# Escuela de Aprendizaje Profundo: Regresión Lineal y Redes Neuronales

**Mario Ezra Aragón**, Juan Luís García Mendoza, Adrian Pastor López Monroy, Manuel Montes y Gómez, y Luís Villaseñor Pineda

México, Octubre 07, 2020

CONACYT

CIMAT

INAOE

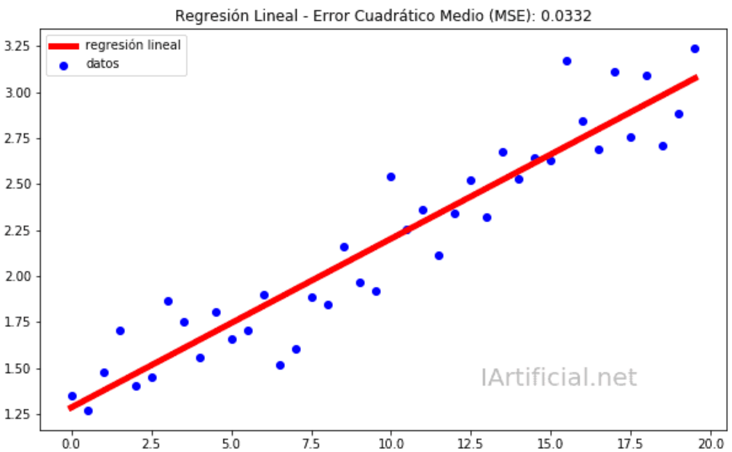
# Temas

- Fundamentos de regresión lineal
  - Gradiente Descendente
  - ¿Qué es y cómo funciona la regresión lineal?
  - Ejercicio
- Introducción a redes neuronales
  - Perceptrón
  - Atacando complejidad con redes neuronales
  - Backpropagation
  - Ejercicio

# Créditos

- El contenido de esta presentación académica es una recopilación de notas que provienen de diversas fuentes incluyendo: libros, ensayos, cursos en línea, etc. A continuación se enlistan las principales.
  - Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
  - https://medium.com/dair-ai/a-simple-neural-network-from-scratch-with-pytorch-and-google-colab-c7f3830618e0
  - https://www.kaggle.com/aakashns/pytorch-basics-linear-regression-from-scratch
  - https://www.youtube.com/watch?v=IKloEocn3Hw&ab_channel=codificandobits
  - https://www.youtube.com/watch?v=hutg0JpDbPY&ab_channel=codificandobits
  - https://www.youtube.com/watch?v=MRIv2IwFTPg&list=LLJhad2jIxGdaWdQpyBbN-oQ&index=9&ab_channel=DotCSV

# Regresión Lineal

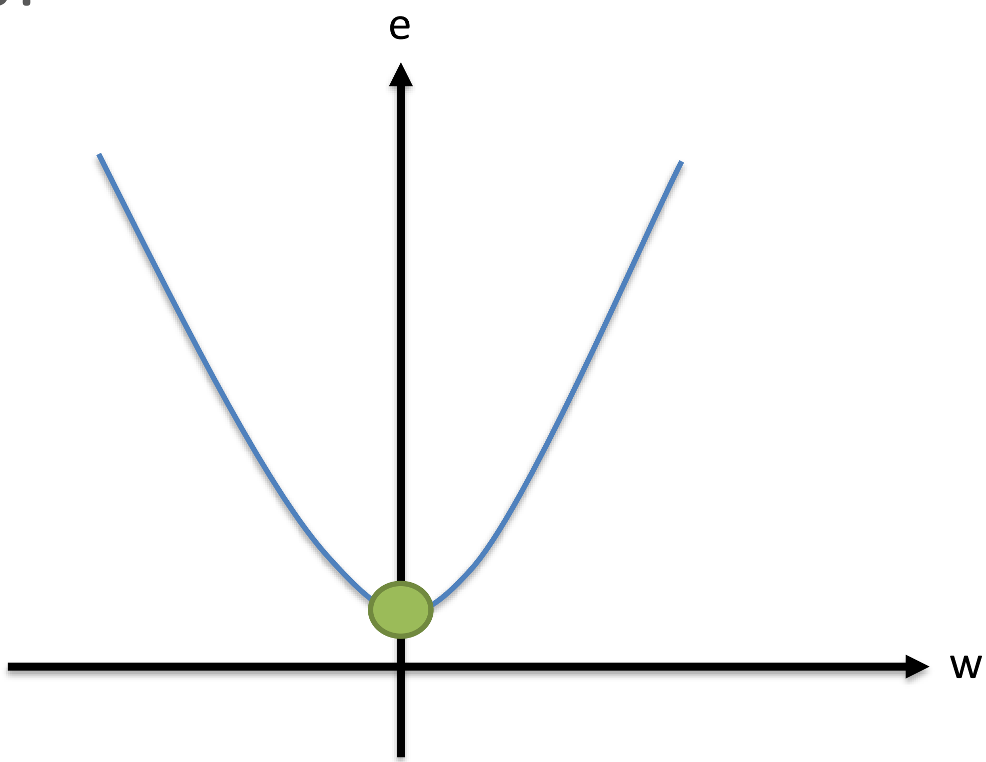


# El Gradiente Descendente

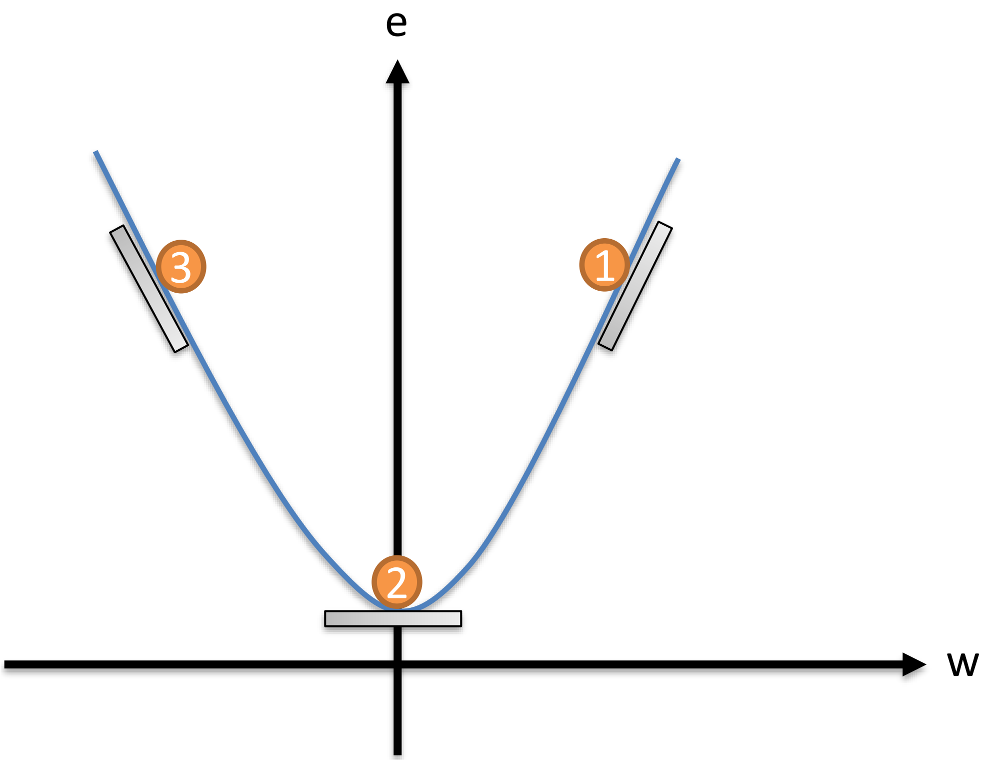
- ¿Qué es?

# El Gradiente Descendente

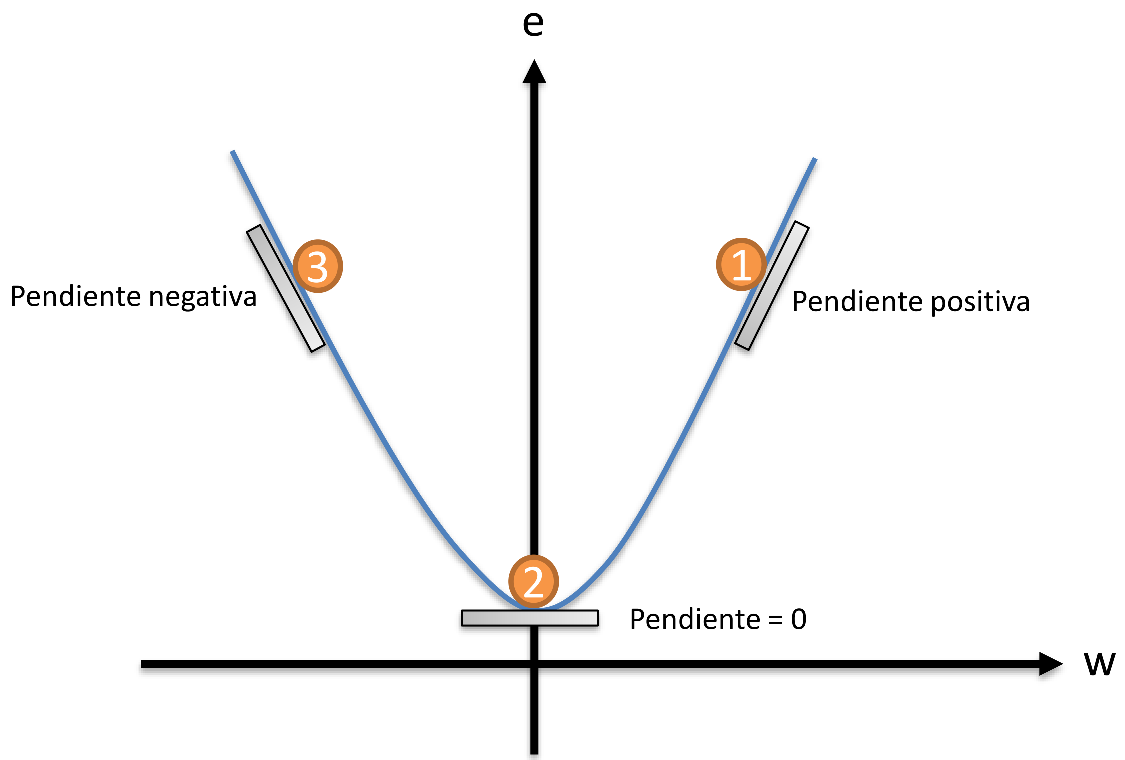
- ¿Qué es?



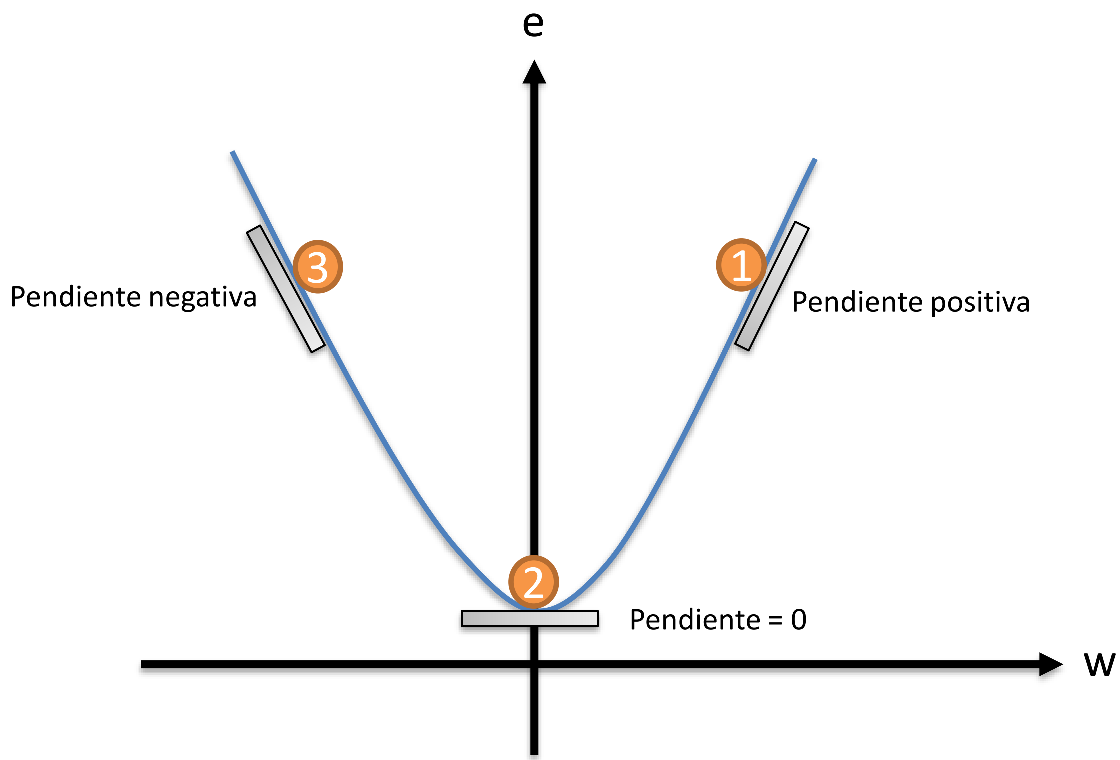
# El Gradiente Descendente



# El Gradiente Descendente



# El Gradiente Descendente



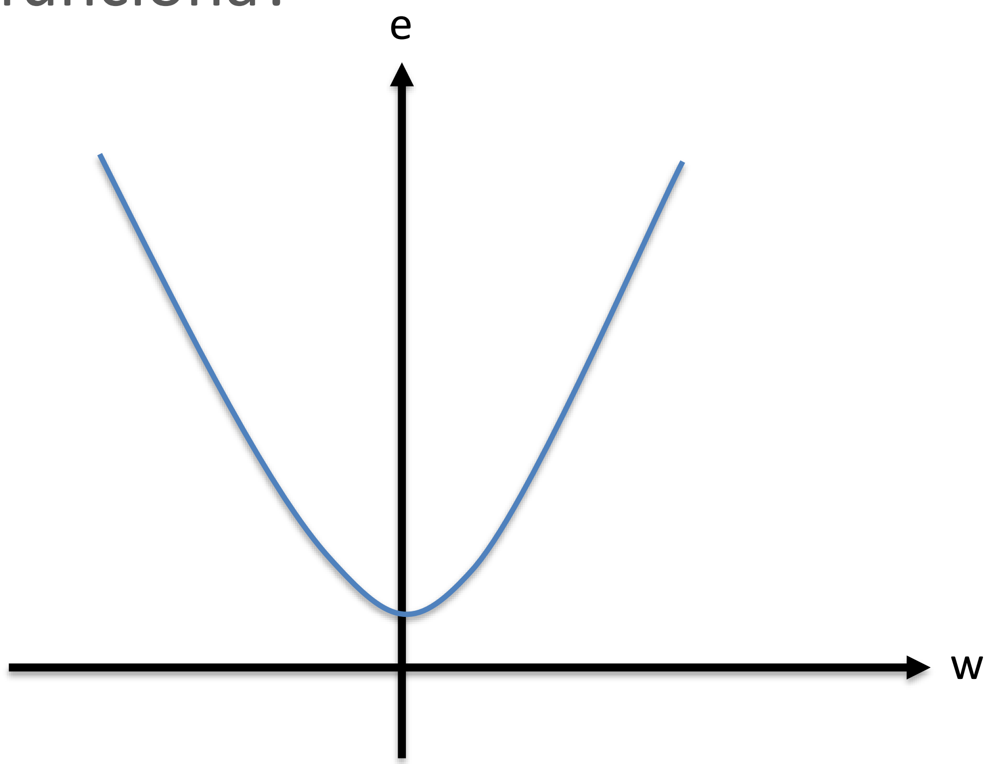

Gradiente

Pendiente

Derivada

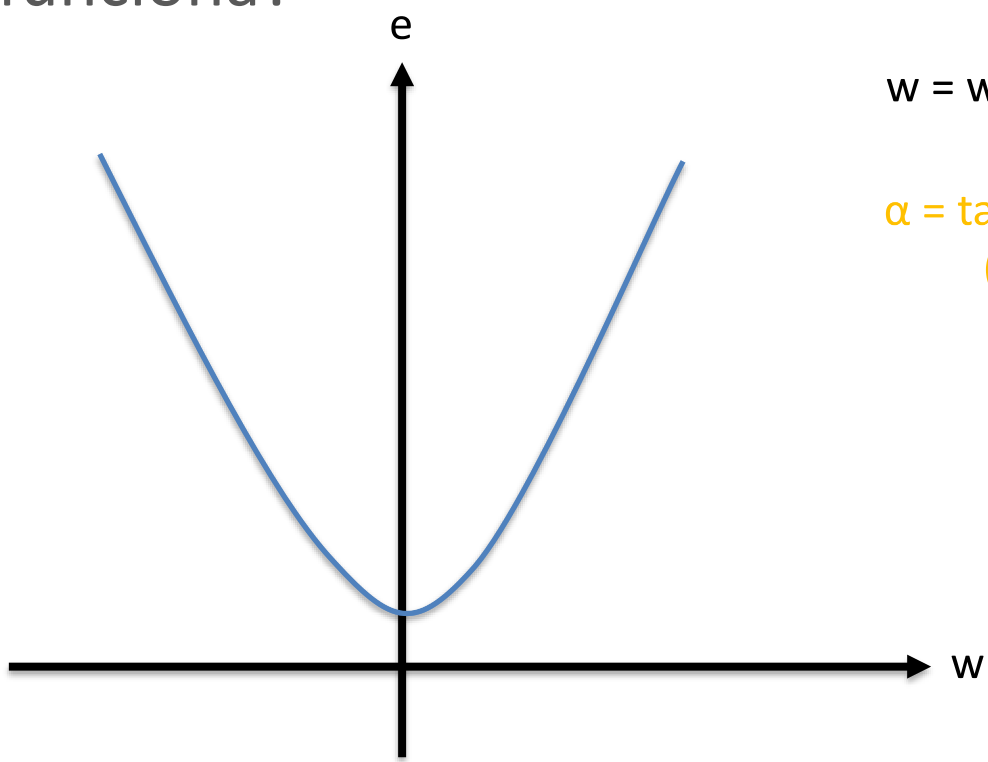
# El Gradiente Descendente

- ¿Cómo funciona?



# El Gradiente Descendente

- ¿Cómo funciona?

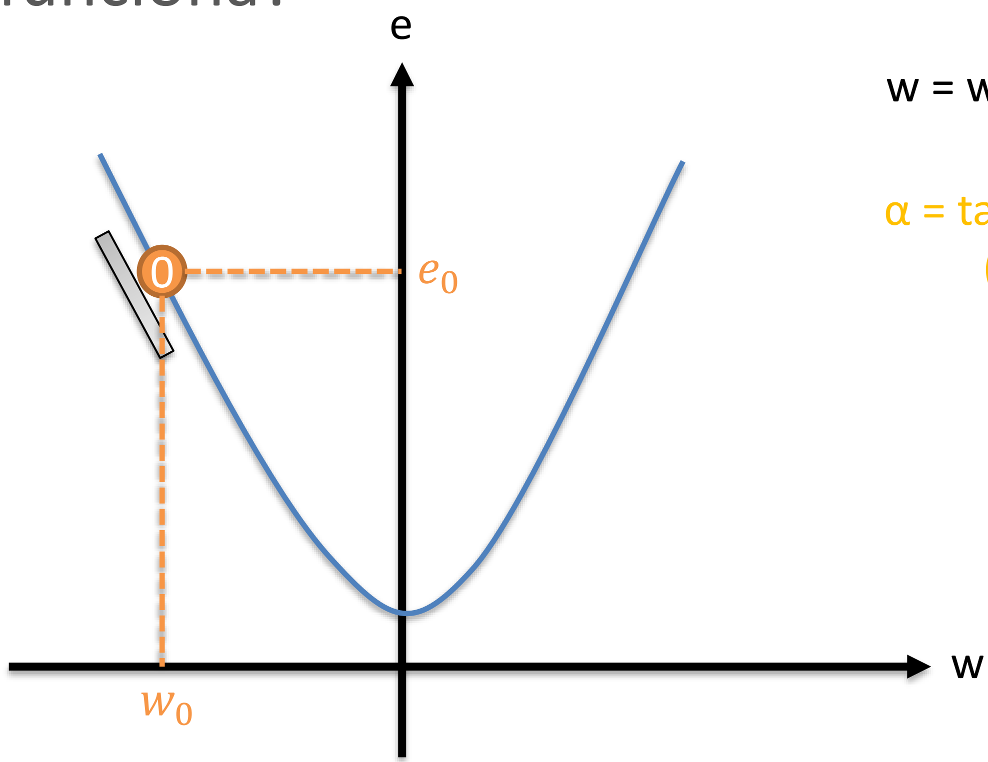



$w = w - \alpha \cdot \text{gradiente}$

$\alpha$ = tasa de aprendizaje (learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

