COMPUTACIÓN

TEC | Tecnológico de Costa Rica

Aprendizaje automático

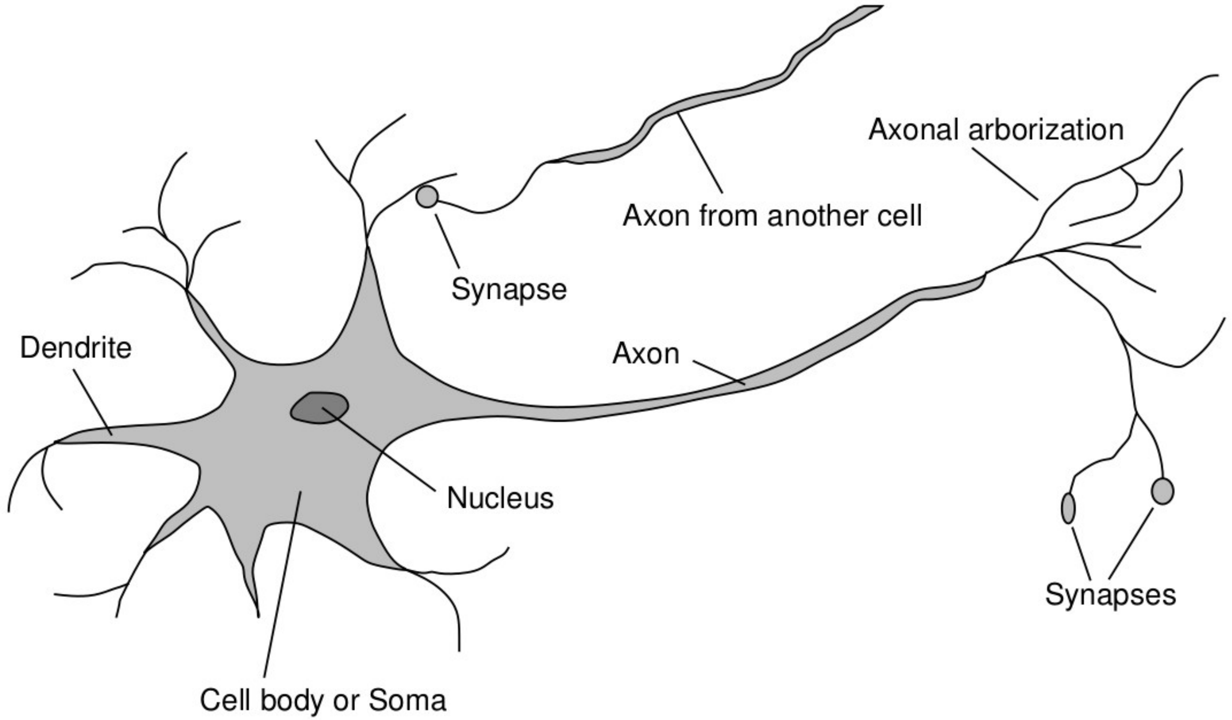
Redes neuronales

María Auxiliadora Mora, ITCR, julio 2023

Contenidos

- Introducción
- Redes neuronales
- Perceptrón
- Perceptrón multicapa (MLP)
- Aplicaciones de las redes neuronales

