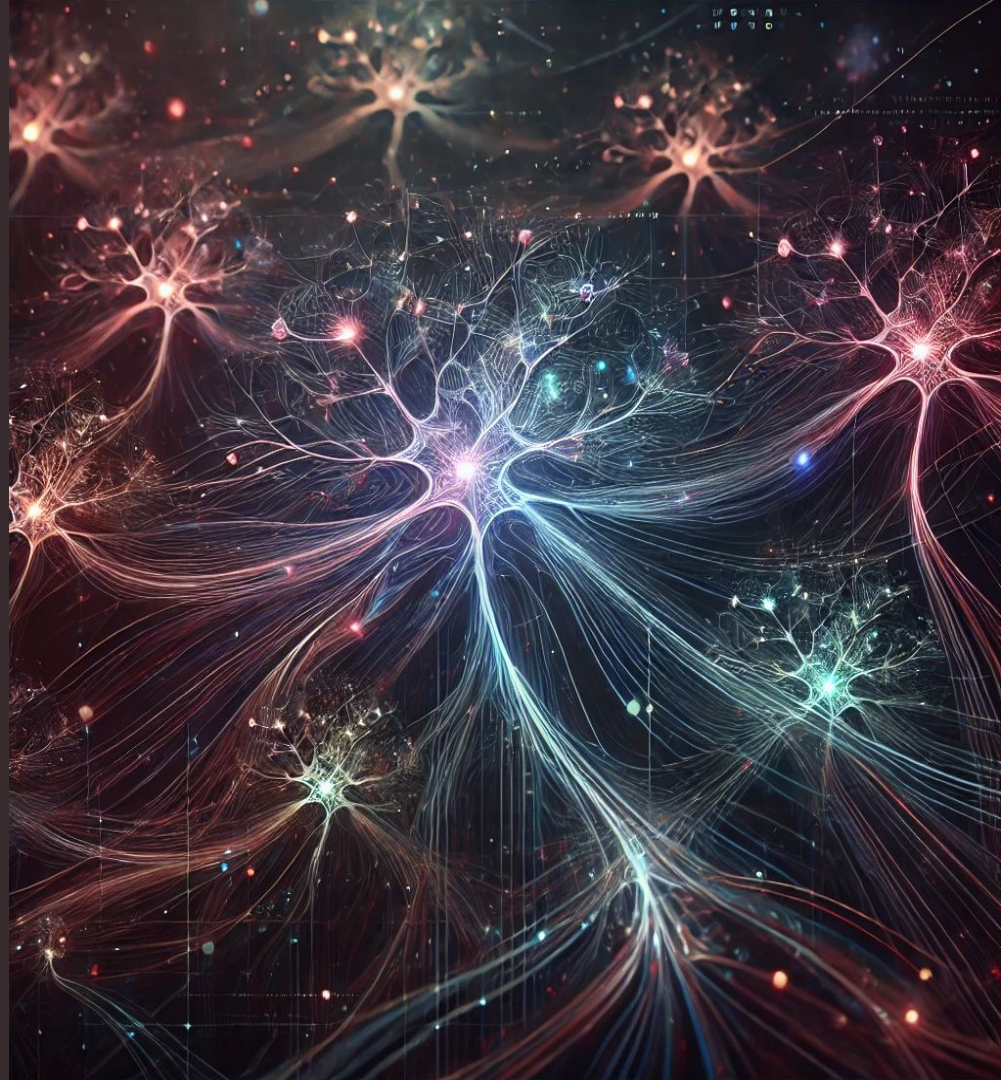


Perceptrón Multicapa

Sistemas de
Inteligencia Artificial

Segundo Cuatrimestre 2025

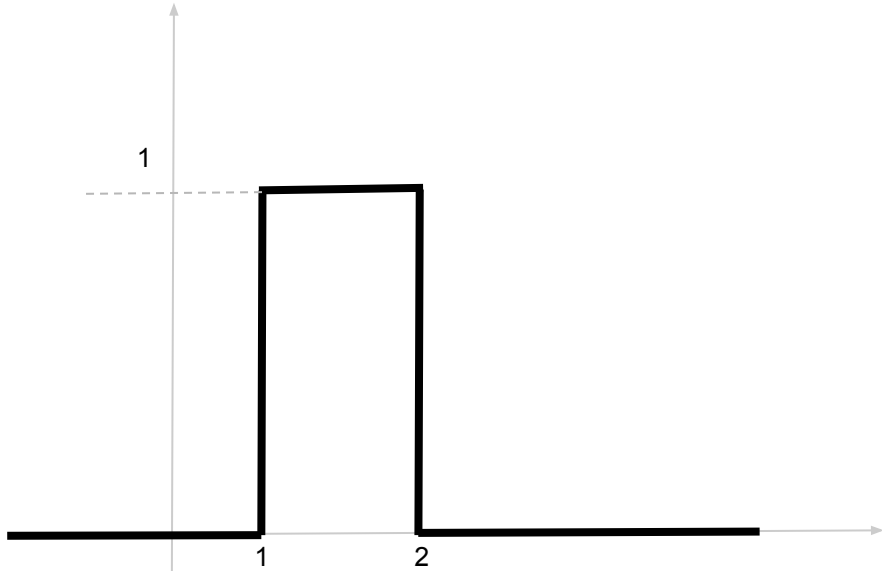
Rodrigo Ramele
Eugenia Piñeiro
Alan Pierri
Santiago Reyes
Marina Fuster
Luciano Bianchi
Marco Scilipoti
Paula Oseroff
Joaquín Girod



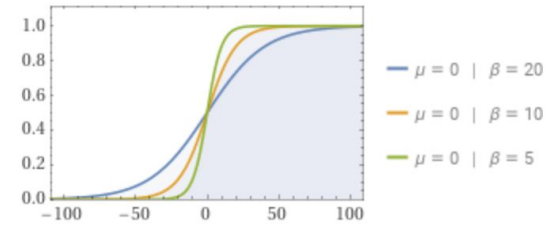
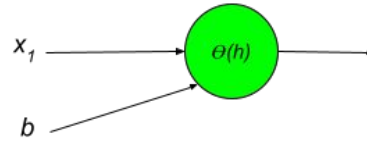
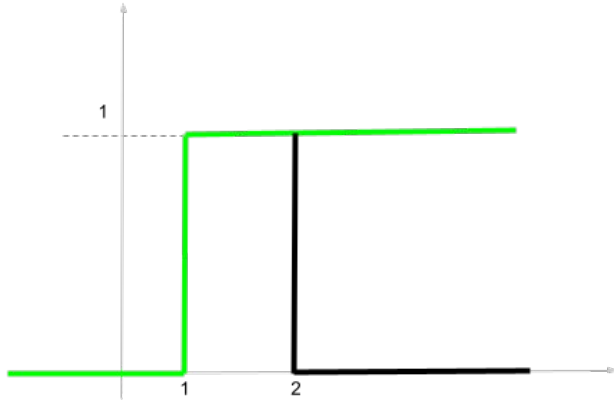
RESUMEN DE LA CLASE ANTERIOR

- ¿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

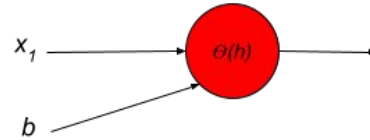
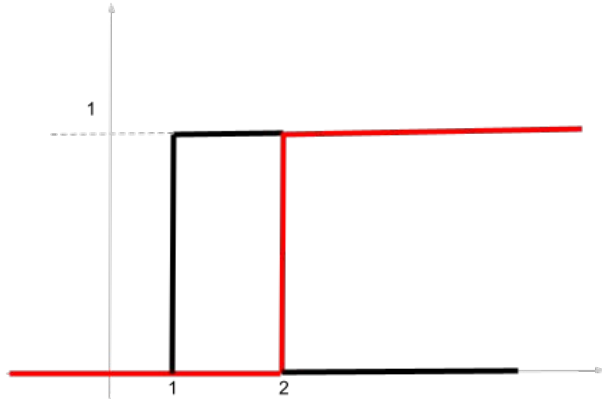


BUSCANDO ALTERNATIVAS...

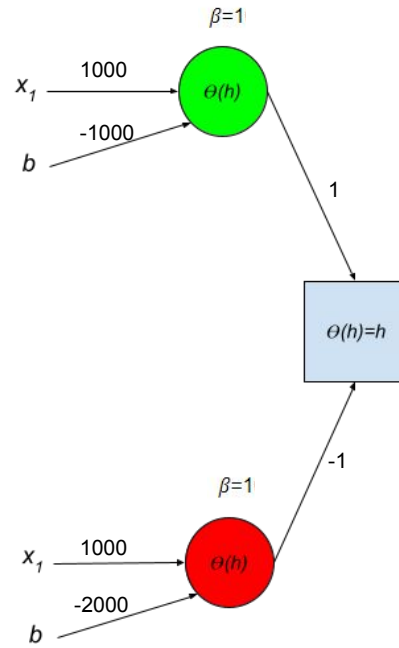
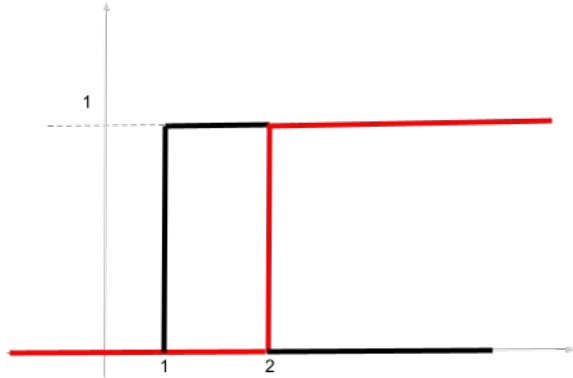
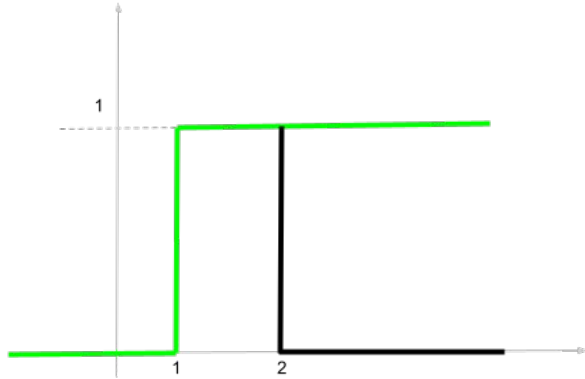


$$\theta(x) = \frac{1}{1 + \exp^{-2\beta x}}$$

$$Im = (0, 1)$$



BUSCANDO ALTERNATIVAS...



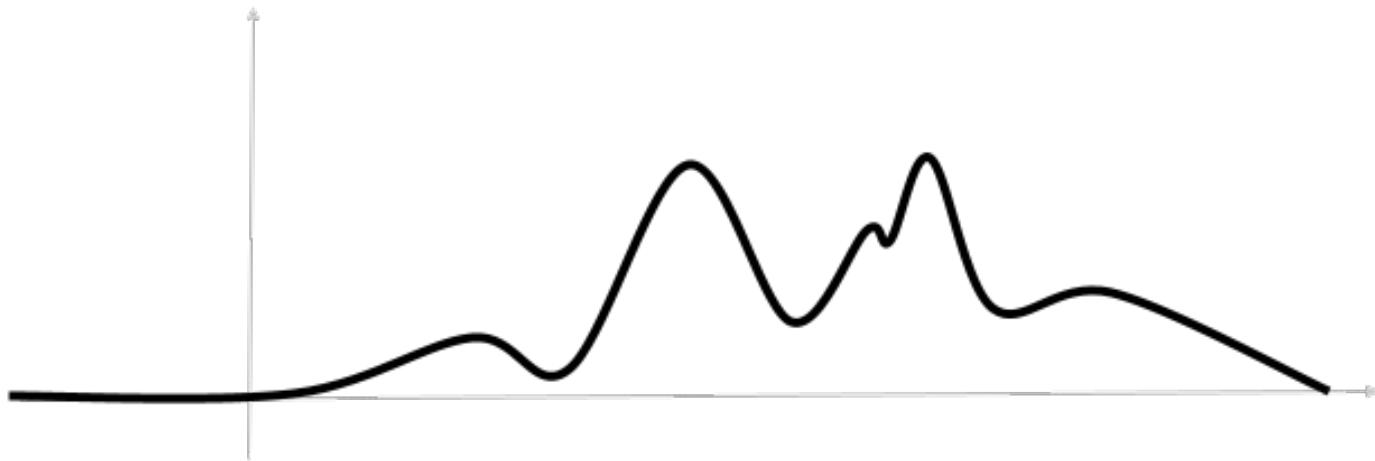
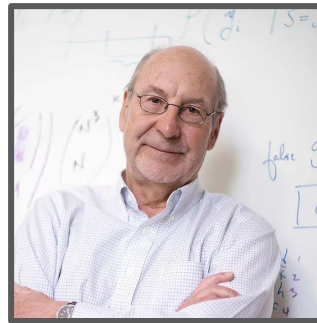
¿CÓMO PUEDO COMBINAR LOS PERCEPTRONES?