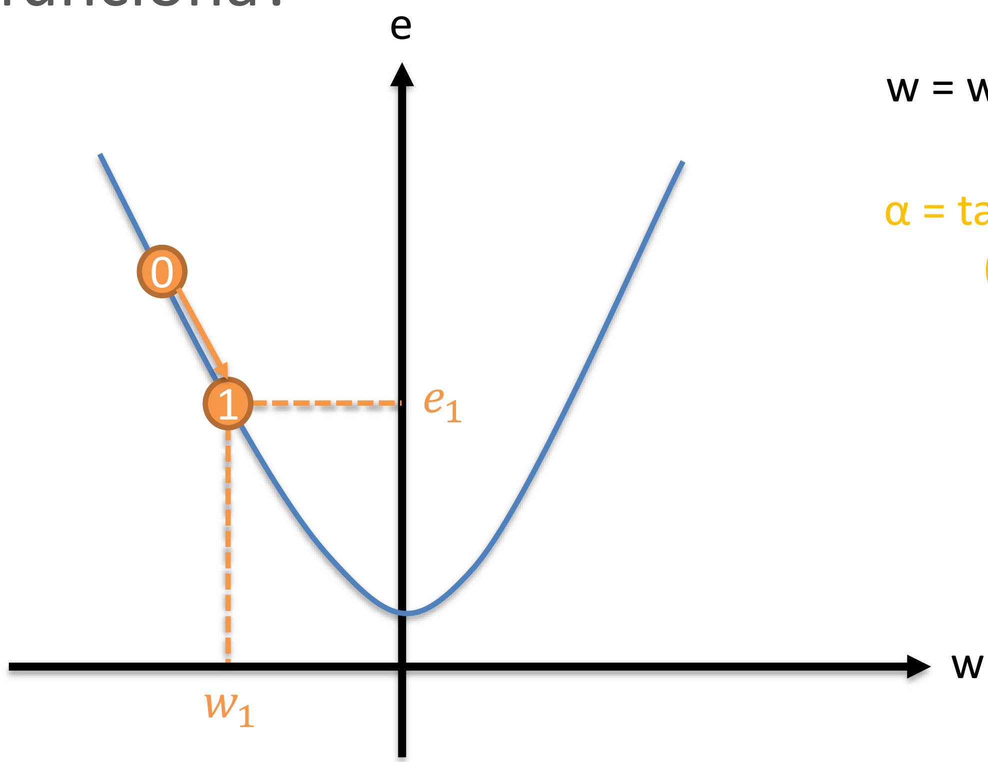



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

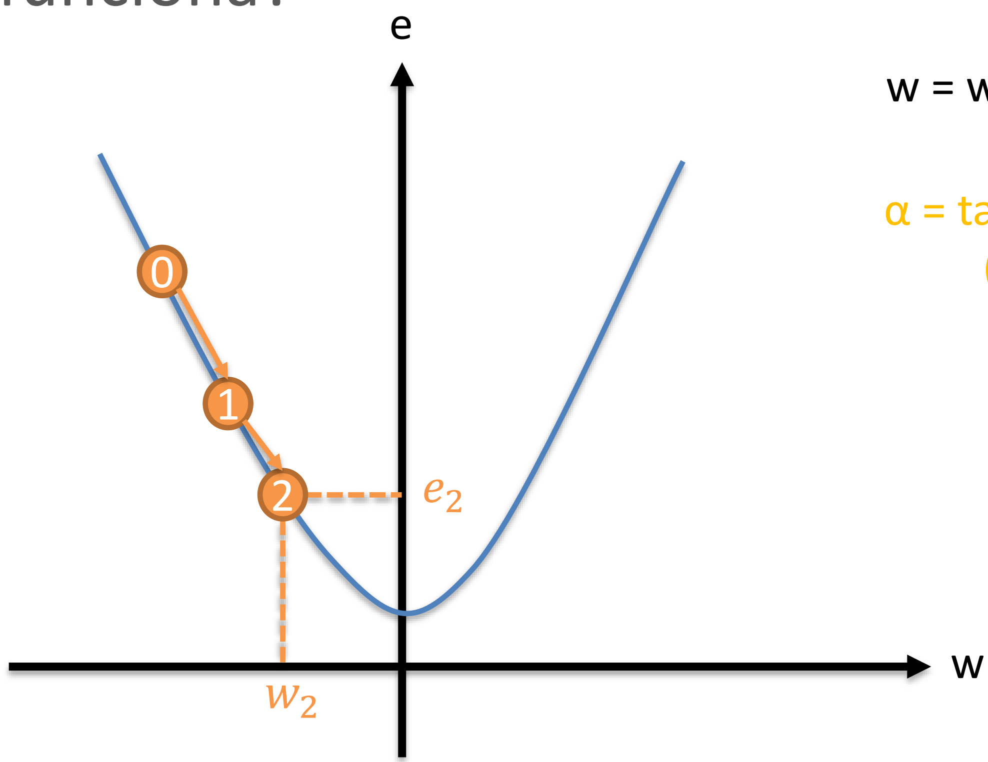



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

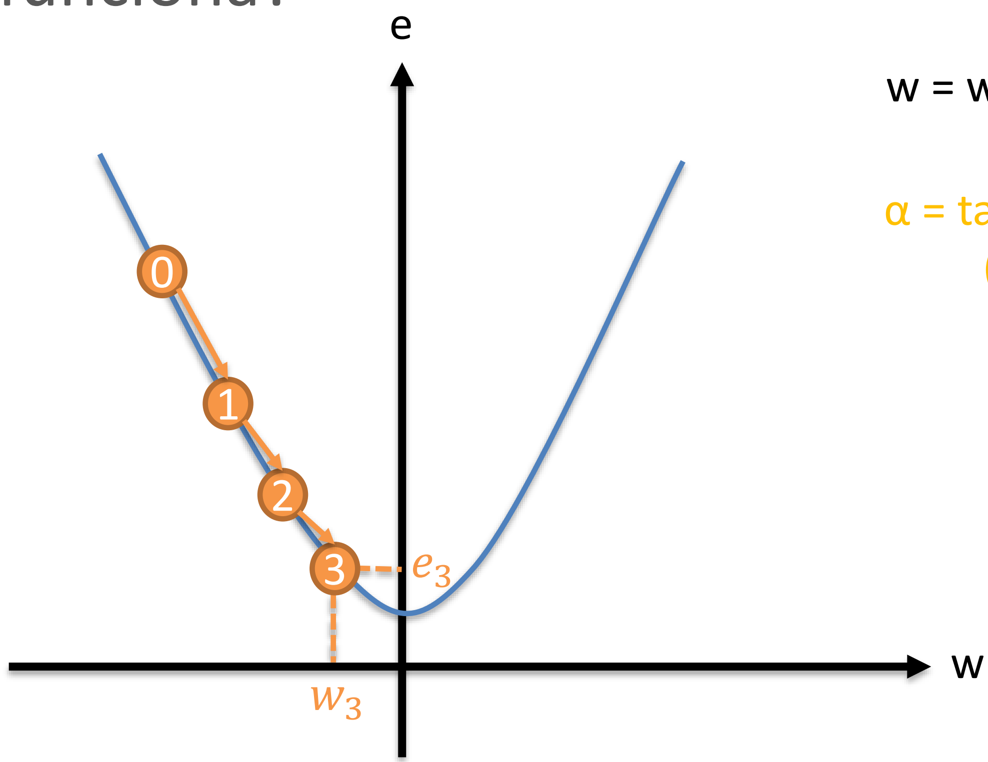



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

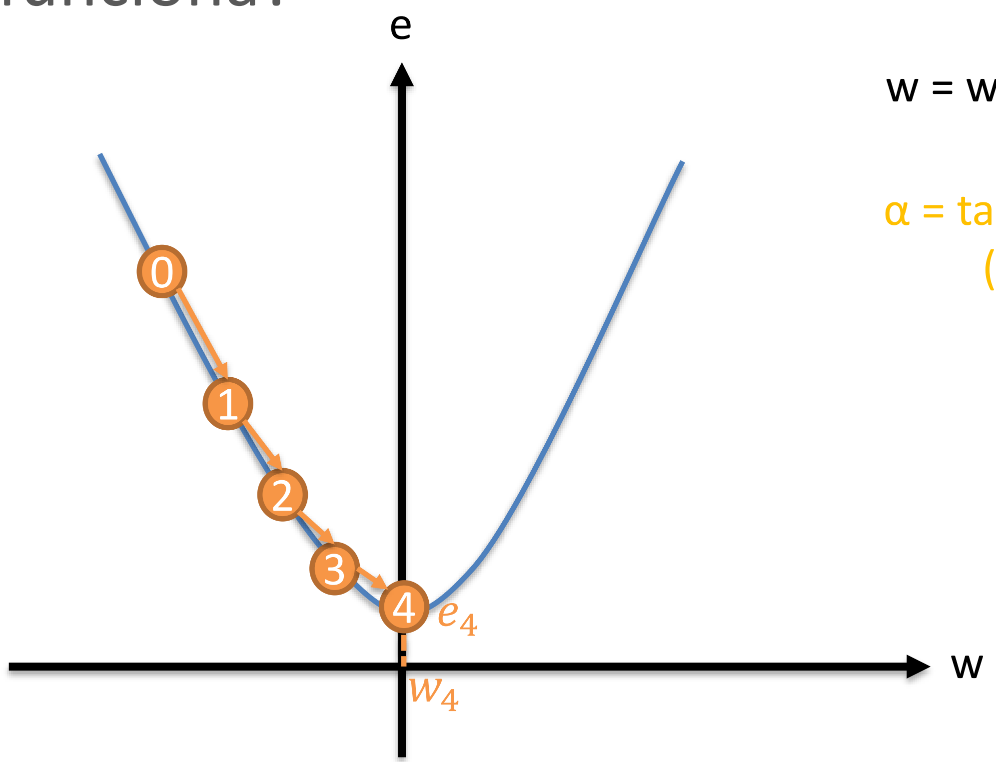



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

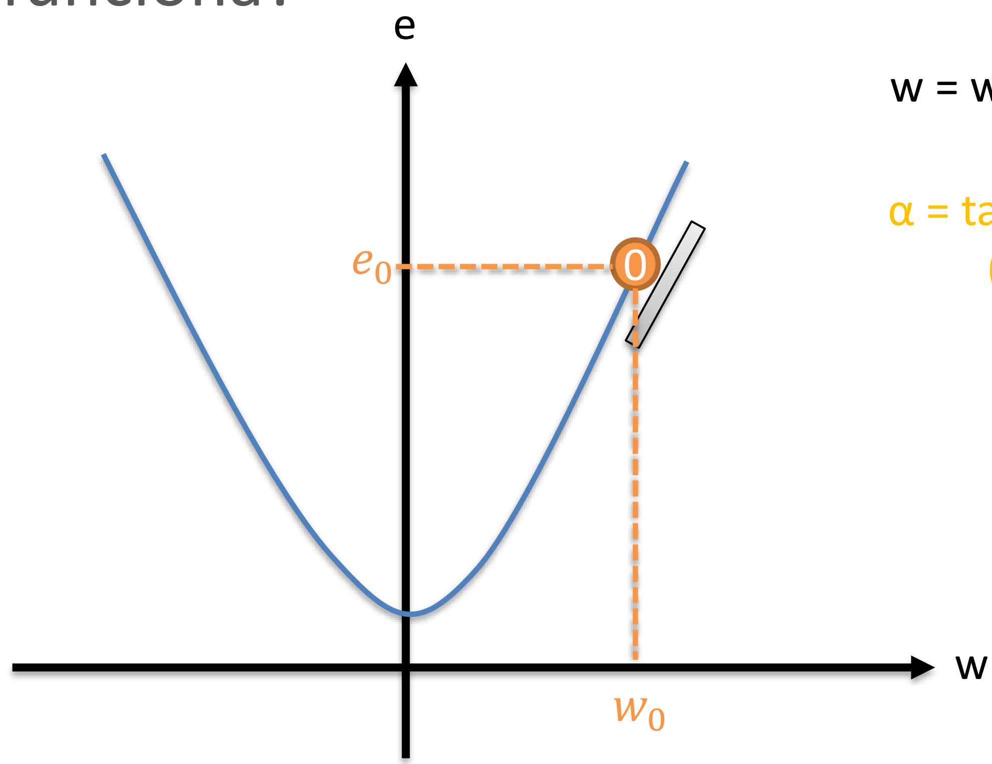



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

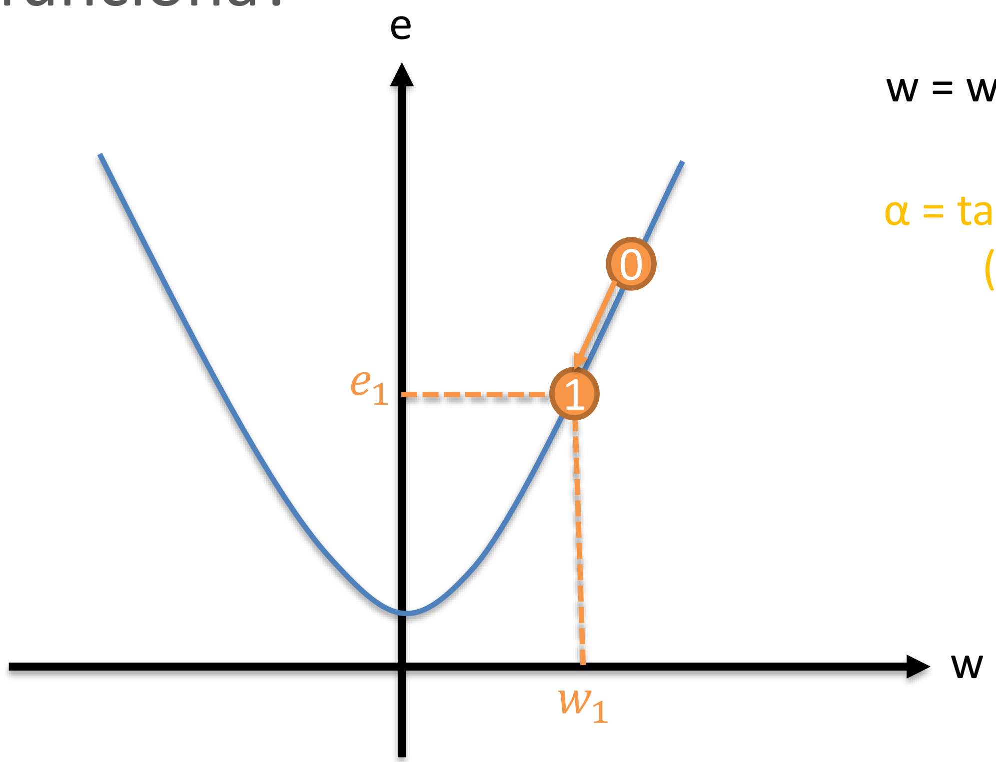



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

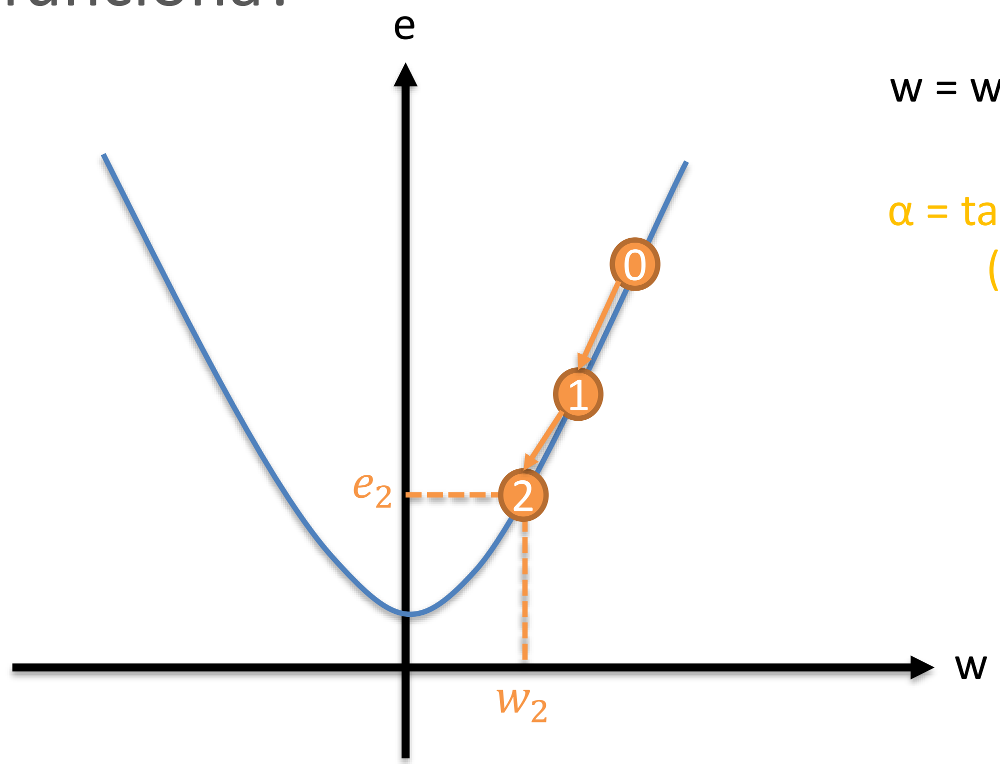



w = w – α · gradiente

α = tasa de aprendizaje (learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

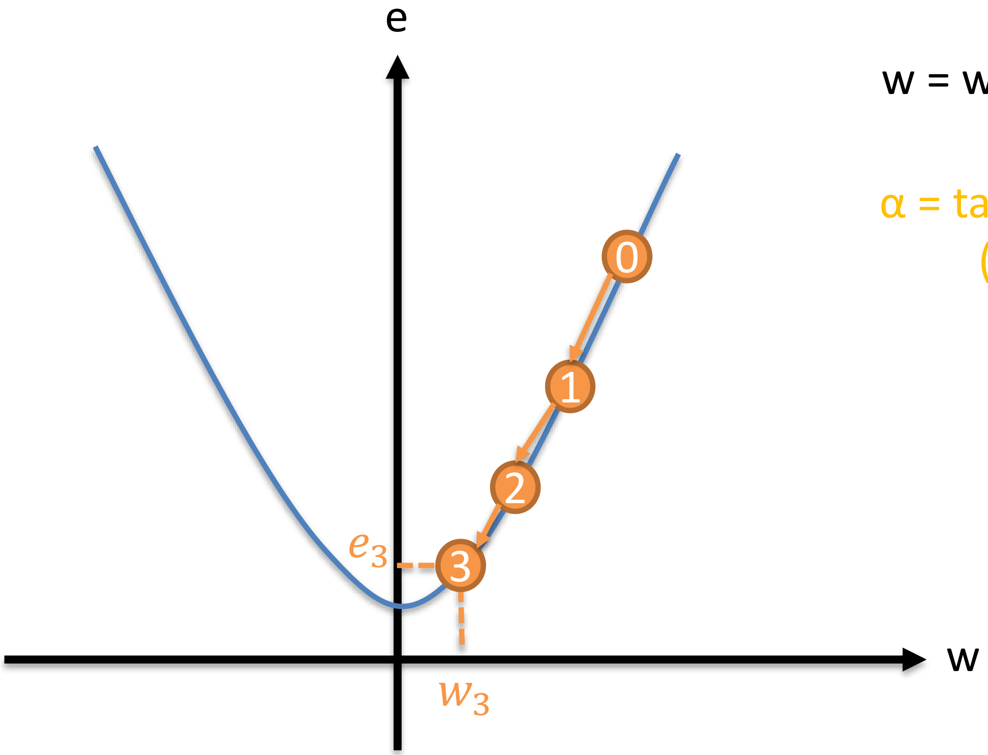



w = w – α · gradiente

α = tasa de aprendizaje (learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?

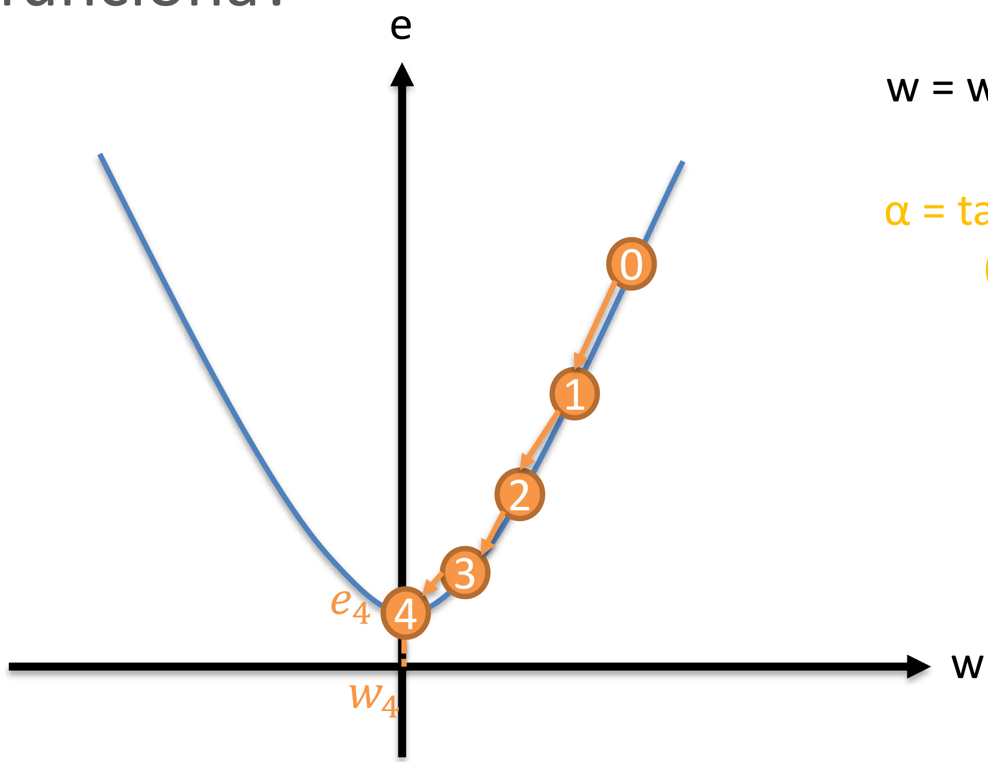



w = w – α · gradiente

α = tasa de aprendizaje
(learning rate)

# El Gradiente Descendente

- ¿Cómo funciona?




w = w – α · gradiente

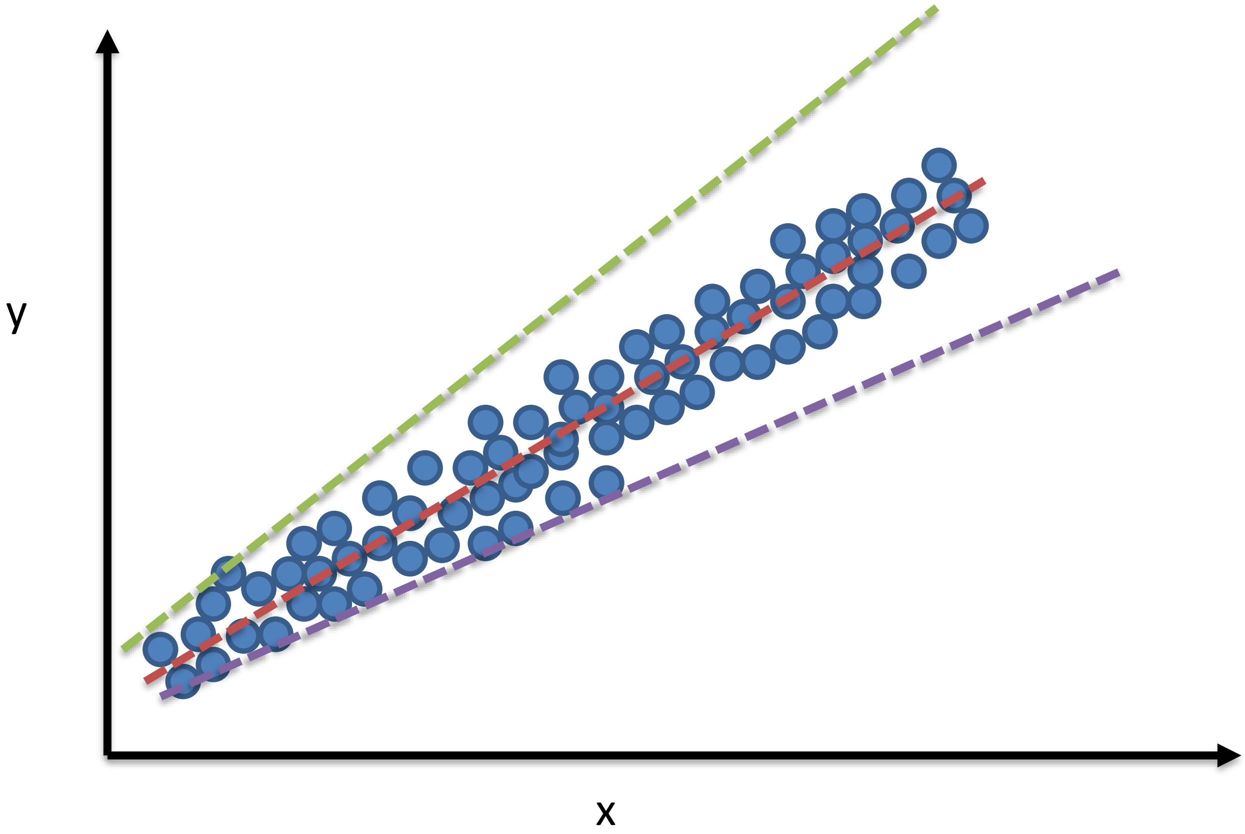
α = tasa de aprendizaje
(learning rate)

# Regresión Lineal

- ¿Qué es?

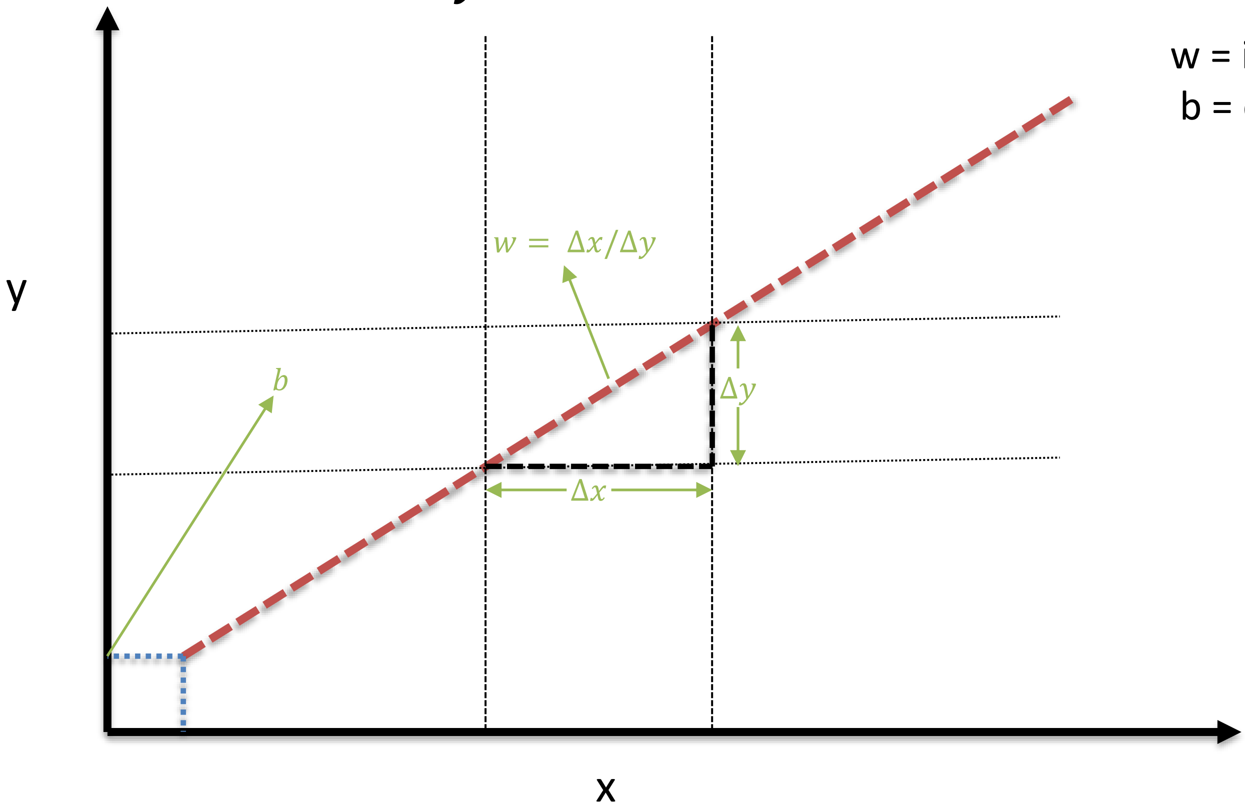
# Regresión Lineal

- ¿Qué es?



# Ecuación de una recta

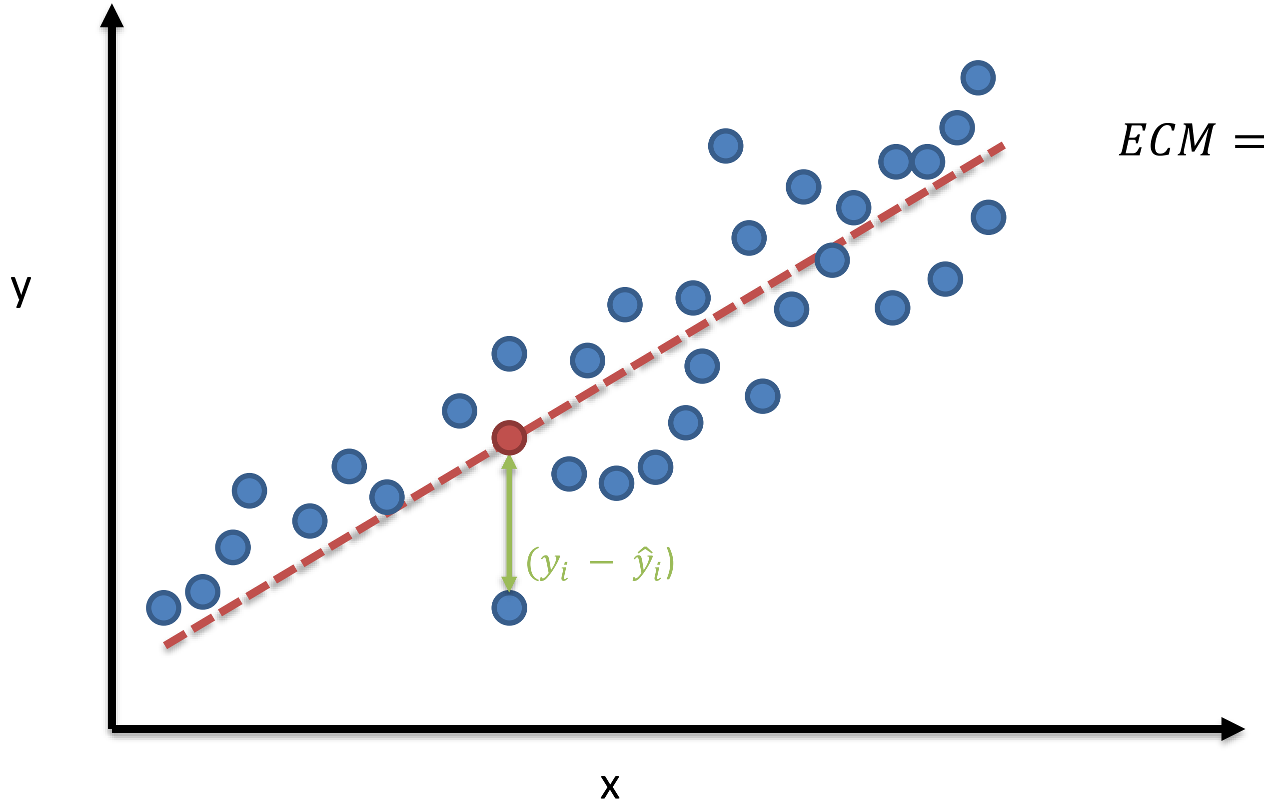
$$y = wx + b$$




w = inclinación que tiene la línea recta
b = donde corta la línea en y

- ¿Cómo se obtiene esto?

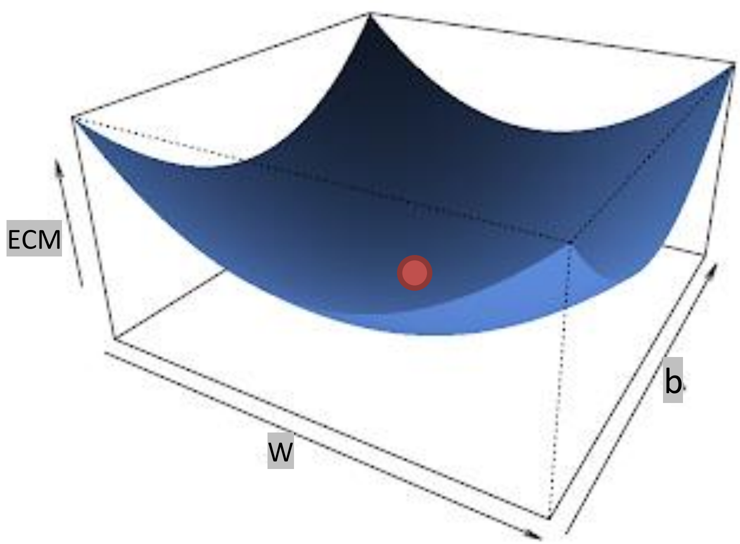
# Función de costo o perdida




$$ECM = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

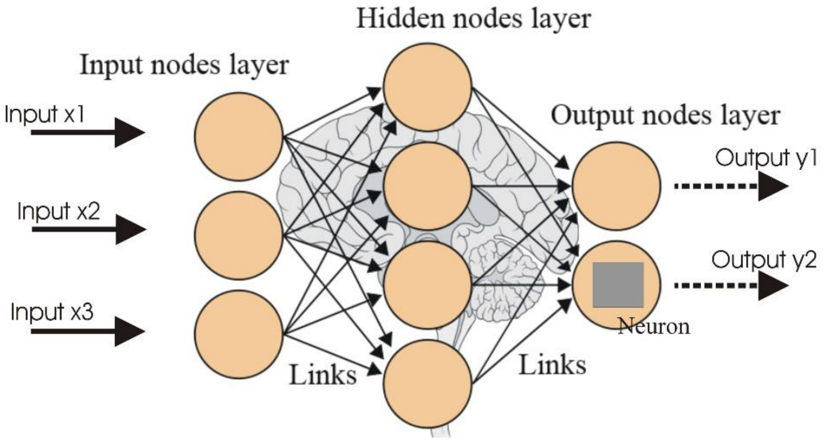
# Cálculo de w y b usando el gradiente descendente

- $ECM = \frac{1}{N} \sum_{i=1}^{N} (y_i - wx_i - b)_2$



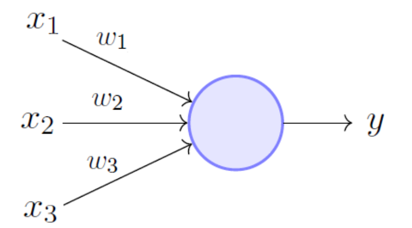
# Ejercicio/Ejemplo

# Redes Neuronales



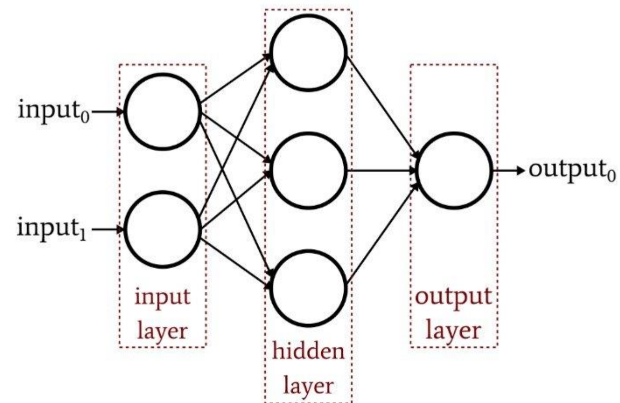
# Breve Historia

- Entre las décadas de 1950 y 1960 el científico Frank Rosenblatt, inspirado en el trabajo de Warren McCulloch y Walter Pitts creó el Perceptrón.

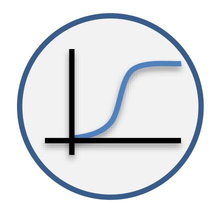



Perceptron Model (Minsky-Papert in 1969)

- En 1965 se amplia el perceptrón al perceptrón multi capa.

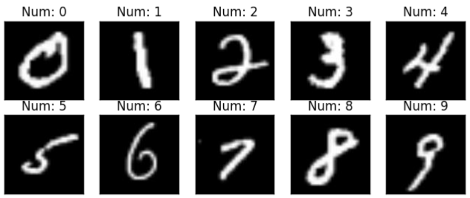


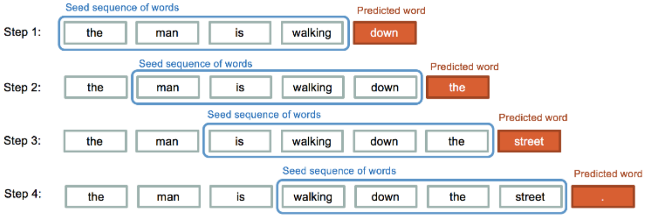

- The 1980's: Neuronas sigmoidales, redes feedforward y backpropagation.



