

Perceptrón multicapa

Retroalimentación

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

24 de febrero de 2025

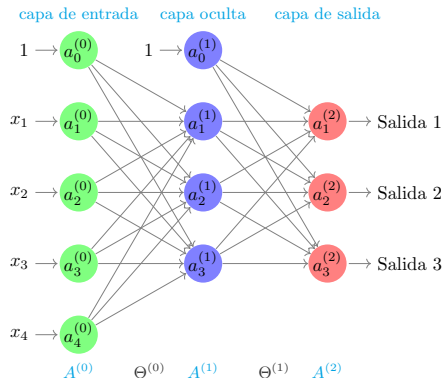


Entrenamiento

- 1 Entrenamiento
- 2 Función de error
- 3 Cálculo del gradiente

Problema de aprendizaje

- Dada una arquitectura de red neuronal, encontrar los pesos tales que la función definida por la red neuronal $h_{\Theta}(\vec{x})$ aproxime, hasta cierta tolerancia, la función de interés $f(\vec{x})$.



Entrenamiento

La estrategia de entrenamiento para encontrar un conjunto de pesos que satisfaga este requerimiento consiste en lo siguiente:

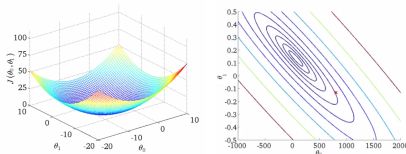
- Definir una función de error o pérdida J que mida la distancia entre los valores deseados y los valores obtenidos con un conjunto de pesos dados Θ .
- Utilizar alguna técnica de optimización de funciones que minimize este error.
- Tanto la arquitectura a considerar, como la función de error y la técnica de optimización dependerán del tipo de función que se desee aproximar.

Retropropagación

La primer técnica altamente efectiva para entrenar redes neuronales se le conoce como *Retropropagación*, *Propagación hacia atrás* o en inglés *Retropropagation* consistió en la combinación siguiente:

- Utilizar la regla de la cadena para obtener el gradiente de la función de error con respecto al conjunto de **pesos**, dado un conjunto de datos de entrenamiento \mathbf{X} con sus etiquetas \mathbf{Y} .
- Utilizar descenso por el gradiente para encontrar un mínimo, lo suficientemente bueno, de la función de error.

$$\Theta' = \Theta - \alpha \nabla_{\Theta} J \quad (1)$$



- Actualmente se puede cambiar la técnica de optimización por algún otro método más avanzado, también depende del gradiente.
- La derivación original utilizaba *diferencias al cuadrado* como función de error. Sin embargo se ha comprobado que esa función es adecuada para problemas de regresión, pero para problemas de clasificación es mejor utilizar entropía cruzada.
- Se puede consultar la derivación del gradiente para diferencias al cuadrado en Russell y Norving 2010

Función de error

- 1 Entrenamiento
- 2 Función de error
- 3 Cálculo del gradiente

Entropía cruzada

- Para problemas de clasificación se recomienda utilizar a la entropía cruzada como función de error. NG 2017
- Esta mide la cantidad de bits necesarios para identificar una clase dada la hipótesis de la red $h_{\Theta}(\vec{x})$, siendo que éstas provienen de la función $y(\vec{x})$. ^[1]

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^{s_L} y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - h_{\Theta}(x^{(i)}))_k \right] \quad (2)$$

donde:

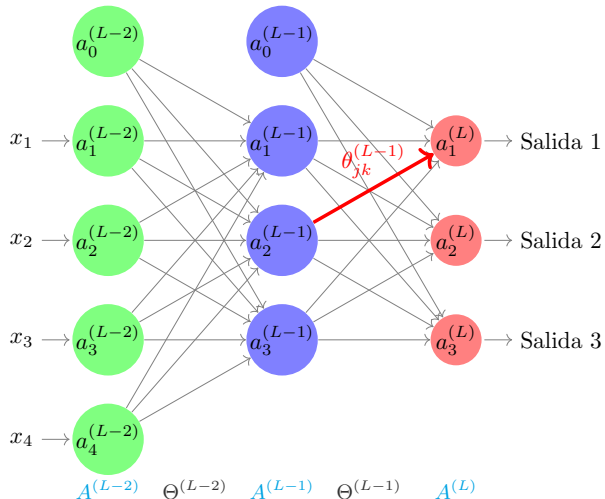
- m es el número de ejemplares de entrenamiento.
- s_L es el número de neuronas en la capa L .
- y_k es el valor para la k -ésima neurona de salida y toma valores en $\{0,1\}$.

