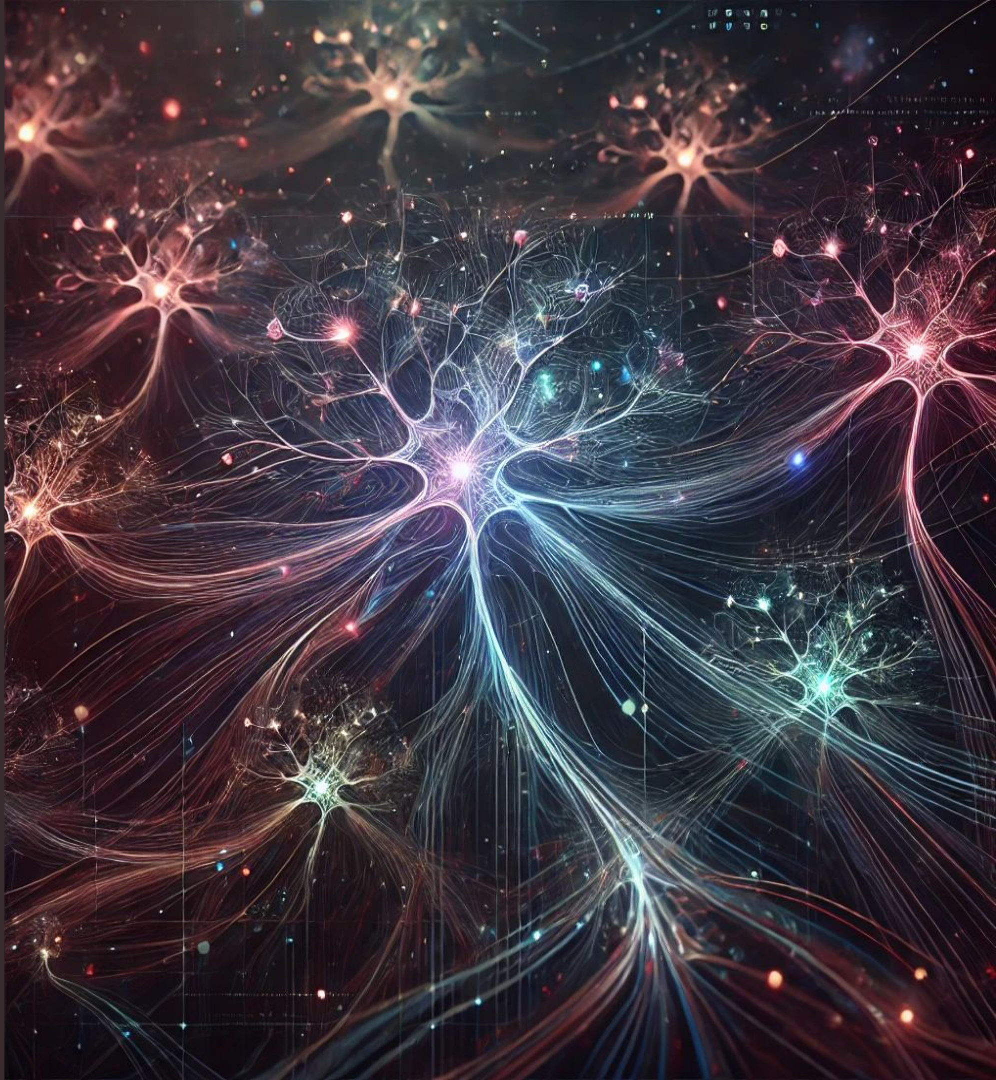
Perceptrón Multicapa

Sistemas de Inteligencia Artificial

Primer Cuatrimestre 2025

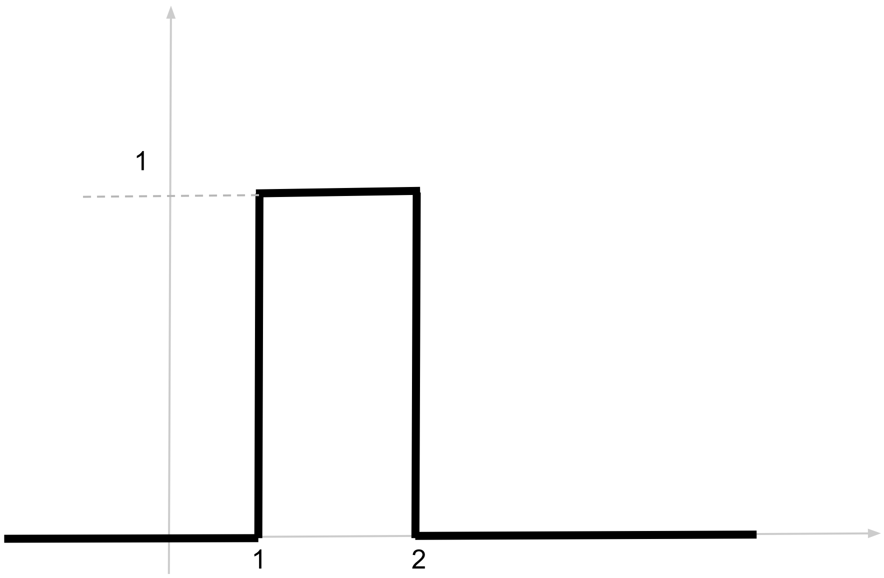
Rodrigo Ramele
Eugenia Piñeiro
Alan Pierri
Santiago Reyes
Marina Fuster
Luciano Bianchi
Marco Scilipoti
Paula Oseroff
Joaquín Girod



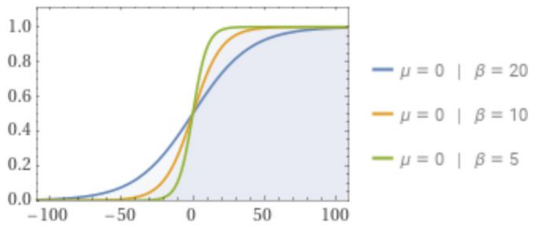
RESUMEN DE LA CLASE ANTERIOR

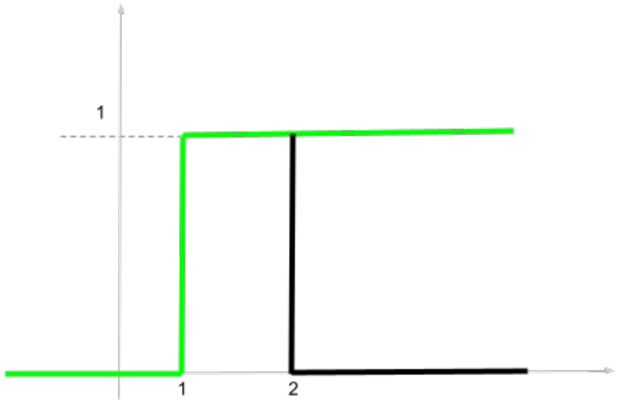
- ¿Quiénes son McCulloch y Pitts? ¿Cuál fue su aporte?
- ¿Qué tipo de problemas podemos resolver con el perceptrón simple escalón?
- ¿Con qué mecanismo encuentro los pesos sinápticos y el bias?
- ¿Qué tipo de problemas puedo resolver al cambiar la función de activación? Por ejemplo: por una lineal o no lineal de la familia de sigmoideas.

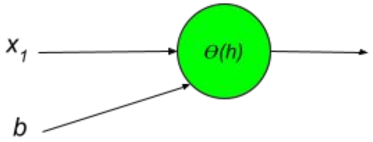
LIMITACIONES DEL PERCEPTRÓN SIMPLE



BUSCANDO ALTERNATIVAS…

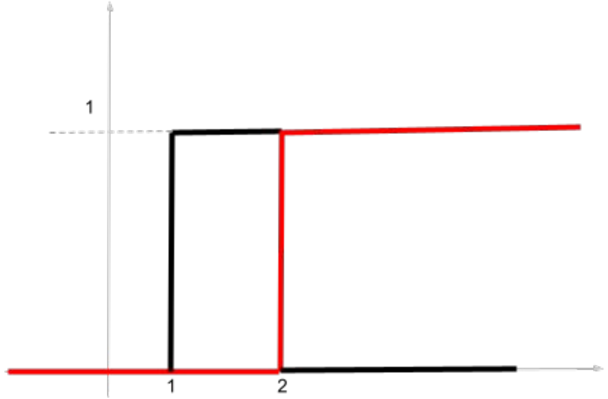


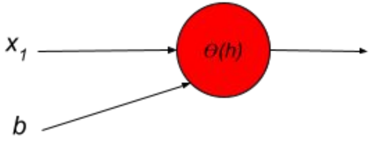




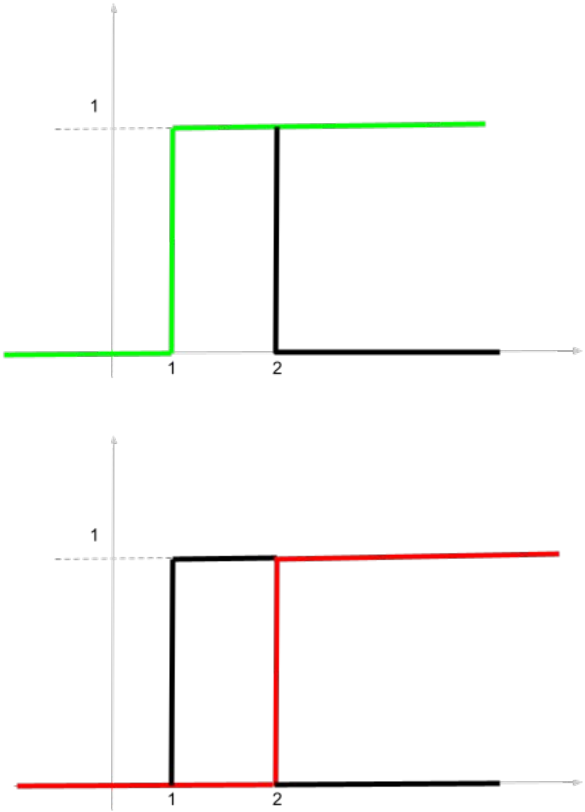
$$\theta(x) = \frac{1}{1+exp^{-2\beta x}}$$

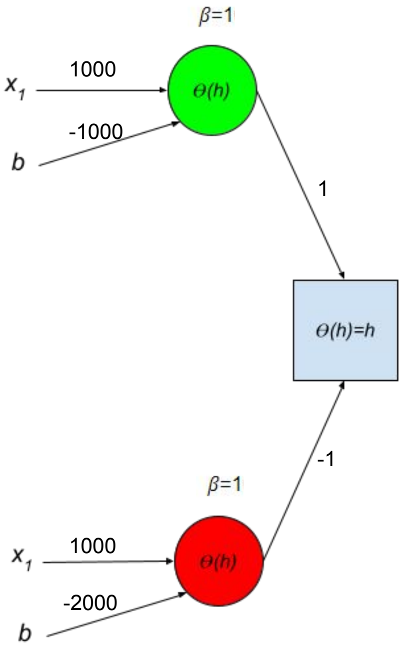
$$Im = (0,\ 1)$$





BUSCANDO ALTERNATIVAS…





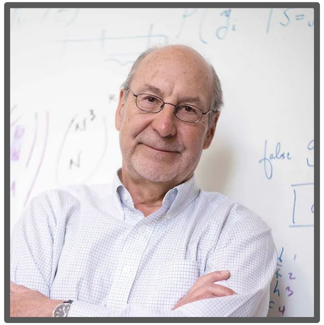
¿CÓMO PUEDO COMBINAR LOS PERCEPTRONES?

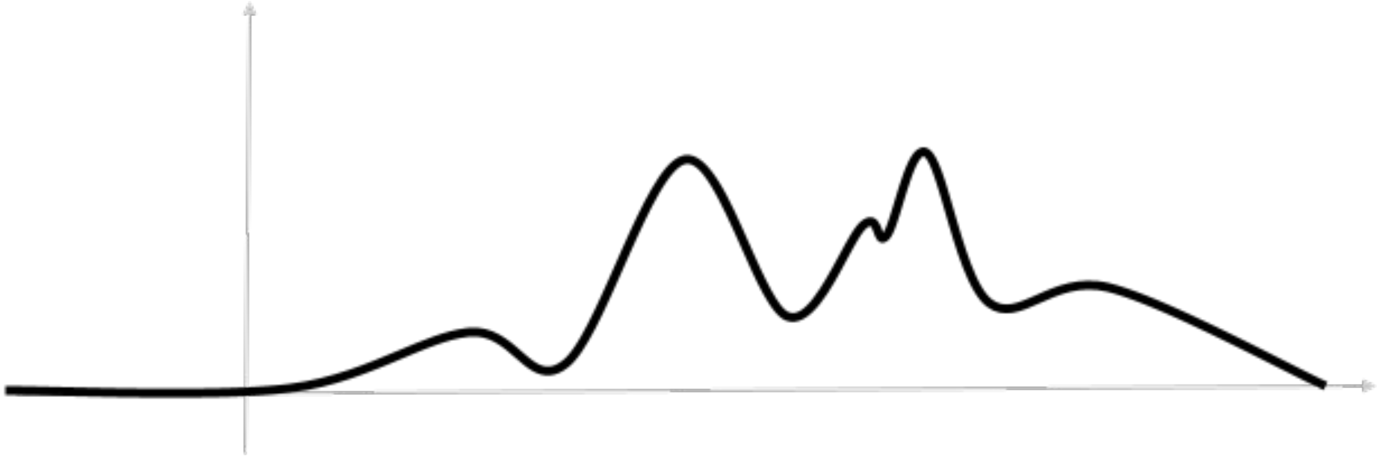
¿Qué clase de problemas puedo representar?

TEOREMA DE APROXIMACIÓN UNIVERSAL

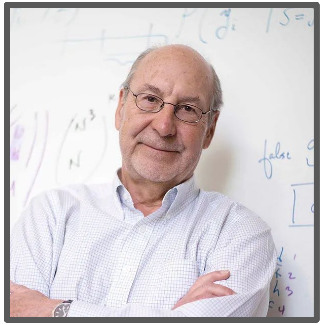
1989 - Cybenko, G.

