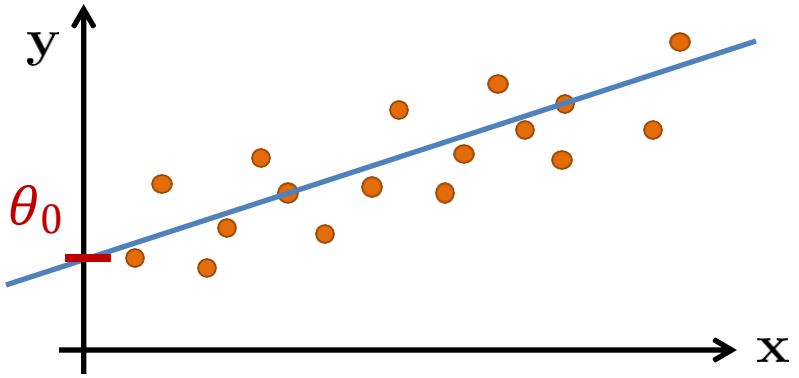
Introducción a las Redes Neuronales

Resumen Unidad 2

Regresión Lineal

- Método de aprendizaje supervisado para encontrar un modelo lineal de la forma

$$\hat{y}_i = \theta_0 + \sum_{j=1}^{d} x_{ij}\theta_j = \theta_0 + x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{id}\theta_d$$



Objetivo: encontrar un modelo que explique un objetivo Y dada la entrada X

Regresión Logística

- Loss Function (Función de pérdida)

$$\mathcal{L}(\hat{y}_i, y_i) = -[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

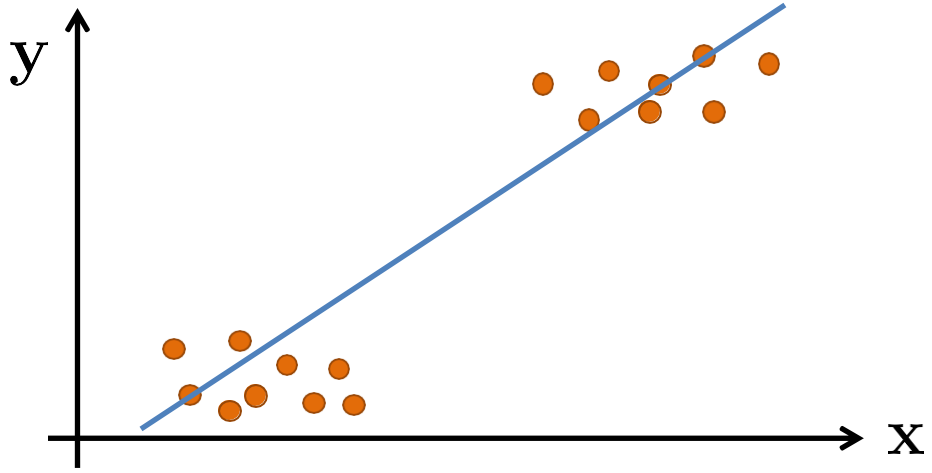
- Cost Function (Función de costos)

$$\mathcal{C}(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log[1 - \hat{y}_i])$$

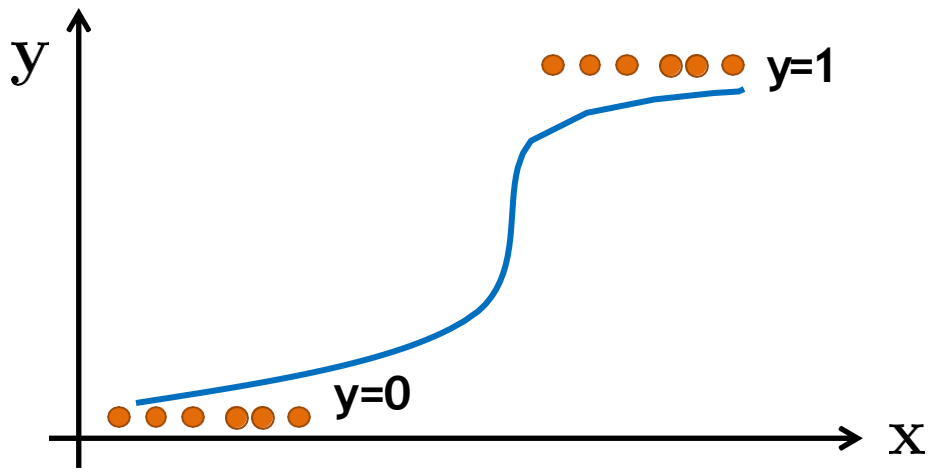
Minimization

$\hat{y}_i = \sigma(x_i\boldsymbol{\theta})$

Lineal vs Logística

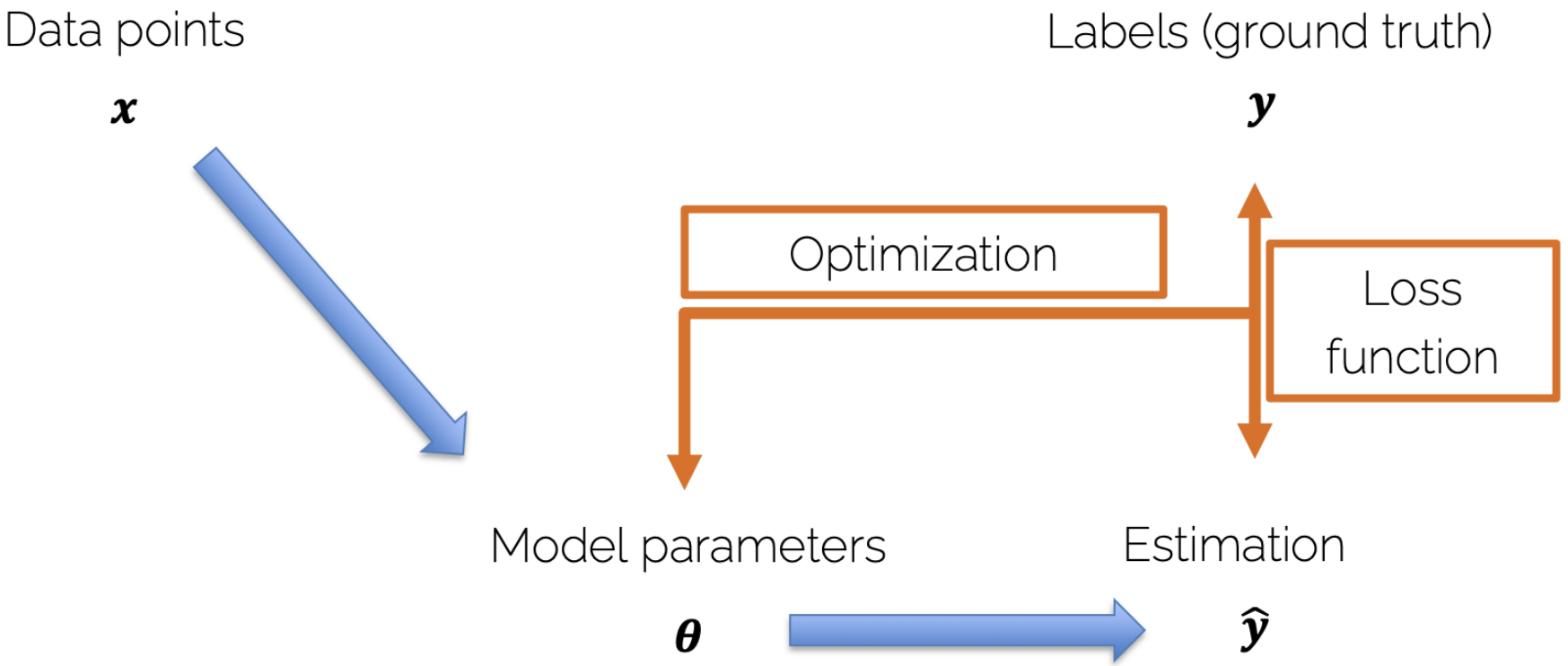


Las predicciones pueden superar el rango de las muestras de entrenamiento
→ en el caso de la clasificación [0;1] esto se convierte en un verdadero problema.



Las predicciones están garantizadas para estar dentro de [0;1]

¿Cómo obtener el modelo?



Linear Score Functions

- Como vimos en regresión lineal

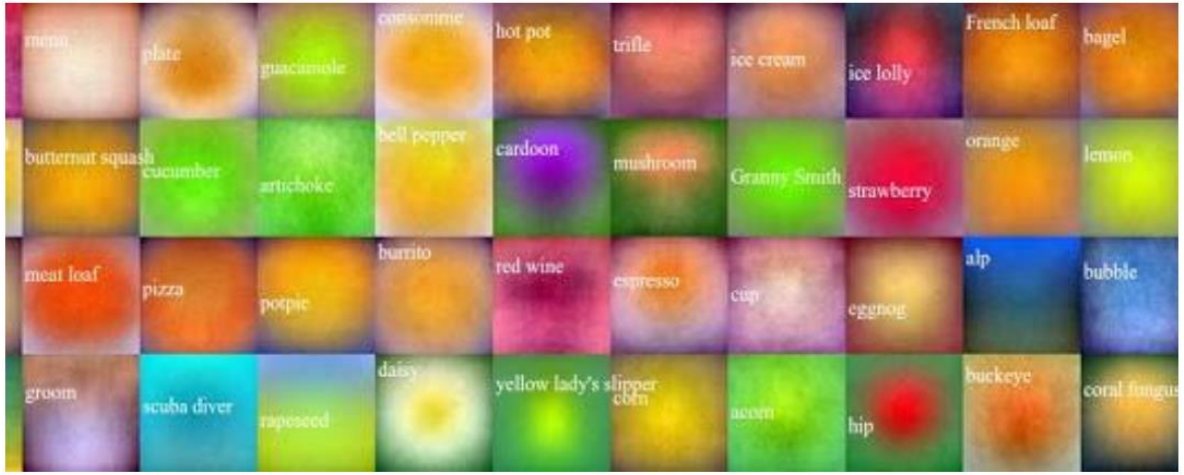
$$f_i = \sum_j w_{i,j} x_j$$

$$\boldsymbol{f} = \boldsymbol{W}\,\boldsymbol{x} \qquad \text{(Matrix Notation)}$$

Linear Score Functions en Imágenes

- Linear score function $\boldsymbol{f = Wx}$

On CIFAR-10

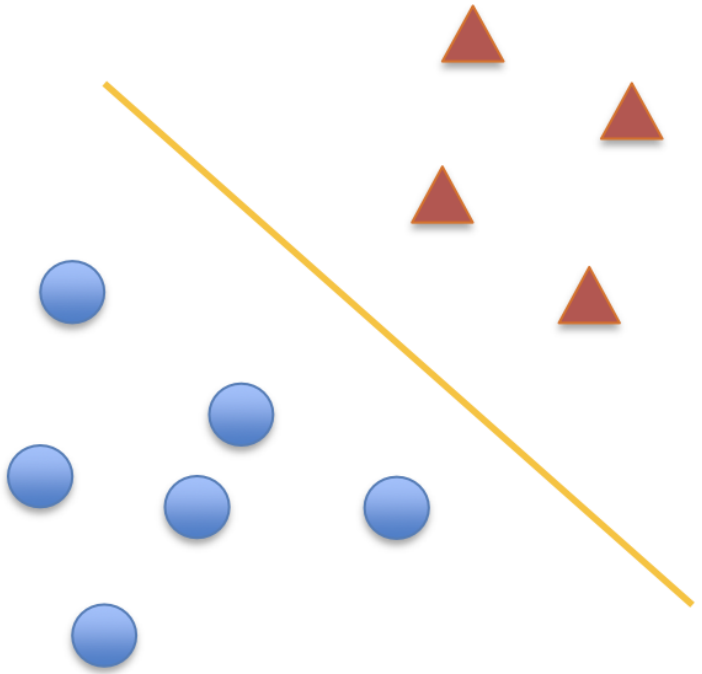


On ImageNet

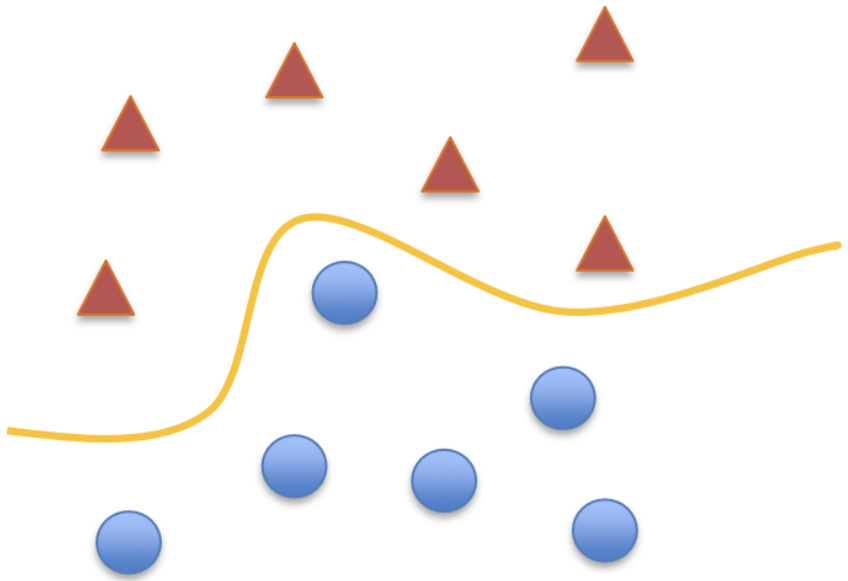
Source:: Li/Karpathy/Johnson

Linear Score Functions

Logistic Regression



Linear Separation Impossible!



