Redes Neuronales

Introducción (TeX)

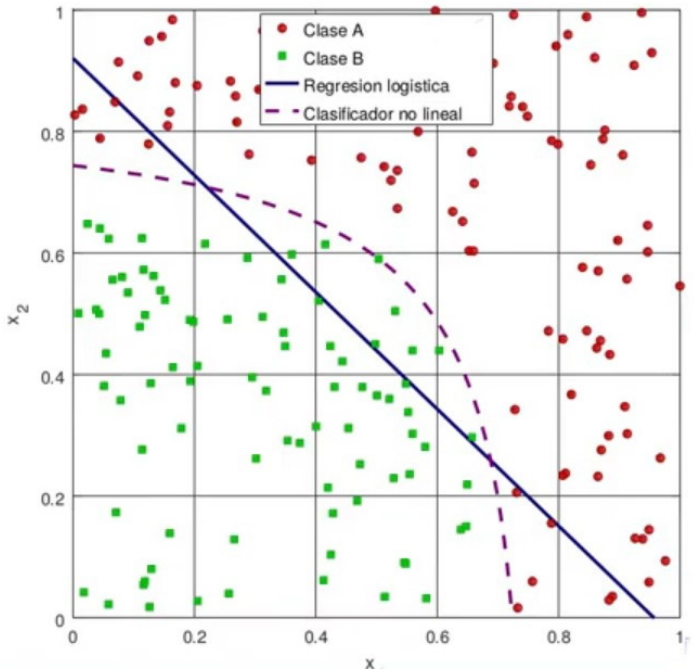
Enero-Julio 2021

Contenido

Introducción

Regresión lógistica

- Primer clasificador que vimos: regresión logística (RL)
- En RL hipótesis: $h_{\underline{\theta}}(\underline{\mathbf{x}}) = 1/(1 + e^{-\underline{\theta}^T \underline{\mathbf{x}}})$
- Predicción usa $h_{\underline{\theta}}(\underline{\mathbf{x}}) > 1/2$ para predecir una determinada clase, es decir $\underline{\theta}^T \underline{\mathbf{x}} = 0$ es la **frontera de decisión**
- Esta hipótesis solo permite partir espacio de entrada en dos con una línea recta o hiperplano:



- ¿Cómo clasificar con fronteras de decisión no lineales?
  - Podemos diseñar hipótesis más complejas
  - Por ejemplo

