

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

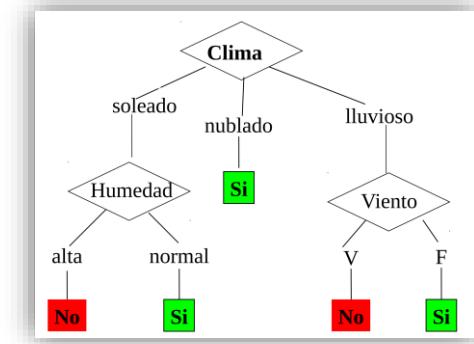
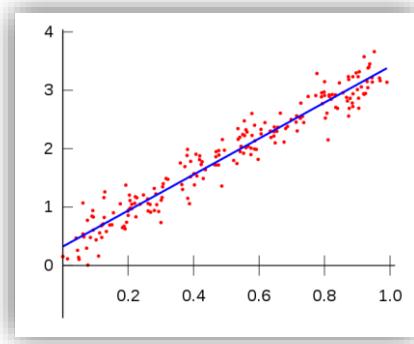
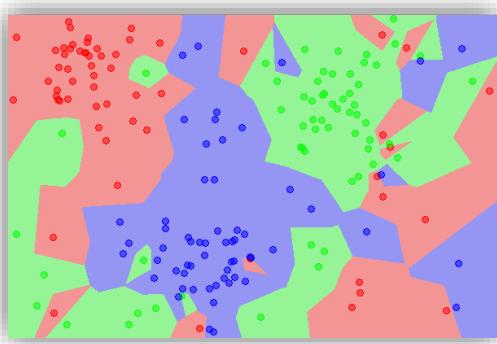


IIC2613 – Inteligencia Artificial

Redes Neuronales

Profesor: Hans Löbel

Reflexionemos un segundo con respecto a las técnicas que hemos visto hasta ahora



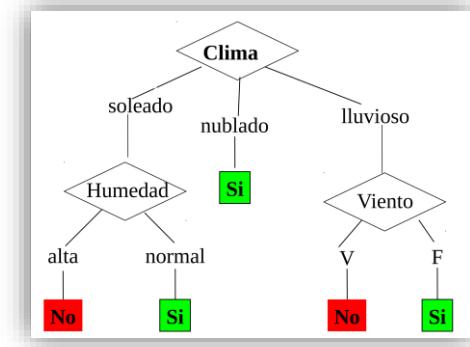
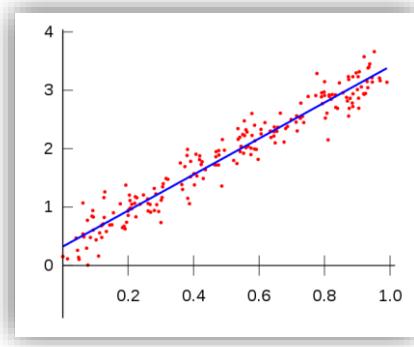
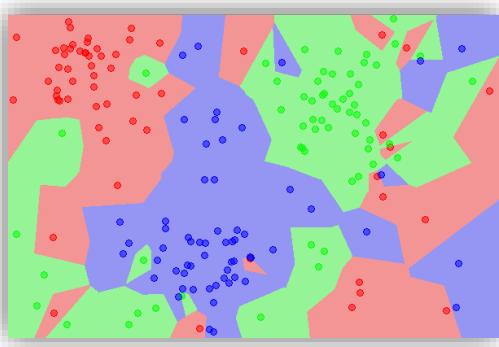
¿Cómo podríamos caracterizarlas?
(datos de entrada)

¿Qué es lo que estamos procesando?

En la práctica, estamos usando características (*features*) de los datos

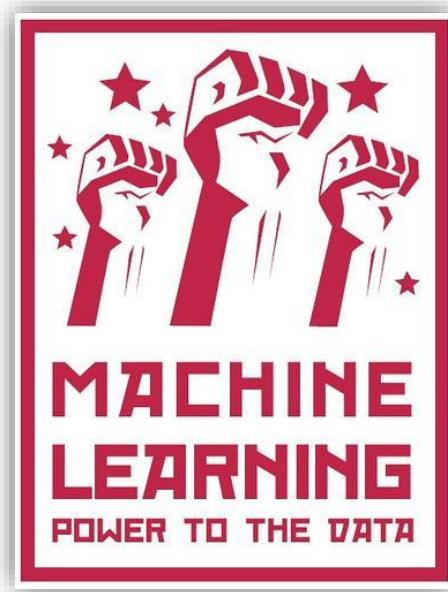
- Entradas seleccionadas tienen que ver más con el conocimiento experto que tengamos (o no) del problema.
- Esto no es ideal por dos razones principales:
 - i. un experto es caro
 - ii. nada asegura que este conocimiento sea el mejor.
- ¿Qué es lo que realmente están aprendiendo estos algoritmo?

¿Qué es lo que realmente están aprendiendo estos algoritmos?



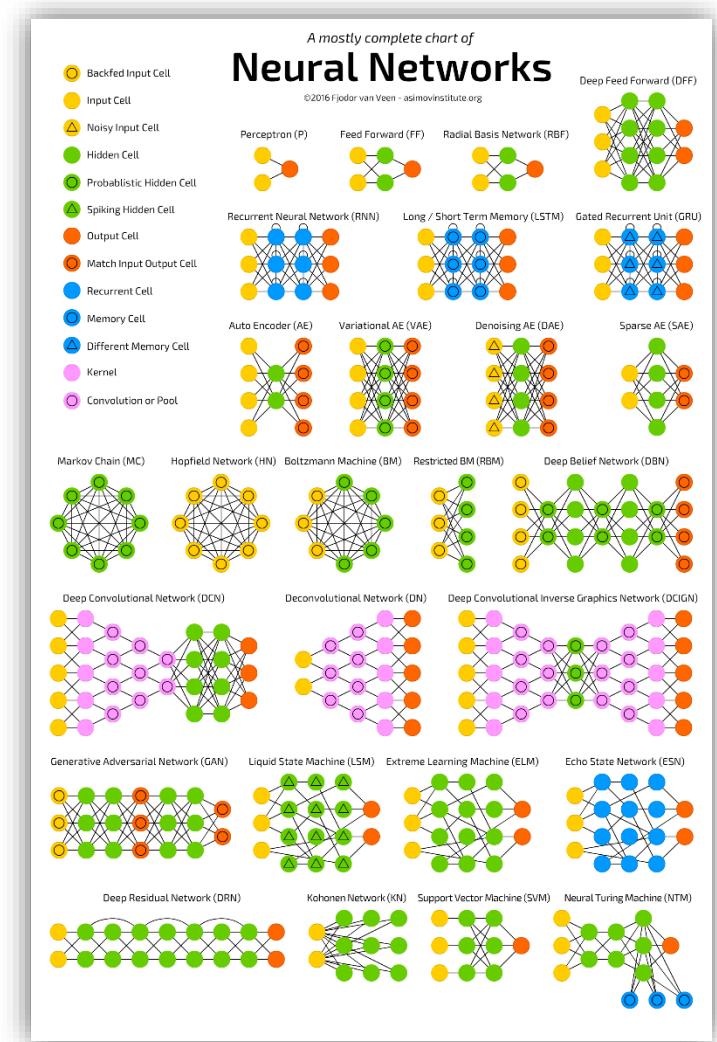
Hasta ahora no estamos “realmente” aprendiendo de los datos, sólo como separarlos (features)

Buscamos un mecanismo de aprendizaje, suficientemente general, para aprender aspectos complejos y desconocidos de los datos



Redes neuronales son capaces de aprender distintas **capas de representaciones de datos**

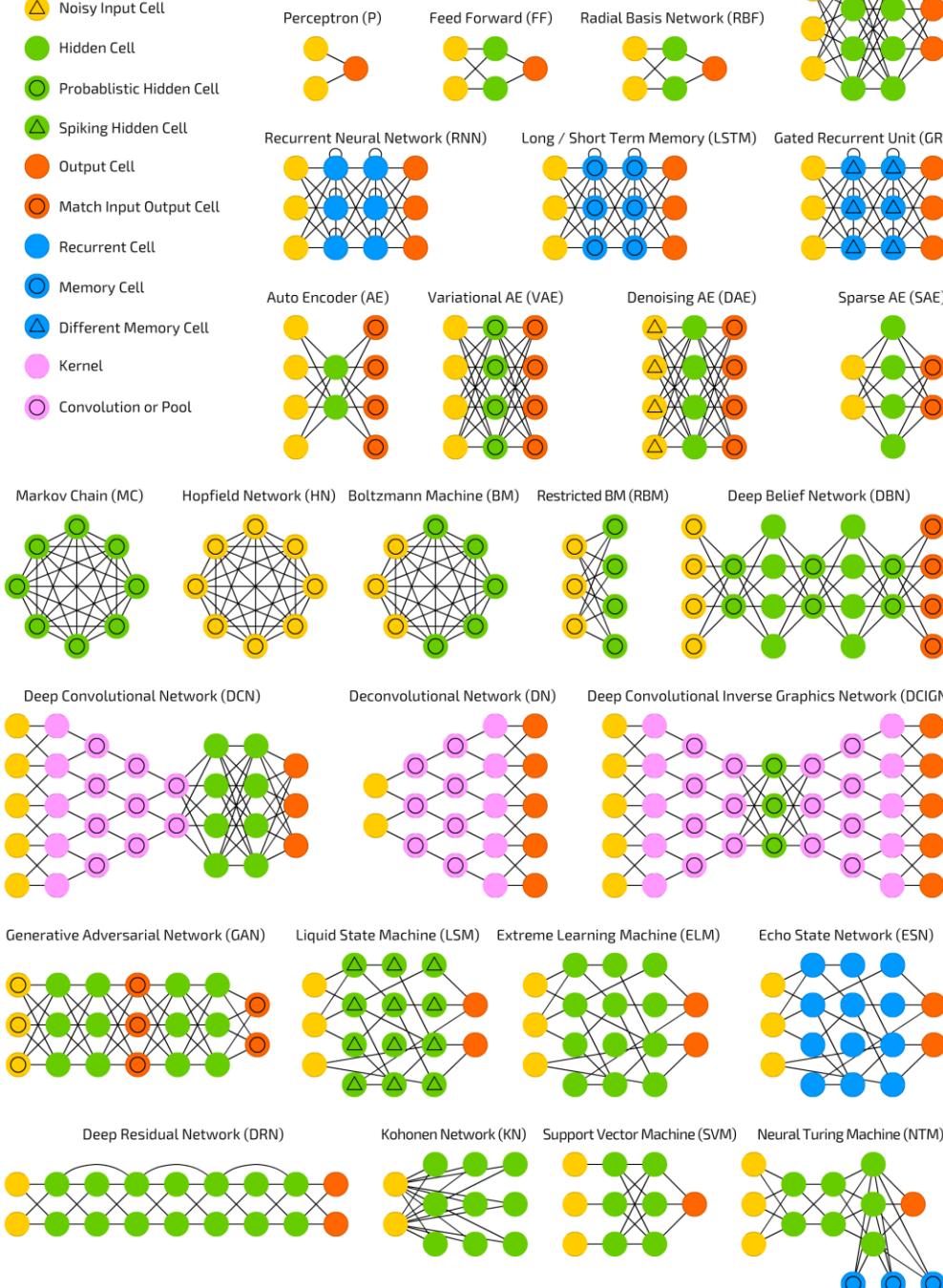
- Método altamente práctico y general para aprender funciones continuas y discretas.
- Brillan en la presencia de grandes volúmenes de datos.
- Existen redes para prácticamente cualquier problema.
- Difíciles de entrenar, interpretabilidad es compleja (caja negra) y sufren de serios problemas de sobreajuste.



Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

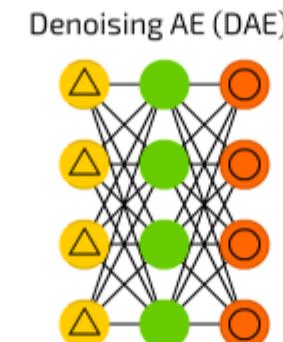
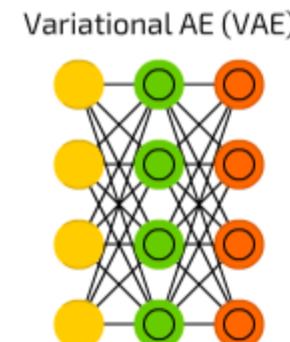
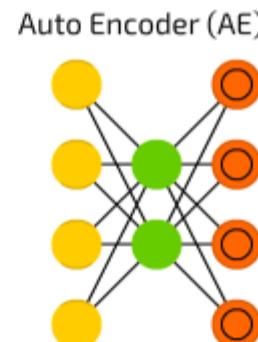
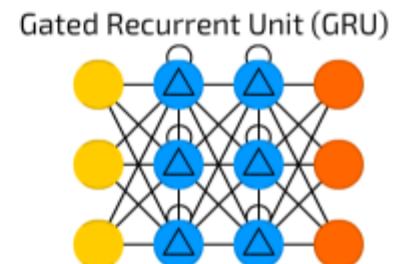
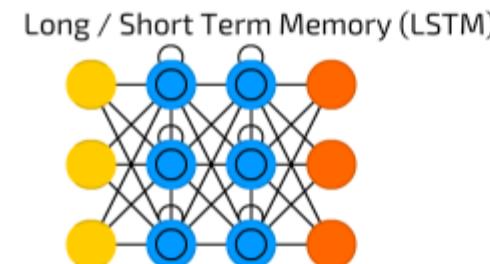
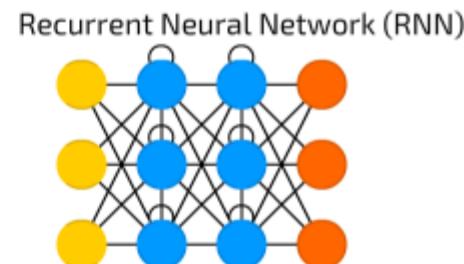
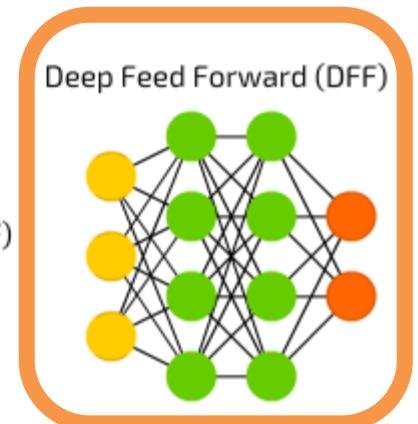
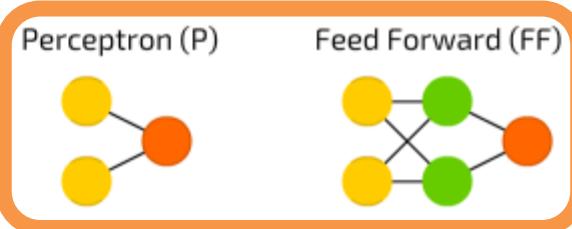


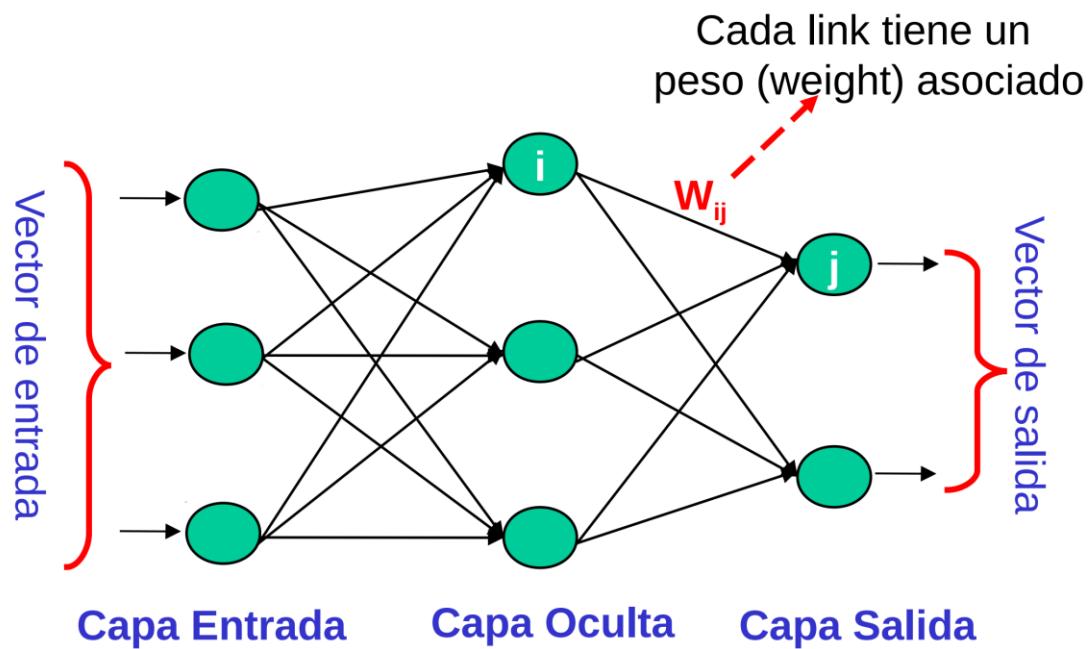
A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- (○) Backfed Input Cell
- (●) Input Cell
- (△) Noisy Input Cell
- (●) Hidden Cell
- (○) Probabilistic Hidden Cell
- (△) Spiking Hidden Cell
- (●) Output Cell
- (○) Match Input Output Cell
- (●) Recurrent Cell
- (○) Memory Cell
- (△) Different Memory Cell
- (●) Kernel
- (○) Convolution or Pool

