Ciencia de la
Computación

Redes Neuronales
Multilayer Perceptron

Cristian López Del Alamo
clopezd@utec.edu.pe
IPRODAM3D - Research group

2022

Programa

1. Introducción
2. Multilayer Perceptron
3. Arquitectura
4. Algoritmo Back Propagation
5. Entrenamiento
6. Testing
7. Aplicaciones

1 Introducción

¿Qué hemos visto?

Aprendizaje supervisado (Clasificación)

- Regresión Logística
- SVM
- KNN
- Árboles de Decisión
- Ensemble method
    - Random Forest
    - Boosting
    - Bagging

Métricas

- Matriz de confusión
- Precision, Recall
- F1 - Score

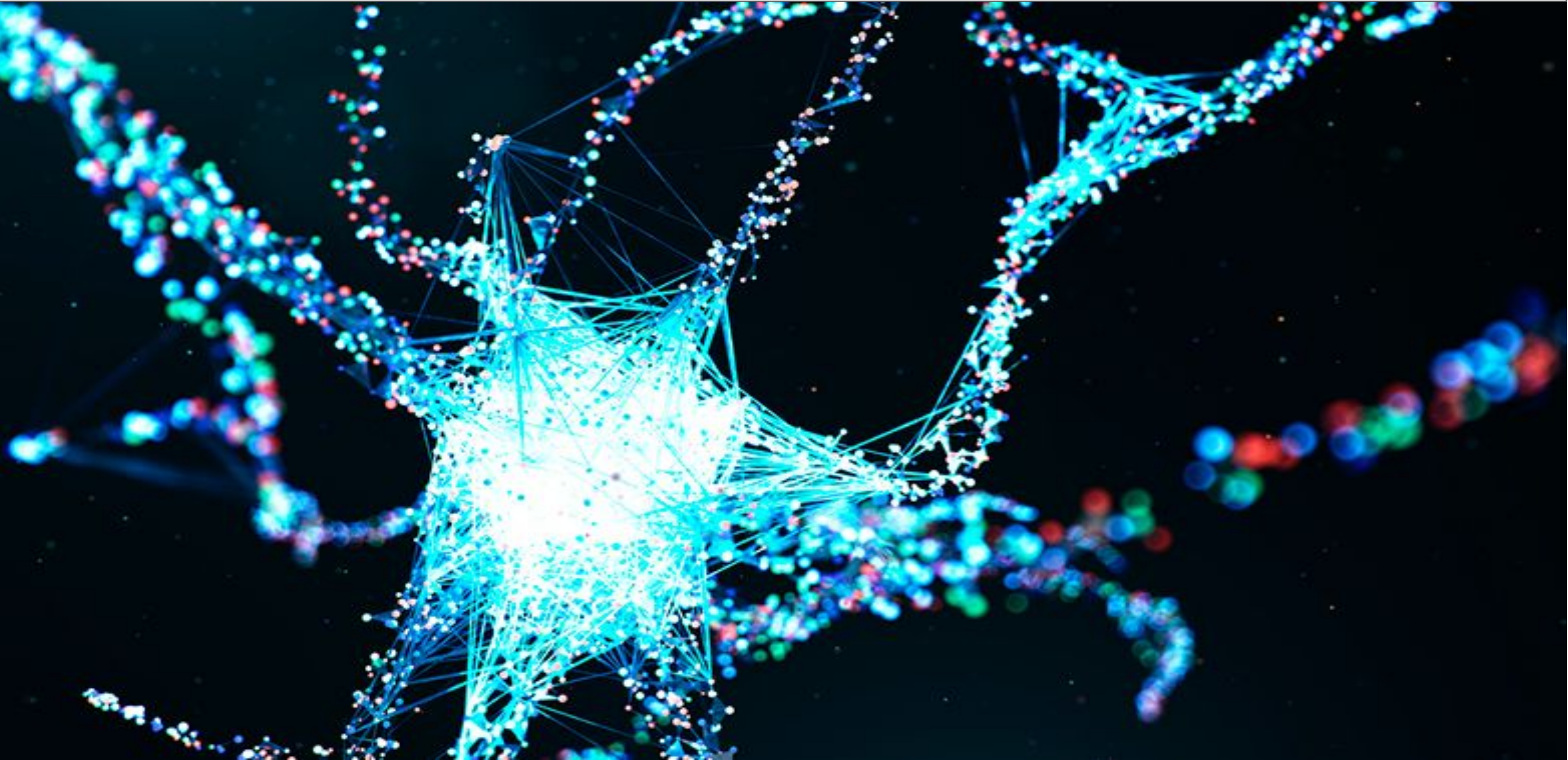
Aprendizaje no supervisado (clustering)

- Kmeans
- DBSCAN
- Mean Shift
- GMM
- Algoritmos Aglomerativos

Métricas

- Silhouette
- Purity
- Entropy
- MI
- Rand Index

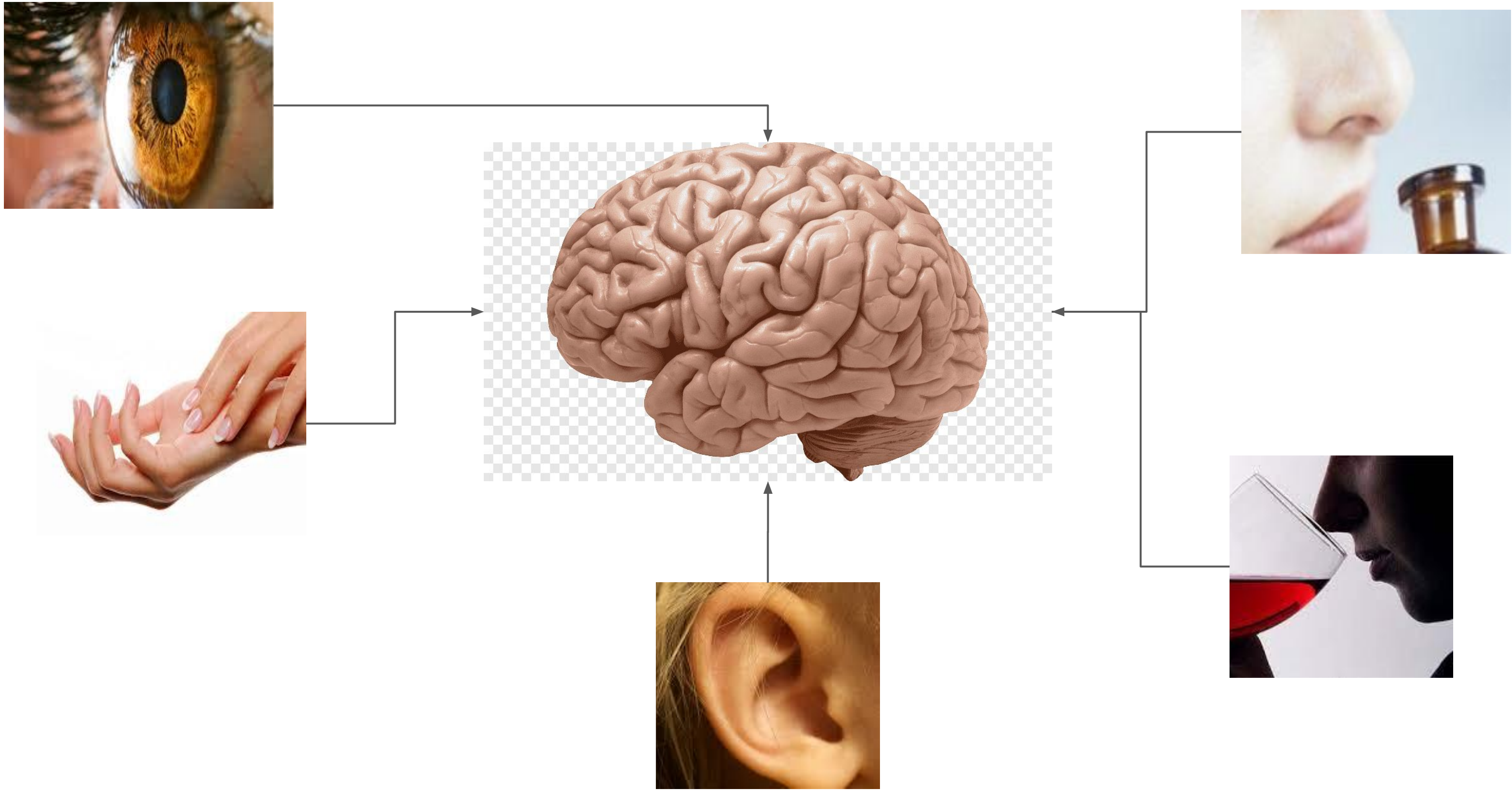
Redes Neuronales Artificiales



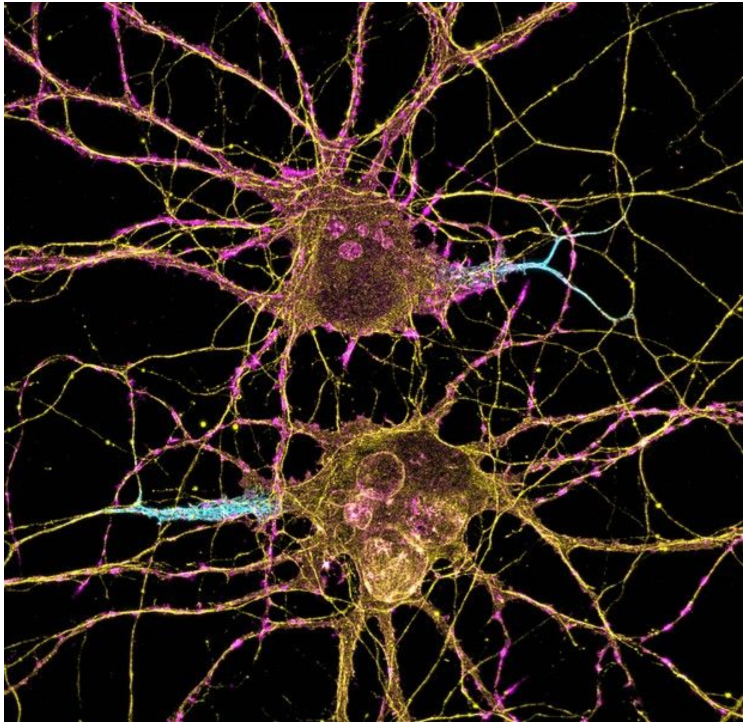
1. Objetivo

- Comprender qué es y cómo funciona una red Neuronal como clasificador.
- Comprender qué le ocurre a un vector cuando lo multiplicamos a una matriz
- Qué hace realmente una MLP a los datos y cómo es que los clasifica.

