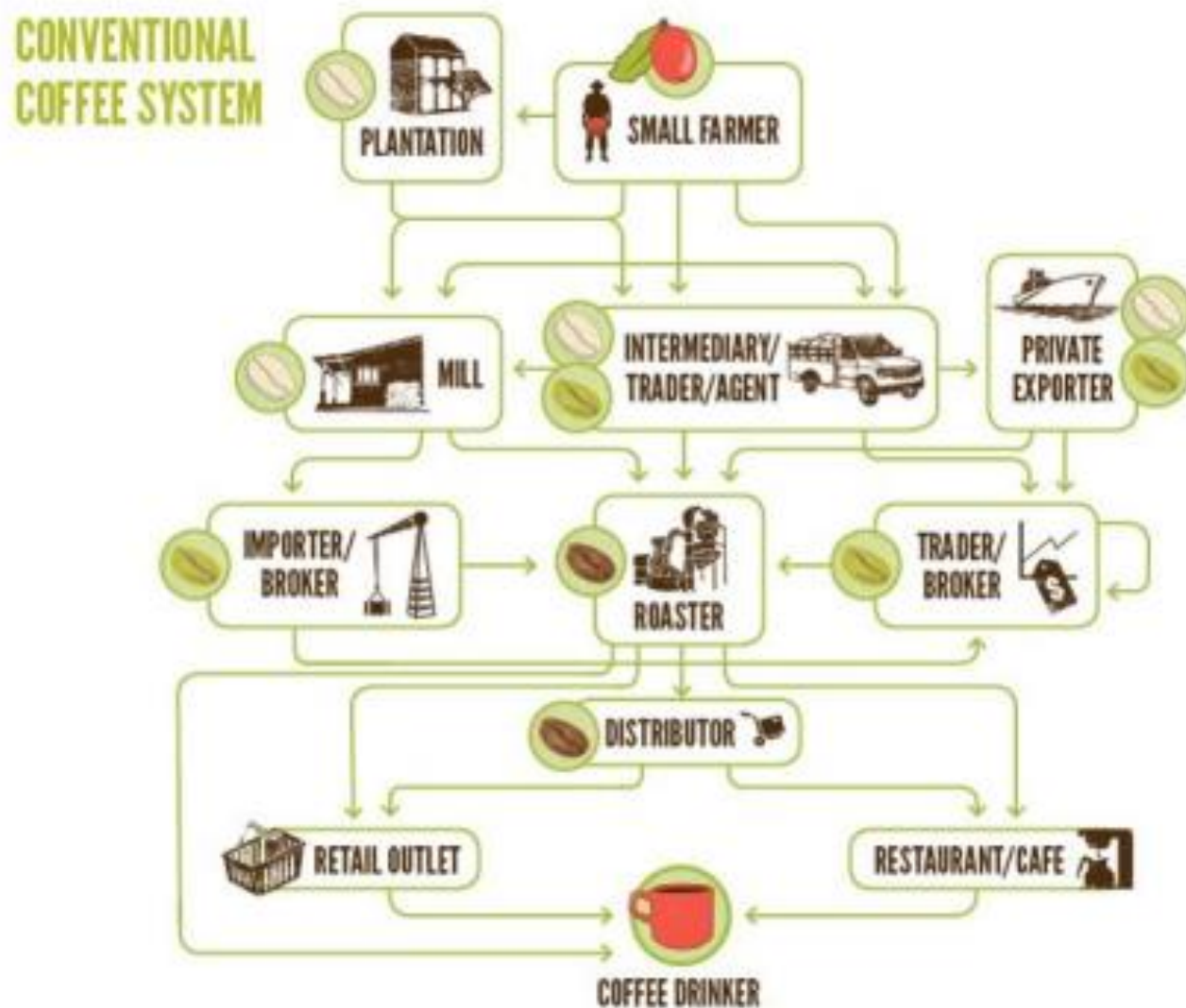
Además el valor viene dado por encadenamiento del error previo