¿Qué clase de problemas
puedo representar?

TEOREMA DE APROXIMACIÓN UNIVERSAL

1989 - Cybenko, G.

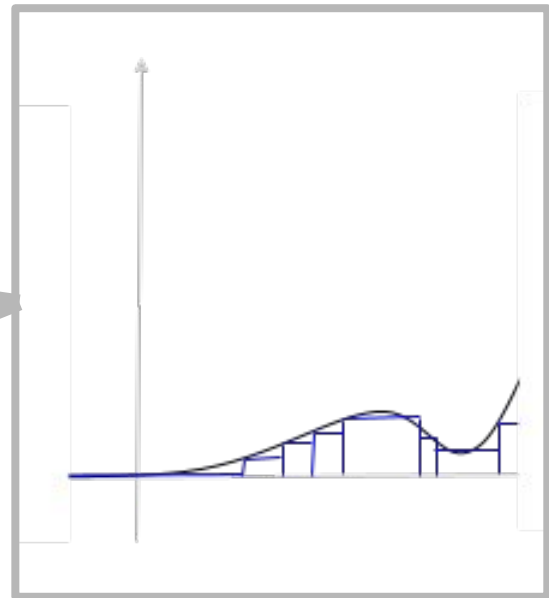
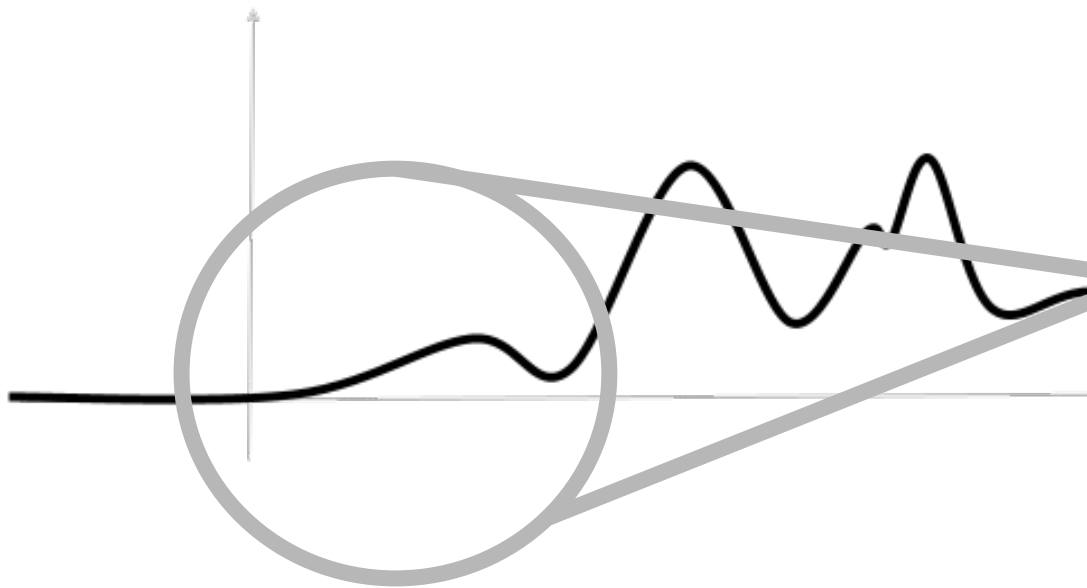
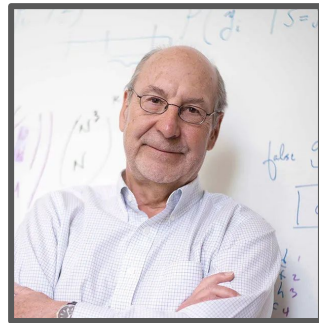
(expansión en 1990, Hornik, K.)

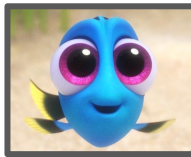


TEOREMA DE APROXIMACIÓN UNIVERSAL

1989 - Cybenko, G.

(expansión en 1990, Hornik, K.)





TEOREMA DE APROXIMACIÓN UNIVERSAL

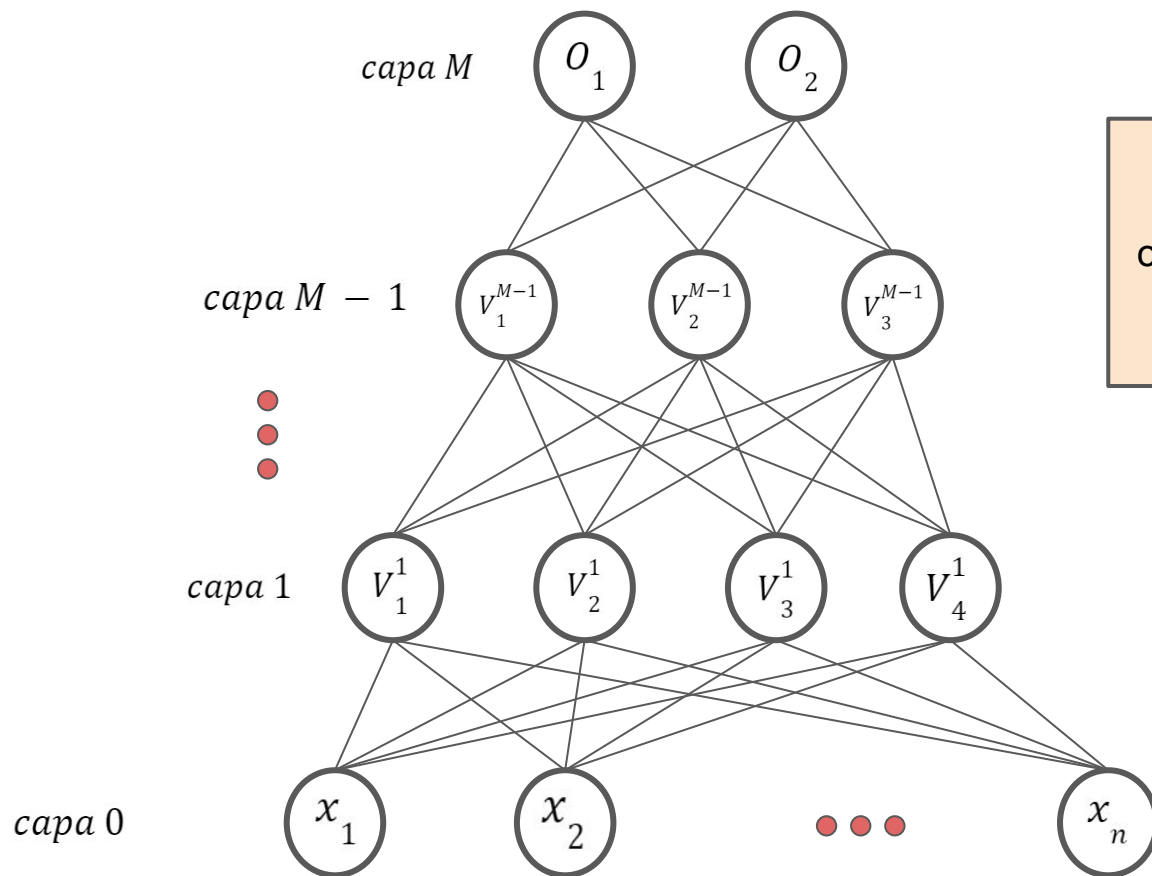
1989 - Cybenko, G.

(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-

$M = \text{cantidad capas}$

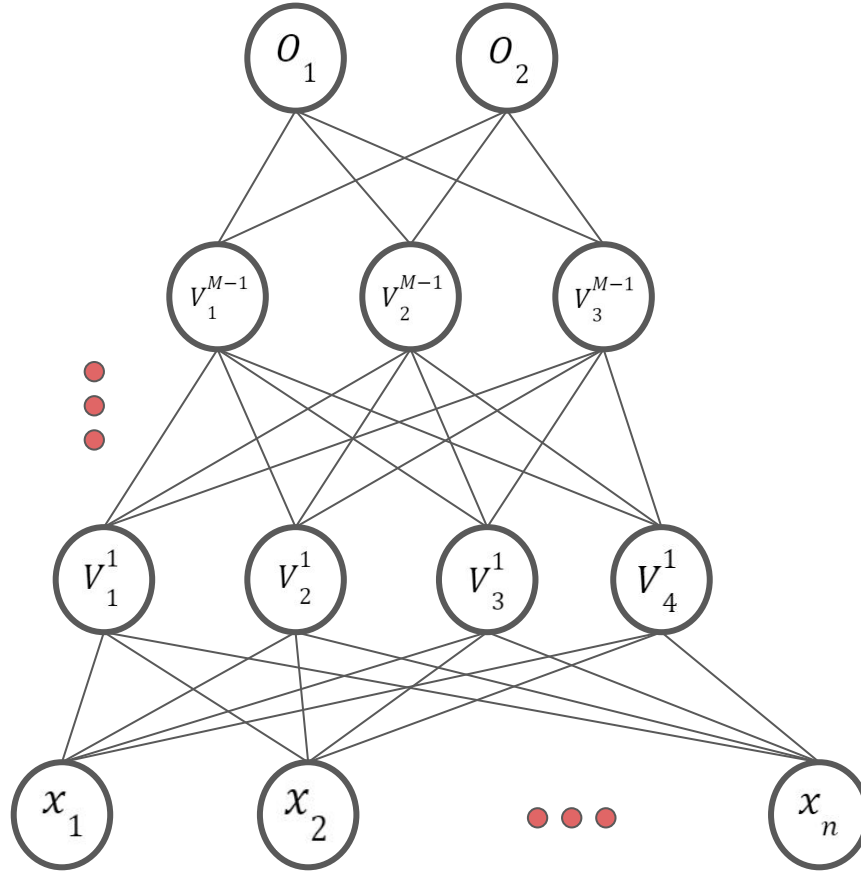
PERCEPTRÓN MULTICAPA



DISCLAIMER: nos olvidamos del bias por un rato

$p = \text{cantidad de datos}$
 $n = \text{cantidad de variables}$

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



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

Salida de la neurona:

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

Salida de la neurona de una capa intermedia:

$$V_j^m = \theta\left(\sum_{k=1}^{m-1} V_k^{m-1} \cdot w_{jk}^m\right) \quad m = 2 \dots M - 1$$

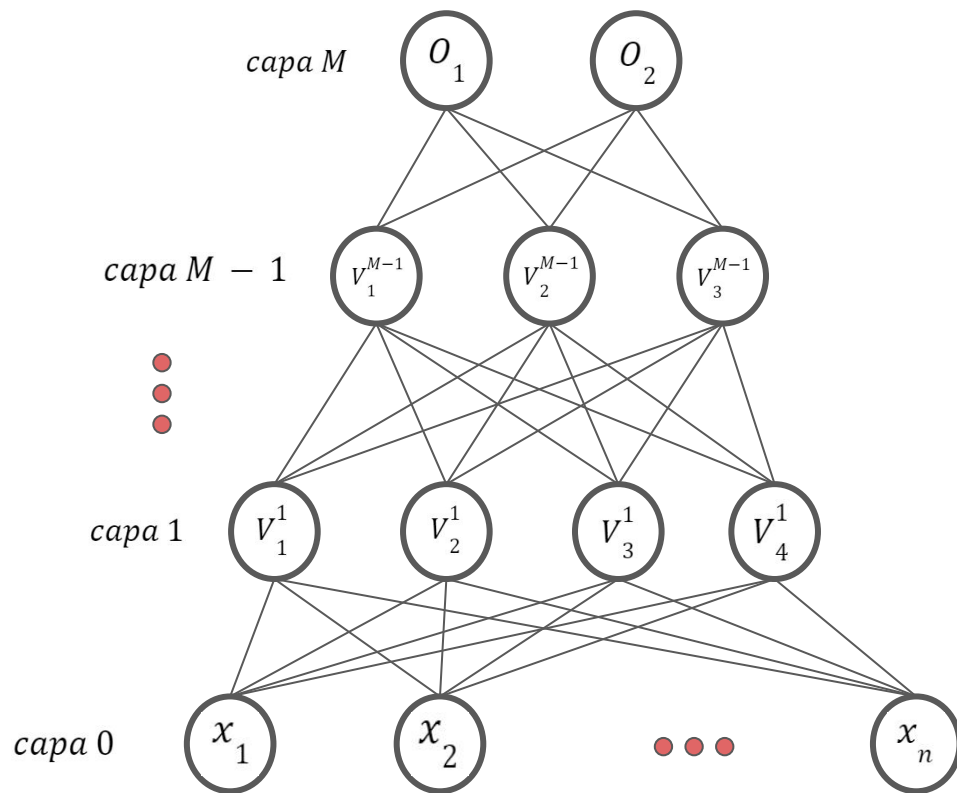
Salida de la neurona de la primera capa intermedia:

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

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

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

$$m = 1 \dots M \quad (O_i = V_i^M, x_i = V_i^0)$$



¿CÓMO VA CAMBIANDO EL ALGORITMO?