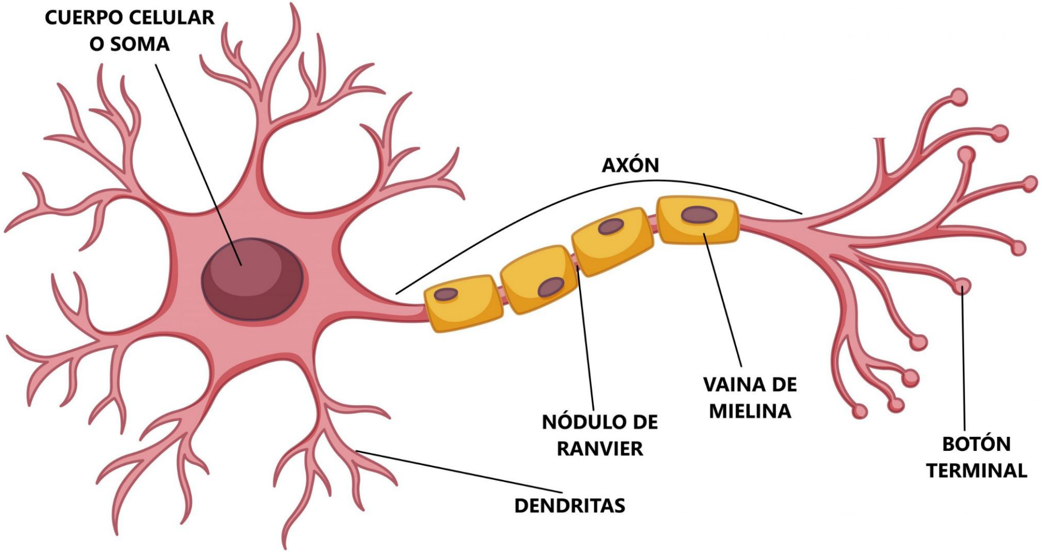
2. Redes Neuronales



2. Redes Neuronales



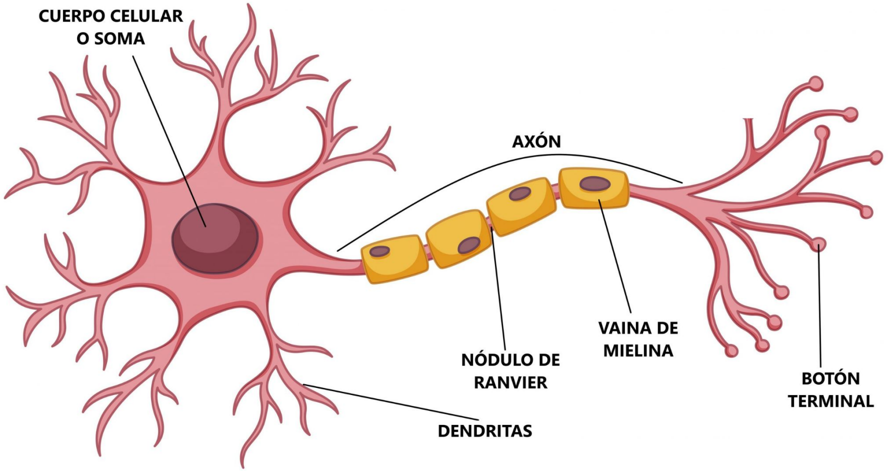
Fuente: [Click]



Fuente: [Click]

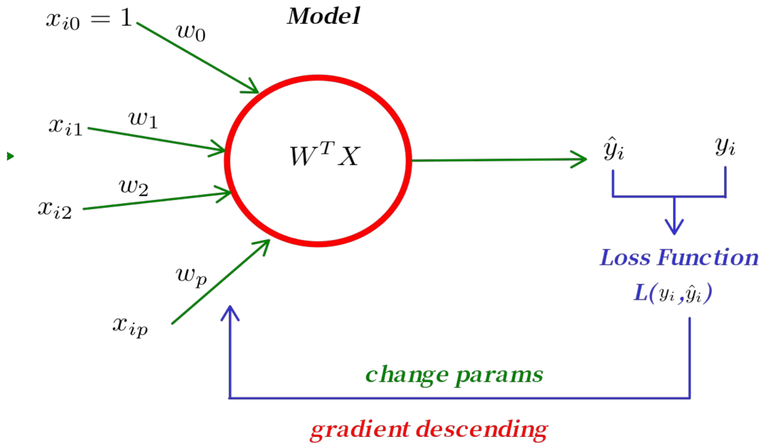
2. Redes Neuronales

Neurona Biológica



Fuente: [Click]

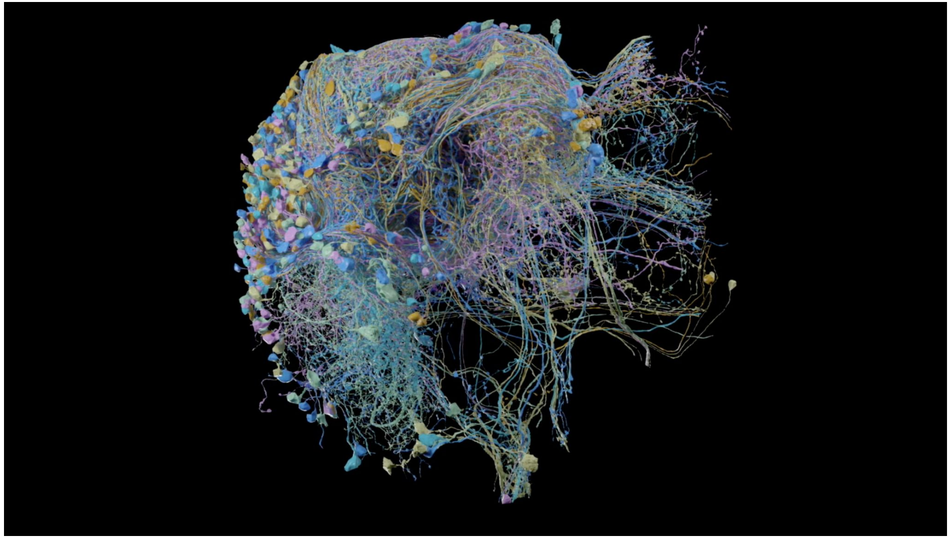
Neurona Artificial



Fuente: [Click]

2. Redes Neuronales

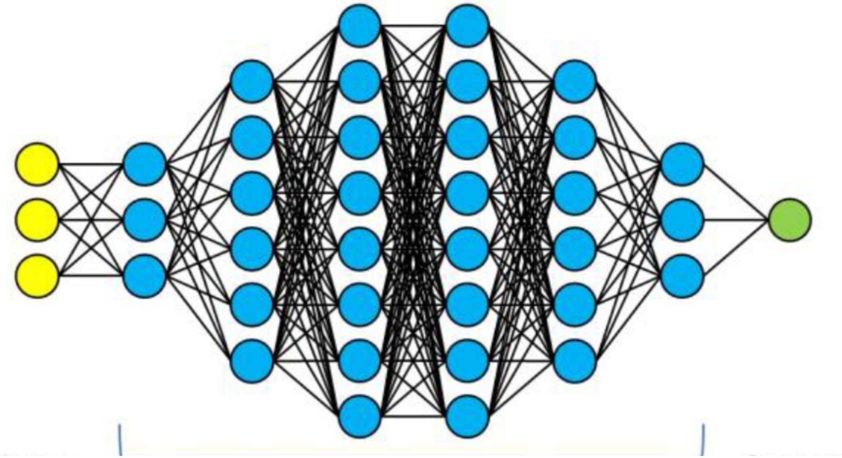
Red Neuronal Biológica



- 100000 neuronas
- Millones de Millones de conexiones

Fuente: Click

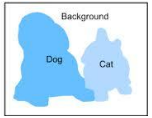
Red Neuronal Artificial



Fuente: Click

2. Redes Neuronales : Aproximador Universal

f() = “Perro”

f() = 

f() = 

f() = 

f() = 

f() = “Gaviota”

2. Redes Neuronales : Aproximador Universal

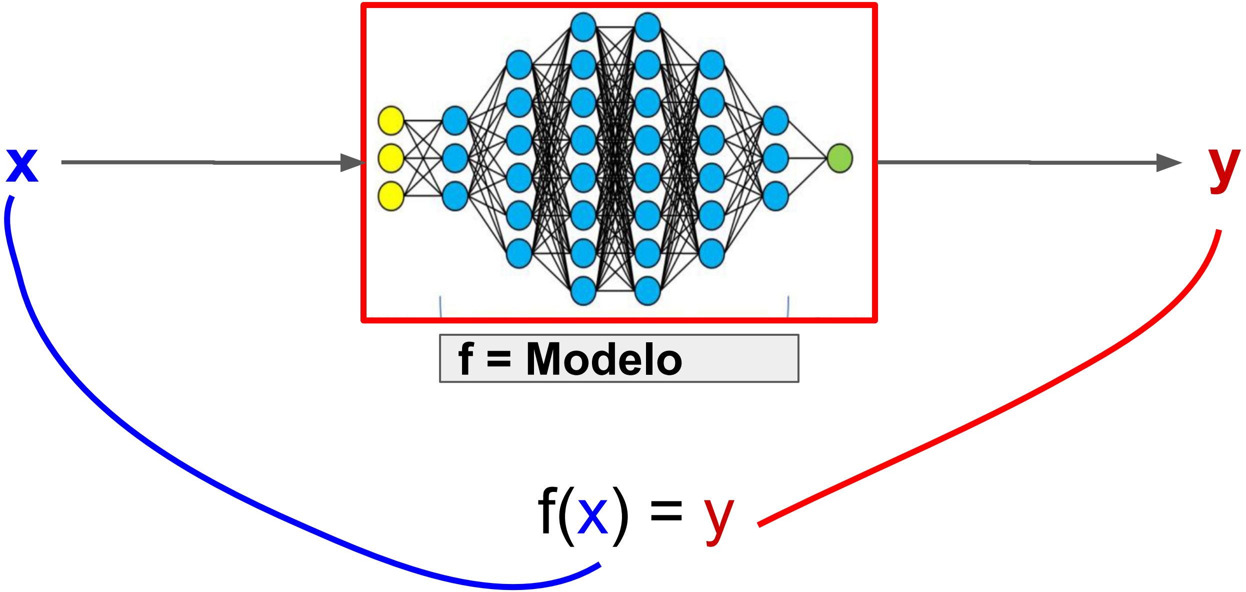
Cualquier **función continua** definida en un **hipercubo unitario n-dimensional** puede ser aproximado por una suma finita de :