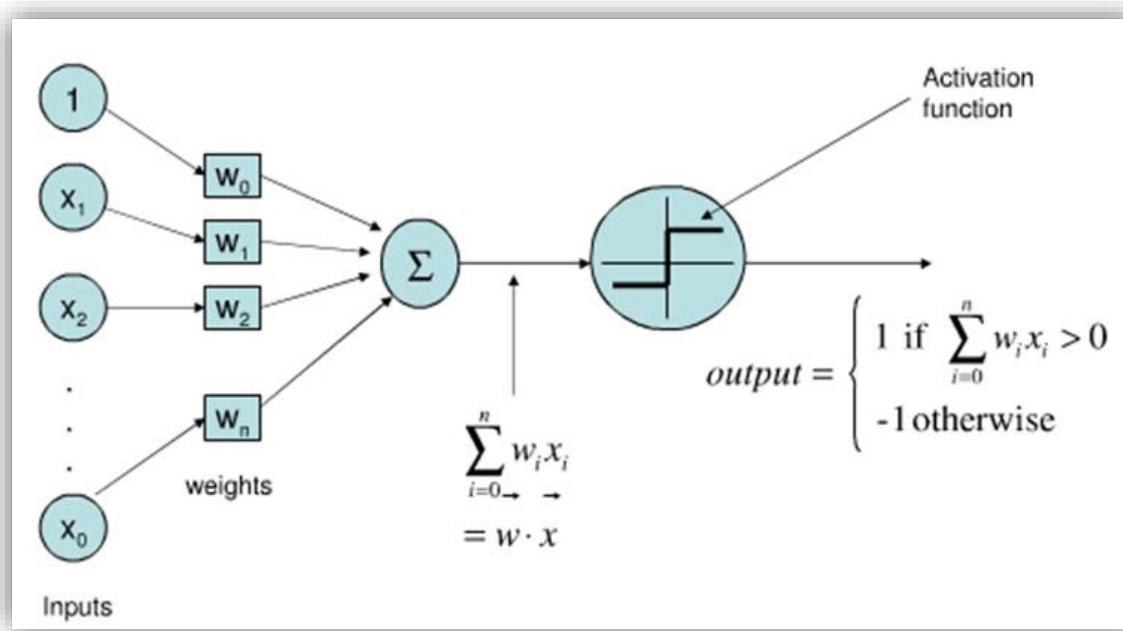




Una ANN es caracterizada por:

1. Estructura de la red, i.e., número de unidades (neuronas en cada) en cada capa y número de capas ocultas.
2. Método para determinar el valor de los pesos w_{ij} que conectan a una neurona i con una neurona j .
3. Función de activación que modula el impulso que envía una neurona a otra.

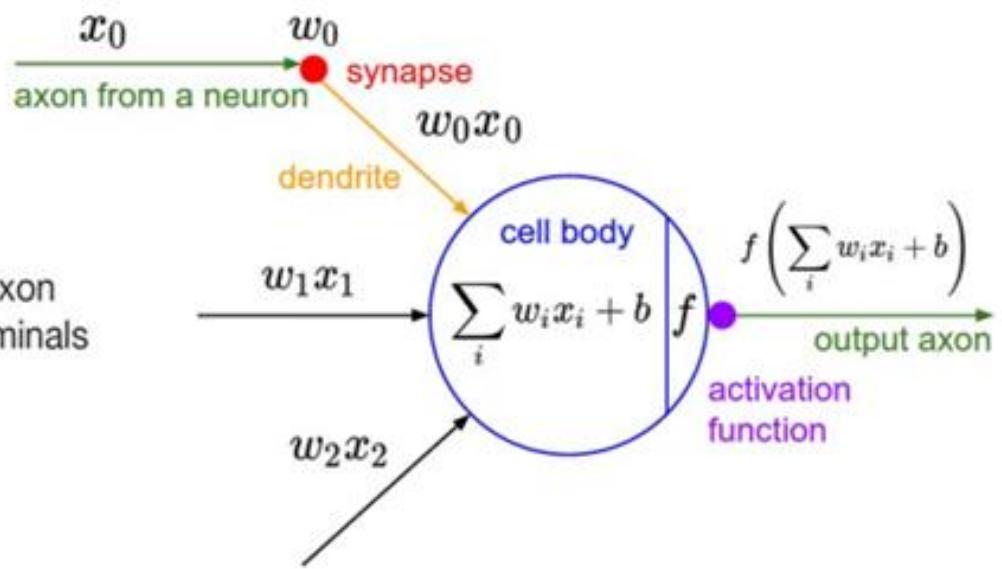
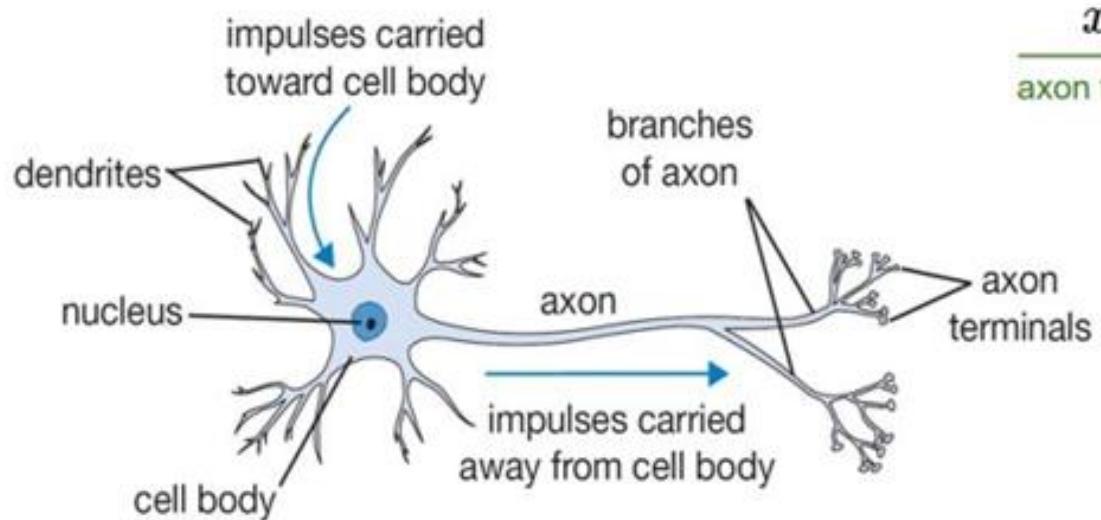
La unidad fundamental de una red neuronal se conoce como **perceptrón**





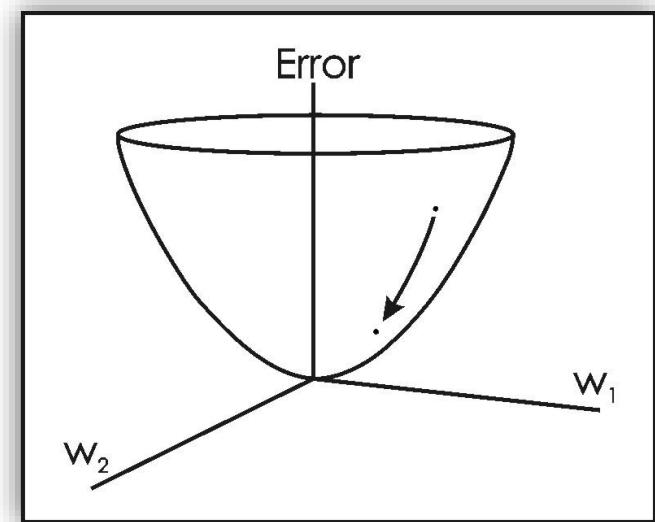
1958 Perceptron





¿Cómo podemos entrenar un perceptrón (lineal)?

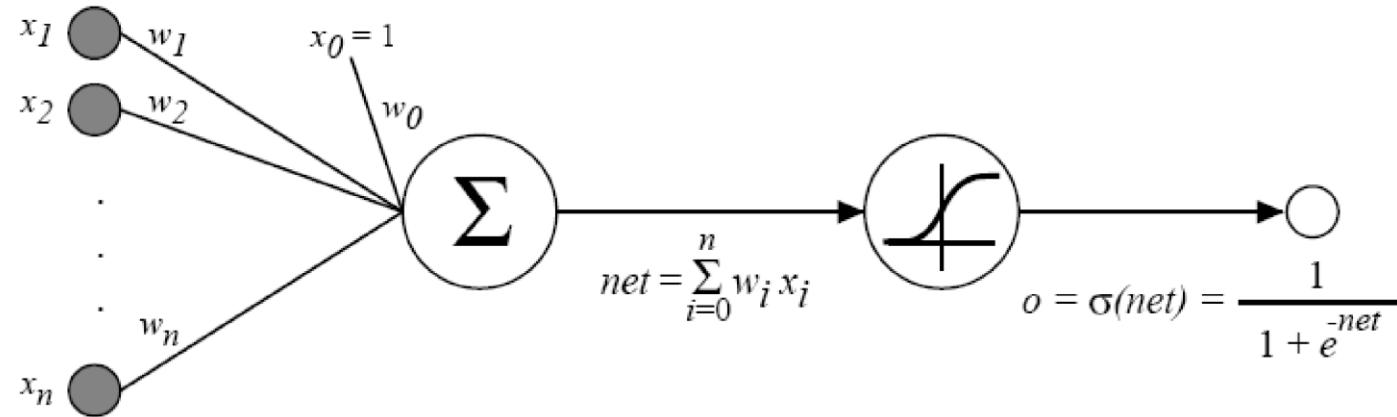
1. ¿Qué es lo que queremos hacer?
2. ¿Cuál podría ser una buena función objetivo para el problema?
3. ¿Qué características tiene esta función objetivo?
4. ¿Qué algoritmo de optimización podemos utilizar para este problema?



El algoritmo de entrenamiento de un perceptrón lineal es igual al que vimos para la regresión lineal

- Inicializar cada peso w_i a un valor aleatorio pequeño (ej. rango [0,1])
- **do**
 - Inicializar gradientes Δw_i a cero
 - **for** (x,y) en set de entrenamiento
 - Aplique (x,y) a perceptron y obtenga salida **out**
 - Para cada peso calcule gradiente y guarde en acumulador:
$$\Delta w_i \leftarrow \Delta w_i + \eta(y - \text{out}) x_i$$
 - end for**
 - Actualice valor de cada peso:
$$w_i^{\text{new}} = w_i^{\text{old}} + \Delta w_i$$
- **while** (condición de término != TRUE).

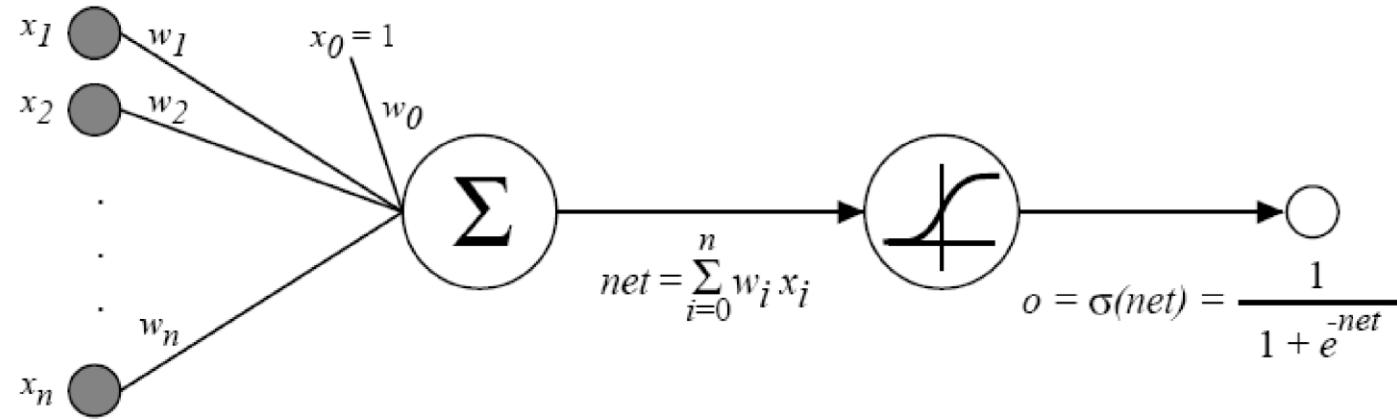
¿Cómo quedaría entonces la regla para un perceptrón **sigmoidal**?



Regla de entrenamiento

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

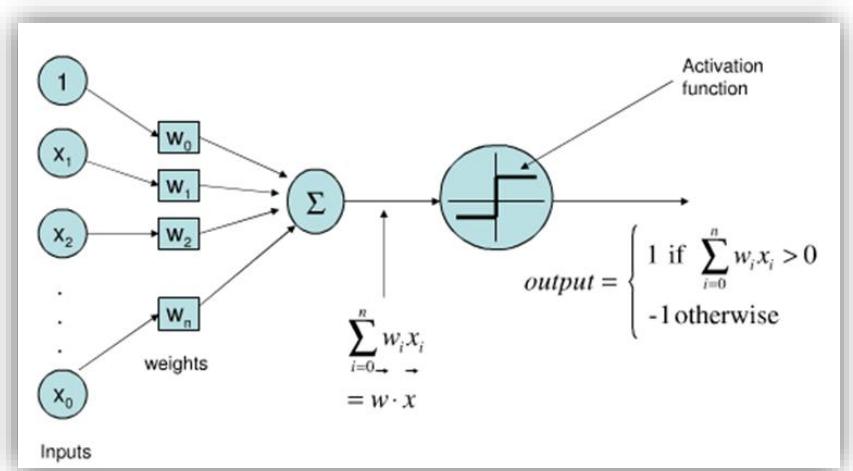
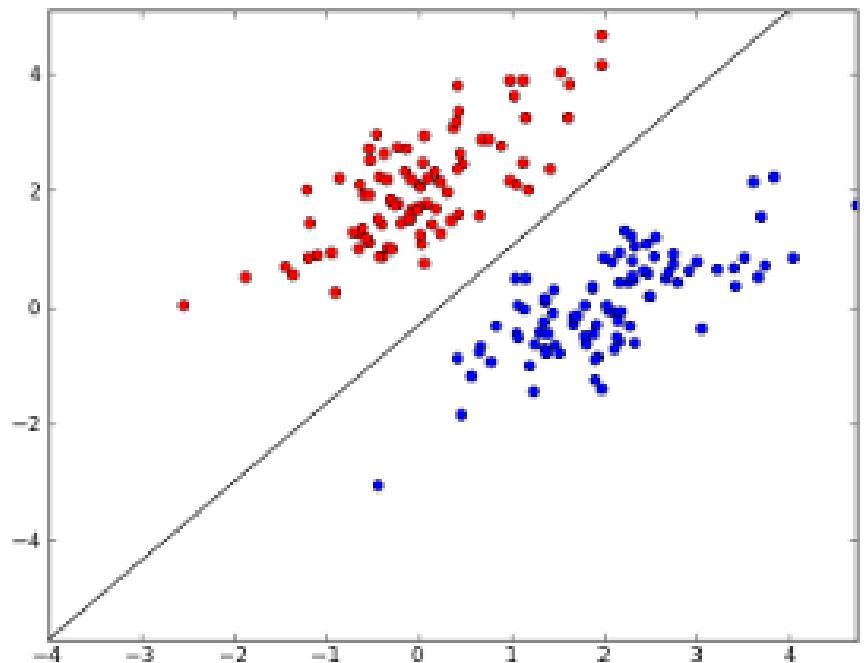
¿Cómo quedaría entonces la regla para un perceptrón **sigmoidal**?