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate  $\eta$ 
```

```
for a fixed number of epochs:
```

```
  For each training example  $\mu$  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:
```

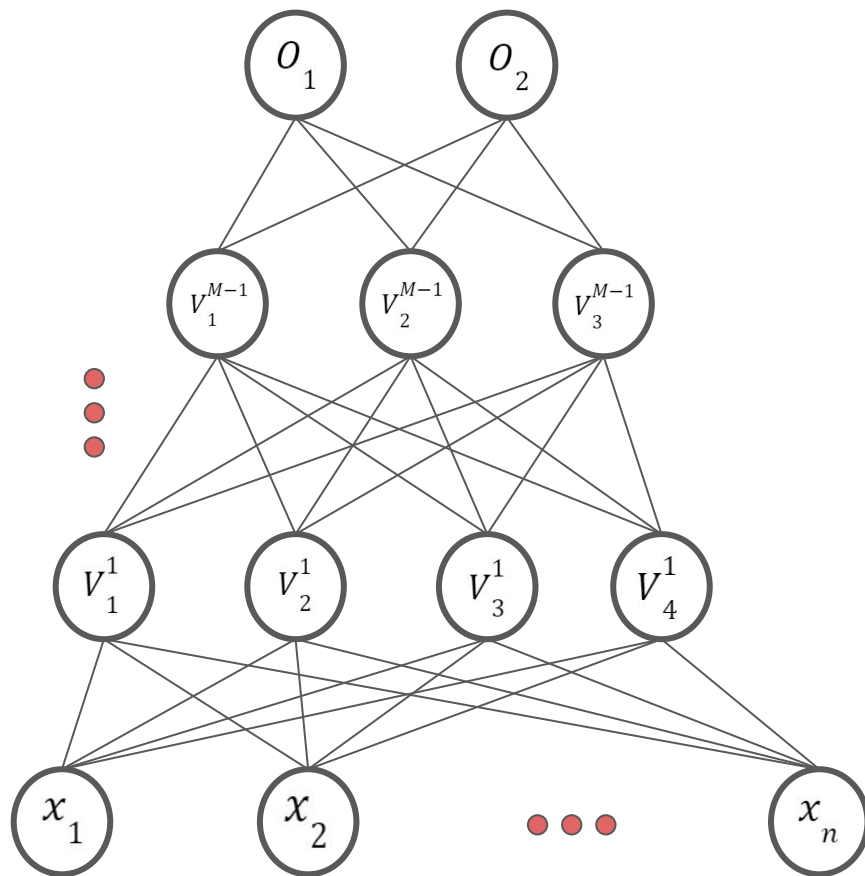
```
      error = f(xμ1, xμ2, ..., xμn)
```

```
      convergence = True if error <  $\epsilon$  else False
```

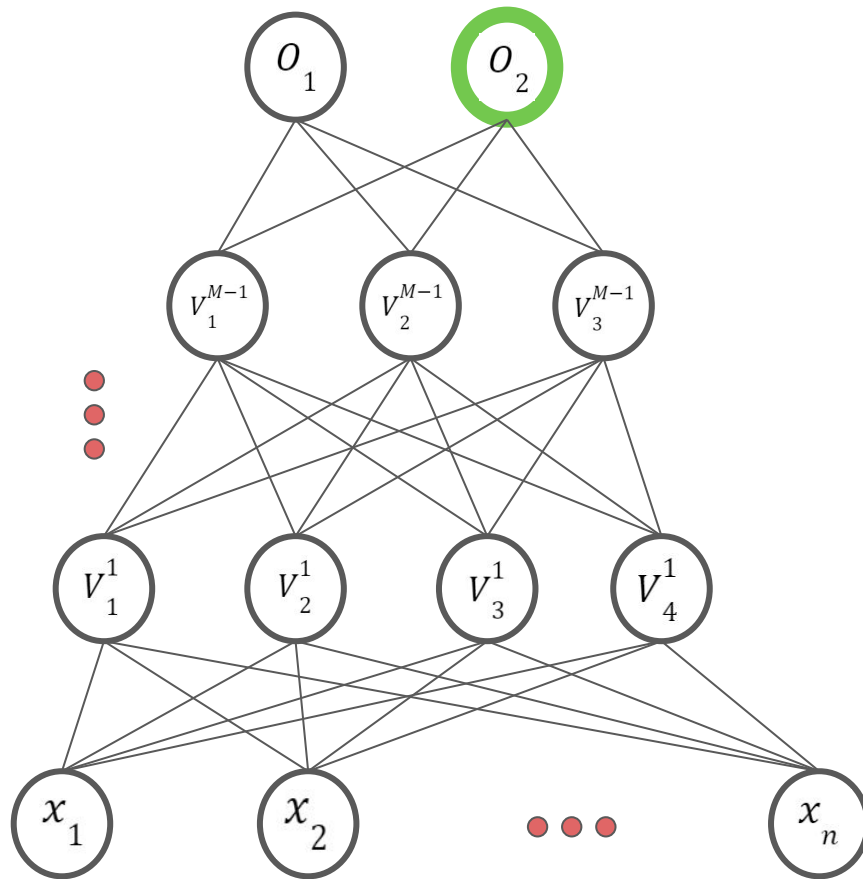
```
      if convergence: break
```