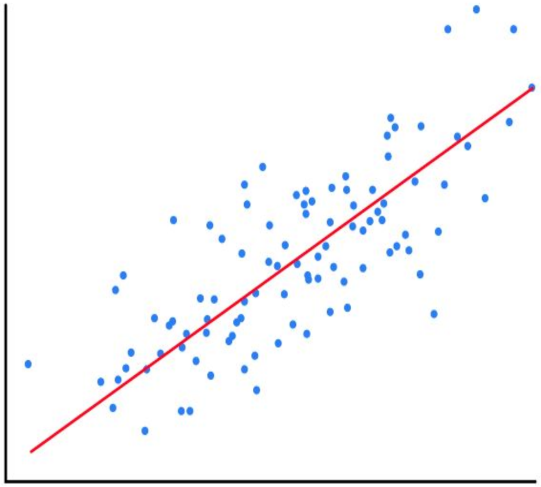
$$\sum_{i=0}^{n} v_i \phi(WX + B),$$

Donde $v_i, b_i \in R, W \in R^n, \ y \ \phi \ es$ una **función continua discriminatoria**

G. Cybenko, 1989

2. Redes Neuronales





Model: $f(x) = w_0 + w_1 x$

Parameters: w_0, w_1

loss: $loss(y, f(x)) = \frac{1}{2n}\sum_{i=0}^{n}(y_i - f(x_i))^2$

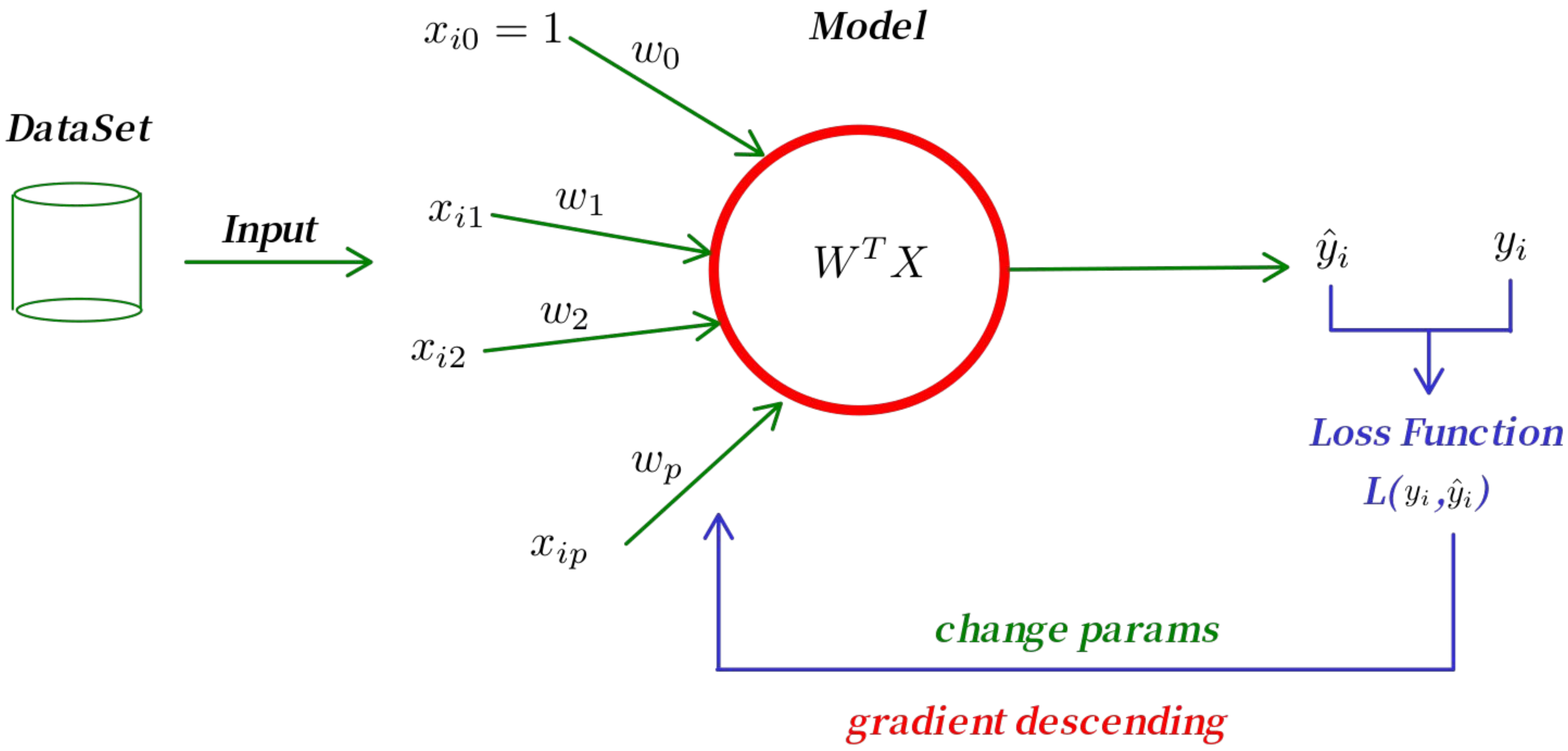
gradiente:

$$\frac{\partial loss}{\partial w_0} = \frac{1}{n}\sum_{i=0}^{n}(y_i - f(x_i))(-1)$$

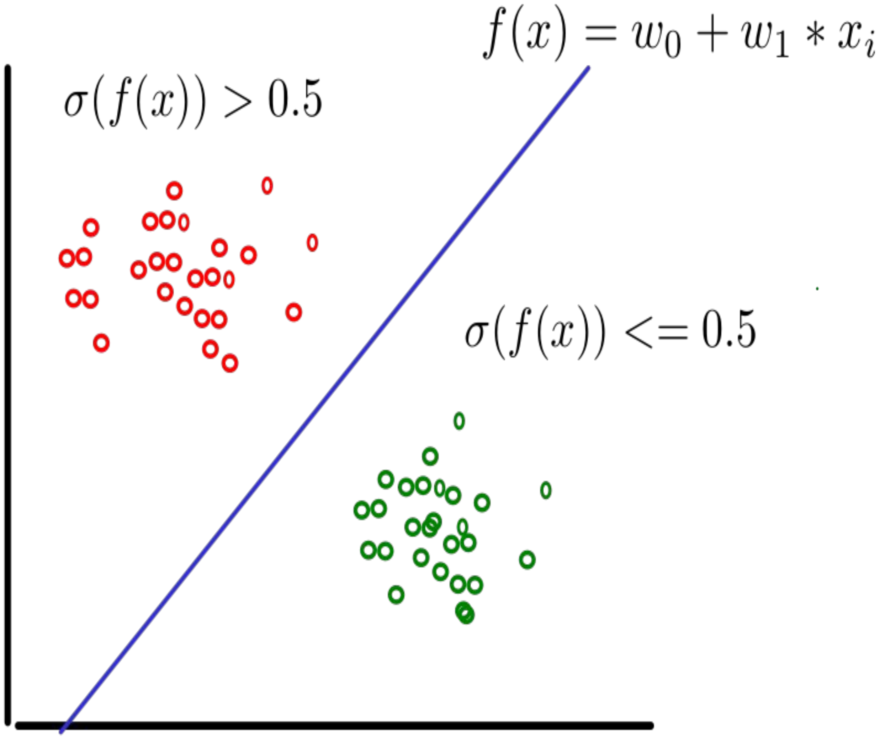
$$\frac{\partial loss}{\partial w_0} = \frac{1}{n}\sum_{i=0}^{n}(y_i - f(x_i))(-x_i)$$