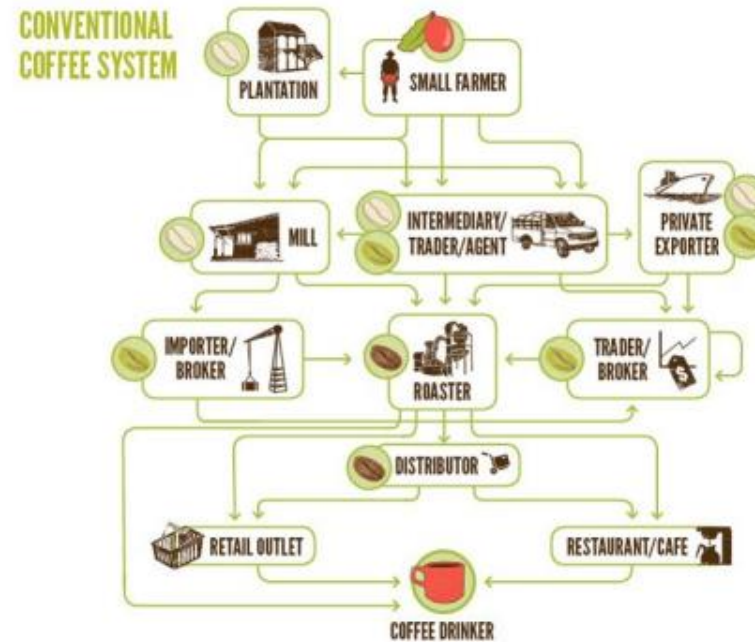
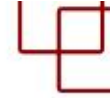
Backpropagación nos ajustará los errores para que el coste sea el mínimo. Para ello, utilizará el Gradiente Descendiente



### 3. Pensemos en el día a día

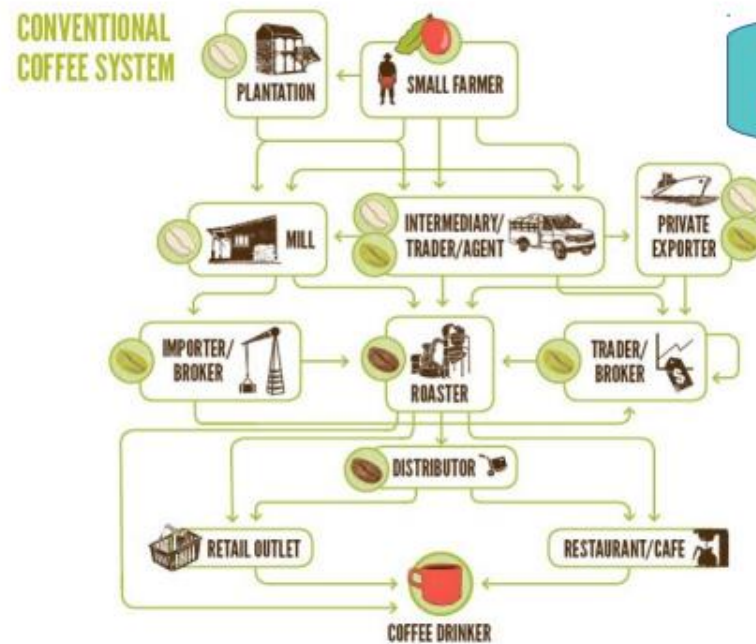
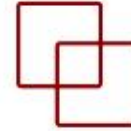


### 3. Pensemos en el día a día



¿De quién es la culpa?