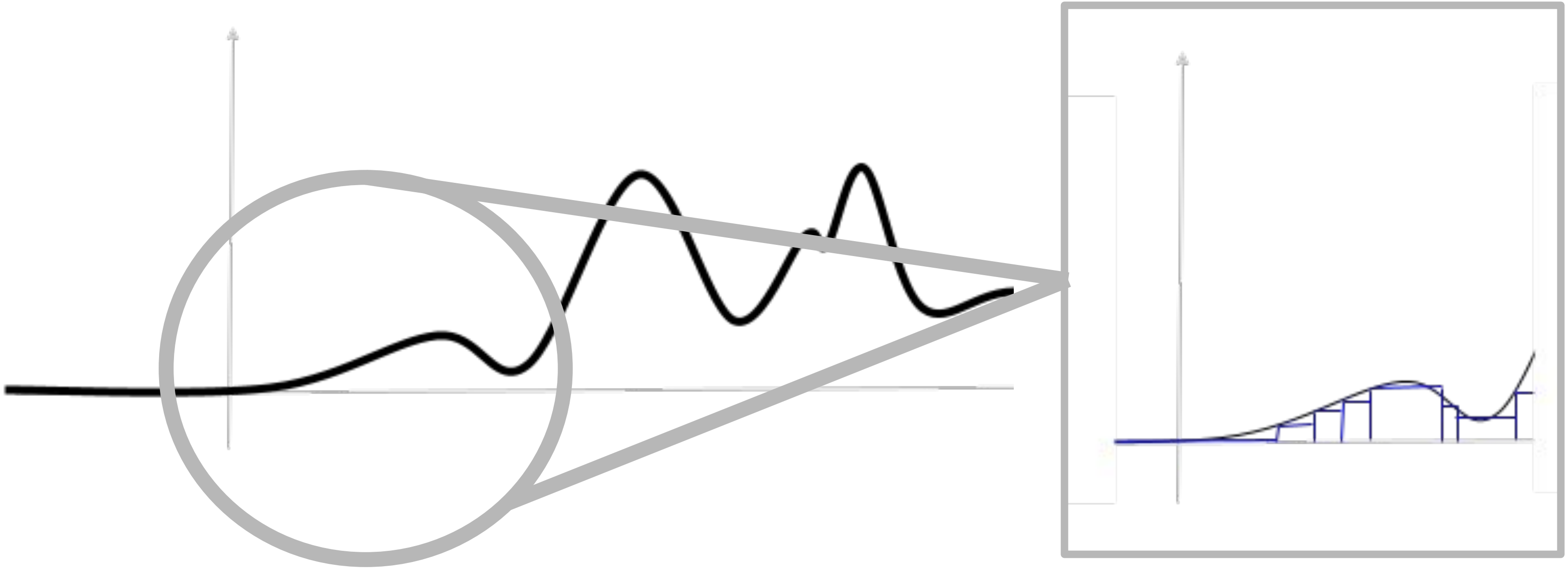
(expansión en 1990, Hornik, K.)



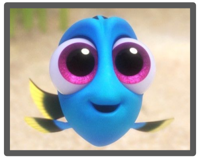


TEOREMA DE APROXIMACIÓN UNIVERSAL
1989 - Cybenko, G.
(expansión en 1990, Hornik, K.)





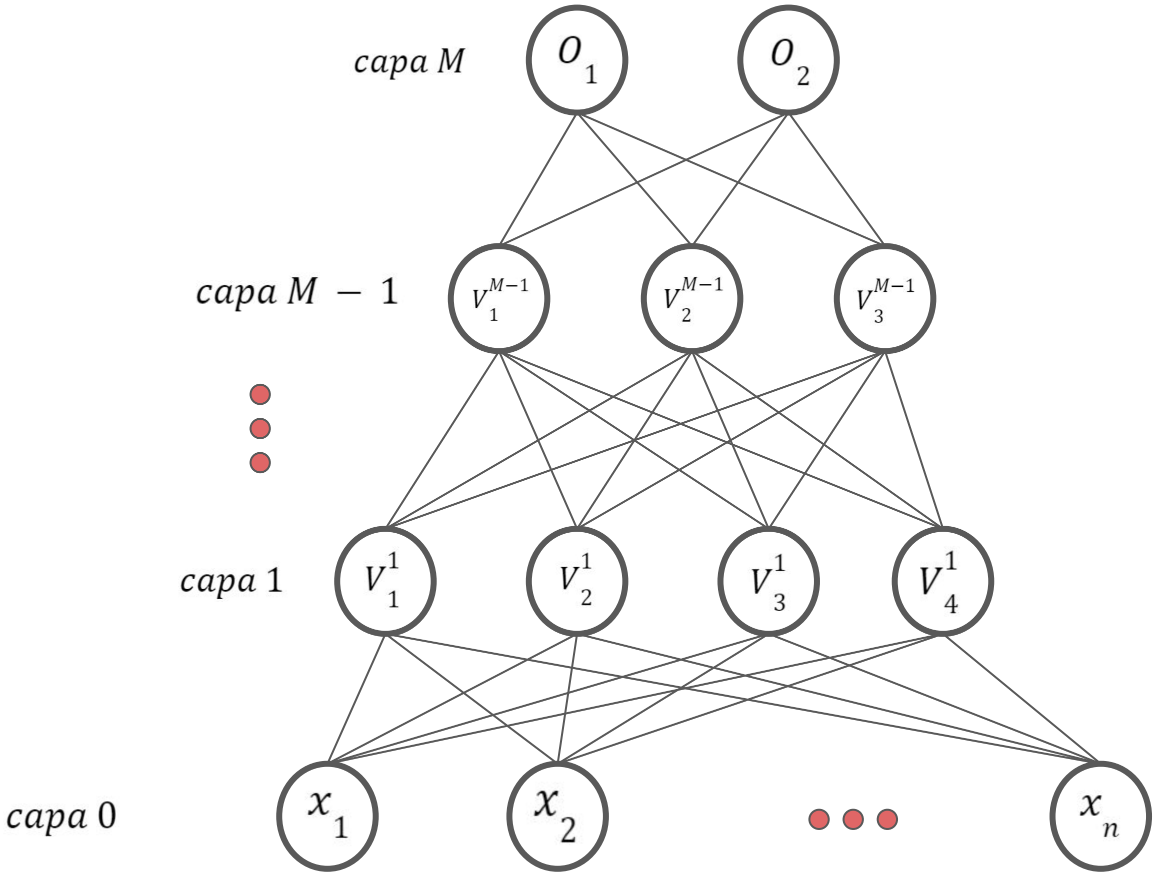
TEOREMA DE APROXIMACIÓN UNIVERSAL
1989 - Cybenko, G.
(expansión en 1991, Hornik, K.)



While the approximating properties we have described are quite powerful, we have focused only on existence. The important questions that remain to be answered deal with feasibility, namely how many terms in the summation (or equivalently, how many neural nodes) are required to yield an approximation of a given quality? What properties of the function being approximated play a role in determining the number of terms? At this point, we can only say that we suspect quite strongly that the overwhelming majority of approximation problems will require astronomical numbers of terms. This feeling is based on the *curse of dimensionality* that plagues multidimensional approximation theory and statistics. Some recent progress con-

$M = cantidad\ capas$

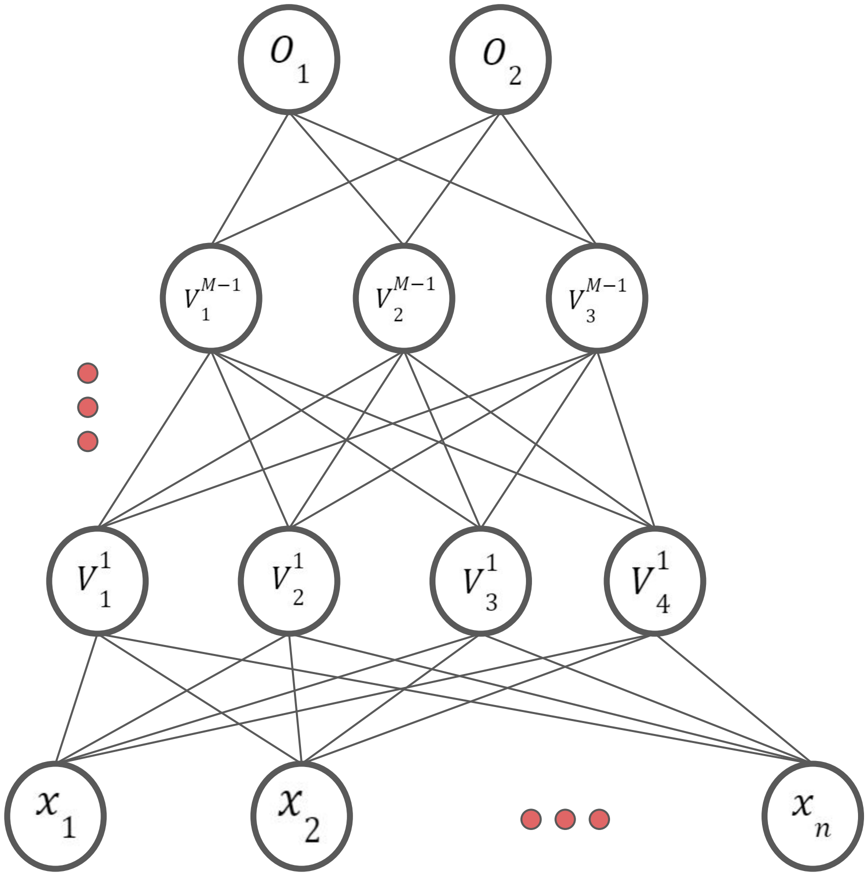
PERCEPTRÓN MULTICAPA



$p = cantidad\ de\ datos$

$n = cantidad\ de\ variables$

¿CÓMO CALCULAMOS LA SALIDA DE LA RED? **FEED-FORWARD PASS**



¿CÓMO CALCULAMOS LA SALIDA DE LA RED? **FEED-FORWARD PASS**

Salida de la neurona:

$$O_i = \theta\left(\sum_{k=1} V_k^{M-1} . W_{ik}\right)$$

Salida de la neurona de una capa intermedia:

$$V_j^m = \theta\left(\sum_{k=1} V_k^{m-1} . w_{jk}^m\right) \qquad m = 2 \ldots M - 1$$

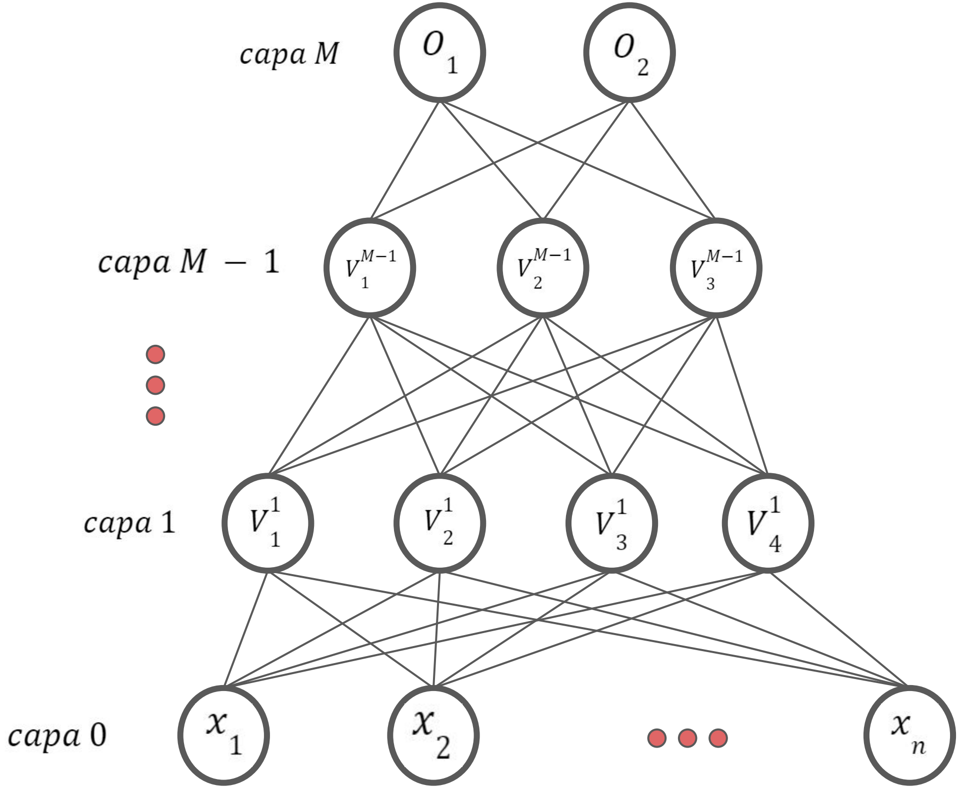
Salida de la neurona de la primera capa intermedia:

$$V_j^1 = \theta\left(\sum_{k=1} x_k^{\mu} . w_{jk}^1\right)$$

¿CÓMO CALCULAMOS LA SALIDA DE LA RED? **FEED-FORWARD PASS**

$$V_j^m = \theta\left(\sum_{k=1} V_k^{m-1} \,.\, w_{jk}^m\right)$$

$$m = 1\ldots M \;\; (O_i = V_i^M \,,\; x_i = V_i^0)$$



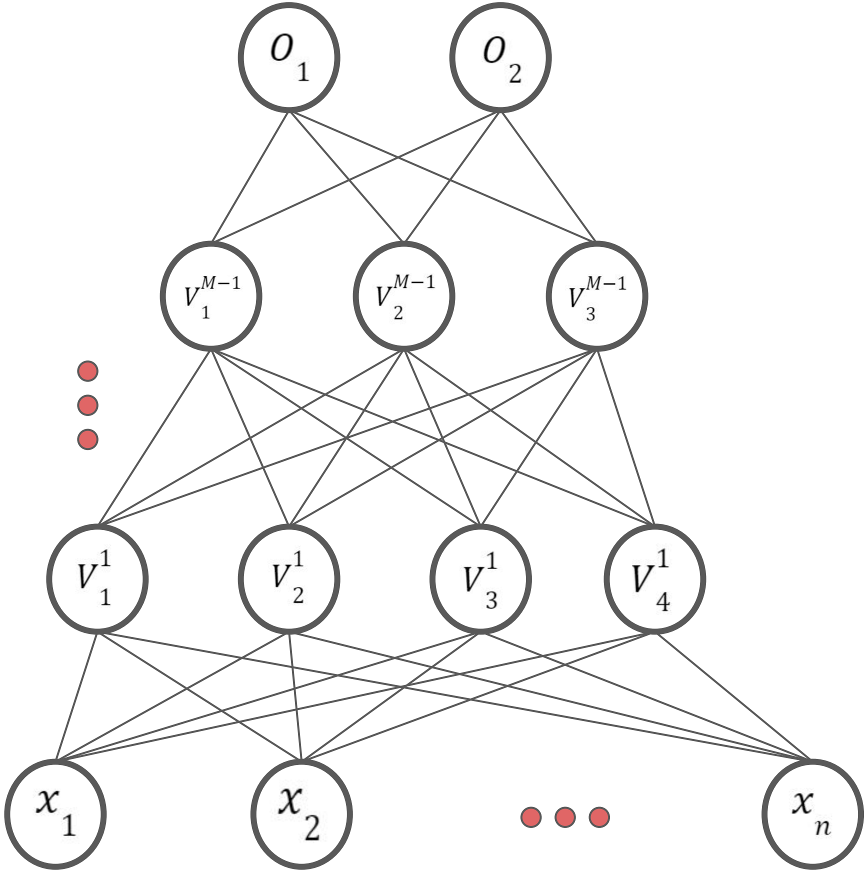
¿CÓMO VA CAMBIANDO EL ALGORITMO?