change parameters: $w_i = w_i - \alpha\frac{\partial loss}{\partial w_i}$

Models: Regression Model



2. Redes Neuronales : Regresión Logística



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

Model: $$f(x) = \frac{1}{1+e^{-W^T X}}$$

Parameters: $$w_0, w_1, w_2, ..., w_p$$

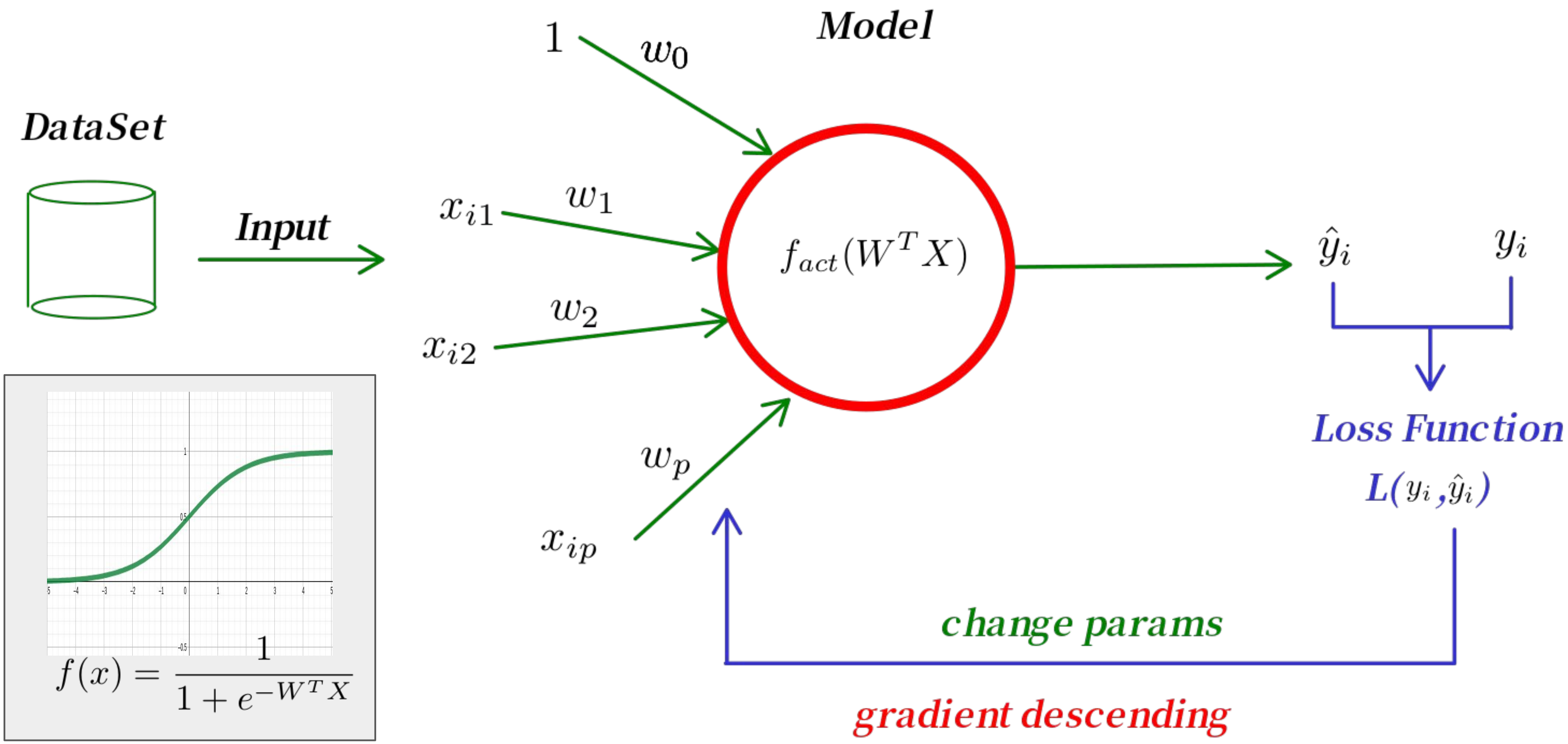
loss:

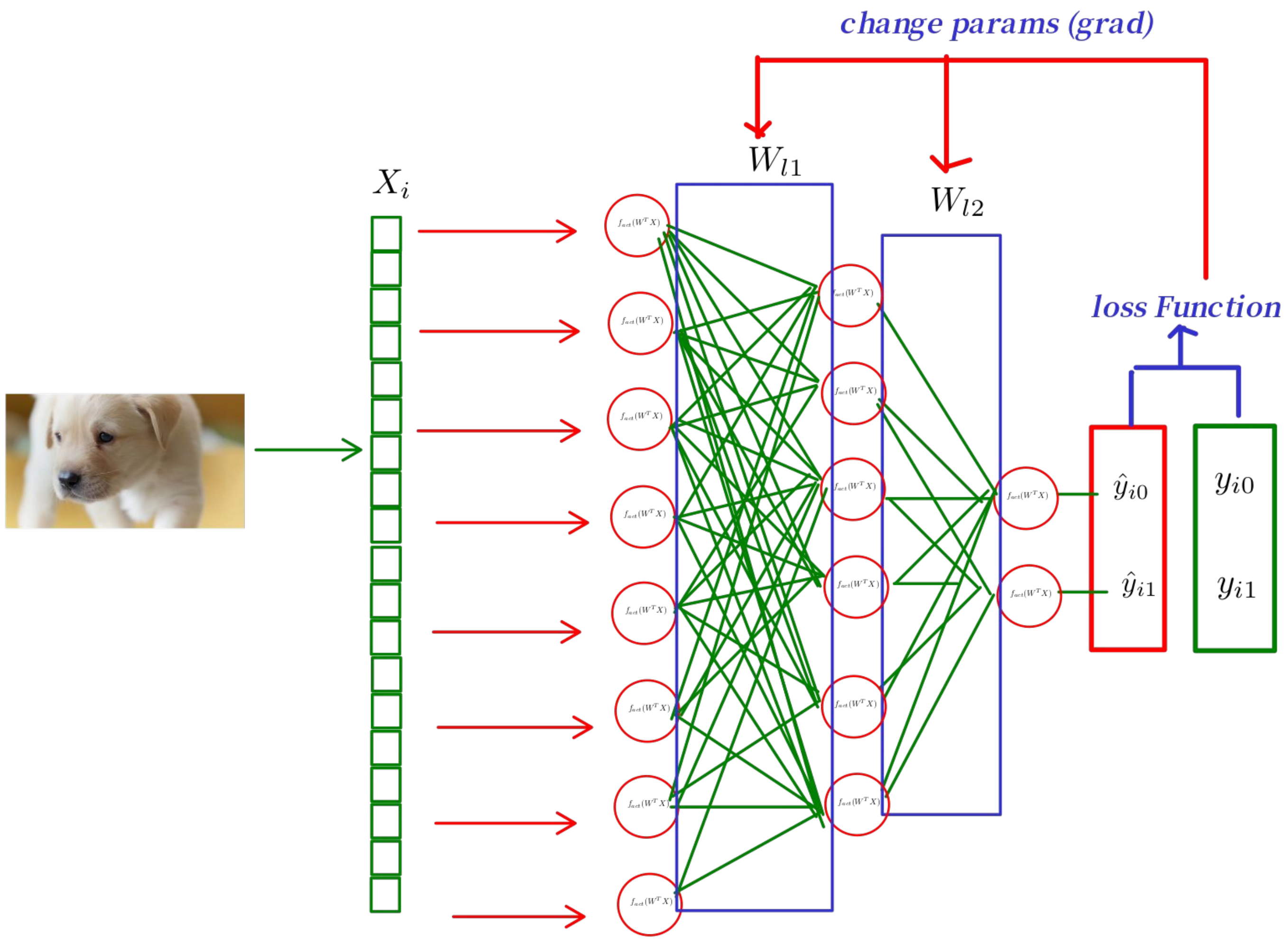
$$loss(y, f(x)) = -\frac{1}{n}\sum_{i=0}^{n}[y_i log(f(x_i)) + (1-y_i) log(1 - log(f(x_i)))]$$

gradiente: $$\frac{\partial loss}{\partial w_j} = \frac{1}{n}\sum_{i=0}^{n}(y_i - f(x_i))(-x_{ij})$$