# Áreas donde se aplican

- Reconocimiento de caracteres
- Reconocimiento de imágenes
- Reconocimiento de Voz
- Generación de texto
- Prevención de fraudes
- Predicción de la bolsa
- Conducción autónoma
- Predicción de enfermedades
- Análisis genético
- Etc…

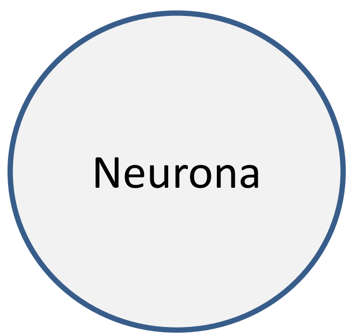


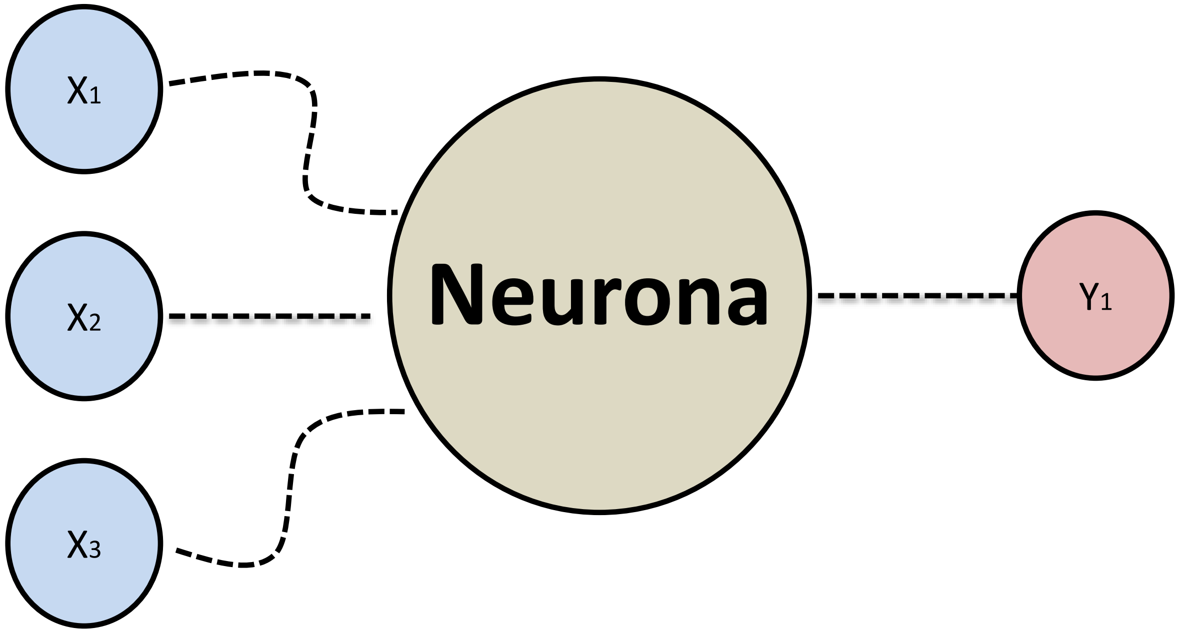

# ¿Cómo funcionan?

- La complejidad de estos sistemas emerge de la interacción de partes más simples trabajando conjuntamente.

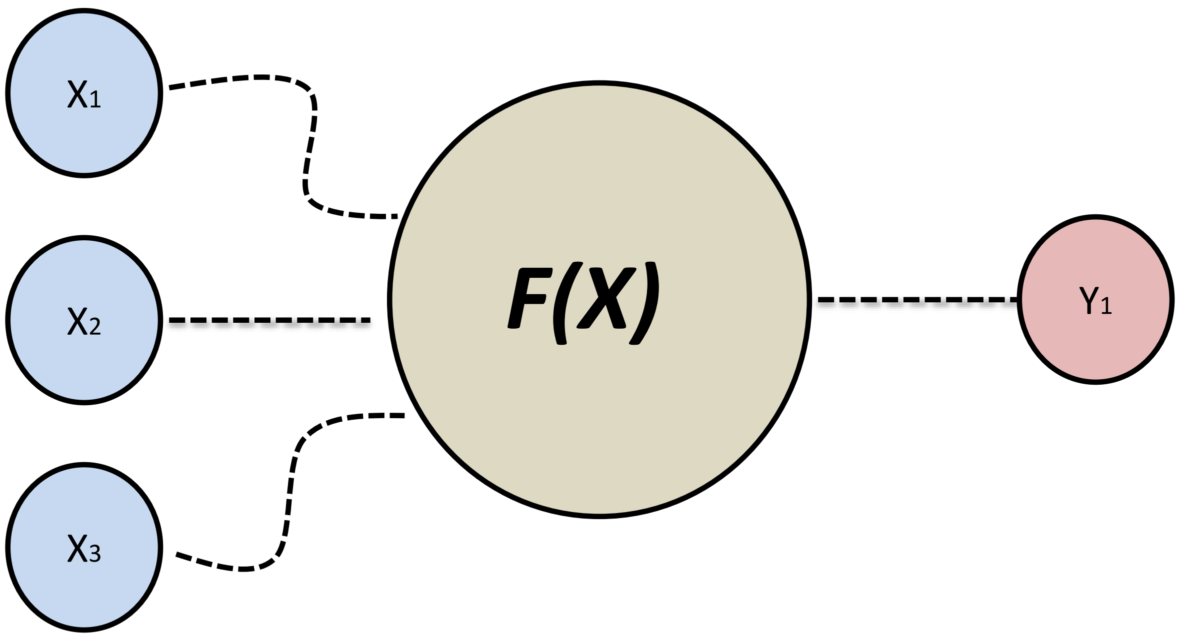
- A cada una de estas partes se le conoce como “Neurona”.



# ¿Qué es una Neurona?

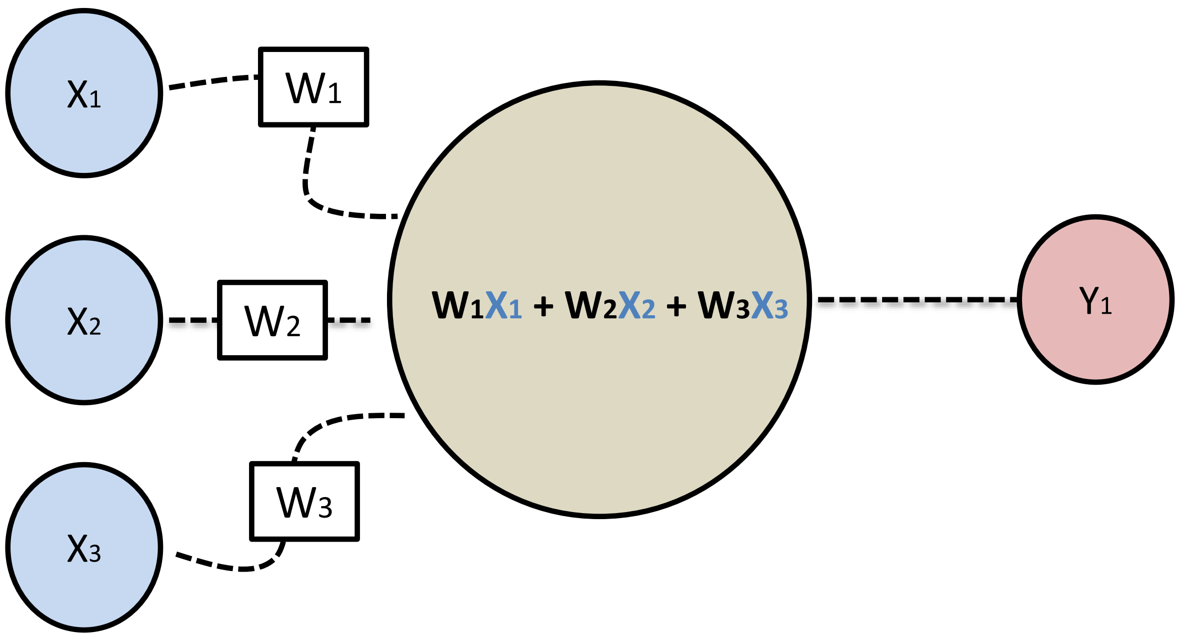


# ¿Qué es una Neurona?



# ¿Qué es una Neurona?

## SUMA PONDERADA

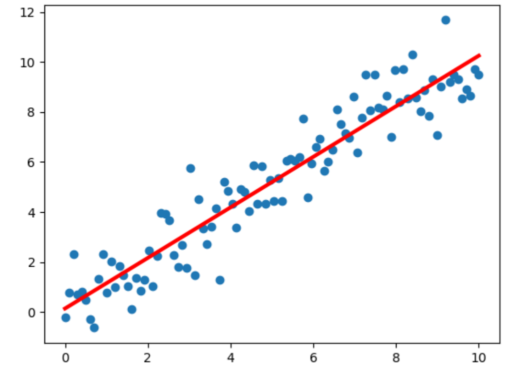


# ¿Qué es una Neurona?

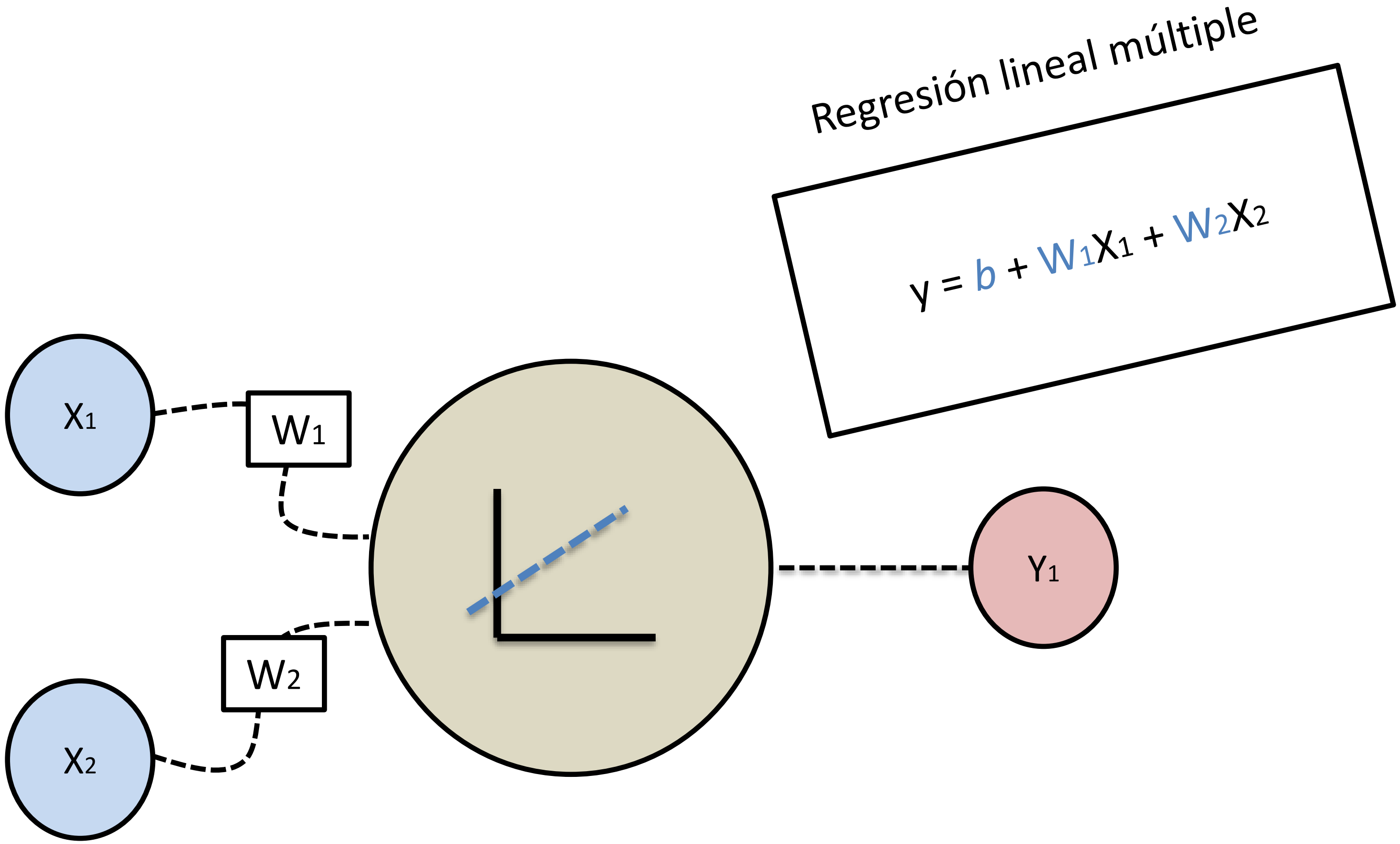
## Regresión Lineal

$W_1X_1 + W_2X_2 + W_3X_3$

$$y = b + W_1X_1 + W_2X_2$$

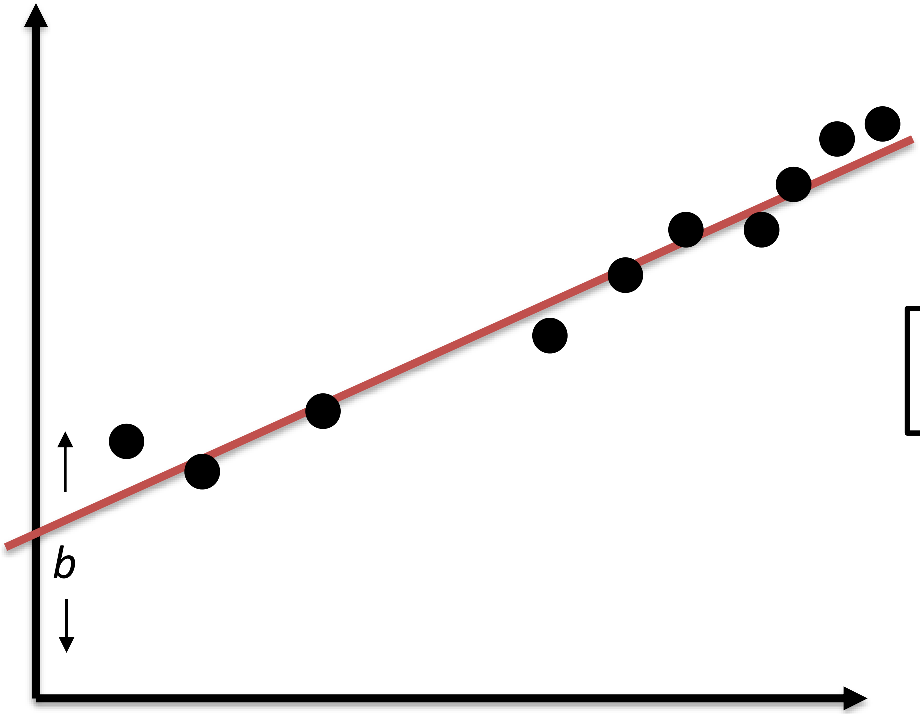


# ¿Qué es una Neurona?




$$y = W_1X_1 + W_2X_2$$

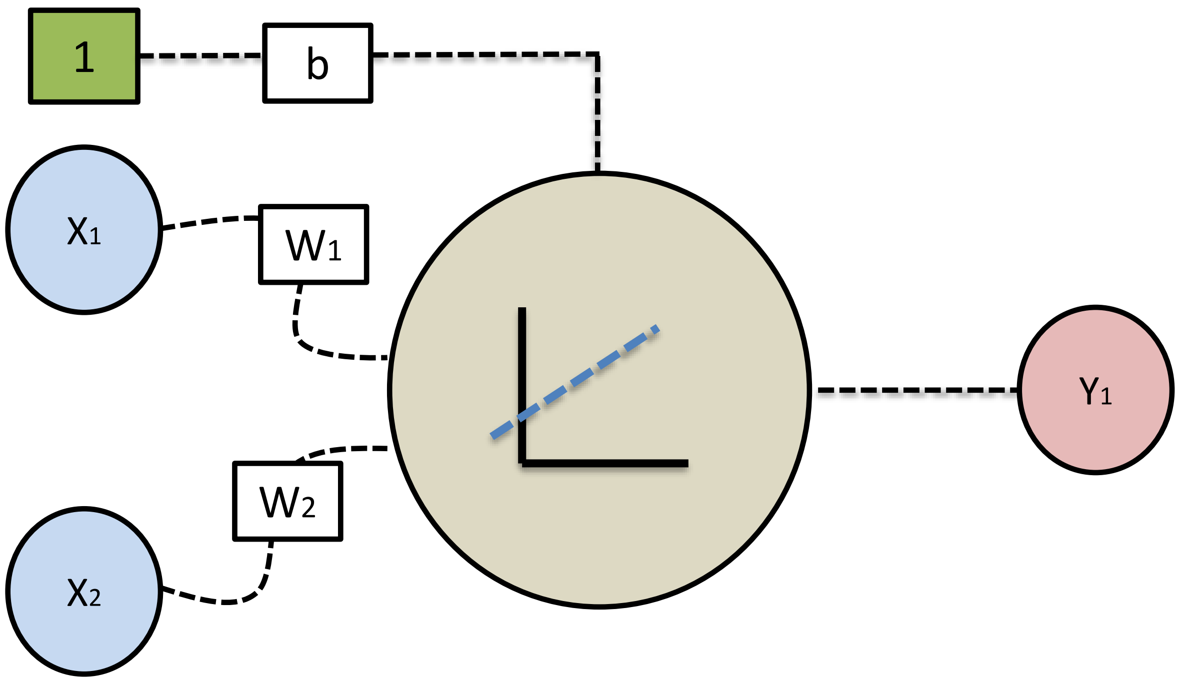
# Regresión Lineal




$y = b + W_1X_1$

Termino independiente

# Agregando el termino independiente




$$y = W_1X_1 + W_2X_2 + b$$

# Ejemplo aplicando un perceptrón

- ¿Qué se necesita para un buen sábado por la noche?

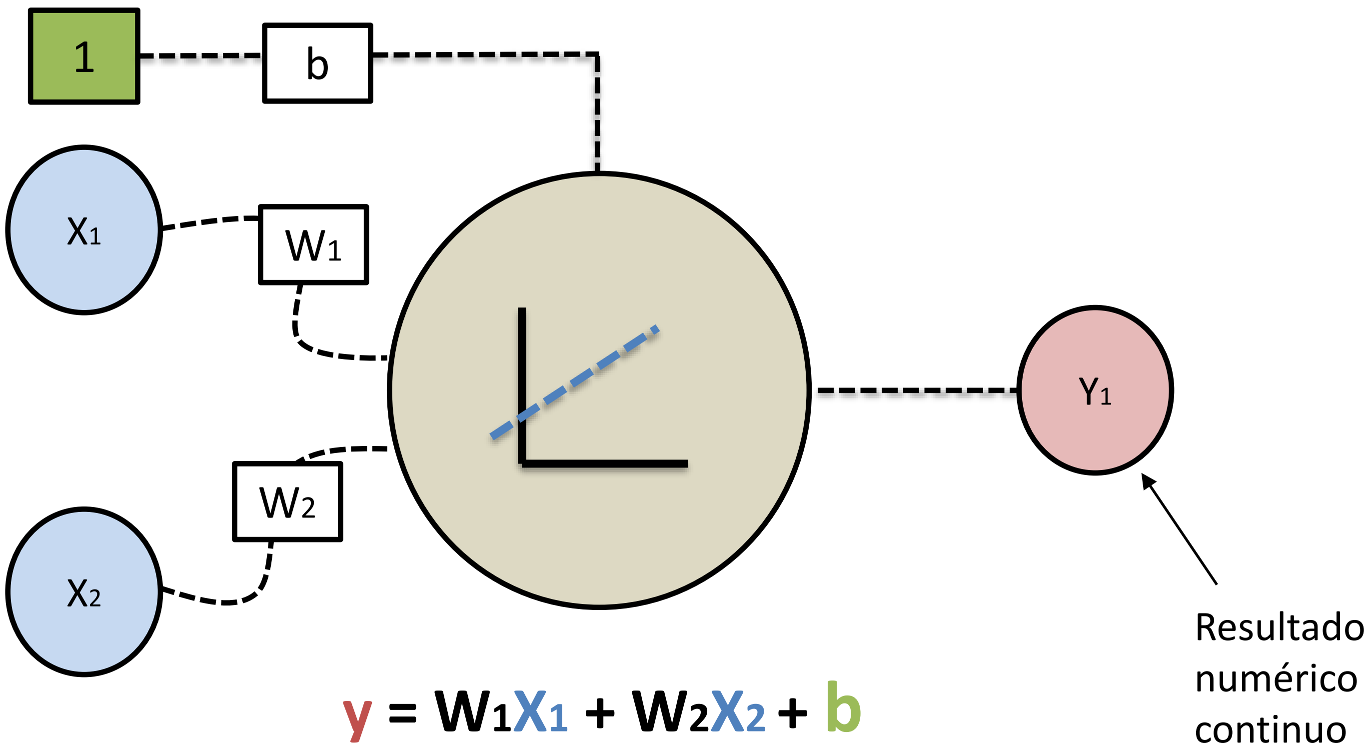
# Ejemplo aplicando un perceptrón

- ¿Qué se necesita para un buen sábado por la noche?


- Variables Binarias

| | 1 | 0 |
|---|---|---|
| VG: $X_1$ |  |  |
| Pizza: $X_2$ |  |  |
| Epic Night: $Y_1$ |  |  |

# Nuestra neurona funciona como modelo de regresión lineal




$$y = W_1X_1 + W_2X_2 + b$$

# ¿Qué podemos hacer?

- WX <= UMBRAL -> Y = 0
- WX > UMBRAL -> Y = 1

- BIAS = -UMBRAL

# ¿Qué podemos hacer?

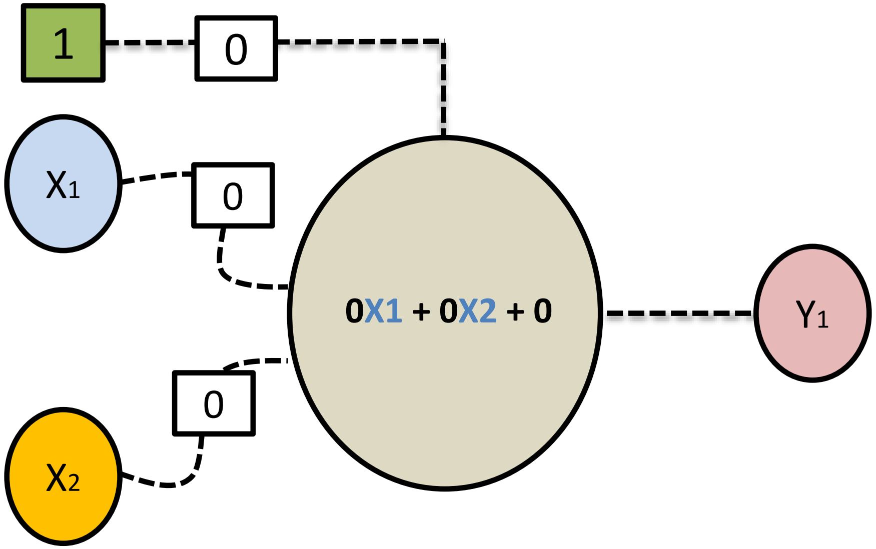
- BIAS = -UMBRAL

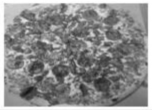
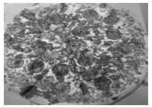
- WX + b <= 0 -> Y = 0
- WX + b > 0 -> Y = 1

# Modelar nuestro plan

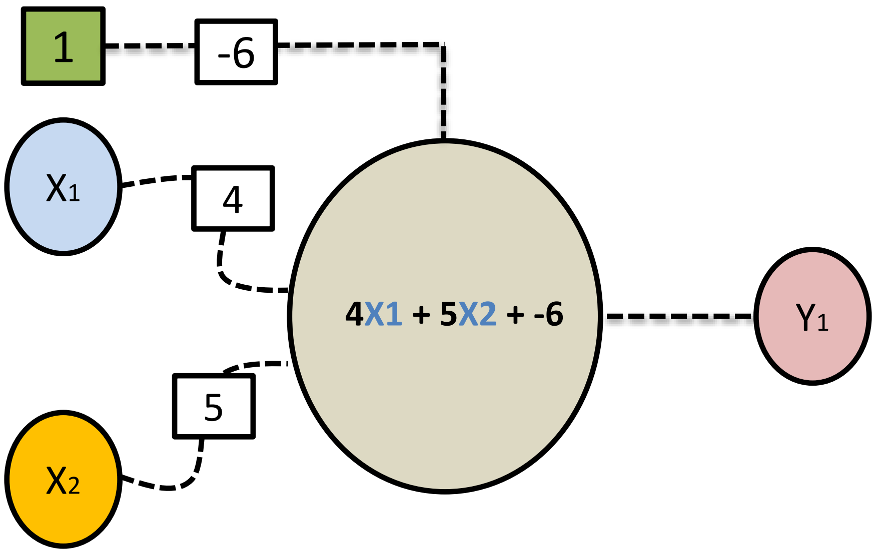
| X1 | X2 | Objetivo | Y |
|---|---|---|---|
|  |  |  | 0 |
|  |  |  | 0 |
|  |  |  | 0 |
|  |  |  | 0 |

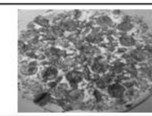
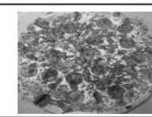
# Modelar nuestro plan

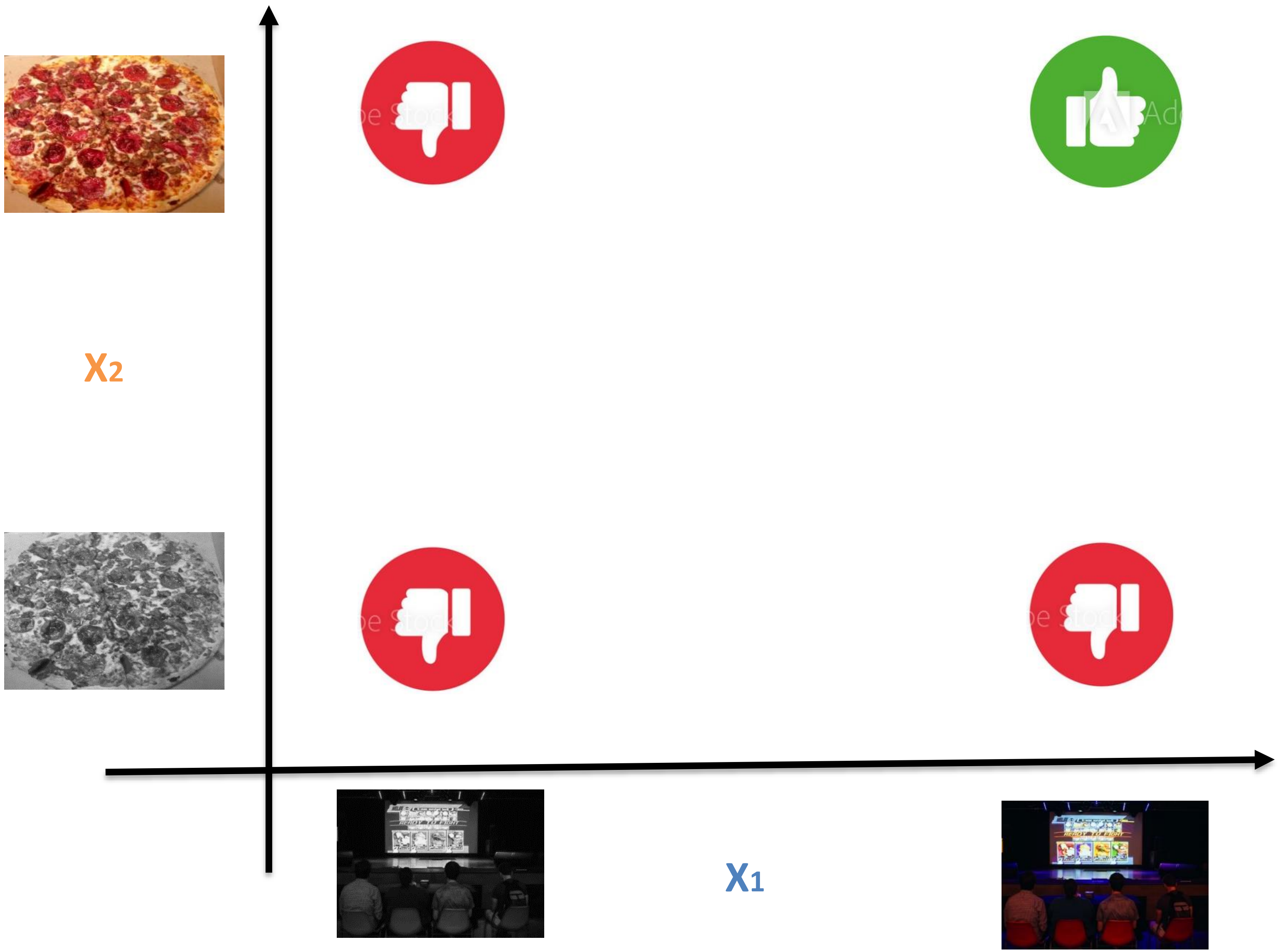



| X1 | X2 | Objetivo | Y |
|---|---|---|---|
|  |  |  | 0 |
|  |  |  | 0 |
|  |  |  | 0 |
|  |  |  | 0 |