Linear Score Functions

- ¿Podemos mejorar la regresión lineal?
  - Multiplicar con otra matriz de pesos $\boldsymbol{W_2}$

$$\hat{\boldsymbol{f}} = \boldsymbol{W_2} \cdot \boldsymbol{f}$$
$$\hat{\boldsymbol{f}} = \boldsymbol{W_2} \cdot \boldsymbol{W} \cdot \boldsymbol{x}$$

  - El funcionamiento sigue siendo lineal.

$$\widehat{\boldsymbol{W}} = \boldsymbol{W_2} \cdot \boldsymbol{W}$$
$$\hat{\boldsymbol{f}} = \widehat{\boldsymbol{W}}\, \boldsymbol{x}$$

  - Solución → ¡añadir no linealidad!

Red Neuronal

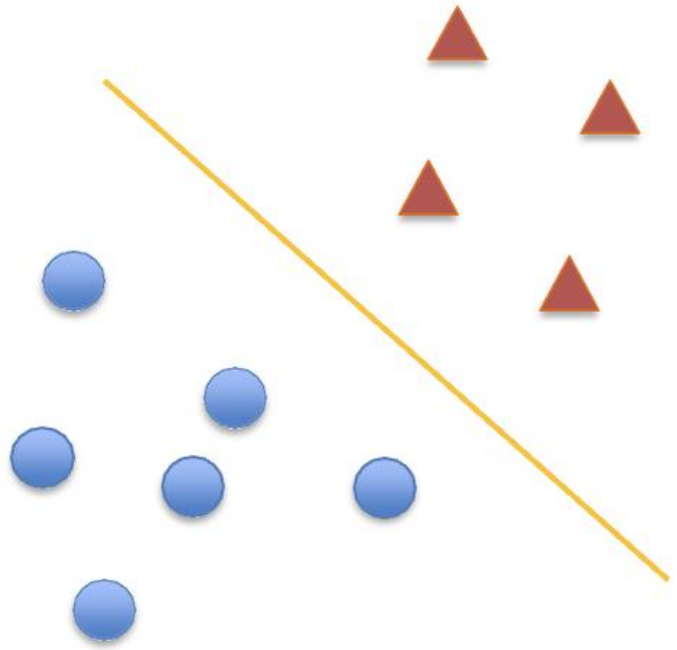
- **Linear score function $f = Wx$**

- **La red neuronal es un anidamiento de funciones**
  - **2 capas:** $f = W_2 \max(0, W_1 x)$
  - **3 capas:** $f = W_3 \max(0, W_2 \max(0, W_1 x))$
  - **4 capas:** $f = W_4 \tanh(W_3, \max(0, W_2 \max(0, W_1 x)))$
  - **5 capas:** $f = W_5 \sigma(W_4 \tanh(W_3, \max(0, W_2 \max(0, W_1 x))))$
  - Hasta ciento de capas

Introducción a las Redes Neuronales

Red Neuronal

Regresión logística

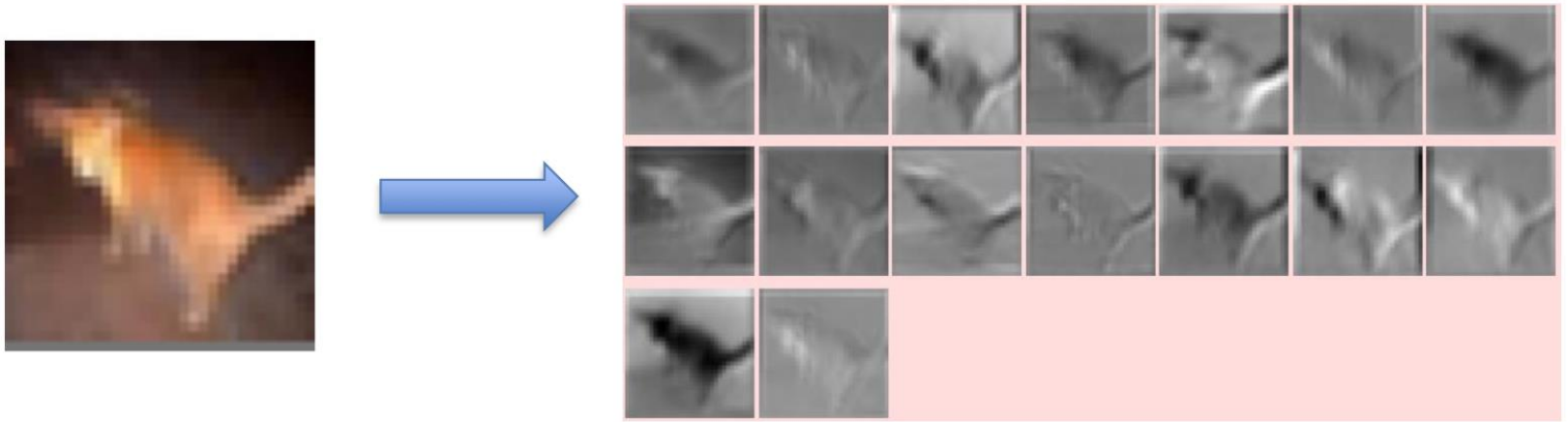


Redes neuronales

Red Neuronal

- Non-linear score function $\boldsymbol{f = \ldots(\max(0, W_1 x))}$

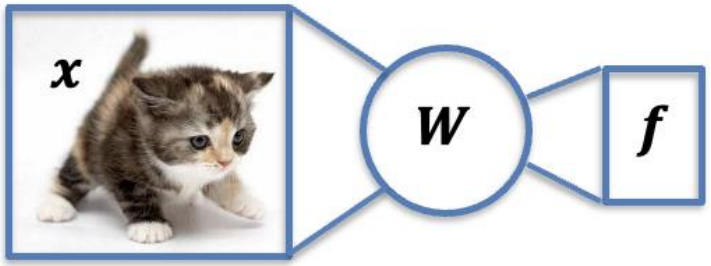
On CIFAR-10



Visualización de las activaciones de la primera capa.

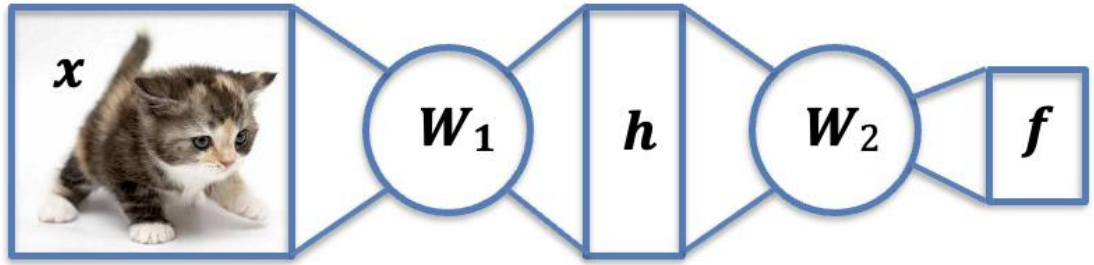
Red Neuronal

Red de 1 capa: $f = Wx$



$128 \times 128 = 16384$ 10

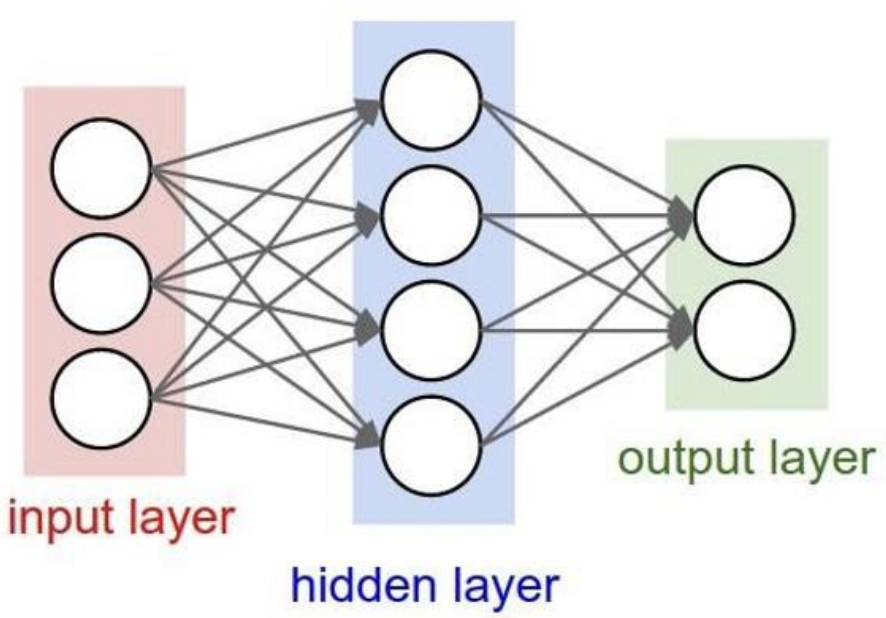
Red de 2 capas: $f = W_2 \max(0, W_1 x)$



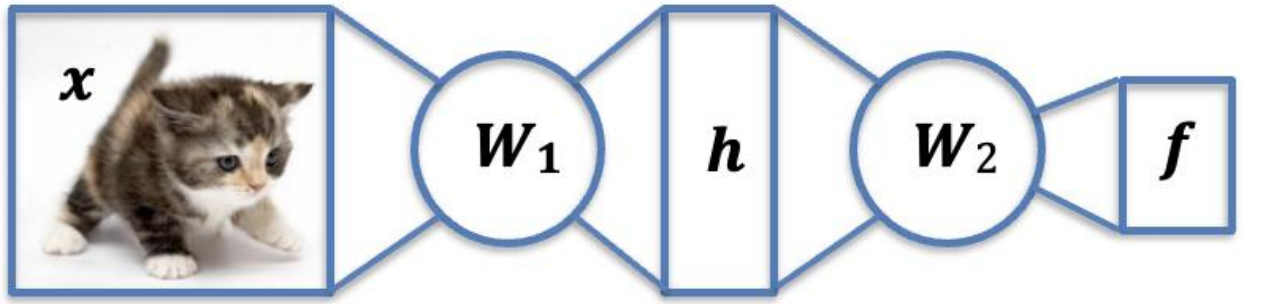
$128 \times 128 = 16384$ 1000 10

¿Por qué es útil esta estructura?

Red Neuronal



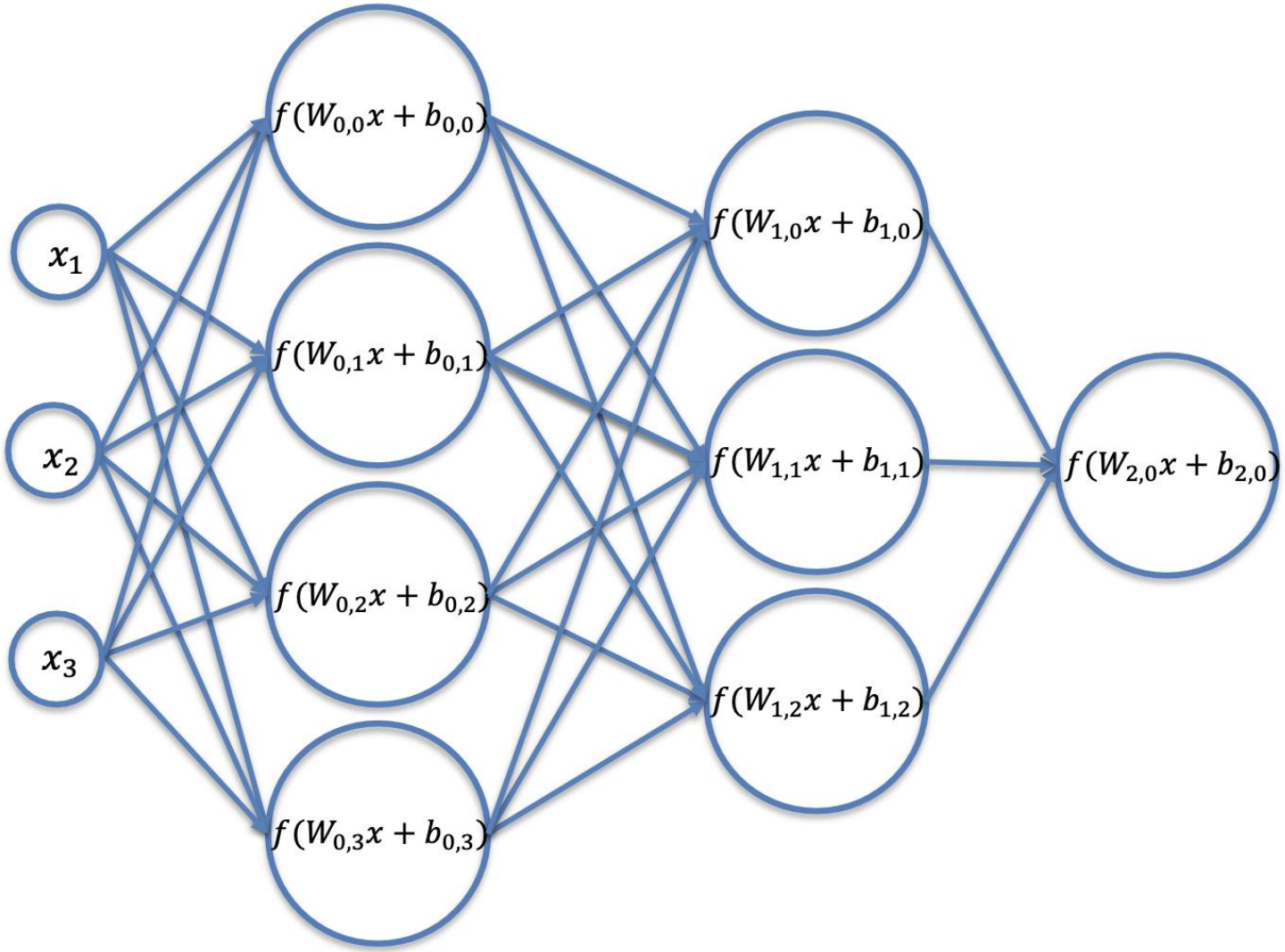
Red de 2 capas: $f = W_2 \max(0, W_1 x)$