change parameters: $$w_i = w_i - \alpha\frac{\partial loss}{\partial w_i}$$

2. Redes Neuronales : Regresión Logística

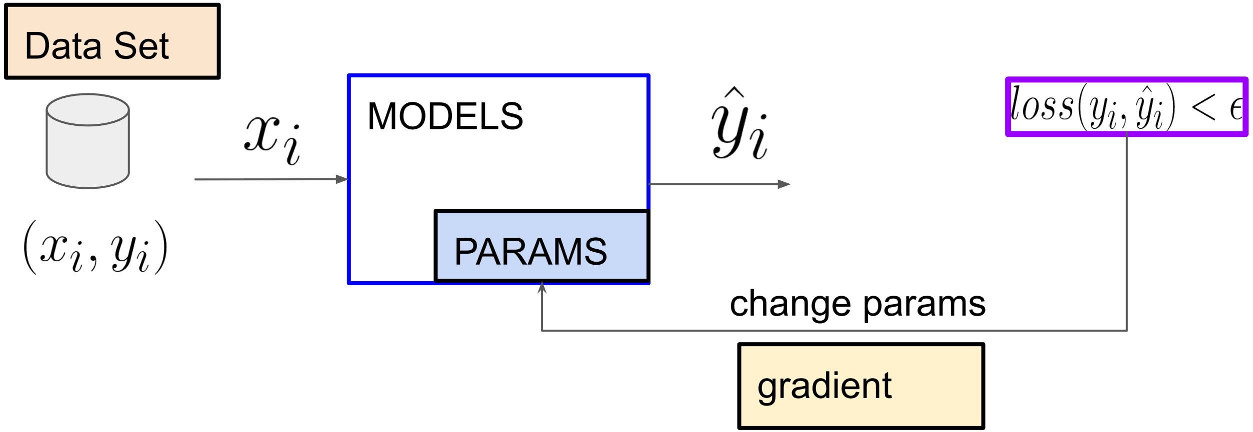




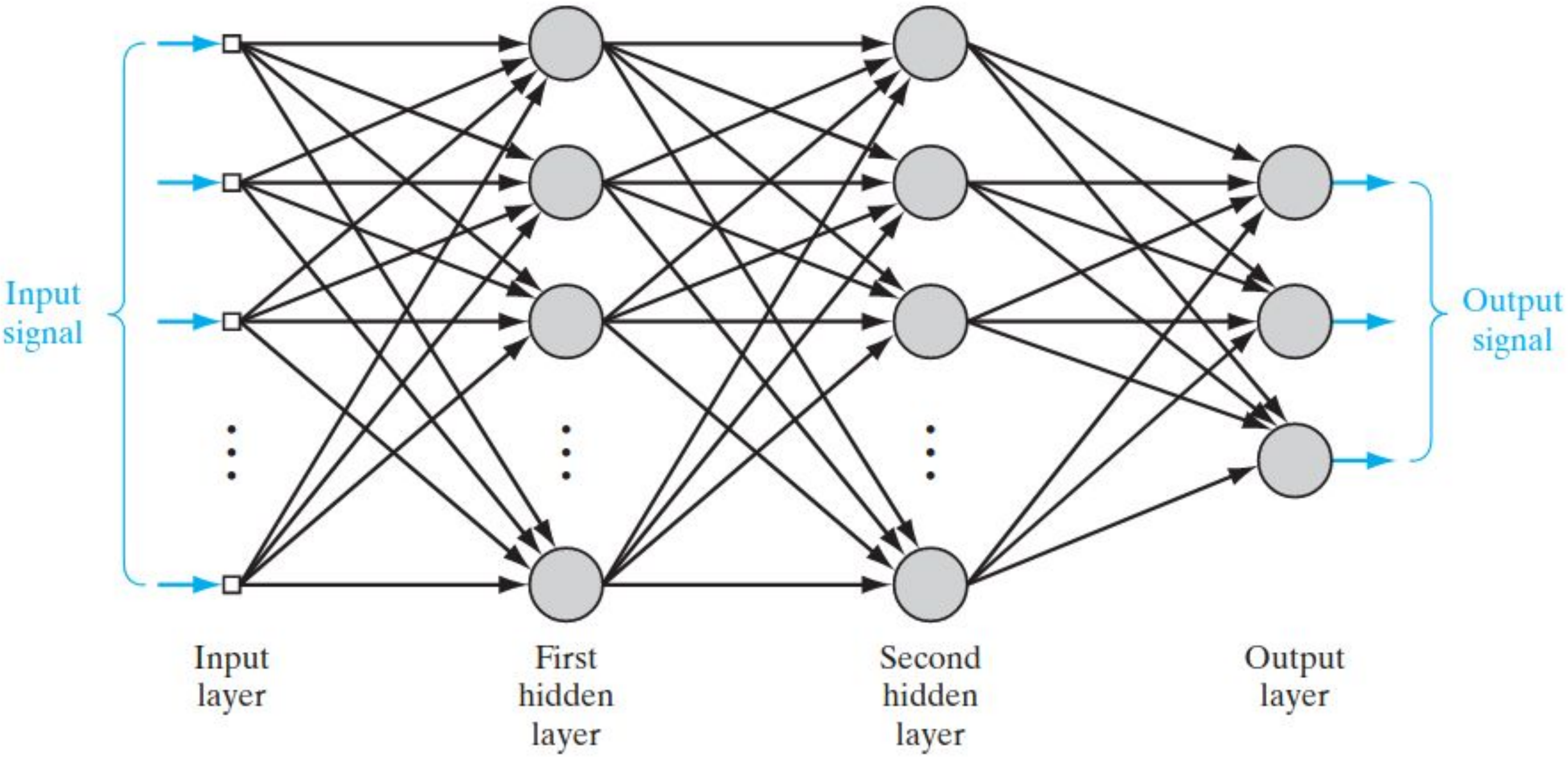
change params (grad)
W_{l1}
W_{l2}
X_i
loss Function
$f_{act}(W^TX)$
$\hat{y}_{i0}$
$\hat{y}_{i1}$
y_{i0}
y_{i1}

Explicación

2. Redes Neuronales : Modelos Supervisados

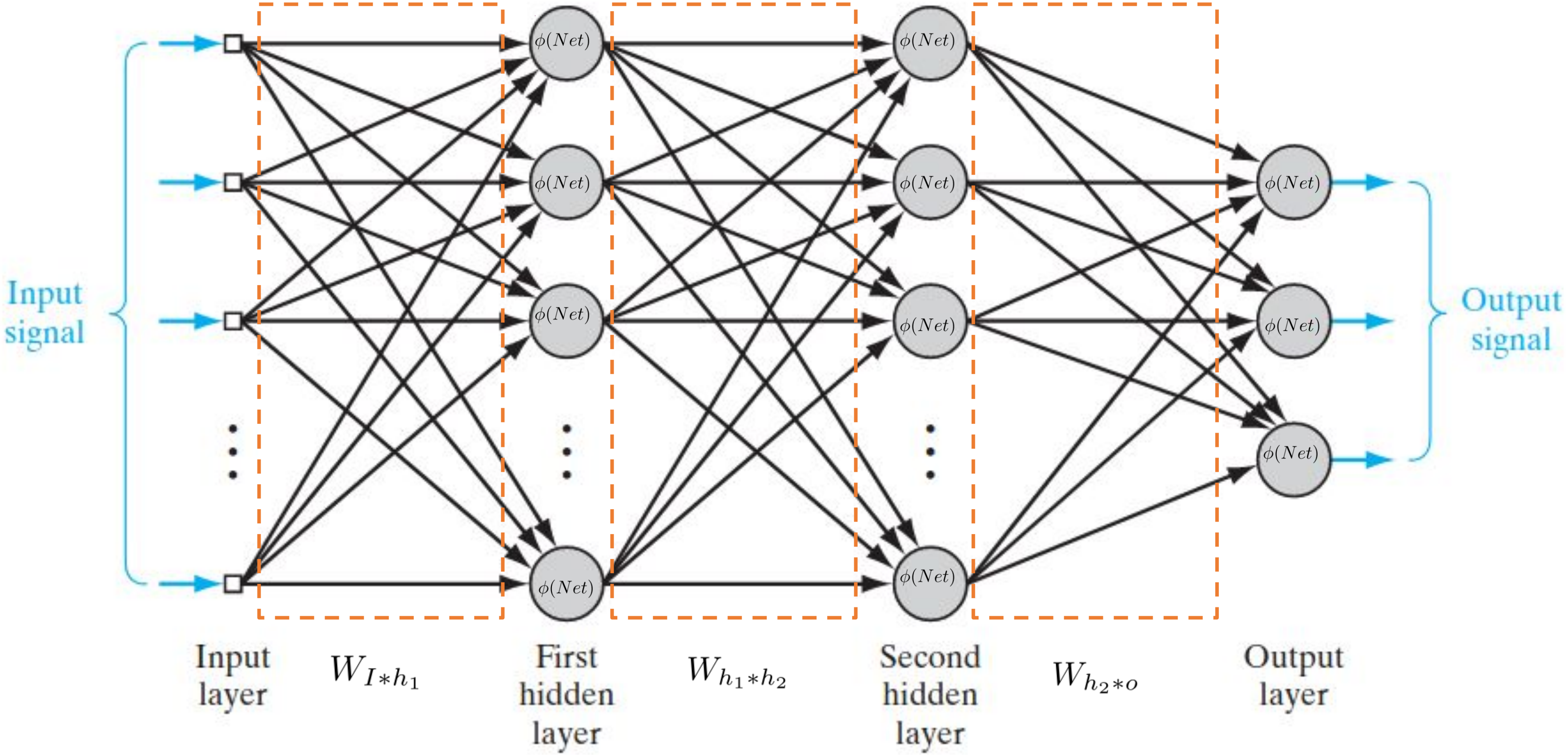


2 Multilayer Perceptron


Input signal
Output signal
Input layer
First hidden layer
Second hidden layer
Output layer

Forward

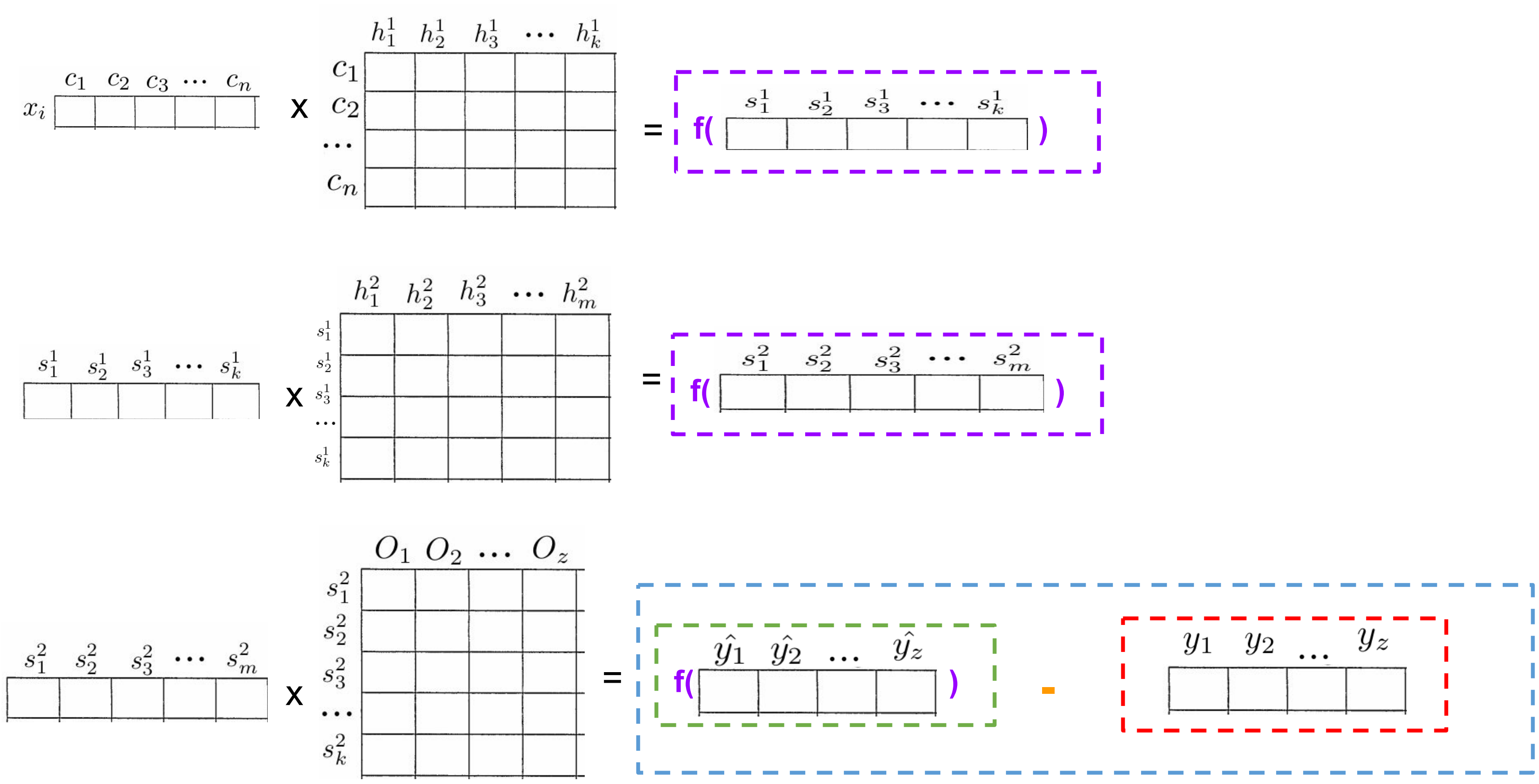
Forward



Input signal
$\phi(Net)$
Output signal
Input layer
W_{I*h_1}
First hidden layer
$W_{h_1*h_2}$
Second hidden layer
W_{h_2*o}
Output layer

