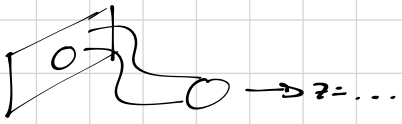
Redes Neuronales

Modelo de aproximación de funciones (levemente) inspiradas a las neuronas.

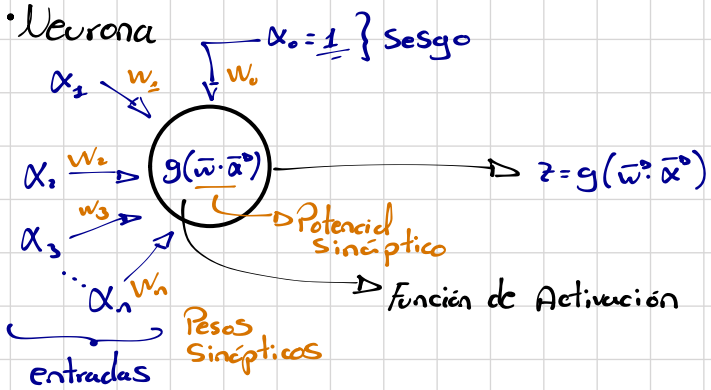
- Sirve para problemas de regresión y clasificación.

- Frank Rossenblatt (1957-1969)



En base a una imagen, adivinar cuál es el número (perceptrones).

- Neurona

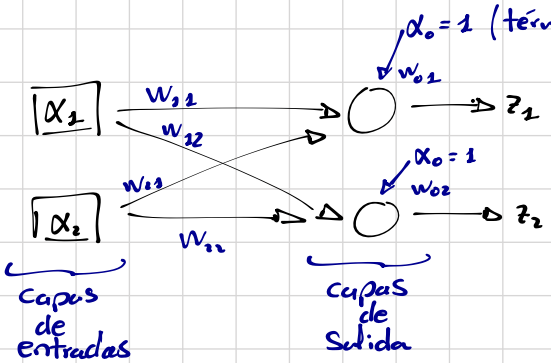


$\vec{x} = (x_0, x_1, \ldots, x_n)$

$\vec{w} = (w_0, w_1, \ldots, w_n)$

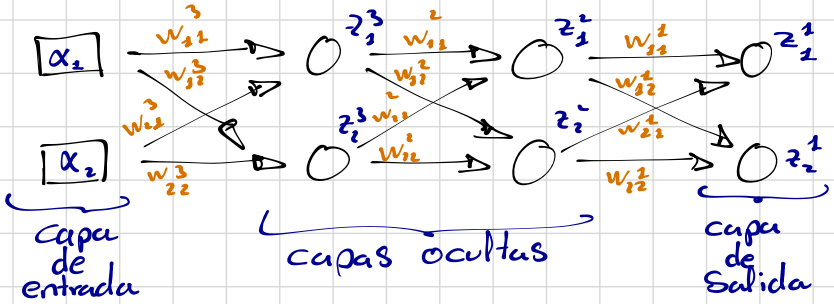
- Perceptrón Simple (porque solo tiene una capa de neuronas).
  - Conjunto de capas (capas de salida y capas de entradas) todas conectadas entre sí.

  Las neuronas no tienen conexiones entre sí (independientes).



- Perceptrón Multicapa.

Me dice que características necesito.



La idea sería que a cada capa, que función usar, en base al problema que se nos plantee.

Capa de Salida

Problema

Regresión	Clasificación
↓	↓
Función Lineal	Función Logística

Capas Ocultas

Problema

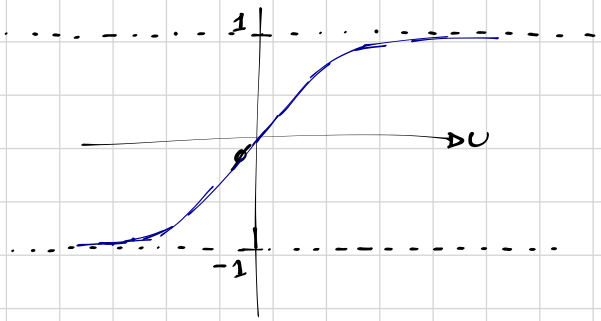
Regresión	Clasificación
↓	↓
Función Lineal	Función Logística

↓ No tiene sentido

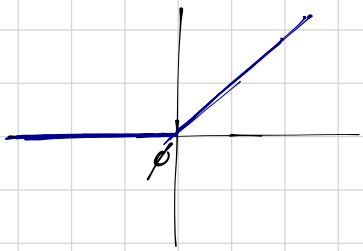
↳ Para concatenar funciones lineales entre varias capas ocultas, lo hacemos en la última y listo (capa salida).

↓ otras funciones

- tangente hiperbólica

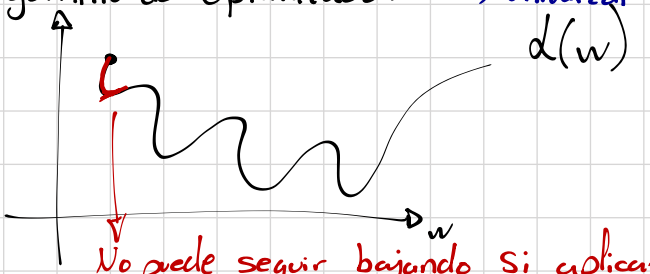


- Relu (rectified linear unit)



- Función de pérdida a optimizar (por defecto)
  - Regresión → Error Cuadrático Medio
  - Clasificación → Entropía cruzada binaria

- Algoritmo de Optimización → Minimizar errores



No puede seguir bajando si aplicamos el gradiante decreciente.

↳ Utilizamos una alternativa → Gradiente Estocástico

↳ MIN $\alpha(\vec{w}) = \frac{1}{2}\left(y^i - h_{\vec{w}}(\vec{x}_i)\right)^2$ → SE CALCULA PARA UN PESO ALEATORIO.

↳ Para calcular la corrección de los pesos de una capa, y gracias al mismo, corregir el error del resto de pesos.

↳ $\vec{w} \leftarrow \vec{w} + \alpha \cdot \frac{d\alpha(\vec{w})}{d\vec{w}}$

Algoritmo de retropropagación de error (Backprop).