### 3. Pensemos en el día a día



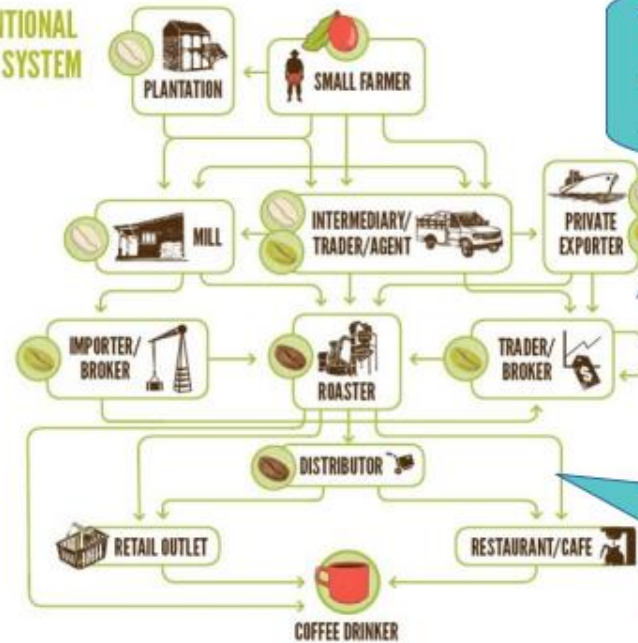
Tenemos que analizar toda la Cadena para ver quién es el culpable



### 3. Pensemos en el día a día



CONVENTIONAL  
COFFEE SYSTEM



Viendo el porcentaje de implicación de cada neurona en la predicción y, así, corregirlo



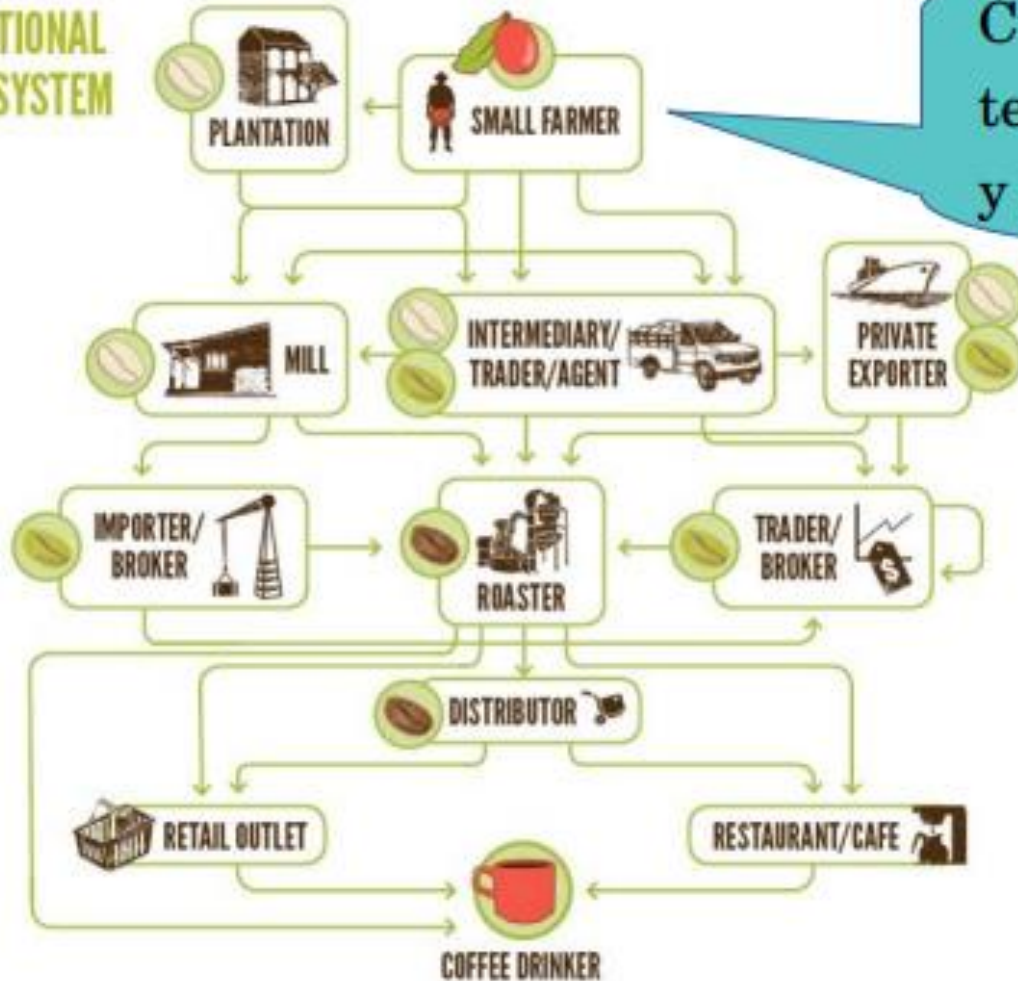
Para ello, analizamos el error y vamos analizando hacia atrás para corregirlo