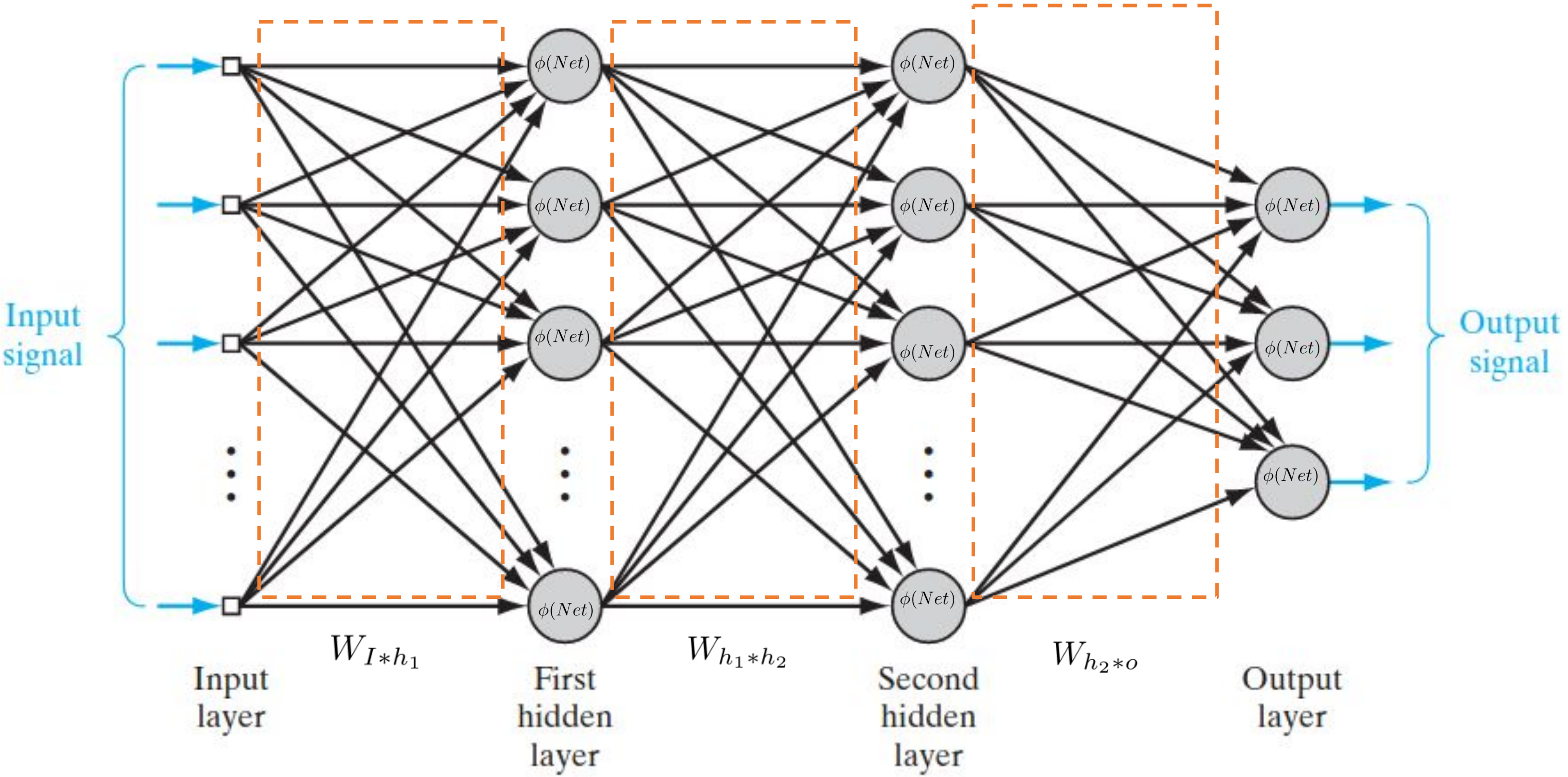
Forward

Forward



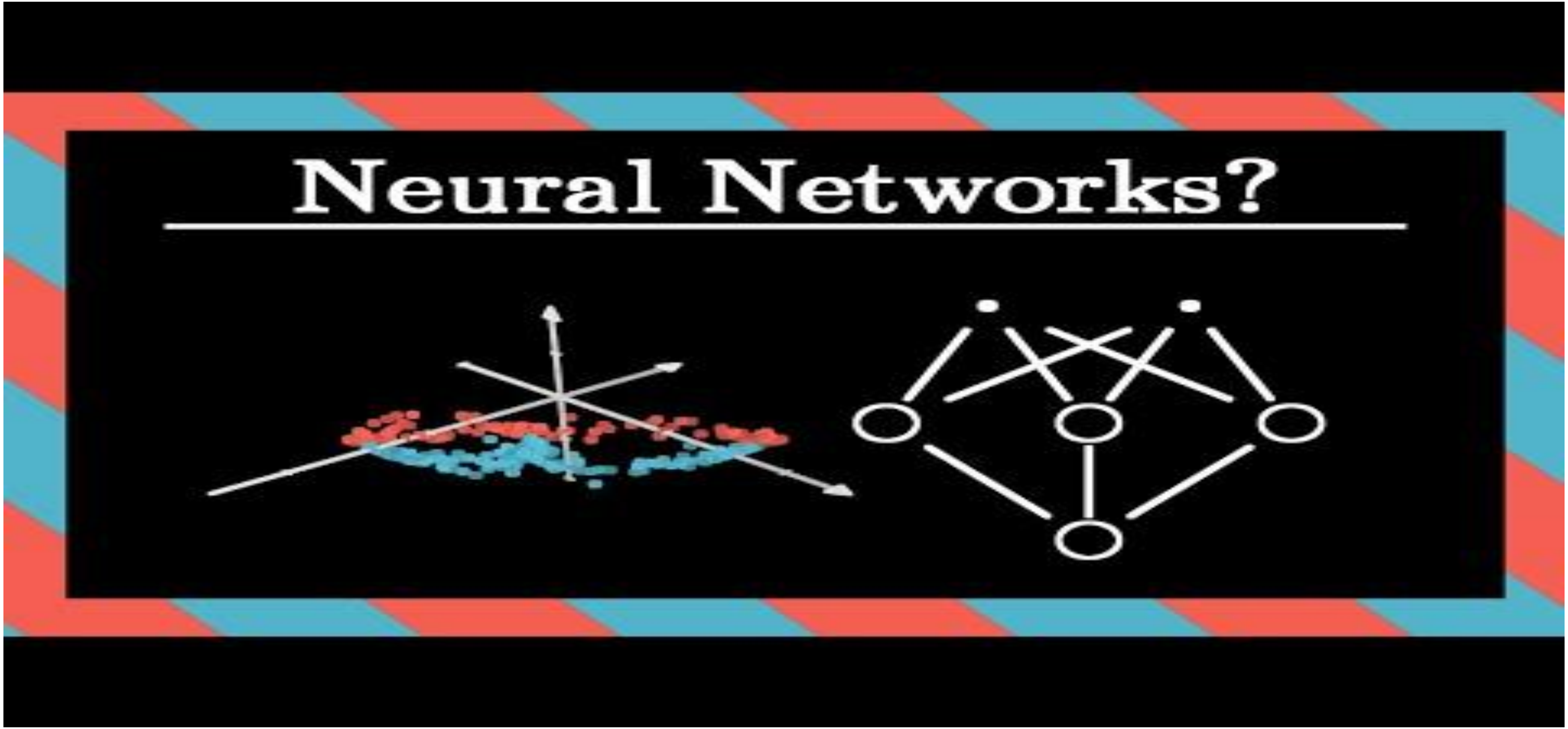
Loss Function (L_2)

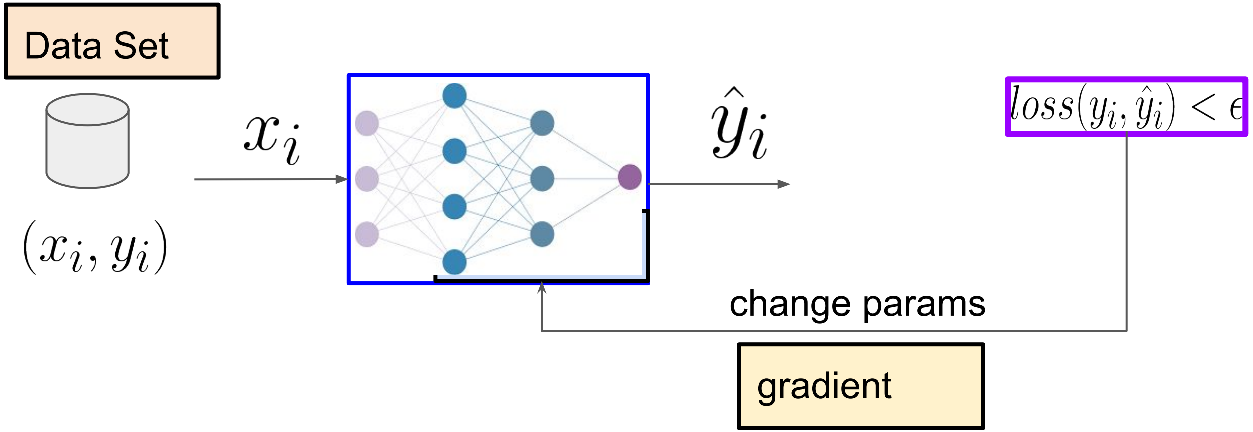
¿Qué realmente está ocurriendo?