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate η

for a fixed number of epochs:
    For each training example μ in the dataset:
        1. Compute activation given by feed forward pass:
```

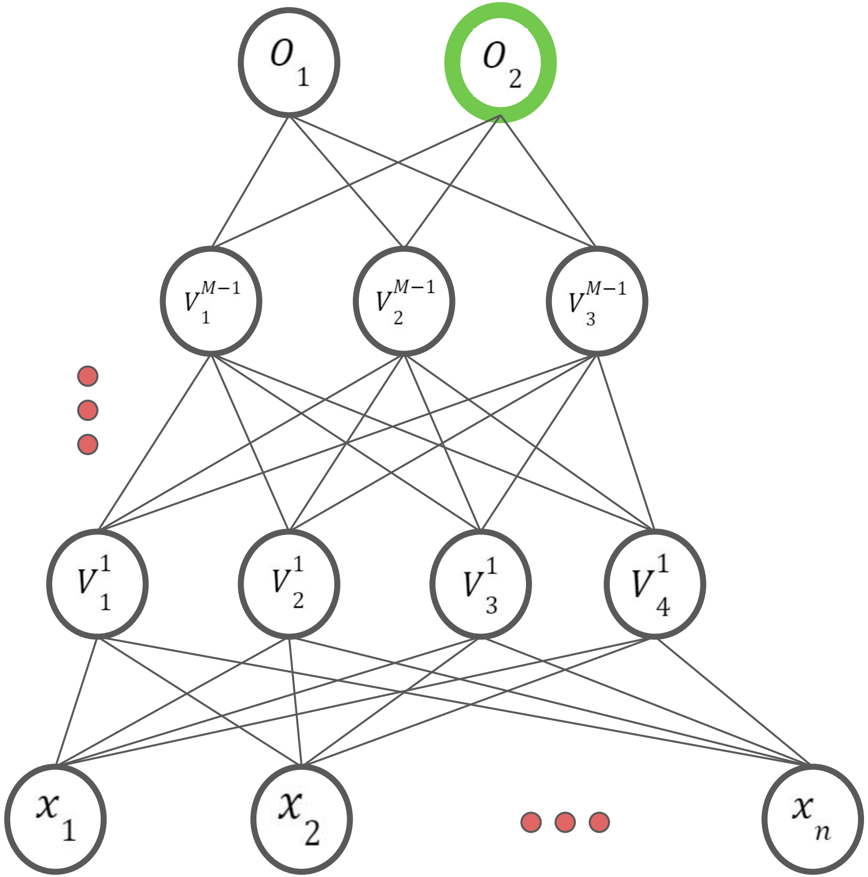
$$O_j = \theta\left(\sum_k V_k^{M-1} \cdot w_{jk}^M\right),$$

```
               where j is the index of output neuron,
               M is the index of output layer,
               k is the index of neuron from previous layer.
               sum is over the qty. of neurons from the previous layer.

        2. Update the weights and bias:
           ???

        3. Calculate perceptron error:
           error = f(xμ1, xμ2, ..., xμn)
           convergence = True if error < ε else False
           if convergence: break