Cerebro - neuronas

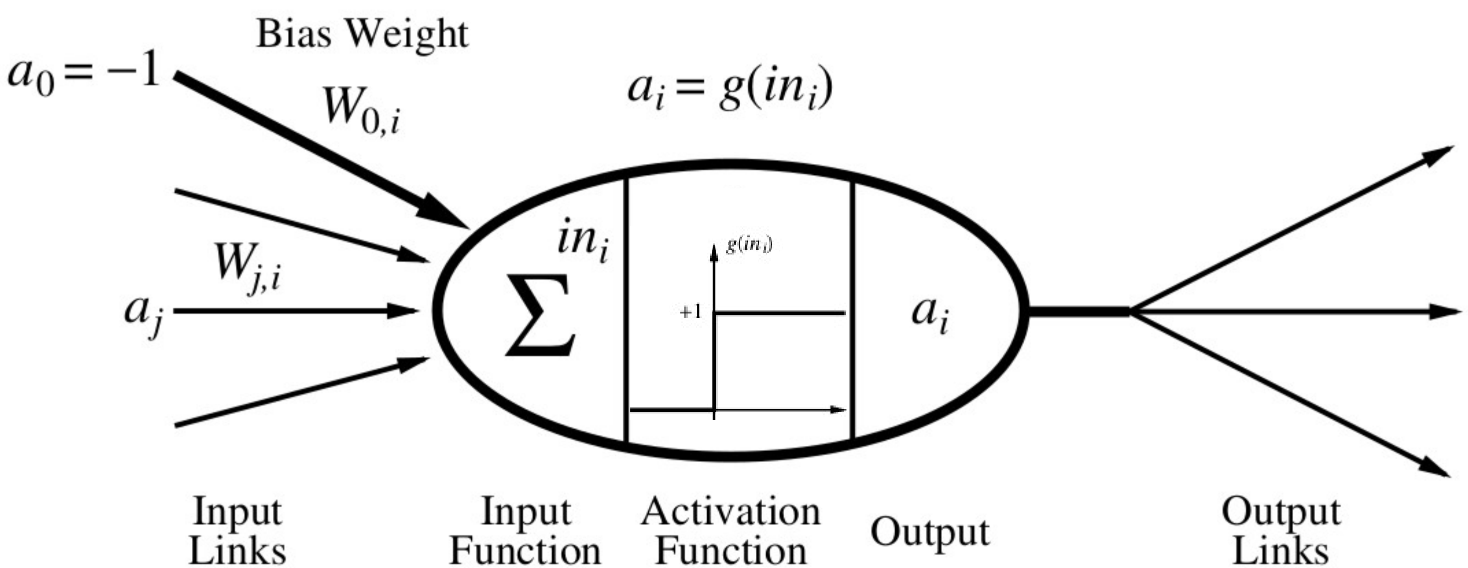


(Russell and Norvig, 2009)

La unidad McCulloch–Pitts (1943)

- Modelo matemático de una neurona (unidad) usada para aproximar la función hipótesis (h)
- La salida es una función lineal de las entradas:

$$a_i = g(in_i) = g(\Sigma_j W_{j,i} a_j)$$



$\Sigma_j W_{j,i} a_j$ = 1 si la sumatoria sobrepasa el umbral
= 0 si no

(Russell and Norvig, 2009)

Redes Neuronales

- Compuestas por nodos y arcos dirigidos.
- Un enlace desde la unidad i a la unidad j sirve para propagar la activación a_i de i a j.
- Cada enlace también tiene un peso $w_{i,j}$ asociado, lo que determina la fuerza y el signo de la conexión.
- Cada unidad tiene una entrada ficticia $a_0 = 1$ con un peso asociado $w_{0,j}$.
- Cada unidad j

1)primero calcula una suma ponderada de sus entradas.

$$in_i = \Sigma_j W_{j,i} a_j$$

2)Luego aplica la función de activación

$$g(in_i) = g \Sigma_j W_{j,i} a_j$$

Perceptrón de una capa

Las entradas son multiplicadas por los pesos y la sumatoria de esos resultados se pasa a la función de activación para generar la salida.