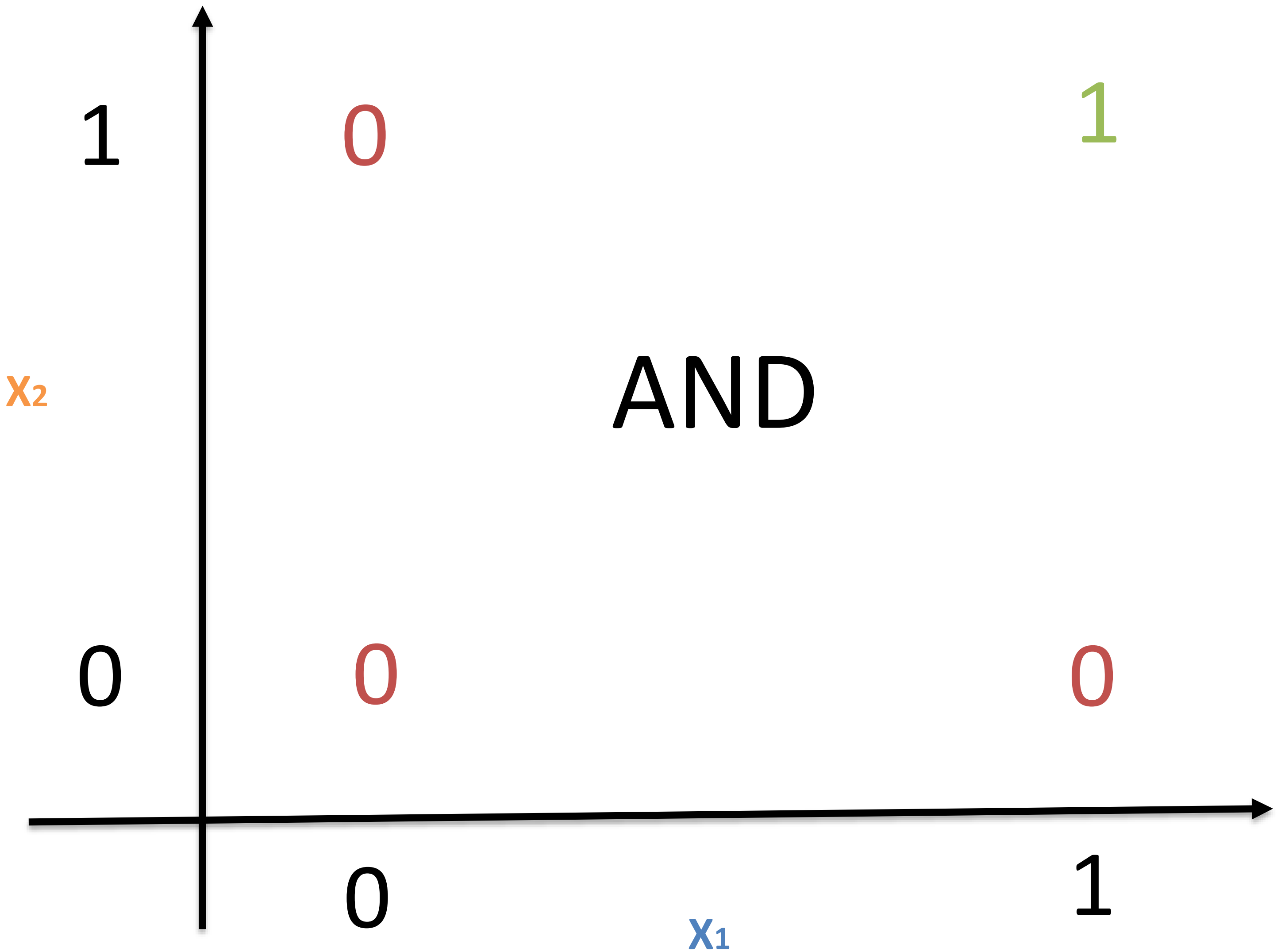
# Modelar nuestro plan



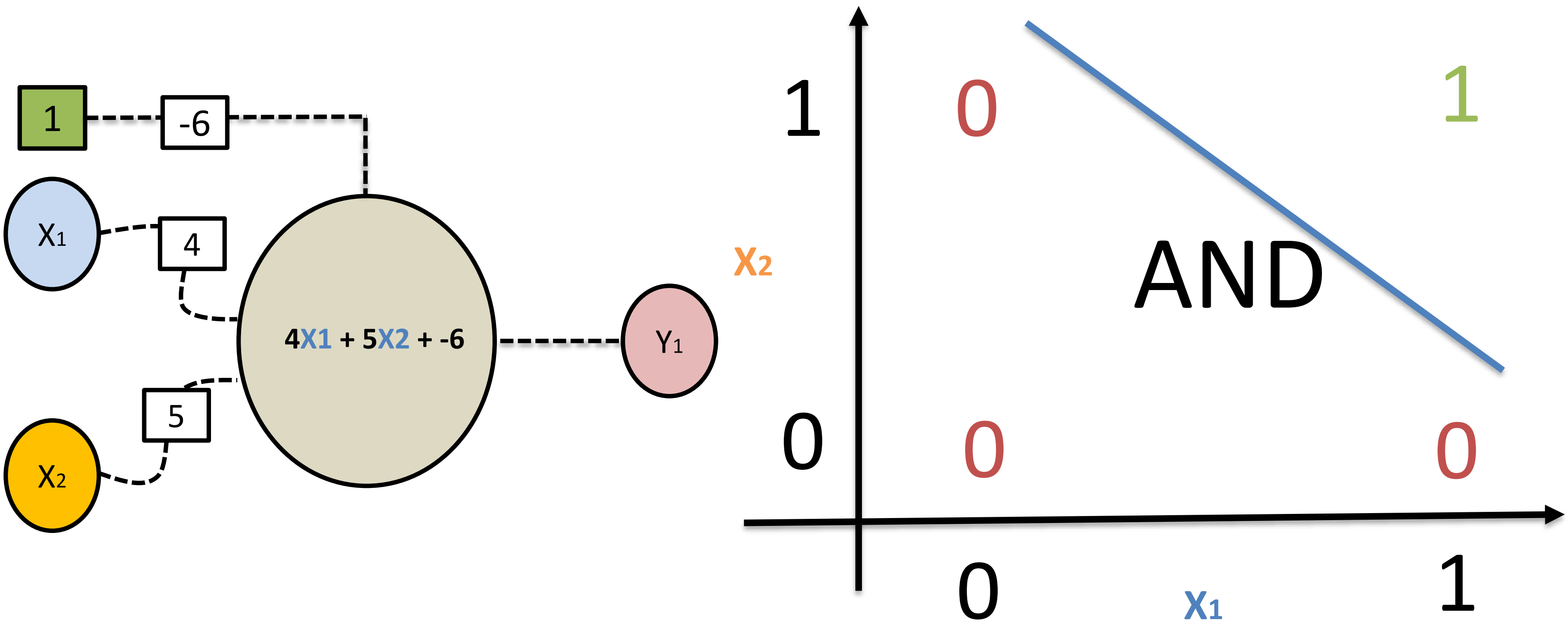

| X1 | X2 | Objetivo | Y |
|---|---|---|---|
|  |  |  | -6 |
|  |  |  | -2 |
|  |  |  | -1 |
|  |  |  | 3 |