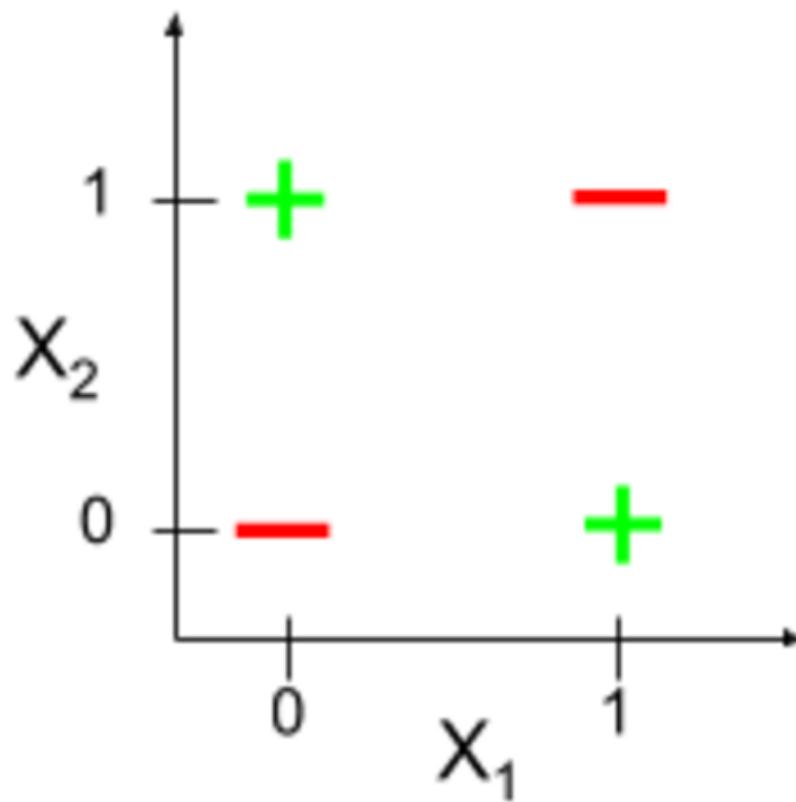
Regla de entrenamiento

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

$$\frac{\partial E}{\partial w_i} = - \sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}$$



Perceptrón puede clasificar problemas linealmente separables



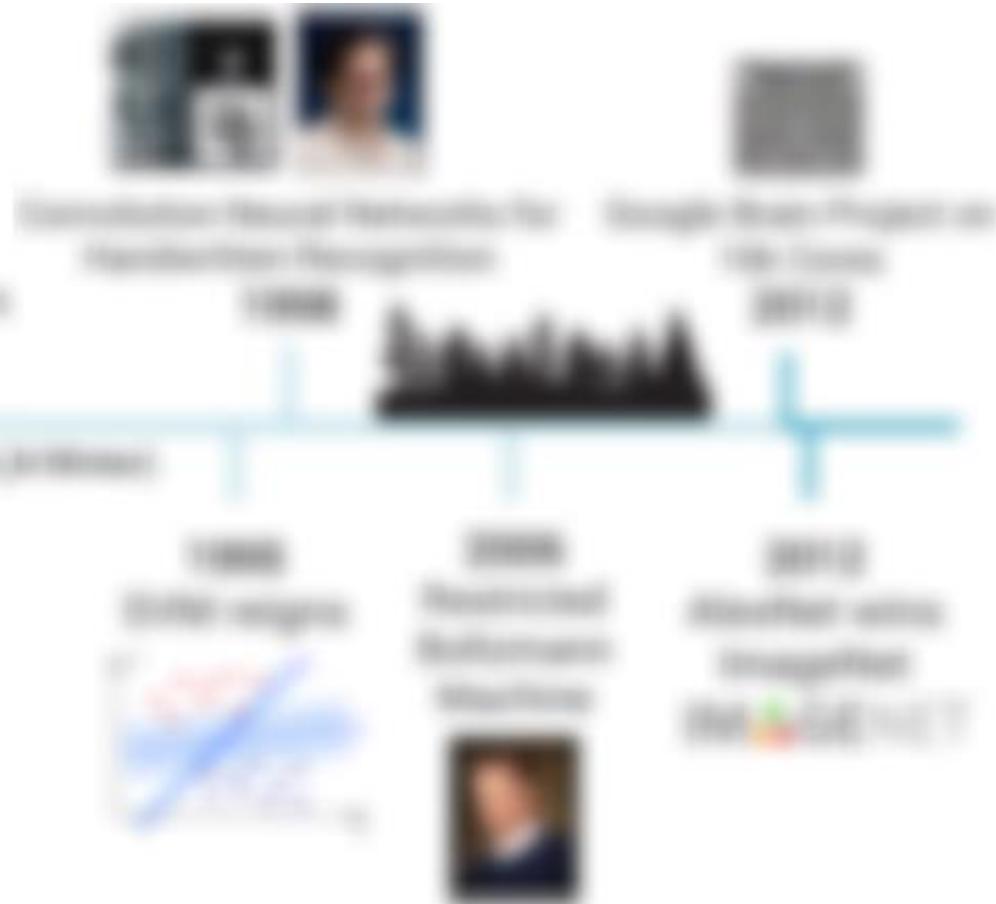
¿Cómo podríamos extender al perceptrón, para que sea capaz de resolver este tipo de problemas (**no linealmente separables**)?



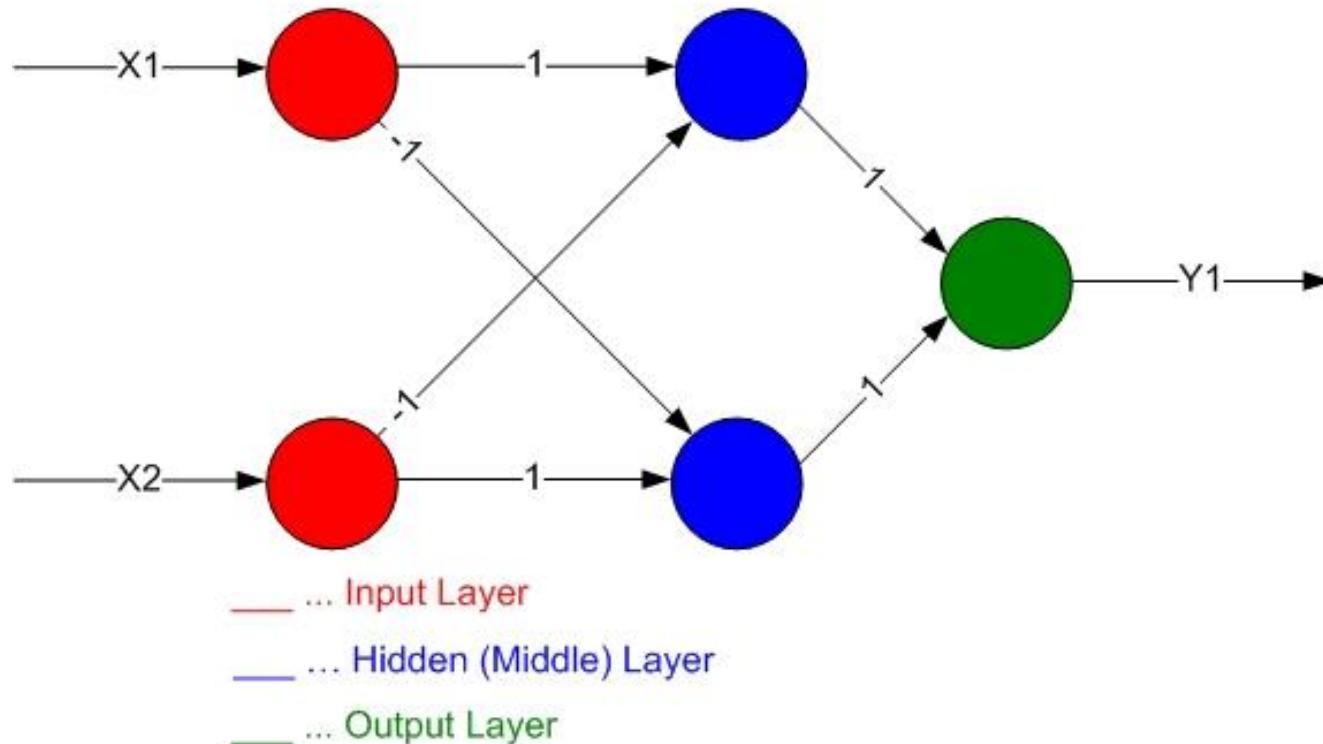
1958 Perceptron



1969
Perceptron criticized

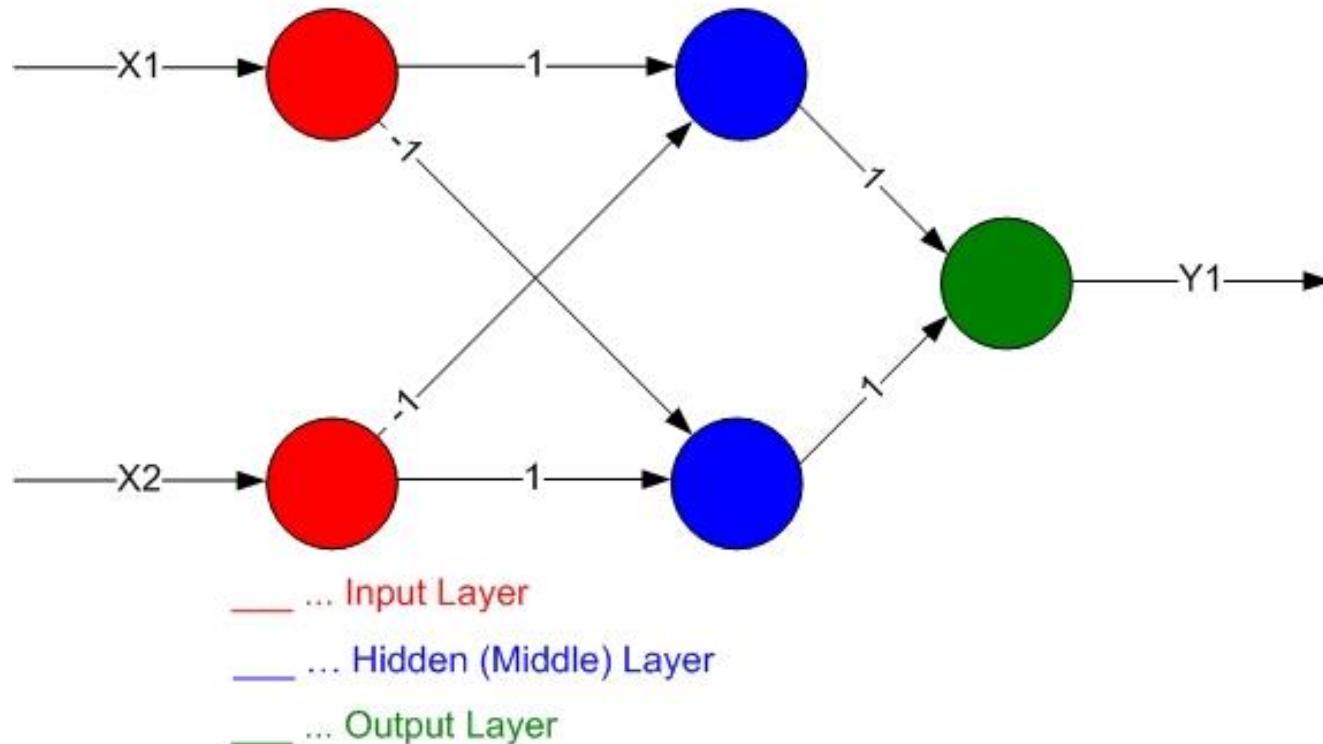


$$Y_1 = \text{XOR}(X_1, X_2)$$



Al agregar una nueva capa (oculta), podemos representar conceptos intermedios entre el input y el output.

$$Y_1 = \text{XOR}(X_1, X_2)$$



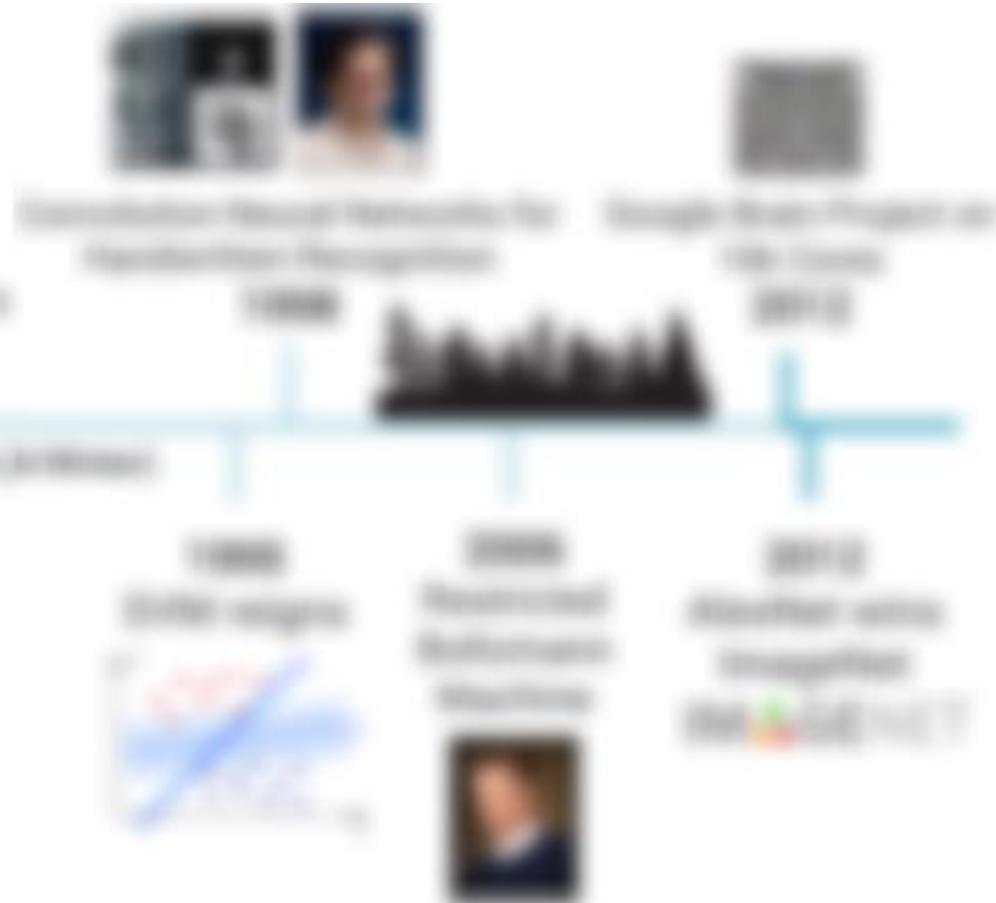
¿Cómo podemos **aprender** estos nuevos conceptos de nivel **intermedios**?