```
End
```

AHORA... ¿CÓMO LO ENTRENAMOS? 🧠🧠



AHORA... ¿CÓMO LO ENTRENAMOS? 🧐🧐

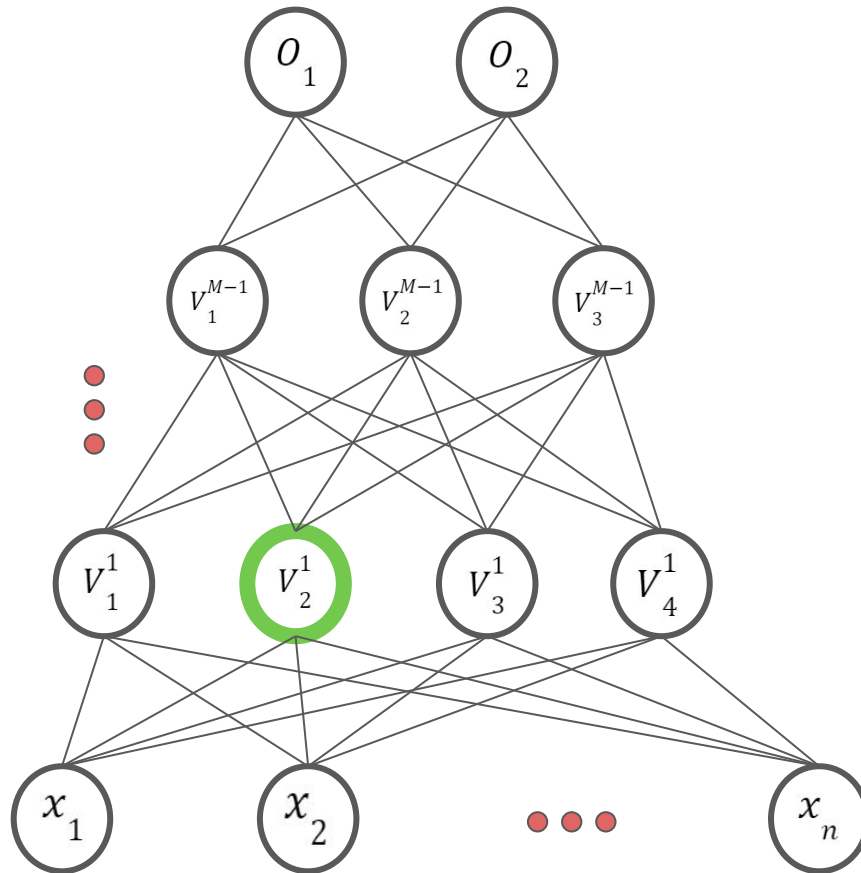


Recordemos...

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

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

AHORA... ¿CÓMO LO ENTRENAMOS? 🧠🧠

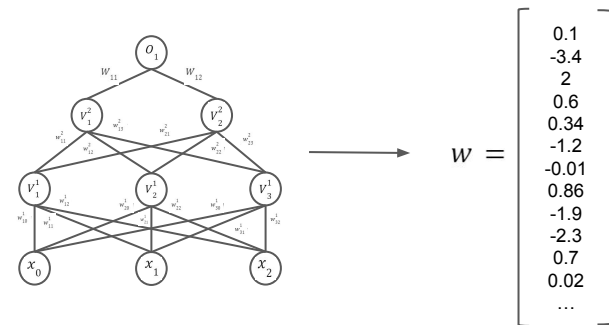
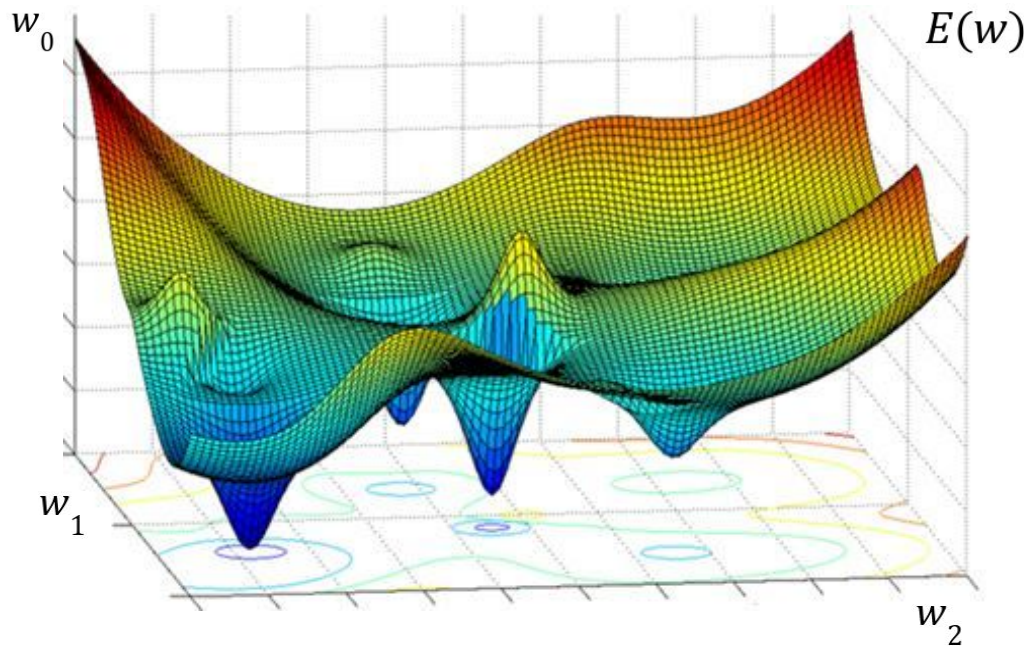


Recordemos...

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

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

ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL



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

$$\frac{\partial E}{\partial w_{11}^1}$$

$$\frac{\partial E}{\partial w_{11}^2}$$

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

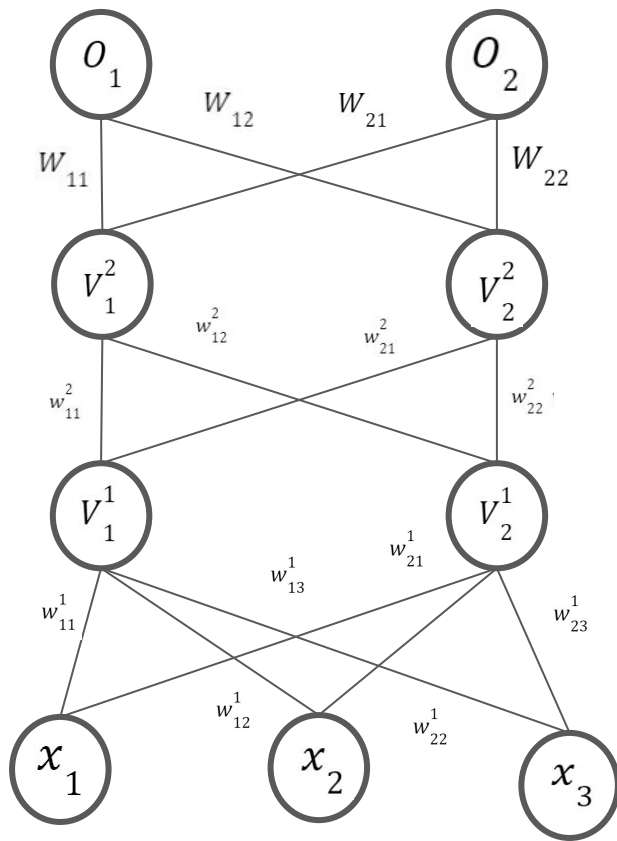
Índice	Descripción
i	índice de la neurona de la capa de salida (o siguiente)
j	índice de la neurona de la capa intermedia
k	índice de la neurona de entrada o de la capa anterior
m	índice de la capa intermedia
p	cantidad datos
μ	dato en particular

w_{jk}^m = pesos sinápticos

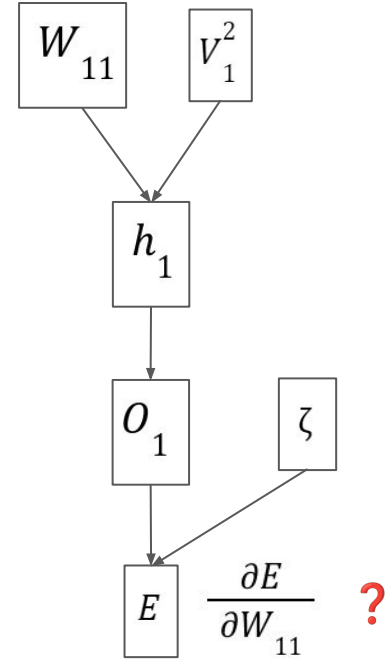
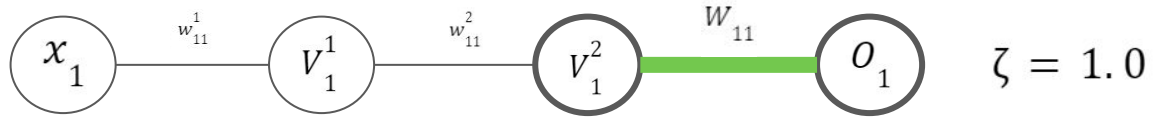
V_j^m = neurona de capa intermedia

W_{ij} = pesos sináticos de la última capa

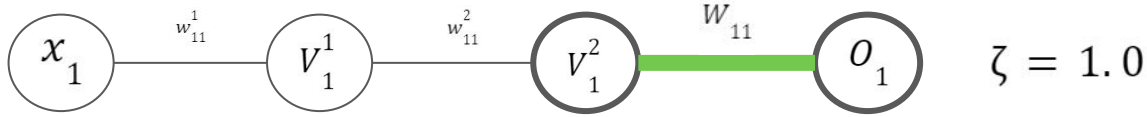
O_i = neurona de capa de salida



ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

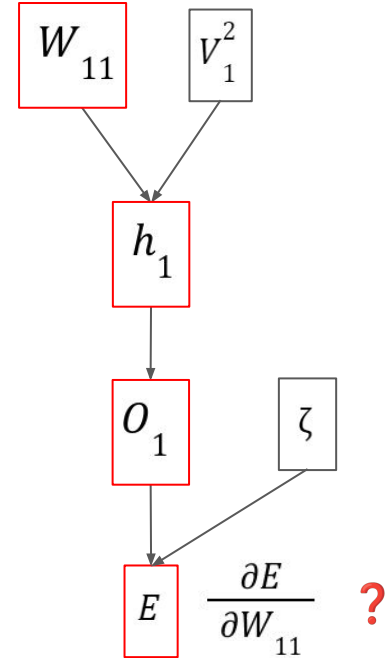


ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

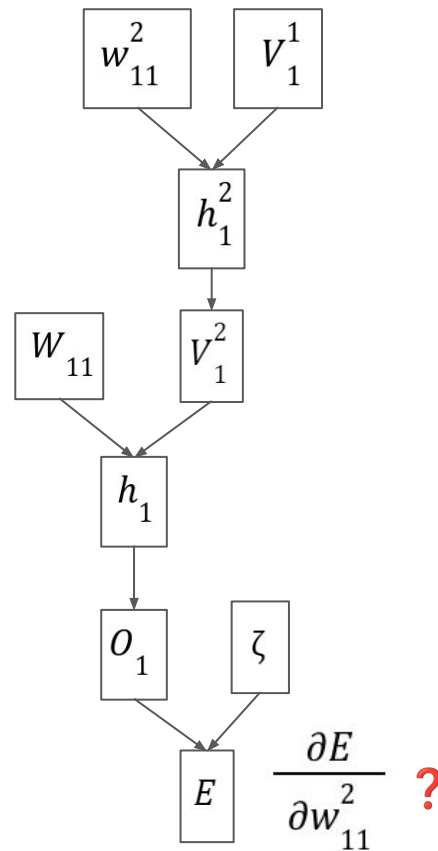
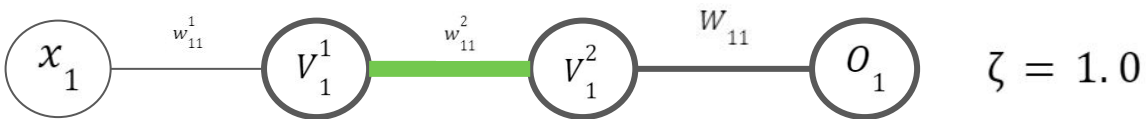


Aplicamos la regla de la cadena:

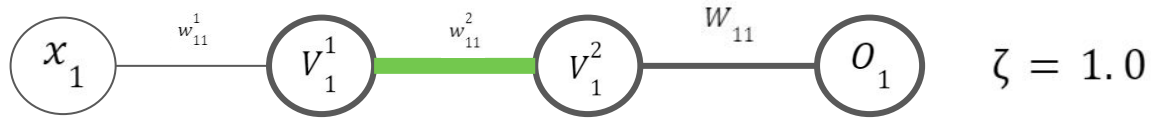
$$\frac{\partial E}{\partial W_{11}} = \frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial h_1} \frac{\partial h_1}{\partial W_{11}}$$



ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

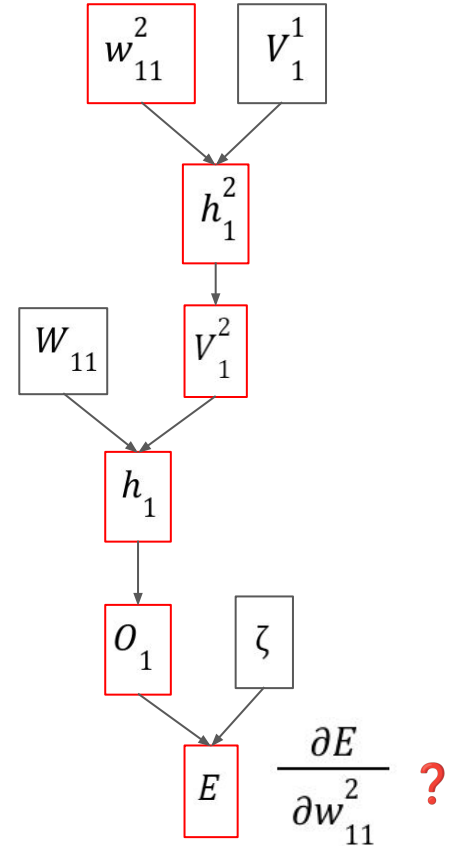


ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

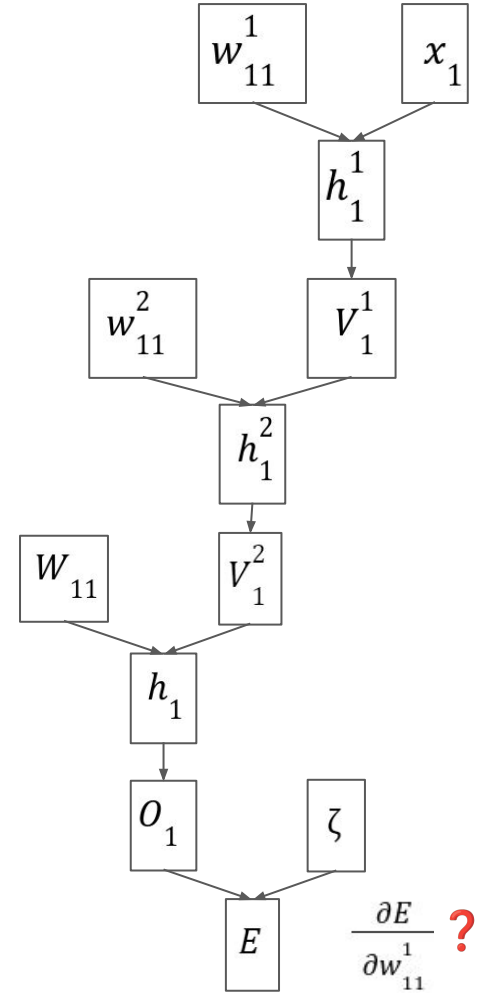
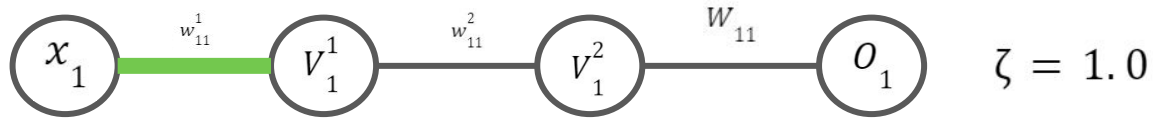


Aplicamos la regla de la cadena:

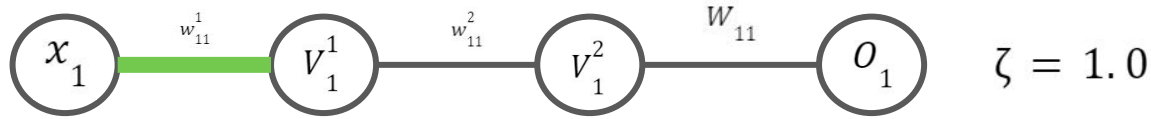
$$\frac{\partial E}{\partial w_{11}^2} = \frac{\partial E}{\partial o_1} \frac{\partial o_1}{\partial h_1} \frac{\partial h_1}{\partial V_1^2} \frac{\partial V_1^2}{\partial h_1^2} \frac{\partial h_1^2}{\partial w_{11}^2}$$



ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL

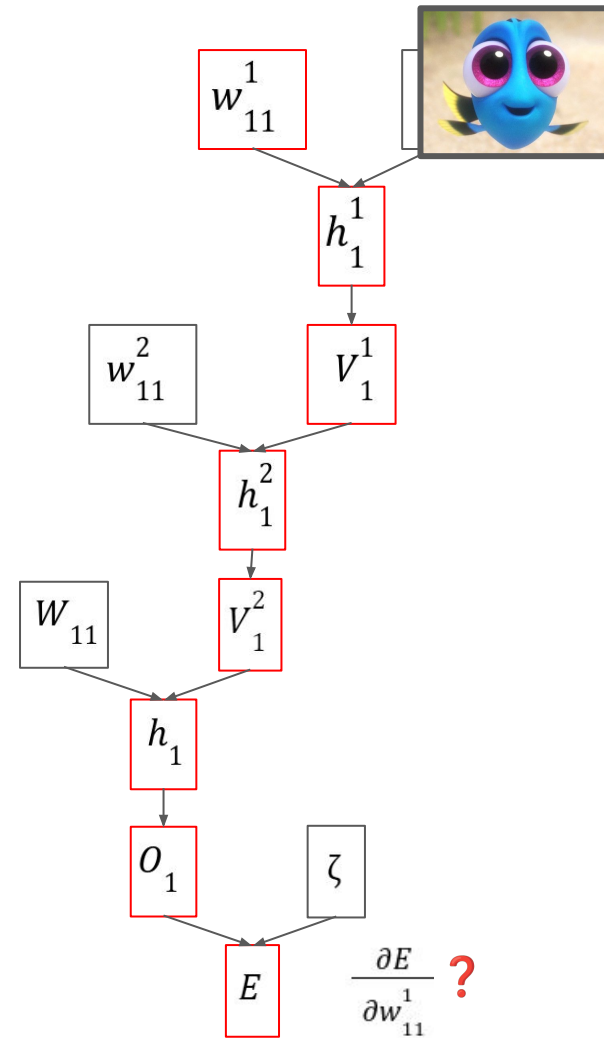


ENTRENAMIENTO: EXPLICACIÓN CONCEPTUAL



Aplicamos la regla de la cadena:

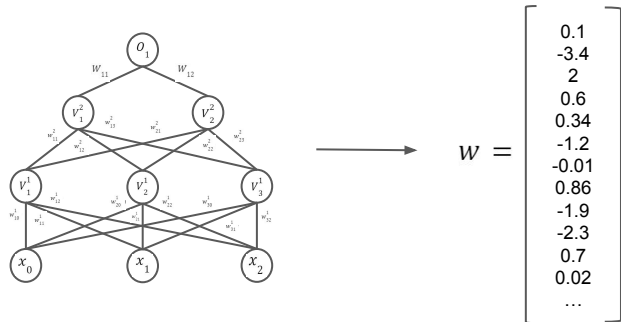
$$\frac{\partial E}{\partial w_{11}^1} = \frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial h_1} \frac{\partial h_1}{\partial V_1^2} \frac{\partial V_1^2}{\partial h_1^2} \frac{\partial h_1^2}{\partial V_1^1} \frac{\partial V_1^1}{\partial h_1^1} \frac{\partial h_1^1}{\partial w_{11}^1}$$



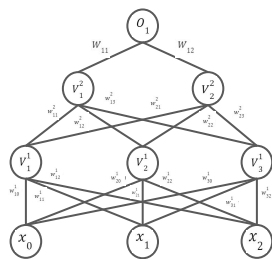
RUMELHART, HINTON, WILLIAMS (1986)



- 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.



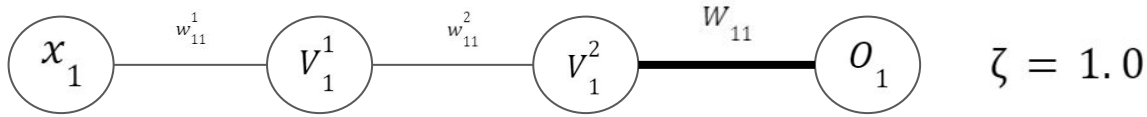
RUMELHART, HINTON, WILLIAMS (1986)



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

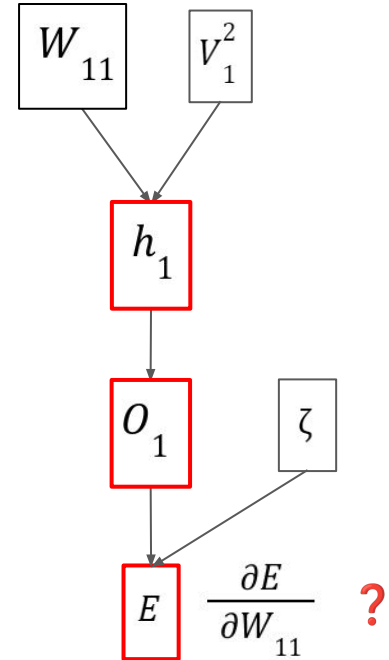
```
# Calculate loss, perform backward pass  
loss = F.cross_entropy(logits, labels)  
loss.backward()  
optimizer.step()  
optimizer.zero_grad()
```

DELTA: EXPLICACIÓN CONCEPTUAL

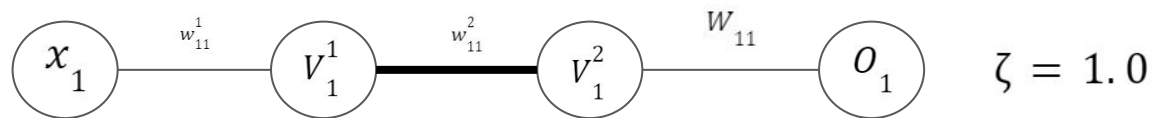


Aplicamos la regla de la cadena:

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

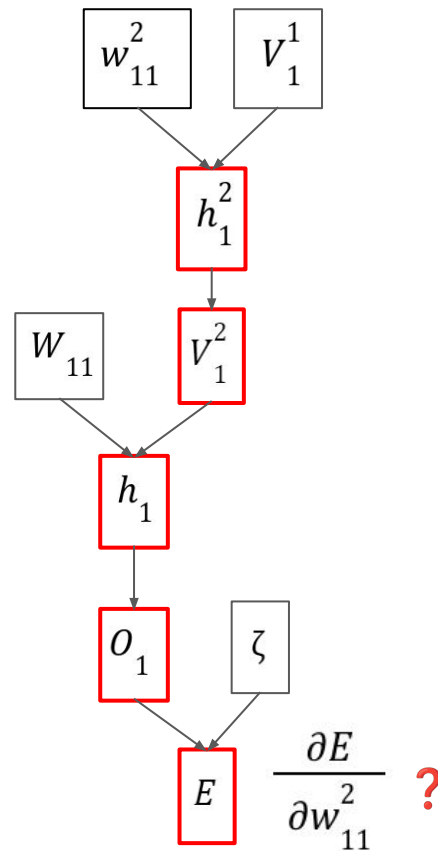


DELTA: EXPLICACIÓN CONCEPTUAL

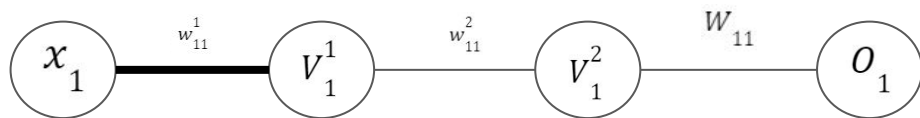


Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial w_{11}^2} = \frac{\partial E}{\partial o_1} \frac{\partial o_1}{\partial h_1} \frac{\partial h_1}{\partial V_1^2} \frac{\partial V_1^2}{\partial h_1^2} \frac{\partial h_1^2}{\partial w_{11}^2}$$



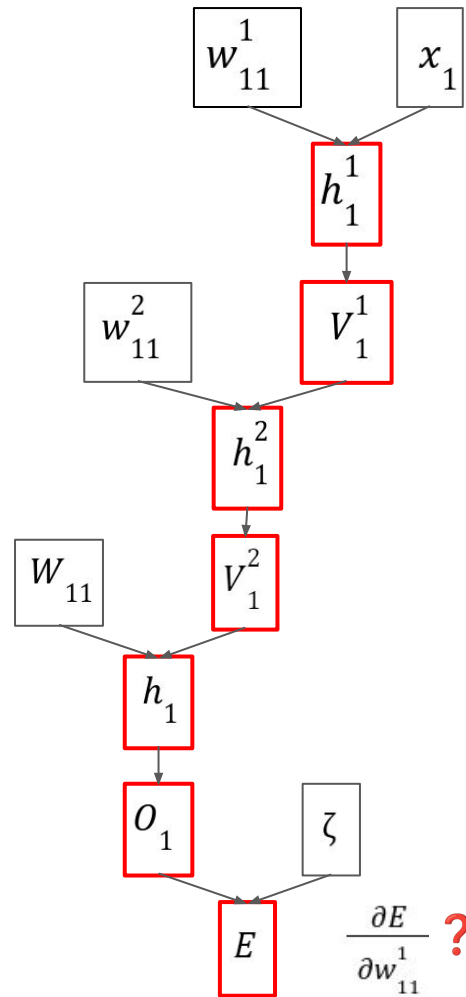
DELTA: EXPLICACIÓN CONCEPTUAL



$$\zeta = 1.0$$

Aplicamos la regla de la cadena:

$$\frac{\partial E}{\partial w_{11}^1} = \frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial h_1} \frac{\partial h_1}{\partial V_1^2} \frac{\partial V_1^2}{\partial h_1^2} \frac{\partial h_1^2}{\partial V_1^1} \frac{\partial V_1^1}{\partial w_{11}^1} \frac{\partial h_1^1}{\partial w_{11}^1}$$



¿CÓMO VA CAMBIANDO EL ALGORITMO?

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate  $\eta$ 
```

```
for a fixed number of epochs:
```

```
  For each training example  $\mu$  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:
```