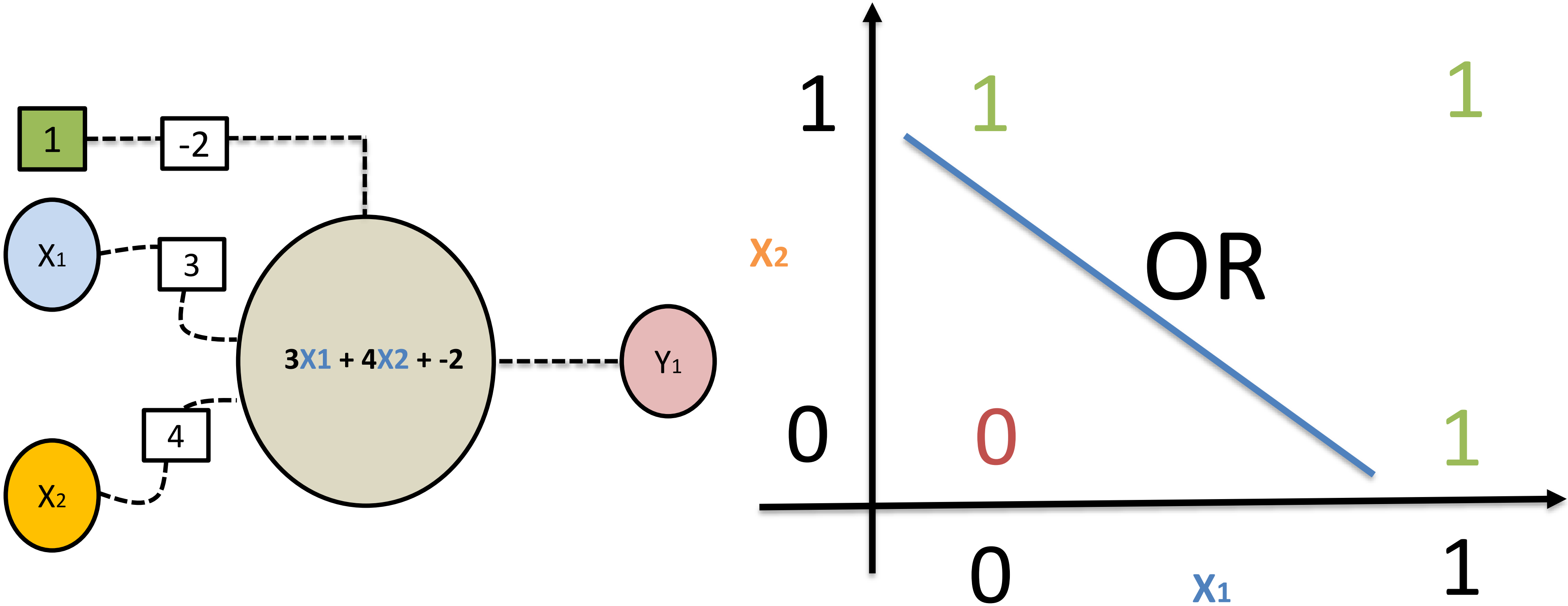


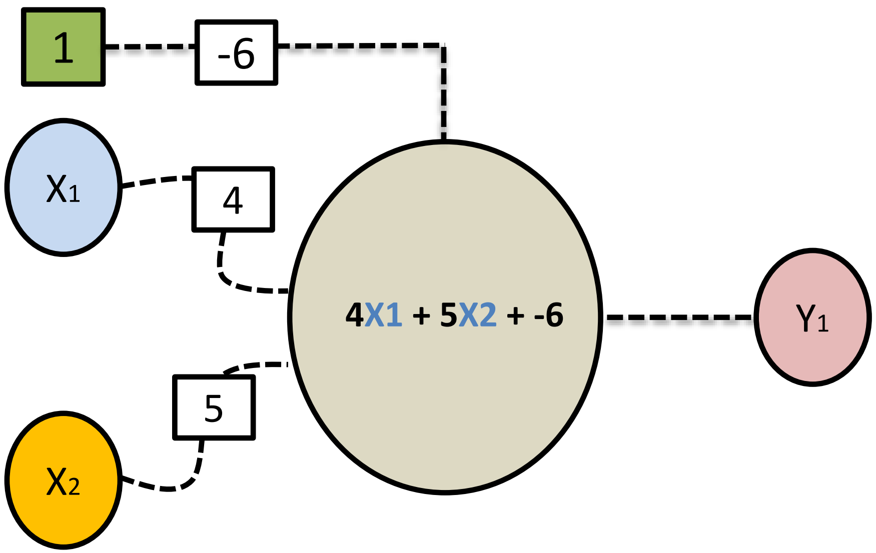
$X_2$
$X_1$

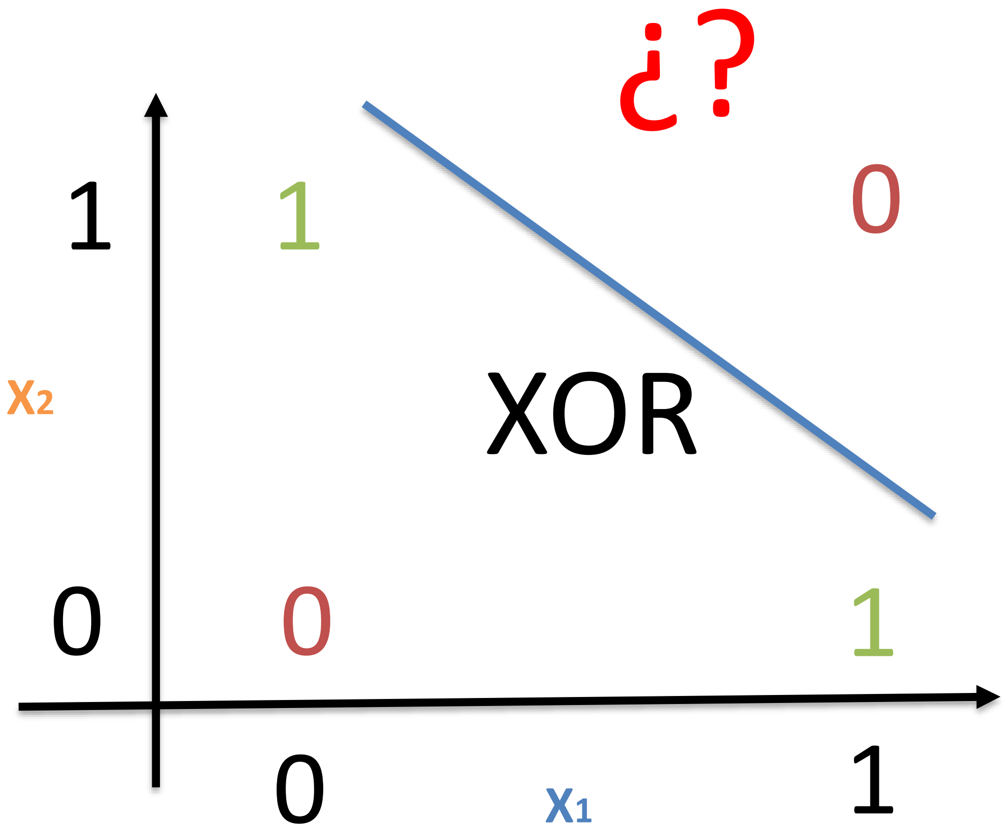


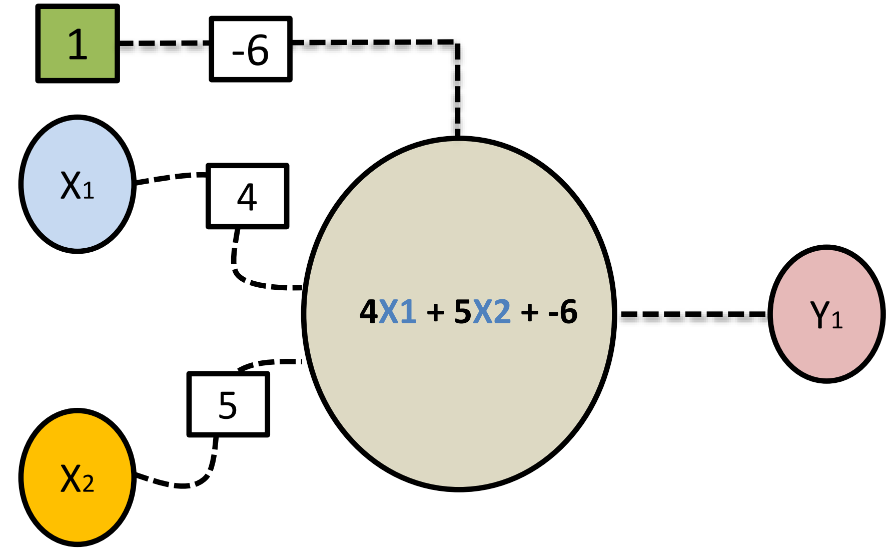
AND
1
0
0
1
0
0
0
1
X2
X1

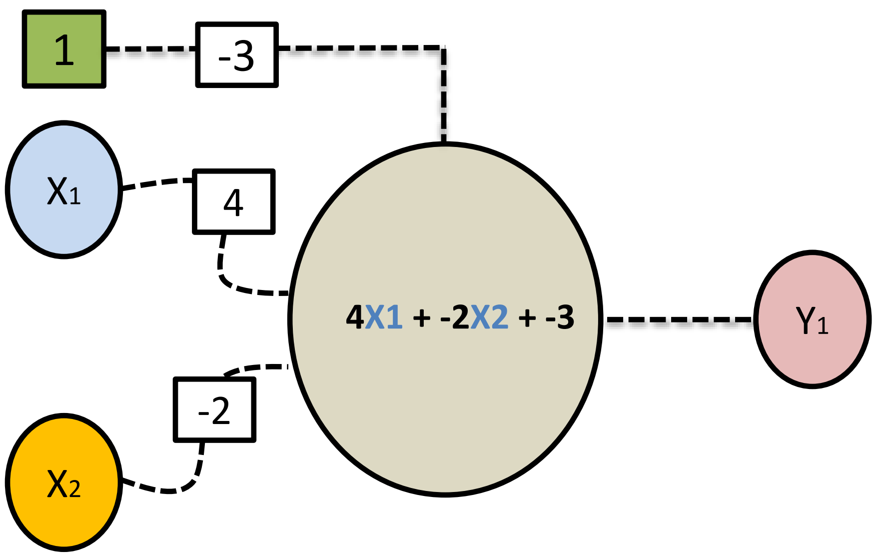

1
-6
X1
4
4X1 + 5X2 + -6
Y1
5
X2
1
0
1
X2
AND
0
0
0
0
X1
1

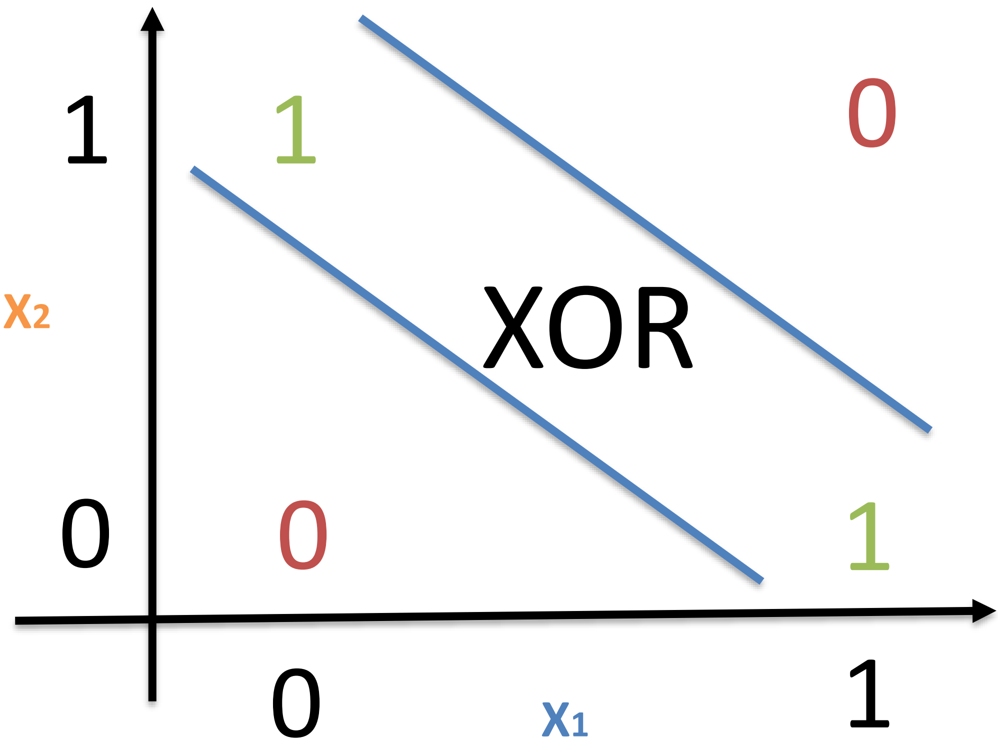

1
-2
X1
3
3X1 + 4X2 + -2
Y1
4
X2
1
1
1
X2
OR
0
0
1
0
X1
1


1
-6
X1
4
4X1 + 5X2 + -6
Y1
5
X2


¿?
1
1
0
X2
XOR
0
0
1
0
X1
1


1
-6
X1
4
4X1 + 5X2 + -6
Y1
5
X2

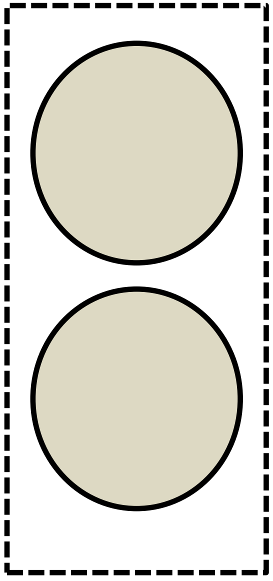

1
-3
X1
4
4X1 + -2X2 + -3
Y1
-2
X2


1
1
0
X2
XOR
0
0
1
0
X1
1

# La red neuronal

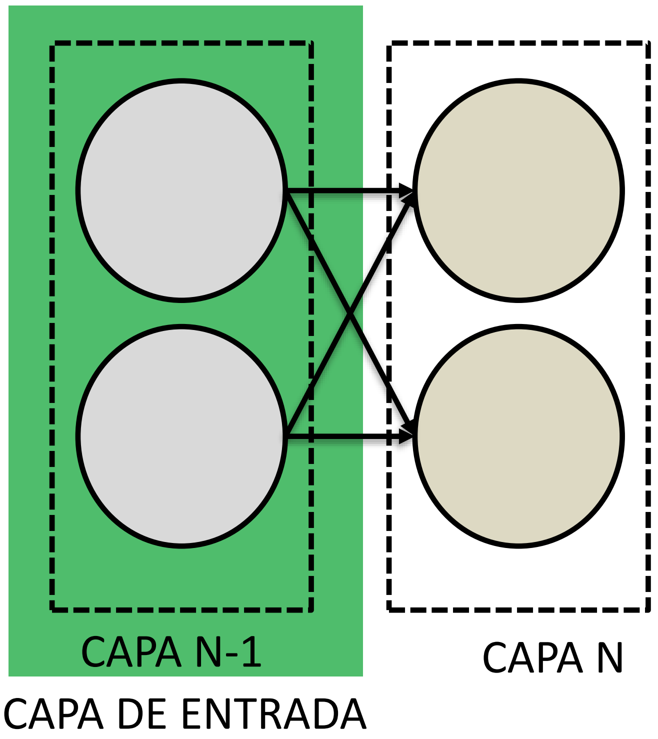
- Comprender la red en una red neuronal

# La red neuronal

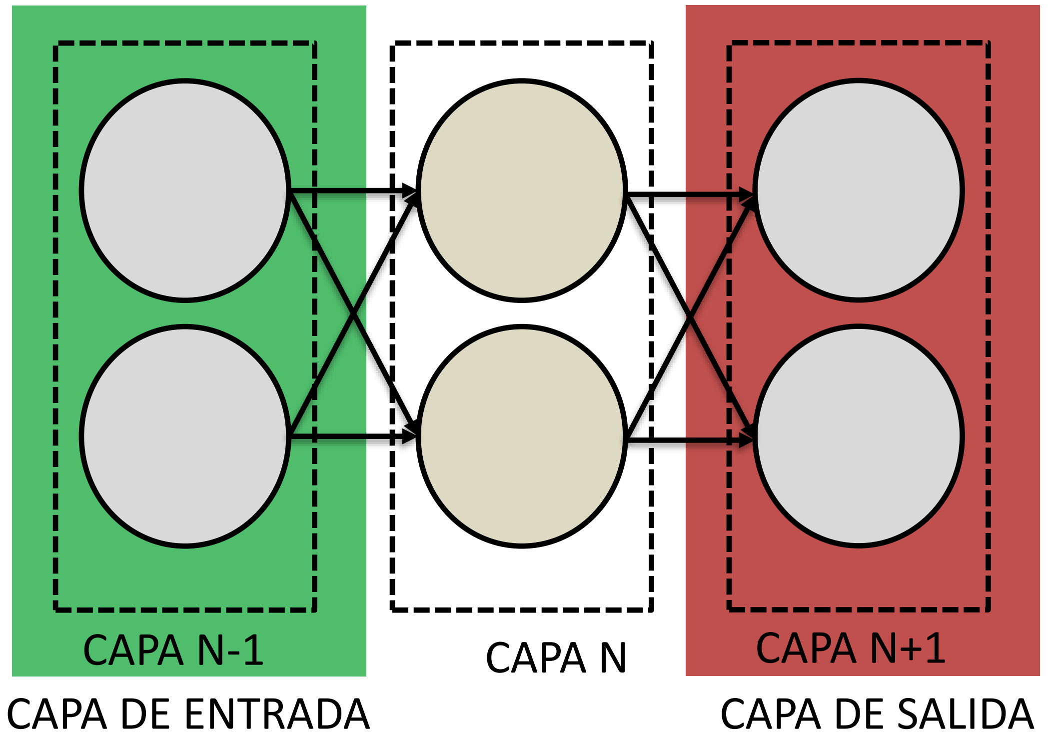


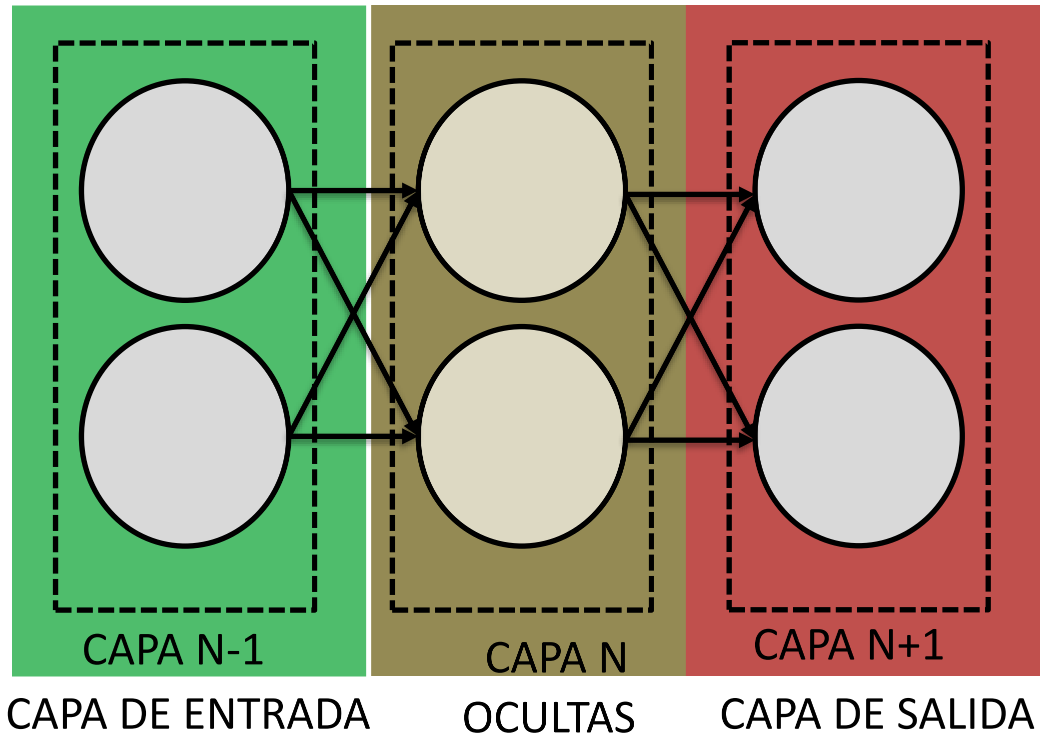
CAPA N

# Información de entrada



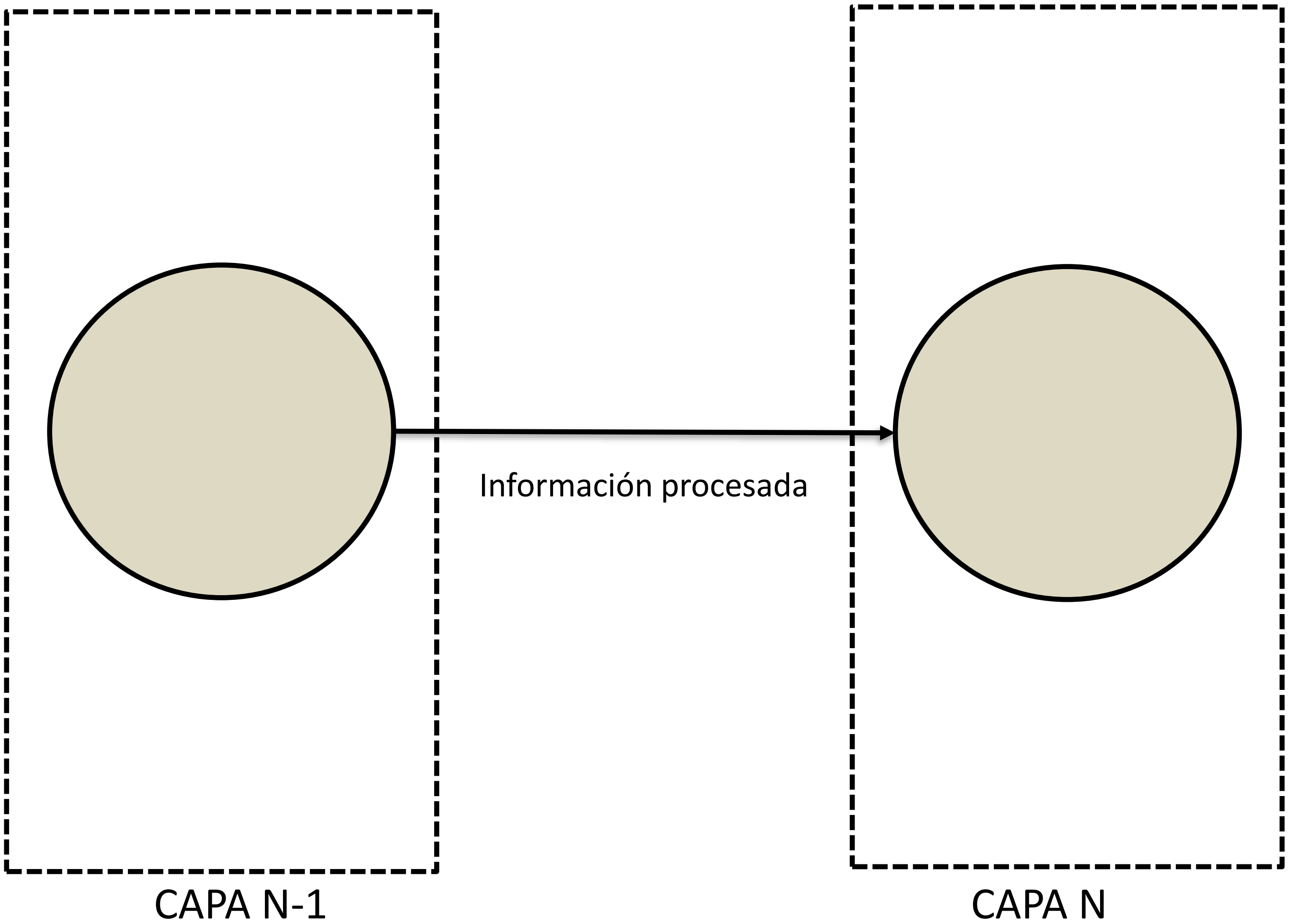
# Información de salida




CAPA N-1
CAPA N
CAPA N+1
CAPA DE ENTRADA
OCULTAS
CAPA DE SALIDA

# La red neuronal

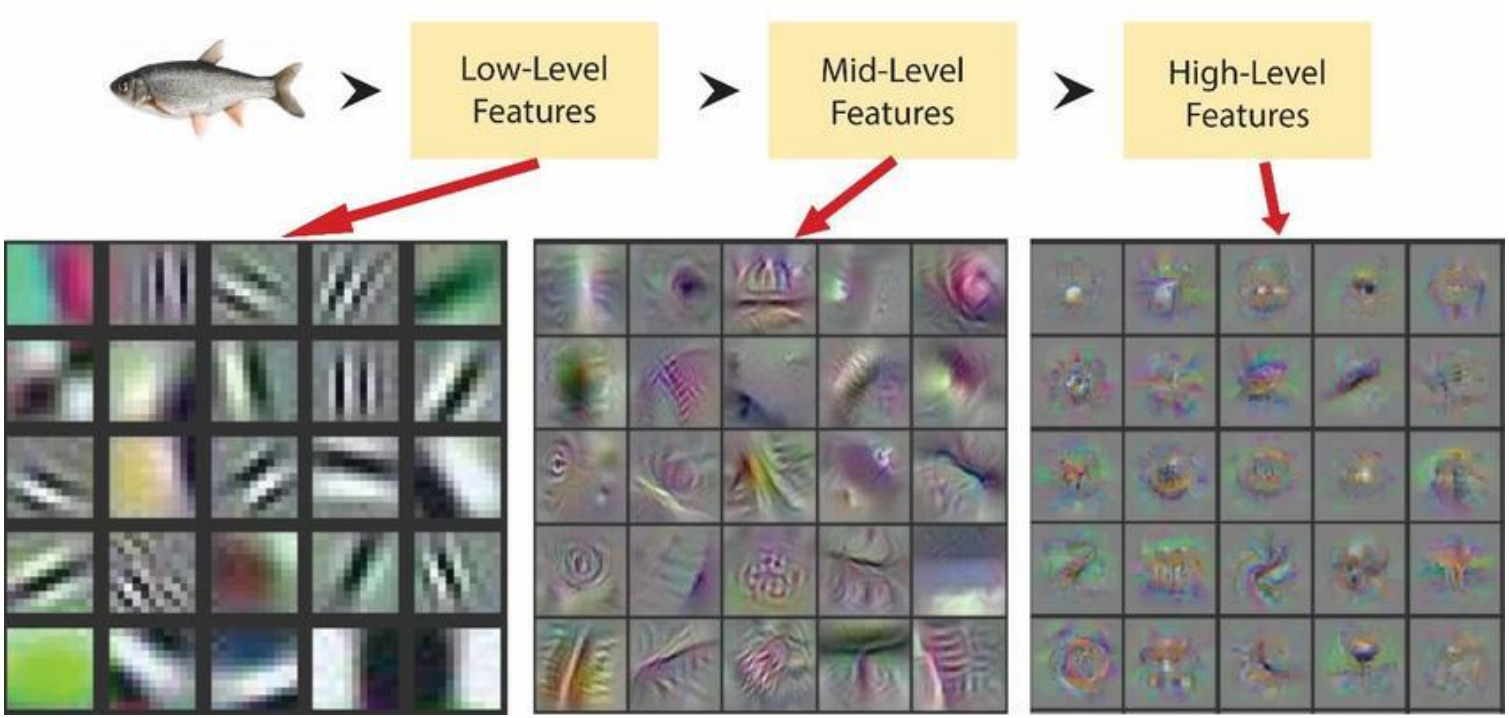
- Ventajas de juntar neuronas



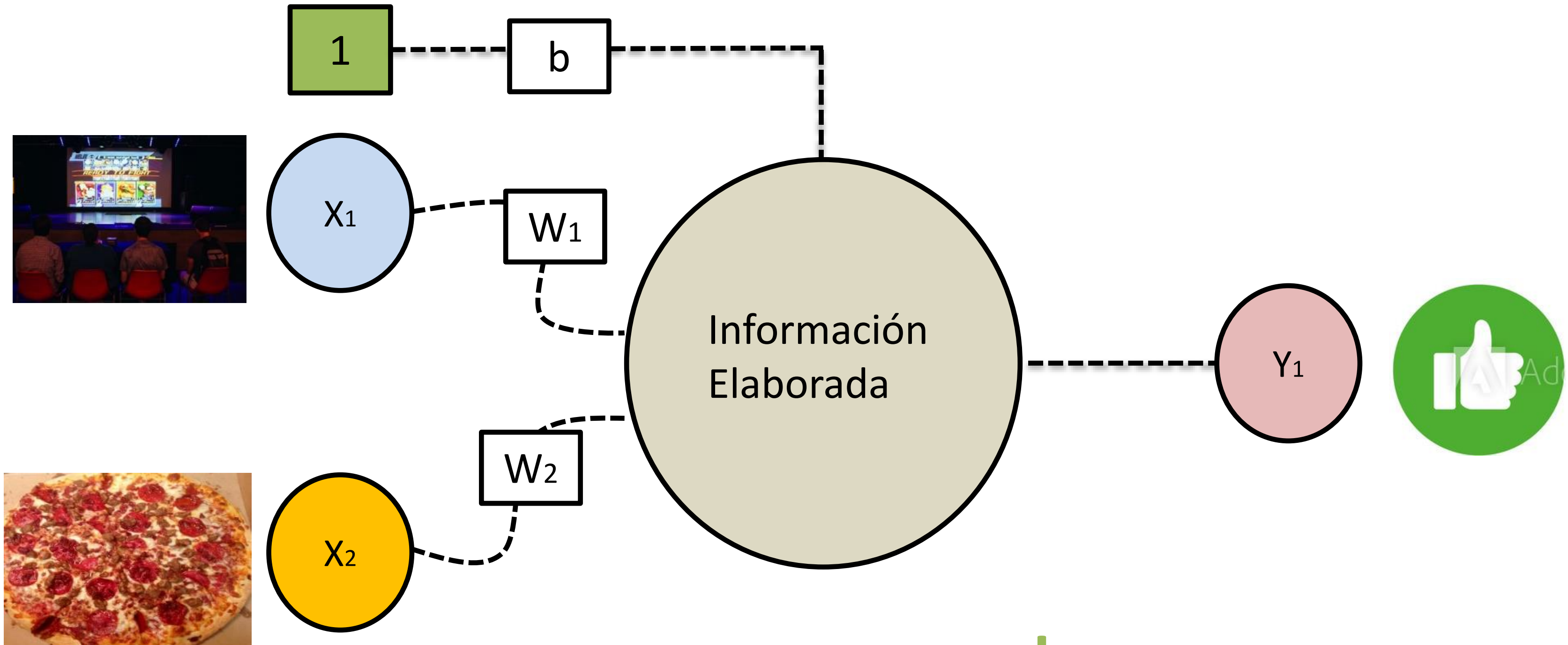
# ¿Qué permite esto?

# ¿Qué permite esto?

- CONOCIMIENTO JERARQUIZADO



# Ejemplo de conocimiento jerárquico

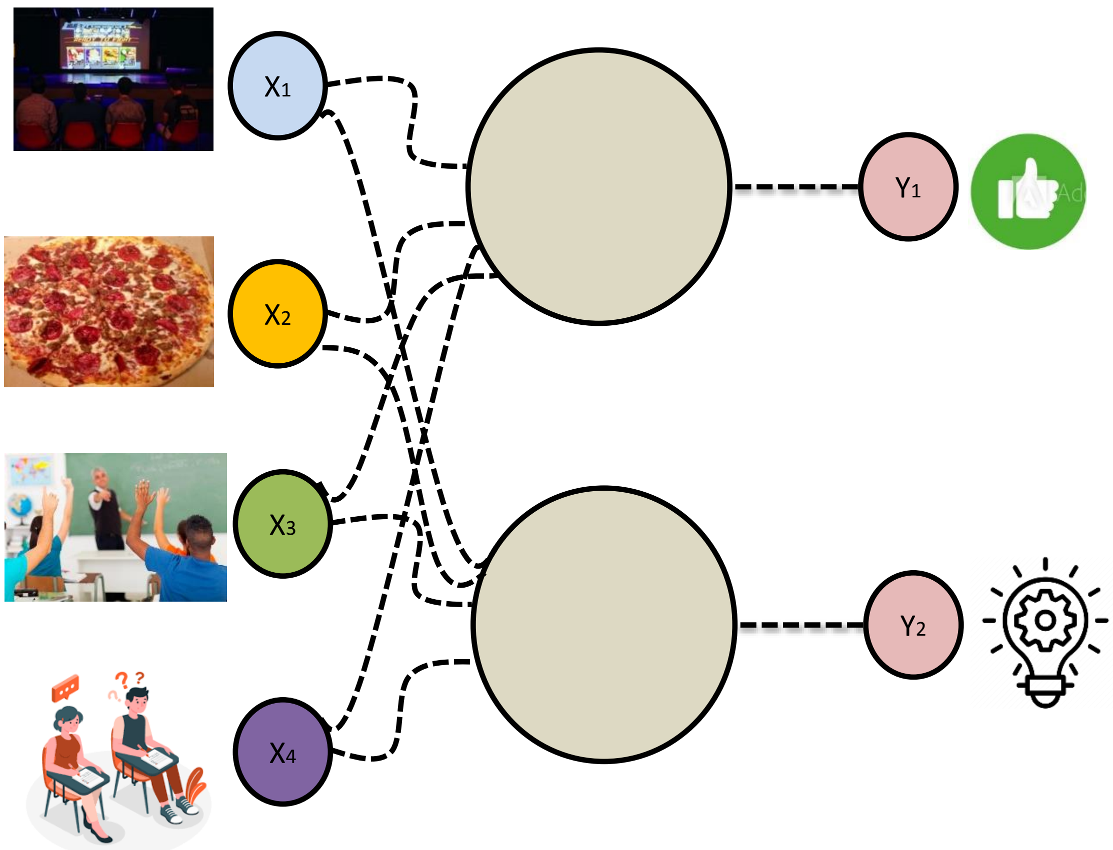



$$y = W_1X_1 + W_2X_2 + b$$

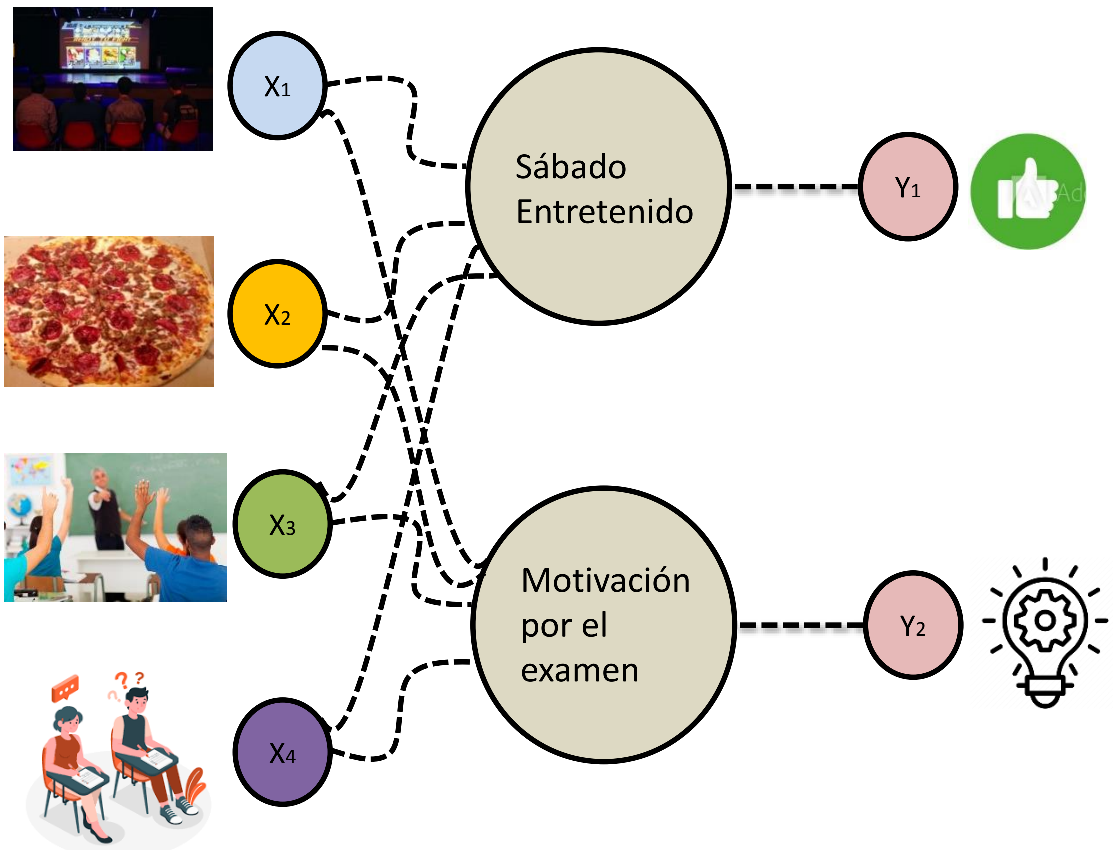
# ¿Qué tal si elaboramos información compleja?

- En lugar de saber si pasaremos un buen sábado por la noche
- Queremos saber si obtendremos una buena nota en el examen del lunes

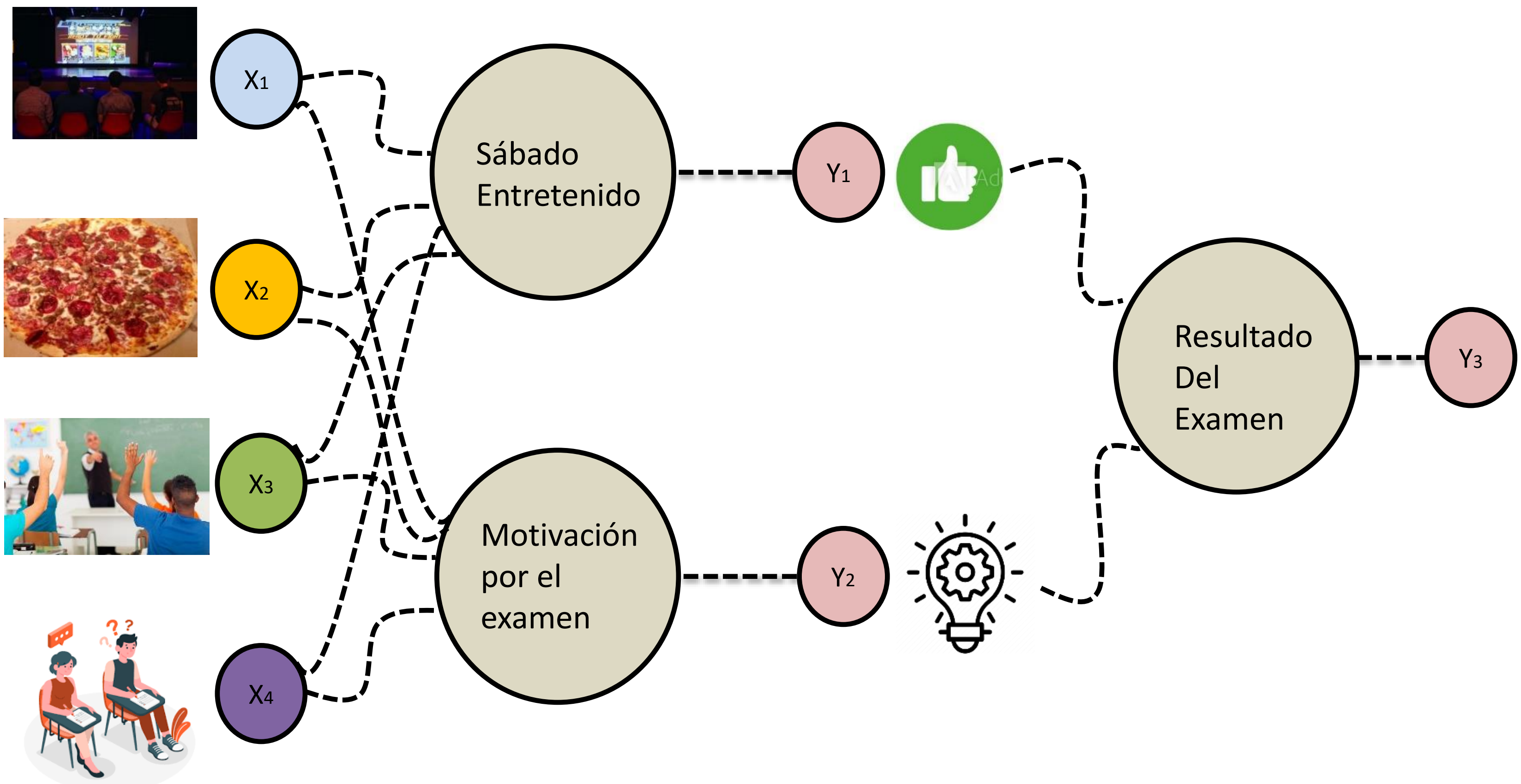
# Ejemplo de conocimiento jerárquico



# Ejemplo de conocimiento jerárquico



# Ejemplo de conocimiento jerárquico

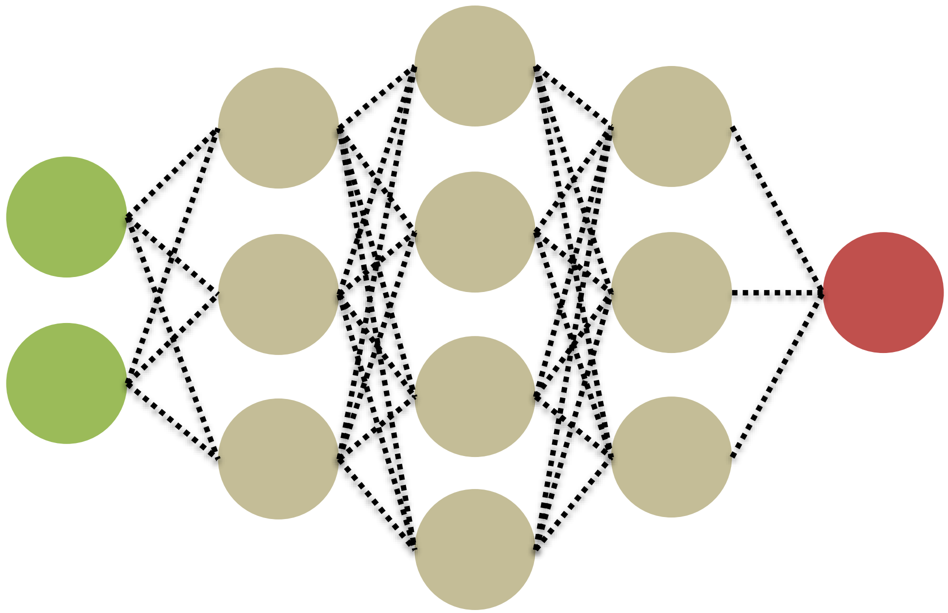


# Puntos Importantes

- Entre más capas añadimos más complejidad agregamos.
- El aprendizaje jerarquizado en mayor profundidad es lo que le da su nombre al Deep Learning

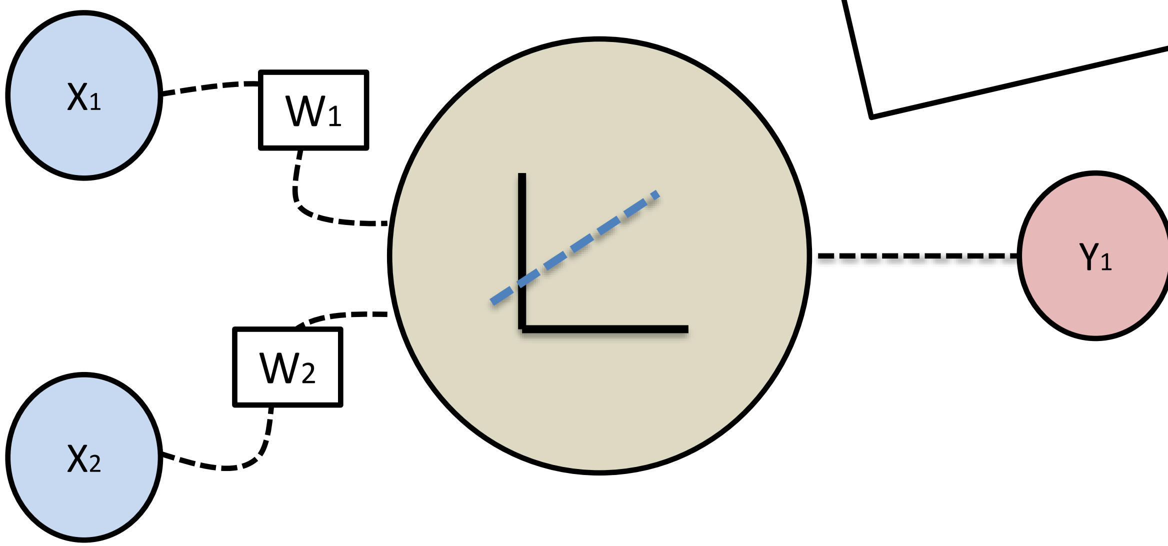
- Pero aun queda un punto muy importante…

- Conectamos varias neuronas de manera secuencial



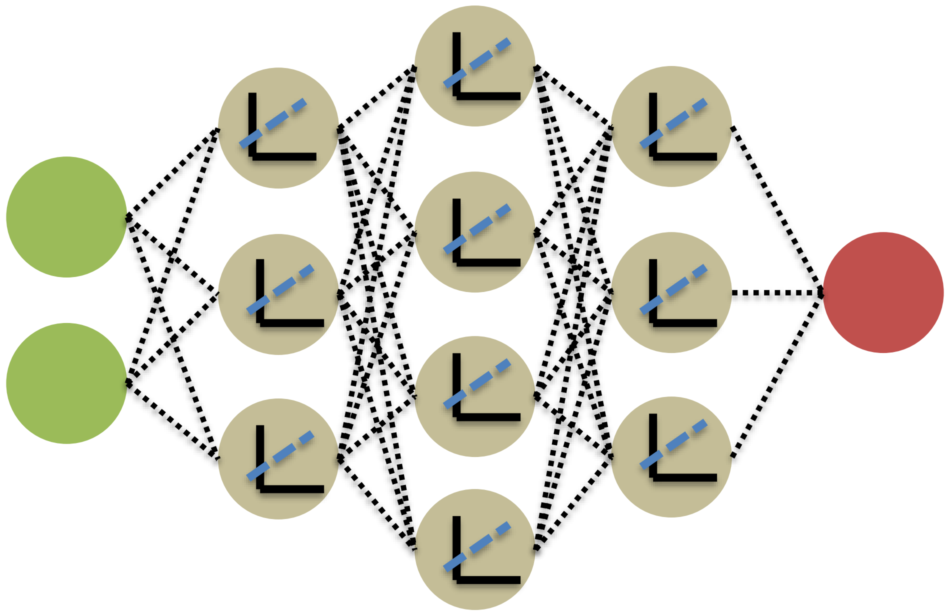
Regresión lineal múltiple

$$y = W_0 + W_1X_1 + W_2X_2$$

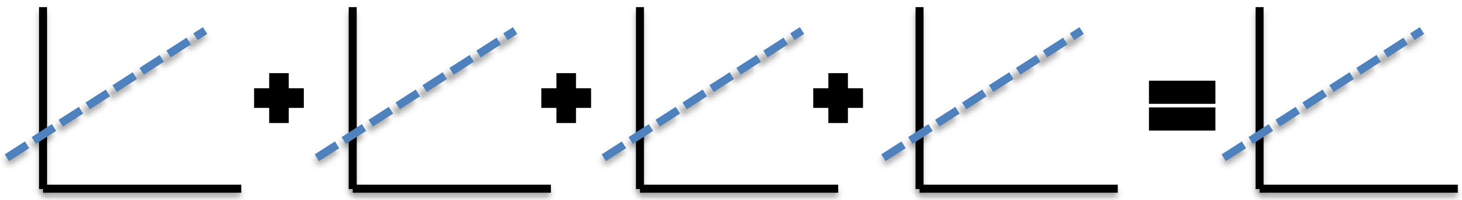



$$y = W_1X_1 + W_2X_2$$

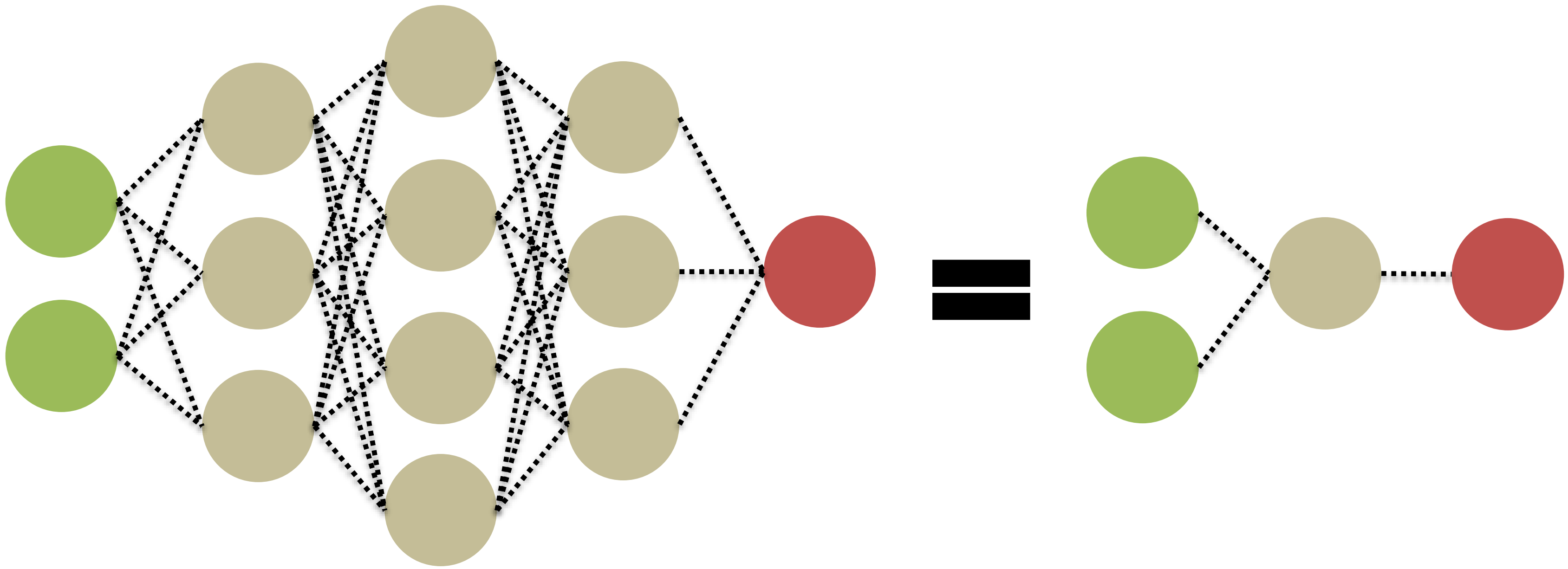
- Concatenar varias regresiones lineales



# ¿Qué problema tenemos?

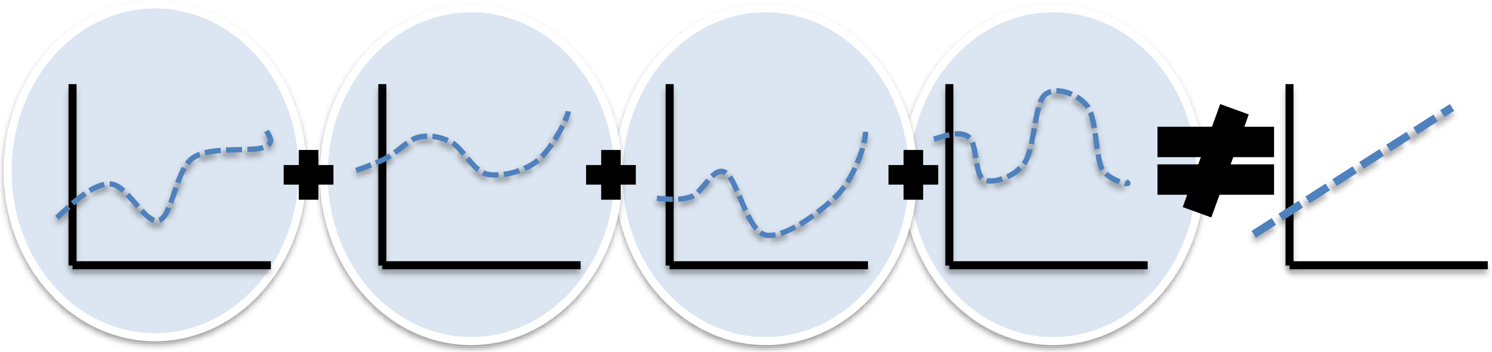


# ¿Qué problema tenemos?



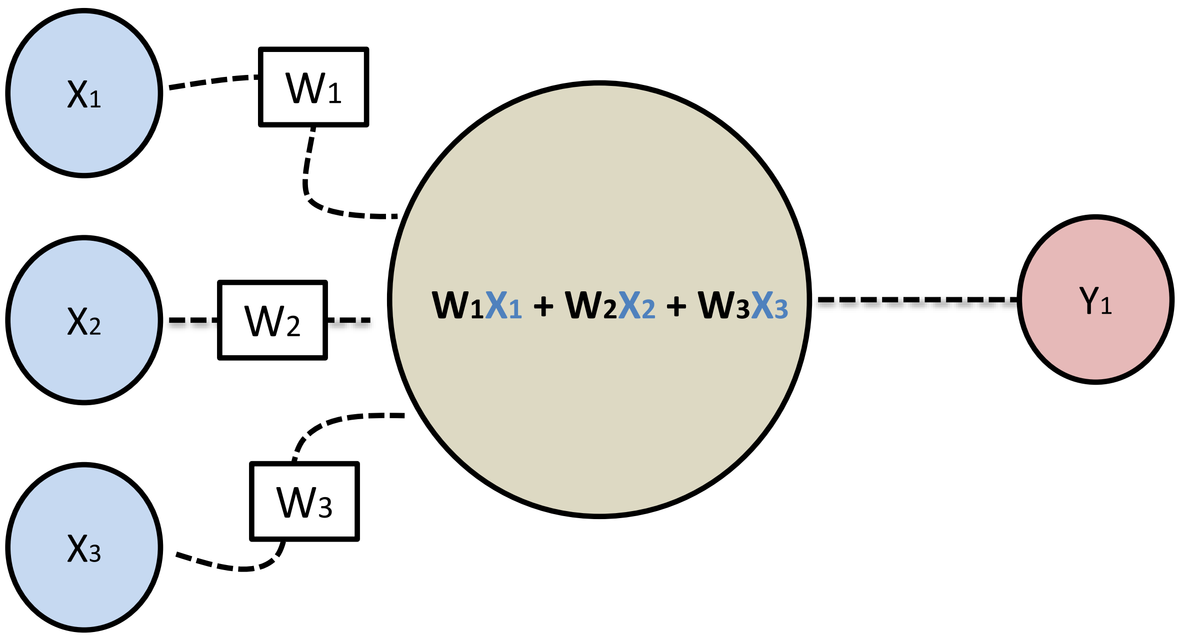
# ¿Qué podemos hacer?

- Agregar alguna manipulación no lineal



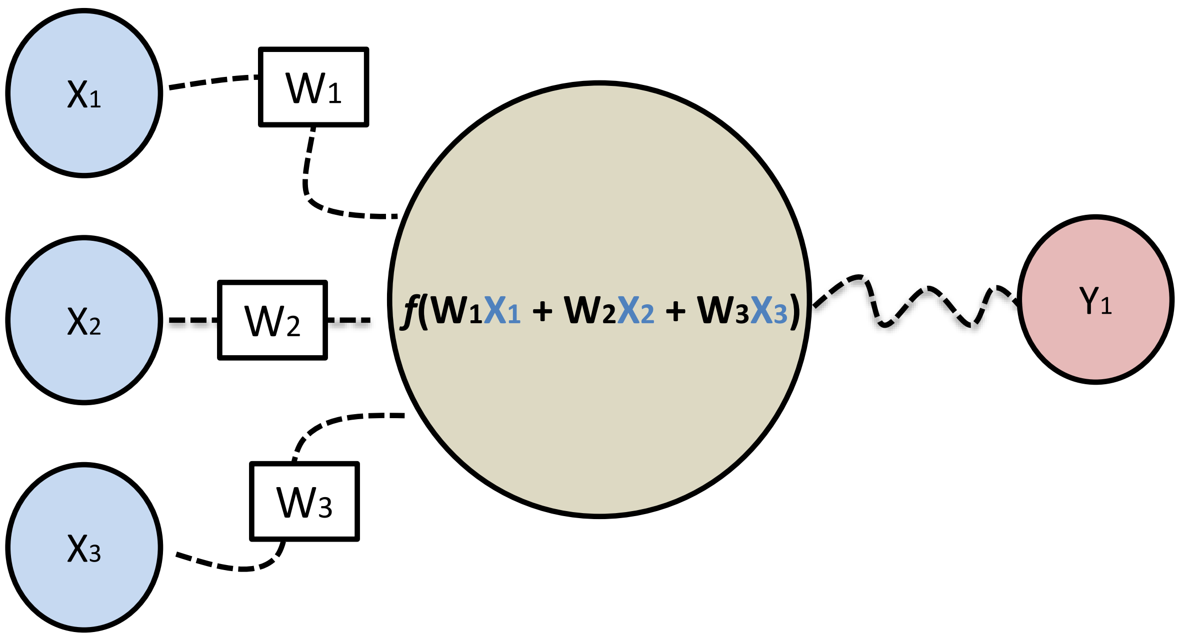
# Funciones de activación

- Teníamos una suma ponderada de las entradas



# Funciones de activación

- Ahora lo pasamos por una función de activación




Función de activación : *f(x)*

# ¿Cómo son estas funciones de activación?

# ¿Cómo son estas funciones de activación?

- WX <= UMBRAL -> Y = 0
- WX > UMBRAL -> Y = 1

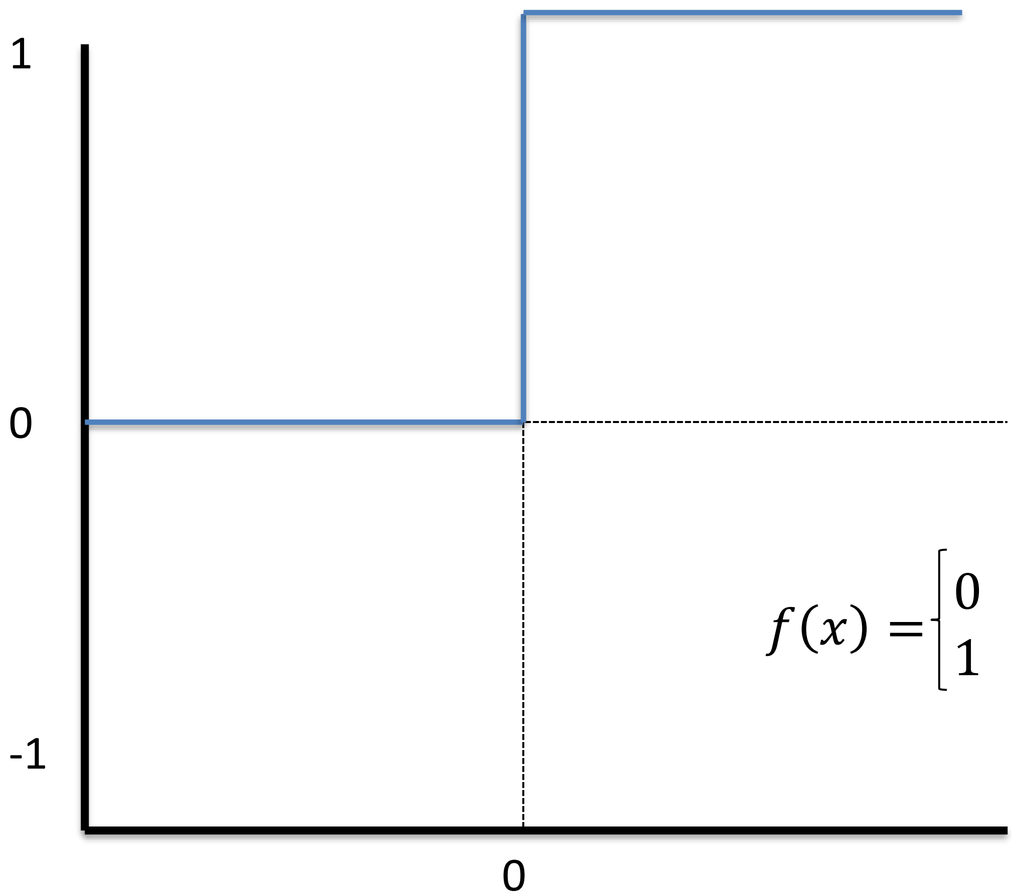
# ¿Cómo son estas funciones de activación?

- WX <= UMBRAL -> Y = 0
- WX > UMBRAL -> Y = 1

- Función de activación
  - $f$(WX) -> Y = {0,1}

# Función de activación

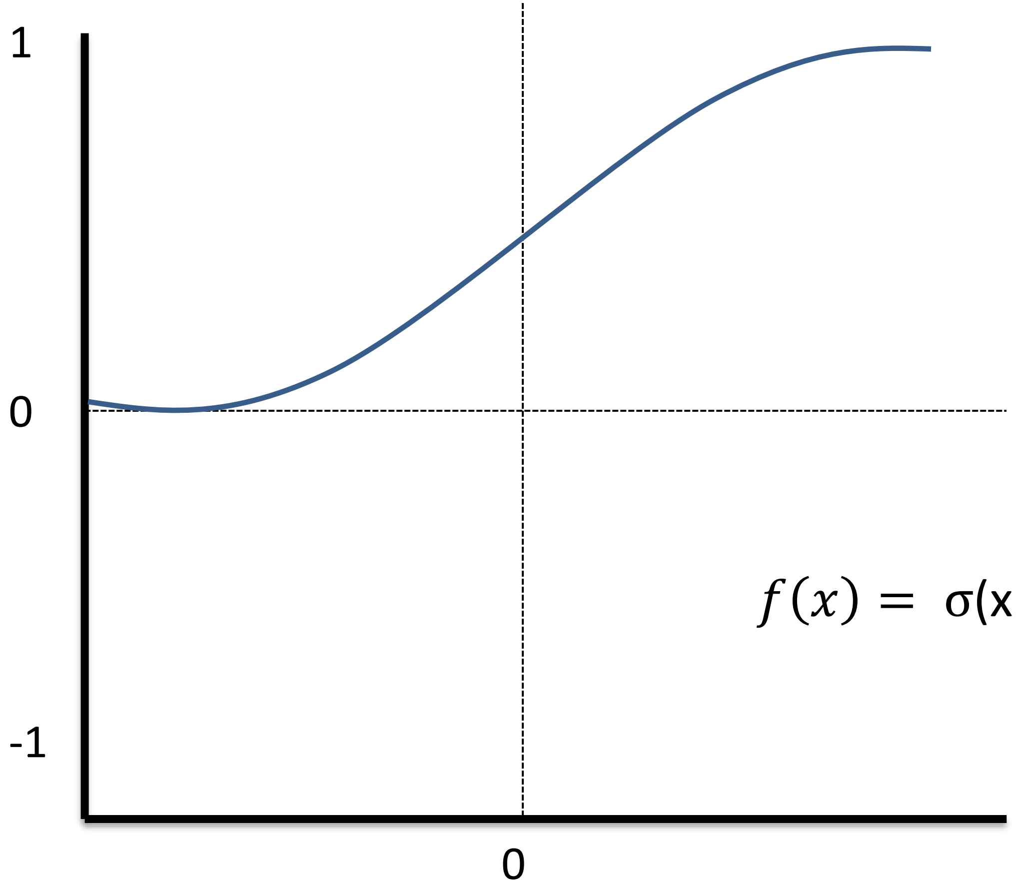
- Escalonada




$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geq 0 \end{cases}$$

# Función de activación

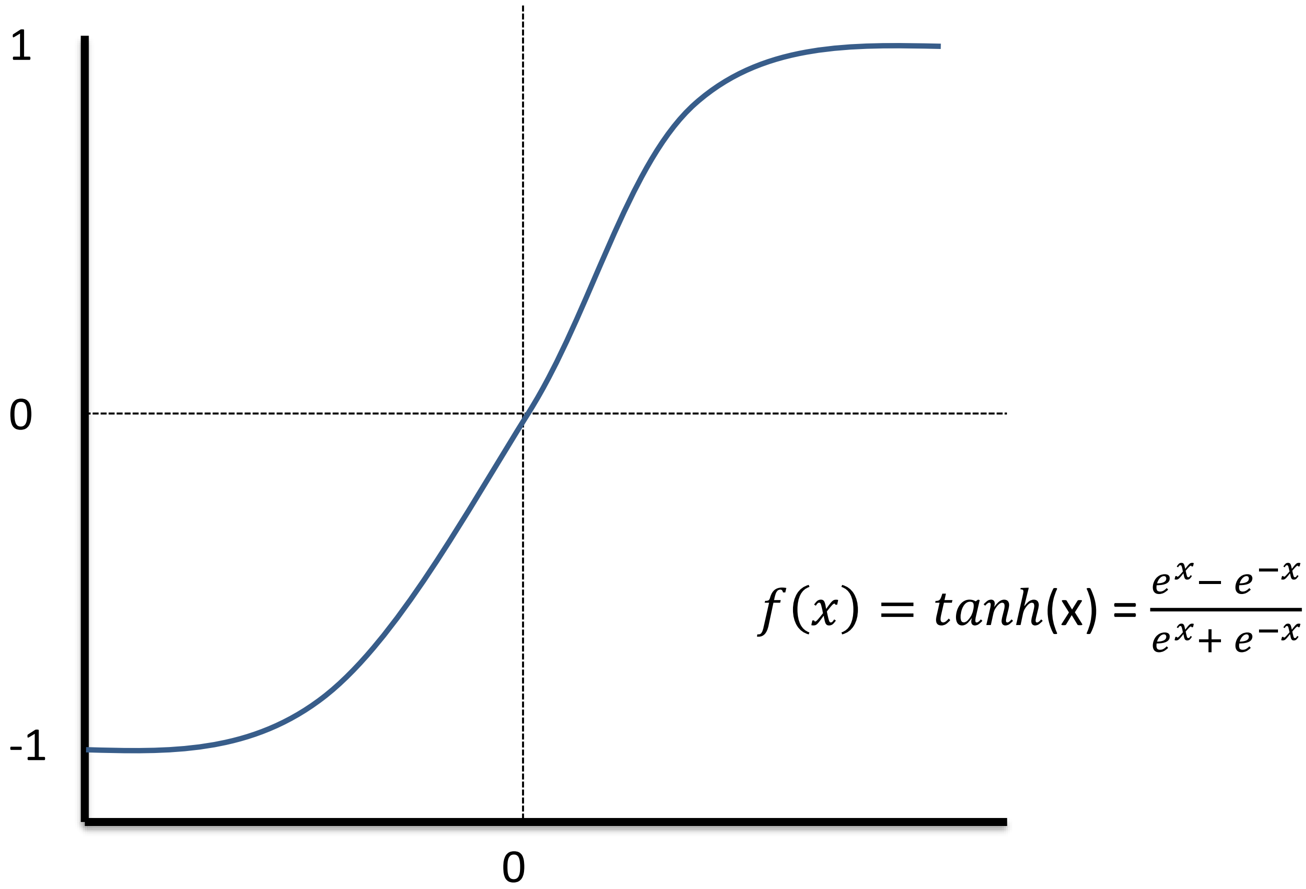
- Sigmoide




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

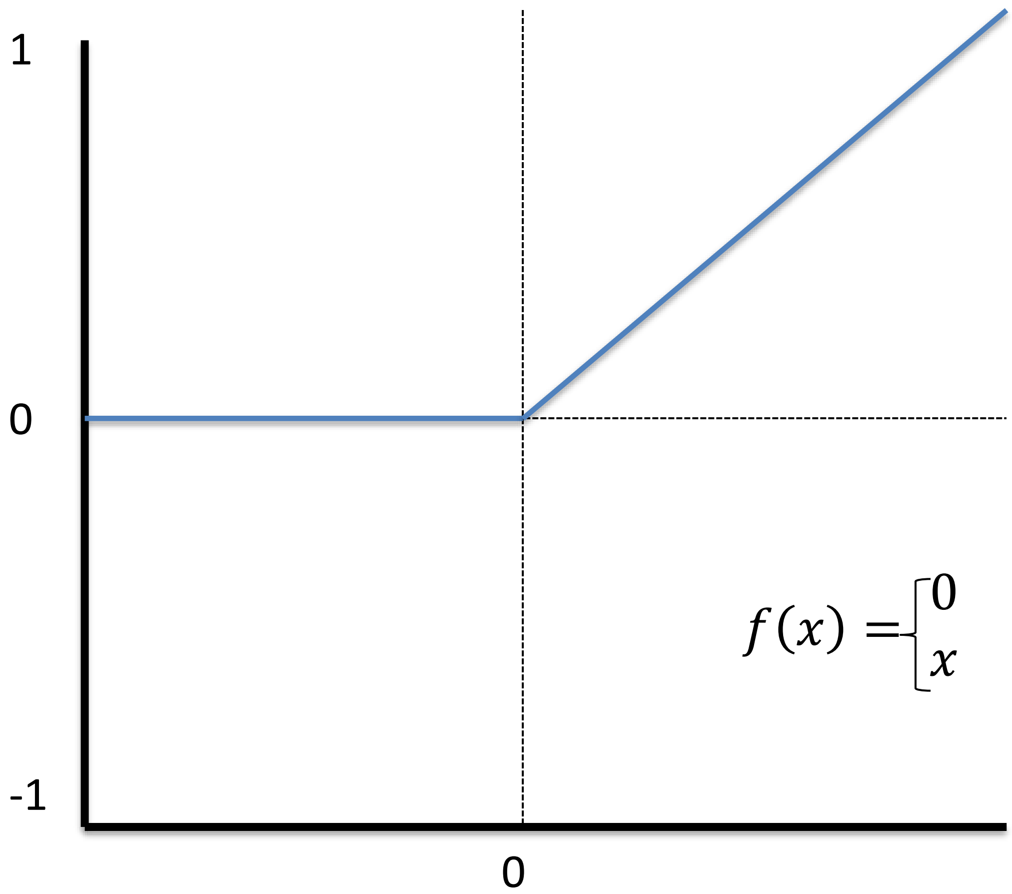
# Función de activación

- TANH



# Función de activación

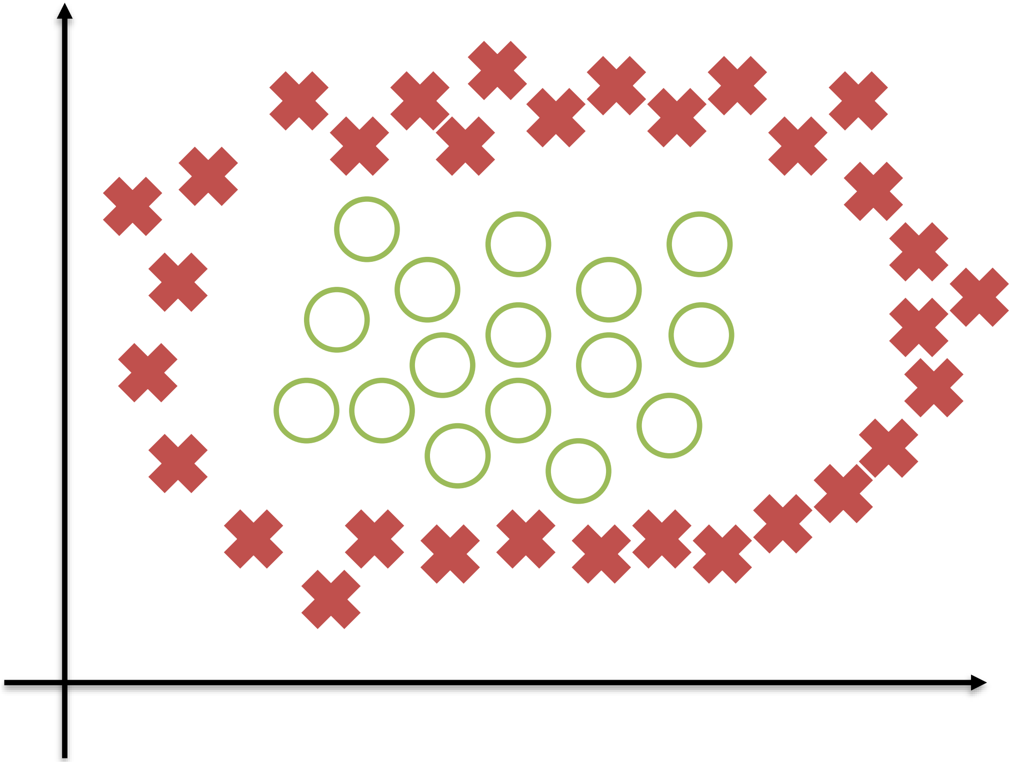
- RELU



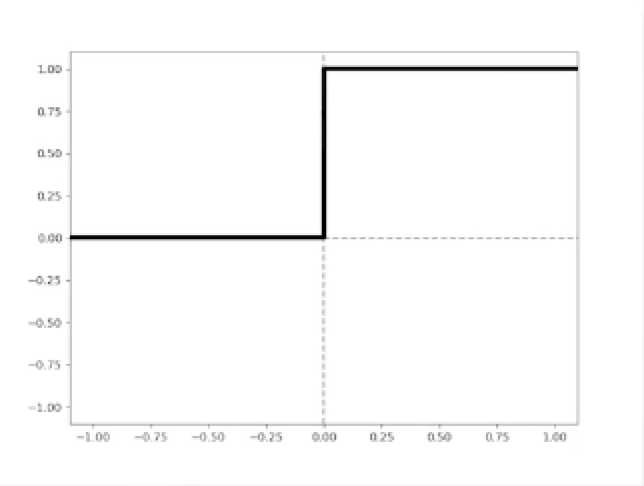

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

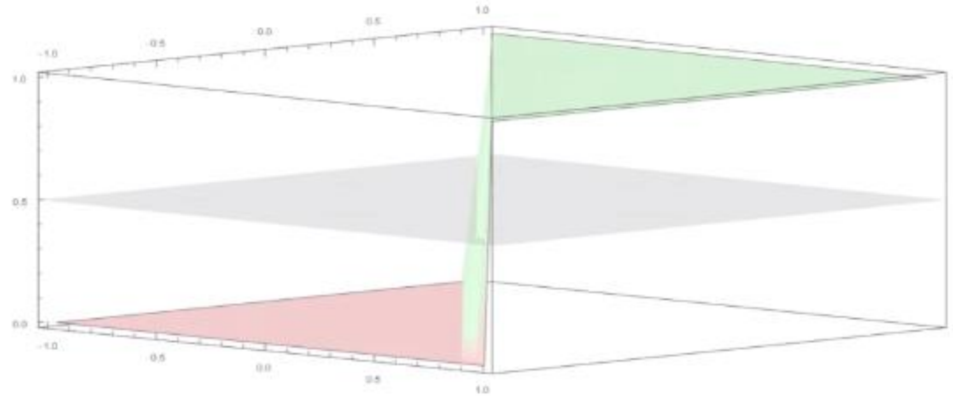
# ¿Cómo podemos aplicarlo?

# ¿Cómo podemos aplicarlo?

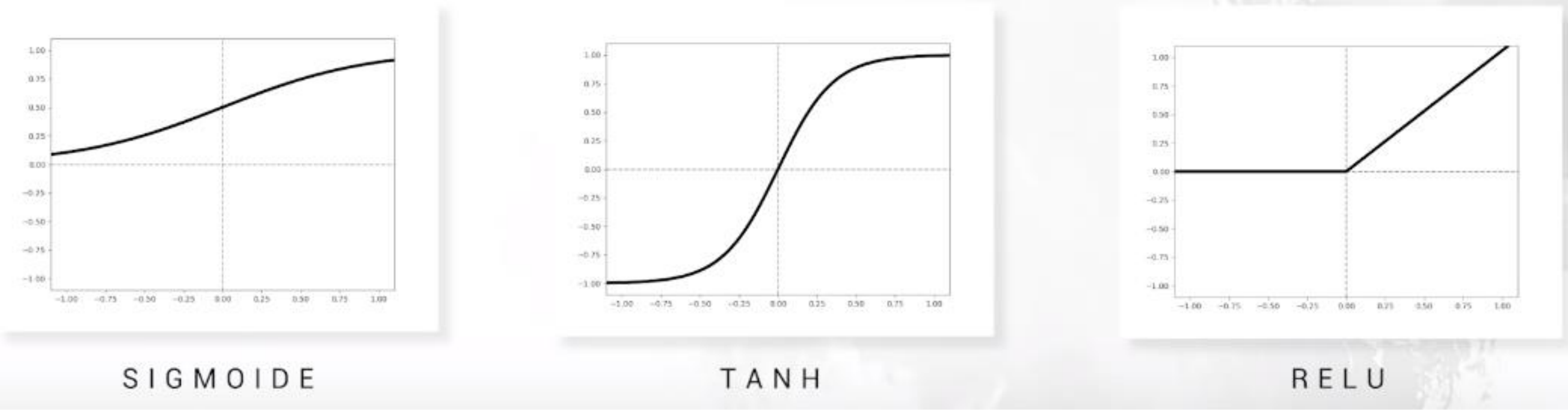


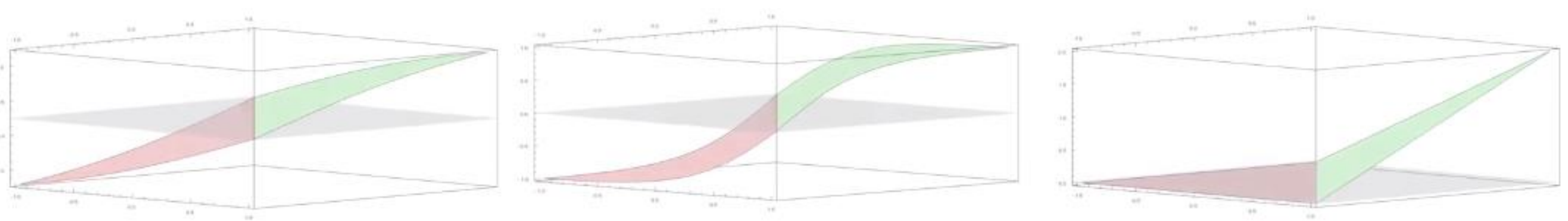
# Visualizar geométricamente



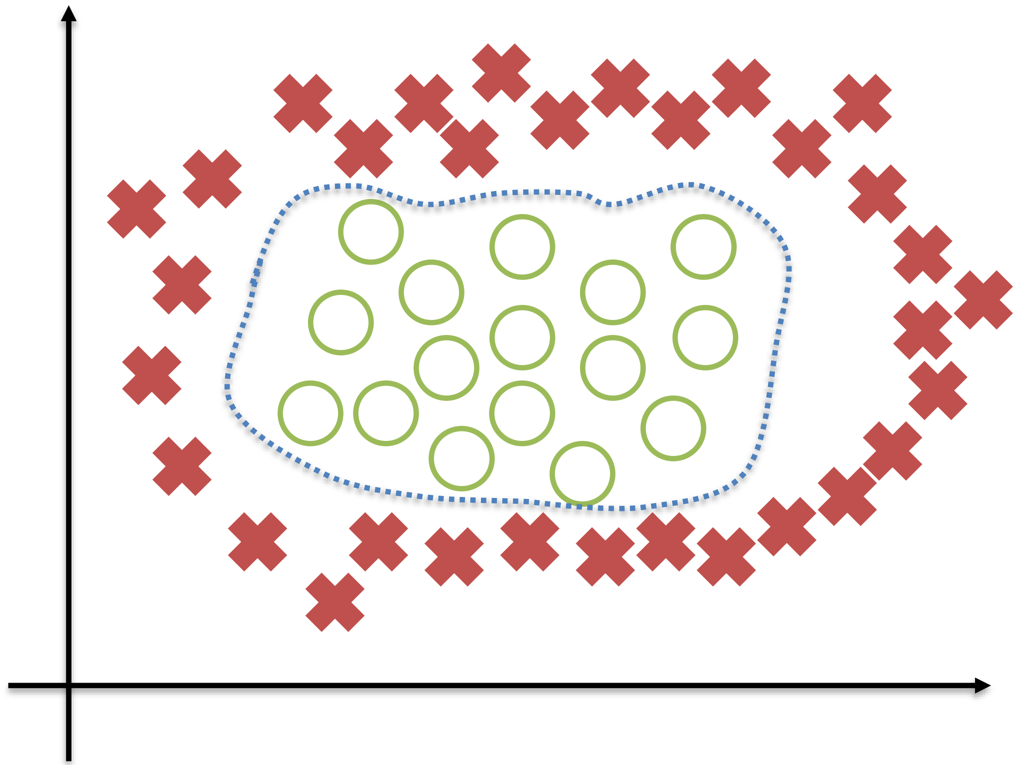



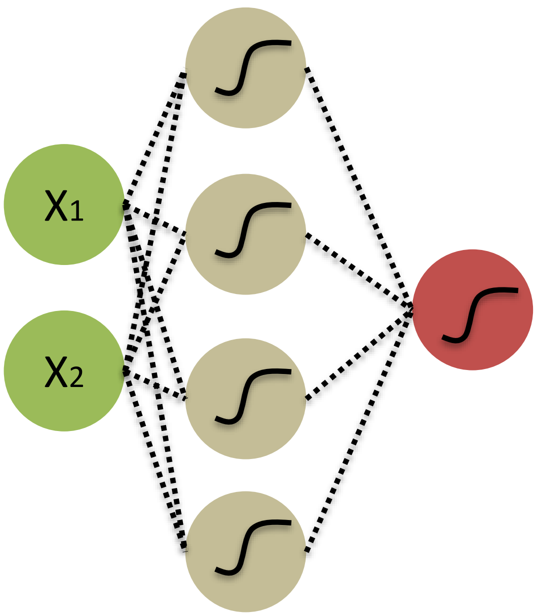
# Visualizar geométricamente

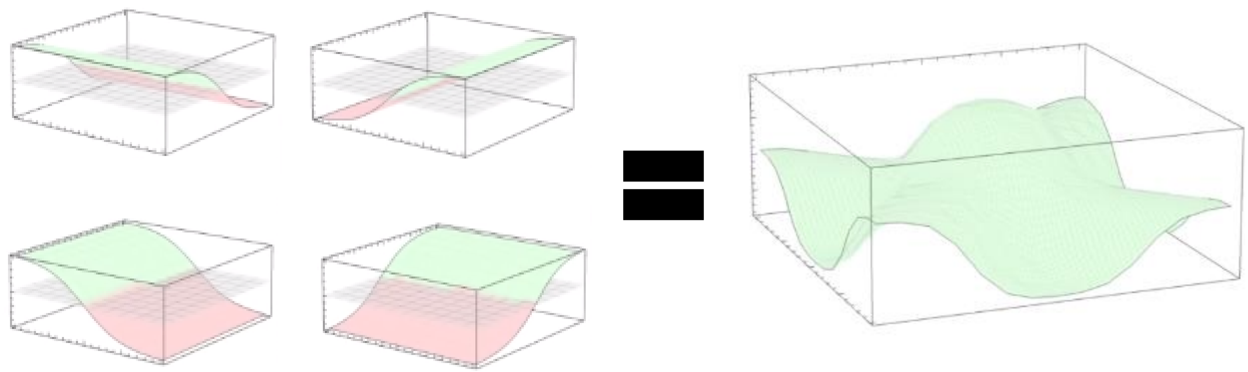


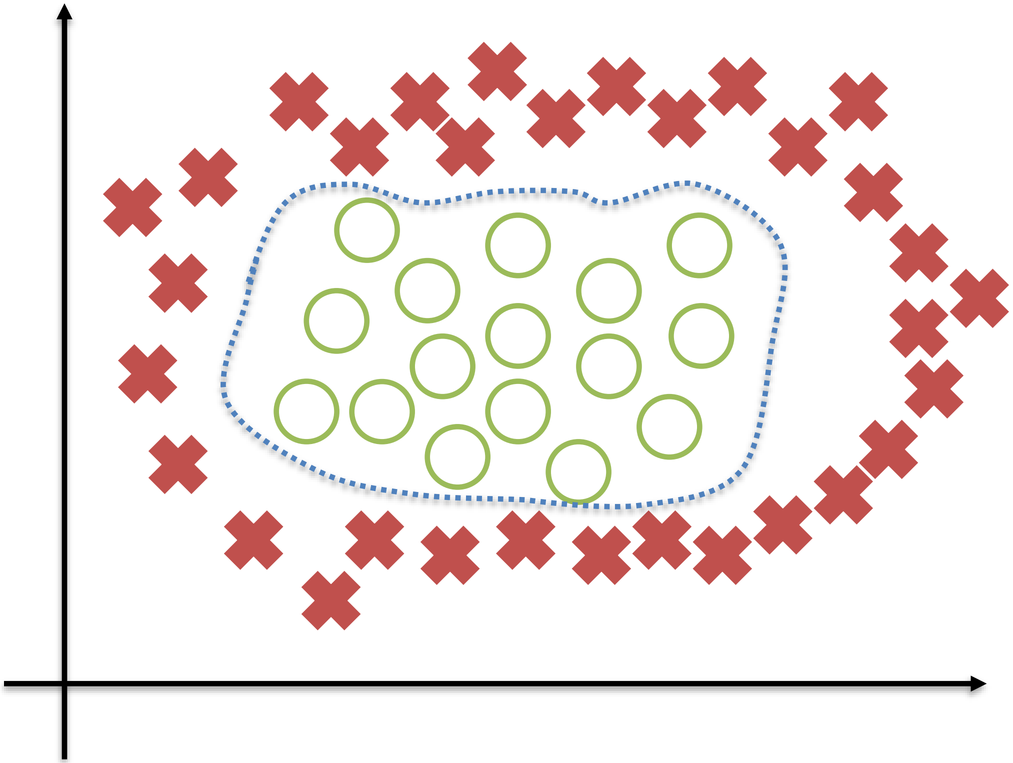



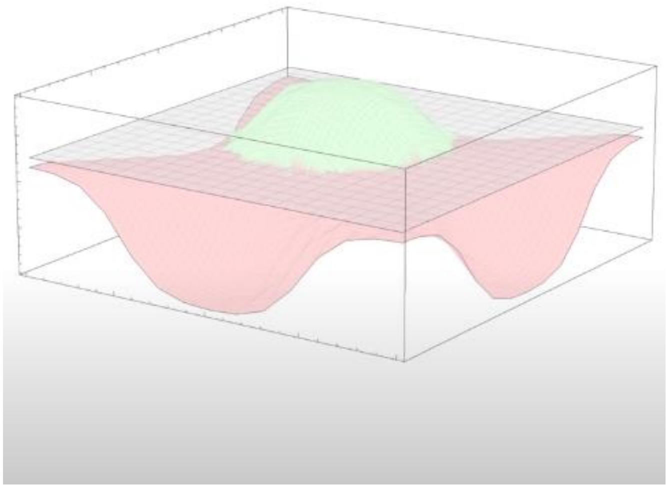
# ¿Cómo podemos encontrar una solución?




X1
X2







# ¿ Y cómo aprende todo esto la red por si misma?

# ¿ Y cómo aprende todo esto la red por si misma?

## Learning representations by back-propagating errors


**David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams***

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA



**We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].**


There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but

# Backpropagation

- Intuición
- Matemáticas detrás de backpropagation

# Ajustar el error dentro de la red por medio del descenso del gradiente

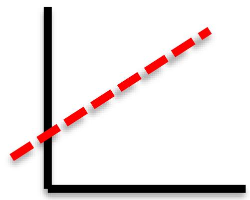
- El gradiente contiene las pendientes para cada una de las dimensiones de nuestra función *f*

- Dentro de una regresión lineal calculamos el costo de variar los parámetros

Regresión lineal

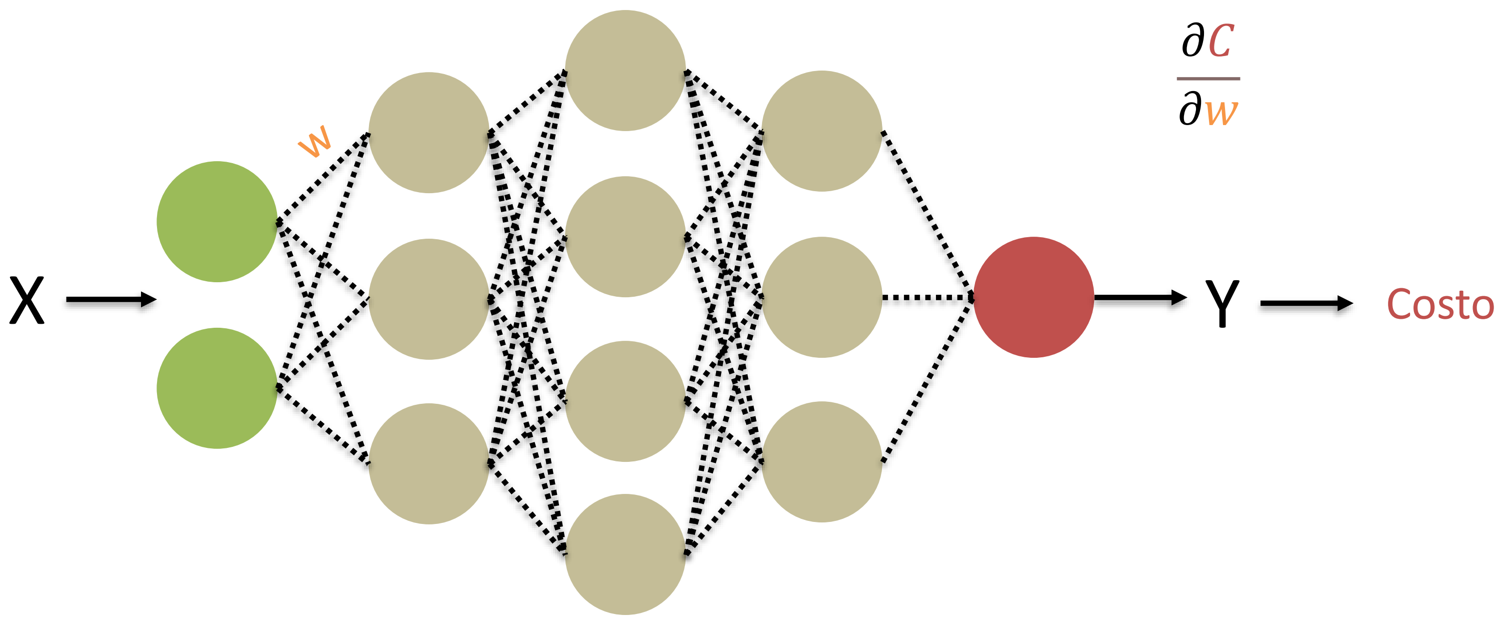
$y = W_0 + W_1 X_1$

¿Cómo varia el costo ante un cambio del parámetro W?

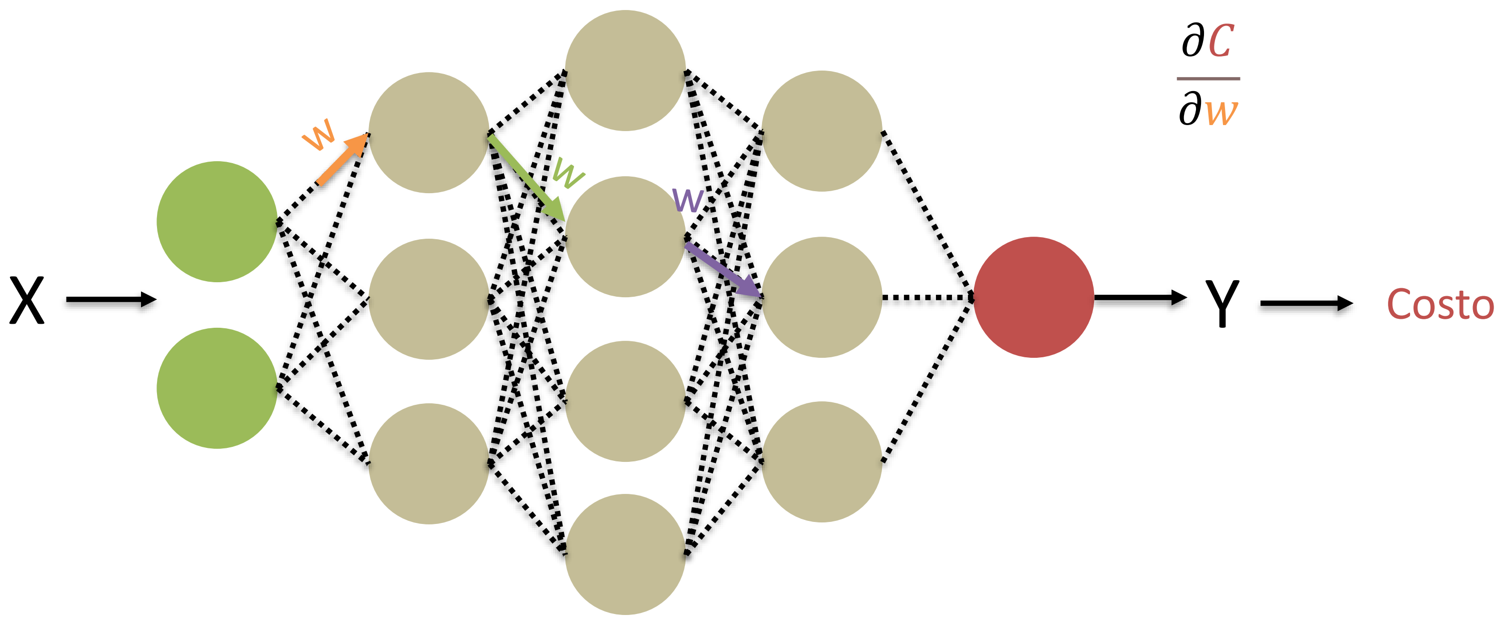


$$\frac{\partial C}{\partial w}$$

# En una red neuronal es complicado calcularlo



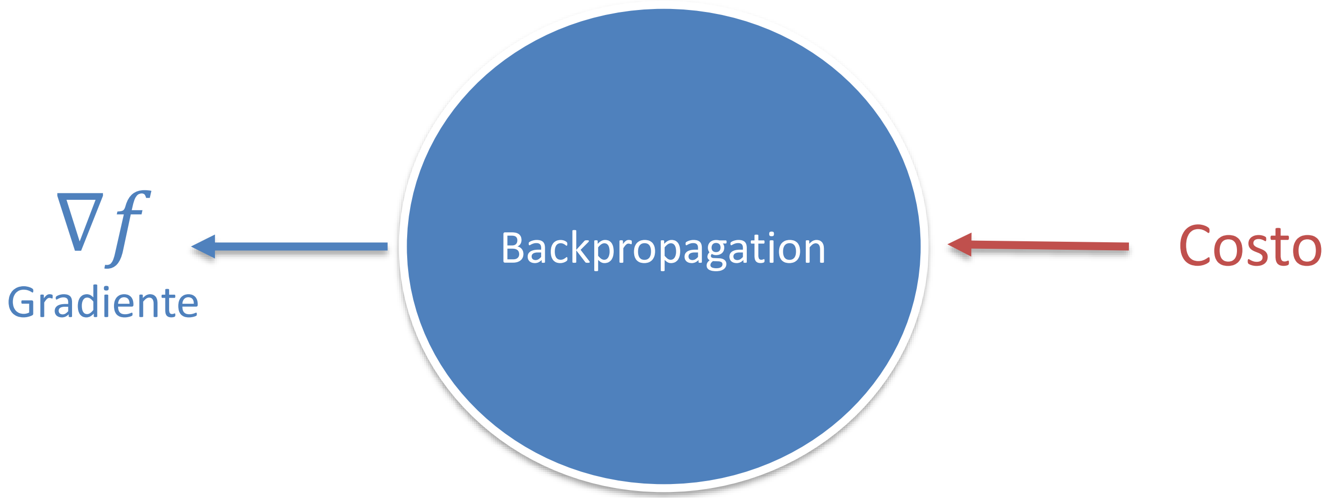
# En una red neuronal es complicado calcularlo



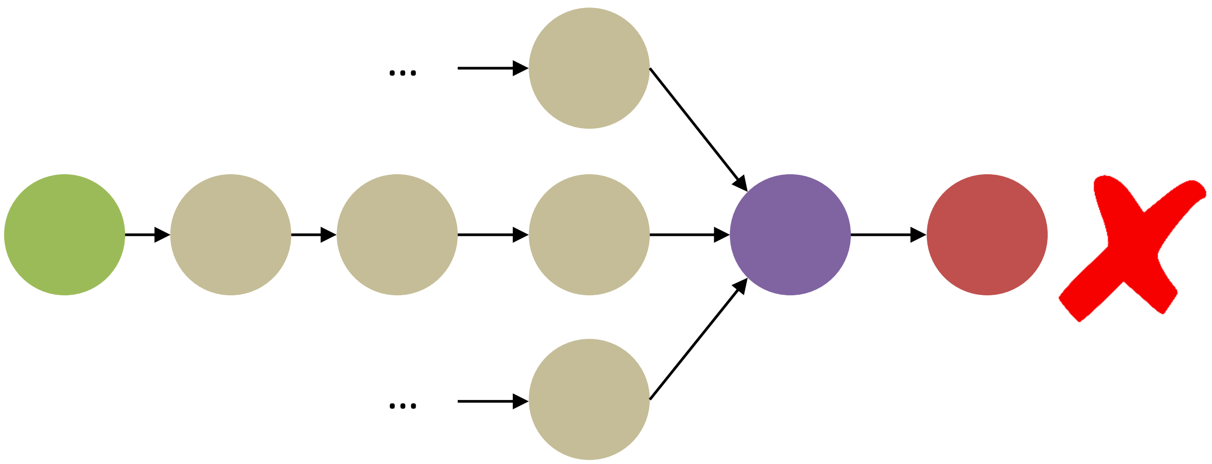

$$\frac{\partial C}{\partial w}$$

# ¿Quién nos da el valor del error?

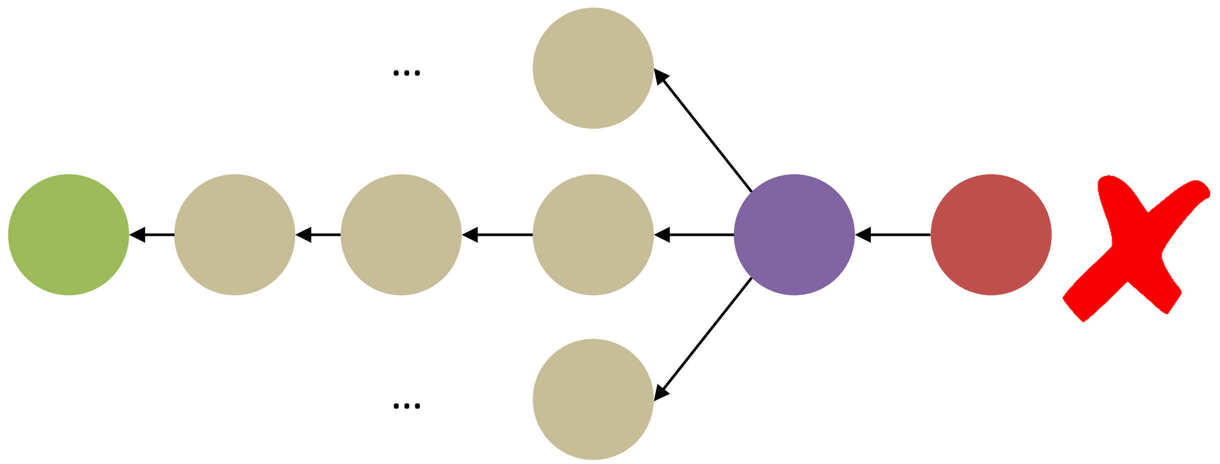
- Con el descenso del gradiente optimizaremos nuestra función de costo usando la técnica de backpropagation.



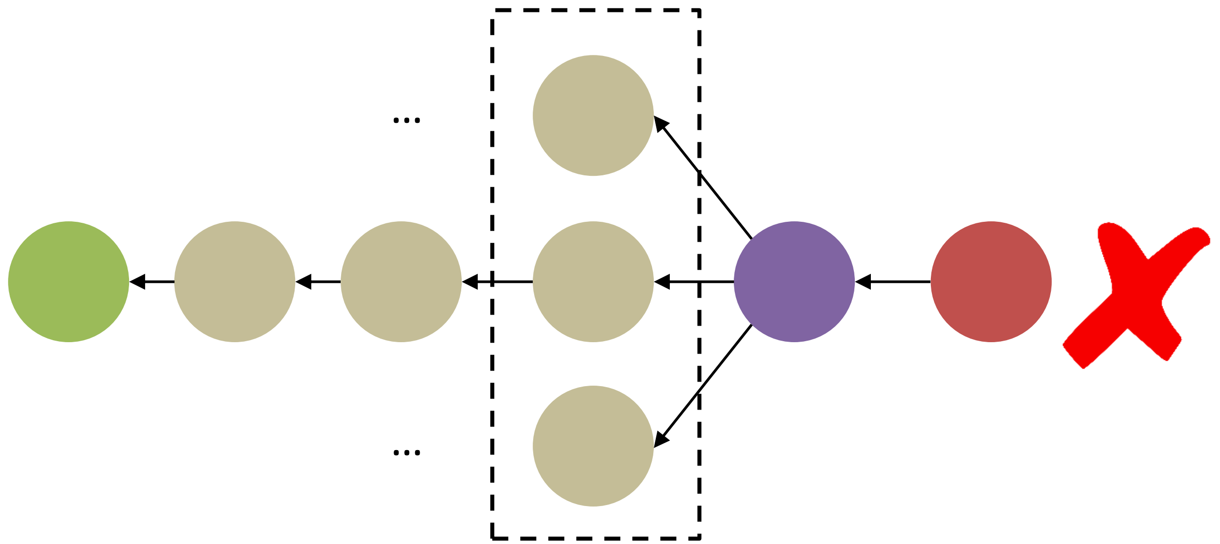
# Intuición del backpropagation



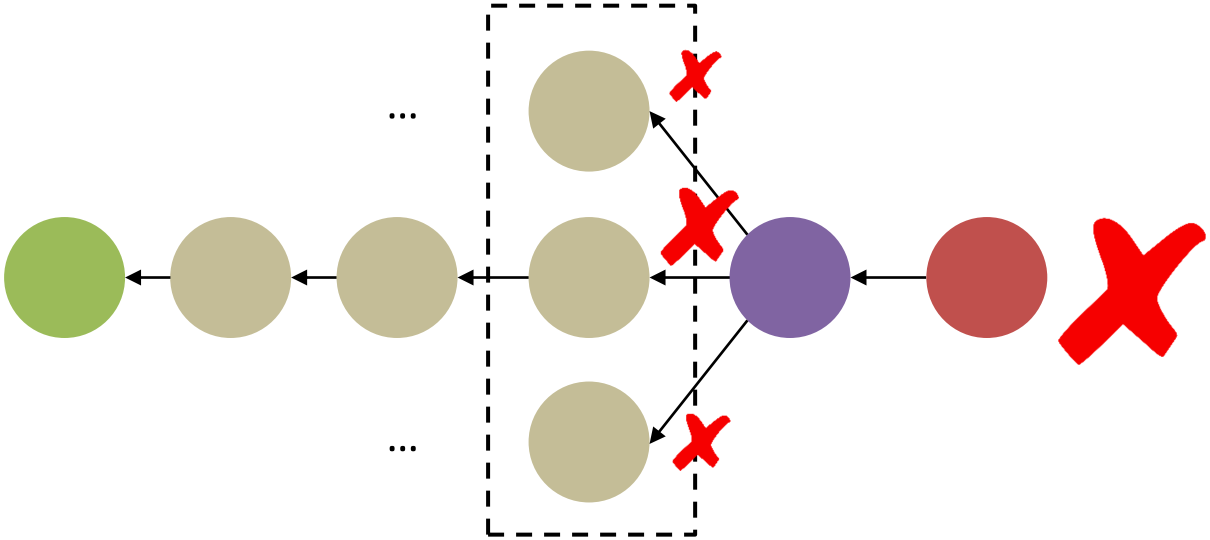
# Intuición del backpropagation



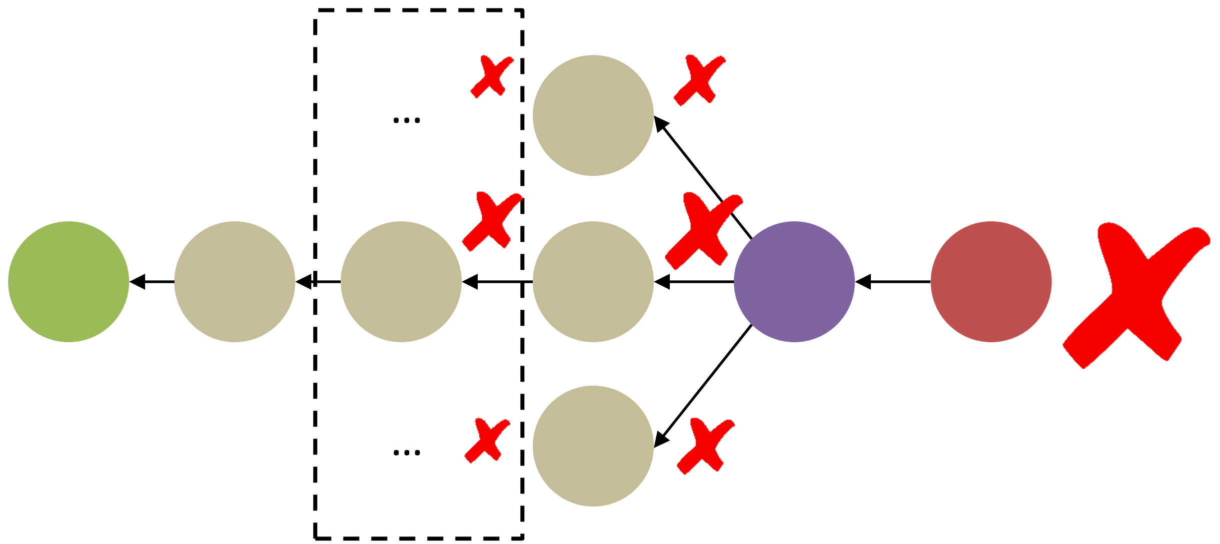
# Intuición del backpropagation



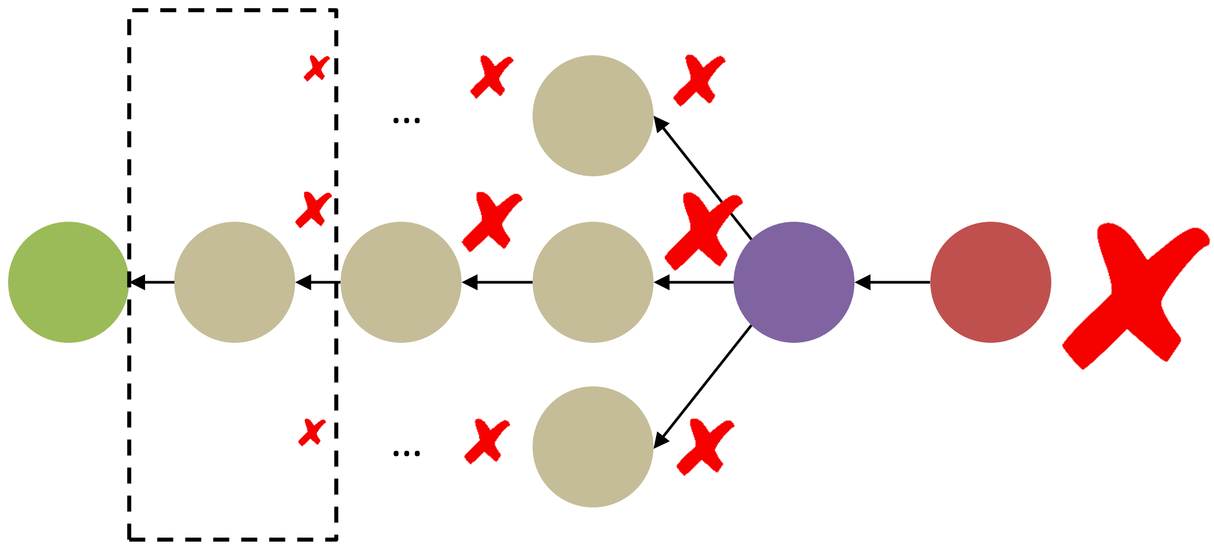
# Intuición del backpropagation



# Intuición del backpropagation



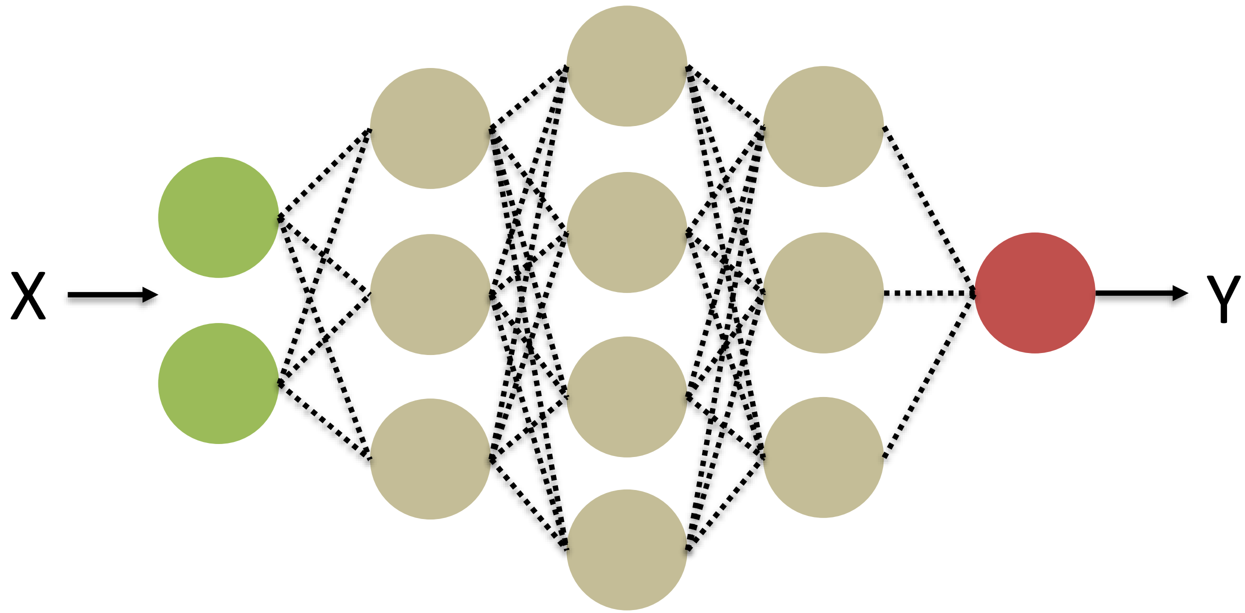
# Intuición del backpropagation



- Esto genera un calculo del error de manera eficiente.
- Los errores son los que utilizan para calcular las derivadas parciales de cada parámetro de la red.
- Antes se utilizaba un algoritmo de fuerza bruta (Perturbación Aleatoria).