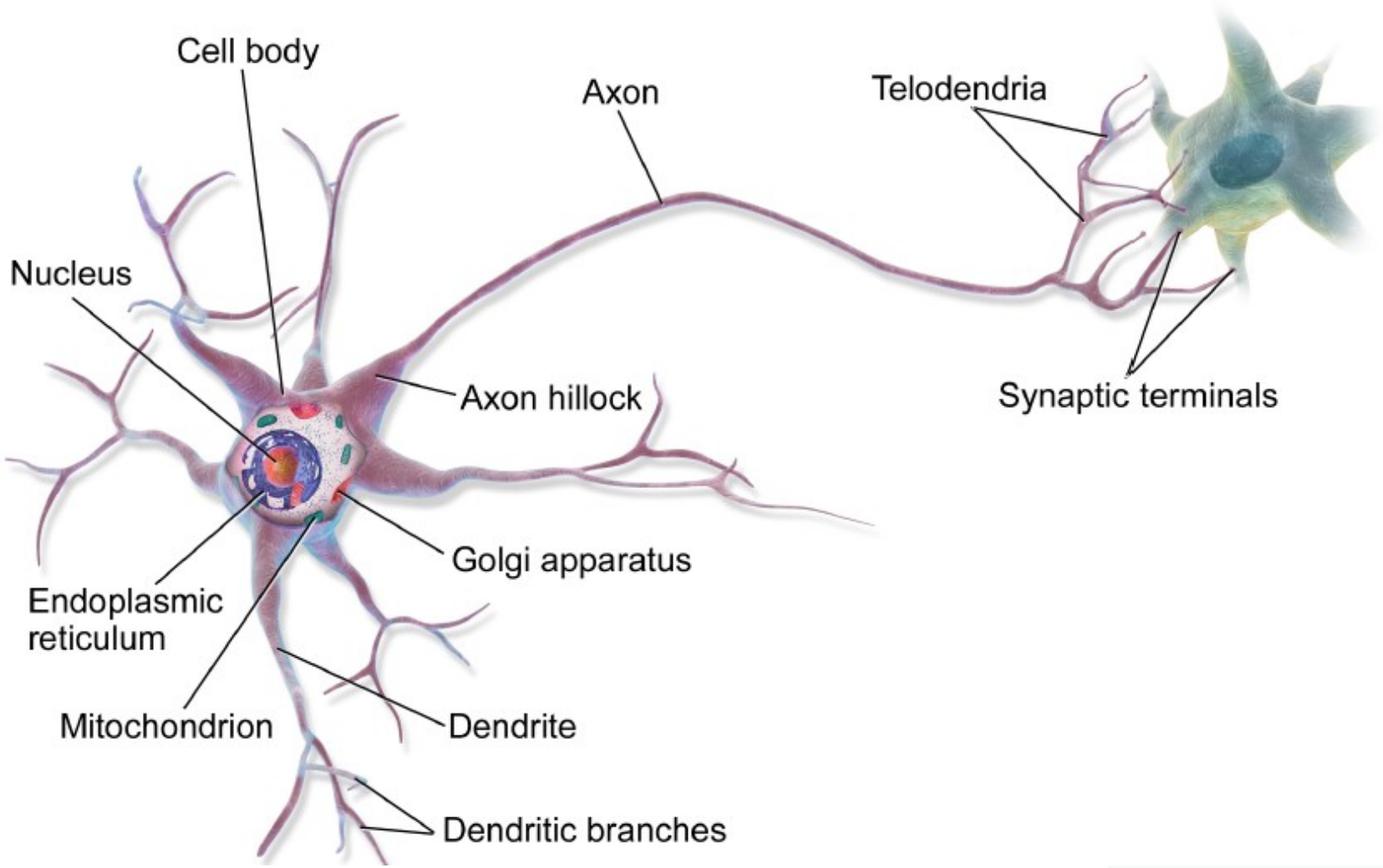
$$h_{\underline{\theta}}(\underline{\mathbf{x}}) = \frac{1}{1 + e^{-\underline{\theta}^T \phi(\underline{\mathbf{x}})}}$$

    donde $\phi(\underline{\mathbf{x}})$ mapea $\underline{\mathbf{x}}$ a un espacio de mayor dimensión.
  - El caso particular

$$\phi\left(\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T\right) = \begin{bmatrix} 1 & x_1 & x_1^2 & x_2 & x_2^2 & x_1 x_2 \end{bmatrix}^T$$

    proyecta de dos a seis dimensiones.
  - Es necesario diseñar $\phi(\underline{\mathbf{x}})$
- Redes neuronales son otra opción.

Redes neuronales biológicas



Redes neuronales biológicas

- Las redes neuronales artificiales se **inspiran** en contraparte biológica
- Abstracción es a muy alto nivel
- Modelos de funcionamiento difieren **fuertemente**
- De interés científico: modelos bioinspirados
- De interés práctico: modelos neuronales "artificiales"
- **Concepto básico**: unidades computacionales simples (neuronas) combinan entradas y calculan un valor de salida, que será entrada a otras neuronas.