### 3. Pensemos en el día a día



#### CONVENTIONAL COFFEE SYSTEM

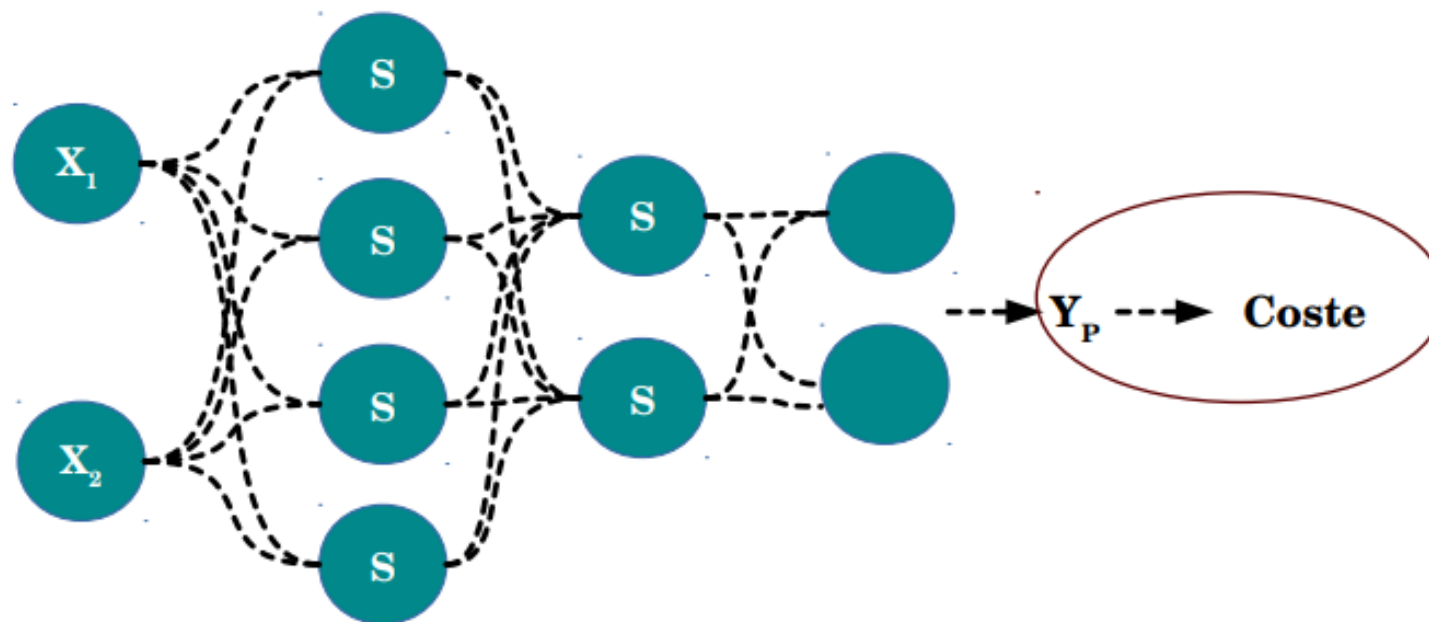


Cuando lleguemos a la primera capa tendremos el error de cada neurona y sus parámetros