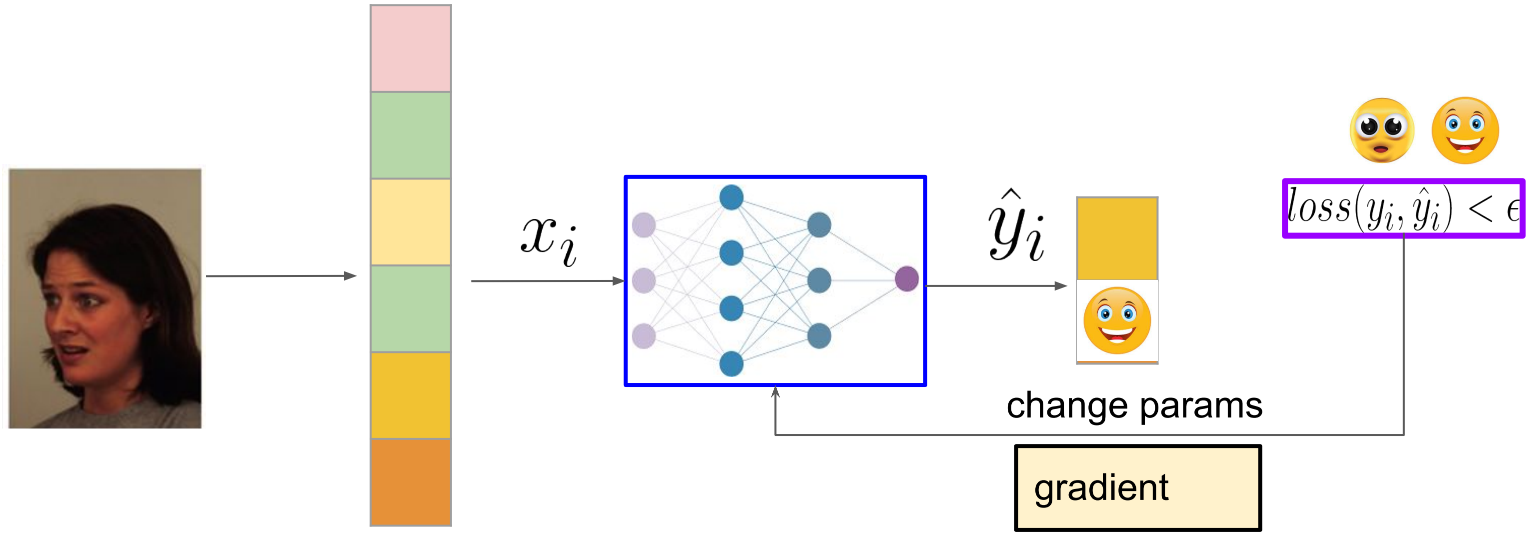
¿Qué le hace una matriz a un vector cuando se multiplican?

Forward

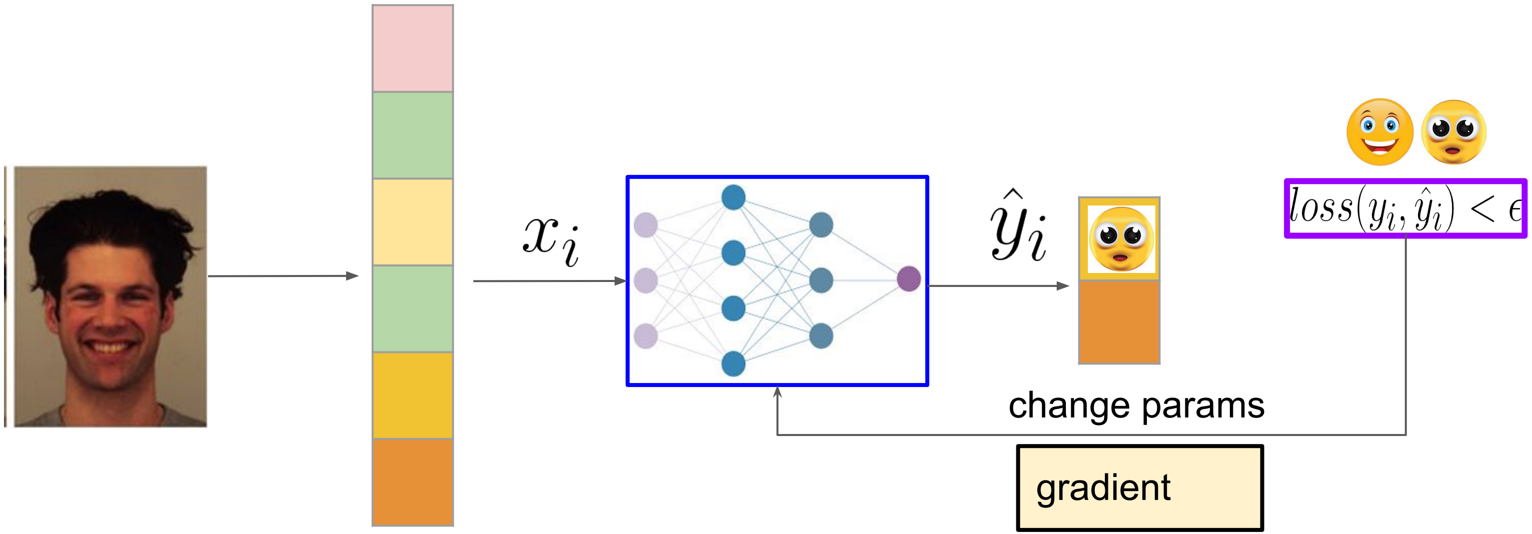



Data Set
(x_i, y_i)
x_i
$\hat{y}_i$
$loss(y_i, \hat{y}_i) < \epsilon$
change params
gradient

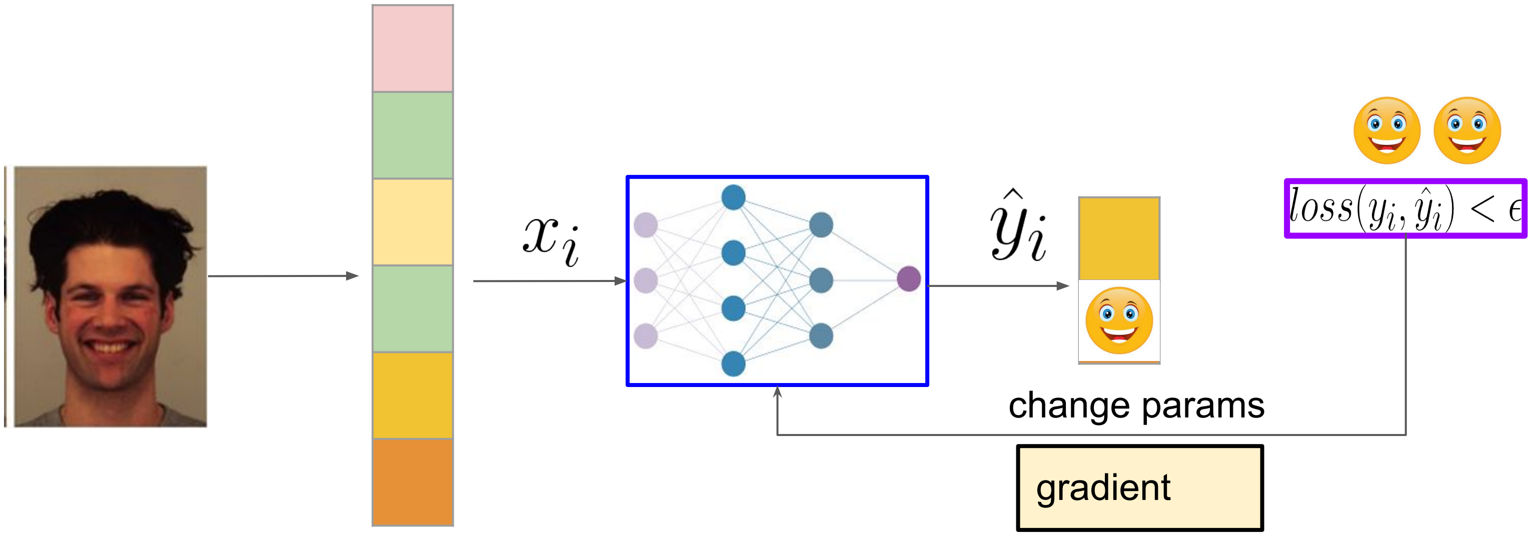
2. Redes Neuronales : Modelos Supervisados



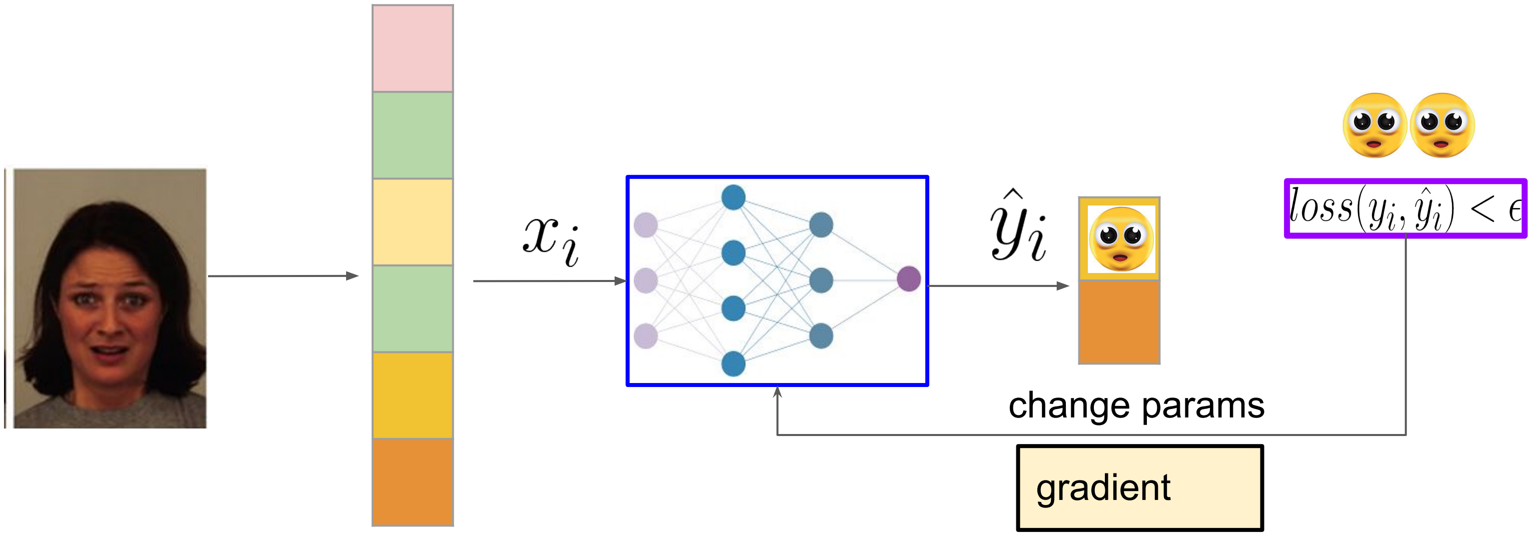
2. Redes Neuronales : **Modelos Supervisados**