Perceptron

- 1957 Frank Rosenblatt, laboratorio aeronáutico de Cornell

Perceptron

- Primera propuesta de "red neuronal" artificial
- En regresión logística usamos $g(z) = \sigma(z)$ sigmoidal
- En el perceptron se usa $g(z) = u(z)$ (escalon unitario)
- En ambas $h_{\underline{\theta}}(\underline{\mathbf{x}}) = g(\underline{\boldsymbol{\theta}}^T \underline{\mathbf{x}})$: hipótesis mapea a [0,1]
- La regla de actualización estocástica es similar al caso anterior:

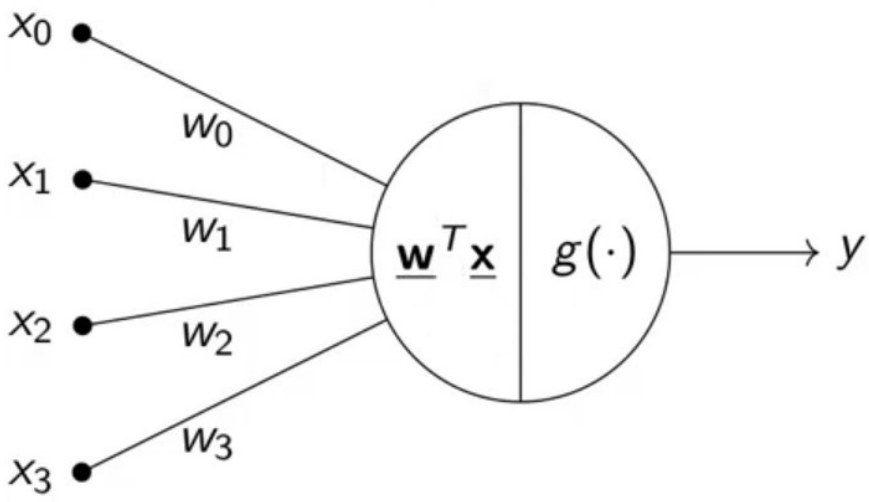
$$\theta_j \leftarrow \theta_j + \alpha(y^{(i)} - h_{\underline{\theta}}(\underline{\mathbf{x}}^{(i)}))x_j^{(i)}$$

  que difieren **mucho** en estilo de aprendizaje por la forma de $h_{\underline{\theta}}(\underline{\mathbf{x}})$
- Perceptron no tiene justificación probabilística

Nuerona arificial

- A la neurona artificial también se le llama **unidad**, para acentuar diferencias con neurona biológica
- Tiene dos tareas: 1) combinar entradas, 2) producir la señal de activación
- La neurona será nodo en un grafo dirigido
- Cada arista de entrada a la neurona tiene asociado un peso
- Generalmente
  - valor de entrada y peso de arista se mezclan para producir "*estímulo*"
  - todos los "*estímulos*" de entrada a la neurona se combinan
  - función de activación produce valor de salida de la neurona a partir de combinación de estímulos
- Existen alternativas para las tres tareas anteriores.
- Veremos el caso más común.

Modelo nuerona artificial



$$y = g(\underline{\mathbf{w}}^T\underline{\mathbf{x}})$$

- $\underline{\mathbf{x}} = [x_0\ x_1\ x_2\ x_3]^T$: entradas
- $\underline{\mathbf{w}} = [w_0\ w_1\ w_2\ w_3]^T$: pesos de entrada
- $\underline{\mathbf{w}}^T\underline{\mathbf{x}}$: combinación lineal de entradas
- $g(\cdot)$: función de activación (no lineal)
- y: salida

Modelo nuerona artificial

- Caso de regresión logística coincide con una neurona artificial clásica
- Los parámetros de la neurona son $\underline{\mathbf{w}} = \underline{\boldsymbol{\theta}}$
- La función de activación es el sigmoide $g(x) = 1/(1 + e^{-x})$