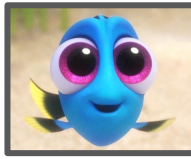
```
      error = f( $x_1^{\mu}, x_2^{\mu}, \dots, x_n^{\mu}$ )
```

```
      convergence = True if error <  $\epsilon$  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”)	$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$?
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”)	$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$	$V_j^m = \theta\left(\sum_{k=1} V_k^{m-1} \cdot w_{jk}^m\right)$ $m = 1 \dots M \quad (O_i = V_i^M, x_i = V_i^0)$
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”)	$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$	$V_j^m = \theta\left(\sum_{k=1} V_k^{m-1} \cdot w_{jk}^m\right)$ $m = 1 \dots M \quad (O_i = V_i^M, x_i = V_i^0)$
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

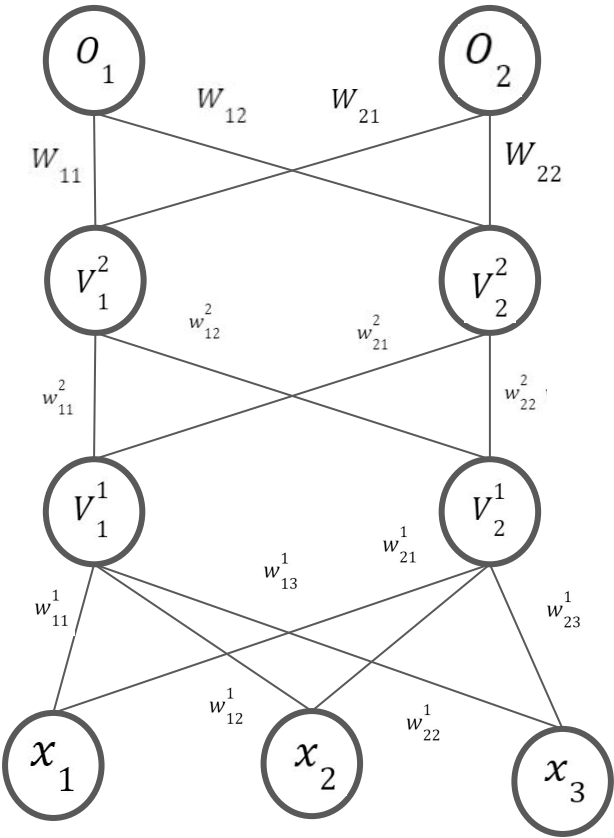
Índice	Descripción
i	índice de la neurona de la capa de salida (o siguiente)
j	índice de la neurona de la capa intermedia
k	índice de la neurona de entrada o de la capa anterior
m	índice de la capa intermedia
p	cantidad datos
μ	dato en particular

w_{jk}^m = pesos sinápticos

V_j^m = neurona de capa intermedia

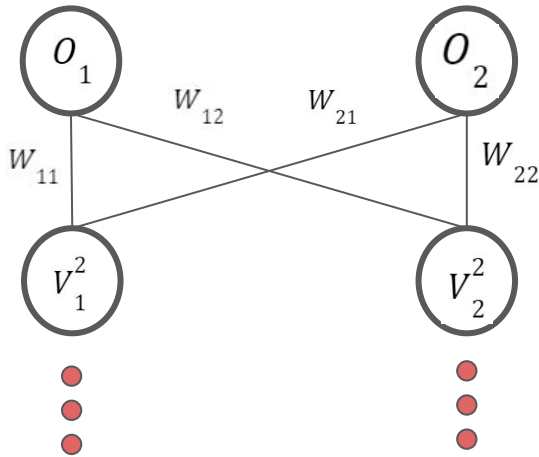
W_{ij} = pesos sináticos de la última capa

O_i = neurona de capa de salida



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

$$E(O) = \frac{1}{2} \sum_{\mu} \sum_i (\zeta_i^{\mu} - o_i^{\mu})^2$$
$$o_i = \theta(h_i)$$
$$h_i = \sum_{j=1} V_j^{M-1} \cdot W_{ij}$$



$$\Delta w = - \eta \left[\frac{\partial E}{\partial w} \right] \longrightarrow \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial o_i} \frac{\partial o_i}{\partial h_i} \frac{\partial h_i}{\partial W_{ij}}$$

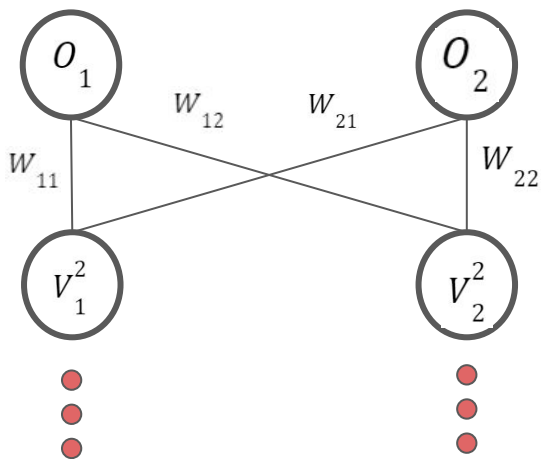
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

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

$$E(O) = \frac{1}{2} \sum_i (\zeta_i^u - o_i^u)^2$$

$$o_i = \theta(h_i)$$

$$h_i = \sum_{j=1}^{M-1} v_j^{M-1} \cdot w_{ij}$$



$$\Delta w = -\eta \left[\frac{\partial E}{\partial w} \right] \longrightarrow \frac{\partial E}{\partial w_{ij}} = \left[\frac{\partial E}{\partial o_i} \right] \left[\frac{\partial o_i}{\partial h_i} \right] \left[\frac{\partial h_i}{\partial w_{ij}} \right]$$

$$\frac{\partial E}{\partial w_{ij}} = (\zeta_i - o_i)(-1) \theta'(h_i)(1) v_j^{M-1}$$

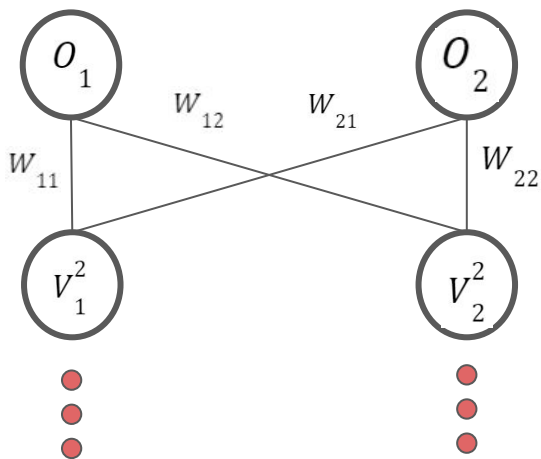
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

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

$$E(O) = \frac{1}{2} \sum_i (\zeta_i^u - o_i^u)^2$$

$$O_i = \theta(h_i)$$

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



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

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

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

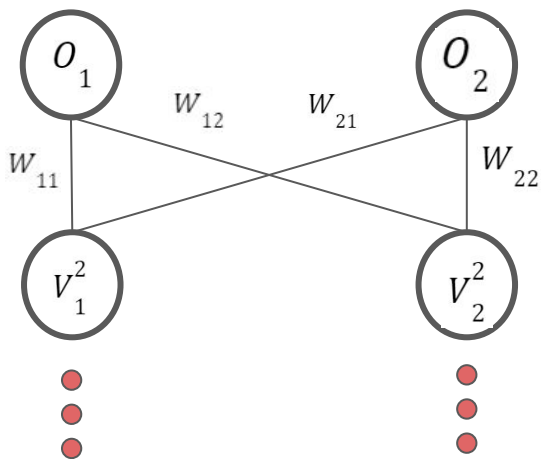
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

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

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

$$O_i = \theta(h_i)$$

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



$$\Delta w = -\eta \left[\frac{\partial E}{\partial w} \right] \longrightarrow \frac{\partial E}{\partial W_{ij}} = \left[\frac{\partial E}{\partial O_i} \right] \left[\frac{\partial O_i}{\partial h_i} \right] \left[\frac{\partial h_i}{\partial W_{ij}} \right]$$

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

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

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

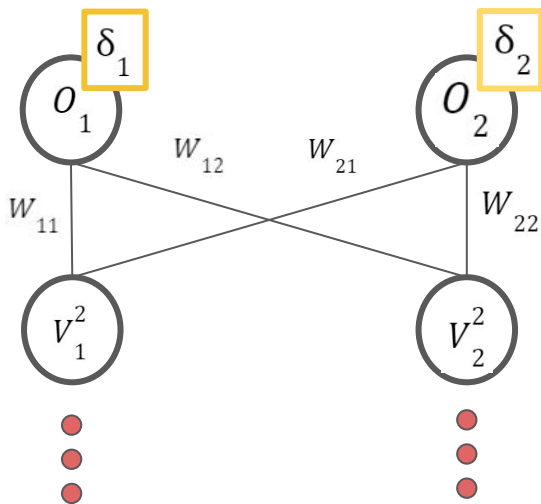
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

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

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

$$O_i = \theta(h_i)$$

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



$$\Delta w = - \eta \left[\frac{\partial E}{\partial w} \right] \longrightarrow \frac{\partial E}{\partial W_{ij}} = \left[\frac{\partial E}{\partial O_i} \right] \left[\frac{\partial O_i}{\partial h_i} \right] \left[\frac{\partial h_i}{\partial W_{ij}} \right]$$

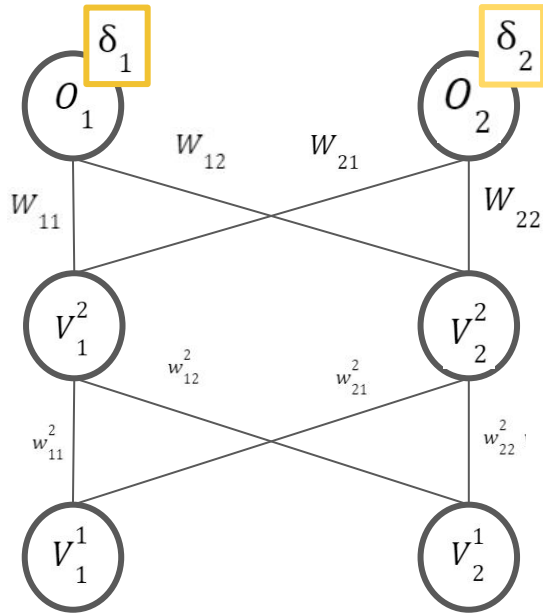
$$\frac{\partial E}{\partial W_{ij}} = (\zeta_i - O_i)(-1) \theta'(h_i)(1) V_j^{M-1}$$

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