1958 Perceptron



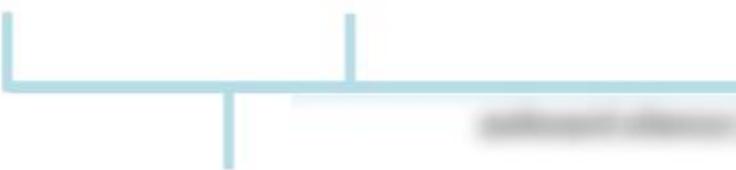
1969
Perceptron criticized



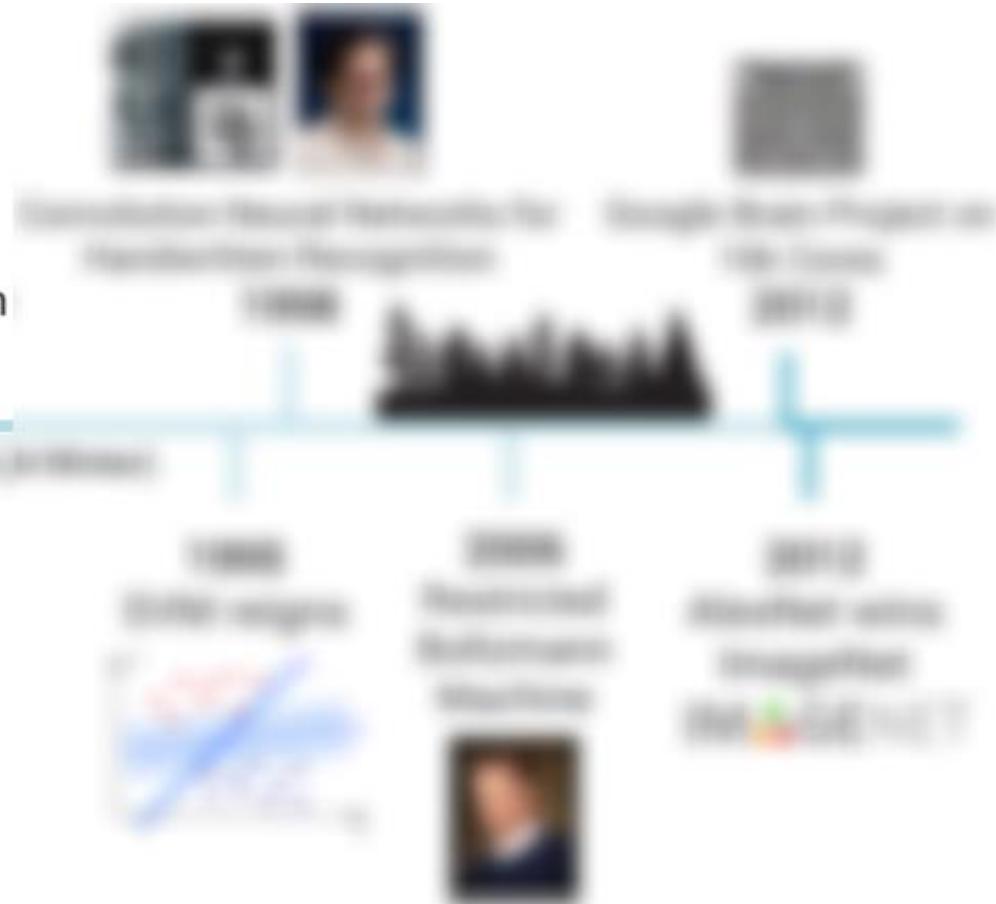


1958 Perceptron

1974 Backpropagation

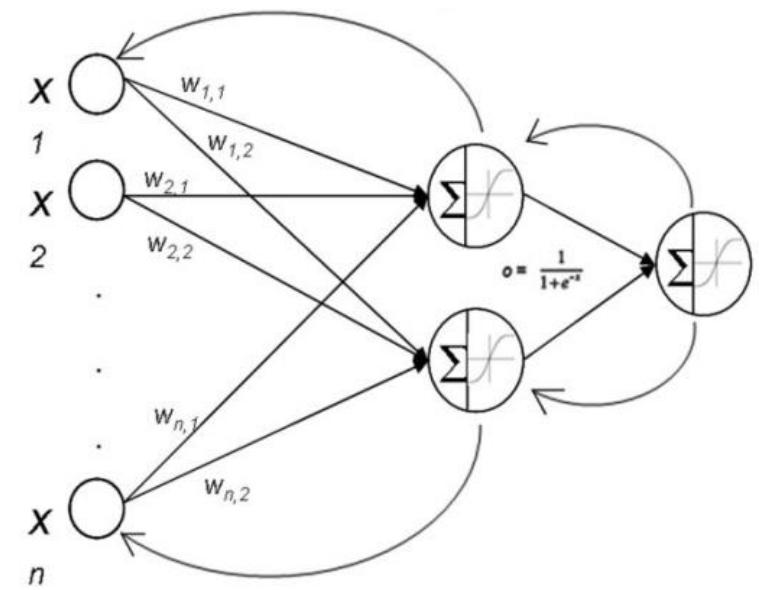


1969
Perceptron criticized

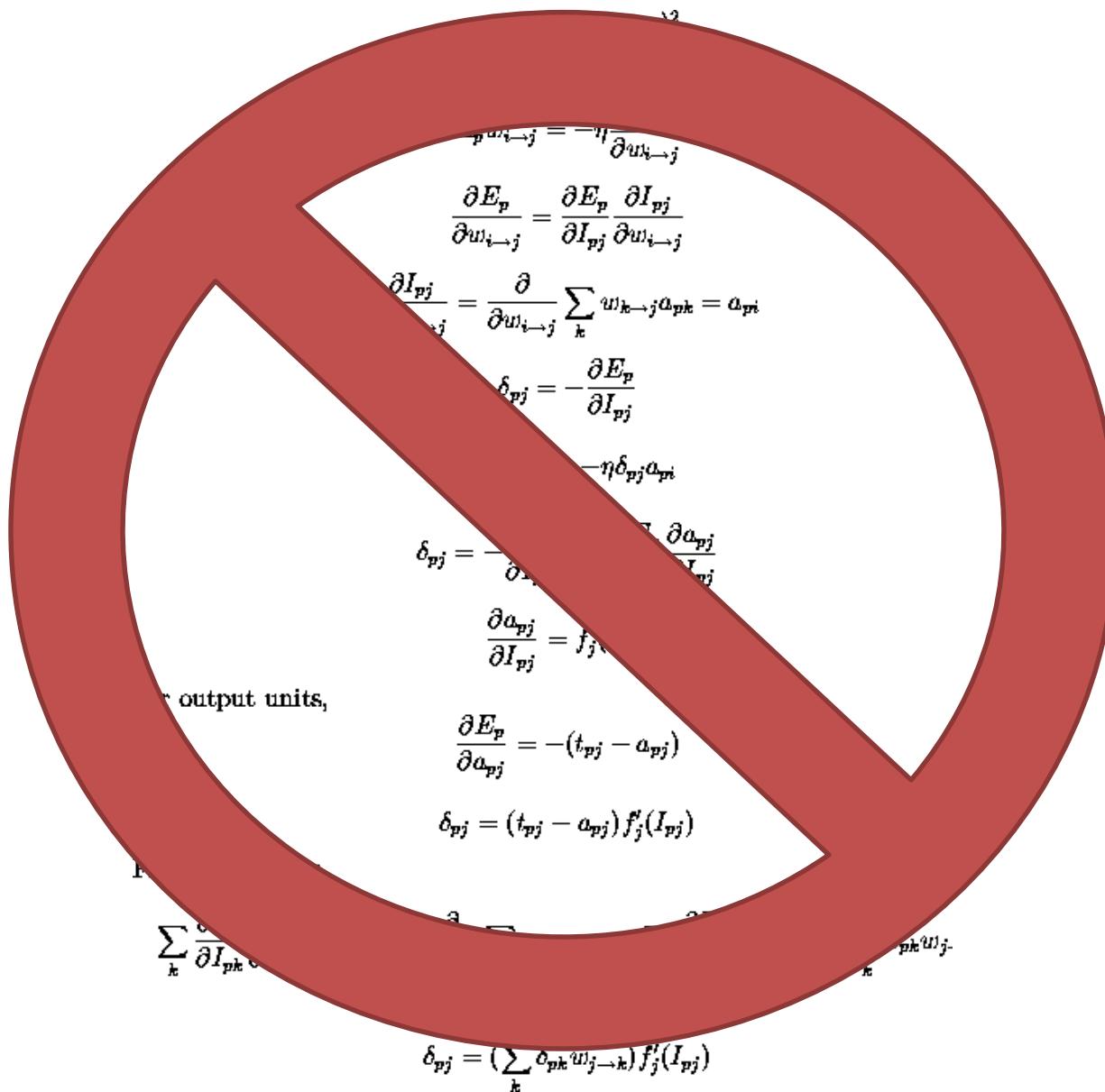


Backpropagation permite **propagar la señal del error** a todos los nodos de una **red** para ajustarlos

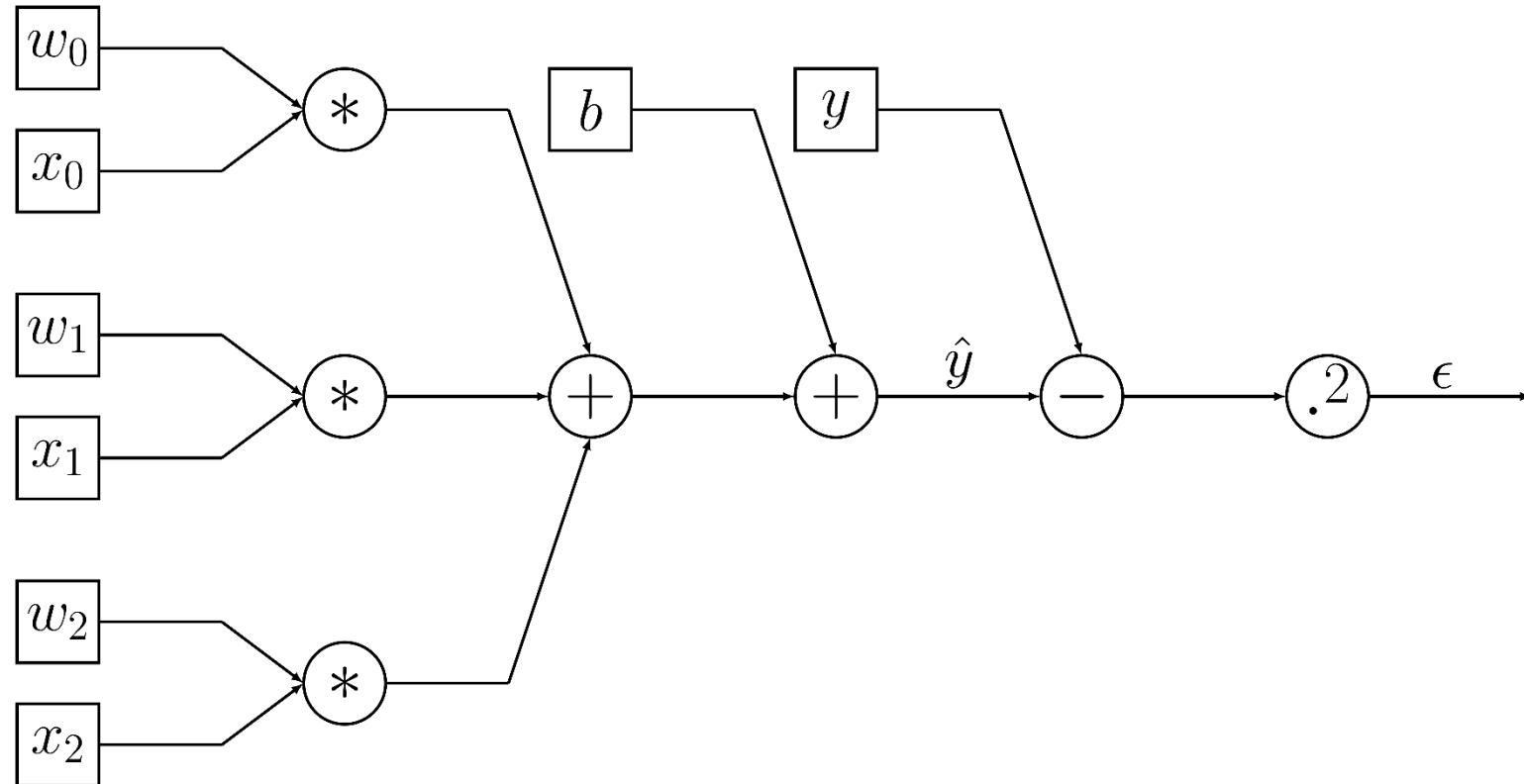
- No es otra cosa que la aplicación recursiva de la **regla de la cadena**.
- En conjunto con el método del descenso del gradiente, permite entrenar **redes de profundidad arbitraria** (pero no necesariamente con buen rendimiento).
- Aplicable a cualquier tipo de red y función diferenciable.
- Existen infinitos detalles en su ejecución, que hacen que la probabilidad de fracaso sea muy alta.



En este curso analizaremos *backpropagation* desde el punto de vista de los **grafos de cómputo**

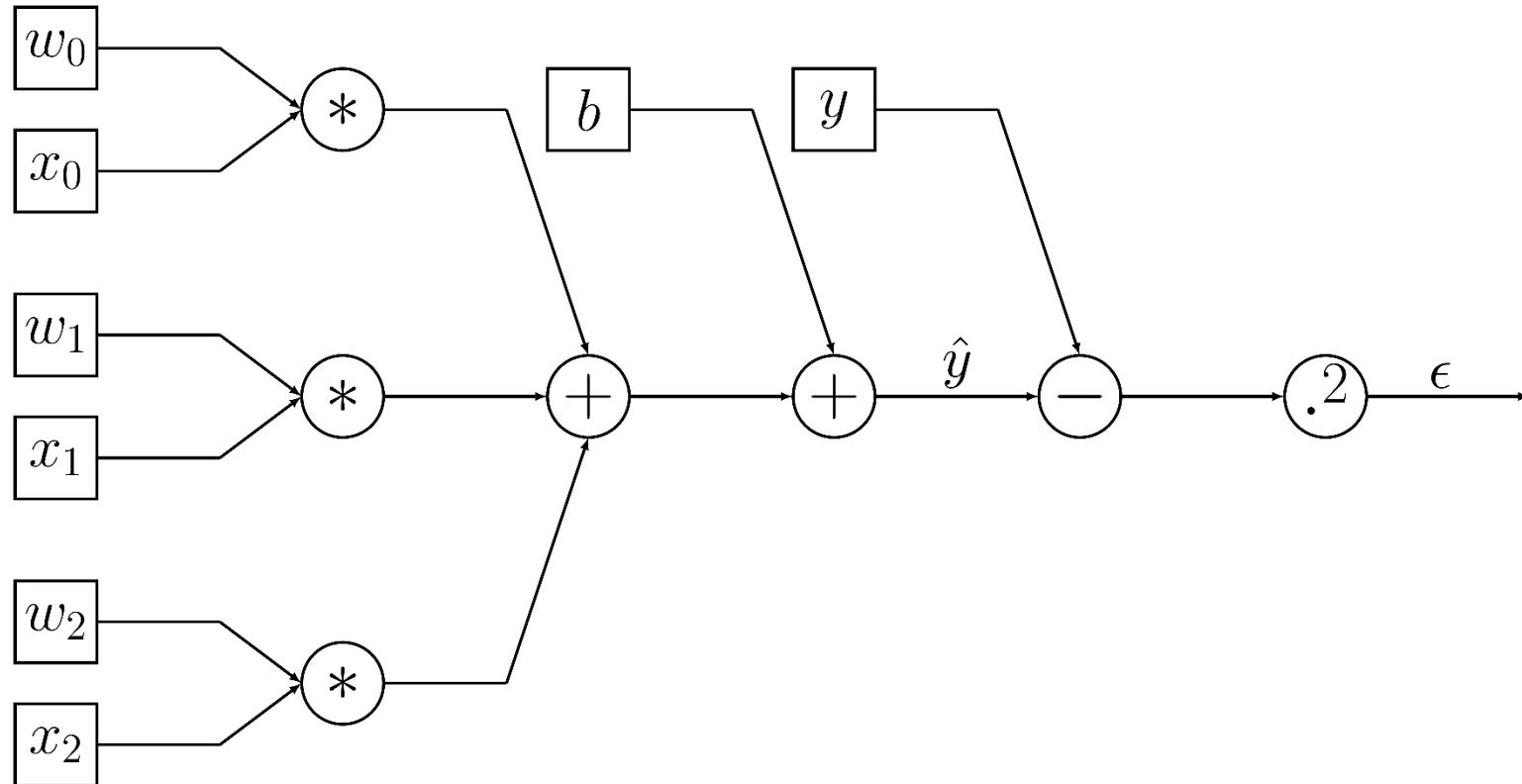


Un grafo de cómputo representa **todas las operaciones** realizadas hasta llegar al output