^[1][WikiEntropia](#)

Cálculo del gradiente

- 1 Entrenamiento
- 2 Función de error
- 3 Cálculo del gradiente

Red neuronal (entrenamiento)



Derivada de la función sigmoide

$$\sigma_{\Theta}(X) = \frac{1}{1 + e^{-X\Theta}}$$

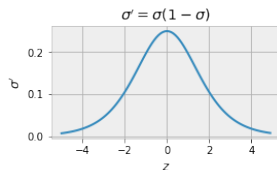
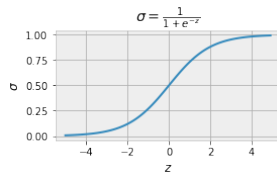
$$\frac{\partial \sigma}{\partial \theta_i} = -\frac{1}{(1 + e^{-X\Theta})^2} e^{-X\Theta} (-x_i) = \frac{1}{1 + e^{-X\Theta}} \frac{e^{-X\Theta}}{1 + e^{X\Theta}} x_i$$

$$= \sigma_{\Theta}(X) \frac{e^{-X\Theta}}{1 + e^{-X\Theta}} x_i$$

$$= \sigma_{\Theta}(X) \frac{-1 + 1 + e^{-X\Theta}}{1 + e^{-X\Theta}} x_i$$

$$= \sigma_{\Theta}(X) \left(\frac{1 + e^{-X\Theta}}{1 + e^{-X\Theta}} - \frac{1}{1 + e^{-X\Theta}} \right) x_i$$

$$\frac{\partial \sigma}{\partial \theta_i} = \sigma_{\Theta}(X) (1 - \sigma_{\Theta}(X)) x_i$$



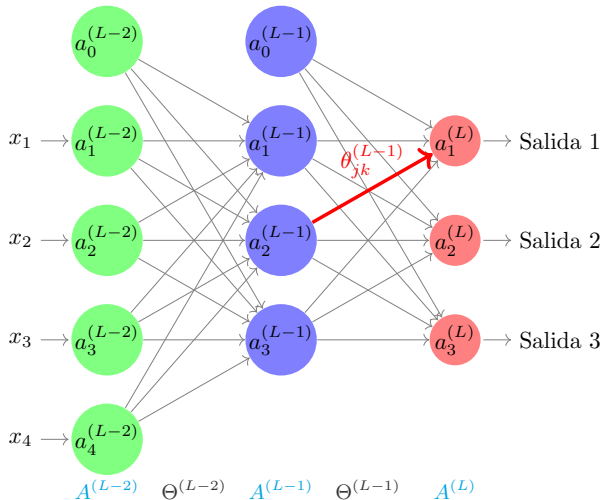
$$\sigma = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial \sigma}{\partial z} = \sigma(1 - \sigma)$$

Temas

- 3 Cálculo del gradiente
 - Derivada para un solo ejemplar

Red neuronal (entrenamiento última capa)



Entrenamiento para un ejemplar

$$J(\Theta) = - \left[\sum_{k=1}^{s_L} y_k^{(i)} \log(\mathbf{a}_k^{(L)}) + (1 - y_k^{(i)}) \log(1 - \mathbf{a}_k^{(L)}) \right] \quad (3)$$

Capa de salida:

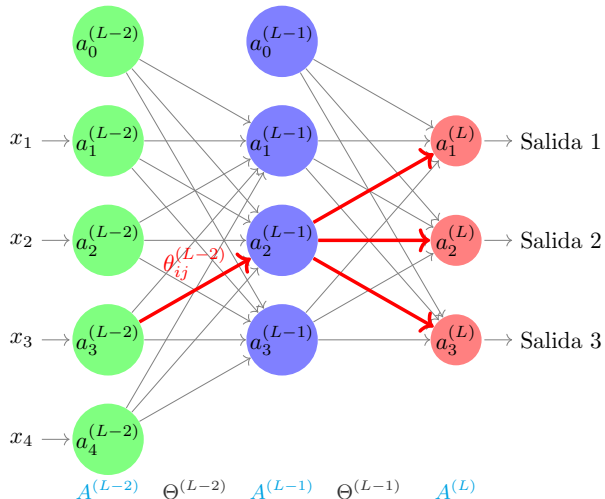
$$\frac{\partial J}{\partial \theta_{jk}^{(L-1)}} = - \left[\frac{y_k}{\mathbf{a}_k^{(L)}} \frac{\partial}{\partial \theta_{jk}^{(L-1)}} g \left(\overbrace{\sum_{j'=0}^{s_L-1} \theta_{j'k} \mathbf{a}_{j'}^{(L-1)}}^{z_k} \right) - \left(\frac{1 - y_k}{1 - \mathbf{a}_k^{(L)}} \right) \frac{\partial}{\partial \theta_{jk}^{(L-1)}} g(z_k) \right] \quad (4)$$