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

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

RETROPROPAGACIÓN: CAPA OCULTA ($j = M-1, j \text{ in } [M-1, \dots, 1]$)



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

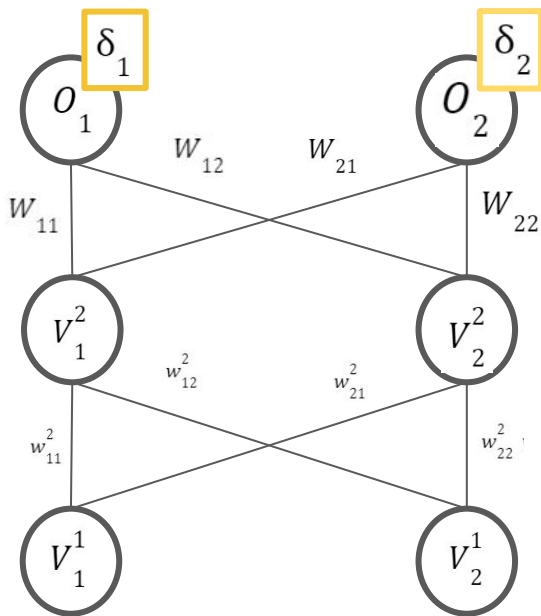
Notar que el ejemplo está hecho para un dato de entrada. Se deberá extender al conjunto.

RETROPROPAGACIÓN: CAPA OCULTA ($j = M-1, j \text{ in } [M-1, \dots, 1]$)

$$E(O) = \frac{1}{2} \sum_i (\zeta_i^u - o_i^u)^2$$

$$O_i = \theta(h_i)$$

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

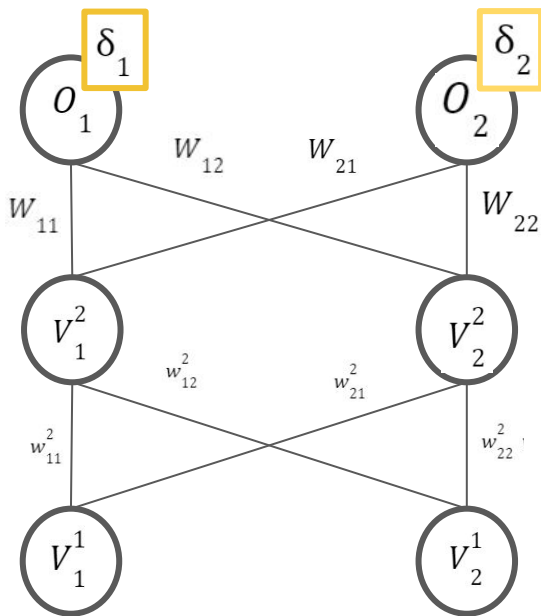


$$\Delta w = -\eta \left[\frac{\partial E}{\partial w} \right] \longrightarrow \frac{\partial E}{\partial w_{jk}^m} = \left[\frac{\partial E}{\partial O} \right] \left[\frac{\partial O}{\partial h} \right] \left[\frac{\partial h}{\partial V_j^m} \right] \frac{\partial V_j^m}{\partial h_j^m} \frac{\partial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

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

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

RETROPROPAGACIÓN: CAPA OCULTA ($j = M-1, j \text{ in } [M-1, \dots, 1]$)



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

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

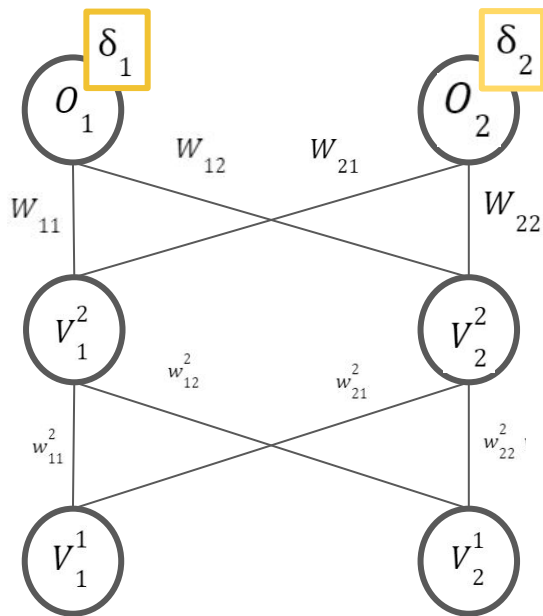
$$\frac{\partial E}{\partial w_{jk}^m} = - \sum_i \delta_i W_{ij} \frac{\partial V_j^m}{\partial h_j^m} \frac{\partial h_j^m}{\partial w_{jk}^m}$$

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

RETROPROPAGACIÓN: CAPA OCULTA ($j = M-1, j \text{ in } [M-1, \dots, 1]$)

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

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



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

$$\frac{\partial E}{\partial w_{jk}^m} = - \sum_i \delta_i W_{ij} \left[\frac{\partial V_j^m}{\partial h_j^m} \right] \left[\frac{\partial h_j^m}{\partial w_{jk}^m} \right]$$

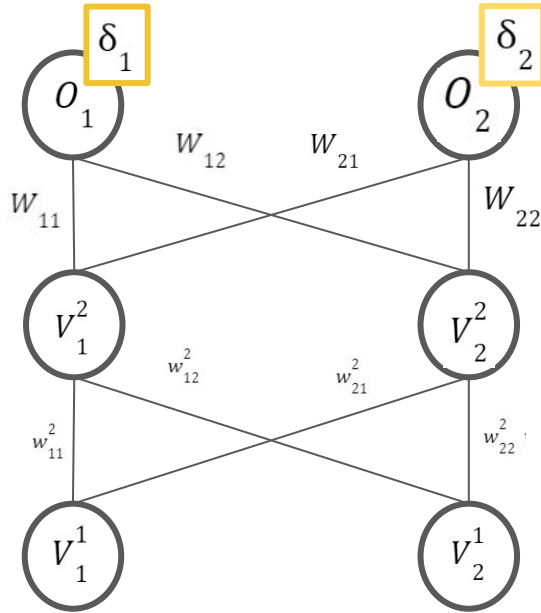
$$\frac{\partial E}{\partial w_{jk}^m} = - \sum_i \delta_i W_{ij} \theta'(h_j^m)(1) V_k^{m-1}$$

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

RETROPROPAGACIÓN: CAPA OCULTA ($j = M-1, j \text{ in } [M-1, \dots, 1]$)

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

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



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

$$\frac{\partial E}{\partial w_{jk}^m} = - \sum_i \delta_i W_{ij} \left[\frac{\partial V_j^m}{\partial h_j^m} \right] \left[\frac{\partial h_j^m}{\partial w_{jk}^m} \right]$$

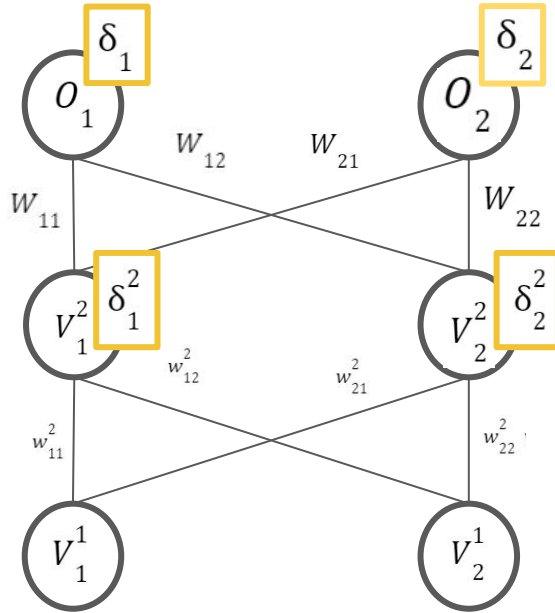
$$\frac{\partial E}{\partial w_{jk}^m} = - \underbrace{\sum_i \delta_i W_{ij} \theta'(h_j^m)(1)}_{\delta_j^m} V_k^{m-1}$$

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

RETROPROPAGACIÓN: CAPA OCULTA ($j = M-1, j \text{ in } [M-1, \dots, 1]$)

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

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



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

$$\frac{\partial E}{\partial w_{jk}^m} = - \sum_i \delta_i W_{ij} \left[\frac{\partial V_j^m}{\partial h_j^m} \right] \left[\frac{\partial h_j^m}{\partial w_{jk}^m} \right]$$

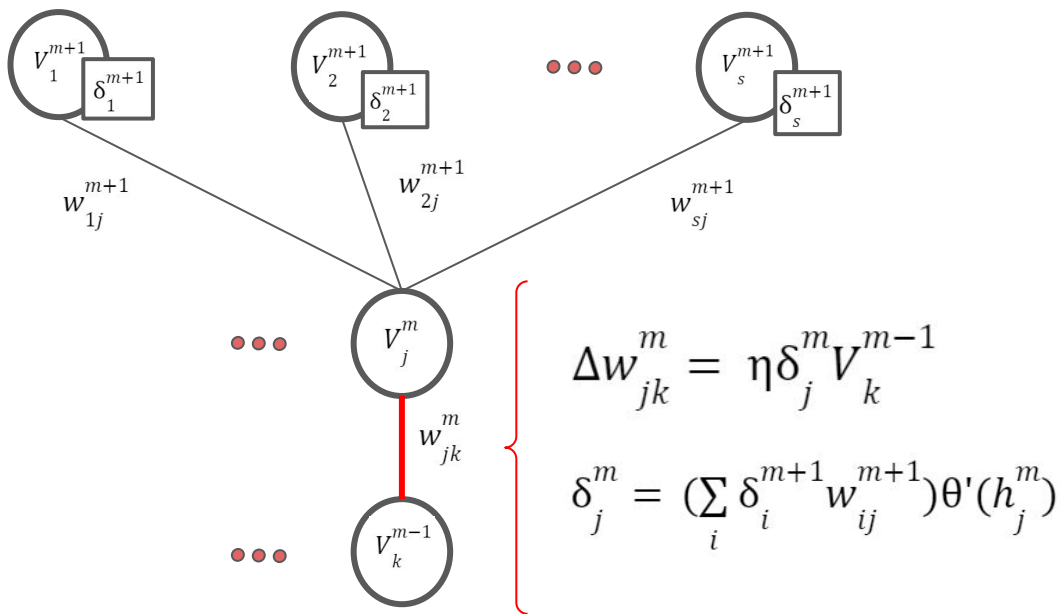
$$\frac{\partial E}{\partial w_{jk}^m} = - \underbrace{\sum_i \delta_i W_{ij} \theta'(h_j^m)(1)}_{\delta_j^m} V_k^{m-1}$$

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

RETROPROPAGACIÓN: CAPA OCULTA (j in $[M-1, \dots, 1]$)

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

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



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”)	$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$	$V_j^m = \theta\left(\sum_{k=1} V_k^{m-1} \cdot w_{jk}^m\right)$ $m = 1 \dots M \quad (O_i = V_i^M, x_i = V_i^0)$
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	$w^{nuevo} = w^{anterior} + \Delta w$ $\Delta w = - \eta \frac{\partial E}{\partial w}$	$\Delta W_{ij} = \eta \delta_i V_j \quad \delta_i = (\zeta_i - O_i) \theta'(h_i)$ $\Delta w_{jk}^m = \eta \delta_j^m V_k^{m-1} \quad \delta_j^m = \left(\sum_i \delta_i^{m+1} w_{ij}^{m+1}\right) \theta'(h_j^m)$

¿CÓMO VA CAMBIANDO EL ALGORITMO?