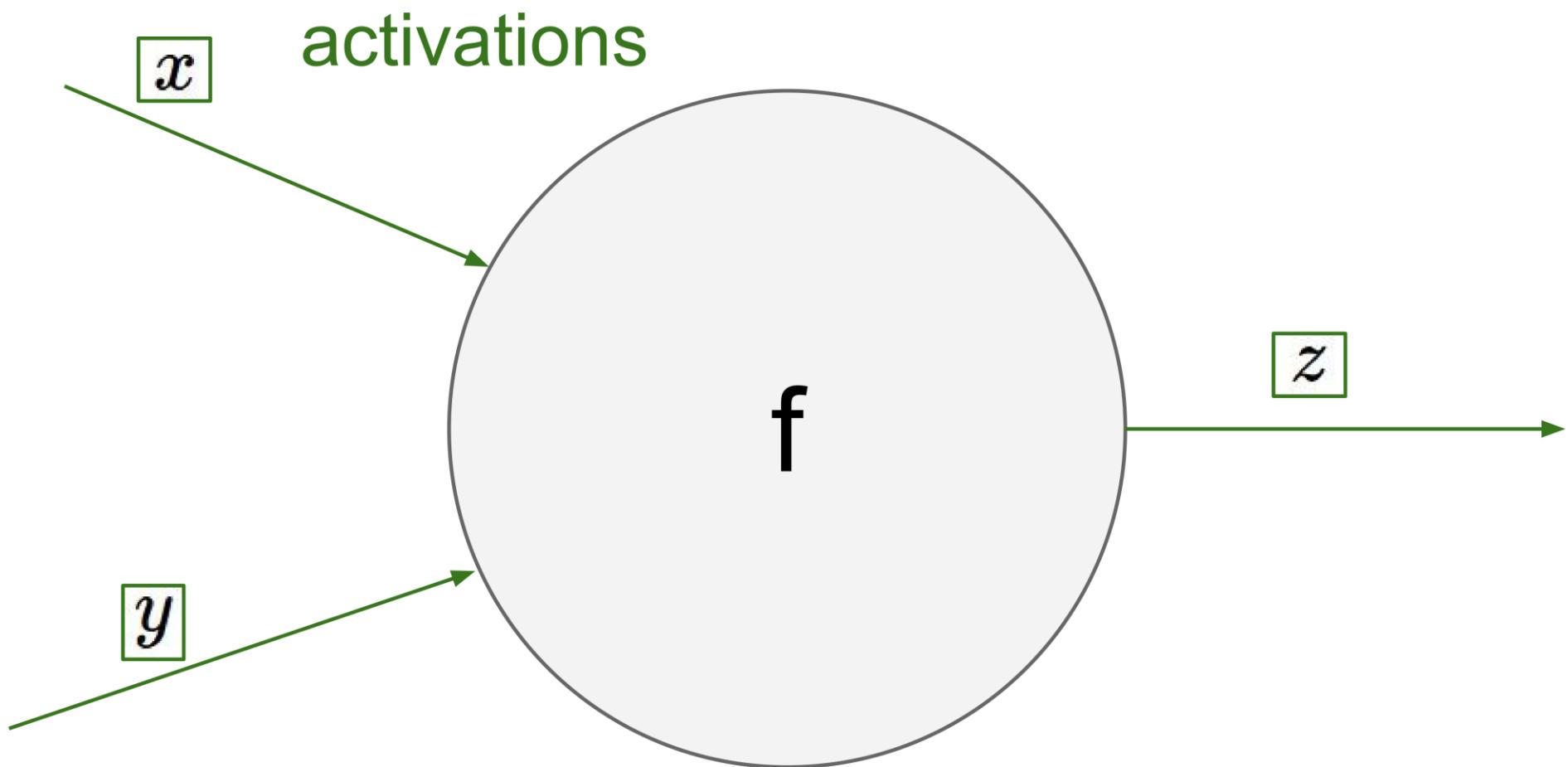


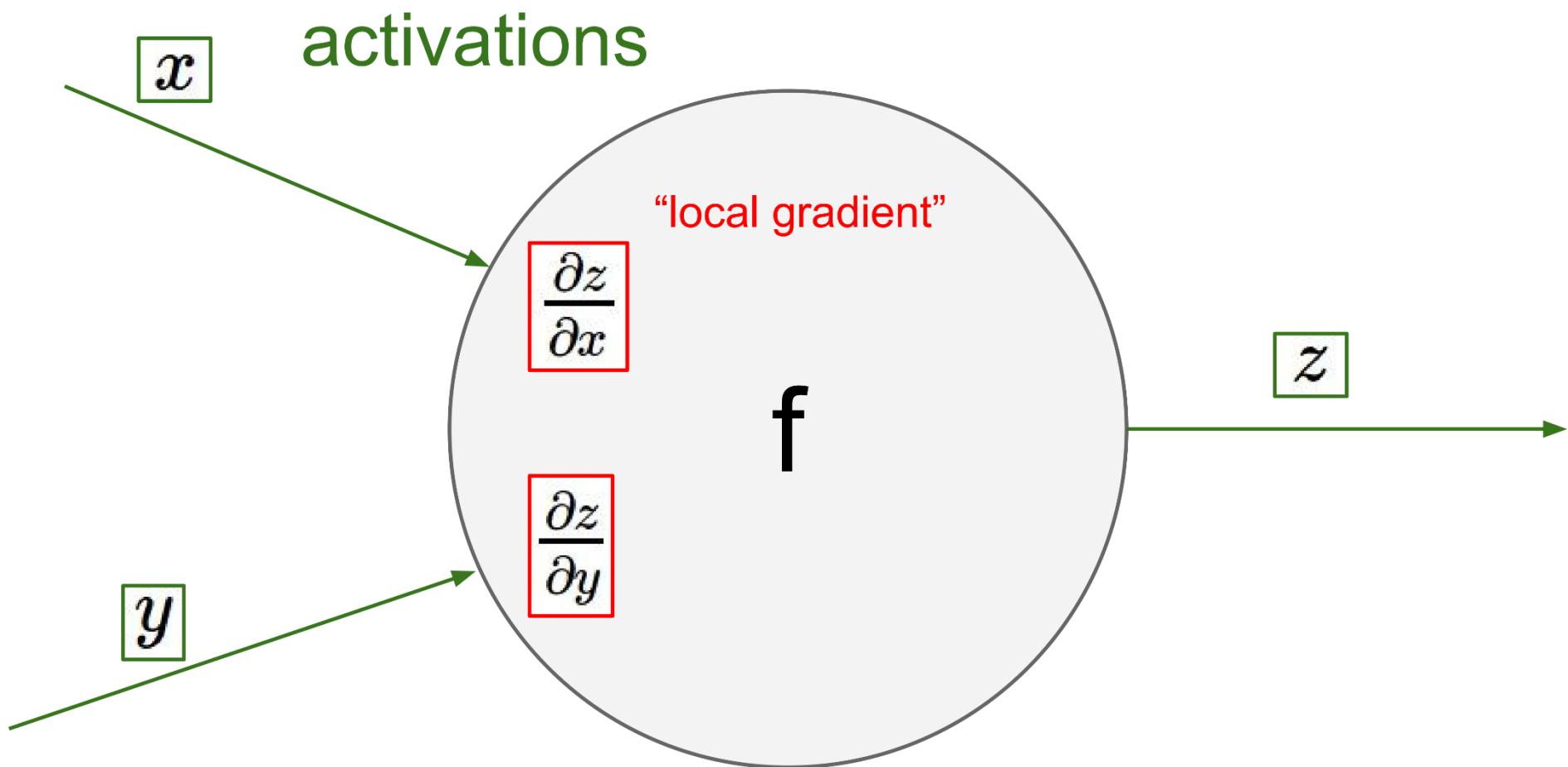
¿Qué función representa este grafo?

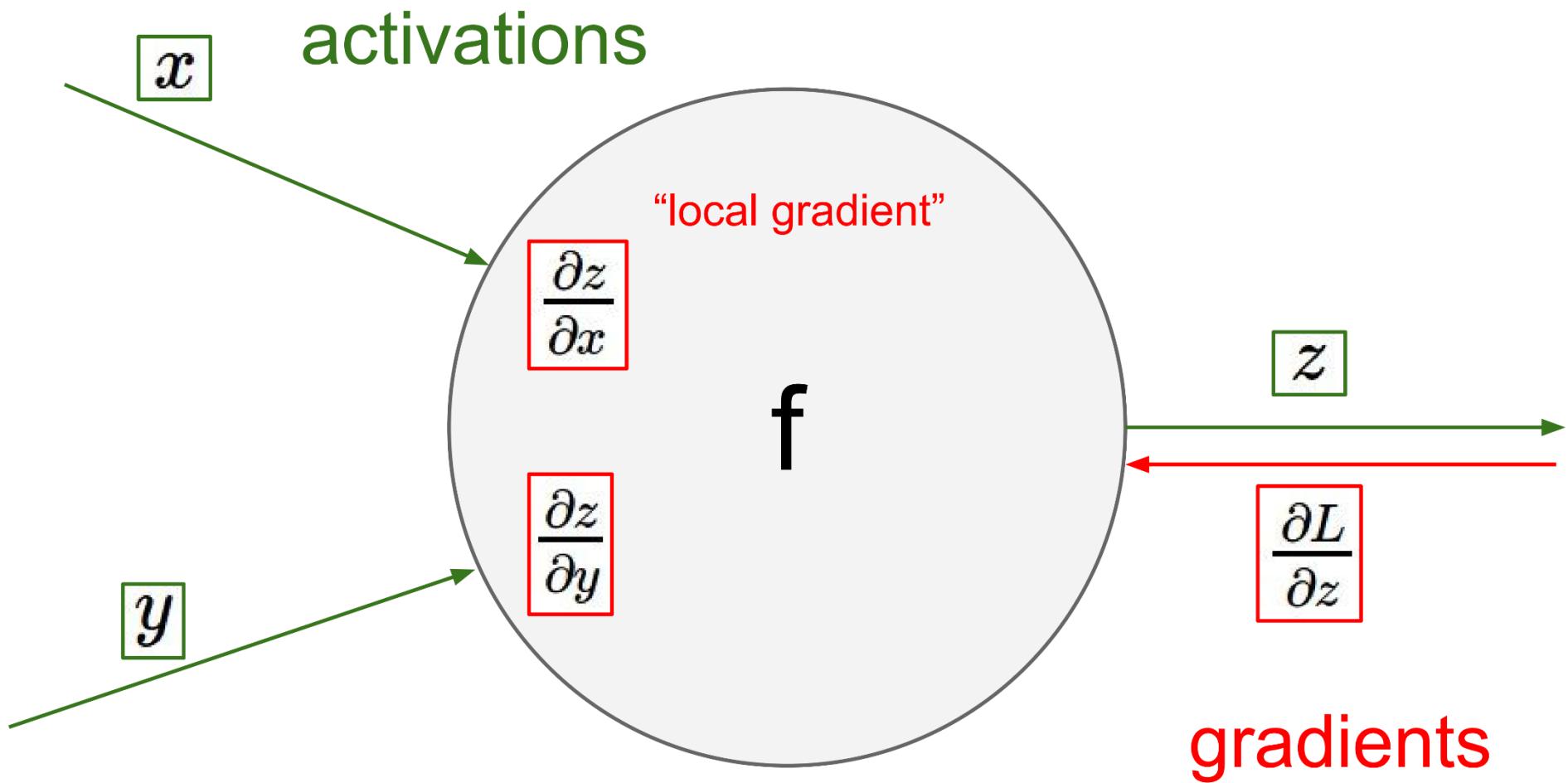
Podemos aprovechar esta **estructura** junto con la regla de la cadena, para obtener el gradiente completo