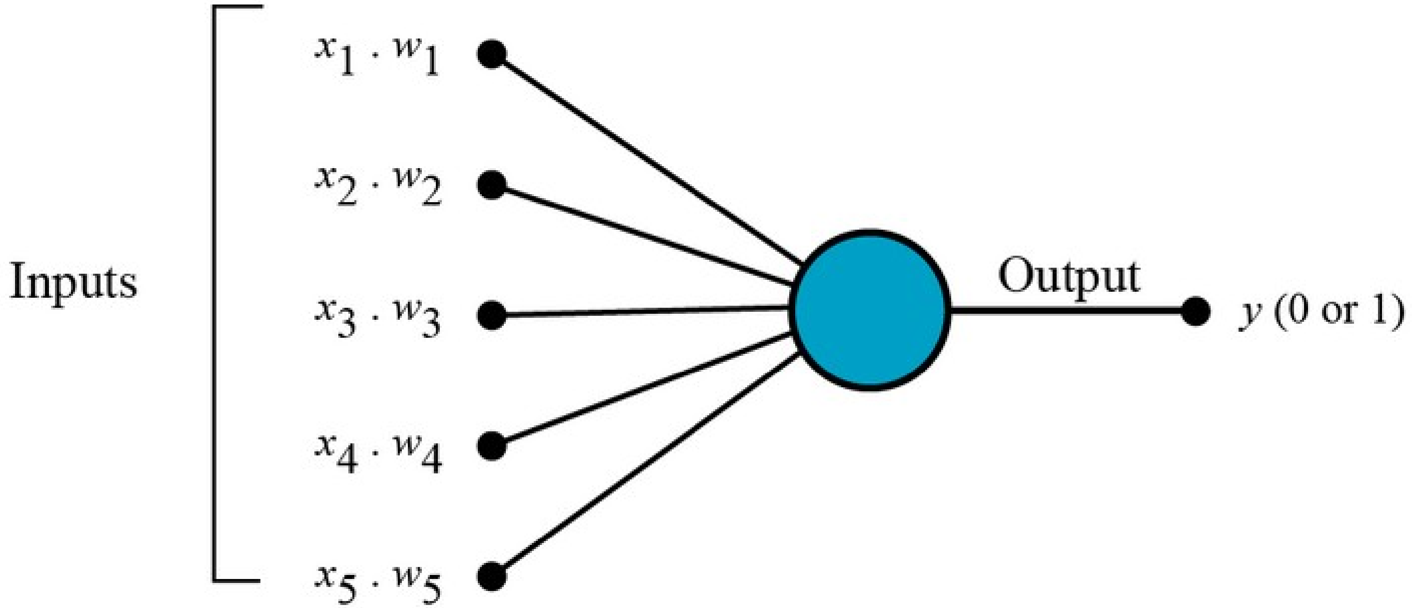
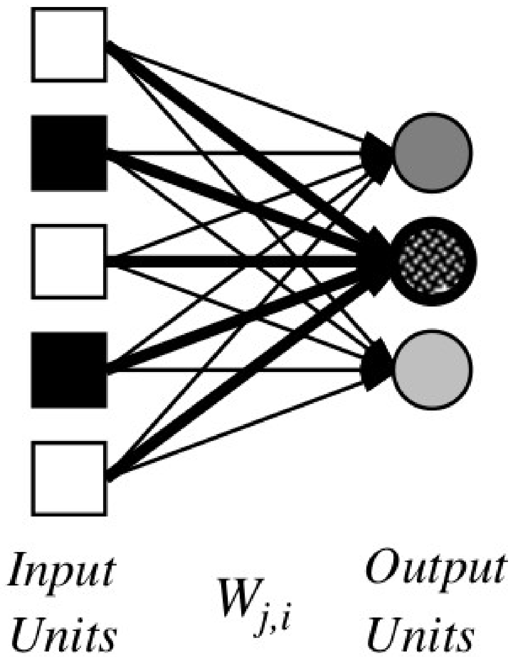
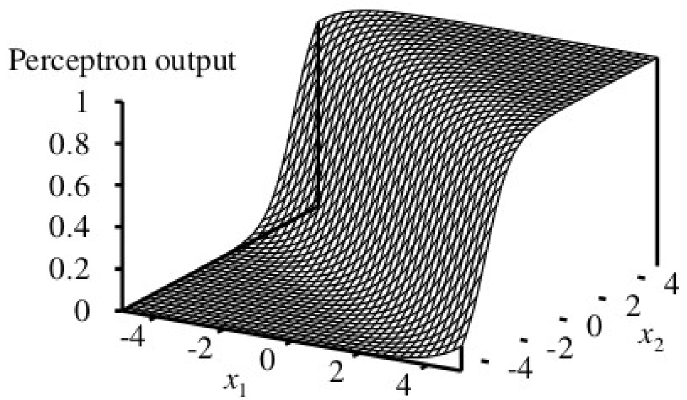


Imagen:https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

Perceptrón de una capa



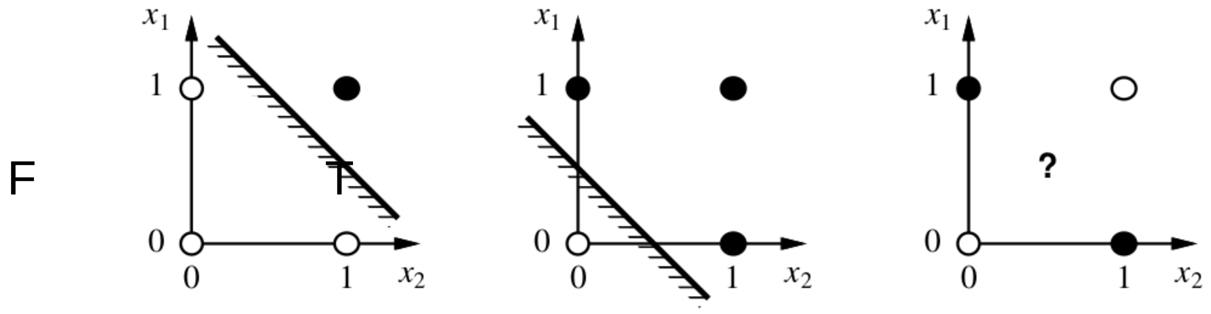


Todas las unidades de salida funcionan separadamente, los pesos no se comparten.