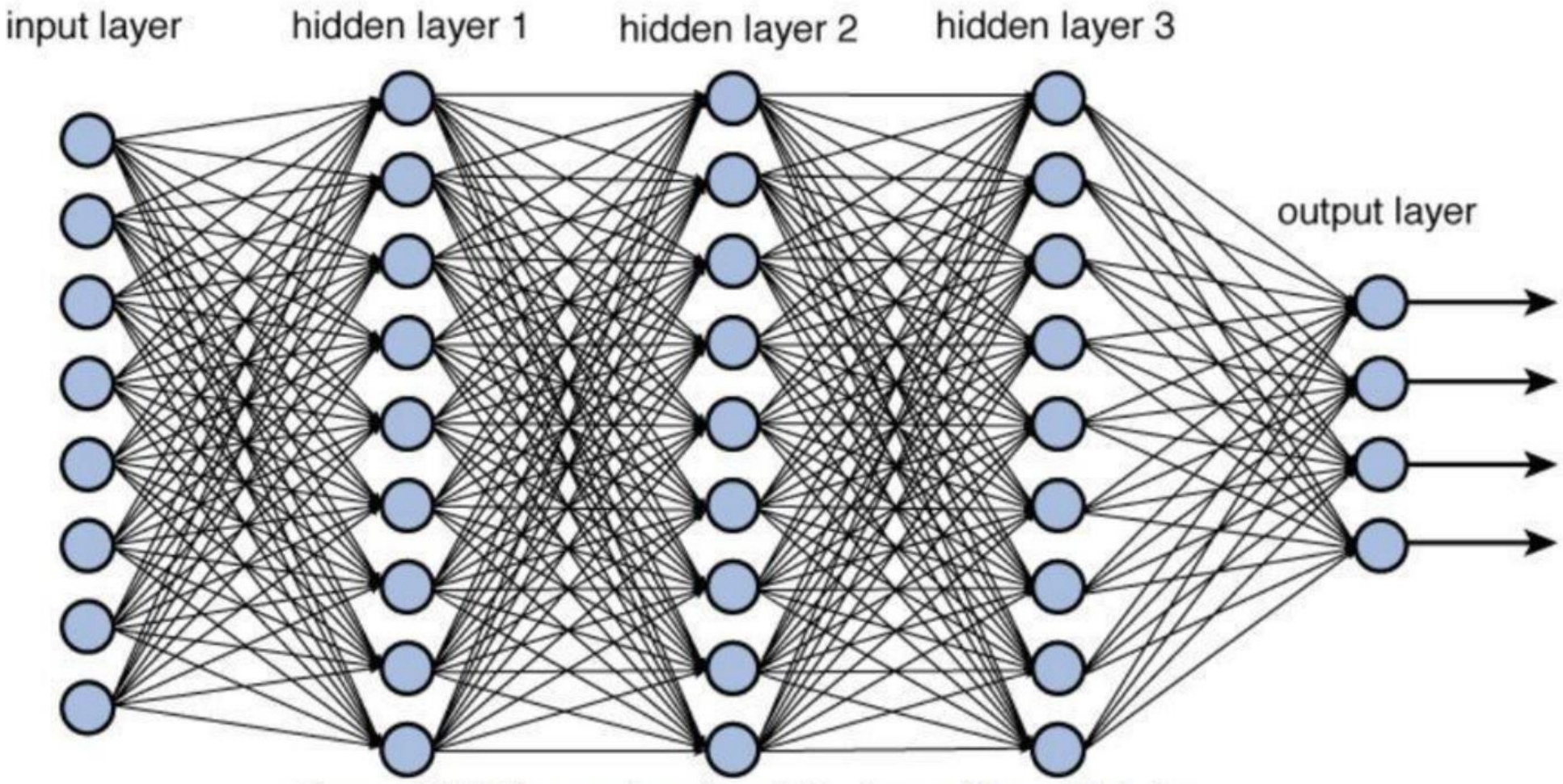
$128 \times 128 = 16384$ | 1000 | 10

Capa de entrada | **Capa oculta (Hidden Layer)** | **Capa de salida**

Red Neuronal

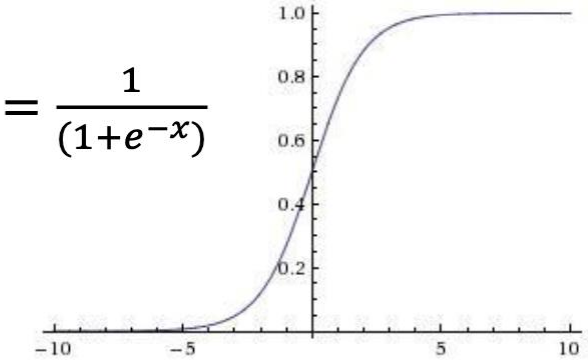


Red Neuronal

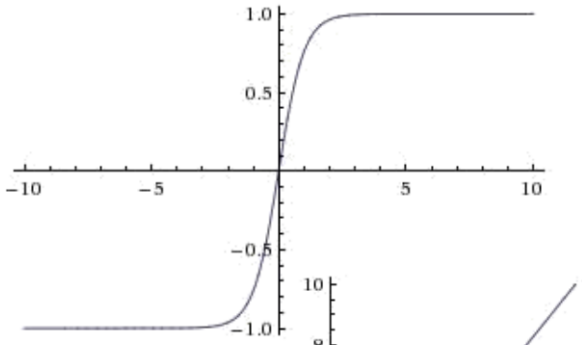


Funciones de activación

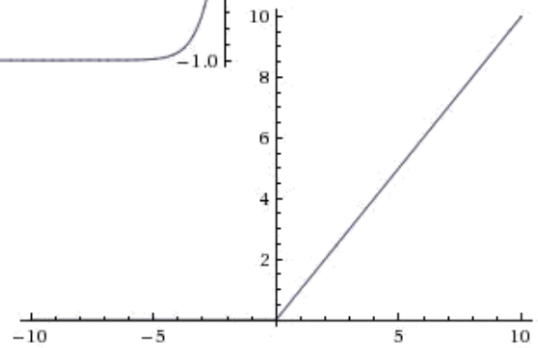
Sigmoid: $\sigma(x) = \frac{1}{(1+e^{-x})}$



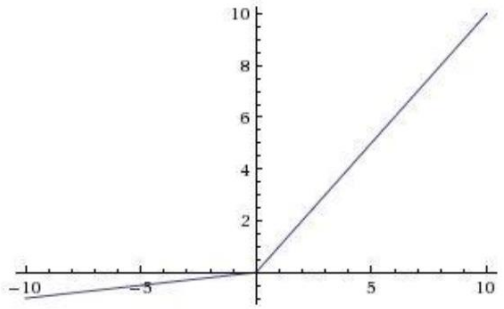
tanh: $\tanh(x)$



ReLU: $\max(0, x)$



Leaky ReLU: $\max(0.1x, x)$



Parametric ReLU: $\max(\alpha x, x)$

Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$

Red Neuronal

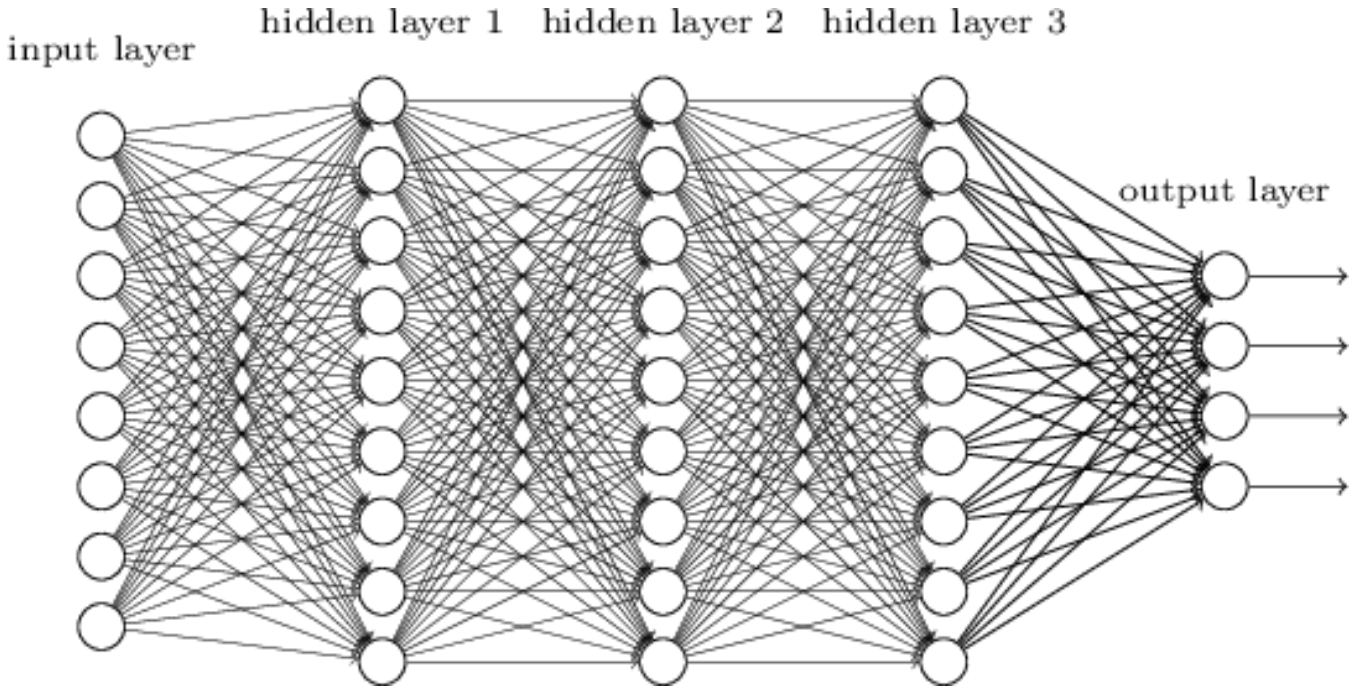
$$f = W_3 \cdot (W_2 \cdot (W_1 \cdot x)))$$

¿Por qué funciones de activación?

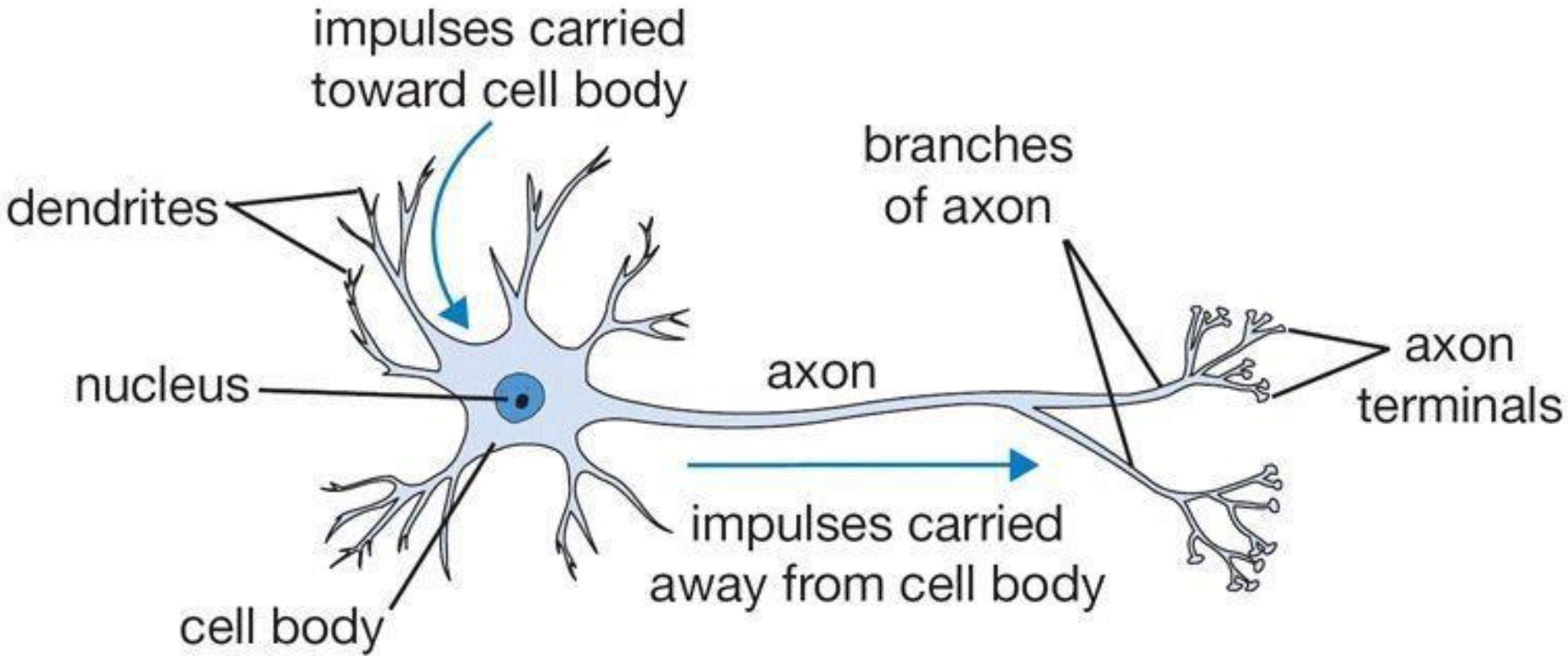
Concatenar capas lineales sería mucho más barato...