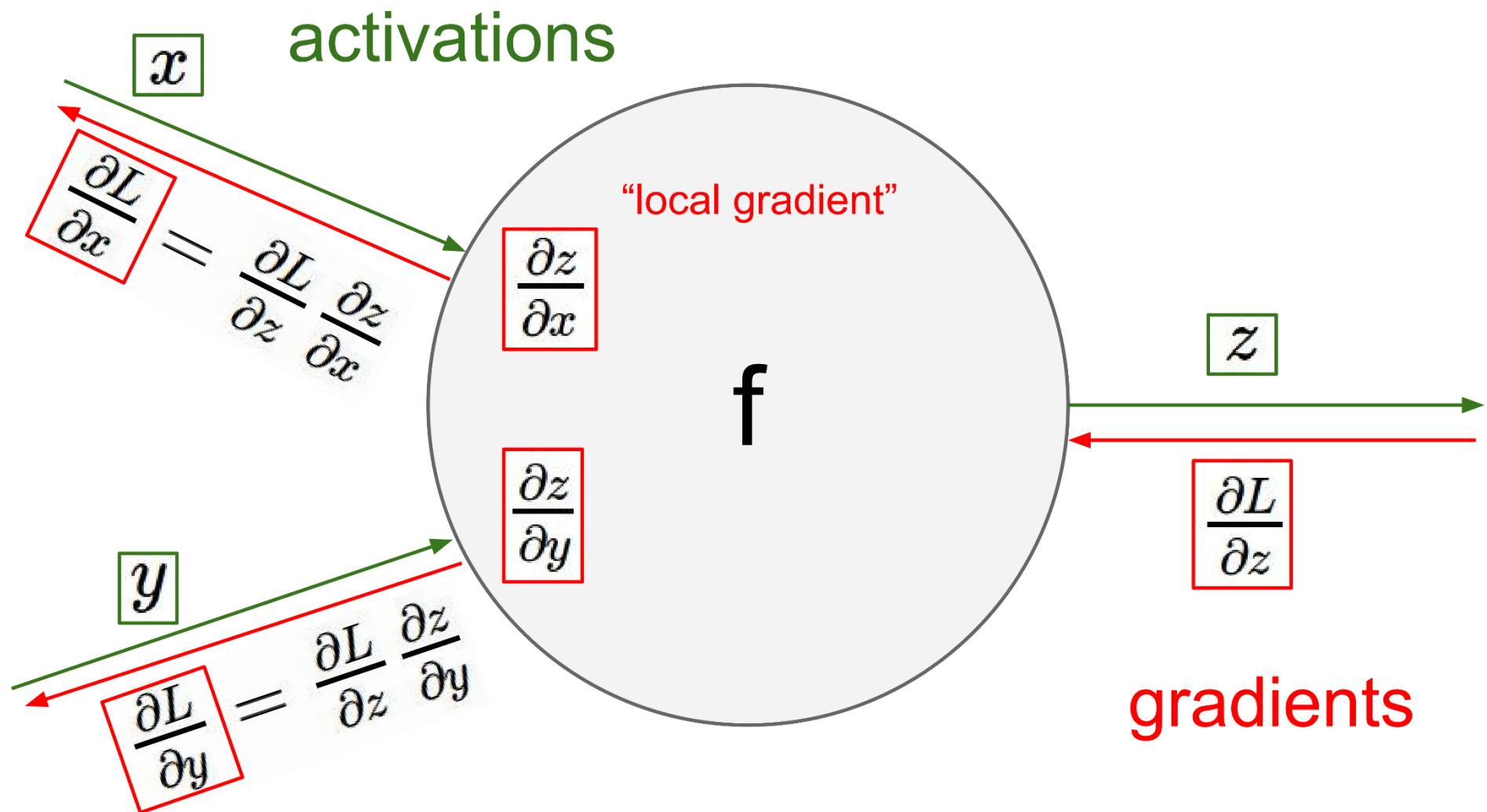


Basta con que sólo analicemos
las derivadas locales

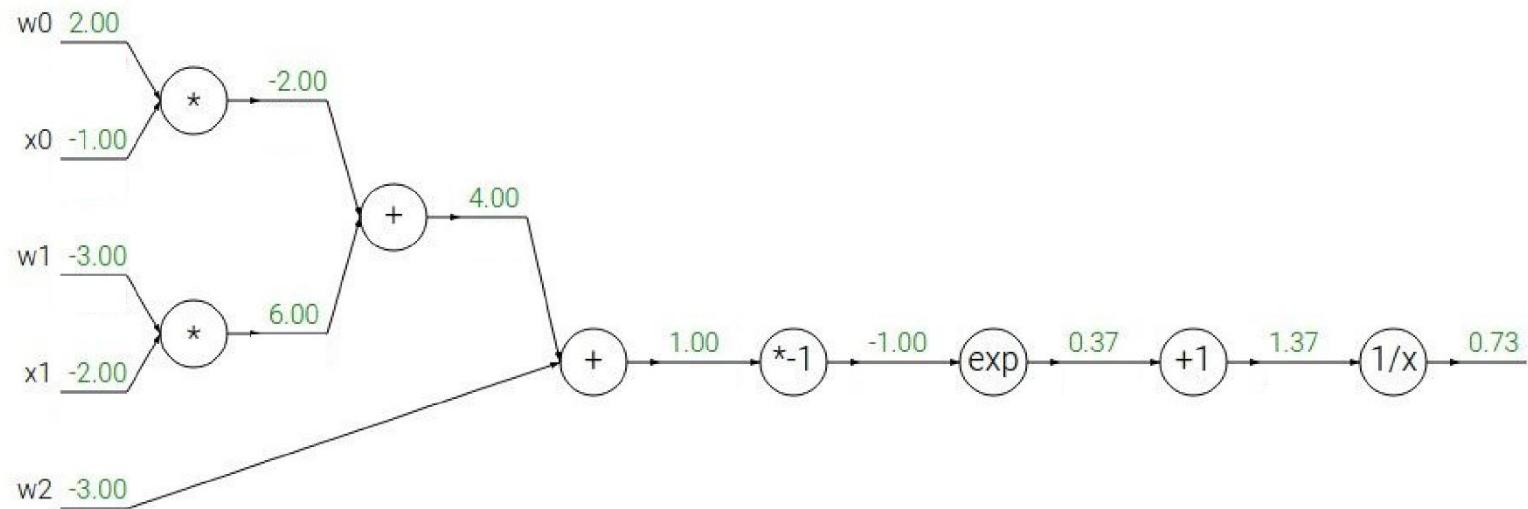




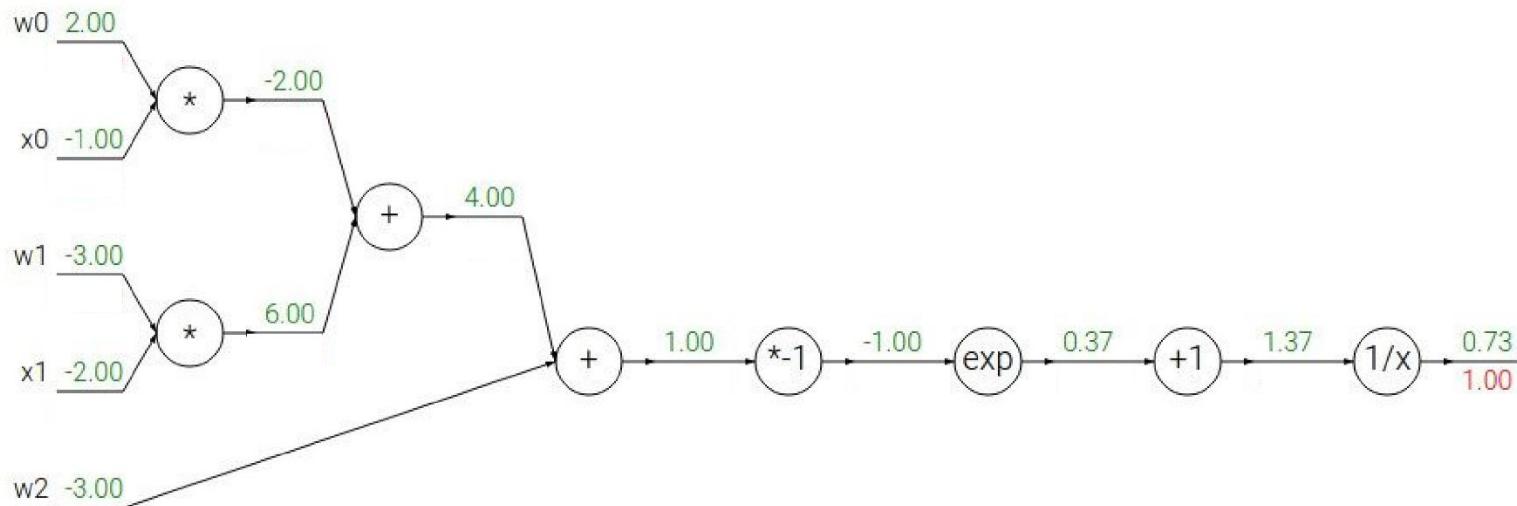




Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

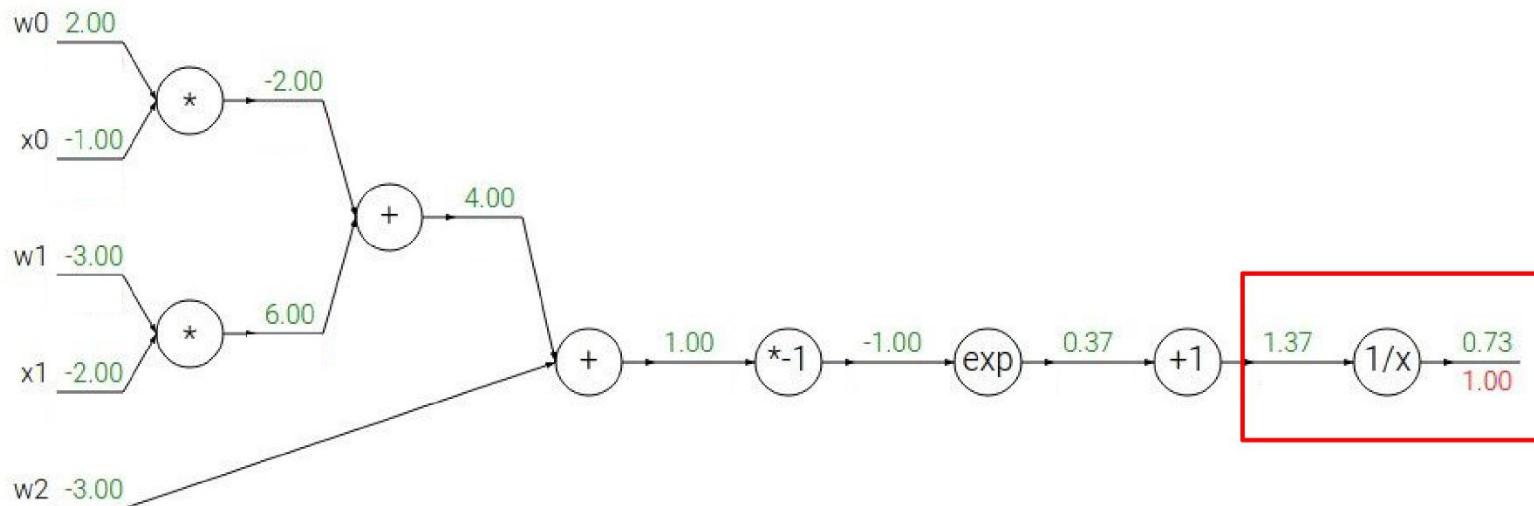
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

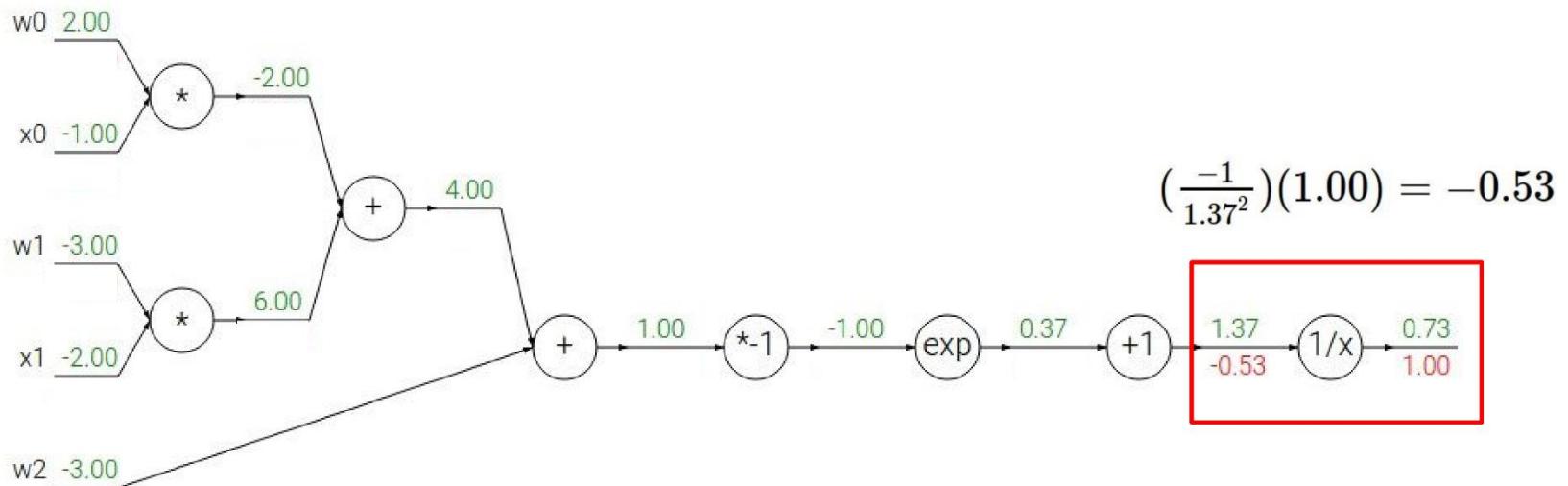
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

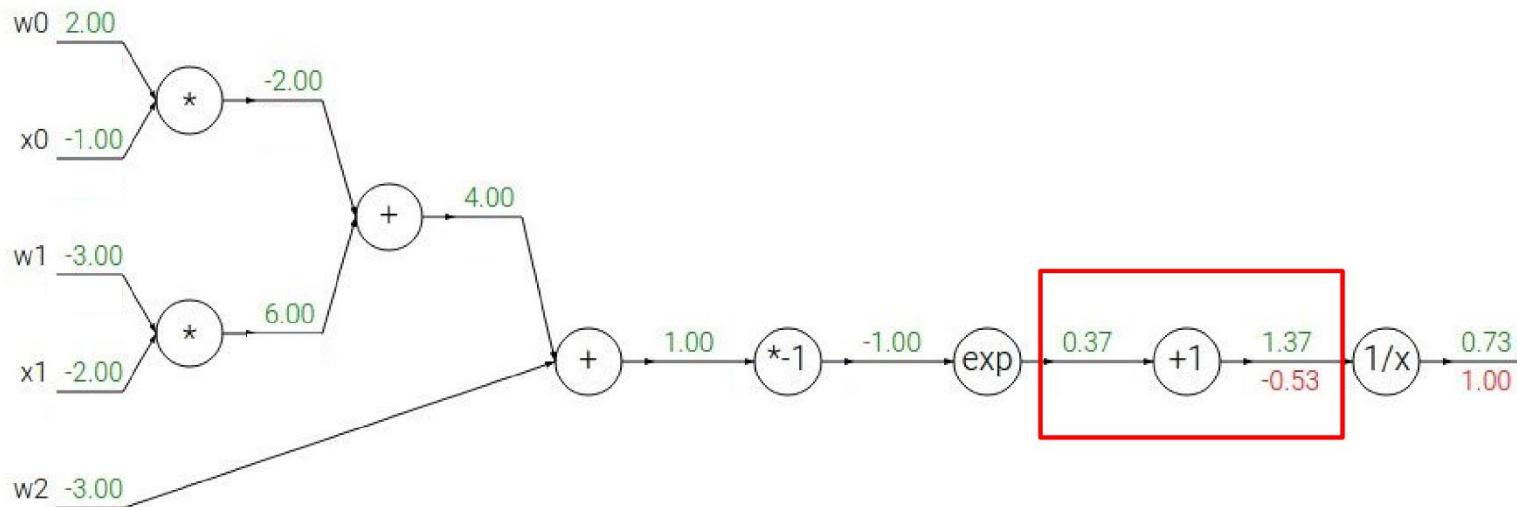
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

\rightarrow

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

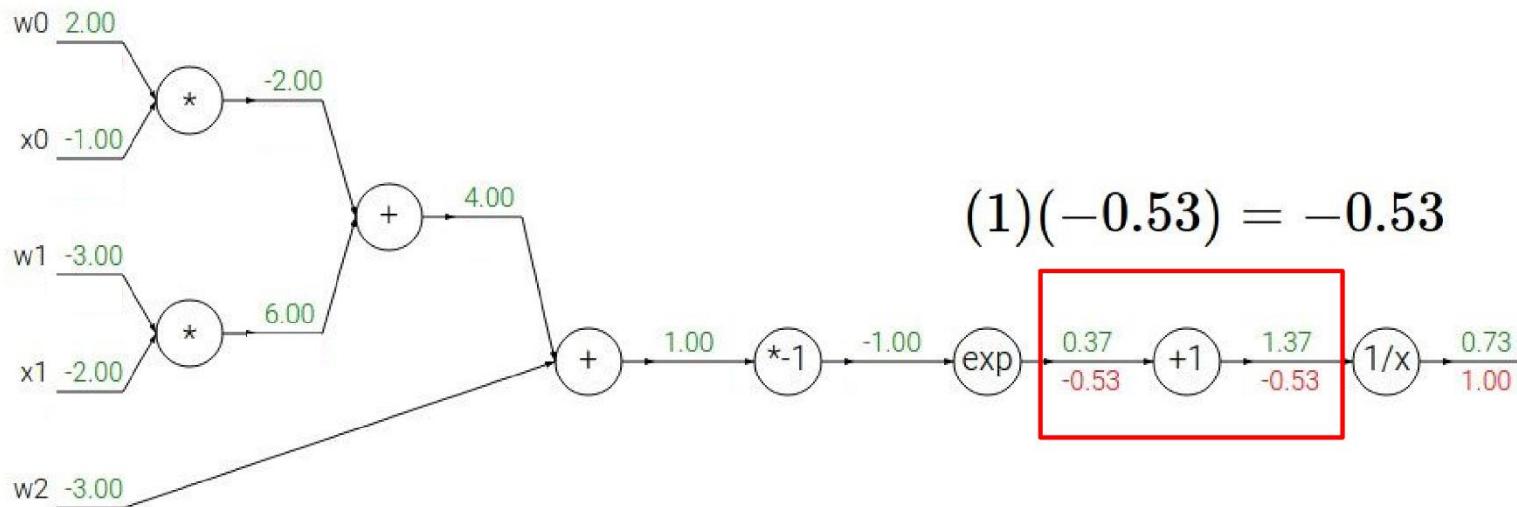
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

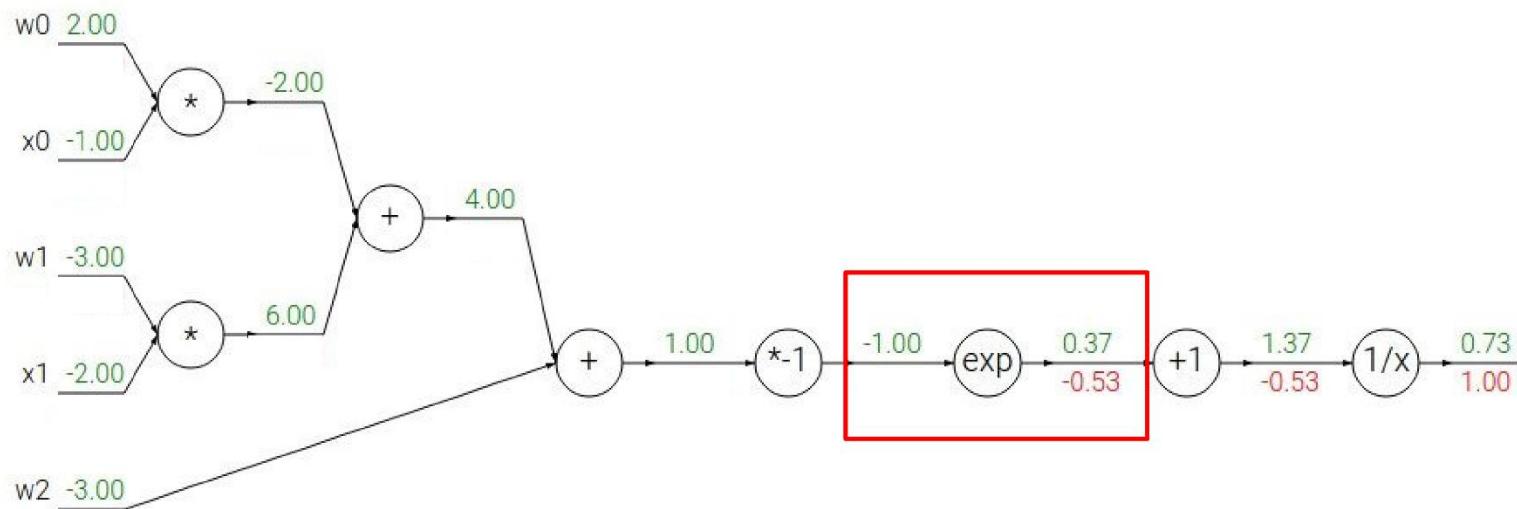
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