Red Neuronal Artificial

Red Neuronal Artificial

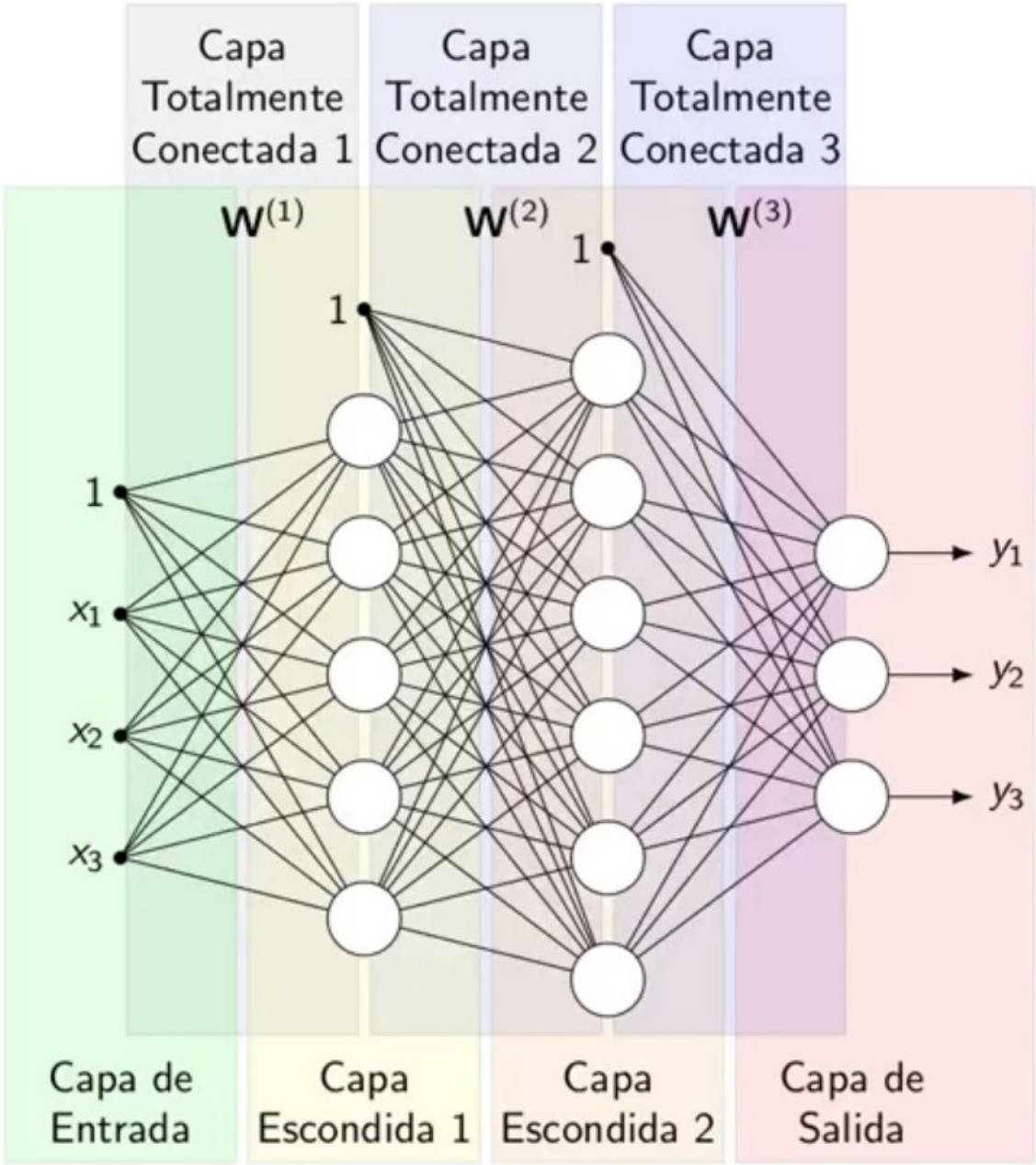
- Una red neuronal interconecta salidas de unas neuronas hacia entradas de otras (grafo dirigido)
- Usualmente las neuronas se organizan en **capas**
- La red transforma un vector $\underline{\mathbf{x}}$ de entrada en un vector $\underline{\mathbf{y}}$ de salida

$$\underline{\mathbf{y}} = \underline{\mathbf{h}}_{\Theta}(\underline{\mathbf{x}})$$

