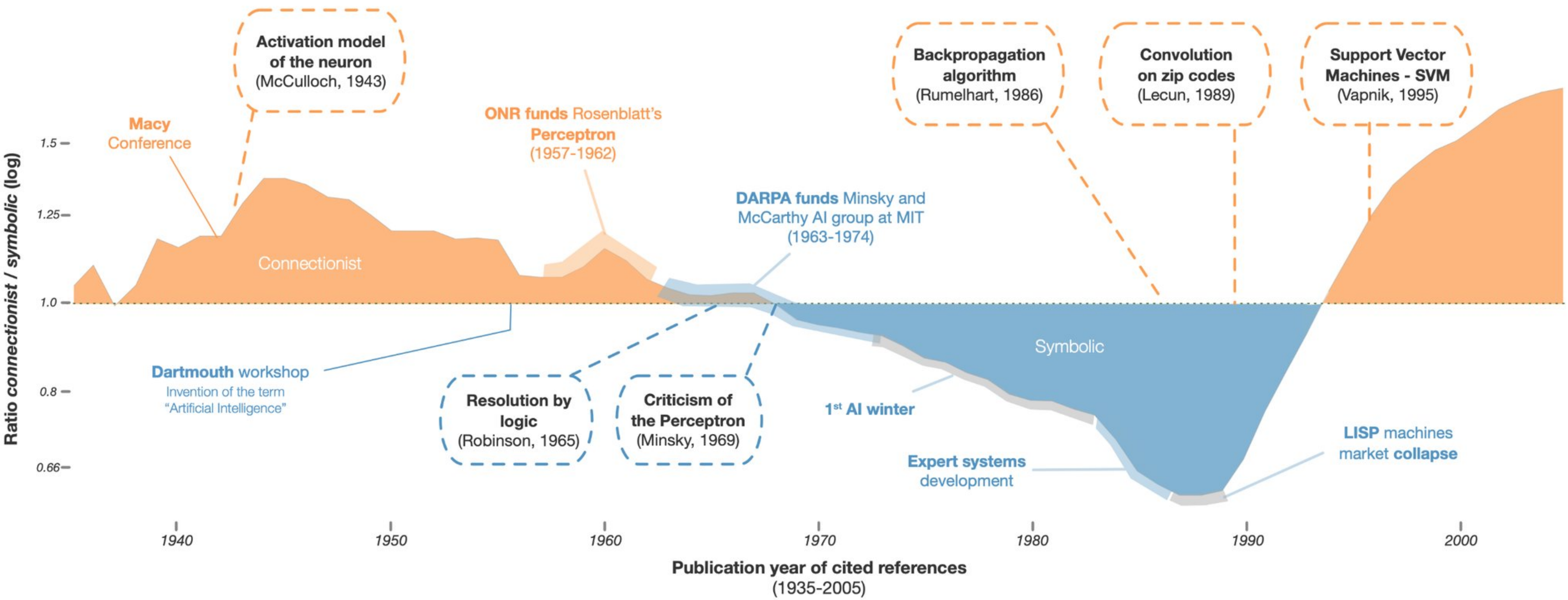
Redes Neuronales

INF-390

Prof: Juan G. Pavez S.

Una historia de dos enfoques



Regresión Logística

- Recordemos la regresión Logística:

$$p(y \mid x, w) = Ber(y \mid \mu(x))$$

- Con $\mu(x) = E[y \mid x] = p(y = 1 \mid x)$

$$\mu(x) = sigm(w^T x)$$

Red Neuronal de 1 Capa

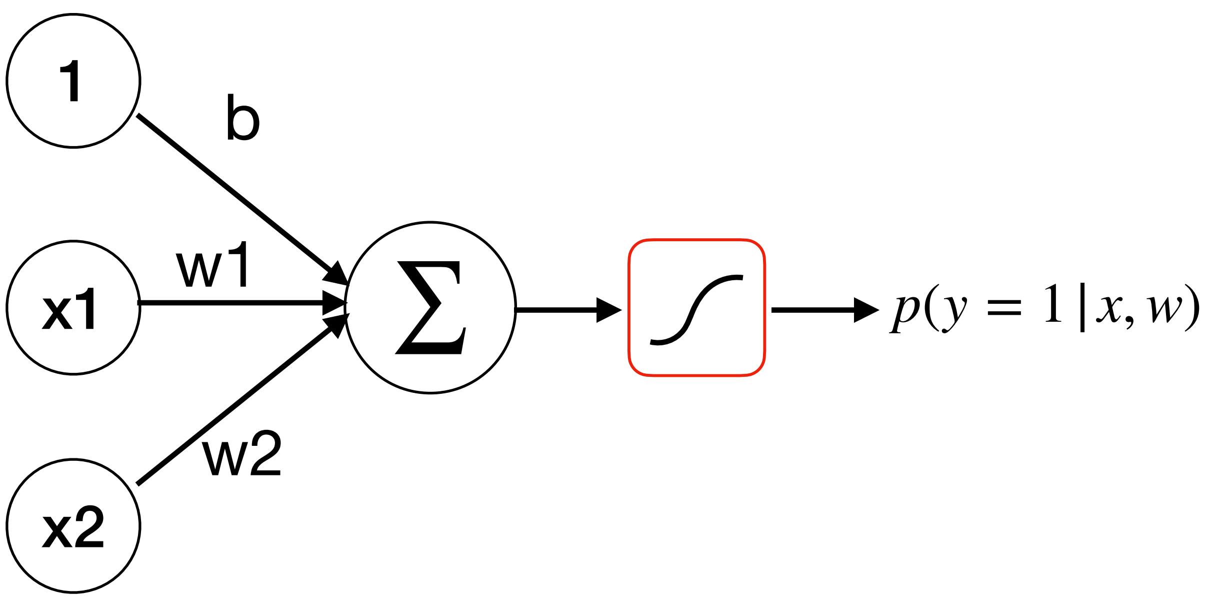
- **Regresión Logística or Red Neuronal de 1 Capa**

$$p(y \mid x, w) = Ber(y \mid \mu(x))$$

- Con $\mu(x) = E[y \mid x] = p(y = 1 \mid x)$

$$\mu(x) = sigm(w^T x) = b + w_1 x_1 + w_2 x_2 = w^T x$$

$$x = (1, x_1, x_2)$$



Red Neuronal de 1 Capa

- **Regresión Logística or Red Neuronal de 1 Capa**

$$p(y|x, w) = Ber(y|\mu(x))$$

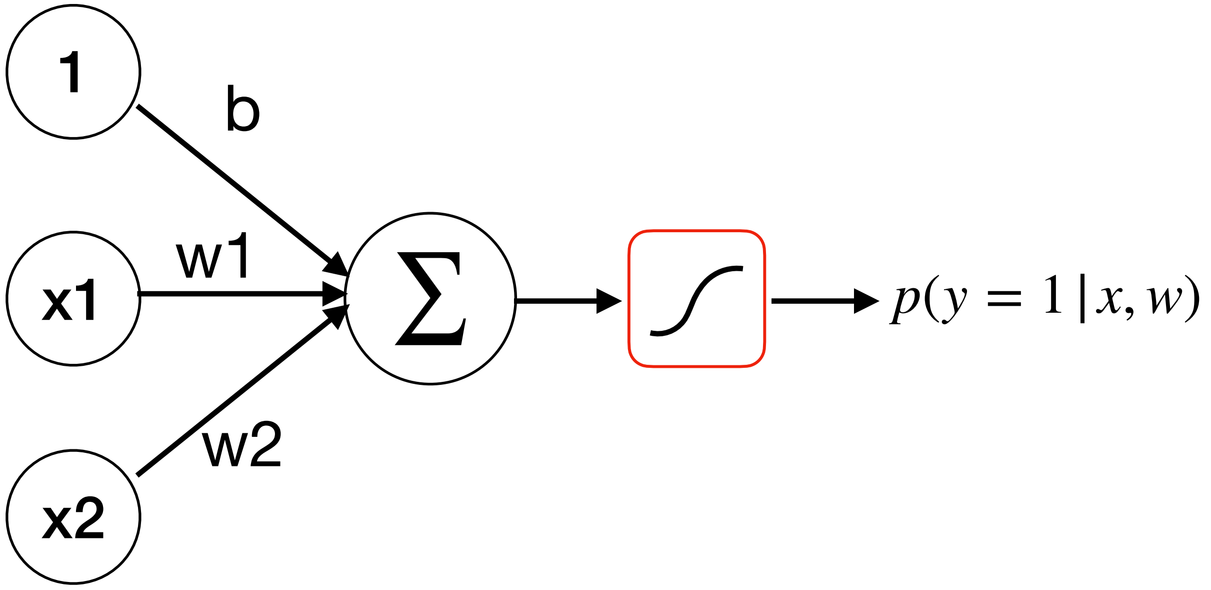
$$\mu(x) = E[y|x] = p(y = 1|x)$$

- Con

$$\mu(x) = sigm(w^T x) = b + w_1 x_1 + w_2 x_2 = w^T x$$

$$x = (1, x_1, x_2)$$

Nota Historica: Si la sigmoide es reemplazada por un umbral lineal (y=1 si $w^T x > t$)
esto es conocido como **perceptron** (*Rosenblat,1960*).

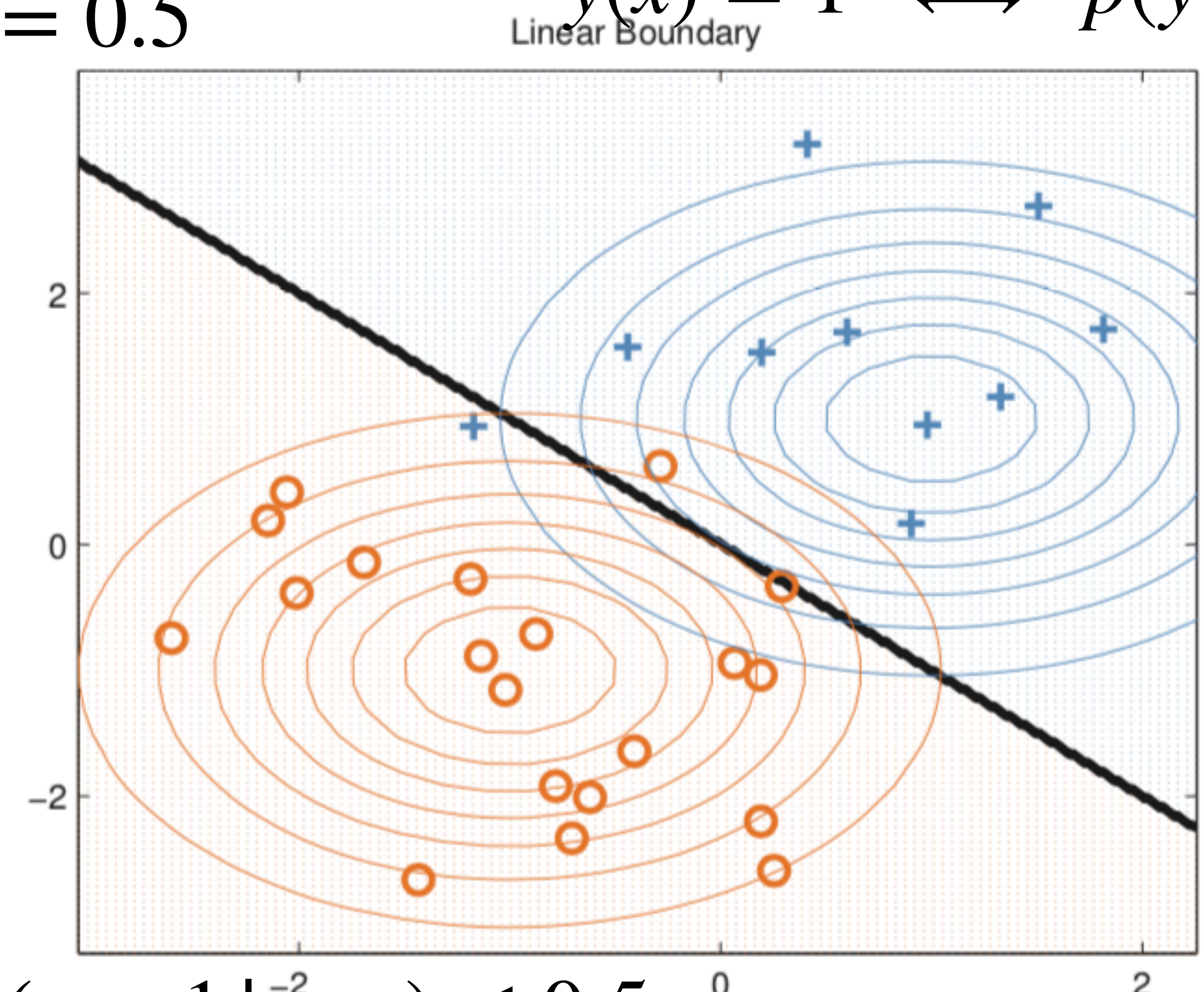


Red Neuronal de 1 Capa

$$\mu(x) = sigm(w^T x) = \frac{1}{1 + e^{-w^T x}} = \frac{e^{w^T x}}{e^{w^T x} + 1}$$