De la derivada de la suma sólo queda $\mathbf{a}_{j'}$ con $j' = j$.

$$= - \left[\frac{y_k}{\mathbf{a}_k^{(L)}} - \left(\frac{1 - y_k}{1 - \mathbf{a}_k^{(L)}} \right) \right] \overbrace{g'(z_k)}^{g'=g(1-g)} \mathbf{a}_j^{(L-1)} = - \left[\frac{y_k(1 - \mathbf{a}_k^{(L)}) - \mathbf{a}_k^{(L)}(1 - y_k)}{\mathbf{a}_k^{(L)}(1 - \mathbf{a}_k^{(L)})} \right] \overbrace{\mathbf{a}_k^{(L)}(1 - \mathbf{a}_k^{(L)})}^{g'(z_k)} \mathbf{a}_j^{(L-1)} \quad (5)$$

$$= - \mathbf{a}_j^{(L-1)} \underbrace{(y_k - \mathbf{a}_k^{(L)})}_{\delta_k} \quad (6)$$

Red neuronal (entrenamiento penúltima capa)



Entrenamiento 2

$$J(\Theta) = - \left[\sum_{k=1}^{s_L} y_k^{(i)} \log(\mathbf{a}_k^{(L)}) + (1 - y_k^{(i)}) \log(1 - \mathbf{a}_k^{(L)}) \right] \quad (7)$$

Capa de anterior:

$$\frac{\partial J}{\partial \theta_{ij}^{L-2}} = - \sum_{k=1}^{s_L} \left[\frac{y_k}{\mathbf{a}_k^{(L)}} \frac{\partial}{\partial \theta_{jk}^{(L-2)}} g \left(\overbrace{\sum_{j'=0}^{s_L-1} \theta_{j'k} \mathbf{a}_{j'}^{(L-1)}}^{z_k} \right) - \left(\frac{1 - y_k}{1 - \mathbf{a}_k^{(L)}} \right) \frac{\partial}{\partial \theta_{jk}^{(L-2)}} g(z_k) \right] \quad (8)$$

$$= - \sum_{k=1}^{s_L} \underbrace{(y_k - \mathbf{a}_k^{(L)})}_{\delta_k} \theta_{jk} \frac{\delta}{\delta \theta_{ij}^{(L-2)}} \mathbf{a}_j^{(L-1)} \quad (9)$$

$$= - \sum_{k=1}^{s_L} \underbrace{(y_k - \mathbf{a}_k^{(L)})}_{\delta_k} \theta_{jk} \frac{\delta}{\delta \theta_{ij}^{(L-2)}} g \left(\overbrace{\sum_{i'=0}^{s_L-2} \theta_{i'j}^{(L-2)} \mathbf{a}_{i'}^{(L-2)}}^{z_j} \right) \quad (10)$$

Entrenamiento 3

$$= - \underbrace{\sum_{k=1}^{s_L} \delta_k \theta_{jk}}_{\delta_j} \overbrace{g'(z_j)}^{g(a_j^{(L-1)})(1-g(a_j^{(L-1)}))} a_i^{(L-2)} \quad (11)$$

Resumiendo:

$$\delta_k^{(L)} = (y_k - a_k^{(L)}) \quad \frac{\partial J}{\partial \theta_{jk}^{(L-1)}} = -a_j^{(L-1)} \delta_k^{(L)} \quad (12)$$

$$\delta_j^{(L-1)} = \left(\sum_{k=1}^{s_L} \delta_k^{(L)} \theta_{jk} \right) g'(z_j) \quad \frac{\partial J}{\partial \theta_{ij}^{(L-2)}} = -a_i^{(L-2)} \delta_j^{(L-1)} \quad (13)$$

Vectorización

Consideremos todos los datos involucrados durante el entrenamiento:

- Hay s_L neuronas de salida, para las cuales se calculará el error.
- Se puede calcular el promedio del gradiente para m ejemplares simultáneamente.
- Si escribimos los datos en forma matricial, es posible paralelizar estos cálculos utilizando operaciones de matrices.