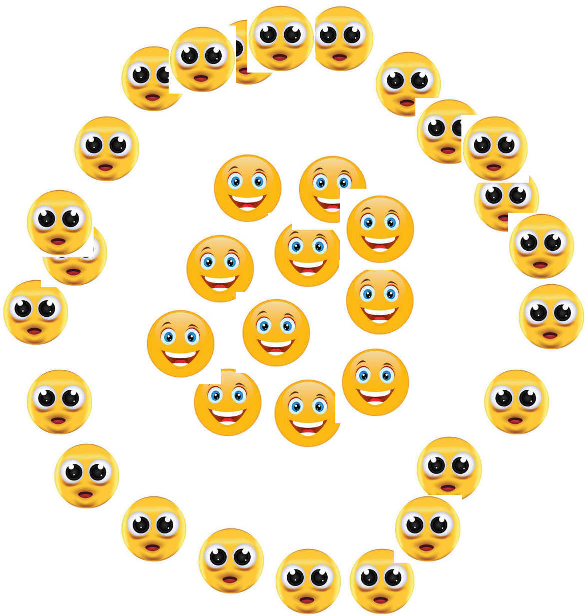
2. Redes Neuronales : Modelos Supervisados



2. Redes Neuronales : **Modelos Supervisados**



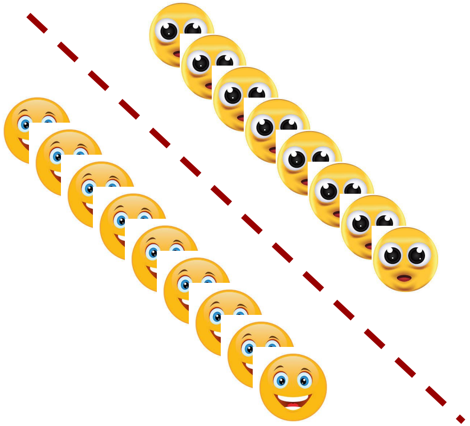
2. Redes Neuronales : Modelos Supervisados



Forward

- Incrementa dimensión
- Rota
- Escala
- Aplica no linealidad

Clasificación



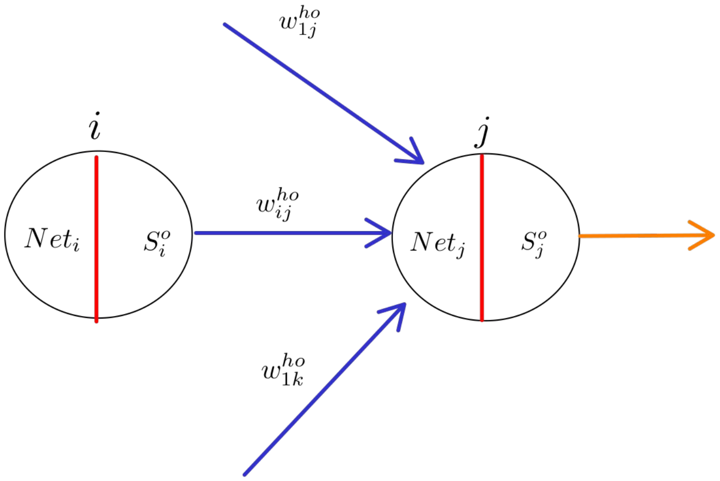
Forward

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & a_{34} & \cdots & a_{3n} \\ a_{41} & a_{42} & a_{43} & a_{44} & \cdots & a_{4n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & a_{nn} \end{pmatrix}$$

Respuesta

BACK PROPAGATION

Retropropagación capa hidden - output



Back Propagation : hidden - output

$$Net_j = s_1^h w_{1j}^{ho} + s_2^h w_{2j}^{ho} + ... + s_i^h w_{ij}^{ho} + ... + s_k^h w_{kj}^{ho}$$

$$s_j = \frac{1}{1+e^{-Net}}$$

$$L = \sum_{i=1}^{n} (s_j^o - s_j^d)/2$$

$$L = (s_1^o - s_1^d)/2 + (s_2^o - s_2^d)/2 + ... + (s_j^o - s_j^d)/2 + ... + (s_{N_o}^o - s_{N_o}^d)/2$$

Retropropagación capa hidden - output

Back Propagation : hidden - output

$$\frac{\partial L}{\partial w_{ij}^{ho}} = \boxed{\frac{\partial L}{\partial s_j^o}} * \boxed{\frac{\partial s_j^o}{\partial Net_j}} * \boxed{\frac{\partial Net_j}{\partial w_{ij}^{ho}}}$$

Regla de la cadena

Retropropagación capa hidden - output

Back Propagation : hidden - output

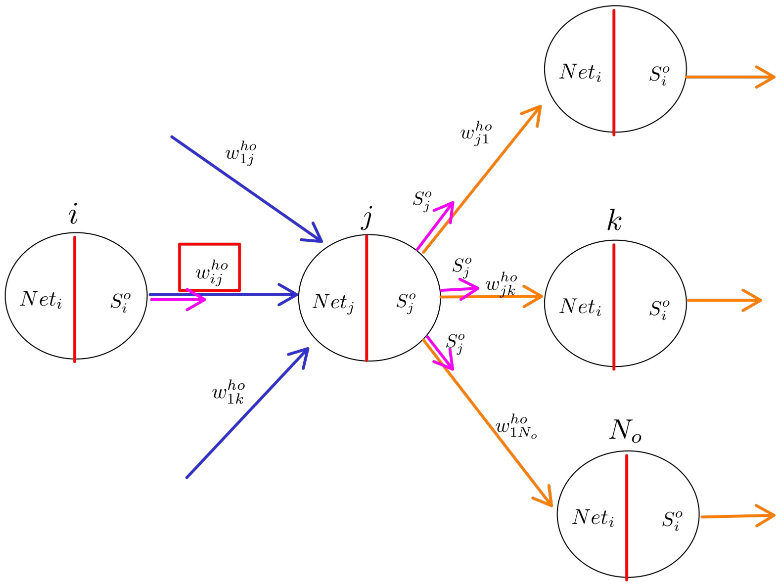
$$\frac{\partial L}{\partial s_j^o} = (s_j^o - s_j^d)$$

$$\frac{\partial s_j^o}{\partial Net_j} = s_j^o(1 - s_j^o)$$

$$\frac{\partial Net_j}{\partial w_{ij}^{ho}} = s_i^h$$

$$\frac{\partial L}{\partial w_{ij}^{ho}} = \boxed{(s_j^o - s_j^d)}\boxed{s_j^o(1 - s_j^o)}\boxed{s_i^h}$$

Retropropagación capa hidden - hidden



Back Propagation : hidden - hidden

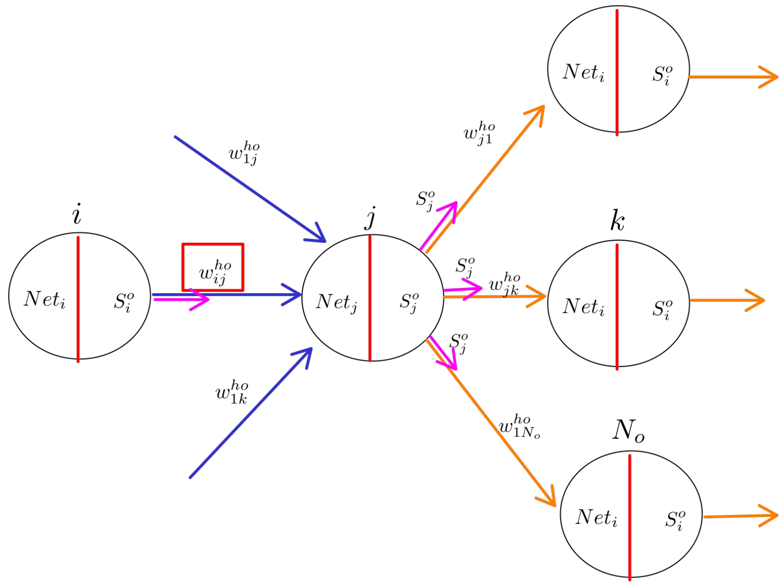
$$\frac{\partial L}{\partial w_{ij}^{hh+1}} = \sum_{k=1}^{N_o} \boxed{\left(\frac{\partial L}{\partial s_k^o} * \frac{\partial s_k^o}{\partial Net_k}\right)} * \frac{\partial s_j^{h+1}}{\partial Net_j} * \frac{\partial Net_j}{\partial w_{ij}^{hh+1}}$$

$$\frac{\partial L}{\partial w_{ij}^{ho}} = \boxed{\frac{\partial L}{\partial s_j^o} * \frac{\partial s_j^o}{\partial Net_j}} * \frac{\partial Net_j}{\partial w_{ij}^{ho}}$$

$$\delta_j = (s_j^o - s_j^d) s_j^o (1 - s_j^o)$$

$$\frac{\partial L}{\partial w_{ij}^{hh+1}} = \boxed{\sum_{k=1}^{N_o} (\delta_k)} * \frac{\partial s_j^{h+1}}{\partial Net_j} * \frac{\partial Net_j}{\partial w_{ij}^{hh+1}}$$

Retropropagación capa hidden - output



Back Propagation : hidden - output

$$\frac{\partial L}{\partial w_{ij}^{hh+1}} = \sum_{k=1}^{N_o} \left(\frac{\partial L}{\partial s_k^o} * \frac{\partial s_k^o}{\partial Net_k} \right) * \boxed{\frac{\partial s_j^{h+1}}{\partial Net_j}} * \boxed{\frac{\partial Net_j}{\partial w_{ij}^{hh+1}}}$$

$$\frac{\partial L}{\partial w_{ij}^{hh+1}} = \sum_{k=1}^{N_o} (\delta_k) * \boxed{\frac{\partial s_j^{h+1}}{\partial Net_j}} * \boxed{\frac{\partial Net_j}{\partial w_{ij}^{hh+1}}}$$

$$\frac{\partial L}{\partial w_{ij}^{hh+1}} = \sum_{k=1}^{N_o} (\delta_k) \boxed{s_j^{h+1}(1 - s_j^{h+1})} \boxed{s_i^h}$$

7 Aplicaciones

PyTorch