Red Neuronal

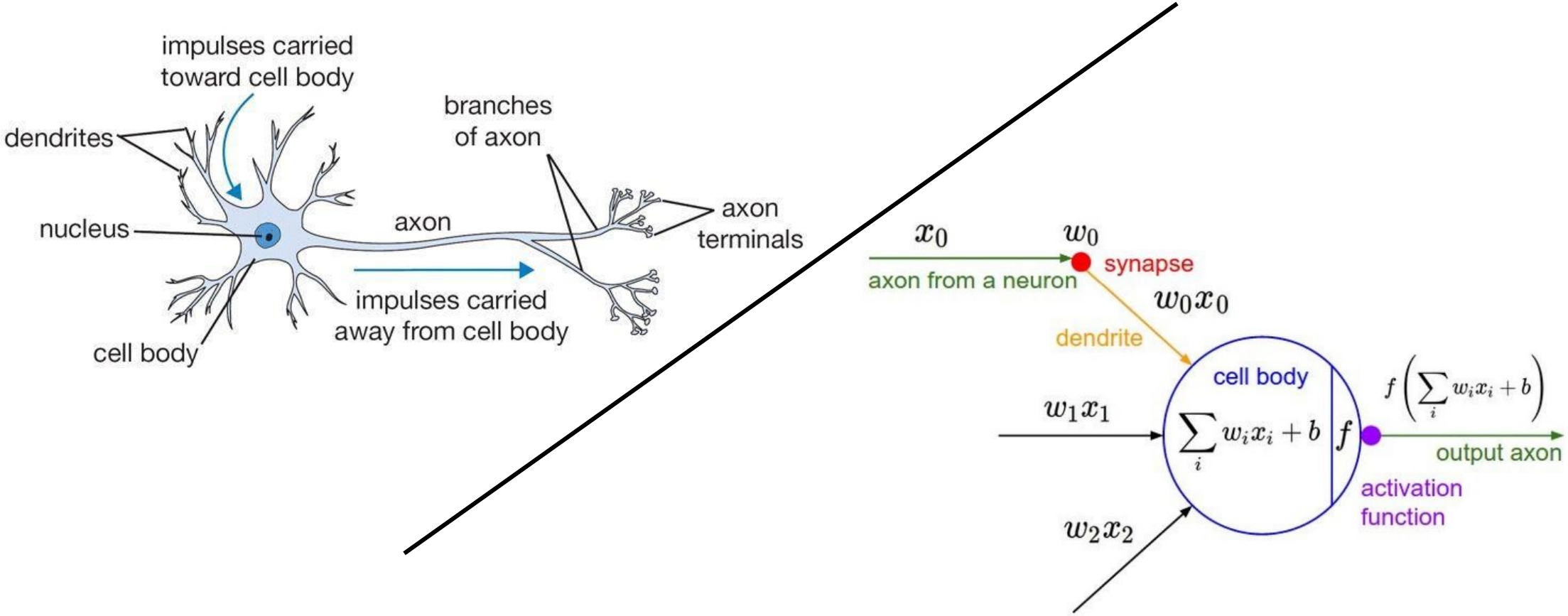
¿Por qué organizar una red neuronal en capas?



Neuronas Biológicas



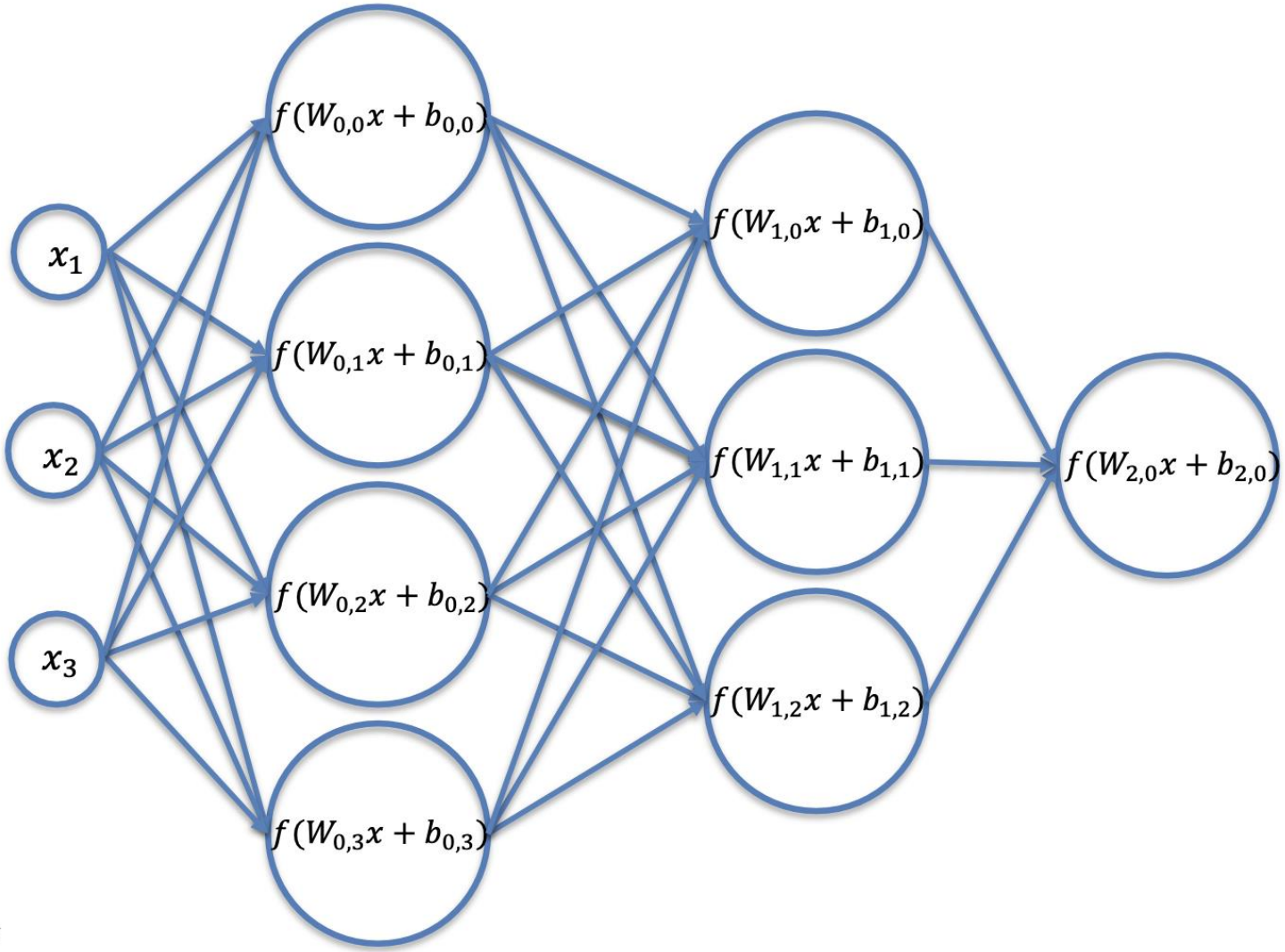
Neuronas Biológicas



Redes Neuronales vs Cerebro

Las redes neuronales se inspiran en el cerebro,
pero ni de lejos en términos de complejidad.

Red Neuronal



Red Neuronal

- **Resumen**
  - Dado un conjunto de datos con pares de entrenamiento de ground truth $[x_i ; y_i]$,
  - Encontrar los pesos y biases $\boldsymbol{W}$ óptimos mediante el Stochastic Gradient Descent, de forma que se minimice la función de pérdida.
    - Calcula gradientes con backpropagation (usa el modo por lotes - batches)
    - Iterar muchas veces sobre el conjunto de entrenamiento (SGD)

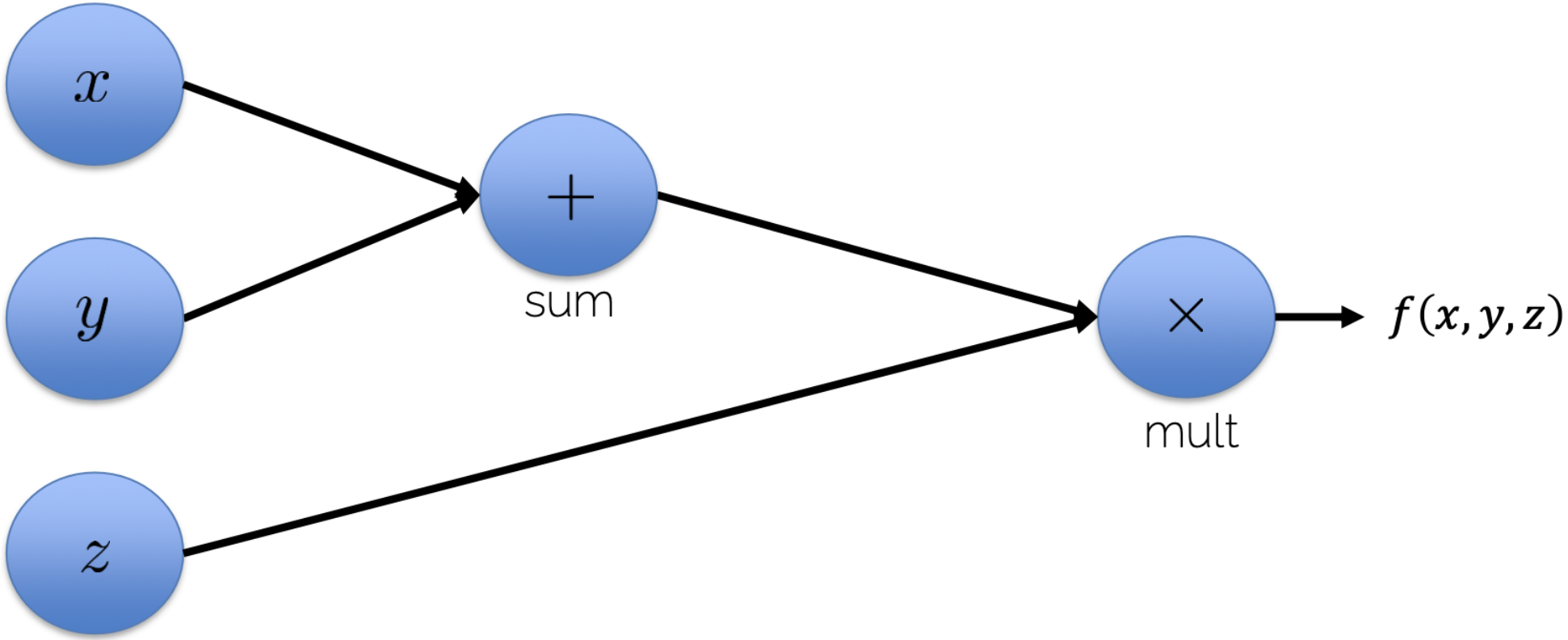
Grafos Computacionales

Grafos Computacionales

- Gráfico direccional
- Las operaciones matriciales se representan como nodos de cálculo.
- Los nodos de vértice son variables u operadores como +, - , *, / , log(), exp()
- Las aristas direccionales muestran el flujo de entradas a los vértices

Grafos Computacionales

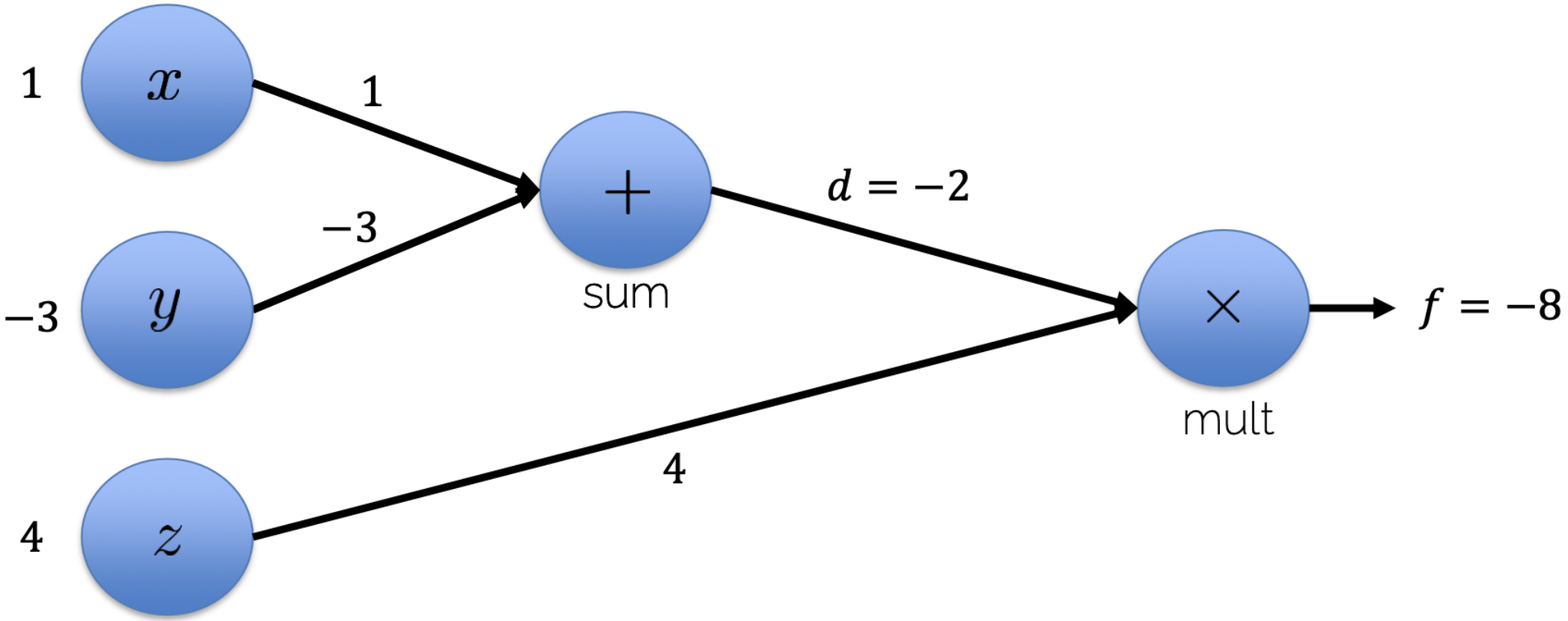
- $f(x, y, z) = (x + y) \cdot z$



Evaluación: Forward Pass

- $f(x, y, z) = (x + y) \cdot z$

Initialization $x = 1, y = -3, z = 4$



Grafos Computacionales

- ¿Por qué hablar de los gráficos computacionales?
- Las redes neuronales tienen arquitecturas complicadas