End
```

AHORA… ¿CÓMO LO ENTRENAMOS? 👀



AHORA… ¿CÓMO LO ENTRENAMOS? 👀

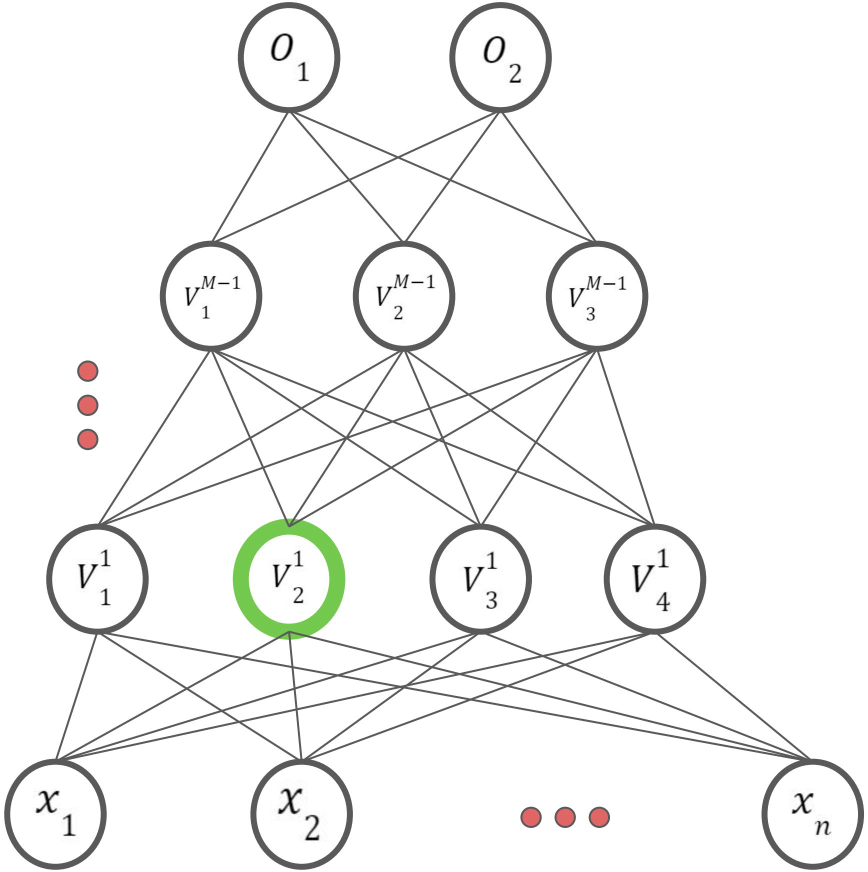


Recordemos…

$$w^{nuevo} = w^{anterior} + \Delta w$$

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

AHORA… ¿CÓMO LO ENTRENAMOS? 👀

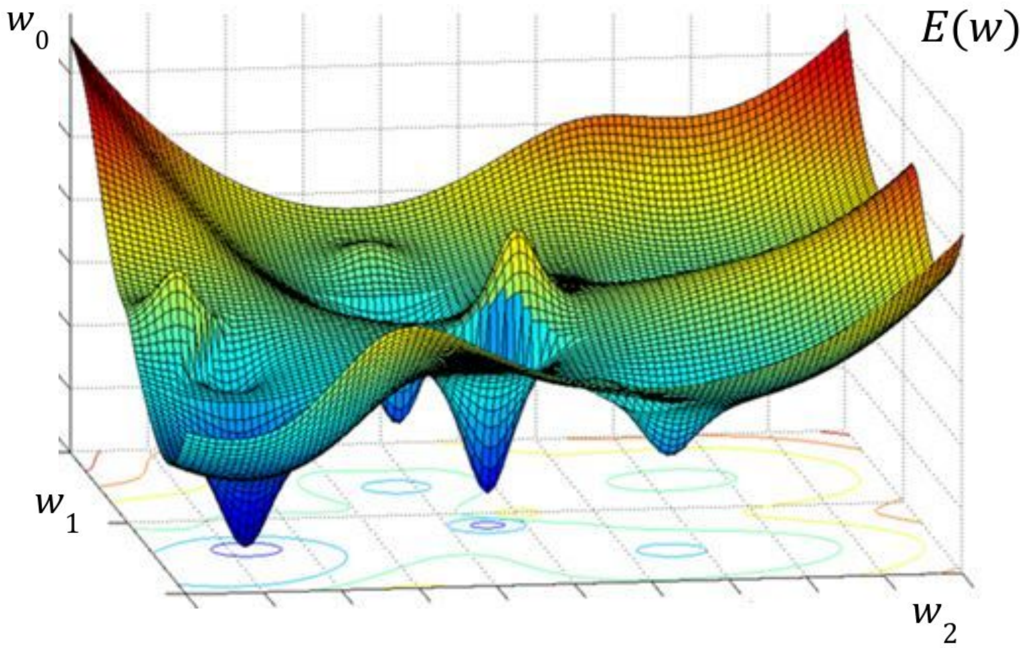


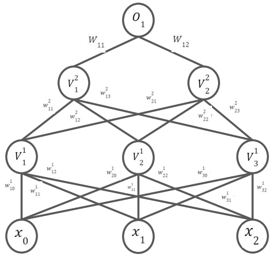
Recordemos…

$$w^{nuevo} = w^{anterior} + \Delta w$$

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL





$$w = \begin{bmatrix} 0.1 \\ -3.4 \\ 2 \\ 0.6 \\ 0.34 \\ -1.2 \\ -0.01 \\ 0.86 \\ -1.9 \\ -2.3 \\ 0.7 \\ 0.02 \\ \dots \end{bmatrix}$$

ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

Mecanismo de optimización: por ahora, gradiente descendente.

$$w^{nuevo} = w^{anterior} + \Delta w$$

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

+

Regla de la cadena de la derivada para capas ocultas

$$\frac{\partial E}{\partial w^2_{11}}$$

$$\frac{\partial E}{\partial w^1_{11}}$$

$$\frac{\partial E}{\partial W_{11}}$$

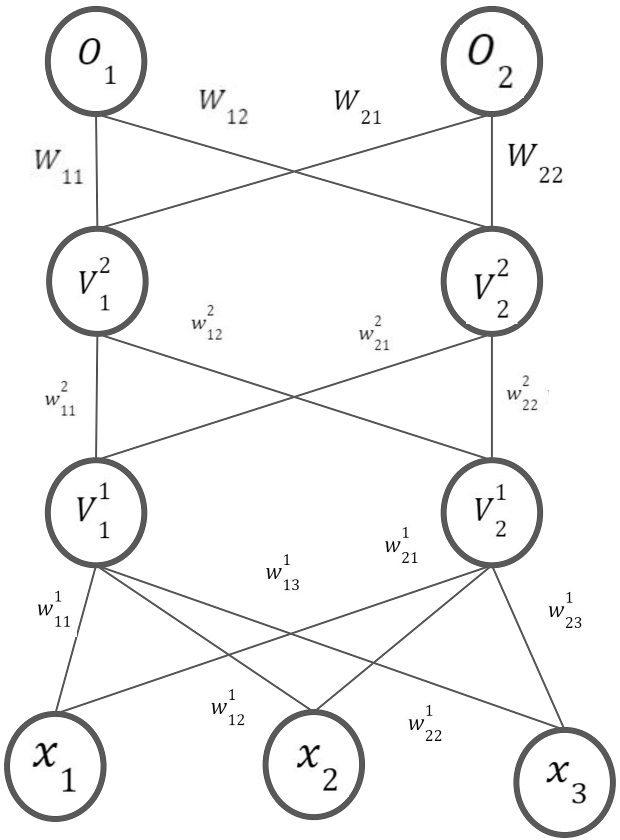
Índice	Descripción
i	índice de la neurona de la capa de salida (o siguiente)
j	índice de la neurona de la capa intermedia
k	índice de la neurona de entrada o de la capa anterior
m	índice de la capa intermedia
p	cantidad datos
μ	dato en particular

$$w^{m}_{jk} = \textit{pesos sinápticos}$$

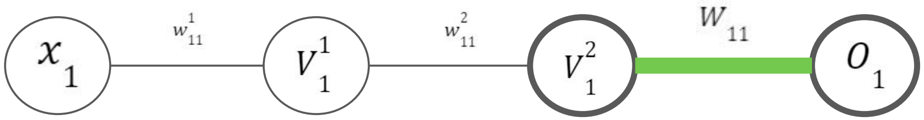
$$V^{m}_{j} = \textit{neurona de capa intermedia}$$

$$W_{ij} = \textit{pesos sináticos de la última capa}$$

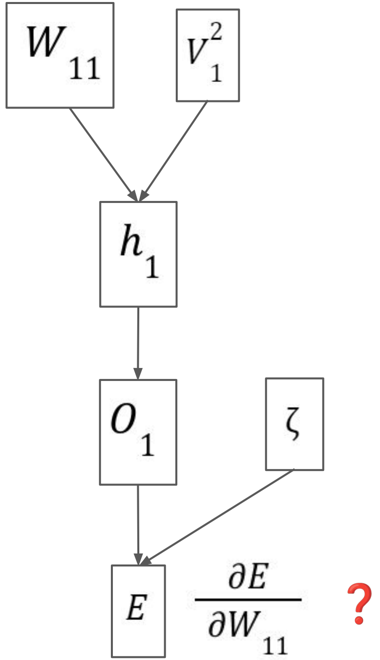
$$O_{i} = \textit{neurona de capa de salida}$$



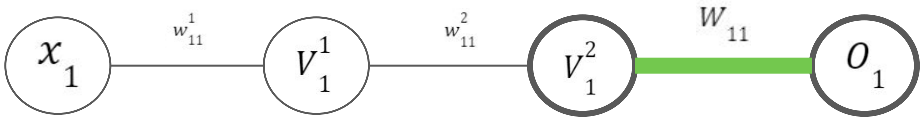
ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL



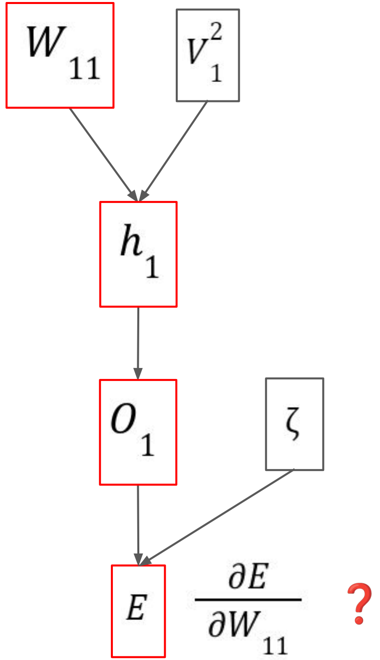
$\zeta = 1.0$



ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL



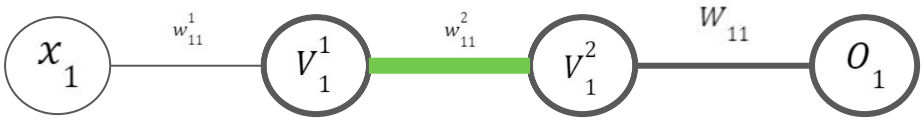
$\zeta = 1.0$



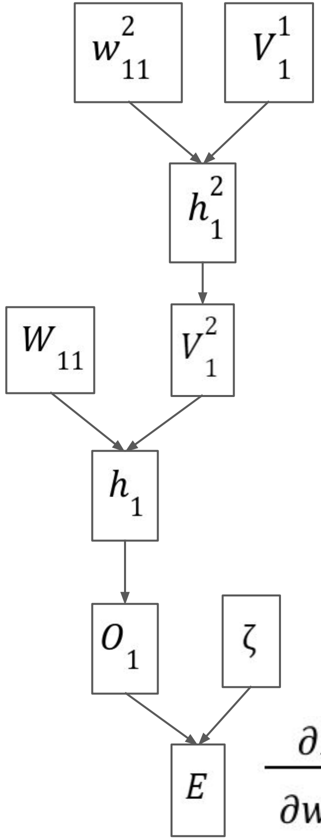
Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial W_{11}} = \frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial h_1} \frac{\partial h_1}{\partial W_{11}}$$

ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

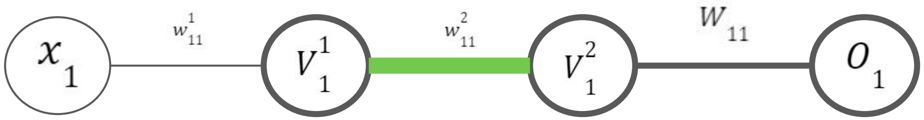


$\zeta = 1.0$

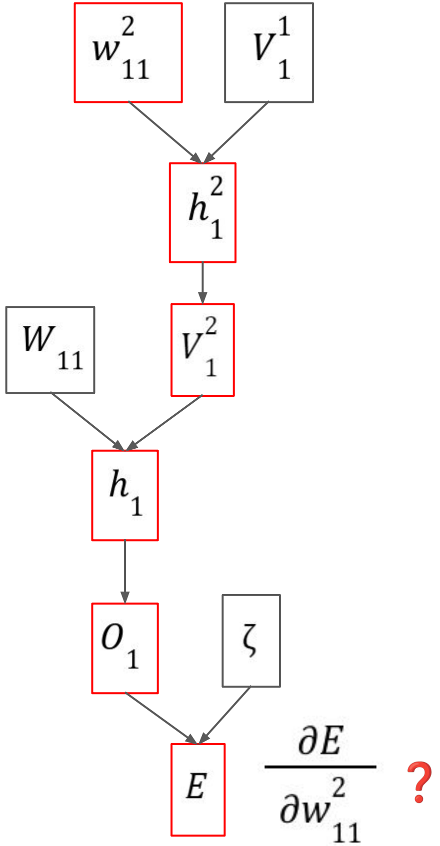


$\frac{\partial E}{\partial w^2_{11}}$?

ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL



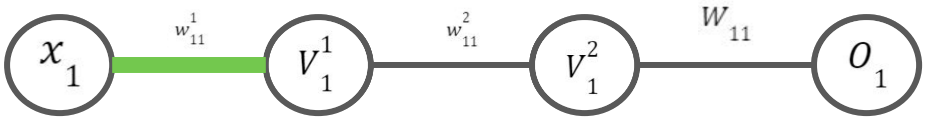
$\zeta = 1.0$



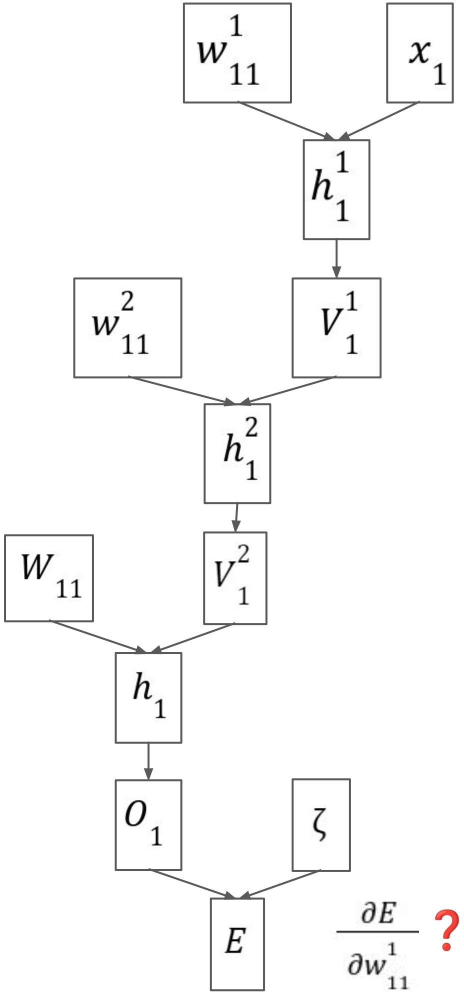
Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial w^2_{11}} = \frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial h_1} \frac{\partial h_1}{\partial V^2_1} \frac{\partial V^2_1}{\partial h^2_1} \frac{\partial h^2_1}{\partial w^2_{11}}$$

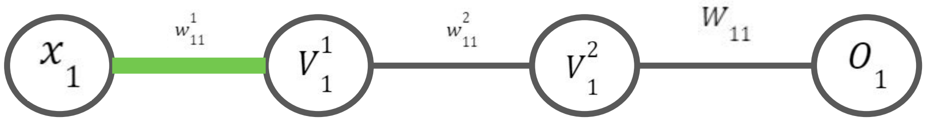
ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL



$\zeta = 1.0$



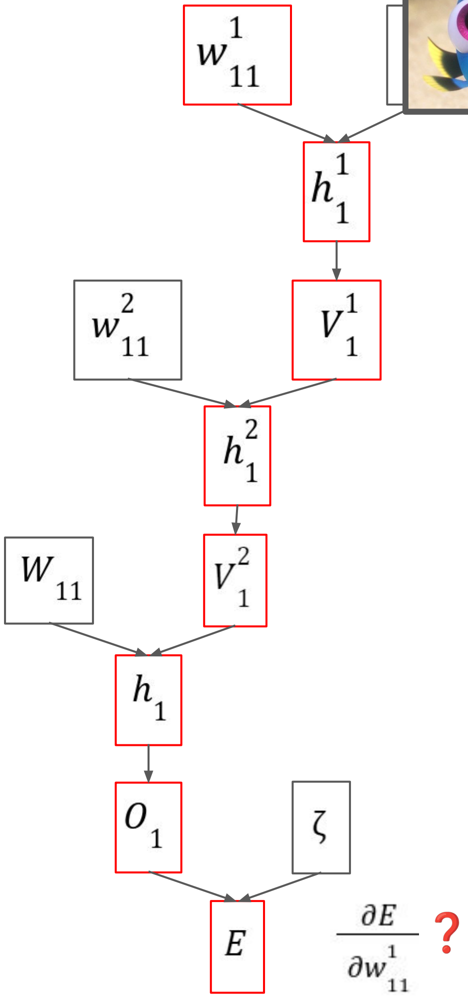
ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

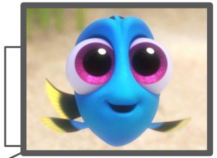


$$\zeta = 1.0$$

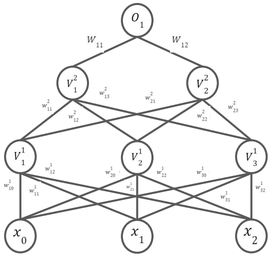
Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial w^1_{11}} = \frac{\partial E}{\partial O_1}\frac{\partial O_1}{\partial h_1}\frac{\partial h_1}{\partial V^2_1}\frac{\partial V^2_1}{\partial h^2_1}\frac{\partial h^2_1}{\partial V^2_1}\frac{\partial V^2_1}{\partial h^1_1}\frac{\partial h^1_1}{\partial w^1_{11}}$$





RUMELHART, HINTON, WILLIAMS (1986)

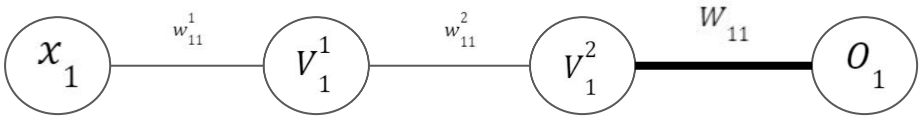


$$w = \begin{bmatrix} 0.1 \\ -3.4 \\ 2 \\ 0.6 \\ 0.34 \\ -1.2 \\ -0.01 \\ 0.86 \\ -1.9 \\ -2.3 \\ 0.7 \\ 0.02 \\ \dots \end{bmatrix}$$

- Trabajan en un nuevo algoritmo para actualizar los pesos de una red neuronal usando **retropropagación** (***backpropagation***)

- Para calcular la actualización de los pesos utilizaremos el algoritmo del gradiente descendente y la regla de la cadena para la diferenciación.

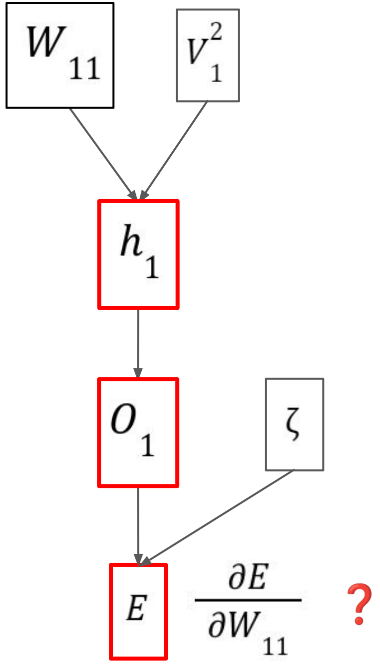
DELTA: EXPLICACIÓN CONCEPTUAL



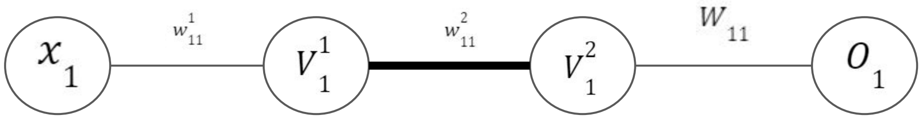
$\zeta = 1.0$

Aplicamos la regla de la cadena:

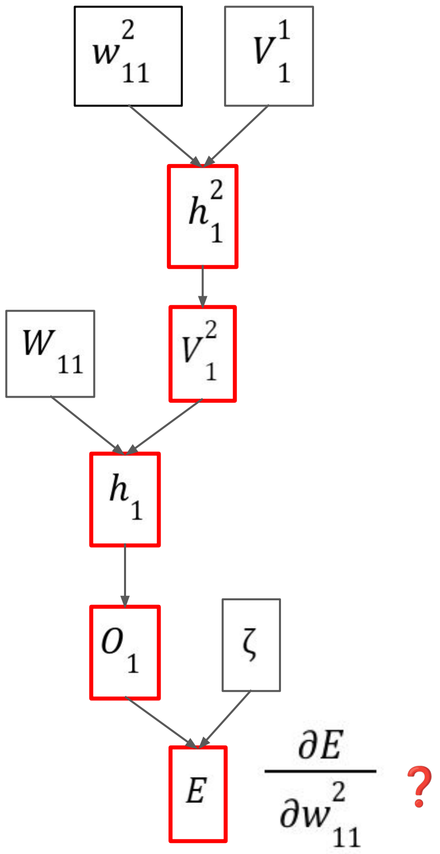
$$\frac{\partial E}{\partial W_{11}} = \boxed{\frac{\partial E}{\partial o_1} \frac{\partial o_1}{\partial h_1}} \frac{\partial h_1}{\partial W_{11}}$$



DELTA: EXPLICACIÓN CONCEPTUAL



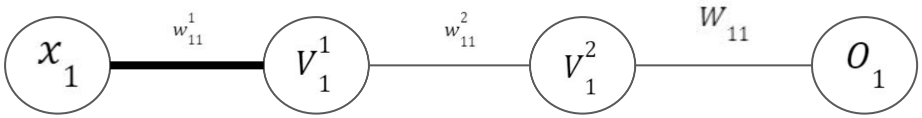
$\zeta = 1.0$



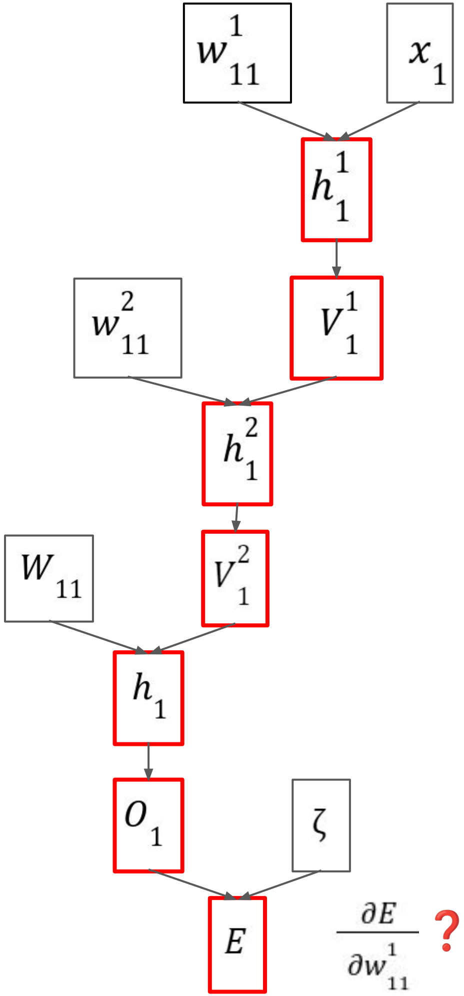
Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial w^2_{11}} = \boxed{\frac{\partial E}{\partial O_1}\,\frac{\partial O_1}{\partial h_1}\,\frac{\partial h_1}{\partial V^2_1}\,\frac{\partial V^2_1}{\partial h^2_1}}\,\frac{\partial h^2_1}{\partial w^2_{11}}$$

DELTA: EXPLICACIÓN CONCEPTUAL



$$\zeta = 1.0$$



Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial w^1_{11}} = \boxed{\frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial h_1} \frac{\partial h_1}{\partial V^2_1} \frac{\partial V^2_1}{\partial h^2_1} \frac{\partial h^2_1}{\partial V^2_1} \frac{\partial V^2_1}{\partial h^1_1}} \frac{\partial h^1_1}{\partial w^1_{11}}$$

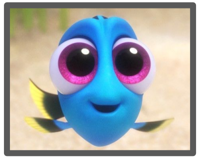
¿CÓMO VA CAMBIANDO EL ALGORITMO?

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate η

for a fixed number of epochs:
    For each training example μ in the dataset:
        1. Compute activation given by feed forward pass:
```

$$O_j = \theta\left(\sum_k V_k^{M-1} \cdot w_{jk}^{M}\right),$$

```
               where j is the index of output neuron,
               M is the index of output layer,
               k is the index of neuron from previous layer.
               sum is over the qty. of neurons from the previous layer.

        2. Update the weights and bias:
           optimizer (GD) with chain rule
           for inner layers (calculated using back-propagation)

        3. Calculate perceptron error:
           error = f(xμ1, xμ2, ..., xμn)
           convergence = True if error < ε else False
           if convergence: break

End
```

¿POR QUÉ FUE TAN IMPORTANTE?



- Permite resolver el problema de determinar cuánto contribuye cada neurona al error del perceptrón multicapa de manera **eficiente**.

- Herramienta práctica para aportar evidencia empírica al Teorema de Aproximación Universal

- Renueva el interés en el área de inteligencia artificial

- El concepto de entrenar redes “deep” se hace posible con back propagation

	PERCEPTRÓN SIMPLE	PERCEPTRÓN MULTICAPA
Proceso para obtener la salida de la neurona ("predecir")	$O = \theta(\sum_{i=0}^{n} x_i . w_i)$	?
Función de costo (medir error con respecto a la salida esperada)	$E(O) = \frac{1}{2}\sum_{\mu=0}^{p-1} (\zeta^{\mu} - O^{\mu})^2$	?
Proceso de "aprendizaje" para ajustar los pesos	$w^{nuevo} = w^{anterior} + \Delta w$ $\Delta w = -\eta \frac{\partial E}{\partial w}$	?

	PERCEPTRÓN SIMPLE	PERCEPTRÓN MULTICAPA
Proceso para obtener la salida de la neurona (“predecir”)	$O = \theta(\sum_{i=0}^{n} x_i . w_i)$	$V_j^m = \theta(\sum_{k=1} V_k^{m-1} . w_{jk}^m)$ $m = 1 \dots M \; (O_i = V_i^M ,\; x_i = V_i^0)$
Función de costo (medir error con respecto a la salida esperada)	$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^\mu - O^\mu)^2$	?
Proceso de “aprendizaje” para ajustar los pesos	$w^{nuevo} = w^{anterior} + \Delta w$ $\Delta w = - \eta \frac{\partial E}{\partial w}$	?

	PERCEPTRÓN SIMPLE	PERCEPTRÓN MULTICAPA
Proceso para obtener la salida de la neurona ("predecir")	$O = \theta(\sum_{i=0}^{n} x_i . w_i)$	$V_j^m = \theta(\sum_{k=1} V_k^{m-1} . w_{jk}^m)$ $m = 1 \ldots M \ (O_i = V_i^M,\ x_i = V_i^0)$
Función de costo (medir error con respecto a la salida esperada)	$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^\mu - O^\mu)^2$	$E(O) = \frac{1}{2} \sum_{\mu} \sum_{i} (\zeta_i^\mu - O_i^\mu)^2$
Proceso de "aprendizaje" para ajustar los pesos	$w^{nuevo} = w^{anterior} + \Delta w$ $\Delta w = - \eta \frac{\partial E}{\partial w}$	?

RETROPROPAGACIÓN: **NOTACIÓN**

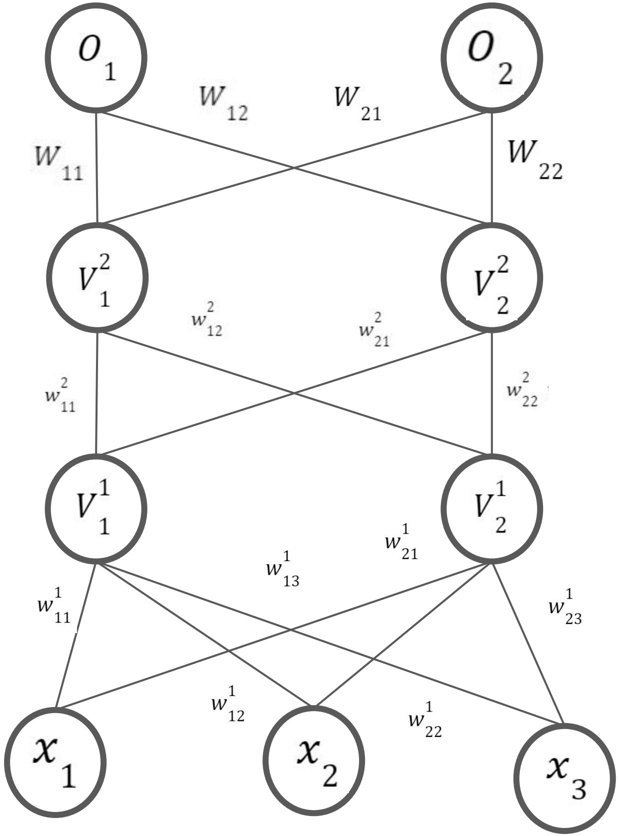
Índice	Descripción
i	índice de la neurona de la capa de salida (o siguiente)
j	índice de la neurona de la capa intermedia
k	índice de la neurona de entrada o de la capa anterior
m	índice de la capa intermedia
p	cantidad datos
μ	dato en particular

$$w^{m}_{jk} = pesos\ sinápticos$$

$$V^{m}_{j} = neurona\ de\ capa\ intermedia$$

$$W_{ij} = pesos\ sináticos\ de\ la\ última\ capa$$

$$O_{i} = neurona\ de\ capa\ de\ salida$$

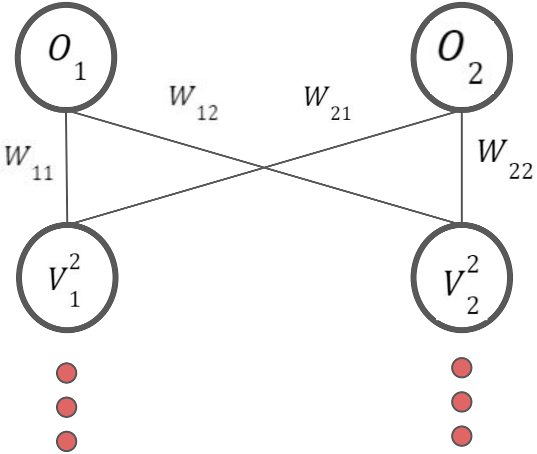


RETROPROPAGACIÓN: CAPA DE SALIDA *(j = M)*

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$

$$O_i = \theta(h_i)$$

$$h_i = \sum_{j=1} V_j^{M-1} . W_{ij}$$



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial O_i}\frac{\partial O_i}{\partial h_i}\frac{\partial h_i}{\partial W_{ij}}$$

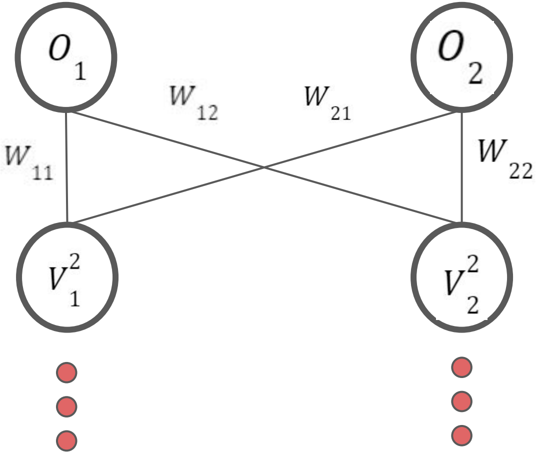
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

RETROPROPAGACIÓN: CAPA DE SALIDA ***(j = M)***

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$

$$O_i = \theta(h_i)$$

$$h_i = \sum_{j=1} V_j^{M-1} . W_{ij}$$



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial W_{ij}} = \boxed{\frac{\partial E}{\partial O_i}}\boxed{\frac{\partial O_i}{\partial h_i}}\boxed{\frac{\partial h_i}{\partial W_{ij}}}$$

$$\frac{\partial E}{\partial W_{ij}} = (\zeta_i - O_i)(-1)\,\theta'(h_i)(1)\,V_j^{M-1}$$

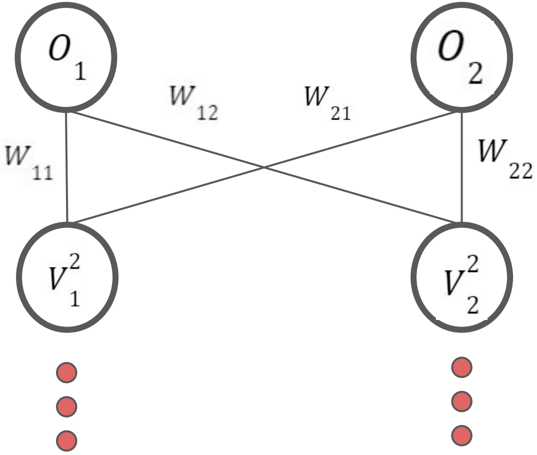
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$

$$O_i = \theta(h_i)$$

$$h_i = \sum_{j=1} V_j^{M-1} . W_{ij}$$

RETROPROPAGACIÓN: CAPA DE SALIDA *(j = M)*



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial W_{ij}} = \boxed{\frac{\partial E}{\partial O_i}}\boxed{\frac{\partial O_i}{\partial h_i}}\boxed{\frac{\partial h_i}{\partial W_{ij}}}$$

$$\frac{\partial E}{\partial W_{ij}} = (\zeta_i - O_i)(-1)\,\theta'(h_i)(1)\,V_j^{M-1}$$

$$\frac{\partial E}{\partial W_{ij}} = -\delta_i V_j^{M-1} \qquad \delta_i = (\zeta_i - O_i)\theta'(h_i)$$

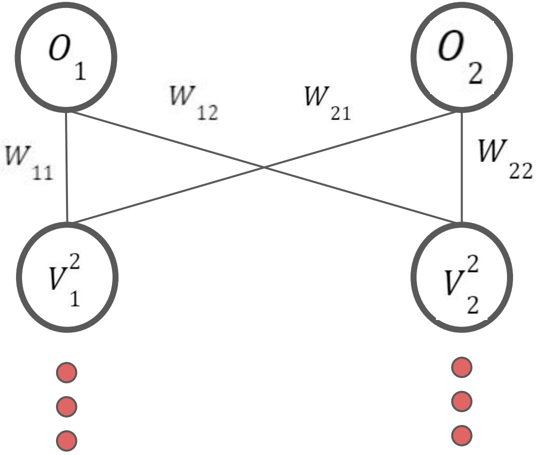
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$

$$O_i = \theta(h_i)$$

$$h_i = \sum_{j=1} V_j^{M-1} . W_{ij}$$

RETROPROPAGACIÓN: CAPA DE SALIDA ***(j = M)***



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial W_{ij}} = \boxed{\frac{\partial E}{\partial O_i}} \boxed{\frac{\partial O_i}{\partial h_i}} \boxed{\frac{\partial h_i}{\partial W_{ij}}}$$

$$\frac{\partial E}{\partial W_{ij}} = (\zeta_i - O_i)(-1)\,\theta'(h_i)(1)\,V_j^{M-1}$$

$$\frac{\partial E}{\partial W_{ij}} = -\delta_i V_j^{M-1} \qquad \delta_i = (\zeta_i - O_i)\theta'(h_i)$$

$$\Delta W_{ij} = \eta \delta_i V_j^{M-1}$$

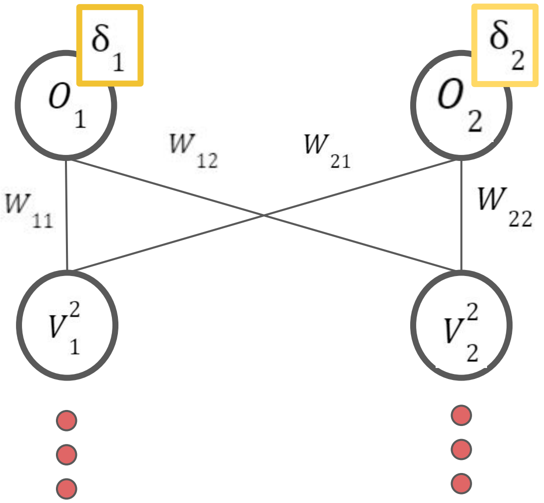
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$

$$O_i = \theta(h_i)$$

$$h_i = \sum_{j=1} V_j^{M-1} . W_{ij}$$

RETROPROPAGACIÓN: CAPA DE SALIDA ***(j = M)***



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial W_{ij}} = \boxed{\frac{\partial E}{\partial O_i}}\boxed{\frac{\partial O_i}{\partial h_i}}\boxed{\frac{\partial h_i}{\partial W_{ij}}}$$

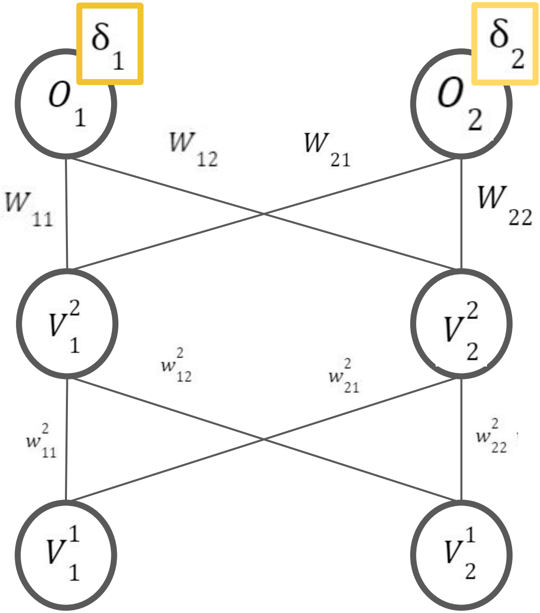
$$\frac{\partial E}{\partial W_{ij}} = (\zeta_i - O_i)(-1)\,\theta'(h_i)(1)\,V_j^{M-1}$$

$$\frac{\partial E}{\partial W_{ij}} = -\delta_i V_j^{M-1} \qquad \delta_i = (\zeta_i - O_i)\theta'(h_i)$$

$$\Delta W_{ij} = \eta \delta_i V_j^{M-1}$$

Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

RETROPROPAGACIÓN: CAPA OCULTA *(j = M-1 , j in [M-1, …, 1])*



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial w^m_{jk}} = \frac{\partial E}{\partial O}\frac{\partial O}{\partial h}\frac{\partial h}{\partial V^m_j}\frac{\partial V^m_j}{\partial h^m_j}\frac{\partial h^m_j}{\partial w^m_{jk}}$$

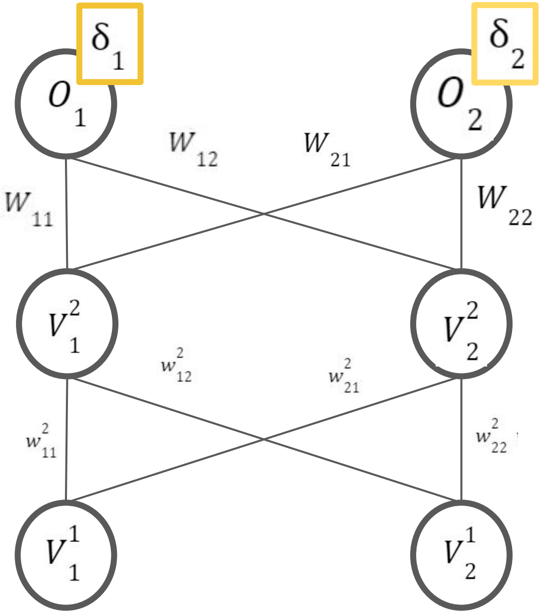
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$

$$O_i = \theta(h_i)$$

$$h_i = \sum_{j=1} V_j^{M-1} . W_{ij}$$

RETROPROPAGACIÓN: CAPA OCULTA ***(j = M-1 , j in [M-1, …, 1])***

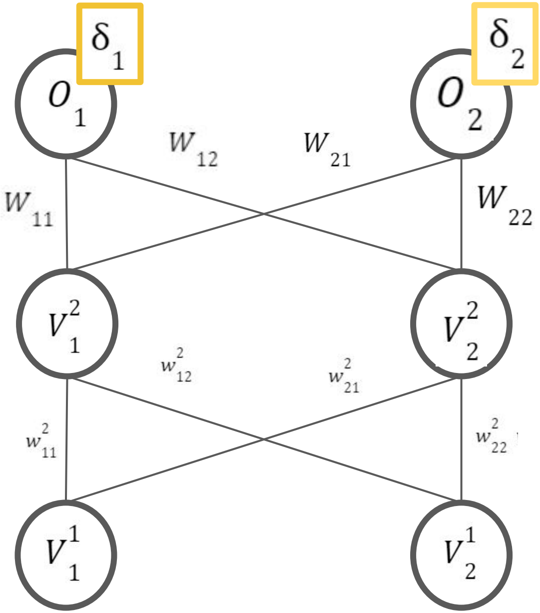


$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial w_{jk}^m} = \boxed{\frac{\partial E}{\partial O}} \boxed{\frac{\partial O}{\partial h}} \boxed{\frac{\partial h}{\partial V_j^m}} \frac{\partial V_j^m}{\partial h_j^m} \frac{\partial h_j^m}{\partial w_{jk}^m}$$

$$\frac{\partial E}{\partial w_{jk}^m} = (-1)\sum_i (\zeta_i - O_i)\,\theta'(h_i)(1)\,W_{ij} \frac{\partial V_j^m}{\partial h_j^m} \frac{\partial h_j^m}{\partial w_{jk}^m}$$

Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

RETROPROPAGACIÓN: CAPA OCULTA ***(j = M-1 , j in [M-1, …, 1])***



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial w^m_{jk}} = \boxed{\frac{\partial E}{\partial O}}\,\boxed{\frac{\partial O}{\partial h}}\,\boxed{\frac{\partial h}{\partial V^m_j}}\,\frac{\partial V^m_j}{\partial h^m_j}\,\frac{\partial h^m_j}{\partial w^m_{jk}}$$

$$\frac{\partial E}{\partial w^m_{jk}} = (-1)\sum_i \underbrace{(\zeta_i - O_i)\,\theta'(h_i)(1)}_{\delta_i}\, W_{ij}\, \frac{\partial V^m_j}{\partial h^m_j}\,\frac{\partial h^m_j}{\partial w^m_{jk}}$$

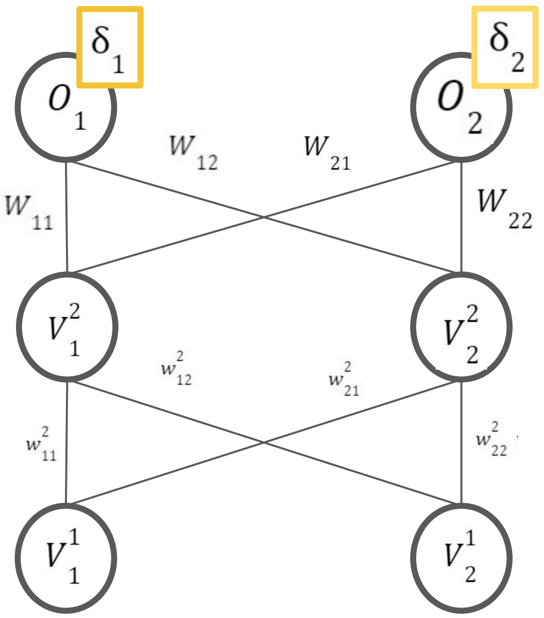
$$\frac{\partial E}{\partial w^m_{jk}} = -\sum_i \delta_i W_{ij}\, \frac{\partial V^m_j}{\partial h^m_j}\,\frac{\partial h^m_j}{\partial w^m_{jk}}$$

Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

$$V_j^m = \theta(h_j^m)$$

$$h_j^m = \sum_{k=1} V_k^{m-1} . w_{jk}^m$$

RETROPROPAGACIÓN: CAPA OCULTA *(j = M-1 , j in [M-1, …, 1])*



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial w_{jk}^m} = \frac{\partial E}{\partial O}\frac{\partial O}{\partial h}\frac{\partial h}{\partial V_j^m}\frac{\partial V_j^m}{\partial h_j^m}\frac{\partial h_j^m}{\partial w_{jk}^m}$$

$$\frac{\partial E}{\partial w_{jk}^m} = -\sum_i \delta_i W_{ij} \boxed{\frac{\partial V_j^m}{\partial h_j^m}} \boxed{\frac{\partial h_j^m}{\partial w_{jk}^m}}$$

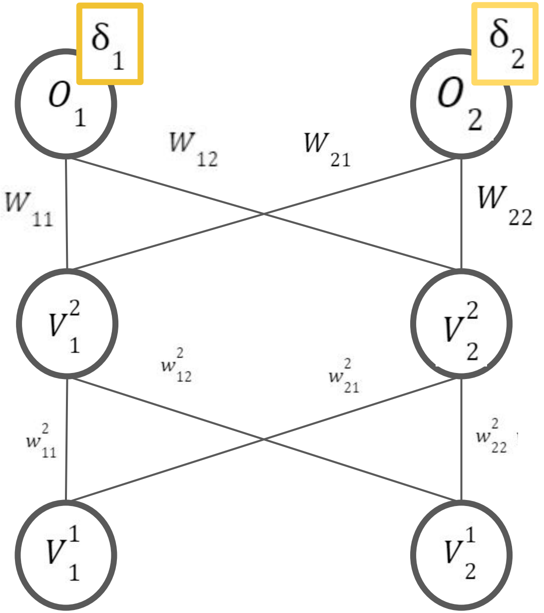
$$\frac{\partial E}{\partial w_{jk}^m} = -\sum_i \delta_i W_{ij} \; \theta'(h_j^m)(1) \; V_k^{m-1}$$

Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

RETROPROPAGACIÓN: CAPA OCULTA ***(j = M-1 , j in [M-1, …, 1])***

$$V_j^m = \theta(h_j^m)$$

$$h_j^m = \sum_{k=1} V_k^{m-1} . w_{jk}^m$$



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial w_{jk}^m} = \frac{\partial E}{\partial O} \frac{\partial O}{\partial h} \frac{\partial h}{\partial V_j^m} \frac{\partial V_j^m}{\partial h_j^m} \frac{\partial h_j^m}{\partial w_{jk}^m}$$

$$\frac{\partial E}{\partial w_{jk}^m} = -\sum_i \delta_i W_{ij} \boxed{\frac{\partial V_j^m}{\partial h_j^m}} \boxed{\frac{\partial h_j^m}{\partial w_{jk}^m}}$$

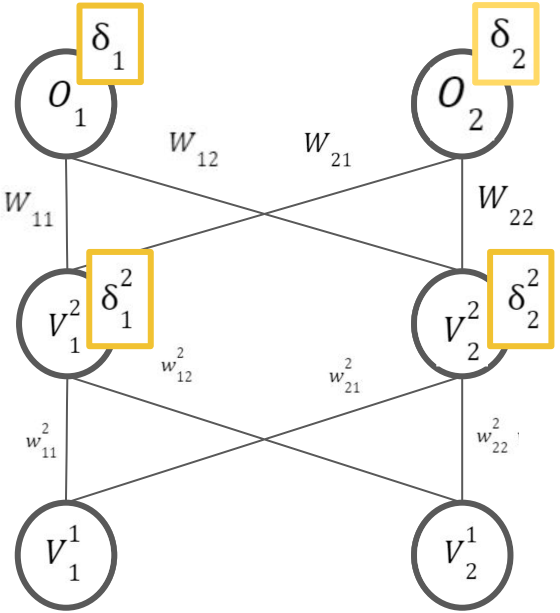
$$\frac{\partial E}{\partial w_{jk}^m} = \underbrace{-\sum_i \delta_i W_{ij} \, \theta'(h_j^m)(1)}_{\delta_j^m} \, V_k^{m-1}$$

$$\Delta w = \eta \delta_j^m V_k^{m-1}$$

$$V_j^m = \theta(h_j^m)$$

$$h_j^m = \sum_{k=1} V_k^{m-1} \,.\, w_{jk}^m$$

RETROPROPAGACIÓN: CAPA OCULTA ***(j = M-1 , j in [M-1, …, 1])***



$$\Delta w = -\eta \boxed{\frac{\partial E}{\partial w}} \longrightarrow \frac{\partial E}{\partial w_{jk}^m} = \frac{\partial E}{\partial O} \frac{\partial O}{\partial h} \frac{\partial h}{\partial V_j^m} \frac{\partial V_j^m}{\partial h_j^m} \frac{\partial h_j^m}{\partial w_{jk}^m}$$

$$\frac{\partial E}{\partial w_{jk}^m} = -\sum_i \delta_i W_{ij} \boxed{\frac{\partial V_j^m}{\partial h_j^m}} \boxed{\frac{\partial h_j^m}{\partial w_{jk}^m}}$$

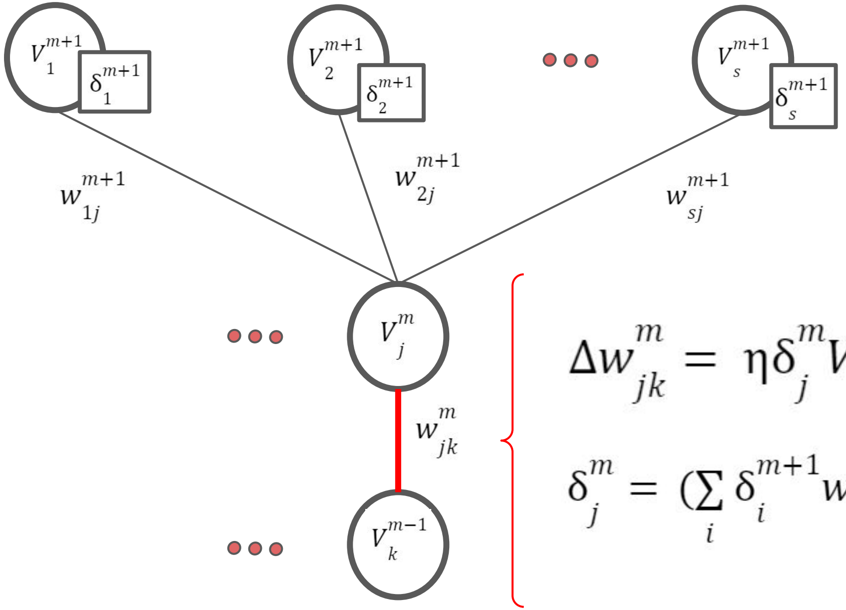
$$\frac{\partial E}{\partial w_{jk}^m} = \underbrace{-\sum_i \delta_i W_{ij} \, \theta'(h_j^m)(1)}_{\delta_j^m} \, V_k^{m-1}$$

$$\Delta w = \eta \delta_j^m V_k^{m-1}$$

RETROPROPAGACIÓN: CAPA OCULTA ***(j in [M-1, …, 1])***

$$V_j^m = \theta(h_j^m)$$

$$h_j^m = \sum_{k=1} V_k^{m-1} \, . w_{jk}^m$$



$$\Delta w_{jk}^m = \eta \delta_j^m V_k^{m-1}$$

$$\delta_j^m = \left(\sum_i \delta_i^{m+1} w_{ij}^{m+1}\right)\theta'(h_j^m)$$

Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

	PERCEPTRÓN SIMPLE	PERCEPTRÓN MULTICAPA
Proceso para obtener la salida de la neurona ("predecir")	$O = \theta(\sum_{i=0}^{n} x_i . w_i)$	$V_j^m = \theta(\sum_{k=1} V_k^{m-1} . w_{jk}^m)$ $m = 1 ... M \ (O_i = V_i^M ,\ x_i = V_i^0)$
Función de costo (medir error con respecto a la salida esperada)	$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^\mu - O^\mu)^2$	$E(O) = \frac{1}{2} \sum_{\mu} \sum_{i} (\zeta_i^\mu - O_i^\mu)^2$
Proceso de "aprendizaje" para ajustar los pesos	$w^{nuevo} = w^{anterior} + \Delta w$ $\Delta w = -\eta \frac{\partial E}{\partial w}$	$\Delta W_{ij} = \eta \delta_i V_j$ $\quad \delta_i = (\zeta_i - O_i)\theta'(h_i)$ $\Delta w_{jk}^m = \eta \delta_j^m V_k^{m-1}$ $\quad \delta_j^m = (\sum_i \delta_i^{m+1} w_{ij}^{m+1})\theta'(h_j^m)$

¿CÓMO VA CAMBIANDO EL ALGORITMO?

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate η

for a fixed number of epochs:
    For each training example μ in the dataset:
        1. Compute activation given by feed forward pass:
```

$$O_j = \theta\left(\sum_k V_k^{M-1} \cdot w_{jk}^{M}\right),$$

```
                where j is the index of output neuron,
                M is the index of output layer,
                k is the index of neuron from previous layer.
                sum is over the qty. of neurons from the previous layer.

        2. Update the weights and bias:
           For each weight w_i:
```

$w_i = w_i + \Delta w_i$, where $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$ (gradient descendent)

and $\frac{\partial E}{\partial w_i}$ is computed using backpropagation

```
        3. Calculate perceptron error:
           error = f(x^μ_1, x^μ_2, ..., x^μ_n)
           convergence = True if error < ε else False
           if convergence: break

End
```

TIP: expresar todo de manera matricial utilizando alguna librería como numpy.

INCREMENTAL/ONLINE (Gradiente Descendente Estocástico)	La actualización de los pesos de la red se hace luego de calcular el Δw **para un elemento** del conjunto de datos
MINI LOTE/MINI BATCH (Gradiente Descendente Estocástico)	La actualización de los pesos de la red se hace luego de calcular el Δw **para un subconjunto de elementos** del conjunto de datos
LOTE/BATCH (Gradiente Descendente)	La actualización de los pesos de la red se hace luego de calcular el Δw **para todos los elementos** del conjunto de datos

¿CÓMO VA CAMBIANDO EL ALGORITMO? VERSIÓN **BATCH**

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate η

for a fixed number of epochs:
    Δws = 0
    For each training example μ in the dataset:
        1. Compute activation given by feed forward pass:

           o_j = θ(∑_k V_k^{M-1} . w_jk^M),
               where j is the index of output neuron,
               M is the index of output layer,
               k is the index of neuron from previous layer.
               sum is over the qty. of neurons from the previous layer.

        2. Calculate the weights and bias:
           For each weight w_i:
               Δws = Δws + Δw_i, where Δw_i = -η ∂E/∂w_i (gradient descendent)

               and ∂E/∂w_i is computed using backpropagation
    w = w + Δws
    Calculate perceptron error:
        error = f(x^μ_1, x^μ_2, ..., x^μ_n)
        convergence = True if error < ε else False
        if convergence: break

End
```

$$o_j = \theta\left(\sum_k V_k^{M-1} \cdot w_{jk}^{M}\right), \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

TIP: expresar todo de manera matricial utilizando alguna librería como numpy.

ESTRATEGIAS DE ENTRENAMIENTO: **ONLINE** / **BATCH**

- ¿Cuál es el costo de cada época? (aprendizaje)
- ¿Cuáles son los requerimientos de memoria?
- ¿Cómo manejan el “ruido”? (ejemplo, outliers)
- ¿Cuánto tarda en converger?
- …

EJERCICIO

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate η

for a fixed number of epochs:
    For each training example μ in the dataset:
        1. Compute activation given by feed forward pass:
```

$$O_j = \theta(\sum_k V_k^{M-1} \cdot w_{jk}^M),$$

```
               where j is the index of output neuron,
               M is the index of output layer,
               k is the index of neuron from previous layer.
               sum is over the qty. of neurons from the previous layer.

        2. Update the weights and bias:
           For each weight wi:
```

$w_i = w_i + \Delta w_i$, where $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$ (gradient descendent)

and $\frac{\partial E}{\partial w_i}$ is computed using backpropagation

```
        3. Calculate perceptron error:
           error = f(xμ1, xμ2, ..., xμn)
           convergence = True if error < ε else False
           if convergence: break

End
```

$error = f(x^\mu_1, x^\mu_2, ..., x^\mu_n)$

RESUMEN

- El perceptrón multicapa me permite modelar transformaciones complejas. En teoría, cualquier función continua puede modelarse con un perceptrón multicapa.

- La arquitectura del perceptŕon multicapa debe realizarse manualmente. A priori, no tenemos una receta que nos permita definir “la mejor arquitectura”.

- Retropropagación provee un mecanismo con ciertas optimizaciones para definir el “error” en las capas ocultas y hallar las actualizaciones de los pesos que nos permiten minimizar la función de costo.

¿Qué estoy optimizando cuando quiero usar una red neuronal?

Función objetivo

$$\min_{x \in \Re^n} f(x),\ \ f : \Re^n \rightarrow \Re, \vec{x} = [x_1, x_2, ..., x_n] \tag{3}$$

$$g_i(\vec{x}) <= 0 \tag{4}$$

$$h_j(\vec{x}) = 0 \tag{5}$$

- Optimización Discreta: Programación entera y combinatoria
- Optimización Discreta: Ecuación diofántica
- Optimización Lineal: Programación Lineal
- Optimización No Lineal sin Restricciones Convexa: garantía de extremos globales.
- Optimización No Convexa: Primer orden: GD, SGD, ADAM,
- Optimización No Convexa: Segundo orden: Newton
- Optimización No Convexa: Cero orden: Bayesiana, Powel, Sim Optimistic Optimization

BIBLIOGRAFÍA

Cybenko, G (1989) ***Approximation by Superpositions of a Sigmoidal Function***, https://link.springer.com/article/10.1007/BF02551274

Hornik, Kurk (1990) ***Approximation capabilities of multilayer feedforward networks***, https://www.sciencedirect.com/science/article/abs/pii/089360809190009T?via%3Dihub

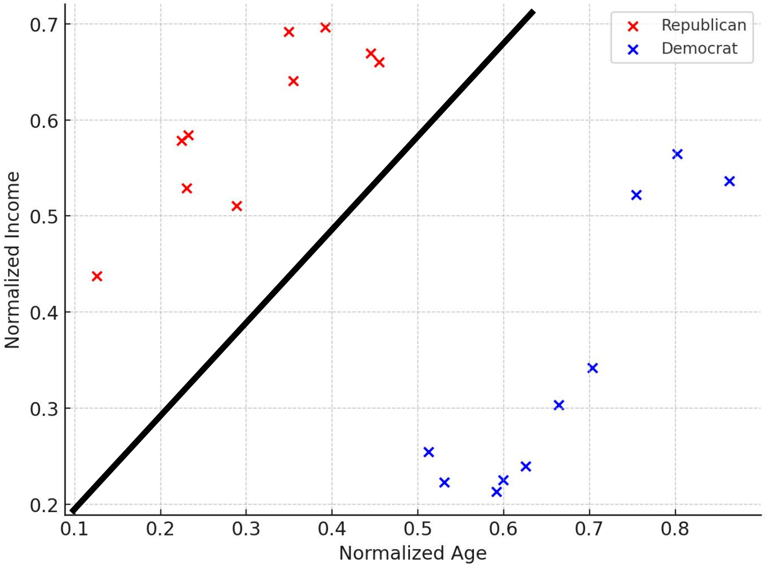
Rumelhart D., Hinton G. & Williams R., ***Learning representations by back-propagating errors***. Nature 323, 533-536 (1986), https://doi.org/10.1038/323533a0

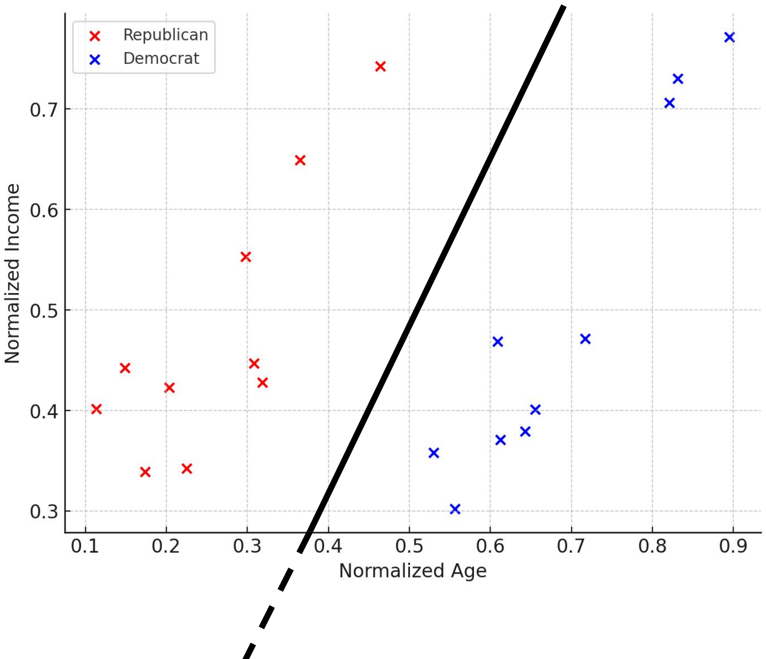
Rodrigo Ramele (2024) ***Reglamento y Apuntes de Sistemas de Inteligencia Artificial***, Capítulo 7.

BIAS EN PERCEPTRÓN MULTICAPA

EXTRA: ¿CÓMO INCLUYO EL BIAS?

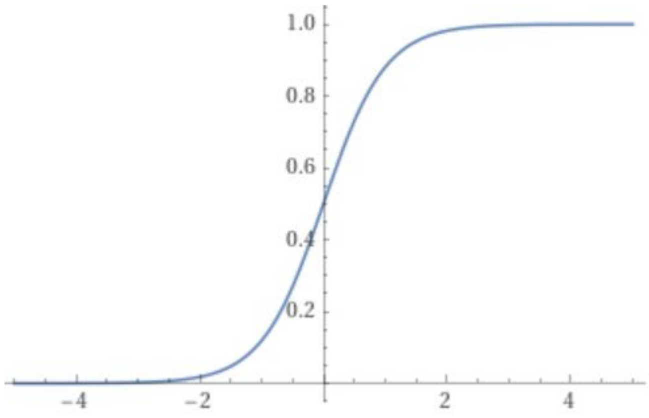
Recordemos que el bias (o umbral) nos permite flexibilizar la forma de la salida de la neurona, porque permite desplazamiento de la función (ver clase 8)



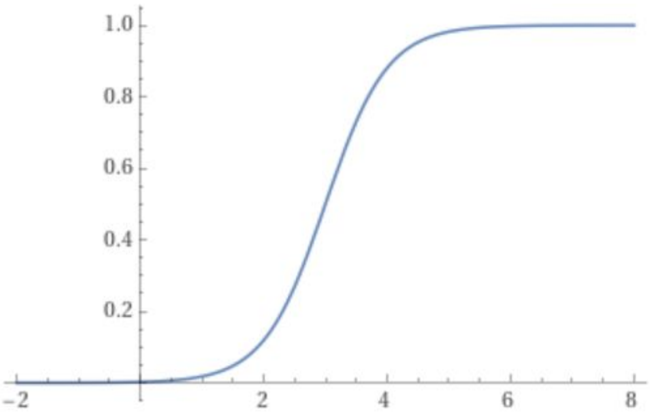


EXTRA: ¿CÓMO INCLUYO EL BIAS?

Recordemos que el bias (o umbral) nos permite flexibilizar la forma de la salida de la neurona, porque permite desplazamiento de la función (ver clase 8)



WolframAlpha Link



WolframAlpha Link

EXTRA: ¿CÓMO INCLUYO EL BIAS?

Podemos incluir un x_0 que permita ajustar un peso w_0. Este peso será equivalente al bias pero puede incorporarse en los cálculos de manera matricial.

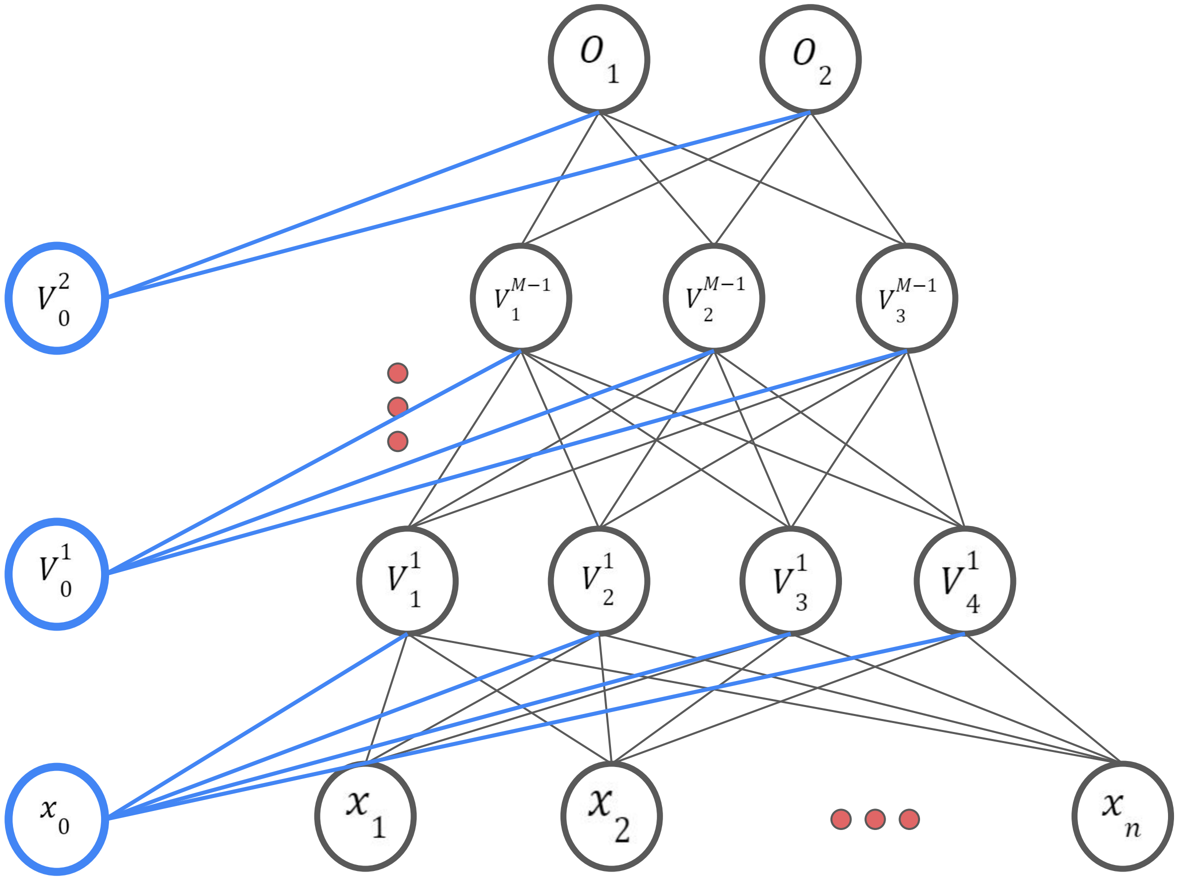
$$O(x) = \sum_{i=1}^{n} x_i . w_i + w_0$$

$$\Downarrow$$

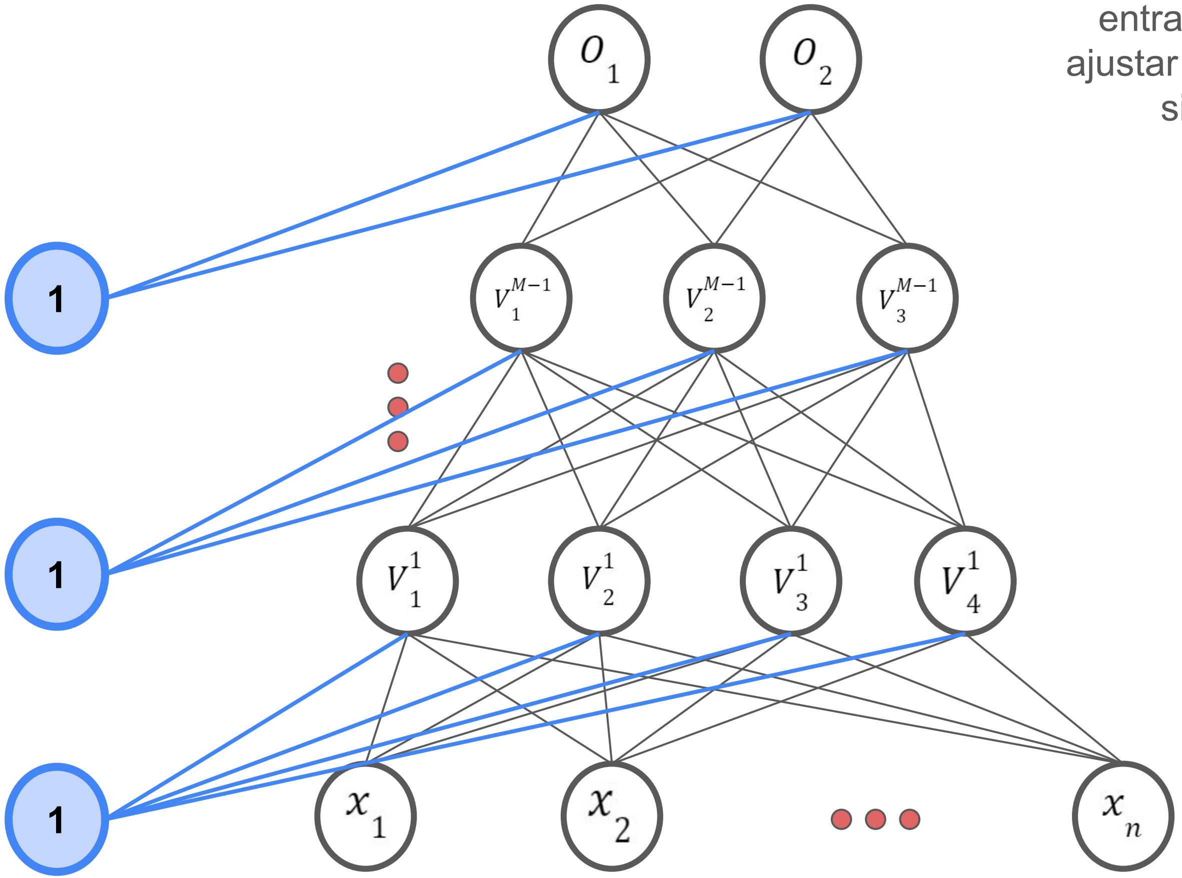
$$\boxed{\theta(\sum_{i=0}^{n} x_i^{\mu} . w_i)}$$

x0	Age	Income	Party
1.0	0.445	0.669	1.0
1.0	0.349	0.692	1.0
1.0	0.232	0.585	1.0
1.0	0.125	0.438	1.0
1.0	0.224	0.579	1.0
1.0	0.23	0.529	1.0
1.0	0.392	0.696	1.0
1.0	0.355	0.641	1.0
1.0	0.455	0.66	1.0
1.0	0.289	0.51	1.0
1.0	0.513	0.255	-1.0
1.0	0.755	0.522	-1.0
1.0	0.626	0.24	-1.0
1.0	0.703	0.342	-1.0
1.0	0.863	0.536	-1.0
1.0	0.6	0.225	-1.0
1.0	0.664	0.303	-1.0
1.0	0.802	0.565	-1.0
1.0	0.592	0.213	-1.0
1.0	0.531	0.223	-1.0

EXTRA: ¿CÓMO INCLUYO EL BIAS?



EXTRA: ¿CÓMO INCLUYO EL BIAS?



Podemos incluir, capa a capa, un valor constante para todos los datos de entrada, de manera tal que podamos ajustar w_0 (bias) como si fuera un peso sináptico más de la red neuronal.

RETROPROPAGACIÓN: **NOTACIÓN**

Índice	Descripción
i	índice de la neurona de la capa de salida (o s
j	índice de la neurona de la capa intermedia
k	**índice de la neurona de entrada o de la capa anterior**
m	índice de la capa intermedia
p	cantidad datos
μ	dato en particular

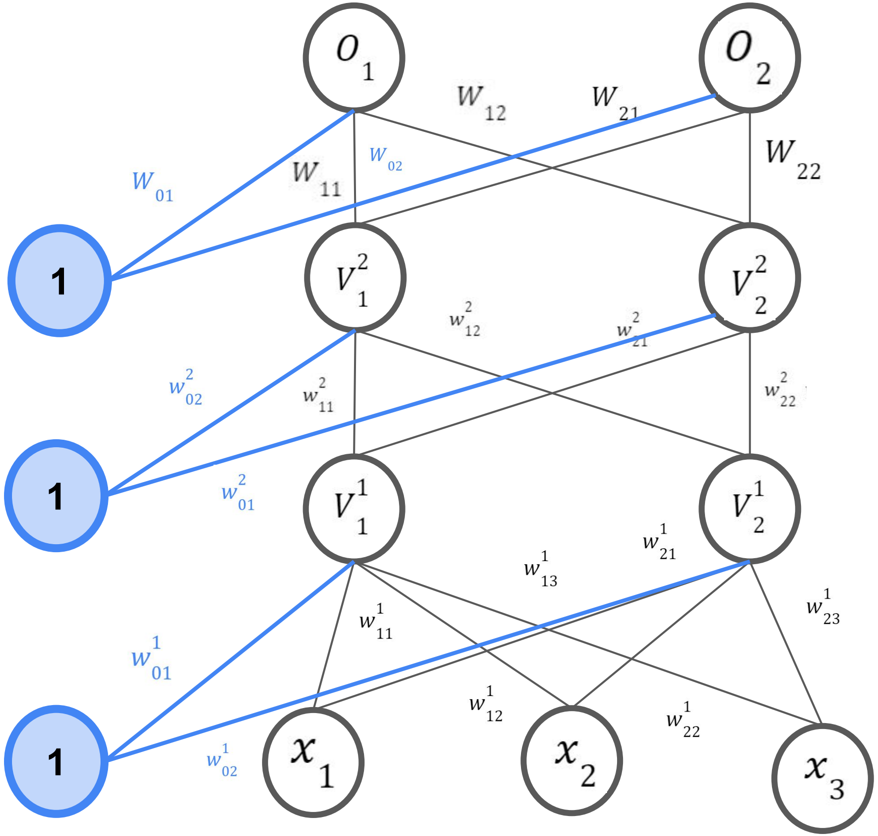
A partir de ahora comienza desde 0

$$w^{m}_{jk} = pesos\ sinápticos$$

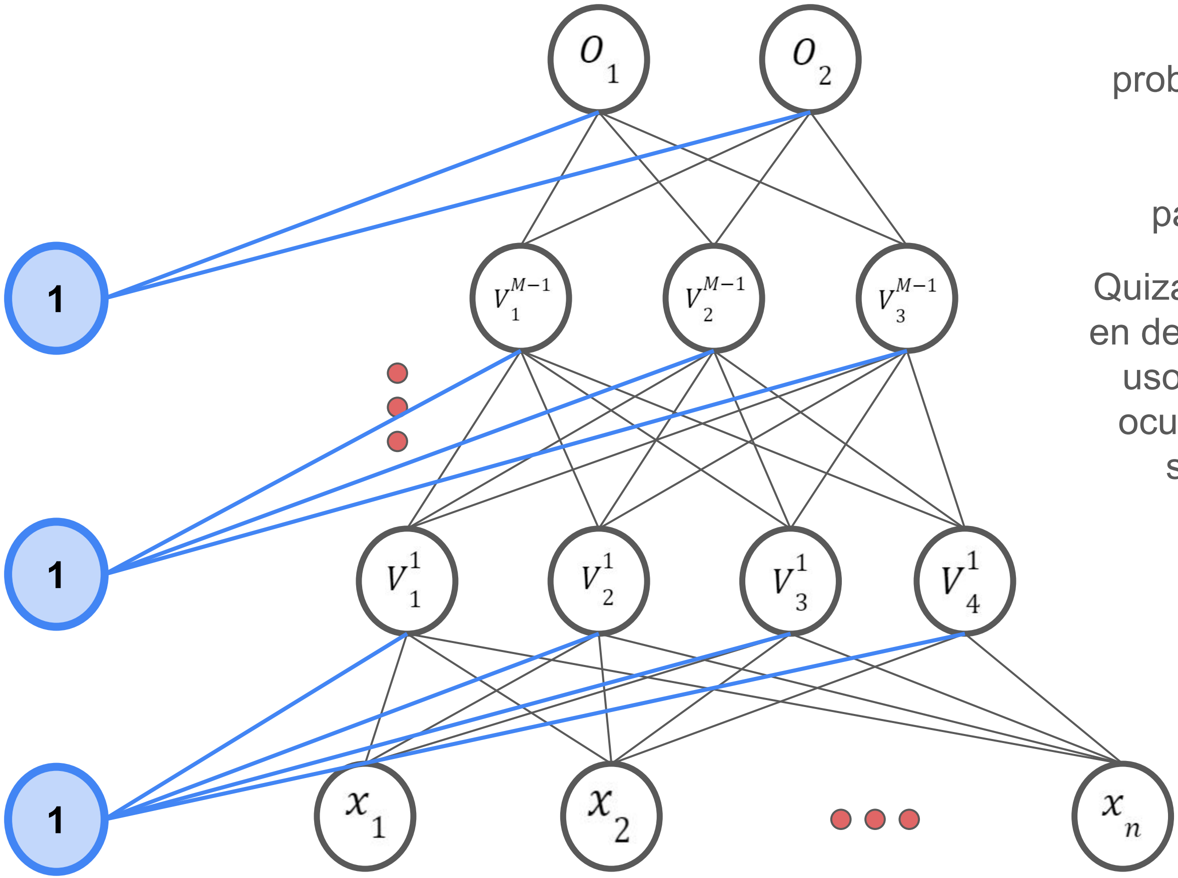
$$V^{m}_{j} = neurona\ de\ capa\ intermedia$$

$$W_{ij} = pesos\ sináticos\ de\ la\ última\ capa$$

$$O_{i} = neurona\ de\ capa\ de\ salida$$



EXTRA: ¿CÓMO INCLUYO EL BIAS?



¿Es estrictamente necesario en input o en todas las capas?

No necesariamente, esto pueden probarlo. Recordar que el rol del bias permite mayor flexibilidad y más parámetros libres para aprender patrones cada vez más complejos.

Quizás es posible alcanzar soluciones en determinadas situaciones sin hacer uso del mismo. Pueden probar ¿qué ocurre si no lo incluyo? ¿si lo incluyo solo en la capa de entrada? ¿si lo incluyo en todas las capas?