\rightarrow

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



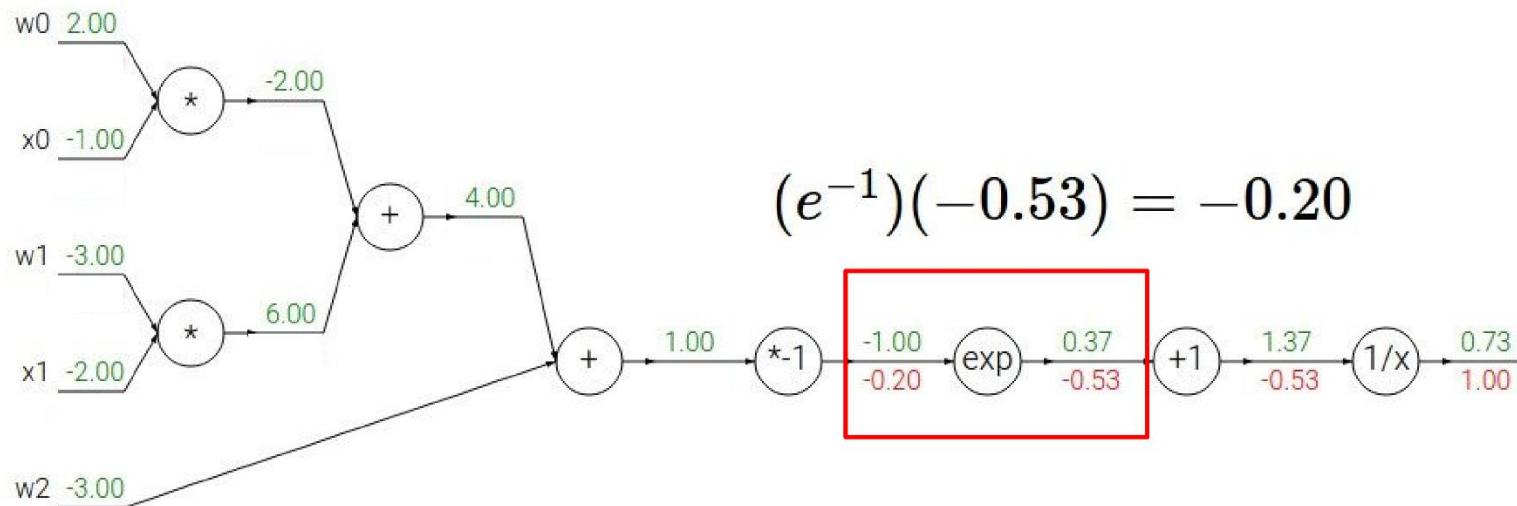
$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



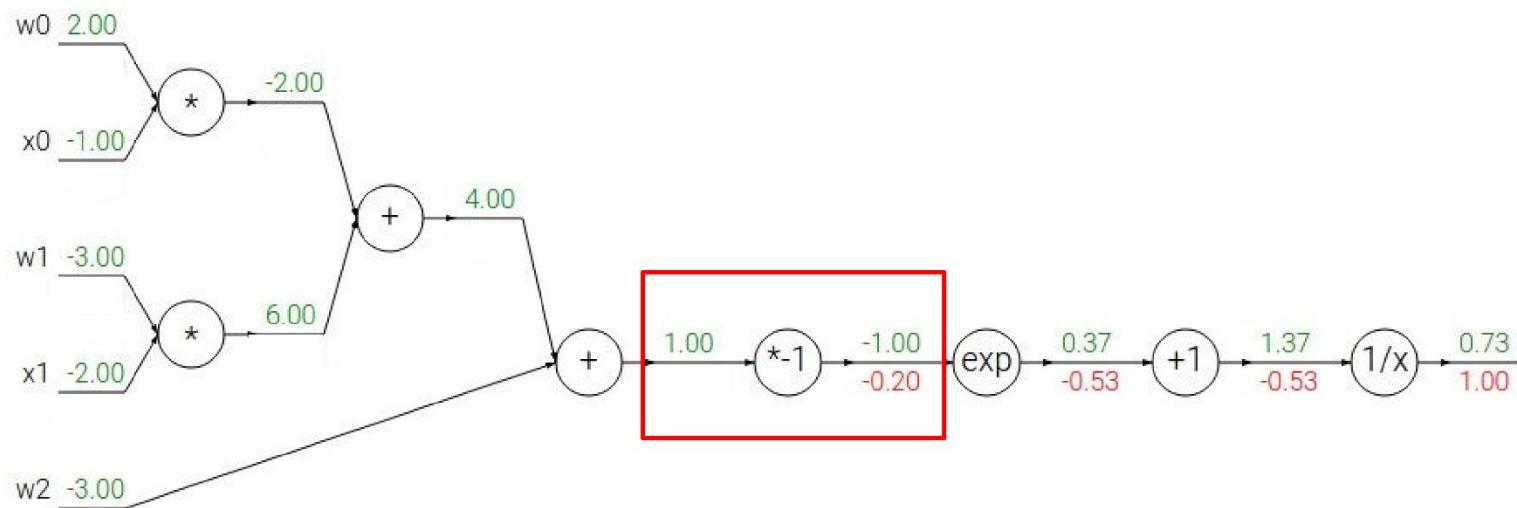
$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

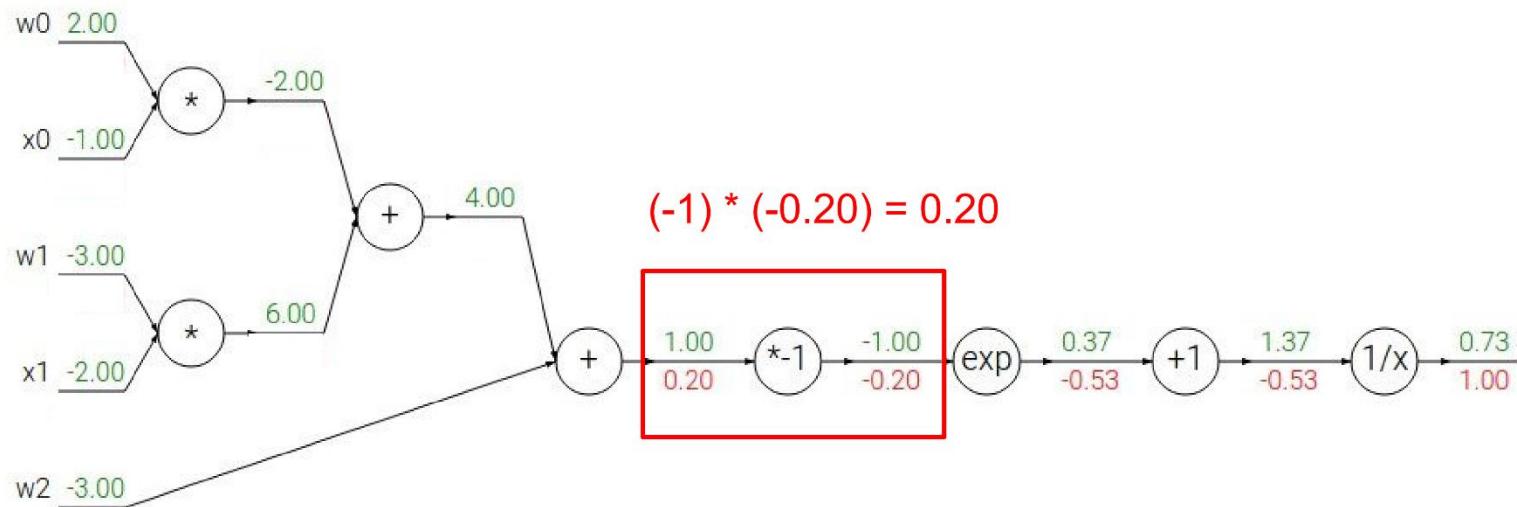
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

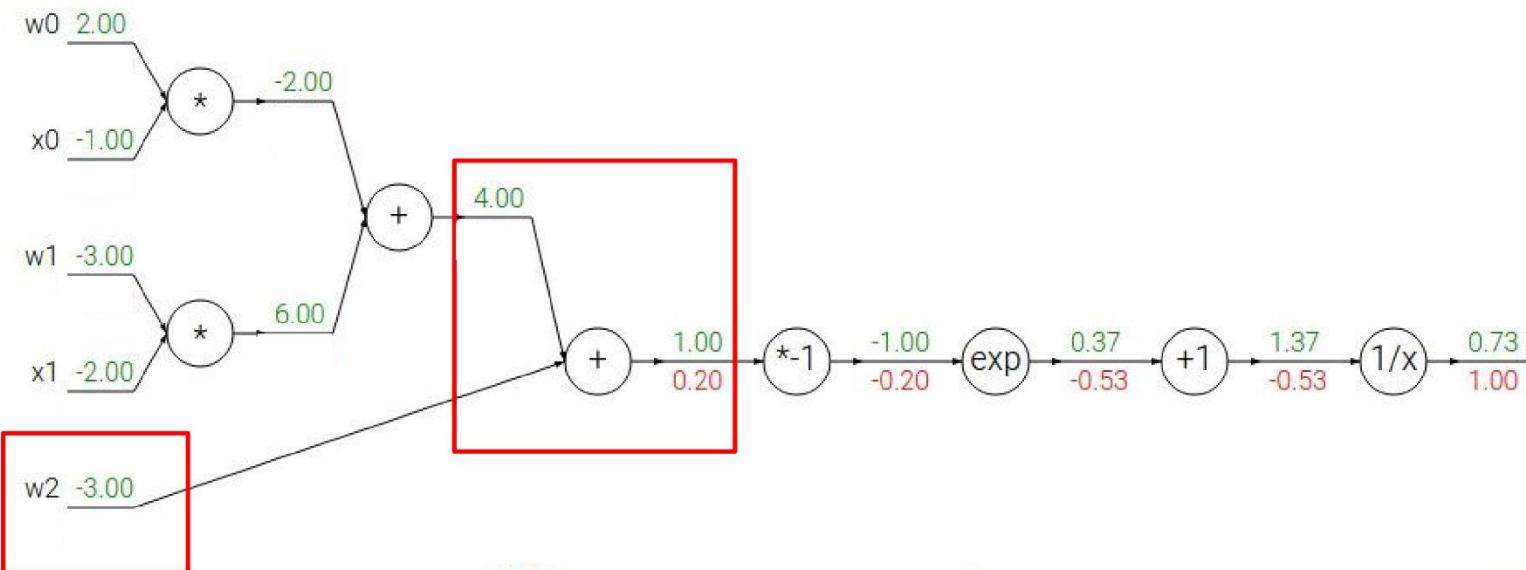
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

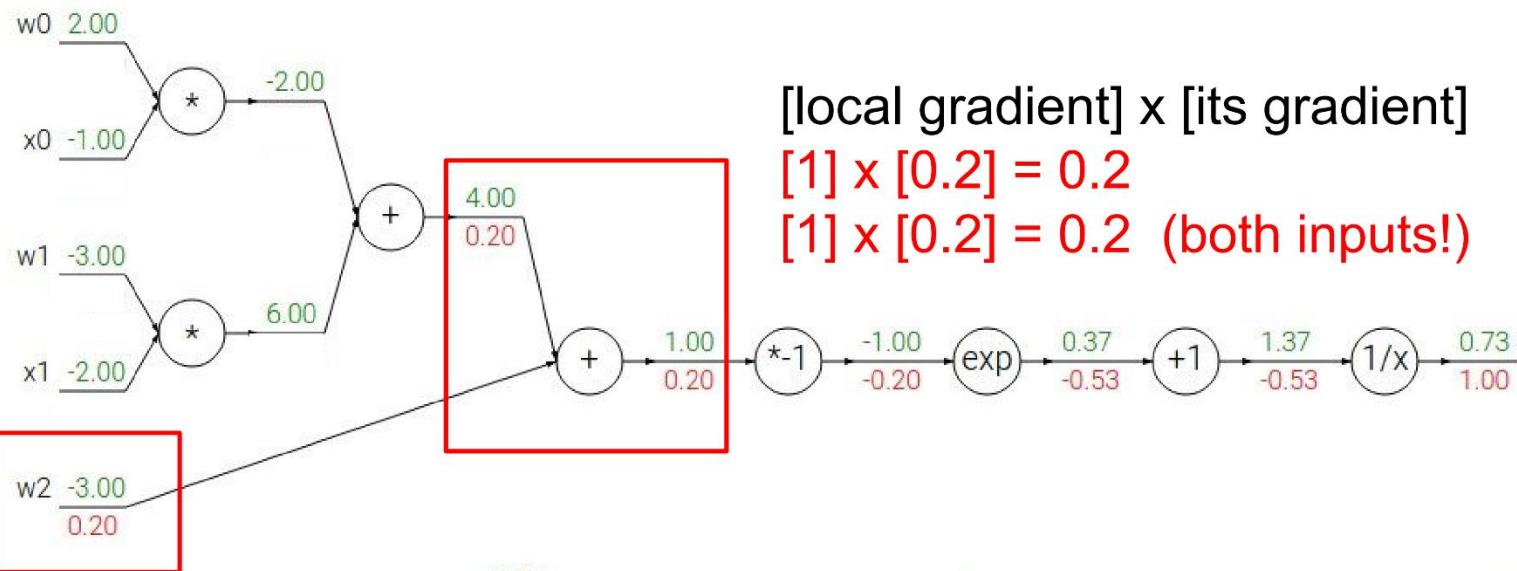
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