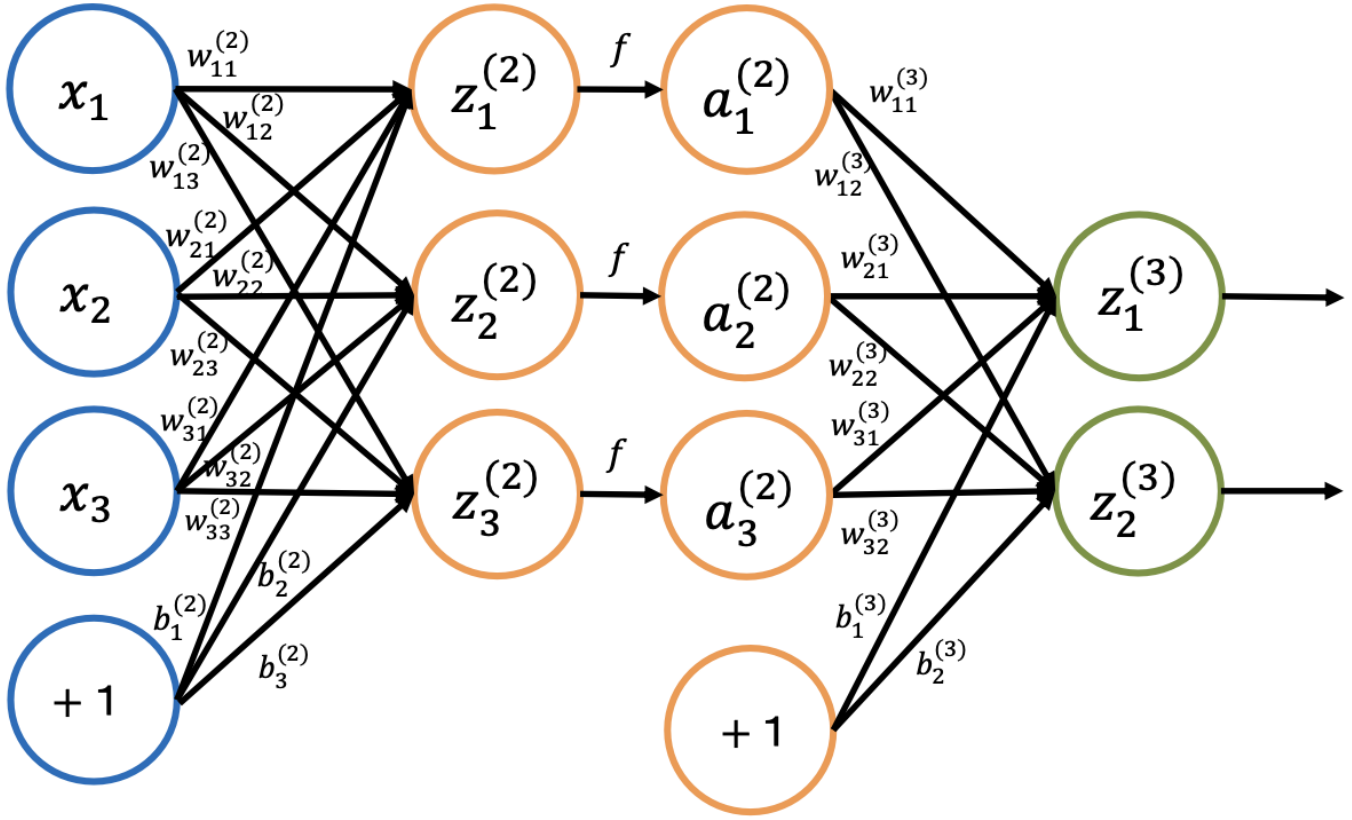
$$\boldsymbol{f} = \boldsymbol{W_5}\,\sigma(\boldsymbol{W_4}\ \tanh(\boldsymbol{W_3}\,, \max(\mathbf{0}, \boldsymbol{W_2}\ \max(\mathbf{0}, \boldsymbol{W_1}\,\boldsymbol{x}))))$$

- Muchas operaciones matriciales
- Representar las NN (Neural Networks) como grafos computacionales.

Grafos Computacionales

- Una red neuronal puede representarse como un grafo computacional...
  - tiene nodos de cálculo (operaciones)
  - tiene aristas que conectan los nodos (flujo de datos)
  - es direccional
  - puede organizarse en capas

Grafos Computacionales



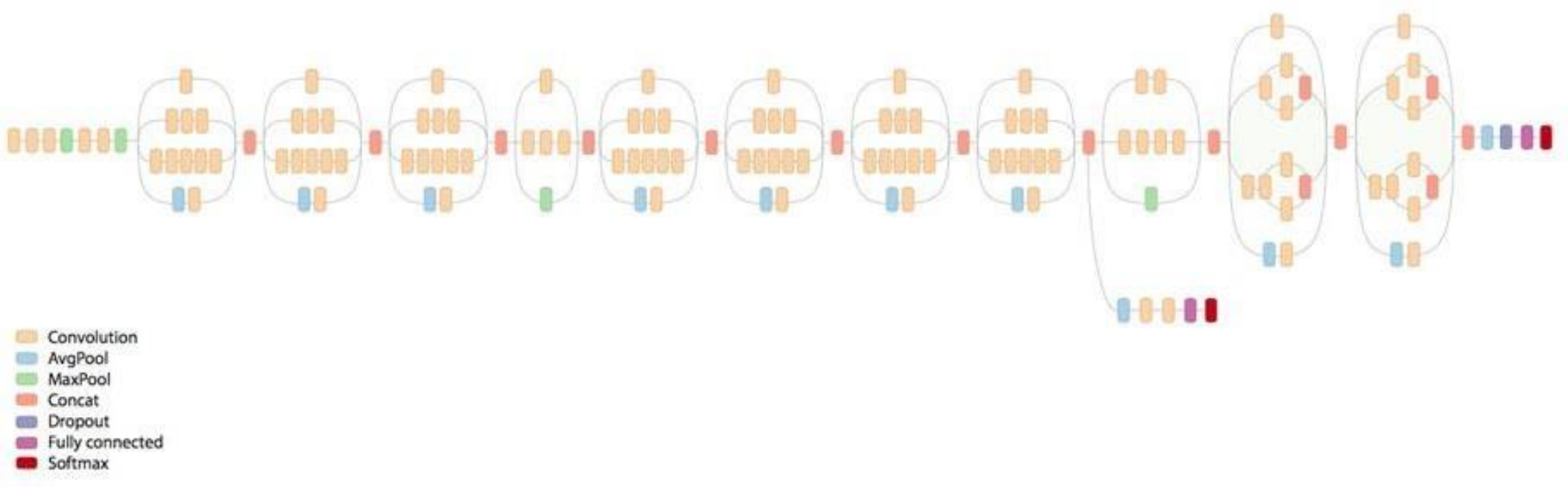
$$z_k^{(2)} = \sum_i x_i w_{ik}^{(2)} + b_k^{(2)}$$

$$a_k^{(2)} = f(z_k^{(2)})$$

$$z_k^{(3)} = \sum_i a_i^{(2)} w_{ik}^{(3)} + b_k^{(3)}$$

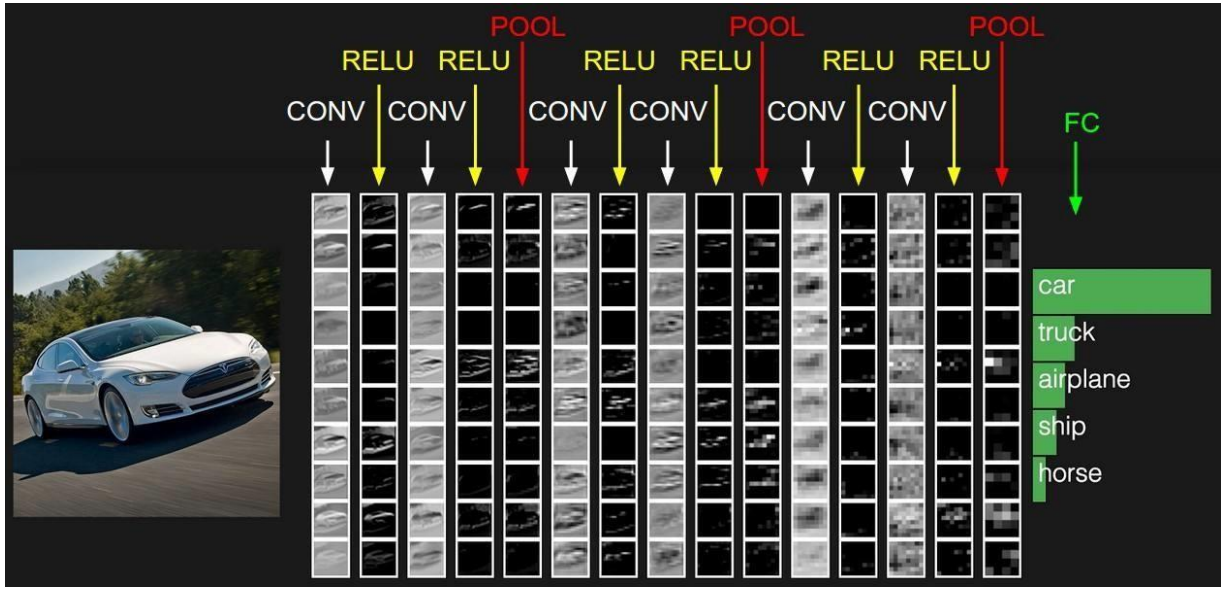
Grafos Computacionales

- **De un conjunto de neuronas a una cadena de computación estructurada**



Grafos Computacionales

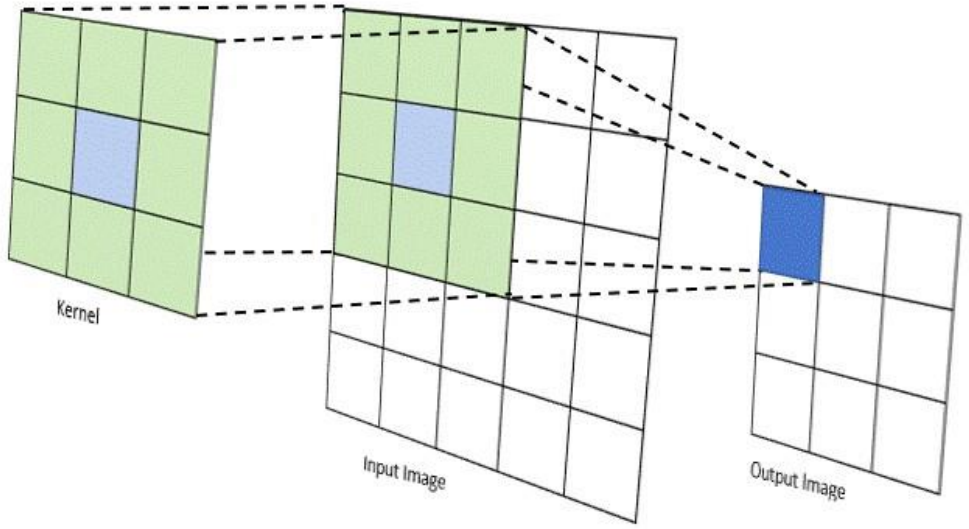
- El cálculo de la red neuronal tiene otros significados:
  - La multiplicación de $\boldsymbol{W}$ y $\boldsymbol{x}$: codificar la información de entrada.
  - La función de activación: seleccionar las características clave



Grafos Computacionales

- Los cálculos de las redes neuronales tienen otros significados:

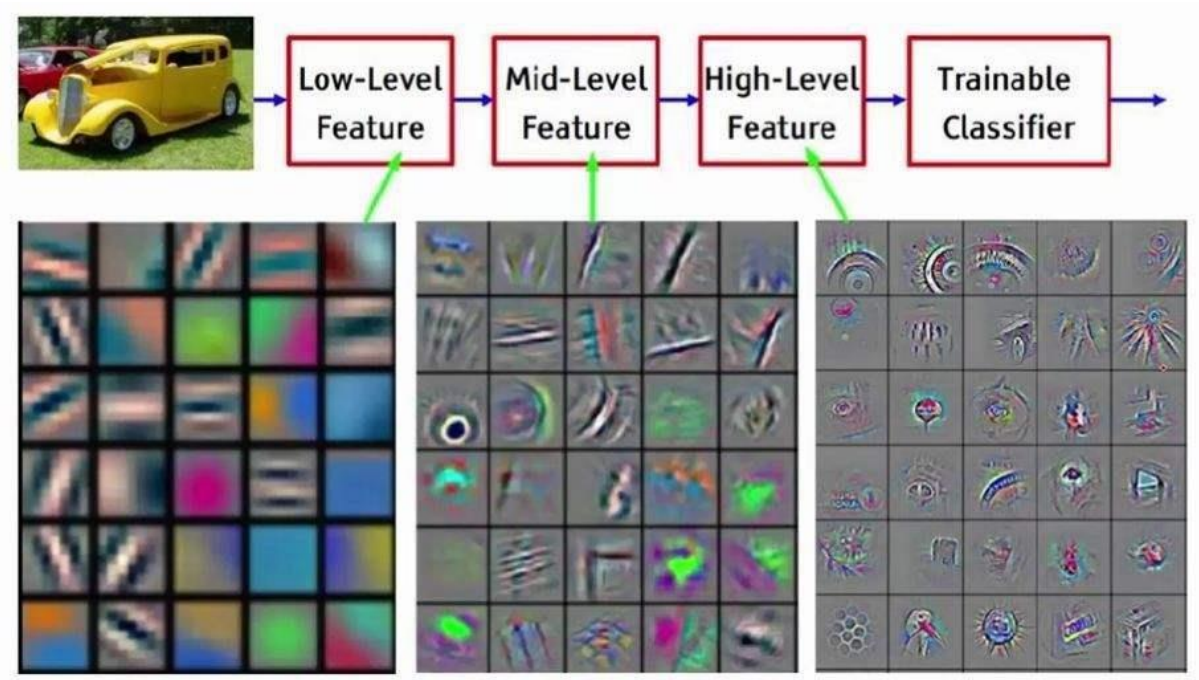
  - Las capas convolucionales: extraen características útiles con pesos compartidos



Grafos Computacionales

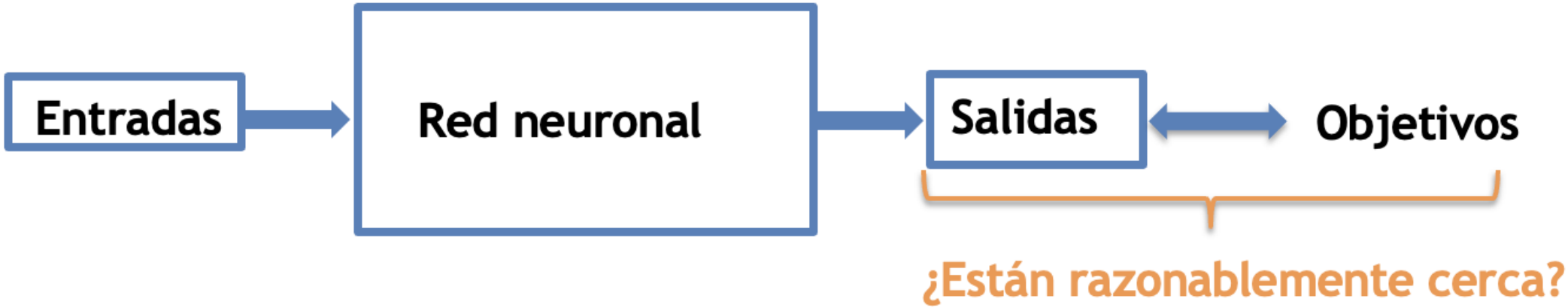
- Los cálculos de las redes neuronales tienen otros significados:

  - Las capas convolucionales: extraen características útiles con pesos compartidos



Loss Functions
(Funciones de Perdida)

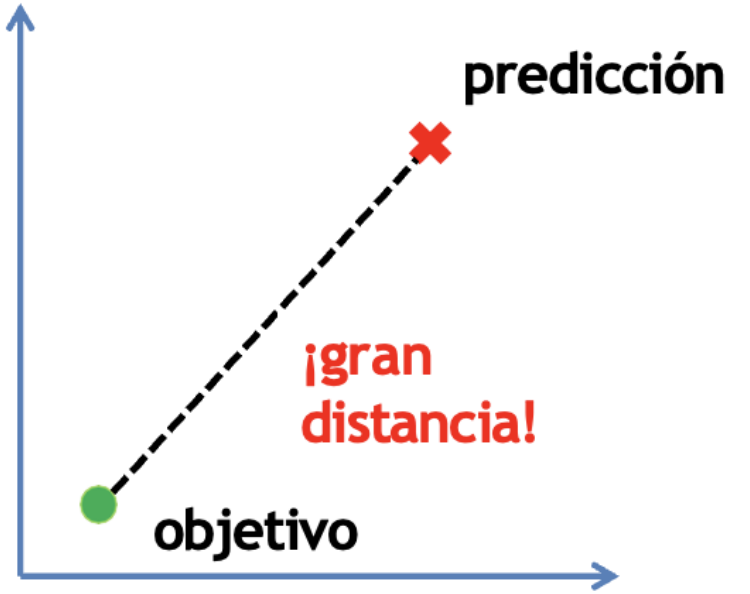
¿Qué hacemos?



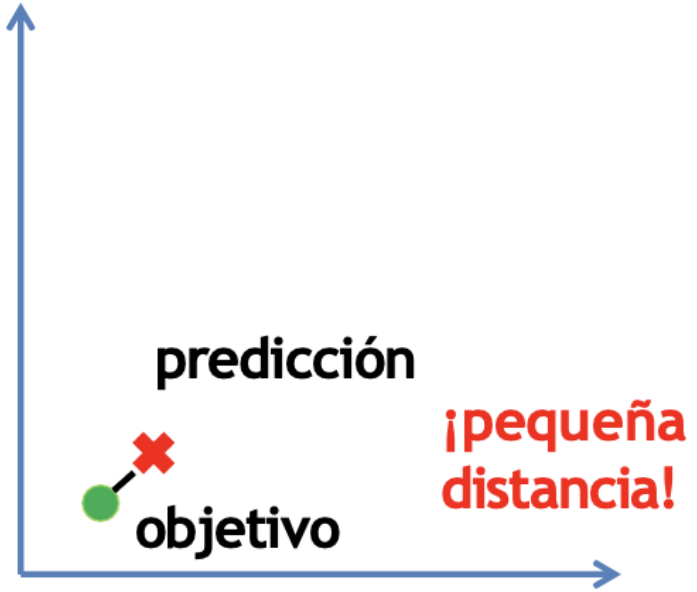
Necesitamos una forma de describir lo cerca que están las salidas de la red (=predicciones) de los objetivos.

¿Qué hacemos?

- Idea: ¡calcular "distancia" entre la predicción y el objetivo!



mala predicción



buena predicción

Loss Functions

- Una función para medir la bondad de las predicciones (o, lo que es lo mismo, el rendimiento de la red).

- Intuitivamente,
  - una gran pérdida indica malas predicciones/rendimiento (→ es necesario mejorar el rendimiento entrenando el modelo).
  - la elección de la función de pérdida depende del problema o la distribución de la variable objetivo

Regression Loss

- L1 Loss:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = \frac{1}{n}\sum_{i}^{n} ||y_i\ - \hat{y}_i||_1$$

- MSE Loss:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = \frac{1}{n}\sum_{i}^{n} ||y_i\ - \hat{y}_i||_2^2$$

Binary Cross Entropy

- Loss Function para clasificación binaria (si/no)

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = -\frac{1}{n}\sum_{i}^{n}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

¡Sí! (0,8)

¡No! (0.2)

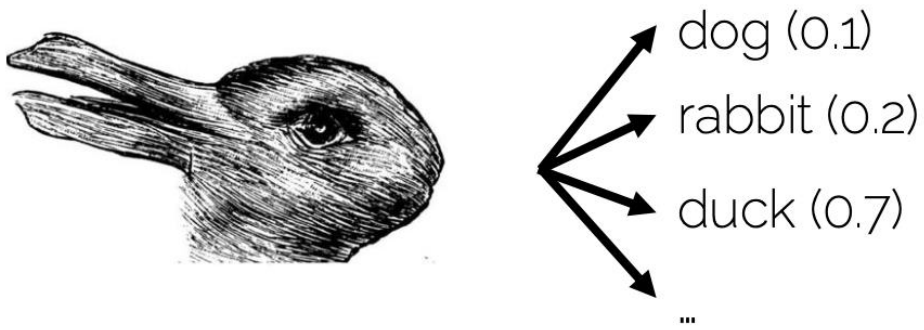
La red predice
la probabilidad de la entrada
perteneciente a la clase "sí"

Cross Entropy

- = Loss Function para clasificación multi-clase

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = -\sum_{i=1}^{n}\sum_{k=1}^{k}(y_{ik} \cdot \log \hat{y}_{ik})$$