```
Initialize multi layer perceptron architecture
Initialize weights w to small random values
Set learning rate  $\eta$ 
```

```
for a fixed number of epochs:
```

```
  For each training example  $\mu$  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  $w_i$ :
```

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

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

```
    3. Calculate perceptron error:
```

$$\text{error} = f(x_1^{\mu}, x_2^{\mu}, \dots, x_n^{\mu})$$

```
    convergence = True if error <  $\epsilon$  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  $\eta$ 
```

```
for a fixed number of epochs:
```

```
     $\Delta \mathbf{w} = \mathbf{0}$ 
```

```
    For each training example  $\mu$  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. Calculate the weights and bias:
```

```
            For each weight  $w_i$ :
```

$$\Delta \mathbf{w} = \Delta \mathbf{w} + \Delta \mathbf{w}_i, \text{ where } \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \text{ (gradient descent)}$$

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

```
     $\mathbf{w} = \mathbf{w} + \Delta \mathbf{w}$ 
```

```
    Calculate perceptron error:
```

```
        error =  $f(x_1^{\mu}, x_2^{\mu}, \dots, x_n^{\mu})$ 
```

```
        convergence = True if error <  $\epsilon$  else False
```

```
        if convergence: break
```

```
End
```

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

PERCEPTRÓN SIMPLE

PERCEPTRÓN MULTICAPA

Proceso para
obtener la salida
de la neurona
("predecir")

Puede usarse Mean
Square Error (MSE)
como función de costo

Función de costo
(medir error con
respecto a la
salida esperada)

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

Proceso de
"aprendizaje"
para ajustar los
pesos

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**:

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

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 :

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

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

3. Calculate perceptron error:

$$\text{error} = f(x_1^{\mu}, x_2^{\mu}, \dots, x_n^{\mu})$$

convergence = True if error < ϵ else False

if convergence: break

End

RESUMEN

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

$$\min_{x \in \mathbb{R}^n} f(x), \quad f : \mathbb{R}^n \rightarrow \mathbb{R}, \quad \vec{x} = [x_1, x_2, \dots, x_n] \quad (3)$$

$$g_i(\vec{x}) \leq 0 \quad (4)$$

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

- 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, Powell, 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>

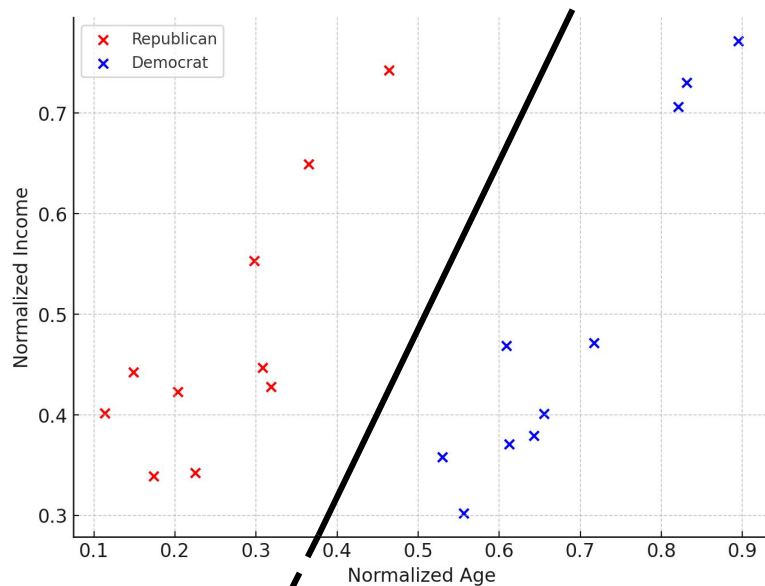
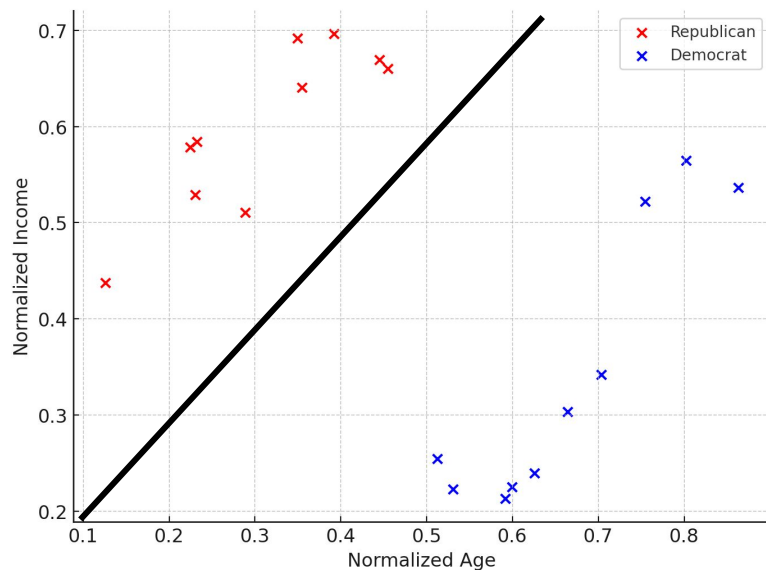
Rodrigo Ramele (2024) ***Reglamento y Apuntes de Sistemas de Inteligencia Artificial***,
Capítulo 7.

BIAS EN PERCEPTRÓN MULTICAPA



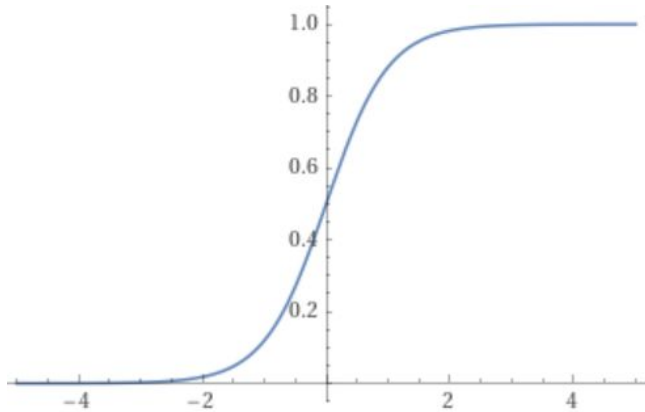
EXTRA: ¿CÓMO INCLUYO EL BIAS?

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)

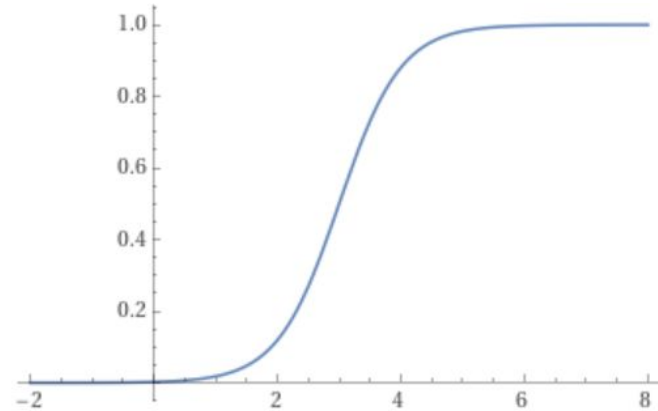


EXTRA: ¿CÓMO INCLUYO EL BIAS?

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.

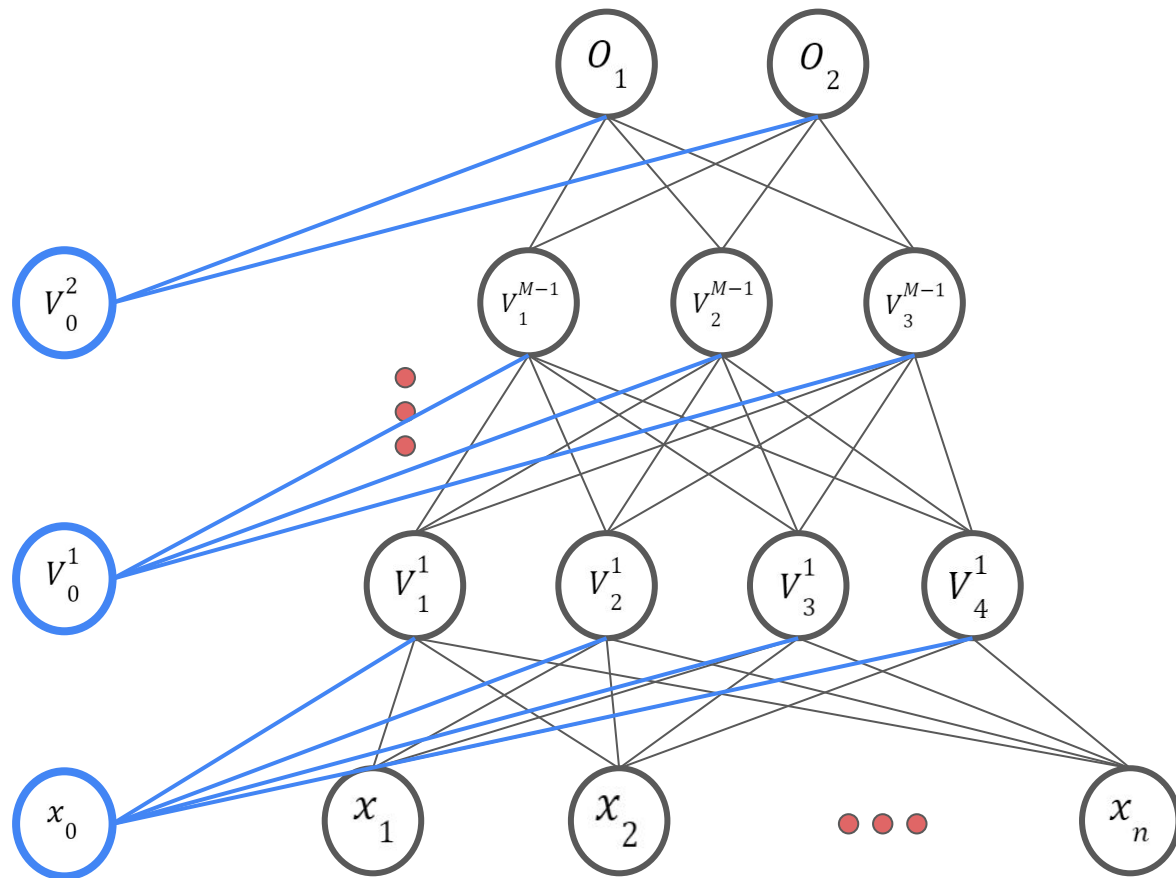
$$O(x) = \sum_{i=1}^n x_i \cdot w_i + w_0$$



$$\theta \left(\sum_{i=0}^n x_i^\mu \cdot w_i \right)$$

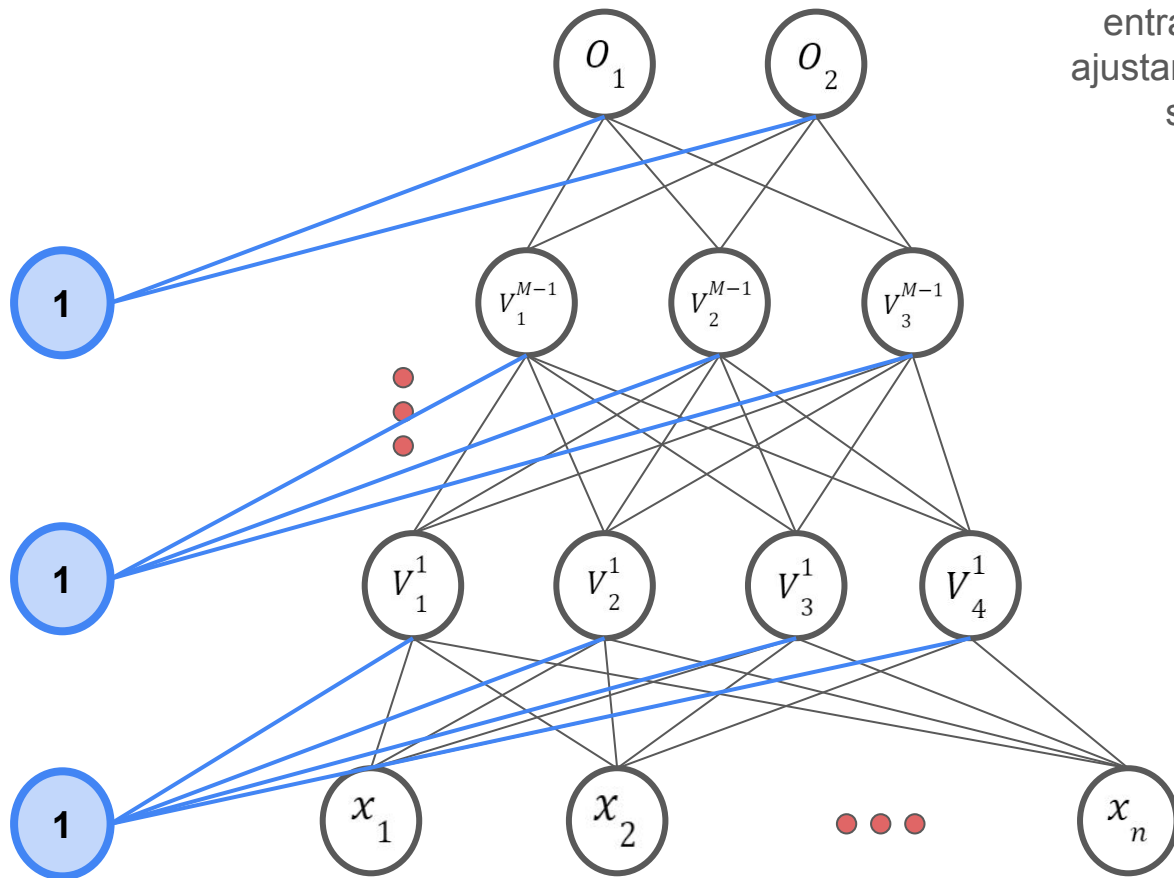
x0	Age	Income	Party
1.0	0.445	0.669	1.0
1.0	0.349	0.692	1.0
1.0	0.232	0.585	1.0
1.0	0.125	0.438	1.0
1.0	0.224	0.579	1.0
1.0	0.23	0.529	1.0
1.0	0.392	0.696	1.0
1.0	0.355	0.641	1.0
1.0	0.455	0.66	1.0
1.0	0.289	0.51	1.0
1.0	0.513	0.255	-1.0
1.0	0.755	0.522	-1.0
1.0	0.626	0.24	-1.0
1.0	0.703	0.342	-1.0
1.0	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

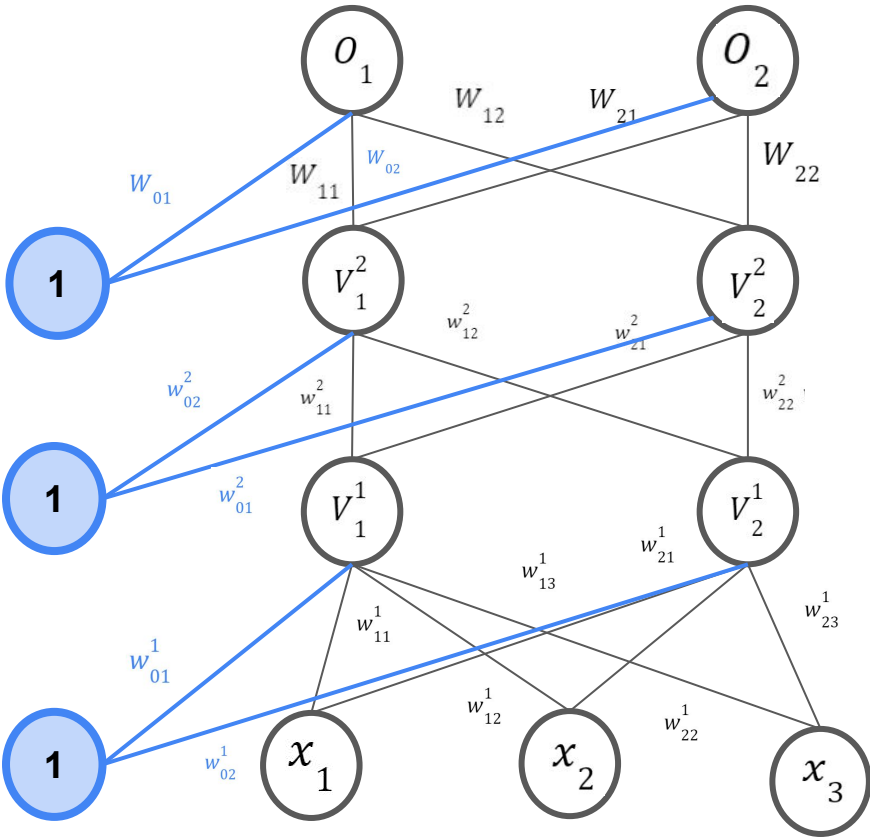
A partir de ahora comienza desde 0

w_{jk}^m = pesos sinápticos

V_j^m = 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?