El ajuste de los pesos mueve la ubicación, la orientación y la inclinación de la pendiente da la curva.

Expresividad de la red de perceptrón de una capa

- Un perceptrón con función de activación umbral puede reconocer operaciones AND, NOT, mayoría, etc. pero no XOR.
- La red representa un separador lineal en el espacio de entradas.
- Si utilizamos la red de una capa vista anteriormente, la salida sería:

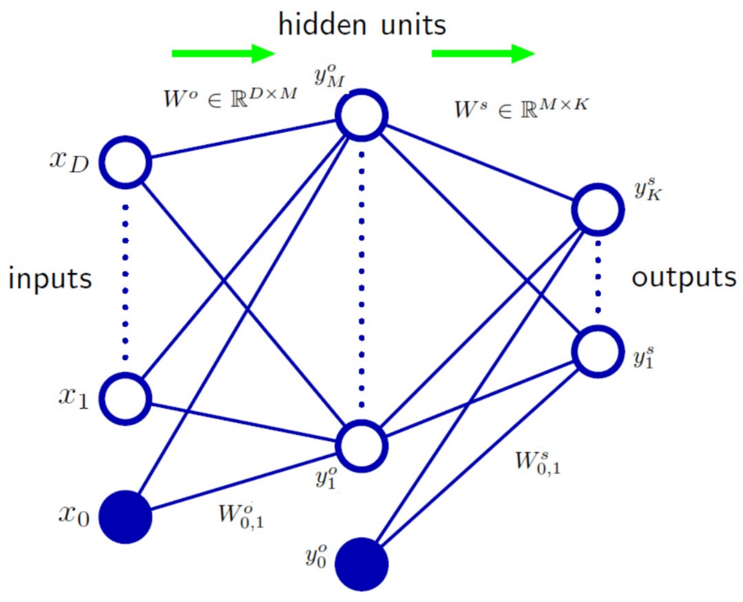


(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

La función XOR no es linealmente separable, por lo que el perceptrón no puede aprenderla. Minsky & Papert (1969) mataron el desarrollo de la redes neuronales con esta demostración.

Perceptrón multicapa (MLP)

- Cada capa conecta N unidades de entrada a M unidades de salida.

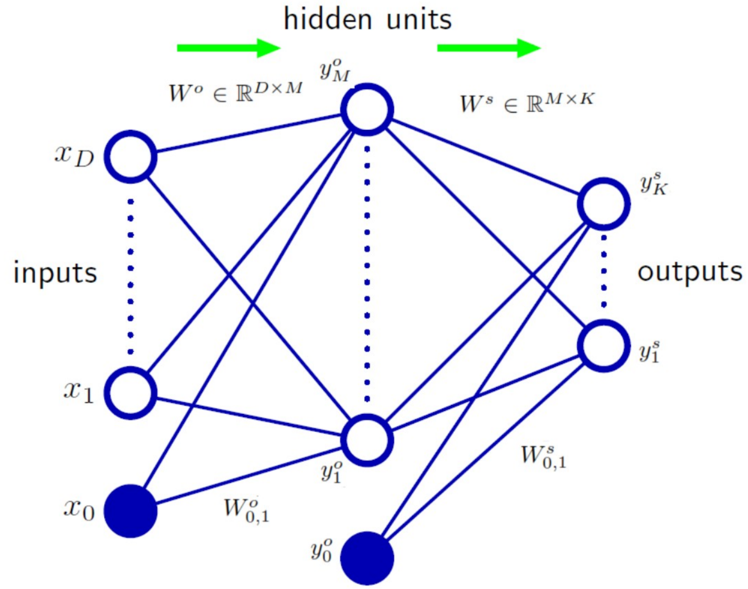


- Una red multicapa que consta de capas totalmente conectadas se denomina perceptron multicapa.
- La RN se pueden usar para problemas de regresión y clasificación.