Esto generaliza el caso de la diapositiva anterior

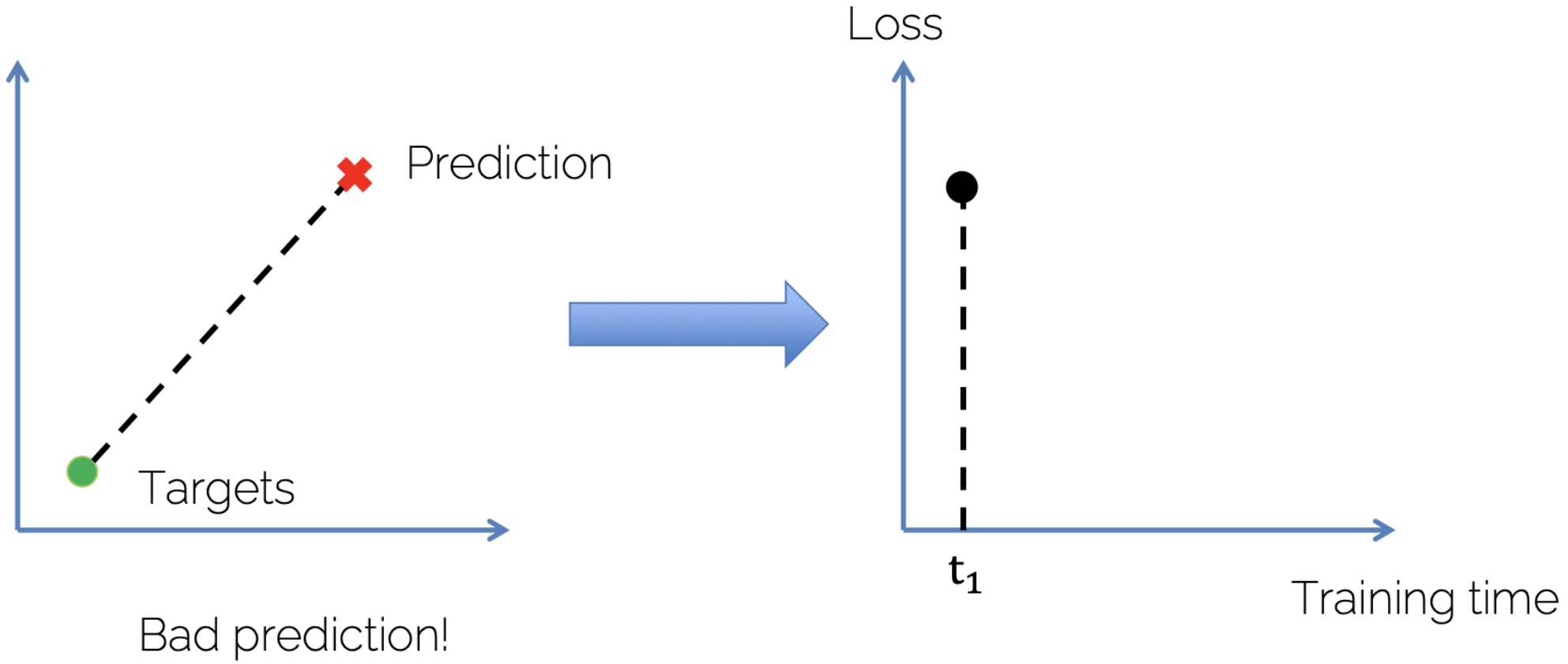


Casos Generales

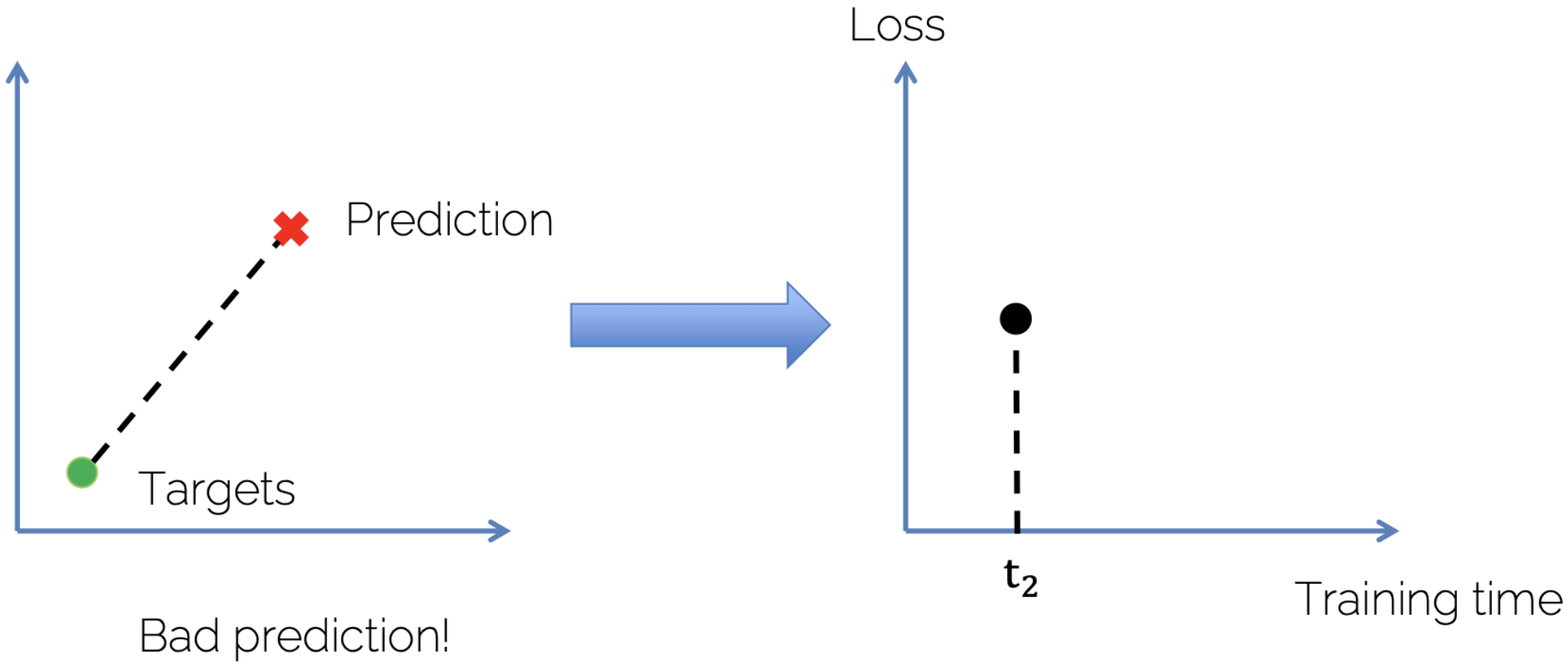
- Ground truth: $\boldsymbol{y}$
- Prediction: $\hat{\boldsymbol{y}}$
- Loss function: $\boldsymbol{L(y, \hat{y})}$

- Motivación:
  - minimizar la pérdida <=> encontrar mejores predicciones
  - predicciones generadas por la NN
  - encontrar mejores predicciones <=> encontrar mejor NN