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^{s_L} y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - h_{\Theta}(x^{(i)}))_k \right] \quad (14)$$

Con:

$$\mathbf{X} = \begin{bmatrix} x_0^{(1)} & \cdots & x_n^{(1)} \\ & \ddots & \\ x_0^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_0^{(1)} & \cdots & y_{s_L}^{(1)} \\ & \ddots & \\ y_0^{(m)} & \cdots & y_{s_L}^{(m)} \end{bmatrix} \quad \Theta^{(l)} = \begin{bmatrix} \theta_{(01)} & \cdots & \theta_{(0s_L)} \\ & \ddots & \\ \theta_{(s_{(l-1)}1)} & \cdots & \theta_{(s_{(l-1)}s_L)} \end{bmatrix} \quad (15)$$

Vectorización 2

Podemos reescribir:

$$\delta_k^{(L)} = (y_k - a_k^{(L)})$$

$$\frac{\partial J}{\partial \theta_{jk}^{(L-1)}} = -\frac{1}{m} \sum_{(i=1)}^m a_j^{(L-1)(i)} \delta_k^{(L)(i)}$$

$$\delta_j^{(L-1)} = \left(\sum_{k=1}^{s_L} \delta_k^{(L)} \theta_{jk} \right) g'(z_j)$$

$$\frac{\partial J}{\partial \theta_{ij}^{(L-2)}} = -a_i^{(L-2)} \delta_j^{(L-1)}$$

$$\Delta^{(L)} = Y - A^{(L)}$$

$$\nabla^{(L-1)} = -\frac{1}{m} \left(A^{(L-1)} \right)^T \Delta^{(L)}$$

$$\Delta^{(L-1)} = \Delta^{(L)} \left(\Theta_{[1:,:]}^{(L-1)} \right)^T \circ g'(Z^{(L-1)})$$

$$\nabla^{(L-2)} = -\frac{1}{m} \left(A^{(L-2)} \right)^T \Delta^{(L-1)}$$

Vectorización 3

Las fórmulas vectorizadas se pueden escribir en forma general con:

Error cometido por la última capa L

$$\Delta^{(L)} = Y - A^{(L)} \quad (16)$$

Error cometido por cualquier capa $l - 1$

$$\Delta^{(l-1)} = \Delta^{(l)} \left(\Theta_{[1:, :]}^{(l-1)} \right)^T \circ g'(Z^{(l-1)}) \quad (17)$$

$$g'(Z^{(l-1)}) = A^{(l-1)} \circ (1 - A^{(l-1)}) \quad (18)$$




Componentes del gradiente con respecto a los pesos en Θ^{l-1}

$$\nabla^{(l-1)} = -\frac{1}{m} \left(A^{(l-1)} \right)^T \Delta^{(l)} \quad (19)$$

Algorithm Propagación hacia atrás.

- 1: $X \leftarrow$ ejemplares de entrenamiento.
 - 2: $Y \leftarrow$ respuestas correctas.
 - 3: $L \leftarrow$ número de capas.
 - 4: Sean las matrices $\Delta_{s_{l+1} \times s_l}^{(l)} \leftarrow 0$ con $l \in [1, L - 1]$.
 - 5: $A^{(1)} \leftarrow X$
 - 6: **for all** $l \in [2, L]$ **do**
 - 7: $A^{(l)} \leftarrow \text{sigmoide}(A^{(l-1)}\Theta^{(l-1)})$ (propagación hacia adelante).
 - 8: **for all** $l \in [L - 1, 1]$ **do**
 - 9: Calcular $\Delta^{(l)}$
 - 10: Calcular $\nabla^{(l)}$
 - 11: **for all** $l \in L - 1$ **do**
 - 12: $\Theta^{(l)} \leftarrow \Theta^{(l)} - \alpha \nabla^{(l)}$
- ▷ Descenso por el gradiente
-

Referencias I

-  NG, Andrew (2017). *Machine Learning*. Coursera. URL:
<https://www.coursera.org/learn/machine-learning>.
-  Russell, Stuart y Peter Norving (2010). *Artificial Intelligence, A Modern Approach*. Ed. por Michael Hirsch. 3a. Pearson Prentice Hall.
-  *Entropía cruzada*, Wikipedia, https://es.wikipedia.org/wiki/Entrop%C3%ADa_cruzada

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual