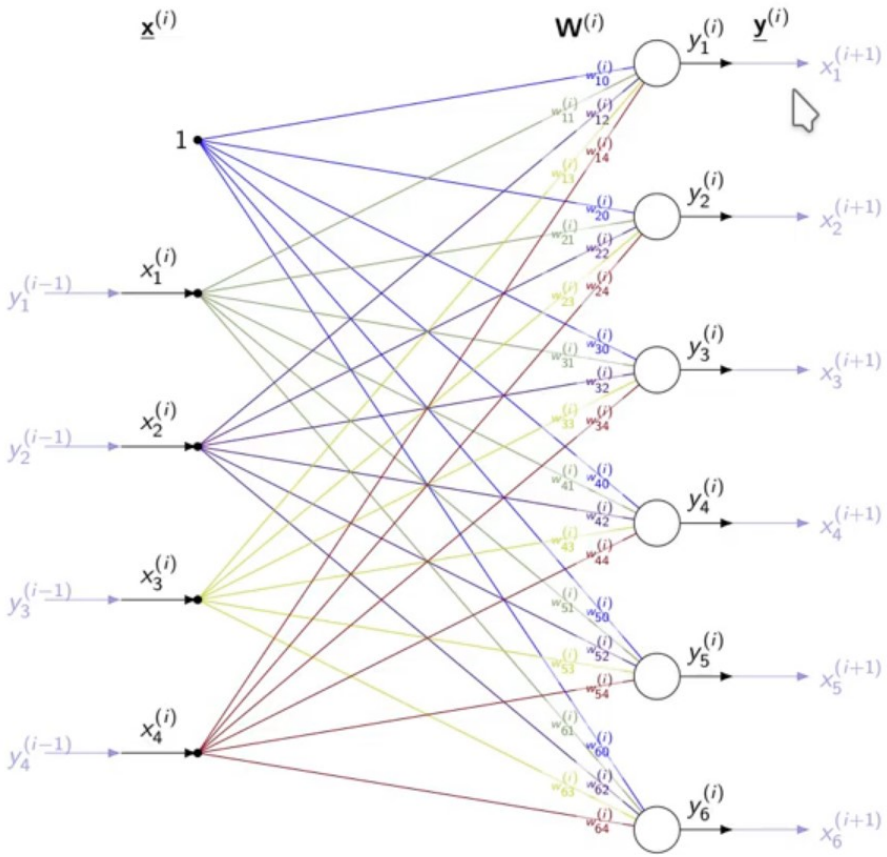
- Generalizamos la función de activación para $\underline{\mathbf{x}} \in \mathbb{R}^n$ como

$$\underline{\mathbf{g}}(\underline{\mathbf{x}}) = \begin{bmatrix} g(x_1) \\ g(x_2) \\ \dots \\ g(x_n) \end{bmatrix} \quad \text{ó a veces} \quad \underline{\mathbf{g}}(\underline{\mathbf{x}}) = \begin{bmatrix} g_1(\underline{\mathbf{x}}) \\ g_2(\underline{\mathbf{x}}) \\ \dots \\ g_n(\underline{\mathbf{x}}) \end{bmatrix}$$

Red Neuronal Artificial



Red Neuronal Artificial



$$\underline{\mathbf{y}}^{(i)} = \underline{\mathbf{g}}(\mathbf{W}^{(i)}\underline{\mathbf{x}}^{(i)})$$

$$\mathbf{W}^{(i)} = \begin{bmatrix} w_{10}^{(i)} & w_{11}^{(i)} & w_{12}^{(i)} & w_{13}^{(i)} & w_{14}^{(i)} \\ w_{20}^{(i)} & w_{21}^{(i)} & w_{22}^{(i)} & w_{23}^{(i)} & w_{24}^{(i)} \\ w_{30}^{(i)} & w_{31}^{(i)} & w_{32}^{(i)} & w_{33}^{(i)} & w_{34}^{(i)} \\ w_{40}^{(i)} & w_{41}^{(i)} & w_{42}^{(i)} & w_{43}^{(i)} & w_{44}^{(i)} \\ w_{50}^{(i)} & w_{51}^{(i)} & w_{52}^{(i)} & w_{53}^{(i)} & w_{54}^{(i)} \\ w_{60}^{(i)} & w_{61}^{(i)} & w_{62}^{(i)} & w_{63}^{(i)} & w_{64}^{(i)} \end{bmatrix}$$

$\underline{\mathbf{w}}_{:,0}^{(i)}$ es vector de **sesgo**/*bias*

Red Neuronal Artificial

- Información en capas totalmente conectadas fluye desde entradas hacia salidas
- Hay otros tipos de capas (convolucionales, de submuestreo, etc.) con también una dirección de flujo de datos.
- Esto es así en toda **red alimentada hacia adelante** (*feed forward network*)
- Si existe realimentación (salidas pasan a entradas de capa actual o anteriores) se habla de **redes recurrentes**
- Con "*pocas*" capas se habla de redes neuronales artificiales "**clásicas**". Con "*muchas*" se habla de redes "**profundas**".
- Ahora nos concentraremos en redes alimentadas hacia adelante

- Salida de una red de $c = 3$ capas con $\Theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}\}$

$$\underline{\mathbf{y}} = \underline{\mathbf{g}}\left(\mathbf{W}^{(3)}\left[\underline{\mathbf{g}}\left(\mathbf{W}^{(2)}\left[\underline{\mathbf{g}}\left(\mathbf{W}^{(1)}\begin{bmatrix}1\\ \underline{\mathbf{x}}\end{bmatrix}\right)\right]\right)\right]\right) = \underline{\mathbf{h}}_{\Theta}(\underline{\mathbf{x}})$$

- La función de activación **debe ser no-lineal**:
  - Si fuese lineal (p. ej. $\underline{\mathbf{g}}(\underline{\mathbf{x}}) = a\underline{\mathbf{x}}$), entonces

$$\underline{\mathbf{y}} = a\mathbf{W}^{(3)}\left[a\mathbf{W}^{(2)}\left[a\mathbf{W}^{(1)}\begin{bmatrix}1\\ \underline{\mathbf{x}}\end{bmatrix}\right]\right] \equiv \widetilde{\mathbf{W}}\begin{bmatrix}1\\ \underline{\mathbf{x}}\end{bmatrix}$$

    es decir, colapsa a **una sola** capa totalmente conectada.
  - Esto es equivalente a regresión lineal (no importa cuántas capas)

- **Predicción** produce $\underline{\mathbf{y}} = \underline{\mathbf{h}}_{\mathbf{\Theta}}(\underline{\mathbf{x}})$
- **Entrenamiento** debe encontrar $\mathbf{\Theta} = \left\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(c)}\right\}$
- Hay varias posibilidades de función de error.
- Por ejemplo, pueden minimizarse los mínimos cuadrados lineales (LLS, *linear least squares*)

$$J(\mathbf{\Theta}) = \frac{1}{2} \sum_{i=1}^{m} \|\underline{\mathbf{y}}^{(i)} - \underline{\mathbf{h}}_{\mathbf{\Theta}}(\underline{\mathbf{x}}^{(i)})\|_2^2$$

- Para encontrar $\mathbf{\Theta}$ debe minimizarse $J(\mathbf{\Theta})$.
- A $J(\mathbf{\Theta})$ se le denomina en este contexto **pérdida** (o *loss*).
- El descenso de gradiente recibe en este contexto el nombre de: **algoritmo de retropropagación** (*backpropagation*):

$$\mathbf{\Theta}' \leftarrow \mathbf{\Theta} - \lambda \nabla_{\mathbf{\Theta}} J(\mathbf{\Theta})$$

Red Neuronal Artificial

- Mientras que $J(\underline{\theta})$ en las regresiones lineal y logística era convexa cuadrática, $J(\mathbf{\Theta})$ en las redes neuronales por lo general no lo es.
- Esto implica que ninguno de los algoritmos utilizados para encontrar el mínimo puede asegurar convergencia a mínimo global
- El problema de optimización se torna complejo y requiere intervención manual
- Ver LeNet
- Ver Tensorflow/Playground