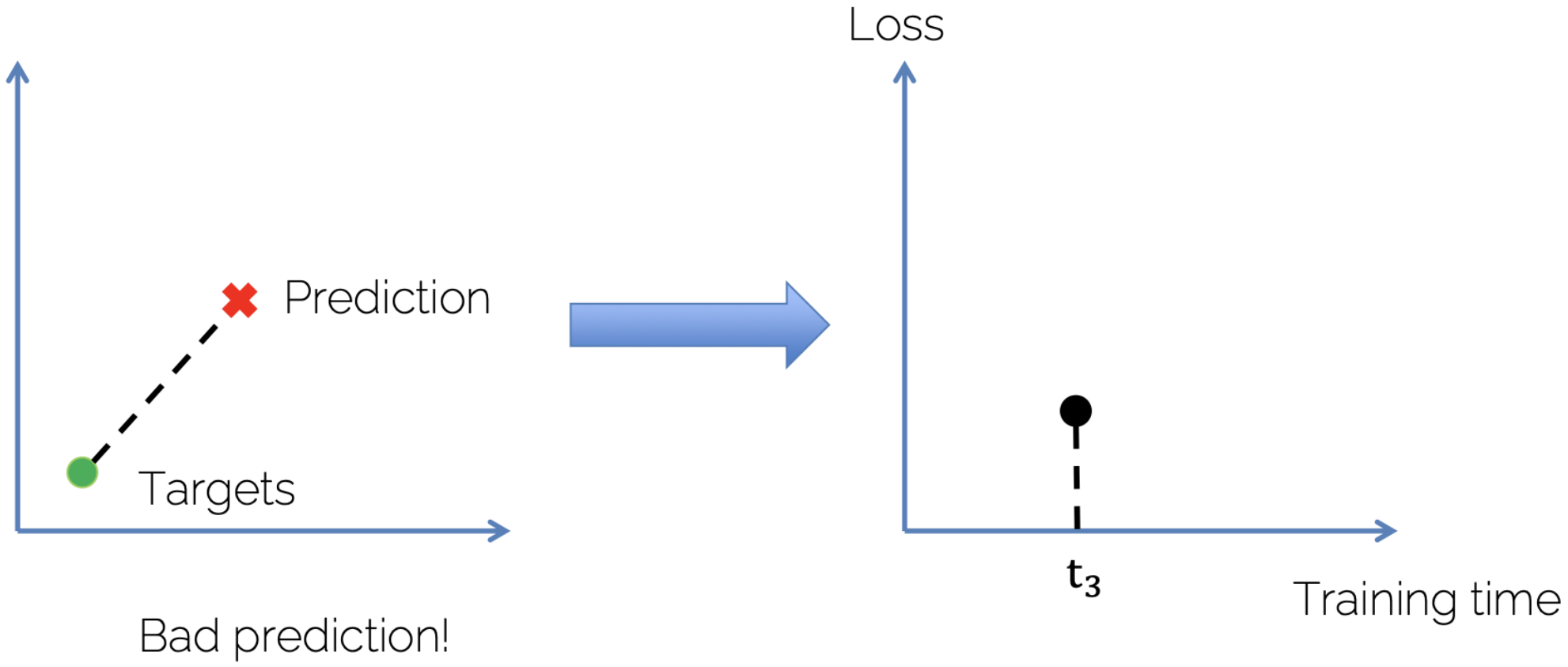
Inicialmente



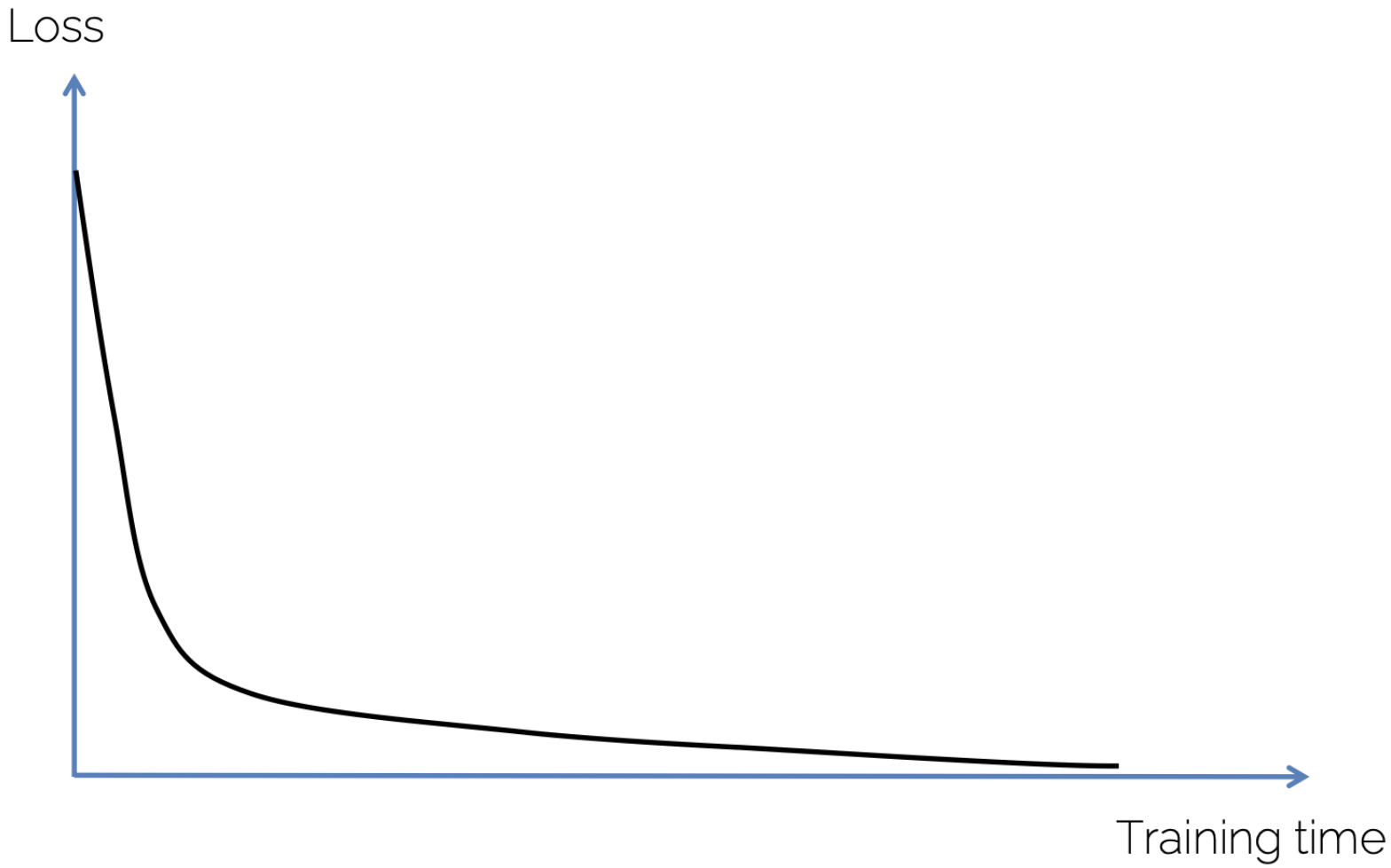
Durante Entrenamiento



Durante Entrenamiento

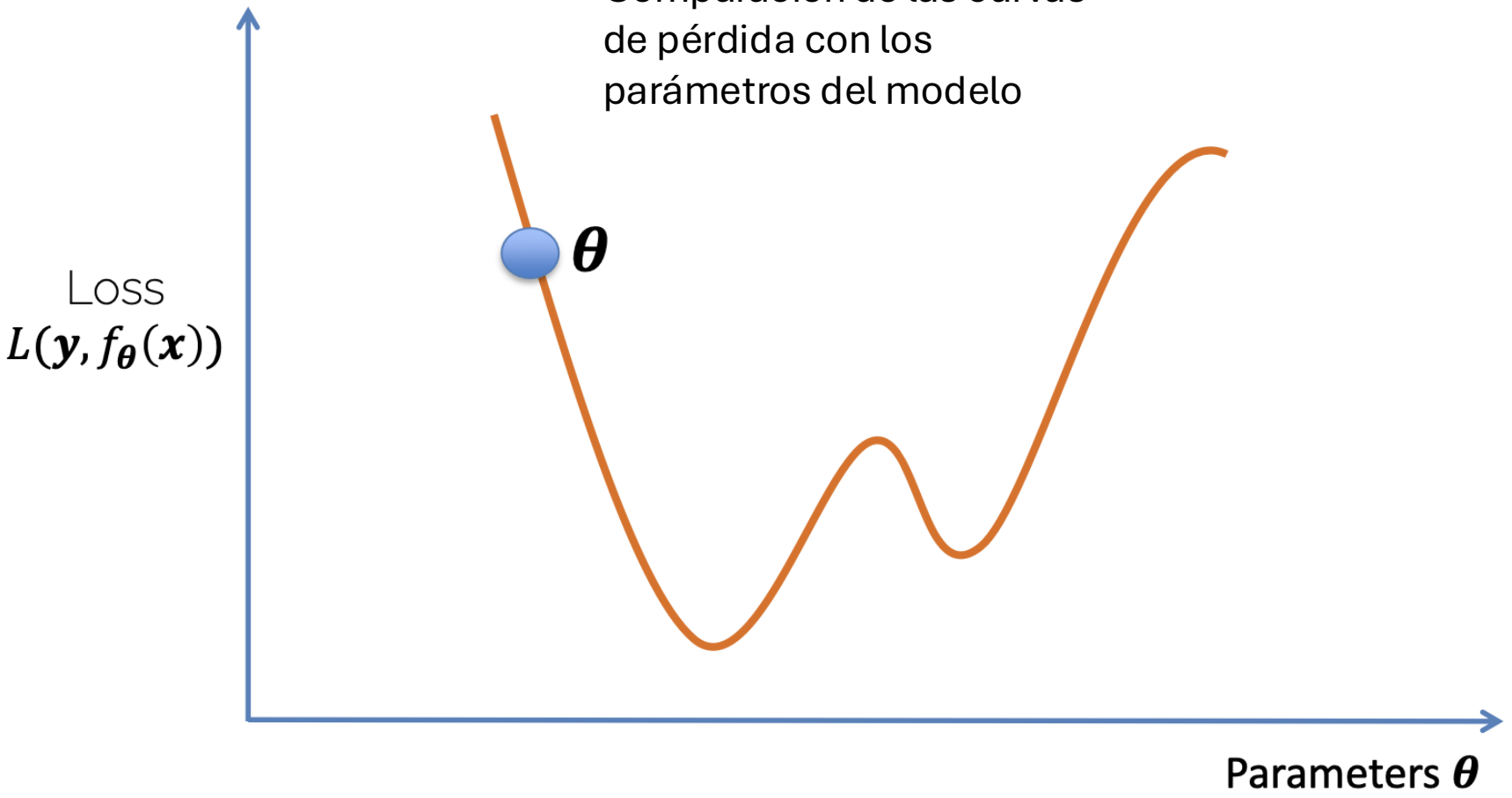


Curva de Entrenamiento



Encontrar mejor NN

Comparación de las curvas de pérdida con los parámetros del modelo



Encontrar mejor NN

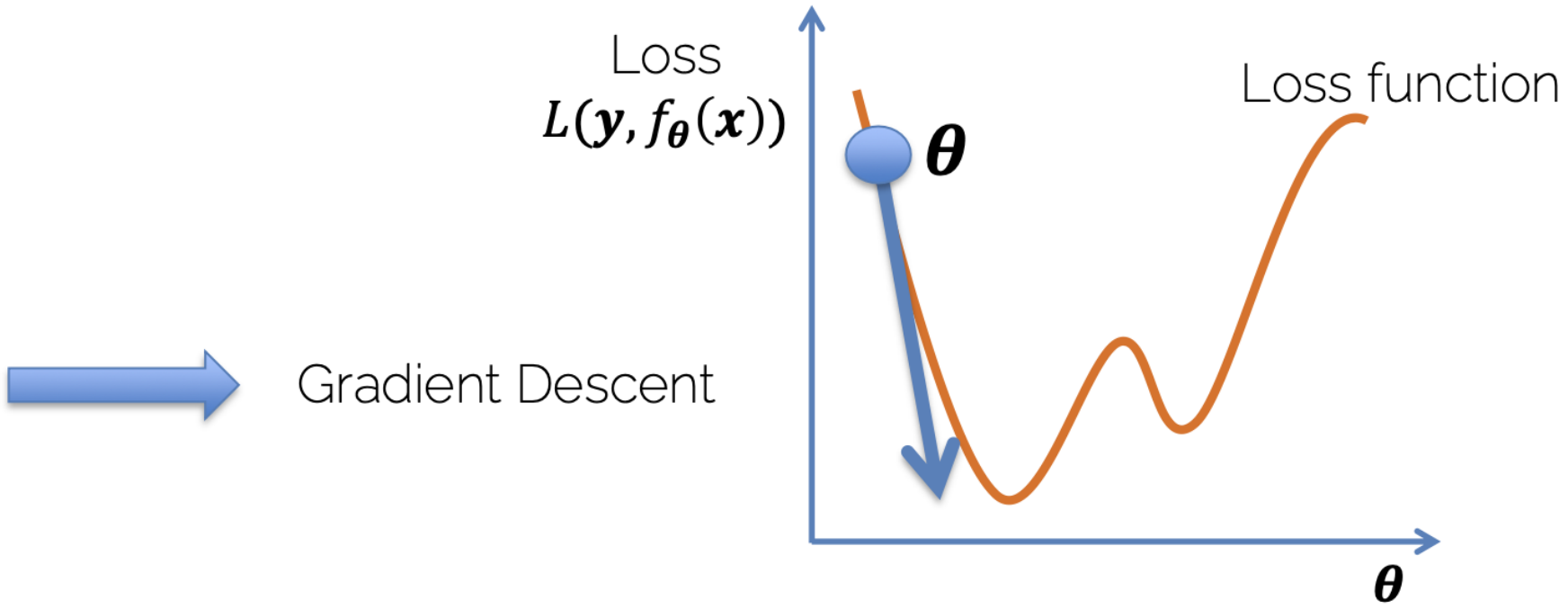
- Loss function: $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$
- Neural Network: $f_{\boldsymbol{\theta}}(\boldsymbol{x})$

Objetivo: minimizar la pérdida con respecto a θ

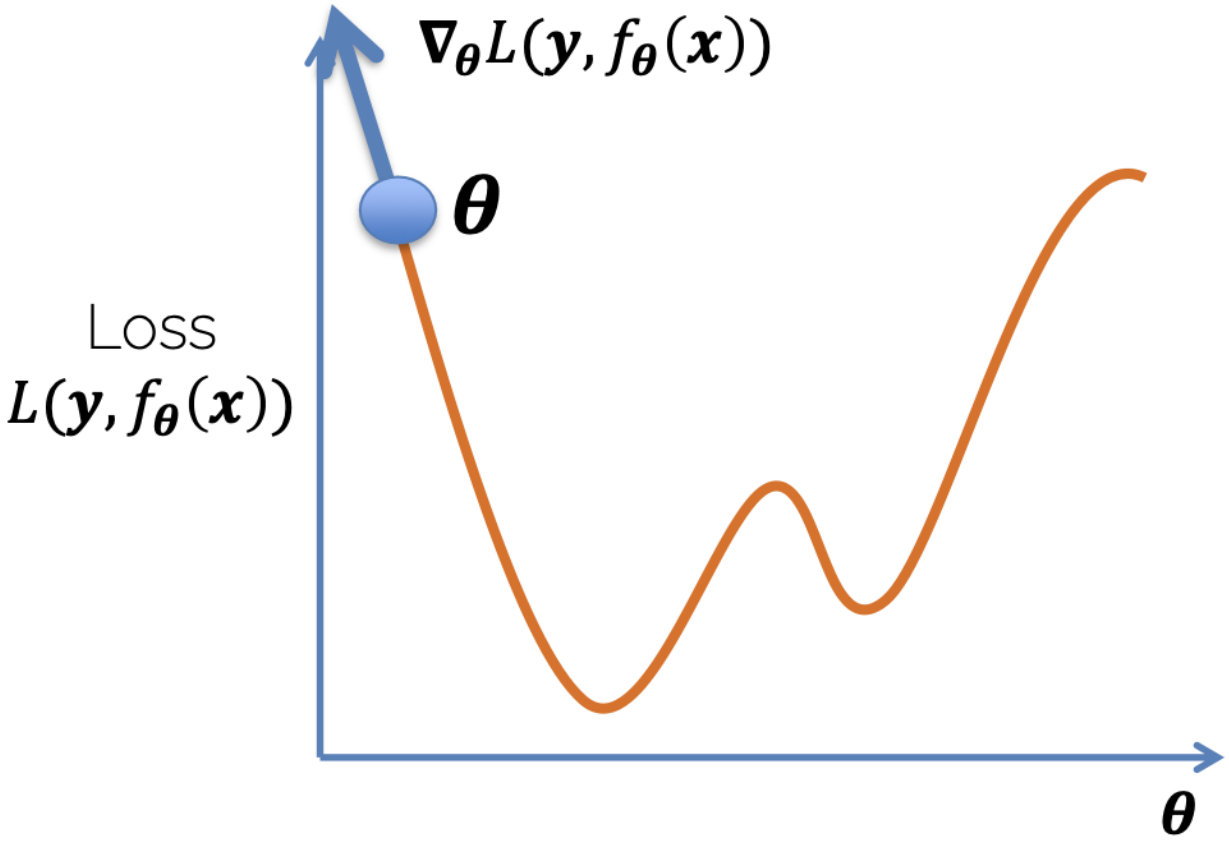
¡Optimización! ¡Entrenamos grafos computacionales con algunas técnicas de optimización!

Encontrar mejor NN

- Gradient-based optimization



Encontrar mejor NN



Learning rate

$$\theta = \theta - \alpha \nabla_\theta L(y, f_\theta(x))$$

$$\theta^* = \arg\min L(y, f_\theta(x))$$

Encontrar mejor NN

- Dadas las entradas $\boldsymbol{x}$ y los objetivos $\boldsymbol{y}$
- Dada una NN de una capa sin función de activación

$$\boldsymbol{f_\theta(x) = Wx}\,, \qquad \boldsymbol{\theta = W}$$

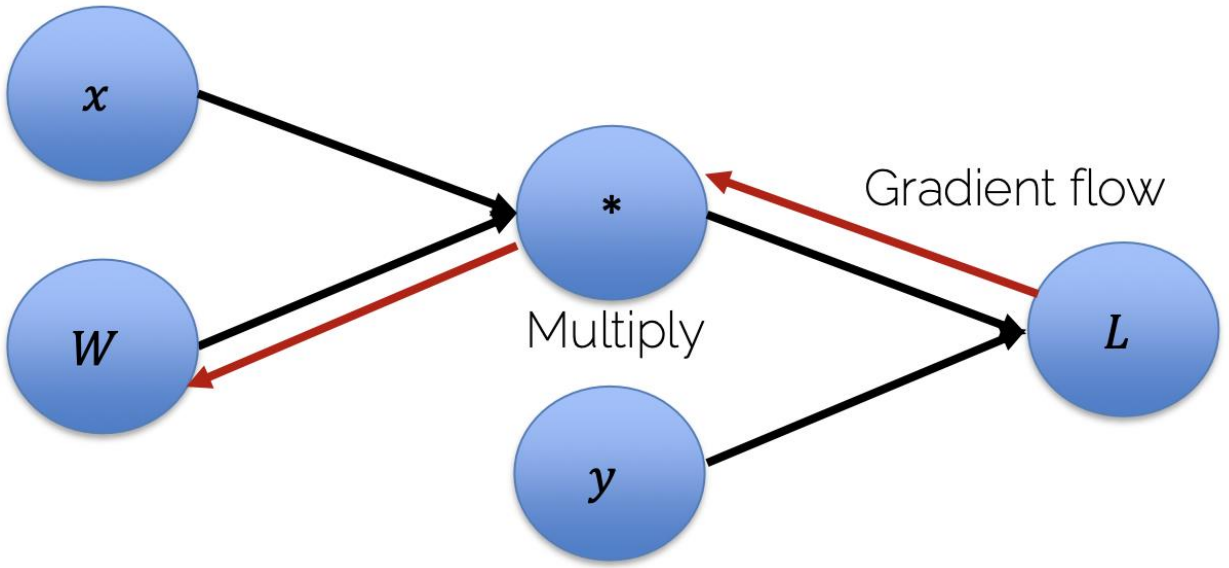
$$\boldsymbol{\theta = \{W, b\}}$$

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||y_i - \hat{y}_i||_2^2$$

Encontrar mejor NN

- Dadas las entradas $\boldsymbol{x}$ y los objetivos $\boldsymbol{y}$
- Dada una NN de una capa sin función de activación

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||y_i - \hat{y}_i||_2^2$$



Encontrar mejor NN

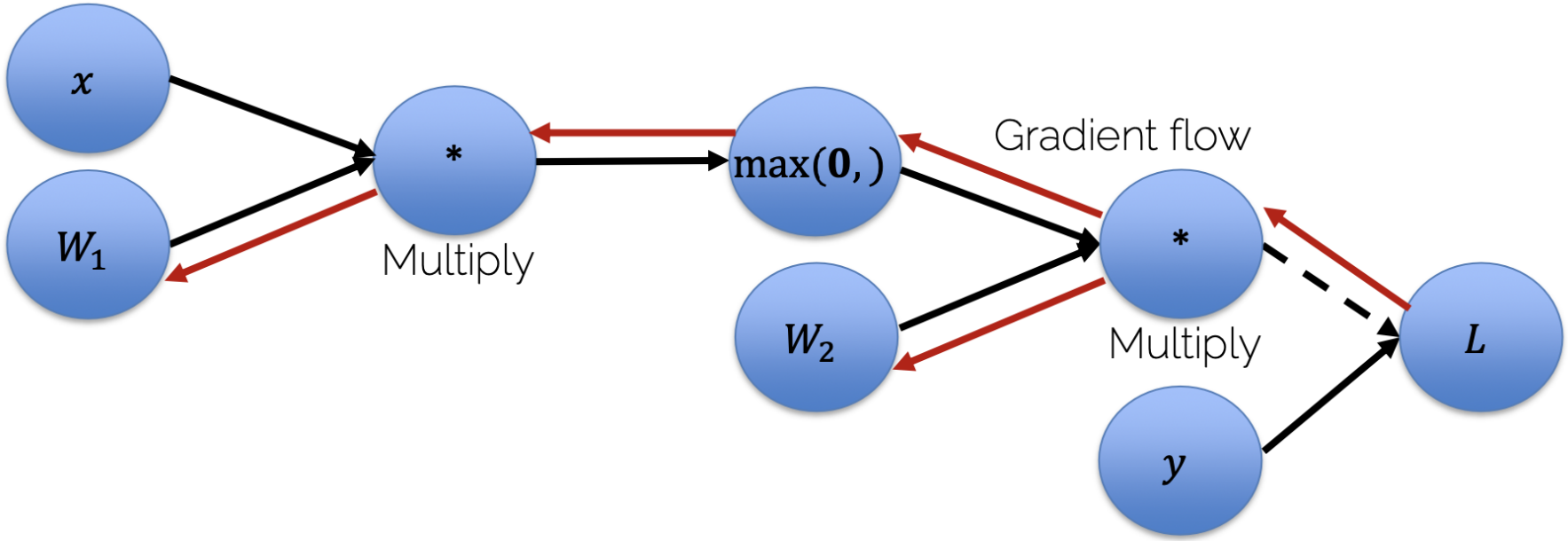
- Dadas las entradas $\boldsymbol{x}$ y los objetivos $\boldsymbol{y}$
- Dada una NN de múltiples capas con muchas activaciones

$$\boldsymbol{f} = \boldsymbol{W_5}\sigma(\boldsymbol{W_4}\tanh(\boldsymbol{W_3},\max(\mathbf{0},\boldsymbol{W_2}\max(\mathbf{0},\boldsymbol{W_1}\boldsymbol{x}))))$$

- Necesidad de propagar gradientes desde el final a la primera capa ($\boldsymbol{W_1}$).

Encontrar mejor NN

- Dadas las entradas x y los objetivos y
- Dada una NN de múltiples capas con muchas activaciones



- Backpropagation: Usar la regla de la cadena para calcular los gradientes

Resumen

- Las redes neuronales son grafos computacionales
- Objetivo: para un conjunto de entrenamiento dado, encontrar los pesos óptimos

- La optimización se realiza mediante soluciones basadas en el gradiente
  - Muchas opciones (más información en las próximas clases)

- Los gradientes se calculan mediante backpropagation
  - Puede modular fácilmente funciones complejas

Próxima clase

Backpropagation y optimización de redes neuronales

Nos vemos el próximo lunes ☺