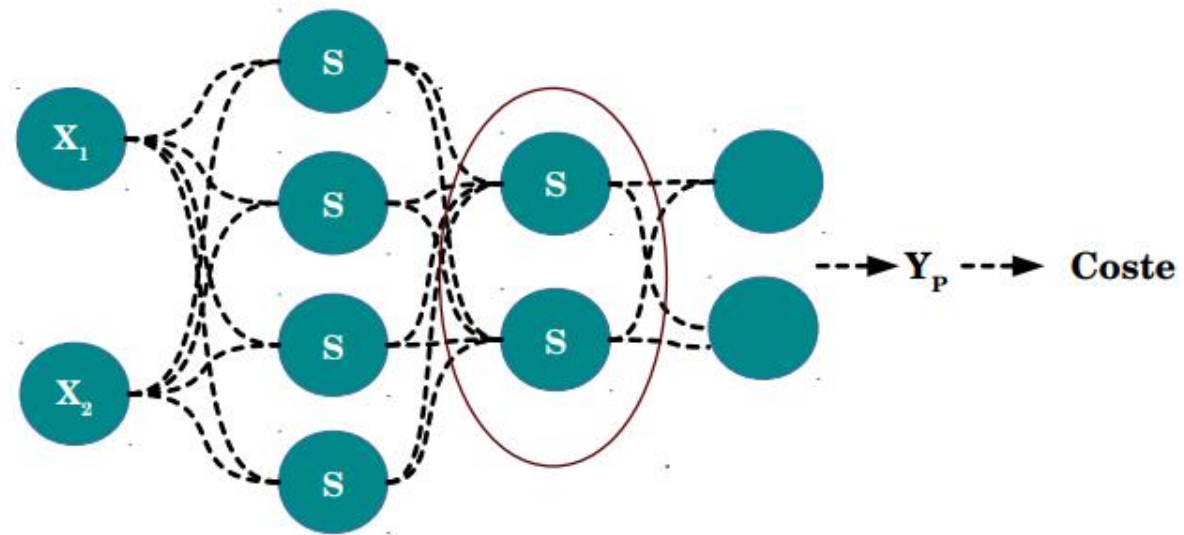


## 4. En MLP

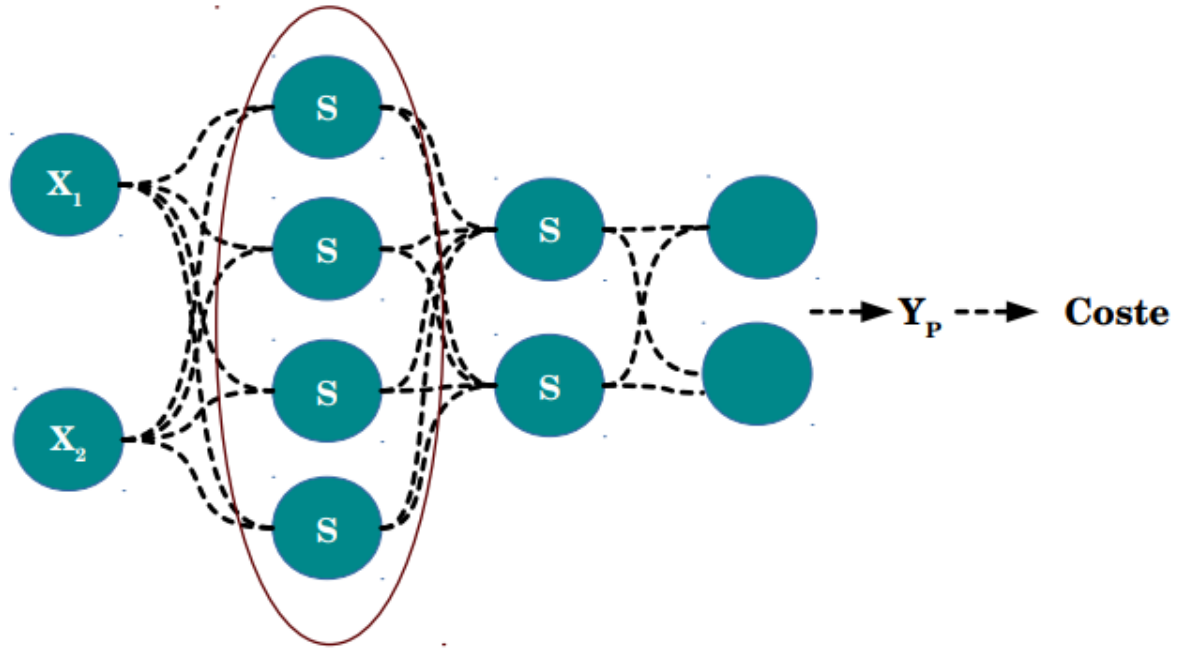
L



## 4. En MLP

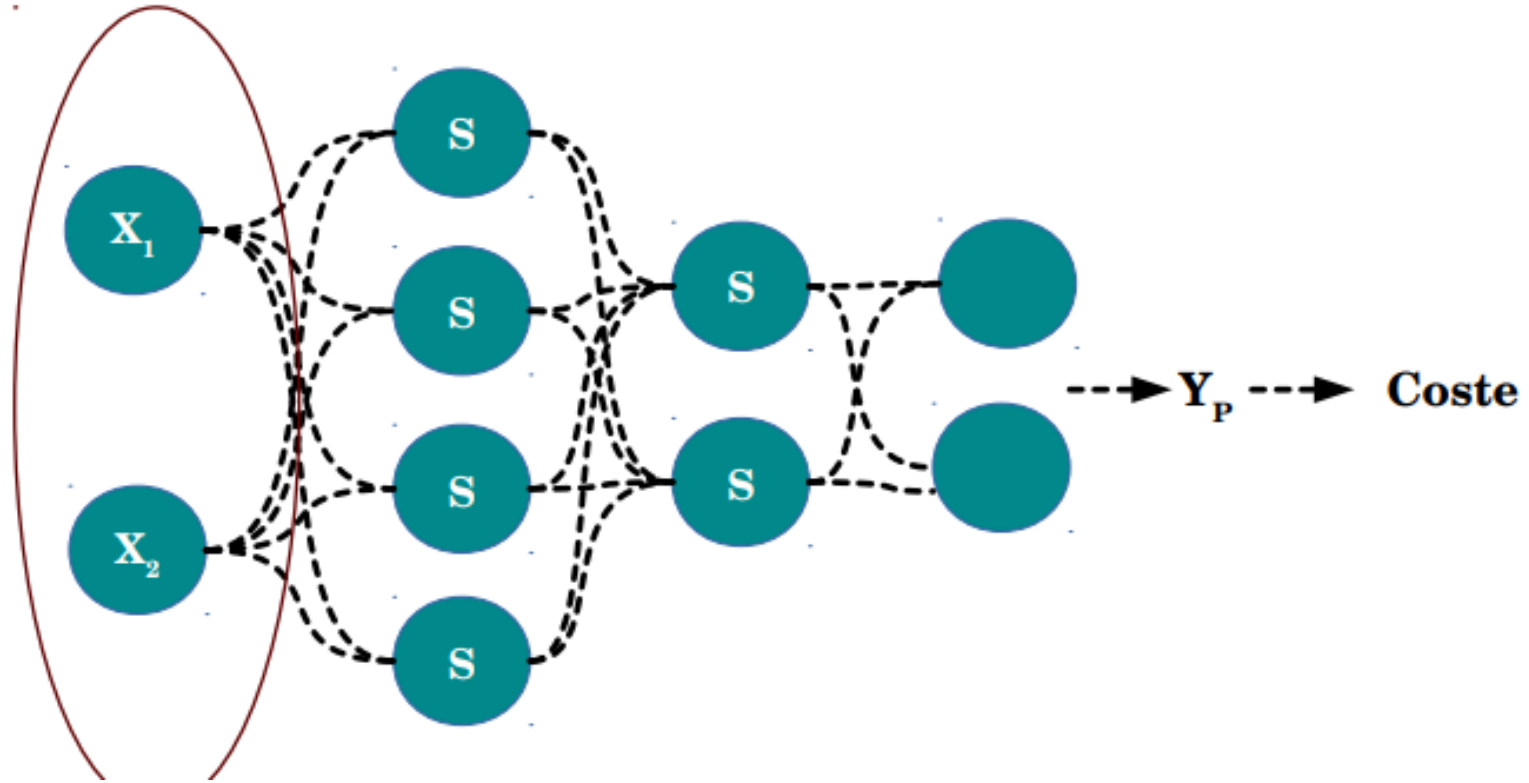


## 4. En MLP

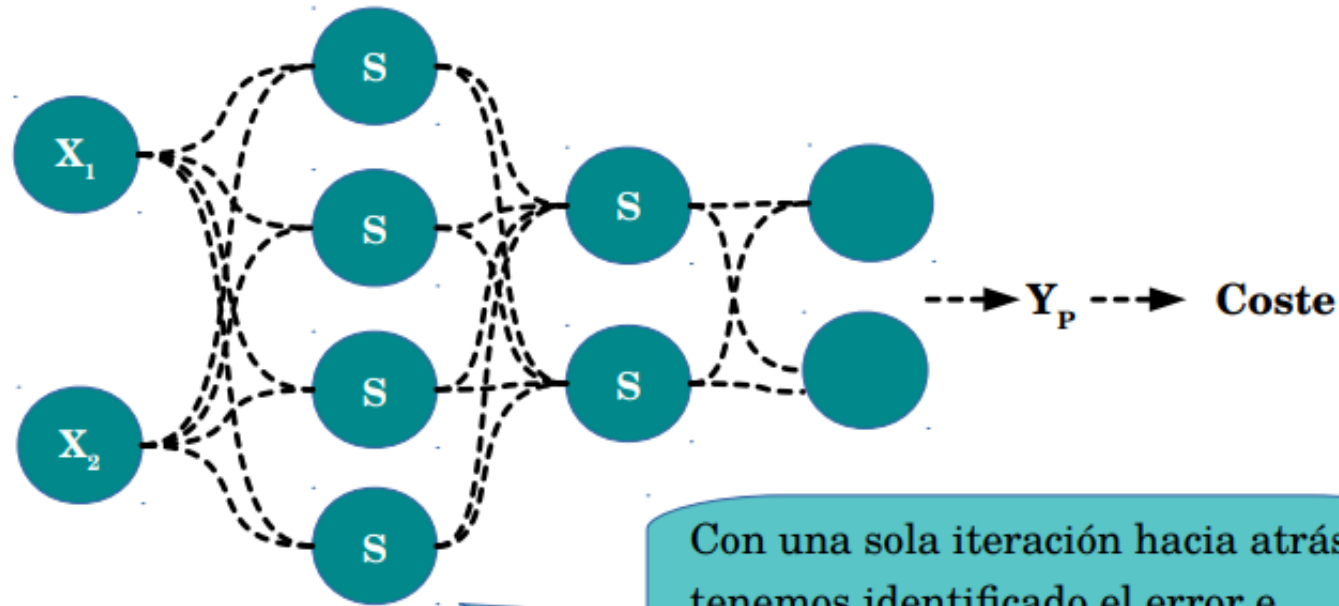




## 4. En MLP



## 4. En MLP



Con una sola iteración hacia atrás  
tenemos identificado el error e  
implicancia de parámetros y neuronas