(Bishop, 2006)

Perceptrón multicapa (MLP)

Alimentación o propagación hacia adelante:



- El peso neto o coeficiente de activación: $p^o_m\left(\vec{x}, W^o\right) = \sum_{d=1}^{D} W^o_{d,m} x_d + W^o_{0,m}$

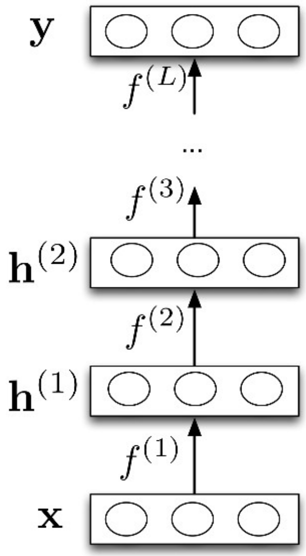
- El peso neto entra a la **función de activación**: $y^o_m\left(\vec{x}, W^o\right) = g^o\left(p^o_m\left(\vec{x}, W^o\right)\right)$

(Bishop, 2006)

Perceptrón multicapa

Una familia parametrizada de funciones.

Cada capa calcula una función, por lo que la red calcula una composición de funciones:

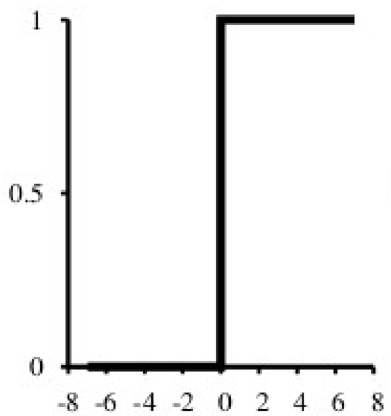


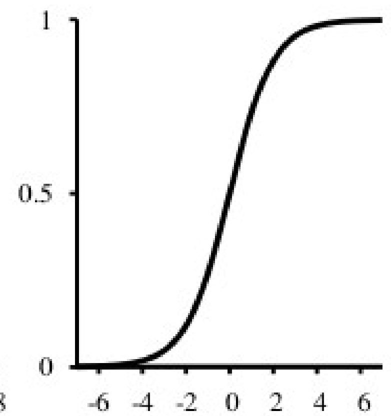
Las redes neuronales proporcionan modularidad: cada capa se implementa como una caja negra.

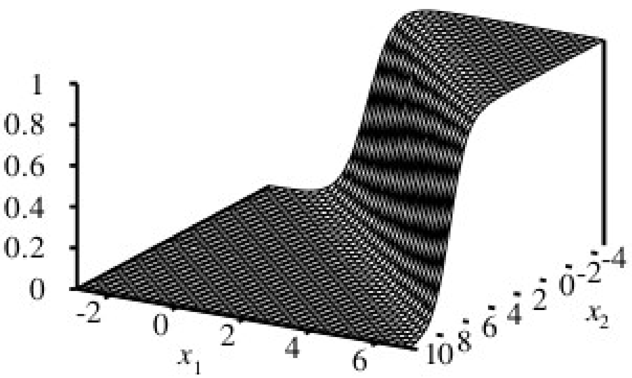
(Grosse, 2018)

¿Para qué sirve la función de activación?

Su propósito principal es **convertir una señal de entrada en una señal de salida**.







Función umbral

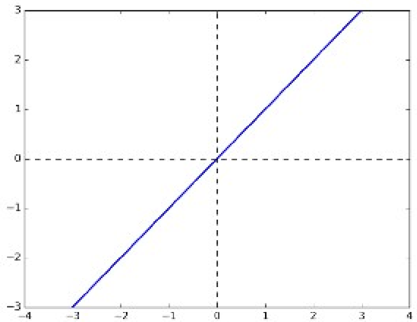
$$y = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

Función sigmoidal

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

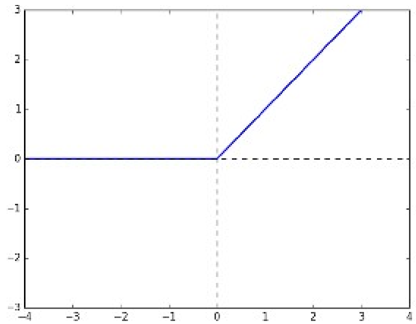
(Russell and Norvig, 2009)