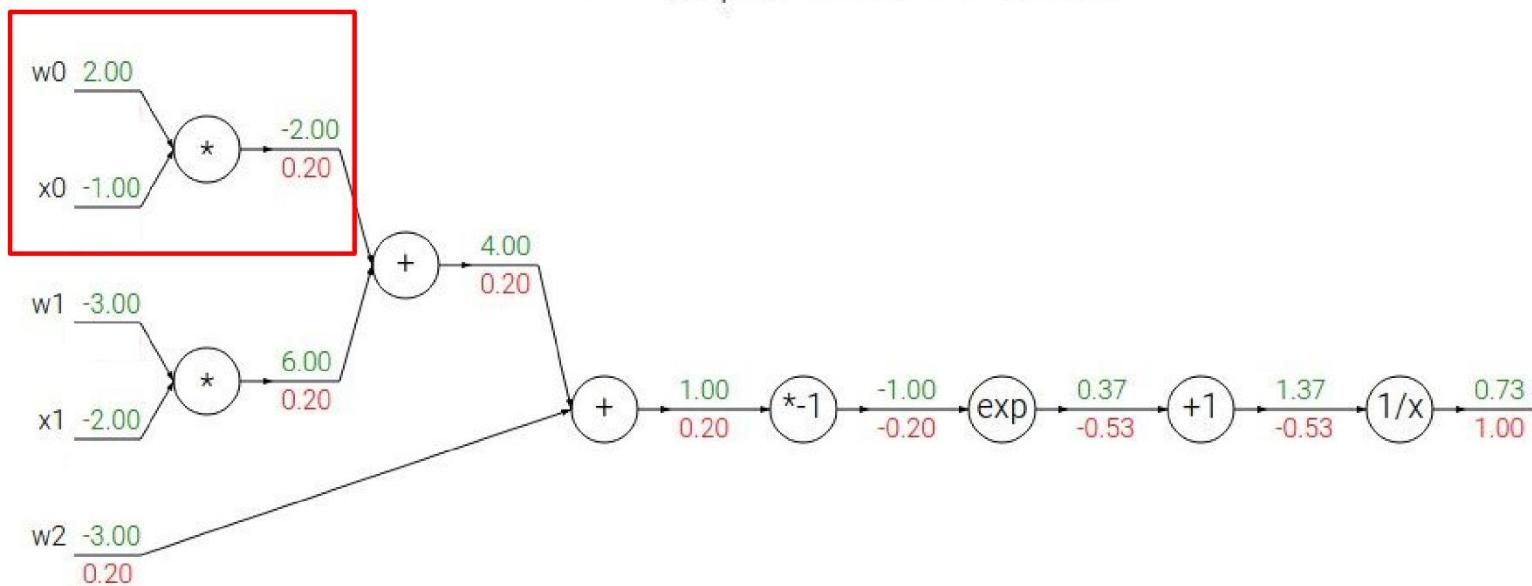
$$f_c(x) = c + x$$

\rightarrow

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

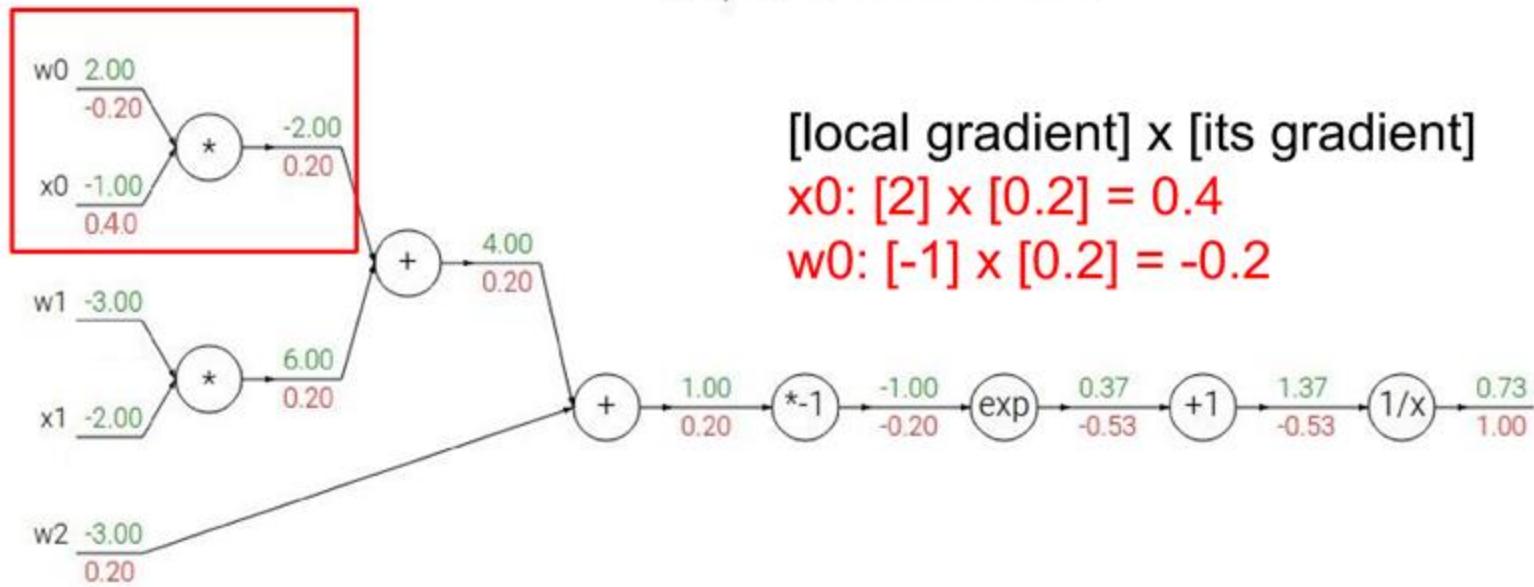
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



[local gradient] x [its gradient]

$$x_0: [2] \times [0.2] = 0.4$$

$$w_0: [-1] \times [0.2] = -0.2$$

$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

\rightarrow

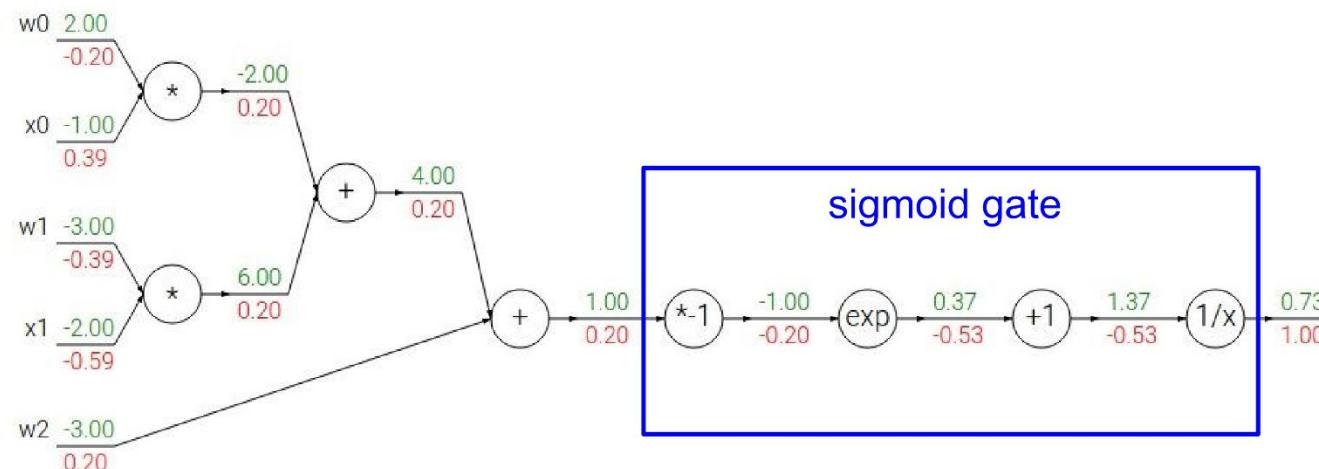
$$\frac{df}{dx} = 1$$

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

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

sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left(\frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

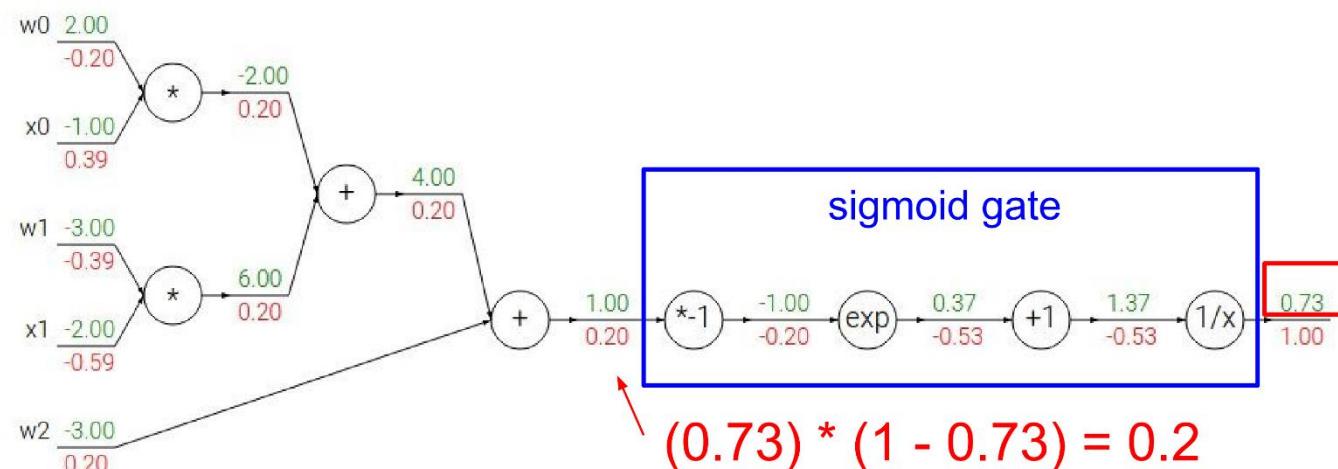


$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

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

sigmoid function

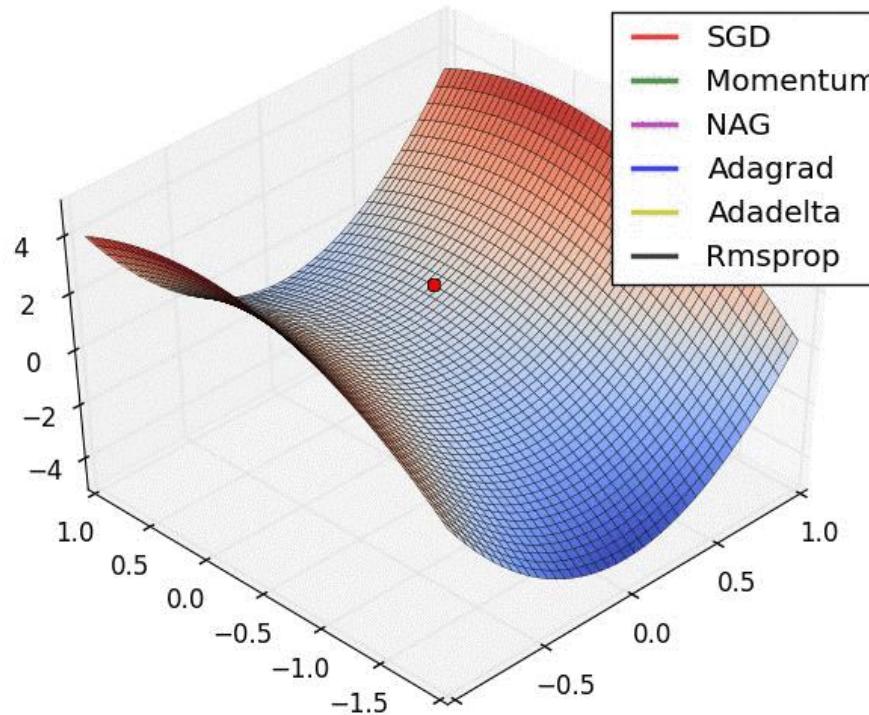
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left(\frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$



Entrenamiento de redes neuronales (profundas) requiere de muchas consideraciones para que sea exitoso

- **Momentum** permite aminorar riesgo de caer en mínimos locales¹

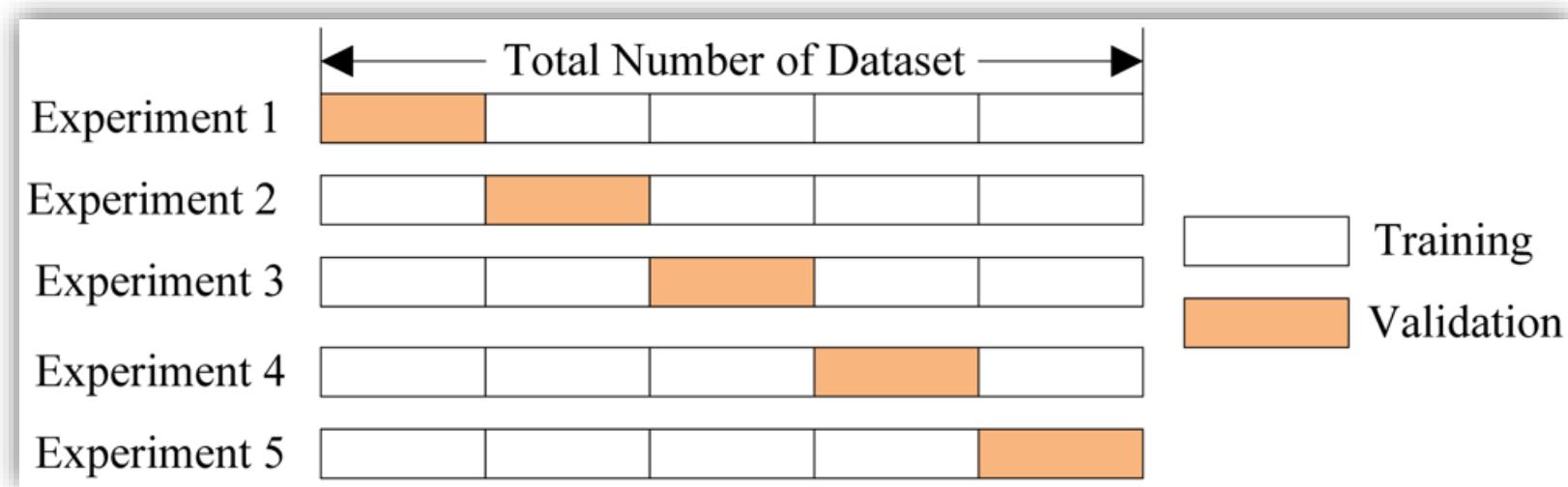
$$v = \gamma v + \alpha \nabla_{\theta} J(\theta; x^{(i)}, y^{(i)})$$
$$\theta = \theta - v$$



¹<https://fosterelli.co/executing-gradient-descent-on-the-earth>

Entrenamiento de redes neuronales (profundas) requiere de muchas consideraciones para que sea exitoso

- **Validación cruzada** permite detectar sobreentrenamiento



Entrenamiento de redes neuronales (profundas) requiere de muchas consideraciones para que sea exitoso

- **Funciones de activación** pueden acelerar o frenar el entrenamiento

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU) [2]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [3]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Entrenamiento de redes neuronales (profundas) requiere de muchas consideraciones para que sea exitoso

- **Inicialización de pesos:** pesos grandes explosión del gradiente, pesos pequeños desvanecimiento del gradiente.
- **Normalización de los datos:** en general, redes neuronales andan mejor con datos normalizados (media 0, varianza 1).

Incluso con todas estas consideraciones, las redes feed-forward con varias capas son muy difíciles de hacer funcionar exitosamente.

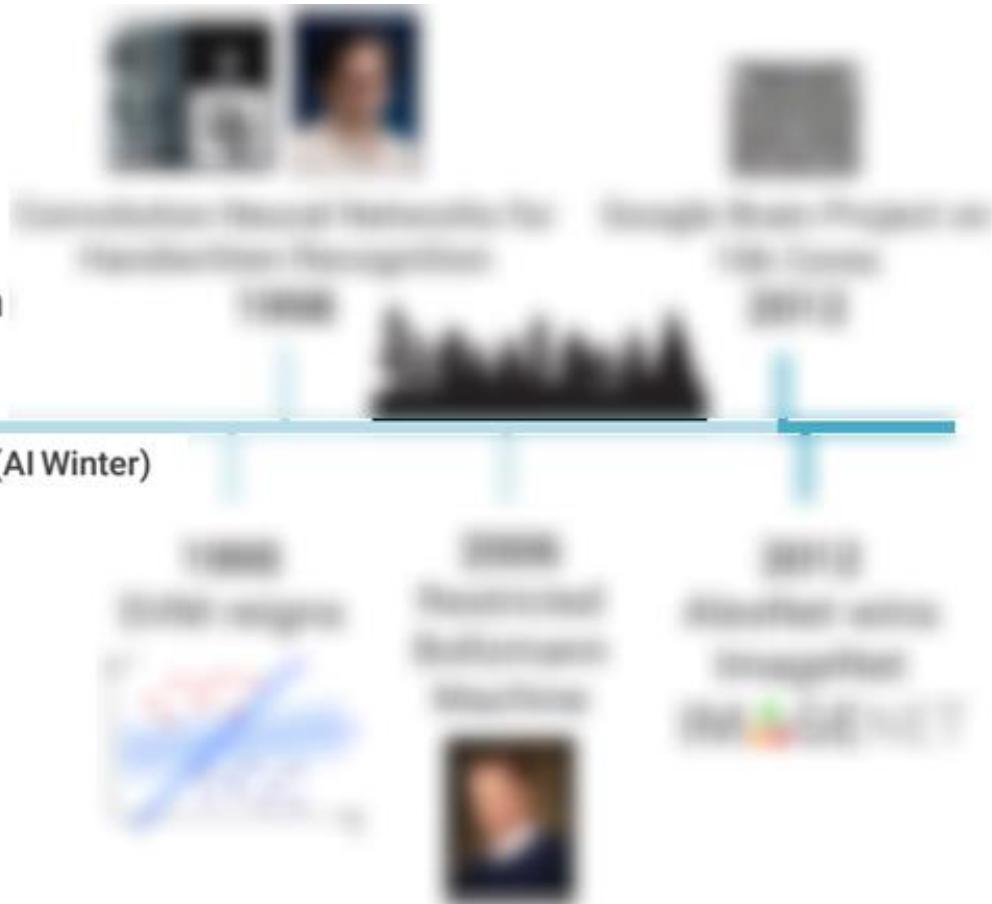


1958 Perceptron

1974 Backpropagation

awkward silence (AI Winter)

1969
Perceptron criticized



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2613 – Inteligencia Artificial

Redes Neuronales

Profesor: Hans Löbel