# Matemáticas detrás de backpropagation

- Inicializamos aleatoriamente nuestra red



# ¿Cómo varia el costo ante un cambio del parámetro W?

$$\frac{\partial C}{\partial w} \longrightarrow$$

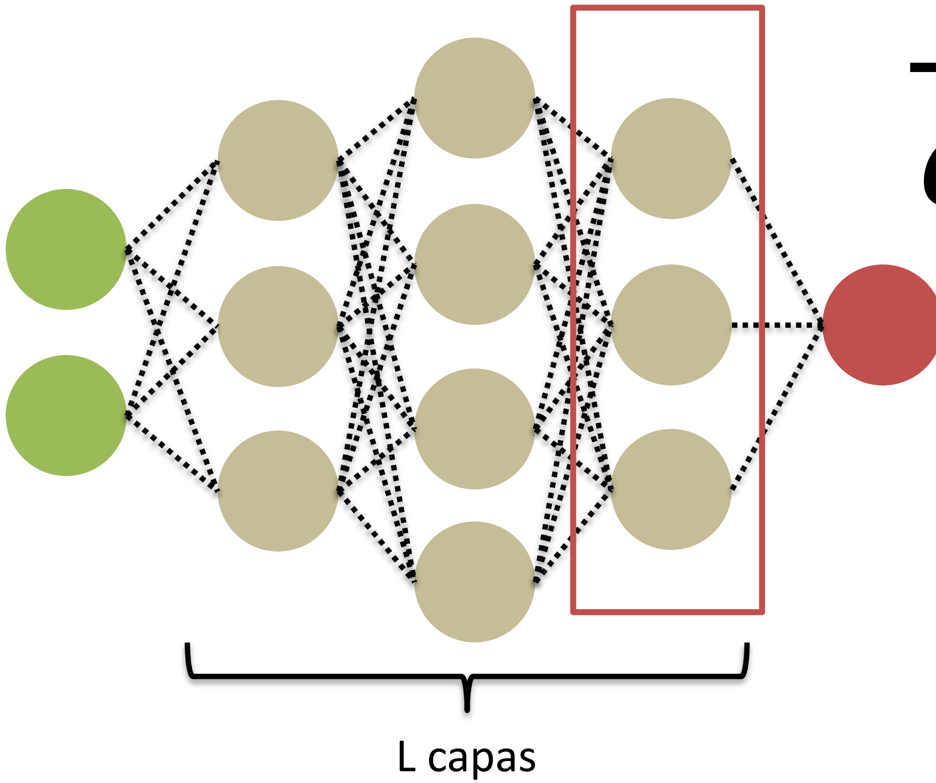
Derivada parcial del costo con respecto a los parámetros de la red

# ¿Cómo varia el costo ante un cambio del parámetro W?

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial w} \qquad \frac{\partial C}{\partial b}$$

Pesos de la red Bias

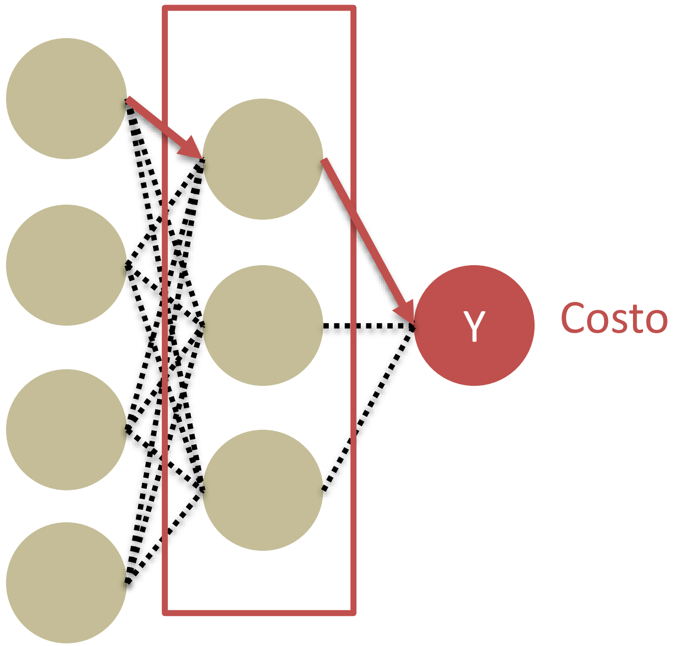
$$\frac{\partial C}{\partial b^L} \quad \frac{\partial C}{\partial w^L}$$



# ¿Cómo lo calculamos?

$$\frac{\partial C}{\partial b^{L}} \quad \frac{\partial C}{\partial w^{L}} \longrightarrow \begin{array}{cc} C & C \\ \uparrow & \uparrow \\ b^{L} & w^{L} \end{array}$$

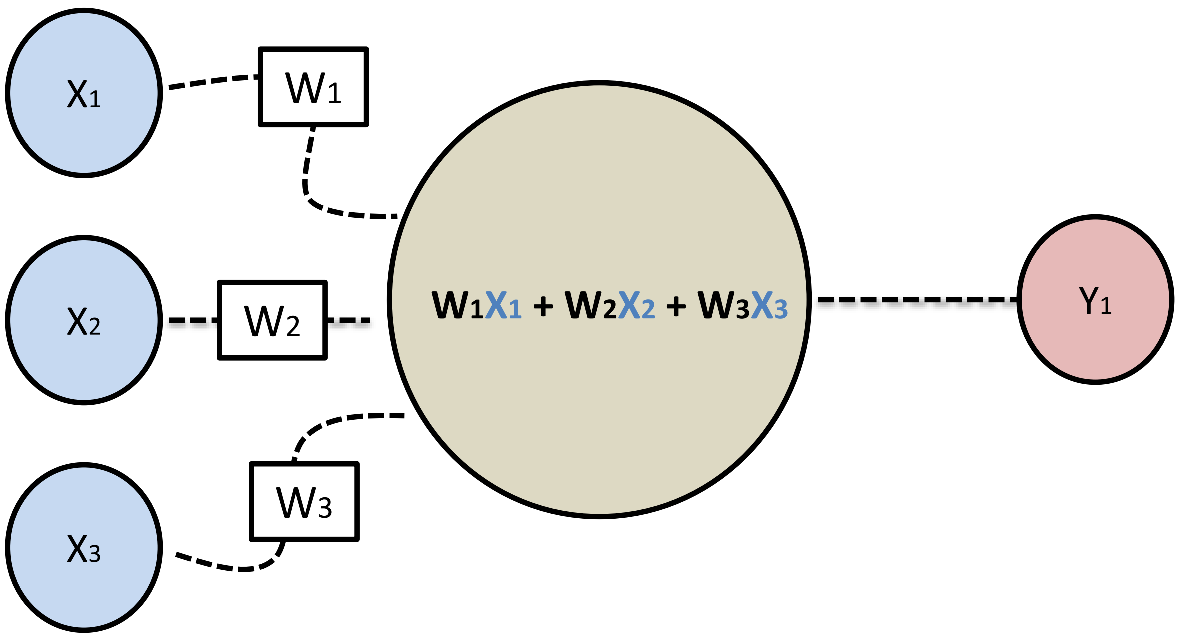
Obtener el camino relacionado
al parámetro y el costo final


Y
Costo

# Derivadas de la última capa

Suma ponderada

$$Z^L = W^L X + b^L$$



# Derivadas de la última capa

$$a(Z^L)$$

Resultado de la suma ponderada

Función de activación

# Derivadas de la última capa

$$C(a(Z^L)) = ERROR$$

Resultado de la suma ponderada

Función de activación

Función de costo

# Derivadas de la última capa

Composición de funciones

$$C(a(Z^L))$$

# Derivadas de la última capa

Composición de funciones

$$C(a(Z^L))$$

Aplicamos regla de cadena

# Derivadas de la última capa

$$\frac{\partial C}{\partial w^{L}}$$

Aplicamos regla de cadena

$$\frac{\partial C}{\partial b^{L}}$$

$$z^{L} = W^{L} a^{L-1} + b^{L}$$

$$C(a^{L}(z^{L}))$$

# Derivadas de la última capa

$$\frac{\partial C}{\partial w^{L}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L}}{\partial w^{L}}$$

$$\frac{\partial C}{\partial b^{L}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L}}{\partial b^{L}}$$

$$Z^{L} = W^{L} a^{L-1} + b^{L} \qquad C(a^{L}(Z^{L}))$$

# Derivadas de la última capa

$$\frac{\partial C}{\partial a^L}$$ → Derivada de la activación con respecto al costo → Como varia el costo de la red cuando variamos la activación de las últimas neuronas

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w^L}$$

# Derivadas de la última capa

Función de costo
Error cuadrático medio

$$C(a_j^L) = \frac{1}{2}\sum_j \left(y_j - a_j^L\right)^2$$

$$\frac{\partial C}{\partial a_j^L} = (a_j^L - y_j)$$

# Derivadas de la última capa

$$\frac{\partial a^{L}}{\partial z^{L}}$$ → Derivada de la activación con respecto a Z → Como varia la salida de la neurona cuando variamos la suma ponderada de la neurona

$$\frac{\partial C}{\partial a_j^L} = (a_j^L - y_j)$$

$$\frac{\partial C}{\partial w^{L}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L}}{\partial w^{L}}$$

# Derivadas de la última capa

Función de activación
Sigmoide

$$a^L(z^L) = \frac{1}{1+ e^{-z^L}}$$

$$\frac{\partial a^L}{\partial z^L} = a^L(z^L) \cdot (1 - a^L(z^L))$$

# Derivadas de la última capa

$$\frac{\partial z^{L}}{\partial w^{L}} \quad \frac{\partial z^{L}}{\partial b^{L}}$$

→ Derivada de Z con respecto a los pesos → Como varia la suma ponderada cuando variamos los parámetros

$$\frac{\partial a}{\partial z} = a^{L}(z^{L}) \cdot (1 - a^{L}(z^{L}))$$

$$\frac{\partial C}{\partial a_j^{L}} = (a_j^{L} - y_j)$$

$$\frac{\partial C}{\partial w^{L}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L}}{\partial w^{L}}$$

# Derivadas de la última capa

Derivada de la suma ponderada

$$z^L = \sum_i a_i^{L-1} w_i^L + b^L$$

$$\frac{\partial z^L}{\partial b^L} = 1 \qquad \frac{\partial z^L}{\partial w^L} = a_i^{L-1}$$

# Derivadas de la última capa

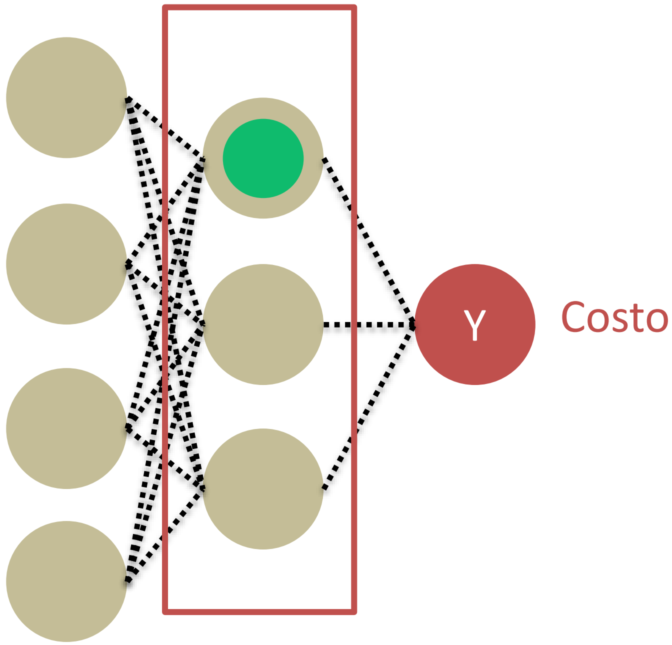
$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial b^L}$$

$$\frac{\partial C}{\partial a_j^L} = (a_j^L - y_j) \qquad \frac{\partial a}{\partial z} = a^L(z^L) \cdot (1 - a^L(z^L)) \qquad \frac{\partial z^L}{\partial b^L} = 1 \qquad \frac{\partial z^L}{\partial w^L} = a_i^{L-1}$$

# Derivadas de la última capa

$$\frac{\partial C}{\partial w^{L}} = \underbrace{\frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}}_{\frac{\partial C}{\partial z^{L}}} \cdot \frac{\partial z^{L}}{\partial w^{L}}$$




$$\frac{\partial C}{\partial z^L}$$

Como varía el error del costo
de acuerdo al cambio en la suma de la neurona

Esta derivada nos dice cuanto responsabilidad tiene
la neurona en el resultado final

$$\frac{\partial C}{\partial z^L}$$

Error atribuido a la neurona

$\delta^L$

$$\frac{\partial C}{\partial w^{L}} = \delta^{L} \cdot \frac{\partial z^{L}}{\partial w^{L}}$$

$$\frac{\partial C}{\partial b^{L}} = \delta^{L} \cdot \frac{\partial z^{L}}{\partial b^{L}}$$

$$\frac{\partial C}{\partial w^{L}} = \delta^{L} \cdot \boxed{\frac{\partial z^{L}}{\partial w^{L}}} = a_i^{L-1}$$

$$\frac{\partial C}{\partial b^{L}} = \delta^{L} \cdot \boxed{\frac{\partial z^{L}}{\partial b^{L}}} = 1$$

$$\frac{\partial C}{\partial w^{L}} = \delta^{L} a_i^{L-1}$$

$$\frac{\partial C}{\partial b^{L}} = \delta^{L}$$

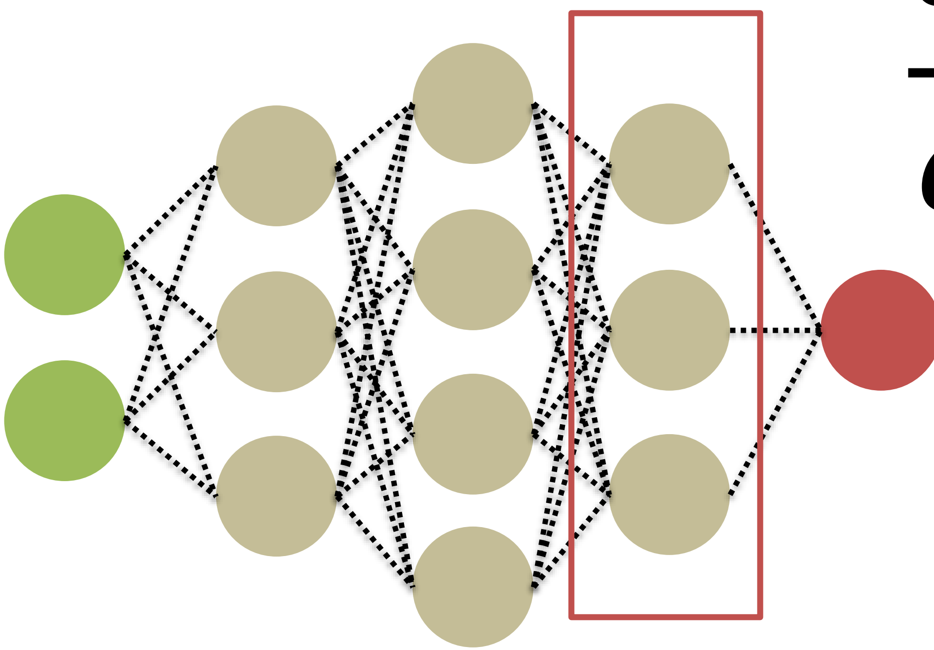
$$\delta^{L} = \frac{\partial C}{\partial a} \frac{\partial a}{\partial z}$$

Error de las neuronas

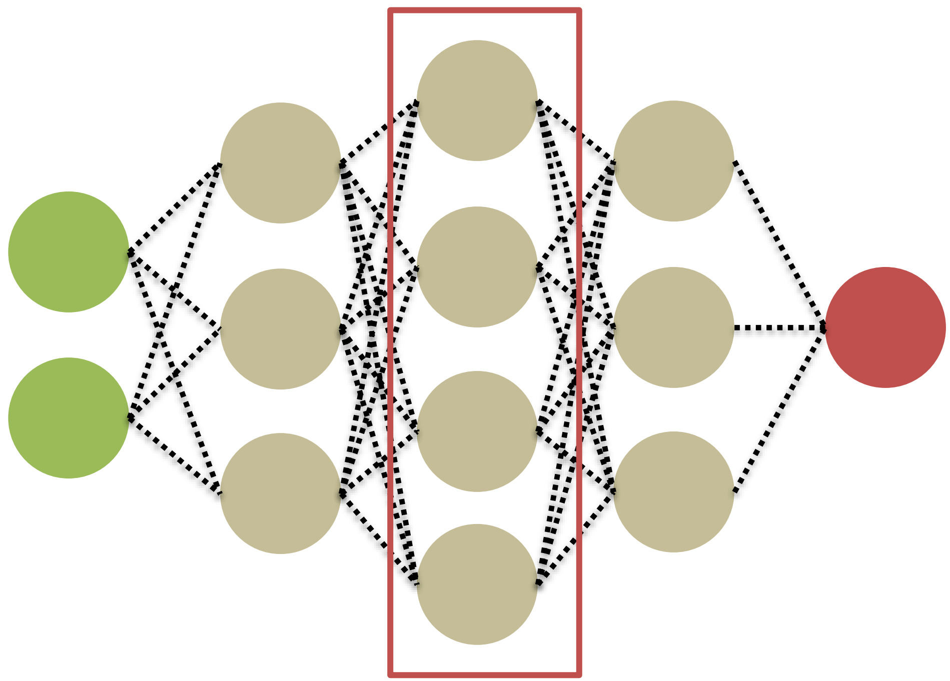
$$\frac{\partial C}{\partial w^{L}} = \delta^{L} a_i^{L-1}$$

Derivadas parciales

$$\frac{\partial C}{\partial b^{L}} = \delta^{L}$$



$$\frac{\partial C}{\partial b} \quad \frac{\partial C}{\partial w}$$



# Derivadas de la siguiente capa

$$\frac{\partial C}{\partial w^{L-1}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L-1}}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}} \cdot \frac{\partial z^{L-1}}{\partial w^{L-1}}$$

$$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L-1}}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}} \cdot \frac{\partial z^{L-1}}{\partial b^{L-1}}$$

# Derivadas de la siguiente capa

$$\frac{\partial C}{\partial w^{L-1}} = \underbrace{\frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}}_{\delta^{L}} \cdot \frac{\partial z^{L-1}}{\partial a^{L-1}} \cdot \underbrace{\frac{\partial a^{L-1}}{\partial z^{L-1}}}_{\text{Derivada función de activación}} \cdot \underbrace{\frac{\partial z^{L-1}}{\partial w^{L-1}}}_{a^{L-2}}$$

$$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}} \cdot \frac{\partial z^{L-1}}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}} \cdot \underbrace{\frac{\partial z^{L-1}}{\partial b^{L-1}}}_{1}$$

# Derivadas de la siguiente capa

$$\frac{\partial C}{\partial w^{L-1}} = \boxed{\frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}} \cdot \boxed{\frac{\partial z^{L-1}}{\partial a^{L-1}}} \cdot \boxed{\frac{\partial a^{L-1}}{\partial z^{L-1}}} \cdot \boxed{\frac{\partial z^{L-1}}{\partial w^{L-1}}}$$

$a^{L-2}$

$$\frac{\partial C}{\partial b^{L-1}} = \boxed{\frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}} \cdot \boxed{\frac{\partial z^{L-1}}{\partial a^{L-1}}} \cdot \boxed{\frac{\partial a^{L-1}}{\partial z^{L-1}}} \cdot \boxed{\frac{\partial z^{L-1}}{\partial b^{L-1}}}$$

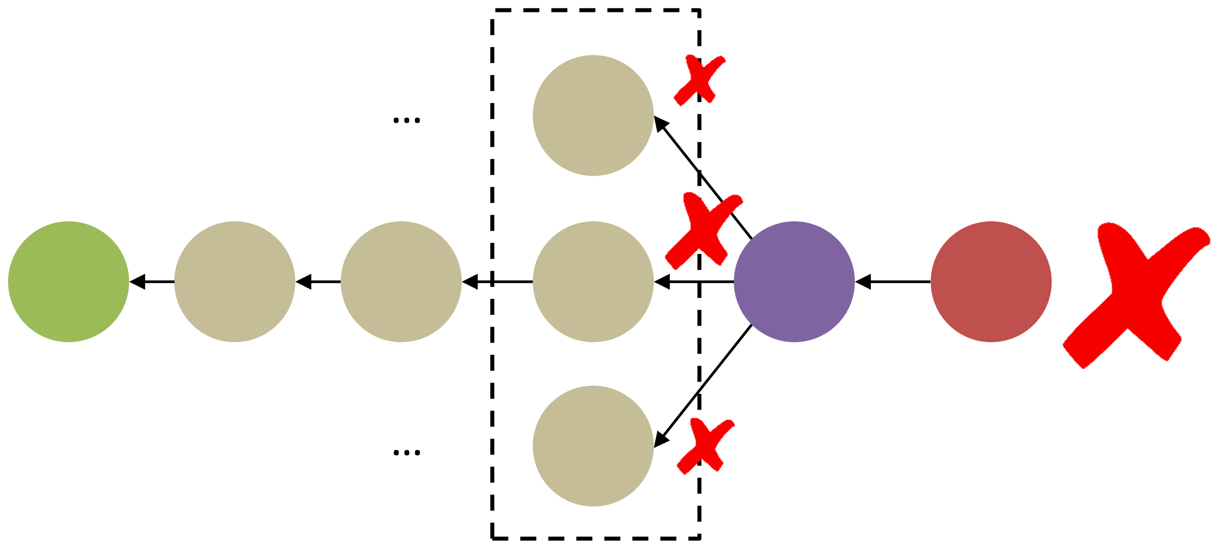
$\delta^{L}$

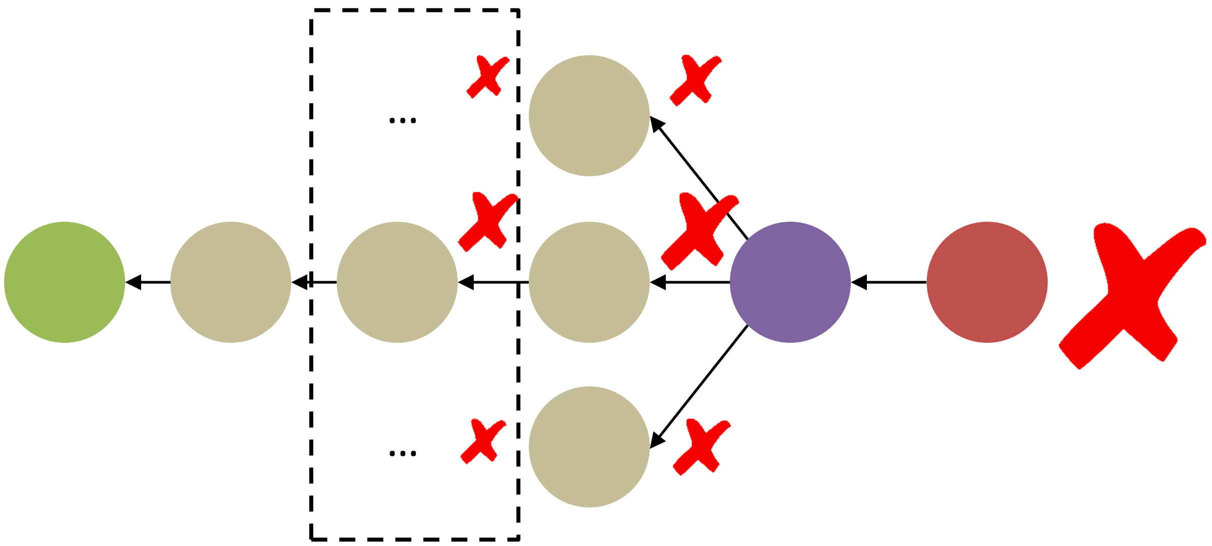
$W^{L}$

Como varía la suma ponderada de una capa
Cuando se varía la salida de una neurona anterior

Derivada
función de
activación

1





# Derivadas de la siguiente capa

$$\frac{\partial C}{\partial w^{L-1}} = \underbrace{\frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}}_{\delta^{L}} \cdot \underbrace{\frac{\partial z^{L-1}}{\partial a^{L-1}}}_{W^{L}} \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}} \cdot \underbrace{\frac{\partial z^{L-1}}{\partial w^{L-1}}}_{a^{L-2}}$$

$$\frac{\partial C}{\partial b^{L-1}} = \underbrace{\frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}}_{\delta^{L}} \cdot \underbrace{\frac{\partial z^{L-1}}{\partial a^{L-1}}}_{W^{L}} \cdot \underbrace{\frac{\partial a^{L-1}}{\partial z^{L-1}}}_{\text{Derivada función de activación}} \cdot \underbrace{\frac{\partial z^{L-1}}{\partial b^{L-1}}}_{1}$$

$$\frac{\partial C}{\partial z^{L-1}} = \delta^{L-1}$$

- 1. Computo del error de la ultima capa

$$\delta^{L} = \frac{\partial C}{\partial a^{L}} \cdot \frac{\partial a^{L}}{\partial z^{L}}$$

- 2. Retro propagar el error a la capa anterior

$$\delta^{L-1} = w^{l} \delta^{l} \cdot \frac{\partial a^{l-1}}{\partial z^{l-1}}$$

- 3. Calcular las derivadas de la capa usando el error

$$\frac{\partial C}{\partial w^{l-1}} = \delta^{L-1} a^{l-2} \qquad \frac{\partial C}{\partial b^{L-1}} = \delta^{L-1}$$

# Ejercicio /Ejemplo

# Gracias por su atención!

Mario Ezra Aragón

*mearagon@inaoep.mx*

En nombre de:

Mario Ezra Aragón, Juan Luís García Mendoza, Adrian Pastor López Monroy, Manuel Montes y Gómez, y Luís Villaseñor Pineda

CONACYT

CIMAT

INAOE

Laboratorio de Tecnologías del Lenguaje
Ciencias Computacionales, INAOE