Algunas funciones de activación



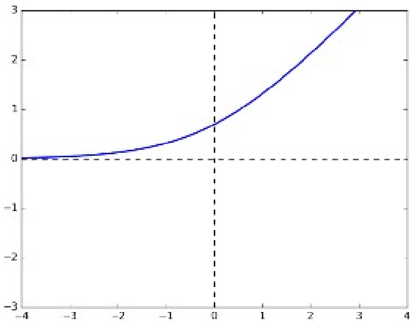
Linear

$$y = z$$



Rectified Linear Unit (ReLU)

$$y = \max(0, z)$$

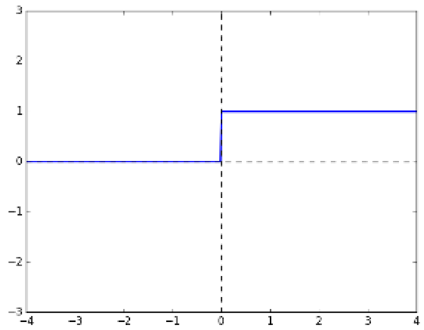


Soft ReLU

$$y = \log 1 + e^{z}$$

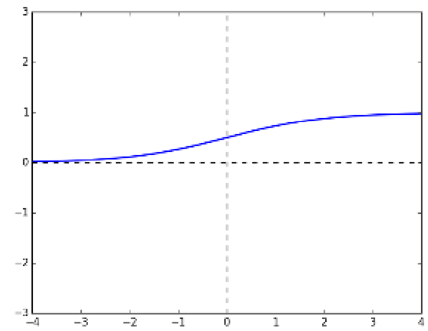
(Grosse, 2018)

Algunas funciones de activación



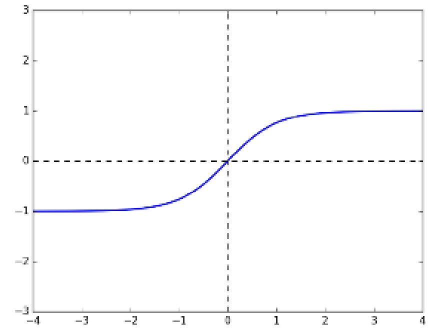
Hard Threshold

$$y = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$



Logistic

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

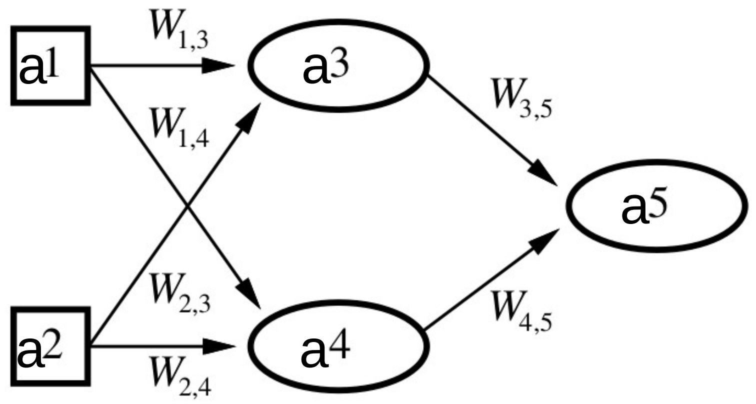


Hyperbolic Tangent (tanh)

$$y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

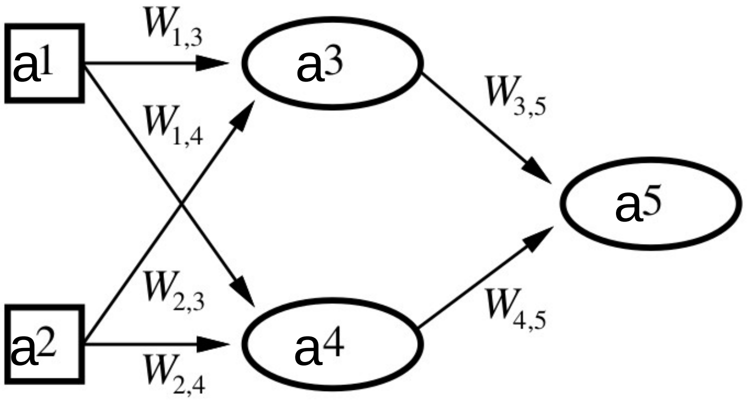
(Grosse, 2018)

Ejemplo de alimentación hacia adelante



Red con alimentación hacia adelante:

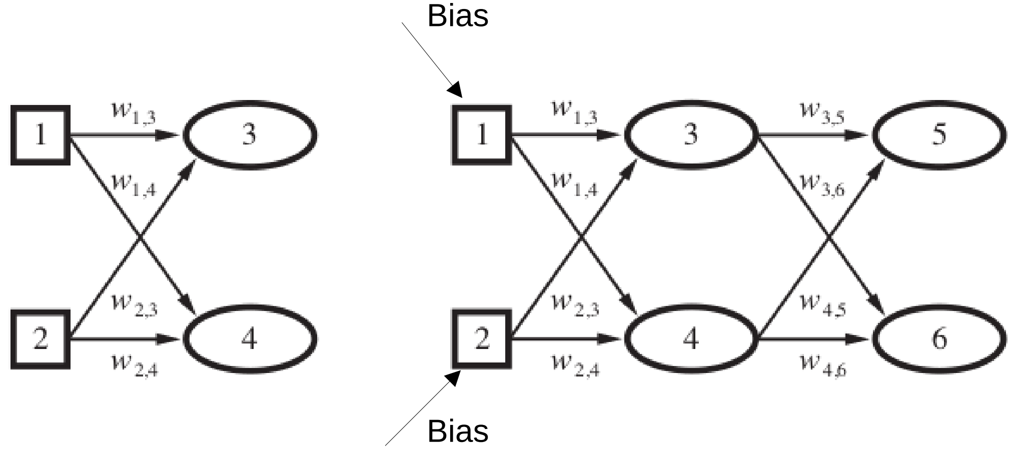
Ejemplo de alimentación hacia adelante



Red con alimentación hacia adelante:

$$\begin{aligned} a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\ &= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \end{aligned}$$

Redes con dos entradas y dos salidas



Una capa

Dos capas

Expresividad de la red de perceptrón

- Las redes de perceptrón multicapas con funciones de activación no lineales son aproximadores universales
- Esto se ha demostrado para varias funciones de activación (Logística, ReLU, etc.)
    - A pesar de que ReLU es "casi" lineal funciona muy bien.
- Prefiera redes compactas.
- Evite el sobre **sobreajuste**:

Sobreajuste es el resultado de un análisis que corresponde muy estrechamente con un conjunto particular de datos y, por lo tanto, puede fallar al predecir futuras observaciones de manera confiable.

La línea verde representa un h sobre ajustado y la línea negra representa un h regularizado.

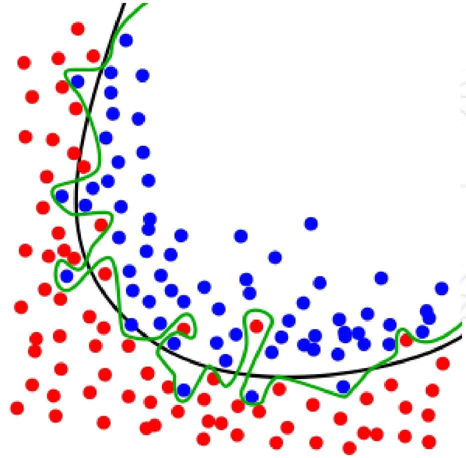
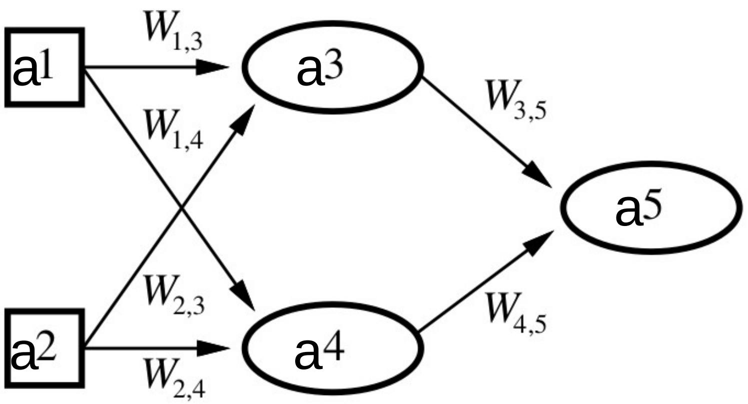


Imagen: wikipedia.org

¿Cómo aprende el perceptrón?