$$p(y = 1 \,|\, x, w) = 0.5$$

$$\hat{y}(x) = 1 \iff p(y = 1 \,|\, x, w) > 0.5$$



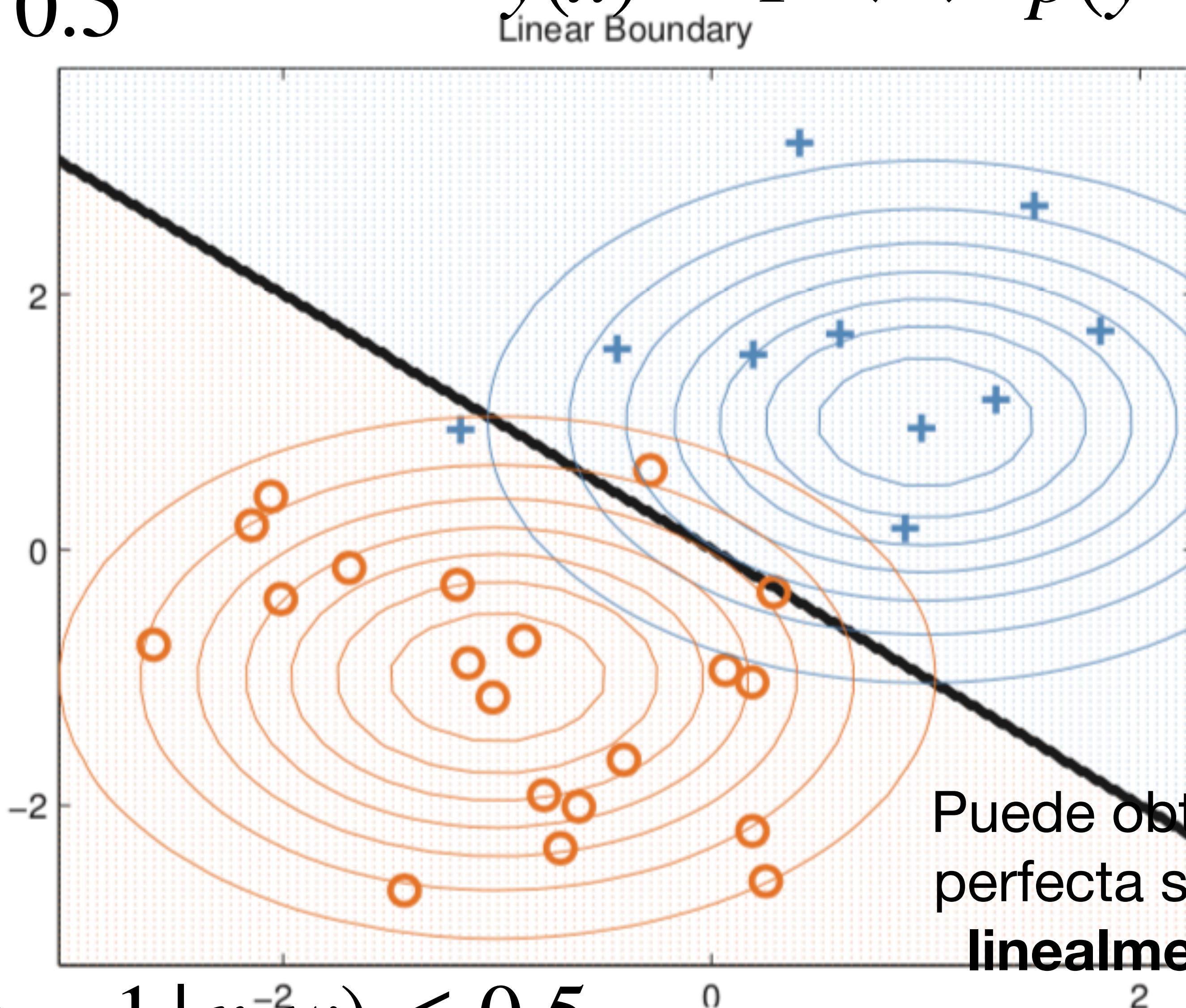
$$\hat{y}(x) = 0 \iff p(y = 1 \,|\, x, w) \leq 0.5$$

Red Neuronal de 1 Capa

$$\mu(x) = sigm(w^T x) = \frac{1}{1+e^{-w^T x}} = \frac{e^{w^T x}}{e^{w^T x}+1}$$

$$p(y=1\,|\,x,w) = 0.5$$

$$\hat{y}(x) = 1 \iff p(y=1\,|\,x,w) > 0.5$$

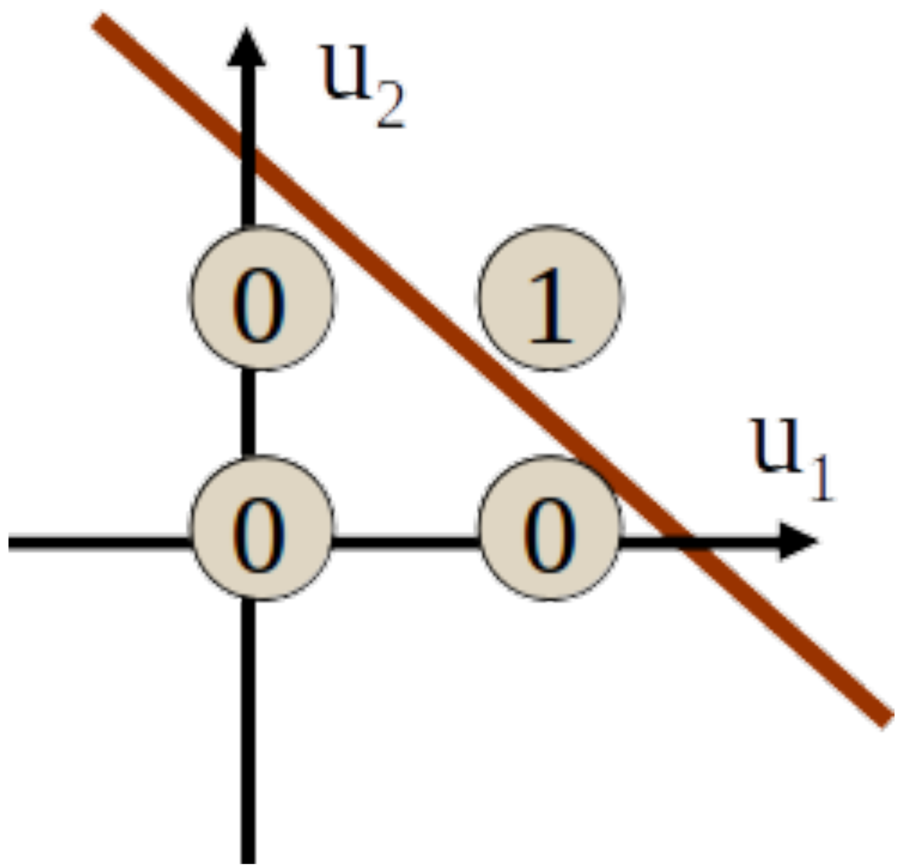


Puede obtener performance perfecta si los ejemplos son **linealmente separables.**

$$\hat{y}(x) = 0 \iff p(y=1\,|\,x,w) \leq 0.5$$

Red Neuronal de 1 Capa

$$\mu(x) = w^T x \qquad \hat{y}(x) = 1 \iff \mu(x) \geq 0$$



Puede separar un AND?

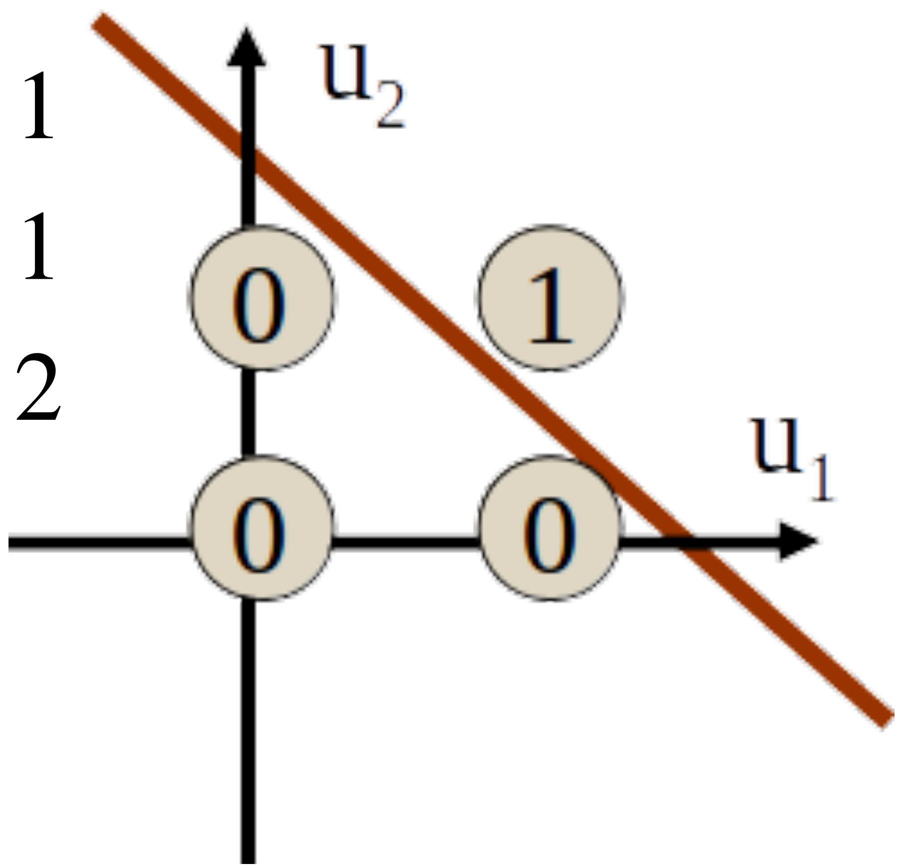
Red Neuronal de 1 Capa

$$\mu(x) = w^T x \qquad \hat{y}(x) = 1 \iff \mu(x) \geq 0$$

$w_1 = 1$

$w_2 = 1$

$w_0 = -2$



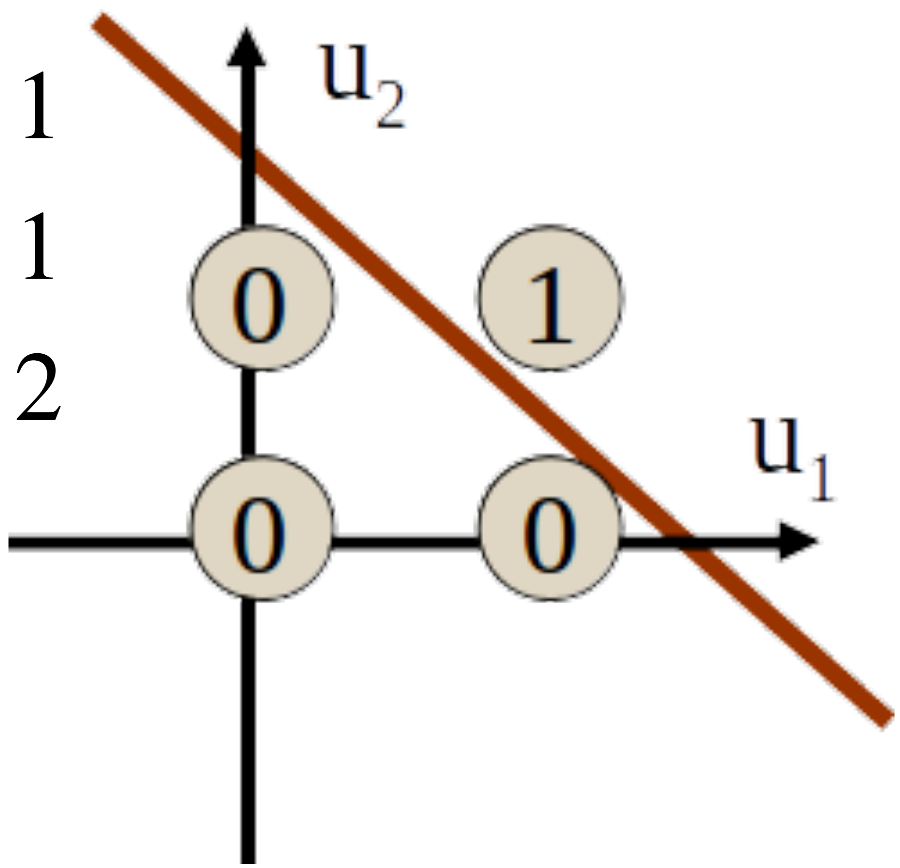
Puede separar un AND?

Sí!

Red Neuronal de 1 Capa

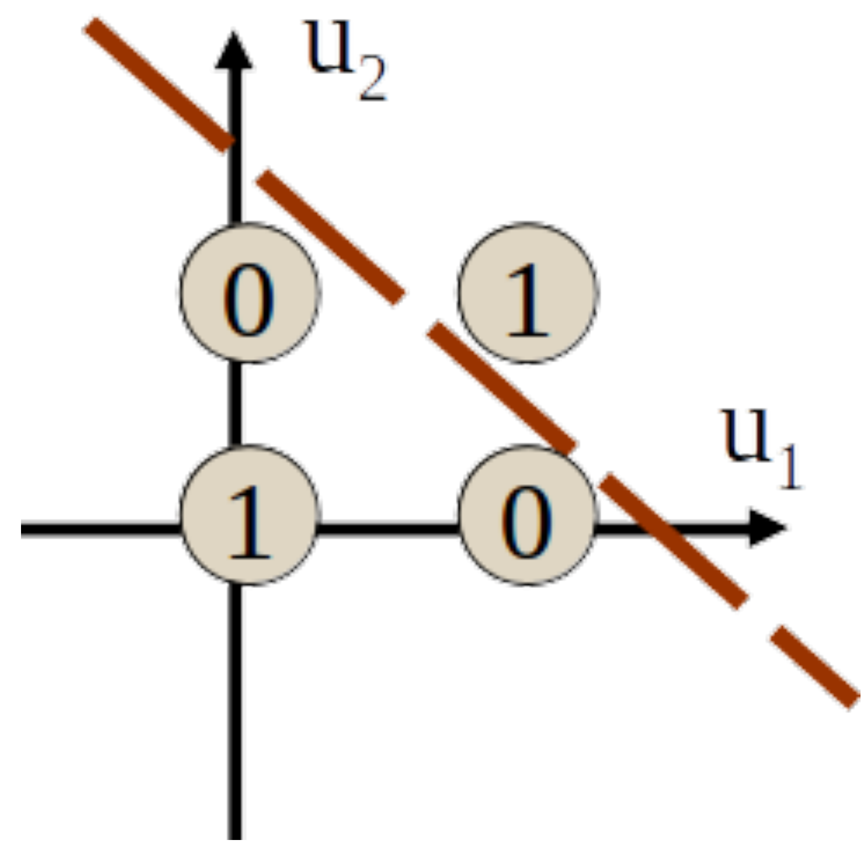
$$\mu(x) = w^T x \qquad \hat{y}(x) = 1 \iff \mu(x) \geq 0$$

$$w_1 = 1$$
$$w_2 = 1$$
$$w_0 = -2$$



Can learn an AND?

Yes!

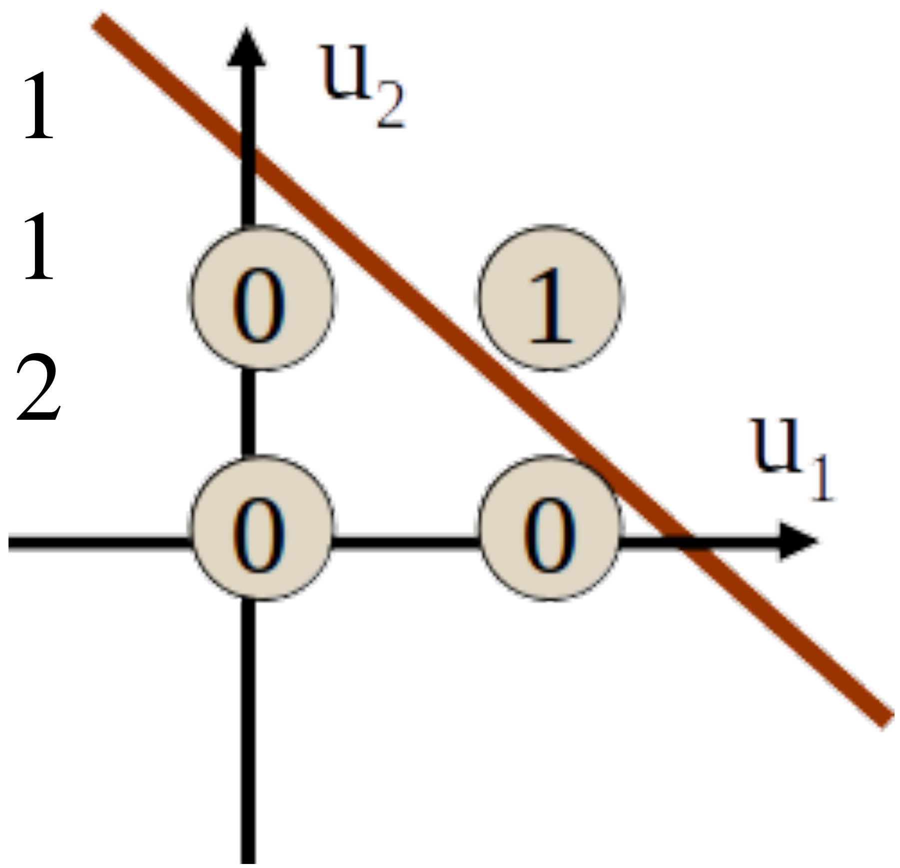


Puede separar un XOR?

Red Neuronal de 1 Capa

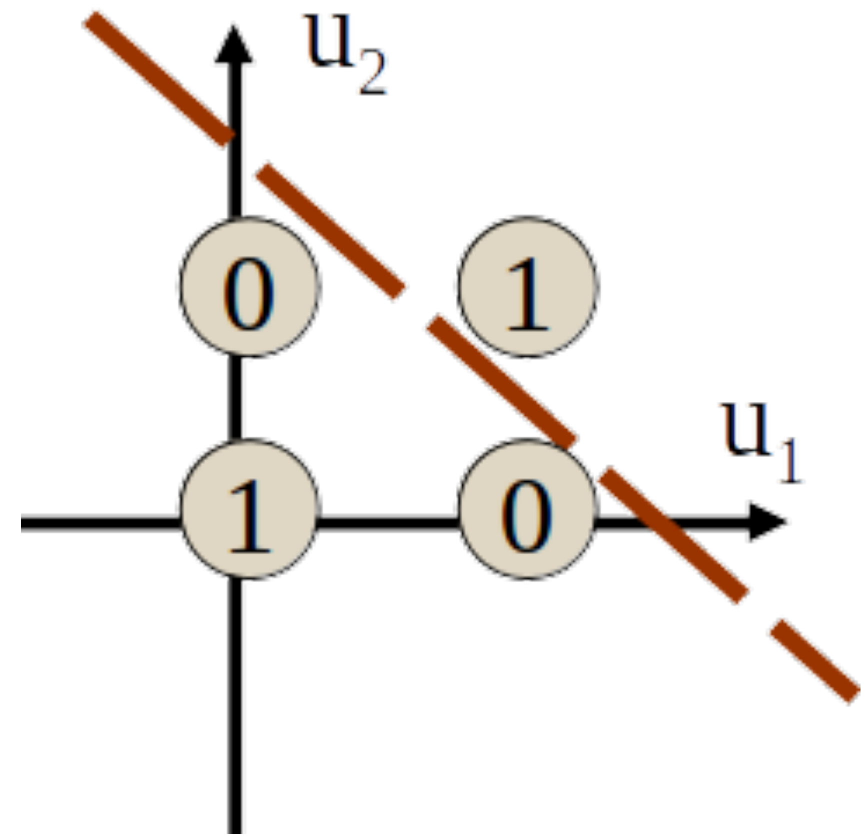
$$\mu(x) = w^T x \qquad \hat{y}(x) = 1 \iff \mu(x) \geq 0$$

$$w_1 = 1$$
$$w_2 = 1$$
$$w_0 = -2$$



Can learn an AND?

Yes!



Can learn a XOR?

No! *(Minsky & Papert 1969)*

Red Neuronal Multicapa

- Podemos crear fronteras de decisión más complejas usando funciones más complejas:

$$p(y|x, w) = Ber(y|\mu(x))$$

$$\mu(x) = sigm(w^T z(x))$$

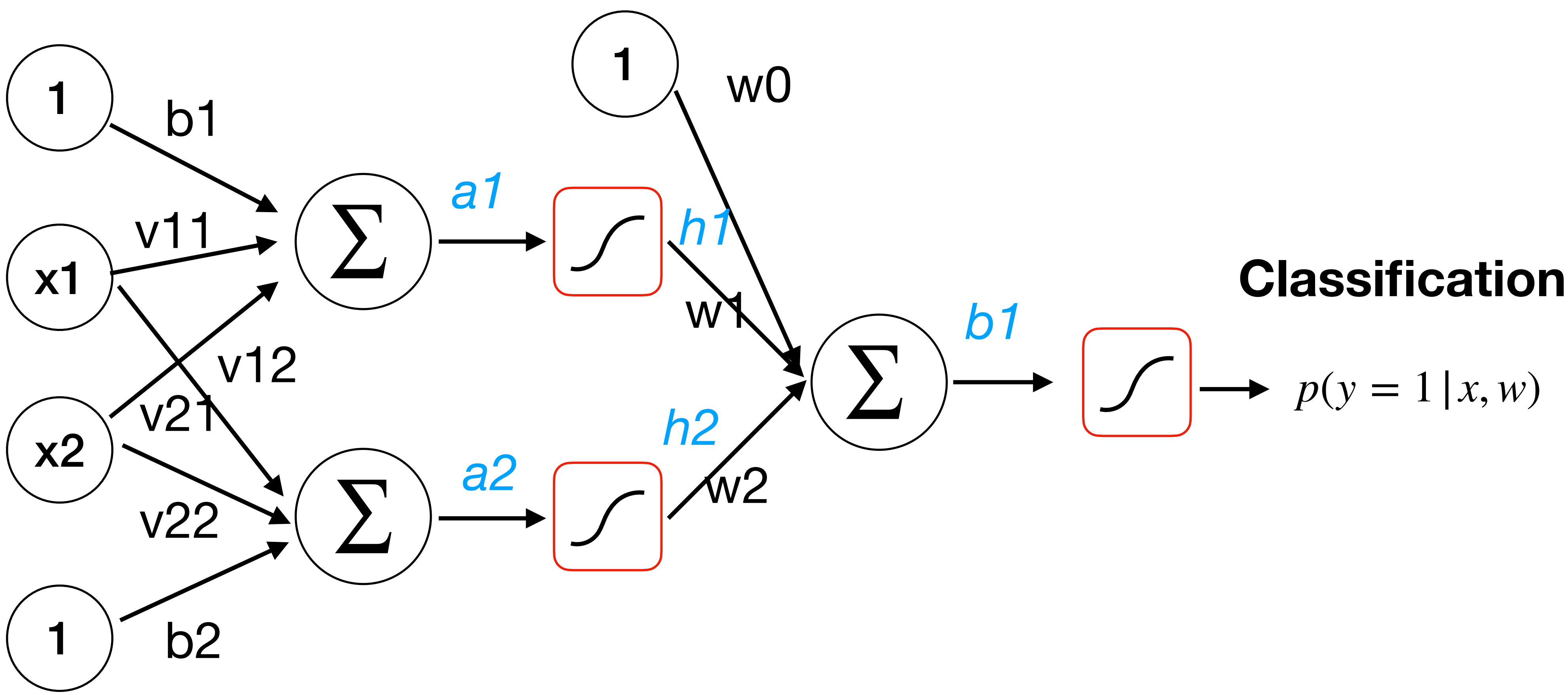
$$z(x) = g(Vx) = [g(v_1^T x), \ldots, g(v_H^T x)]$$

- Donde g es una no linealidad por elemento, por ejemplo una sigmoidal

$$g(v_1^T x) = sigm(v_1^T x)$$

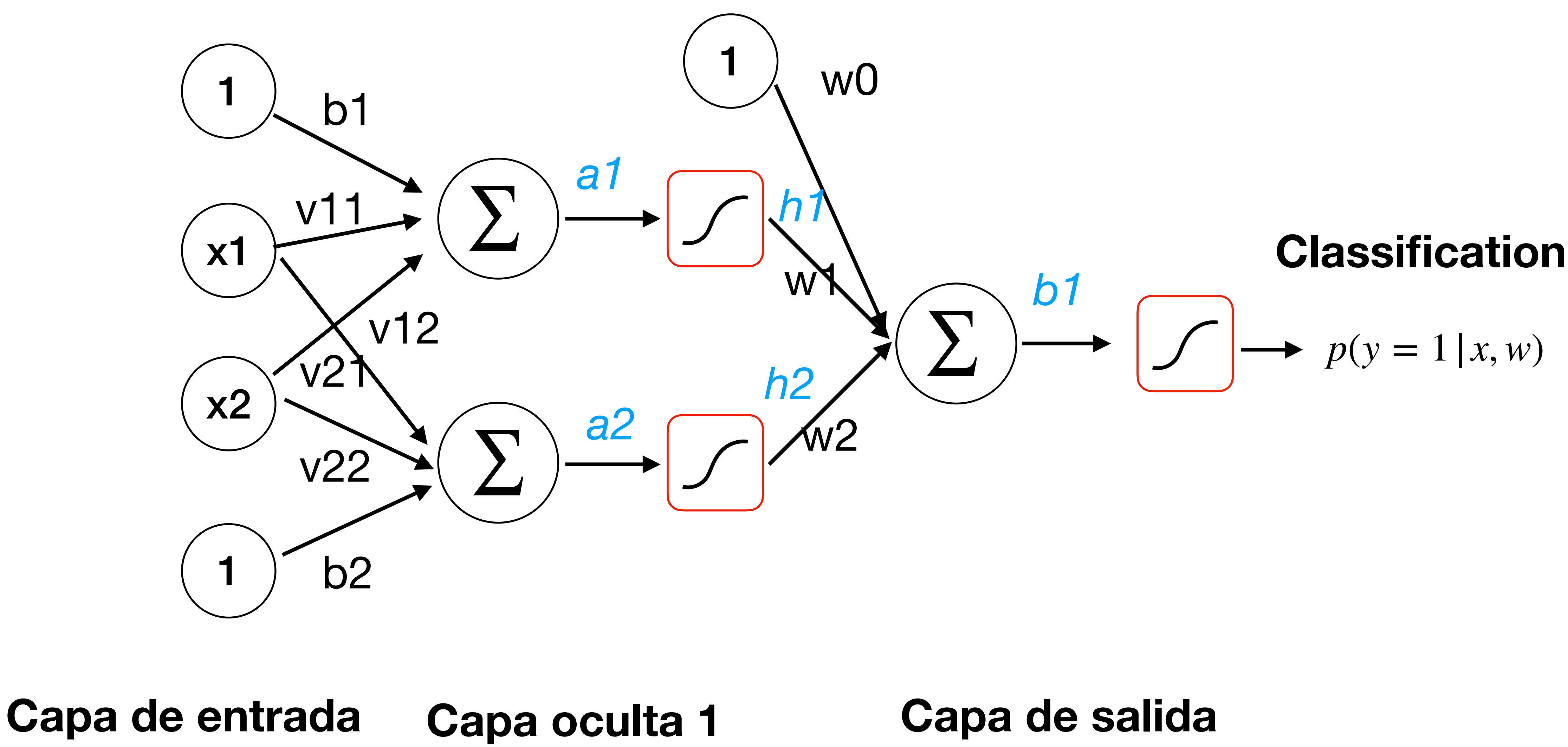
Red Neuronal Multicapa

$$\mu(x) = sigm(w^T z(x)) \quad z(x) = g(Vx) = [g(v_1^T x), \ldots, g(v_H^T x)]$$



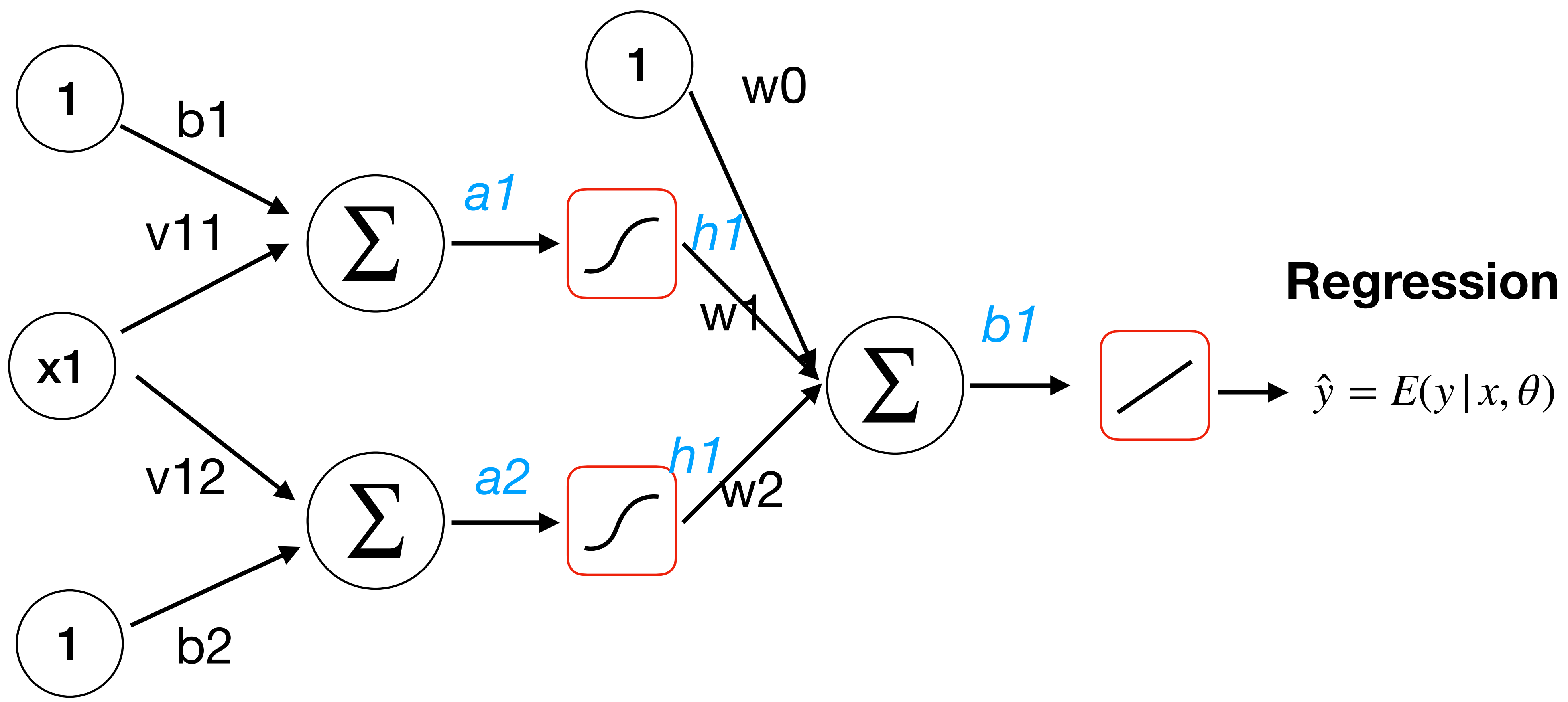
Red Neuronal Multicapa

$$\mu(x) = sigm(w^T z(x)) \quad z(x) = g(Vx) = [g(v_1^T x), \ldots, g(v_H^T x)]$$



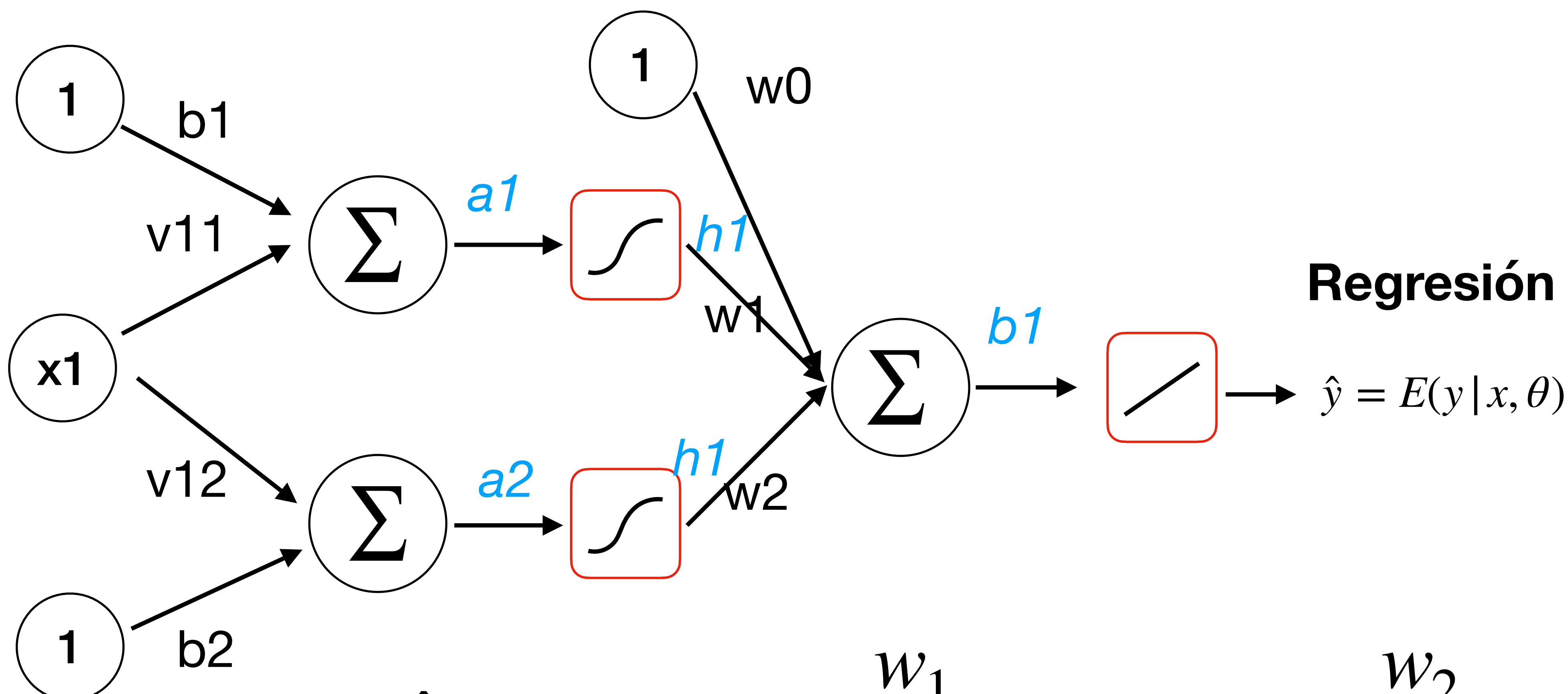
Red Neuronal Multicapa

$$\mu(x) = w^T z(x) \qquad z(x) = g(Vx) = [g(v_1^T x), \ldots, g(v_H^T x)]$$



Red Neuronal Multicapa

$$\mu(x) = w^T z(x) \qquad z(x) = g(Vx) = [g(v_1^T x), \ldots, g(v_H^T x)]$$



$$\hat{y} = w_0 + \frac{w_1}{1 + e^{-b_1 - v_{11}x_1}} + \frac{w_2}{1 + e^{-b_2 - v_{12}x_1}}$$

Red Neuronal Multicapa

$$a_1 = b_1 + v_1 x_1$$

$$a_2 = b_2 + v_2 x_1$$

$$o_1 = \frac{1}{1 + e^{-a_1}} = \frac{1}{1 + e^{-b_1 - v_1 x_1}}$$

$$o_2 = \frac{1}{1 + e^{-a_2}} = \frac{1}{1 + e^{-b_2 - v_2 x_1}}$$

$$\begin{aligned} \hat{y} &= w_0 + w_1 o_1 + w_2 o_2 \\ &= w_0 + \frac{w_1}{1 + e^{-b_1 - v_1 x_1}} + \frac{w_2}{1 + e^{-b_2 - v_2 x_1}} \end{aligned}$$

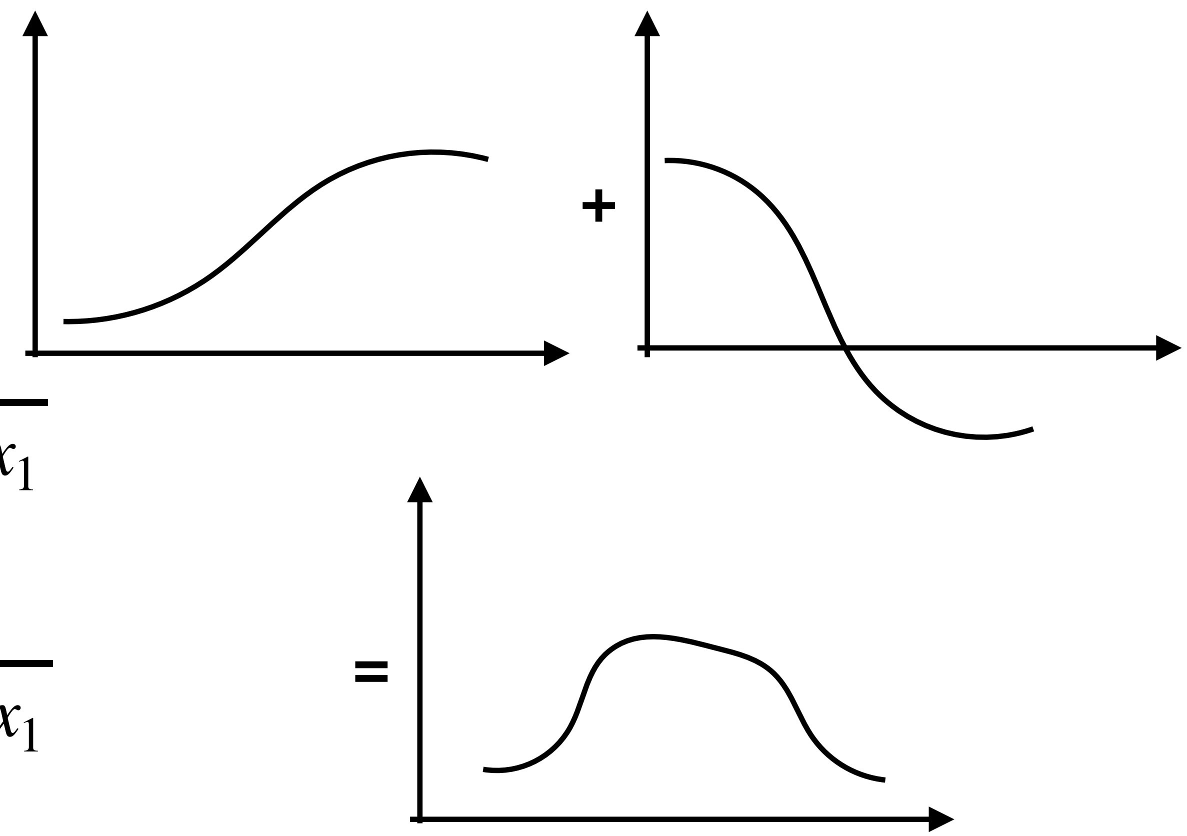
Red Neuronal Multicapa

$$a_1 = b_1 + v_1 x_1$$

$$a_2 = b_2 + v_2 x_1$$

$$o_1 = \frac{1}{1 + e^{-a_1}} = \frac{1}{1 + e^{-b_1 - v_1 x_1}}$$

$$o_2 = \frac{1}{1 + e^{-a_2}} = \frac{1}{1 + e^{-b_2 - v_2 x_1}}$$



$$\hat{y} = w_0 + w_1 o_1 + w_2 o_2$$

Deforma verticalmente

$$= w_0 + \frac{w_1}{1 + e^{-b_1 - v_1 x_1}} + \frac{w_2}{1 + e^{-b_2 - v_2 x_1}}$$

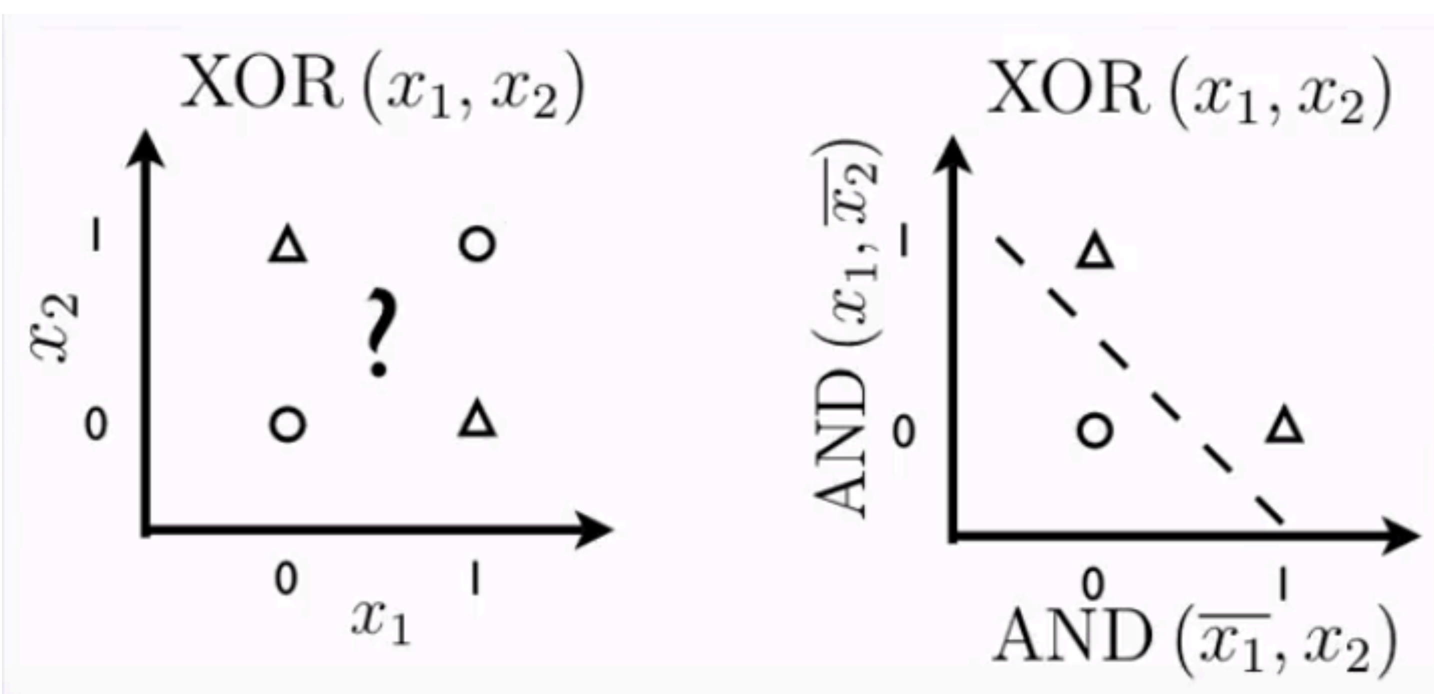
Altura

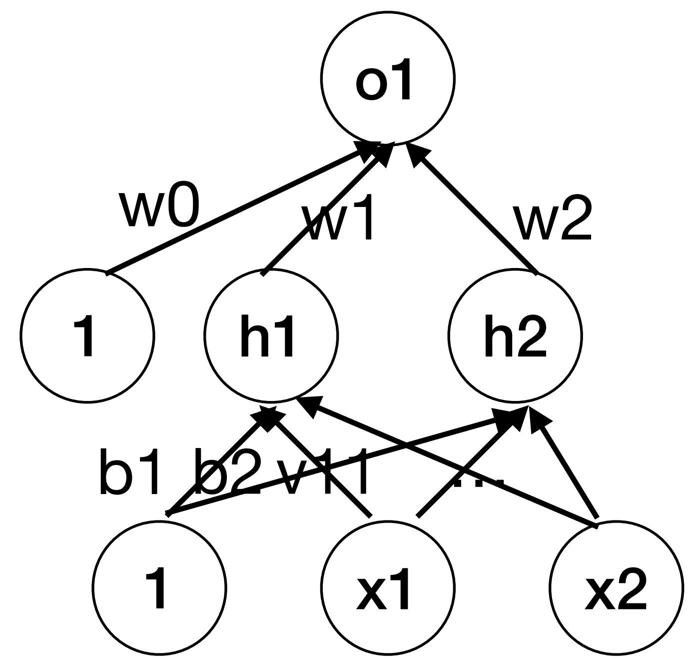
Deforma horizontalmente

Mueve horizontalmente

Red Neuronal Multicapa

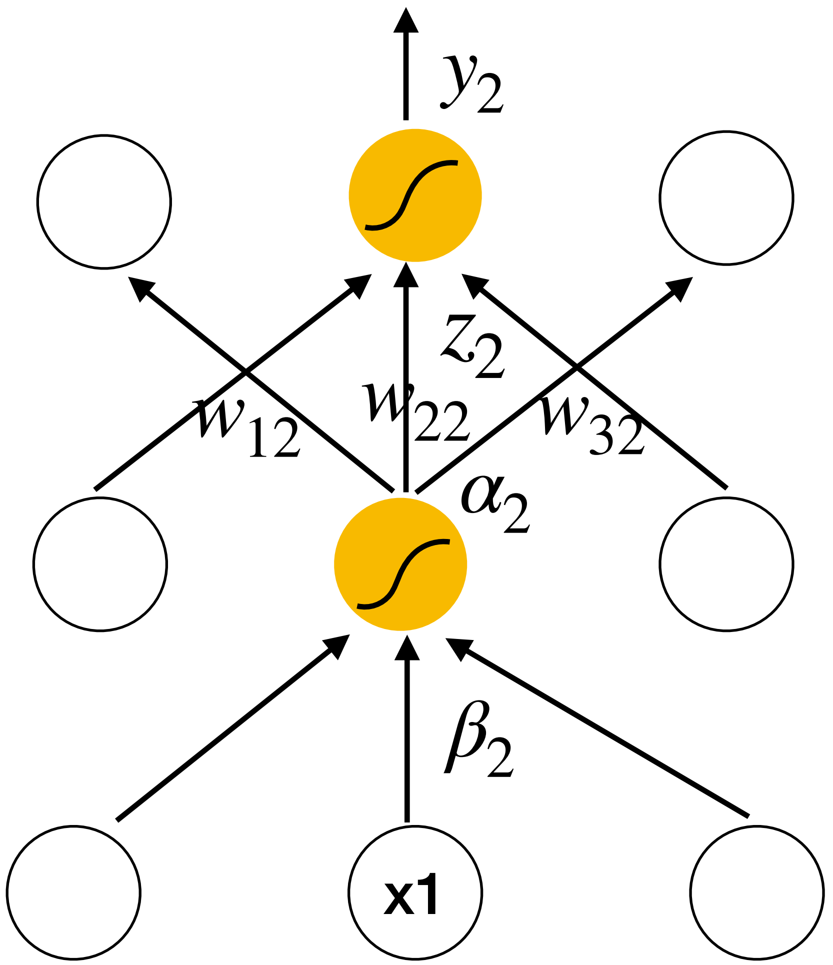
- Otra interpretación: Las capas ocultas aprenden una nueva **representación** de la entrada.
- Consider the XOR example





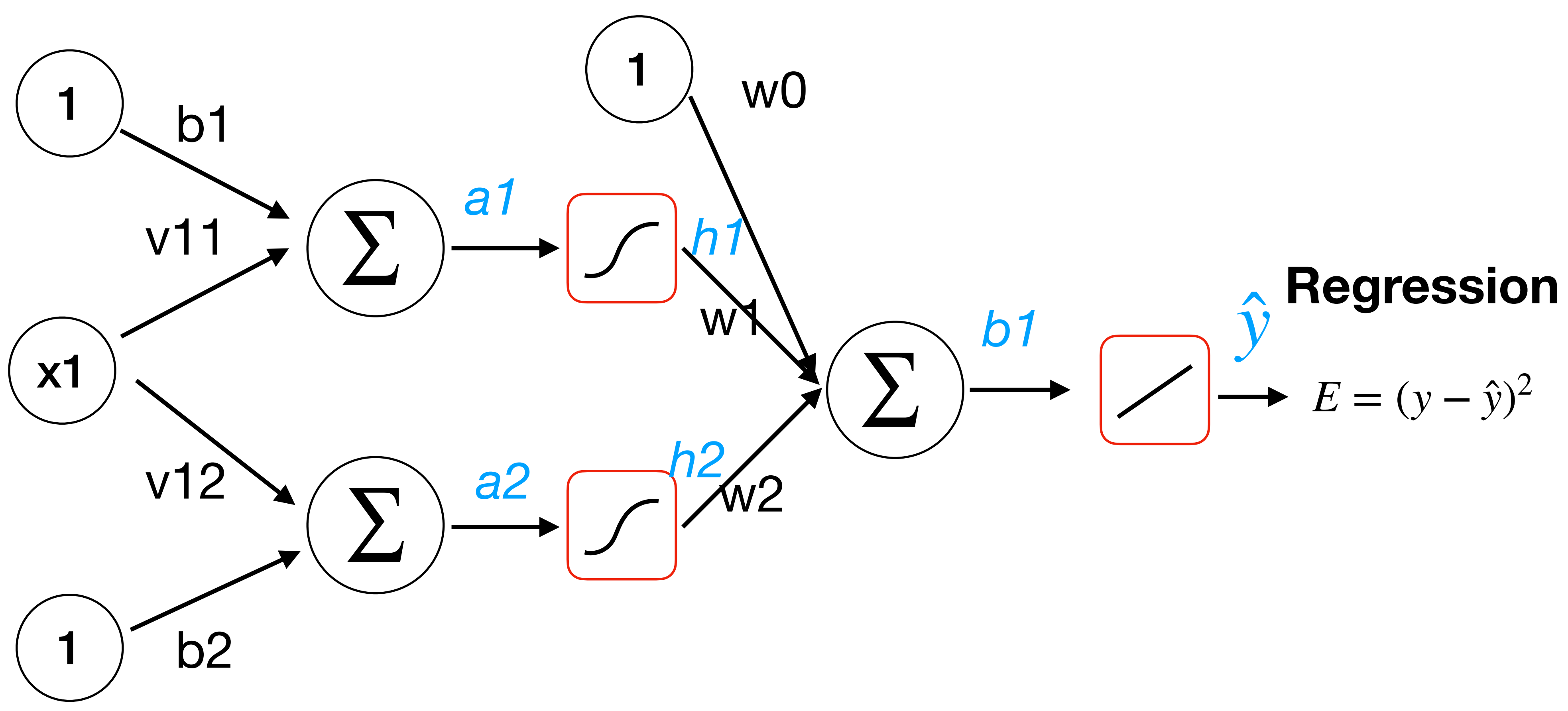
Red Neuronal Multicapa

- **Backpropagation:** Como en la regresión logística necesitamos aprender por **gradiente descendiente** todos los pesos en la red:



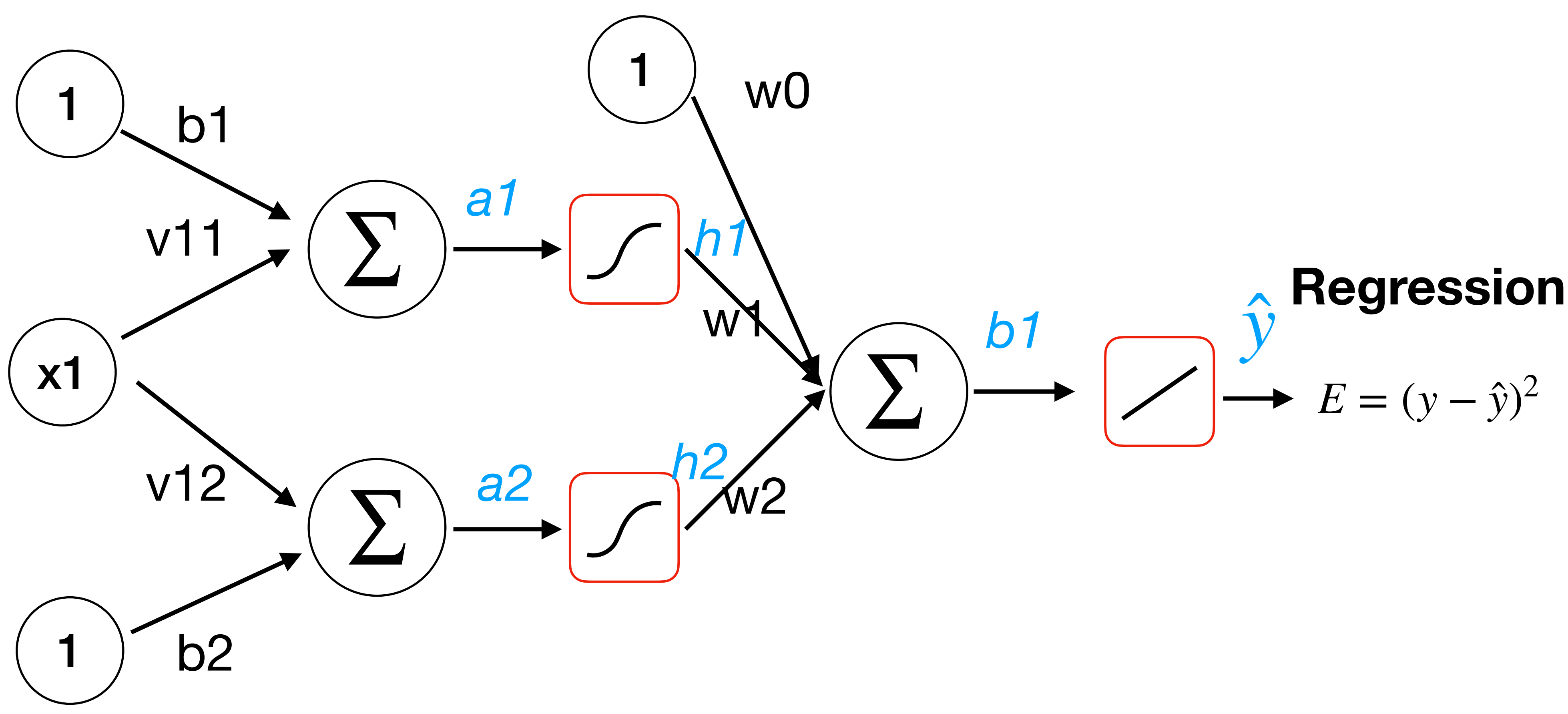
Red Neuronal Multicapa

- **Backpropagation**



Red Neuronal Multicapa

- **Backpropagation**

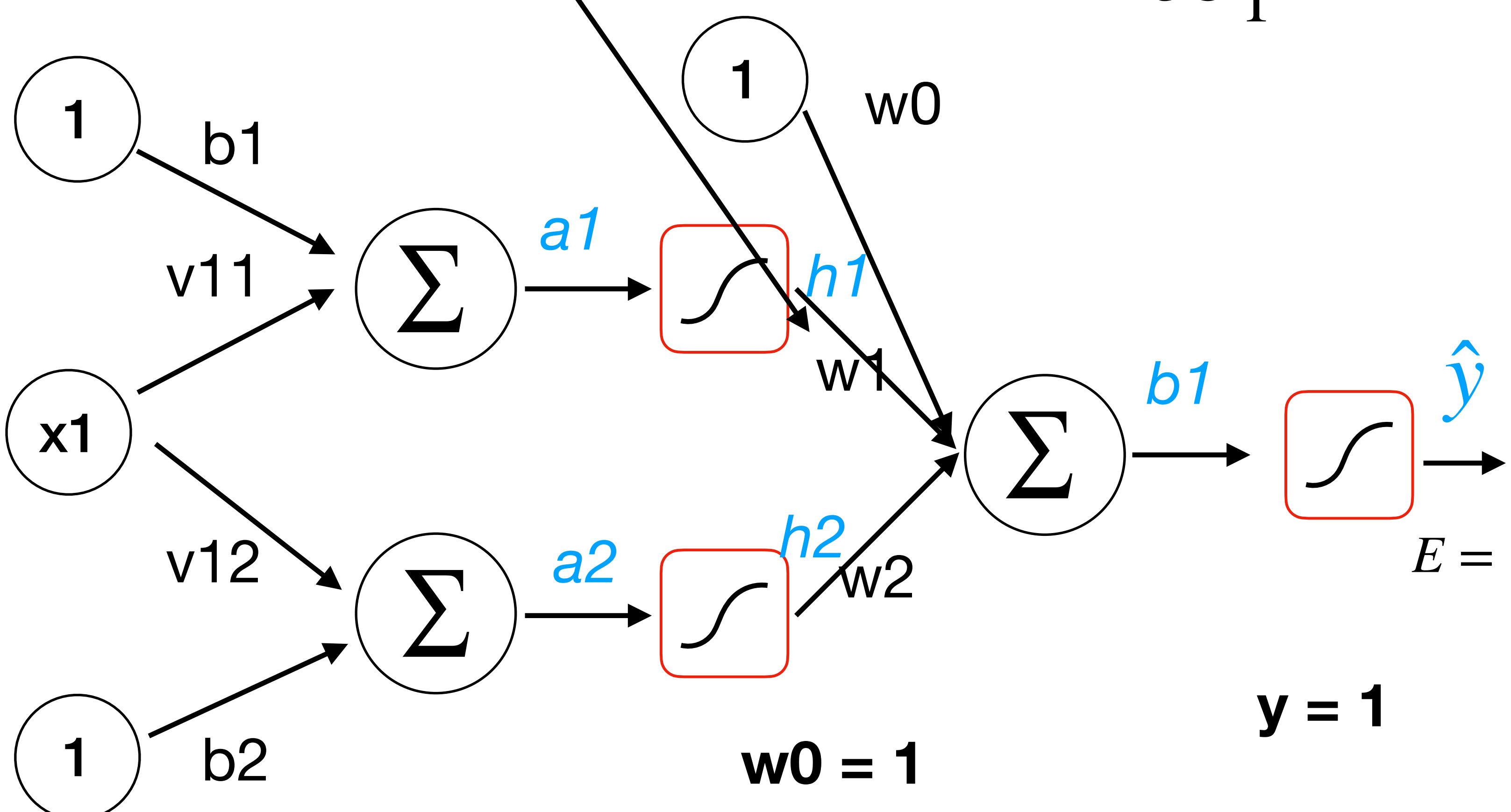


Red Neuronal Multicapa

- **Backpropagation**

Cual es la actualización a w1?

$$\frac{\partial E}{\partial b_1} = (\hat{y} - y)$$



Clasificación

$E = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$

b1 = 0
v11 = 1
v12 = 2
b2 = 1

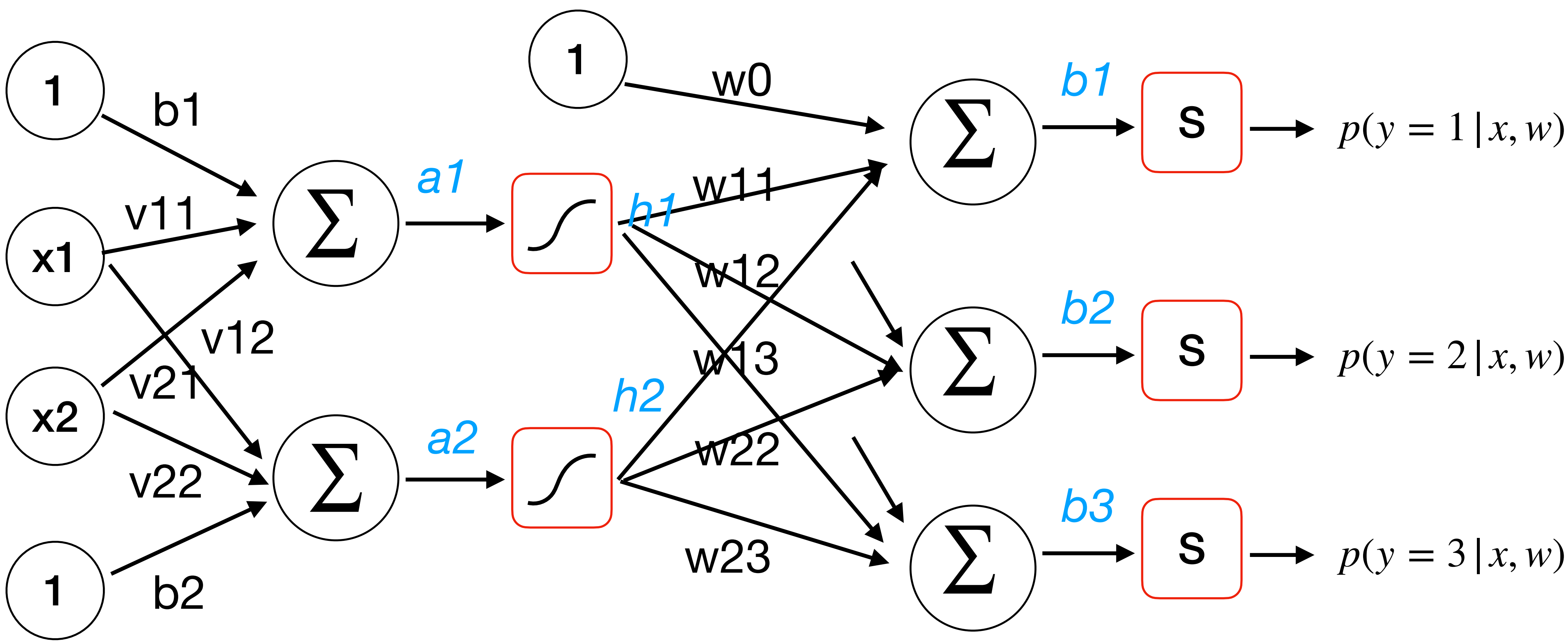
w0 = 1
w1 = 2
w2 = -1

y = 1

sigm([1,1.77, 2, 5]) = [0.73, 0.85, 0.88, 0.99]

Red Neuronal Multicapa

$$\mu(x)_c = S(w^T z(x))_c = \frac{\exp(w_c^T z(x))}{\sum_{c'} \exp(w_{c'}^T z(x))}$$



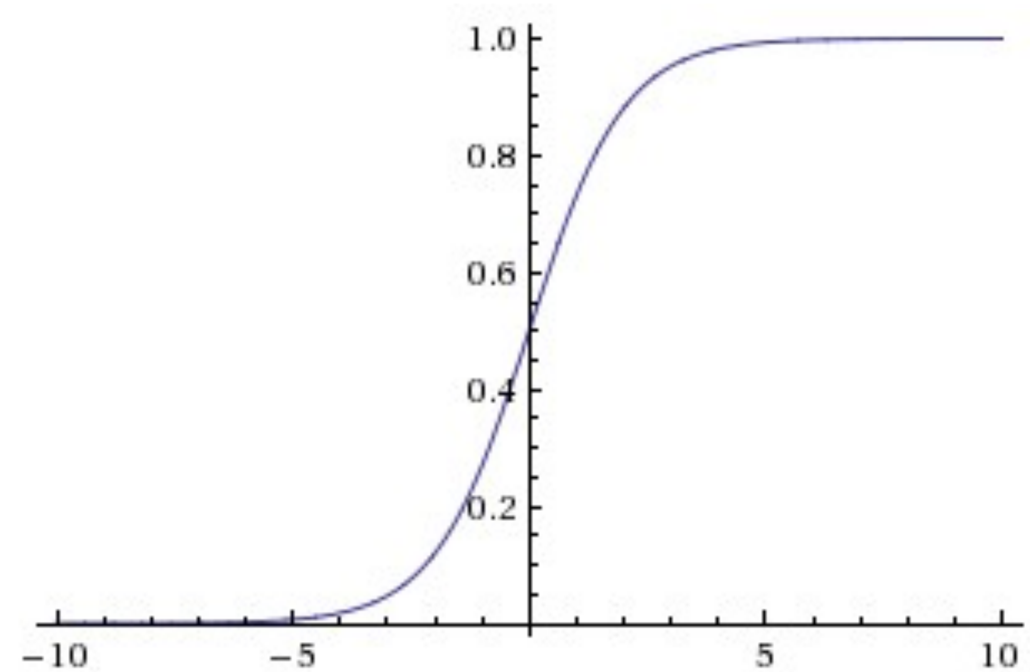
$$L(y, \hat{y}) = -\log P(y \mid x, \theta) = \sum_{i=1}^{N} \sum_{j=1}^{C} I(y_i = j) \log \frac{e^{b_{ij}}}{\sum_{j'} e^{b_{ij'}}}$$

Red Neuronal Multicapa

Función de activación:

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

- La **sigmoide** es raramente usada como función de activación actualmente.
- Satura y mata los gradientes.
- No se centra en cero, moverá las entradas en capas superiores.

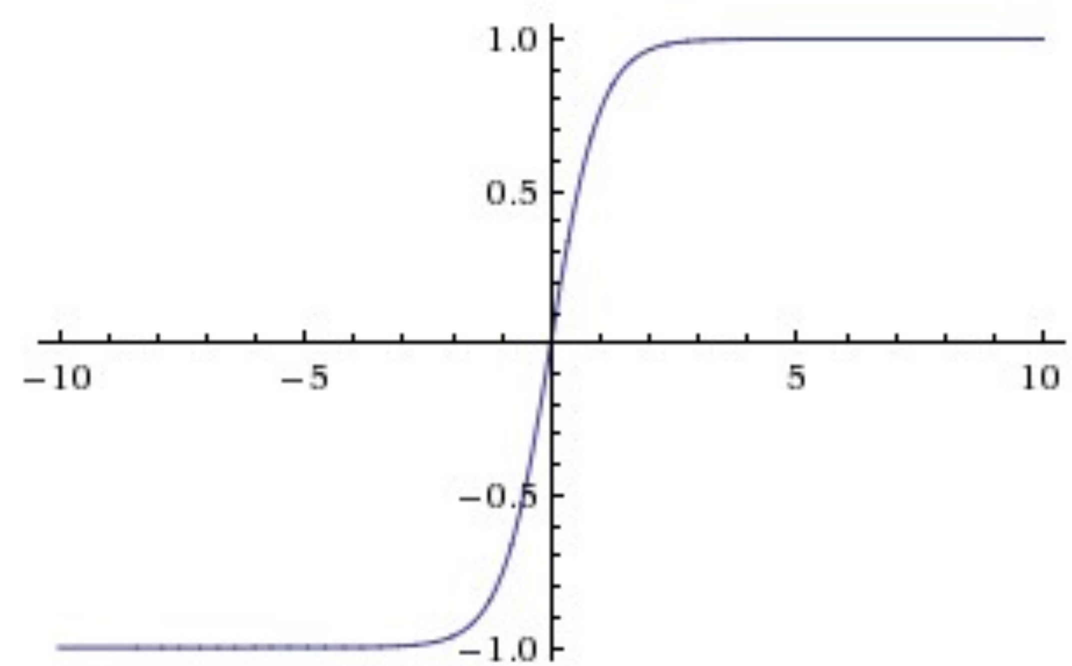


Red Neuronal Multicapa

Función de activación:

$$\mu(x) = tanh(x) = 2sigm(2x) - 1$$

- **Tanh** es más usada que la sigmoide.
- También satura y mata los gradientes.
- Es centrada en cero.
- Es una sigmoide escalada.

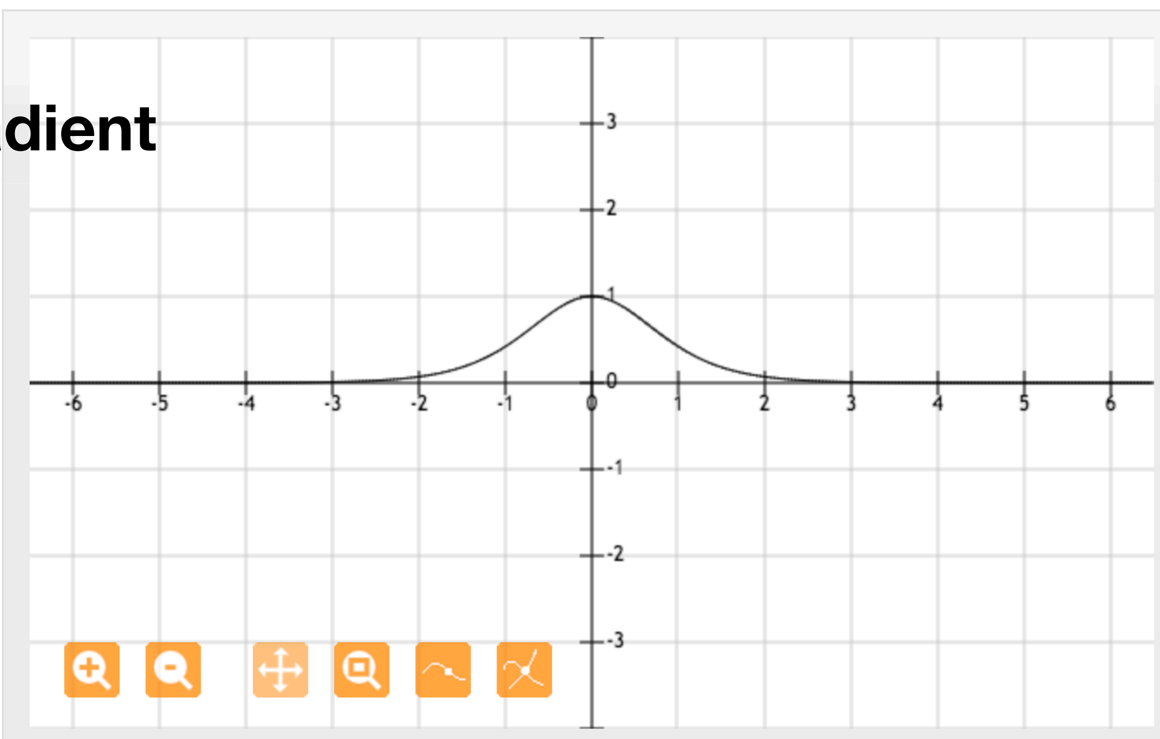


Red Neuronal Multicapa

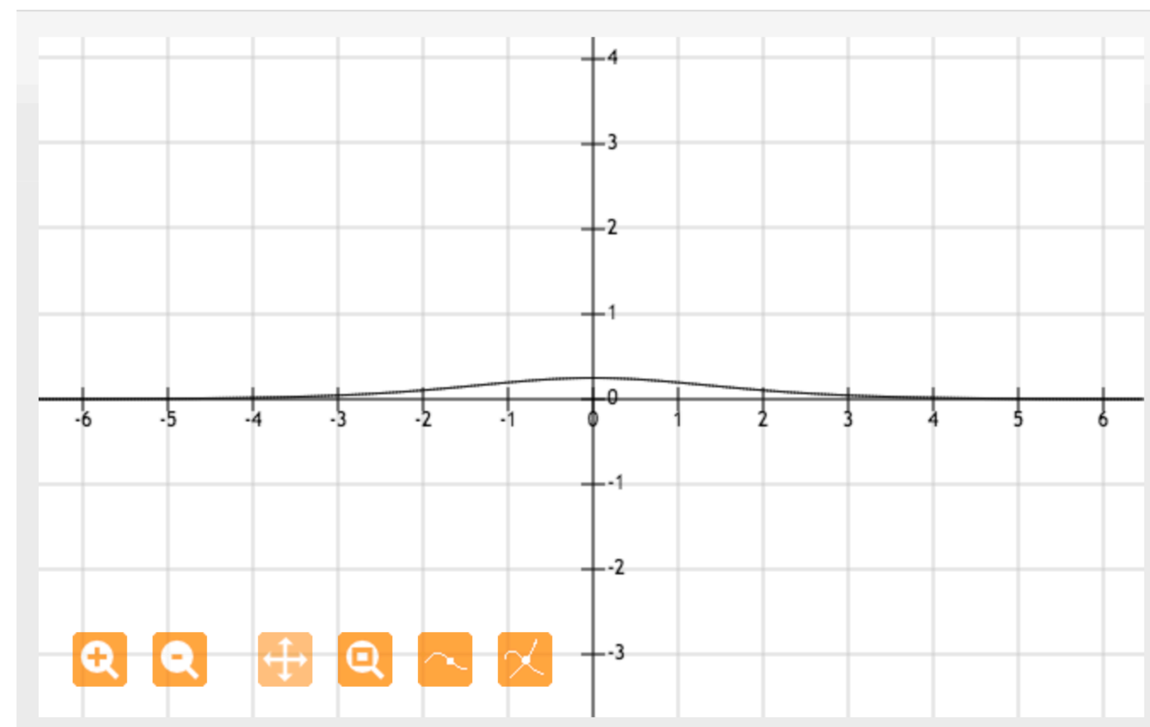
$$\mu(x) = tanh(x)$$

$$\mu(x) = sigm(x)$$

gradient



Input

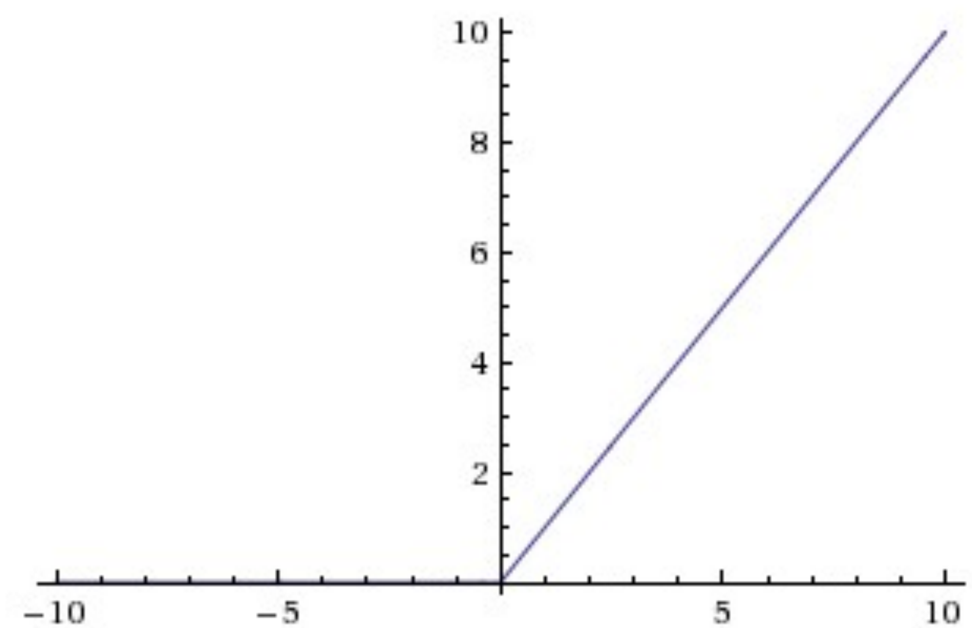


Red Neuronal Multicapa

Función de activación:

- **ReLU** es la función de activación mas usada.
- Operación más rápida ya que sólo involucra un umbral.
- No se satura.
- Produce un efecto de sparsity.
- Pero puede hacer explotar las activaciones o anularlas todas (se debe usar un learning rate menor.)

$$\mu(x) = max(0,x)$$



Con esta debes intentar primero

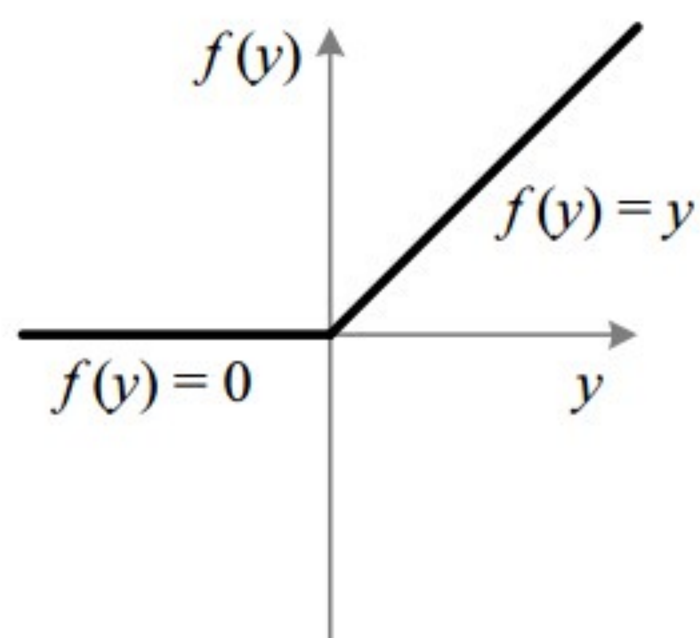
Red Neuronal Multicapa

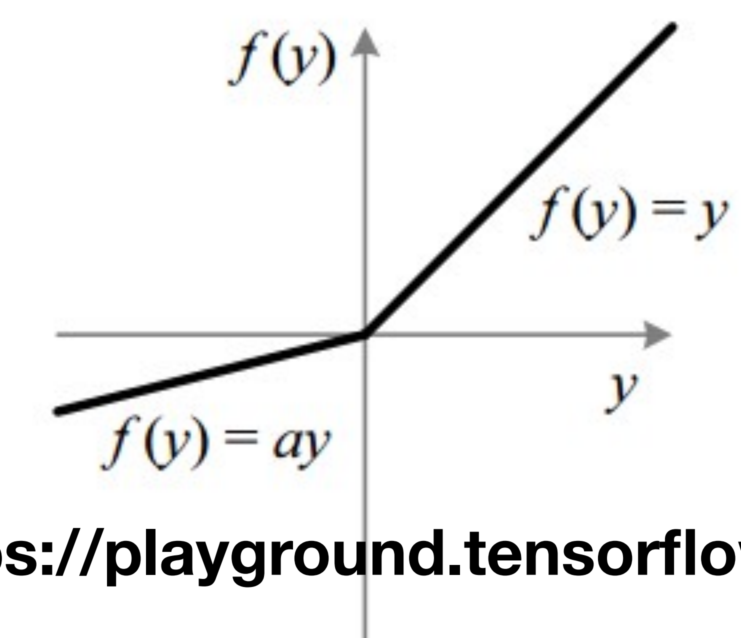
Activation function:

- **LeakyReLU** similar a la ReLU.
- Eliminar el problema de ReLU desvaneciendo.

$$\mu(x) = I(x < 0)(ax) + I(x \geq 0)(x)$$

- Mejoras no son consistentes.





https://playground.tensorflow.org/

Red Neuronal Multicapa

Inicialización de los pesos

- **Mala idea:** Inicializar los pesos en 0.
  - Si dos neuronas tienen el mismo sesgo y los mismos pesos, siempre recibirán el mismo gradiente.
- **Better idea:** Usar pesos aleatorios.
  - Por ejemplo Normal(0,1).

Red Neuronal Multicapa

Inicialización de pesos:

- Varianza de la unidad crece con el número de entradas. Para resolver:

$$w = N(0,1/n_{in}) = N(0,1)\frac{1}{\sqrt{n_{in}}}$$

Red Neuronal Multicapa

Inicialización de pesos:

- Varianza de la unidad crece con el número de entradas. Para resolver:

$$w = N(0,1/n_{in}) = N(0,1)\frac{1}{\sqrt{n_{in}}}$$

Prueba: Asumiendo pesos y entradas centradas.

Red Neuronal Multicapa

Inicialización de pesos:

- Prueba (w e x son independientes y ambas centradas):

$$V(\sum_i w_i x_i) =$$

Asumiendo que son identicamente
Distribuidas e independientes

Red Neuronal Multicapa

Inicialización de pesos:

- Varianza de la unidad crece con el número de entradas. Para resolver:

$$w = N(0,1/n_{in}) = N(0,1)\frac{1}{\sqrt{n_{in}}}$$

- Haciendo el mismo análisis para la pasada backward y forward:

$$w = N(0,2/(n_{in} + n_{out})) = N(0,1)\frac{2}{\sqrt{(n_{in} + n_{out})}}$$ *(Glorot et al.)*

- ReLU no esta centrada así que puede ser reemplazada por:

$$w = N(0,1)\sqrt{\frac{2}{n_{in}}}$$ *(He et al.)*

Red Neuronal Multicapa

Inicialización de pesos: User Glorot o He

- Varianza de la unidad crece con el número de entradas. Para resolver:

$$w = N(0,1/n_{in}) = N(0,1)\frac{1}{\sqrt{n_{in}}}$$

- Haciendo el mismo análisis para la pasada backward y forward:

$$w = N(0,2/(n_{in} + n_{out})) = N(0,1)\frac{2}{\sqrt{(n_{in} + n_{out})}}$$

(Glorot et al.)

- ReLU no esta centrada así que puede ser reemplazada por:

$$w = N(0,1)\sqrt{\frac{2}{n_{in}}}$$

(He et al.)