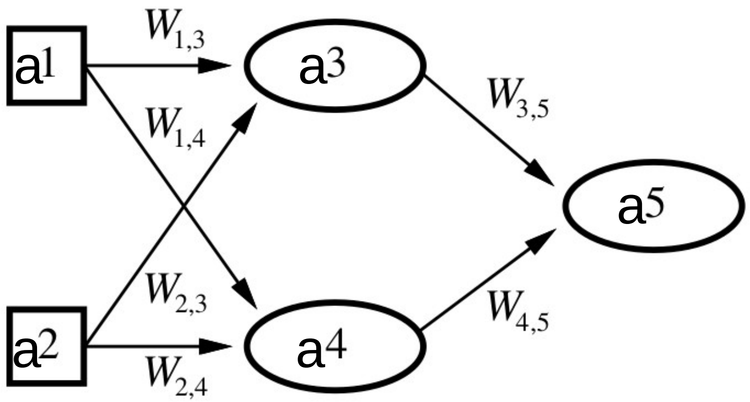
Sabemos que en una red neuronal (NN) el resultado está en función de las entradas, los pesos y la función de activación:



$$
\begin{aligned}
a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\
&= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))
\end{aligned}
$$

¿Cómo aprende el perceptrón?

Sabemos que en una red neuronal (NN) el resultado está en función de las entradas, los pesos y la función de activación:



Aprendizaje consiste en ajustar los pesos para lograr mejores resultados, es decir, reforzar las conexiones que llevan a una salida correcta.

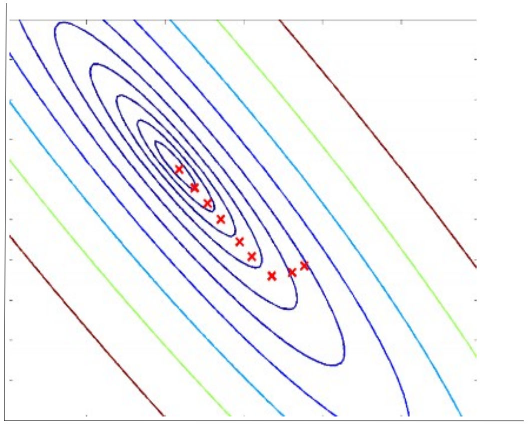
¿Cómo aprende el perceptrón?

Para mejorar el desempeño de la red (aprender), se requiere minimizar la distancia entre la salida z y la salida deseada (error).

$| y – z| \Rightarrow | y – h_w(x)|$ con y = salida conocida.

En términos de una red de dos neuronas se buscan los w1 y w2 que mejoren la medida de desempeño.

w_2



w_1

- El descenso de gradiente se mueve en dirección opuesta de la gradiente (la dirección de la pendiente más pronunciada).
- La función de activación no puede ser umbral.

¿Cómo aprende el perceptrón?

- **El objetivo es encontrar el vector de pesos w que minimice la función de error cuadrático medio**

$$E(\vec{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\|y(\vec{x}_n, \vec{w}) - \vec{t}_n\right\|^2$$

- El mínimo en una función multivariable ocurre cuando el gradiente tiende a cero.

$$\nabla E\left(\vec{w}_{\text{opt}}\right) = 0$$

- La optimización se realiza por medio de un proceso iterativo.

- Entonces la actualización de los pesos en la iteración t se da de la siguiente forma:

$$\vec{w}(\tau+1) = \vec{w}(\tau) - \alpha\,\nabla E(\vec{w}(\tau))$$

Con alfa la tasa de aprendizaje.

Retropropagación

- El error en la capa de salida es claro pero los resultados de las capas ocultas no son visibles.
- Se propaga el error de salida hacia las capas ocultas para ajustar los pesos en estas también. A este proceso se le denomina retro-propagación.
- El proceso de entrenamiento de retropropagación con descenso de gradiente se puede dividir en las siguientes etapas:
  - Propagación del error para calcular las derivadas parciales desde la capa de salida hacia la entrada (hacia atrás).
  - Utilizar el resultado del gradiente evaluado desde la entrada para computar los ajustes a realizar en los pesos, lo que corresponde a la aplicación de la técnica de descenso del gradiente.

Referencias

- Russell, SJ.; Norvig P (2009). Artificial Intelligence : A Modern Approach. Prentice Hall Series.
- Bishop, C (2006). Pattern Recognition and Machine Learning. (S. Calderón, Trans.). Springer.
- Grosse Roger (2018). Multilayer perceptrons. Universidad de Toronto. Recuperado de https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/readings/L05%20Multilayer%20Perceptrons.pdf
- Winston, P. (1992). Artificial Intelligence (3era Edición ed.). Massachusetts: Addison-Wesley.
- Krizhevsky, A., Ilya, S., and Hilton, G. (2012). Imagenet Classification with Deep Convolutional Neural Networks. Advanced in Neural Information Processing Systems. Recuperado de https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf