El perceptrón simple

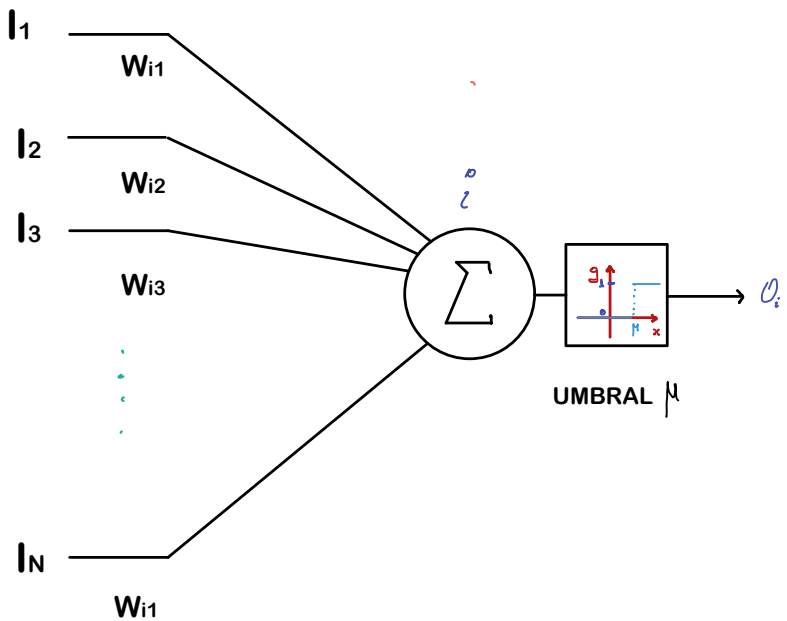
Francisco Tamarit

FAMAF, Universidad Nacional de Córdoba
IFEG, UNC y CONICET

francisco.tamarit@unc.edu.ar

15 de junio de 2023

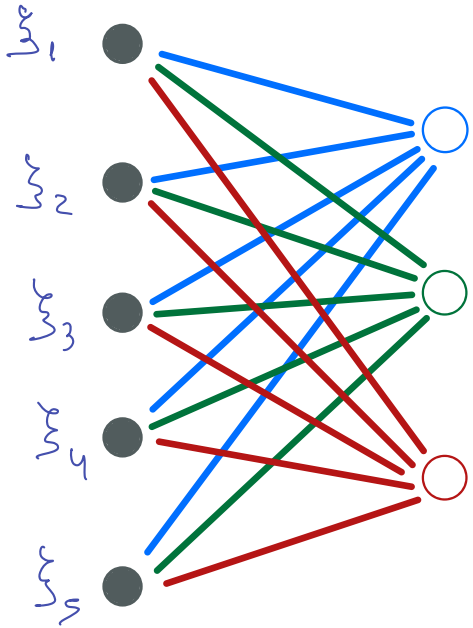
LA NEURONA DE MCCULLOCH Y PITTS (1943)



$O_i = +1$

$O_i = -1$

Consideremos la red neuronal más simple de todas las posibles, que consiste de una capa de N entradas (a las cuales no consideramos neuronas pues solo *presentan* los datos) y M neuronas (ahora sí neuronas) de salida.



Llamamos ξ_k a cada una de las N entradas ($k = 1, 2, \ldots, N$). En principio cada unidad de entrada es un número real y por lo tanto podemos pensar la entrada como un vector de N componentes:

$$\vec{\xi} = (\xi_1, \xi_2, \ldots, \xi_N) \in R^N$$

Llamamos O_i a los estados de cada una de las M neuronas de la

capa de salida y de la misma forma las pensamos como un vector de M componentes:

$$\vec{O} = (O_1, O_2, \ldots, O_M) \in R^M$$

Cada neurona de salida es una neurona de McCulloch y Pitts a la cual le cambiamos solamente la función de activación:

$$O_i = g(h_i - \mu_i)$$

La cantidad h_i modela el estímulo que las N entradas producen sobre la neuronal O_i de la capa de salida, y no es más que una suma lineal de las entradas pero ponderadas por los valores de las sinapsis:

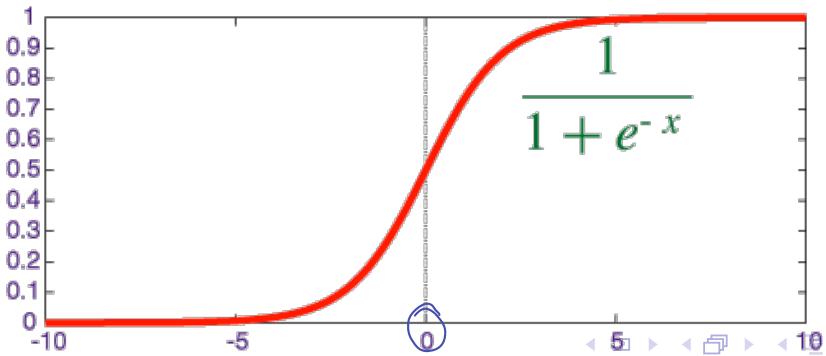
$$h_i = \sum_{i=1}^{N} W_{ik}\, \xi_i = \vec{W}_i \cdot \vec{\xi}$$

La cantidad μ_i modela el llamado *umbral de activación* de la neurona i-ésima. La neurona compara el valor de h_i con su umbral de activación μ_i y produce una señal (que enviará por su axón y sus terminales a otras neuronas) conforme a esta comparación $h_i - \mu_i$.

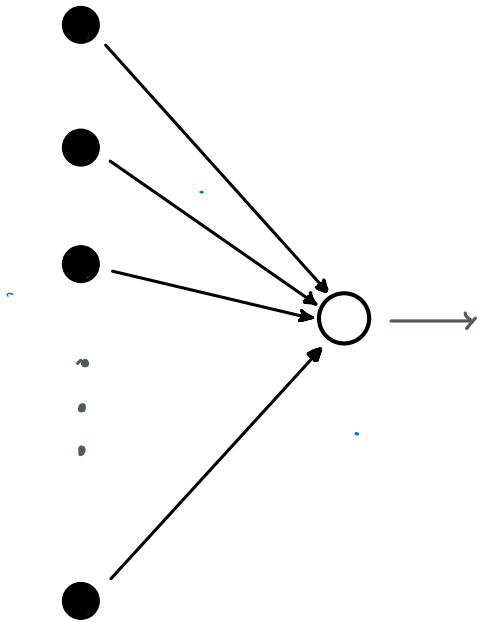
A diferencia de la neuronas originales del trabajo de McCulloch y Pitts, que usaban una función *escalón* (o Heaviside) como función de activación, aquí supondremos que se trata de una función sigmodea, o sea, que se acerca asimptóticamente hacia $+1$ (por debajo) cuando la entrada tiende a $+\infty$ y tiende asimptóticamente a 0 (por arriba) cuando la entrada tiende a $-\infty$. Para esto usaremos la función:

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

Esta función nos garantiza que la respuesta de la neurona satura, pero veremos que para construir redes neuronales profundas no son muy útiles.



Si volvemos a la figura original de esta clase descubrimos que nuestro perceptrón de una única capa se pude descomponer en M perceptrones con una única neurona de salida, a la cual llamamos *perceptrón simple*. Esta es la menor red que podemos construir, con una única neurona. A pesar de sus simplicidad, veremos que puede ser muy útil para realizar una regresión no lineal.



Supongamos que queremos modelar cierto proceso que relaciona, a través de una función, los vectores de $\mathbb{R}^n$ con vectores de $\mathbb{R}^M$, la cual asumimos que no podemos conocer. A nuestra neurona i de salida le corresponde aprender su parte

$$f(\vec{\xi}) = \zeta_i \qquad f : R^N \to R$$

Como carecemos de cualquier información a priori sobre la relación

matemática entre ζ_i y las componentes de $\vec{\xi}$, vamos a suponer que tenemos muchos buenos ejemplos de esta relación para compensar nuestra ignorancia.

Dada cierta función f desconocida y llamada *función objetivo*, deseamos poder hacer estimaciones *adecuadas* de su valor para cualquier valor de la entrada de dicha función, a partir de un conjunto de puntos entrada–salida conocidos a los cuales denominamos *conjunto de entrenamiento*.

Podemos pensar este problema como una *aproximación neuronal* a la función f a partir de puntos conocidos (o medidos), o sea, como una regresión no lineal.

Suponemos que tenemos p pares de entrada–salida de la forma

$$C = \left\{\left(\vec{\xi}^{\alpha}; \zeta^{\alpha}\right)\right\} \qquad \alpha = 1, 2, \ldots, p\,,$$

al cual llamamos **CONJUNTO DE ENTRENAMIENTO**.

CONJUNTO DE ENTRENAMIENTO

x_1	x_2	x_3	x_4	x_N	y_i
0.987	0.987	-1,491	2,233	7,129	0,816
2,765	0,019	-7,428	2,185	2,391	0,129
-4,318	0,779	-4,678	2,398	5,656	0,763
15,897	0,614	-2,231	2,783	4,417	0,665
-4,765	0,332	-0,012	2,365	3,687	0,329
1,823	0,442	-7,992	2,173	2,546	0,417
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
4,561	0,629	-2,544	2,711	3,387	0,417

$$\vec{\xi}^{\alpha} = (\xi_1^\alpha, \xi_2^\alpha, \ldots, \xi_N^\alpha) \rightarrow \zeta_i^\alpha \in R$$

Notemos la diferencia entre lo que el experto nos dice nos debe dar como resulado del cálculo que hace la neurona, a lo cual llamamos ζ_i^μ, y lo que en verdad nos da la neurona, O_i^μ. Notemos también que, a pesar de tener una única neurona de salida, hemos mantenido el subíndice i a los fines de poder construir luego redes más grandes aglomerando estas neuronas simples. La operación de nuestra neurona i está dada por la expresión:

$$O_i^\alpha = g(h_i^\alpha - \mu_i) = g(\vec{W}i^\alpha \cdot \vec{\xi}^\alpha - \mu_i)\,.$$

Nuestro desafío ahora es encontrar un conjunto de N sinapsis y un umbral μ_i que hagan posible que nuestra neurona, con su cálculo simple, aproxime suficientemente bien los datos del conjunto de entrenamiento provistos por el experto. O sea,

$$O_i^\alpha = g(\vec{W}^\alpha \cdot \vec{\xi}^\alpha\, i - \mu_i) \approx \zeta_i^\alpha .$$

Veremos en las próximas clases que si nuestro conjunto de entrenamiento es suficientemente grande y es una buena muestra del universo de posibles asignaciones entre $\vec{\xi}$ y ζ, entonces la red aprenderá a resolver bastante bien el conjunto de entrenamiento. Más aún, podrá hacer buenas estimación de la función f para valores de entrada que no están en el conjunto de entrenamiento y por lo tanto no se usaron en la construcción de las sinapsis y el umbral. O sea, podrá **generalización**.

El algoritmo de descenso por el gradiente

A partir de ahora vamos a presentar el algoritmo de descenso por el gradiente para poder asignar a nuestra neurona valores adecuados de los N valores de las sinapsis $\vec{W}_i$ y el correspondiente umbral μ_i.

Lo primero que supondremos es que *inicialmente* le asignamos a las N sinapsis y al umbral valores aleatorios. En principio, por comodidad y simplicidad, asumiremos que escogemos estos $N+1$ valores con una distribución de probabilidad de valor medio nulo y con desviación estándar 1, y que las elegimos independientes entre si.

A seguir suponemos que le presentamos a la red una tras otra las entradas $\vec{\xi}^{\alpha}$ y calculamos la correspondiente salida O_i^{α}. Con esta salida para cada entrada podemos calcular la diferencia entre lo que dio la red O_i^{α} y lo que debió dar, ζ_i^{α}:

$$(O_i^{\alpha} - \zeta_i^{\alpha}).$$

Con estas diferencias vamos a definir el Error Cuadrático (Ec) como solemos hacer en cualquier curso de estadística:

$$Ec = \sum_{\alpha=1}^{p} (\zeta_i^{\alpha} - O_i^{\alpha})^2.$$

Hemos construido el error como la suma de las diferencias de los cuadrados para evitar que los términos se compensen por diferencias de signos en las diferencias.

Cada vez que le hemos presentado todos los elementos del conjunto de entrenamiento a la red, decimos que ha pasado una **época**.

Recordemos que O_i^α es función de las N sinapsis y del umbral en tanto ζ_i^α es un parámetro fijo que nos brindó el experto y al cual no podemos variar (no es una variable).

$$Ec = \sum_{\alpha=1}^{p} \left(\zeta_i^\alpha - g(\vec{W}_i^\alpha \cdot \vec{\xi^\alpha} - \mu_i) \right)^2$$

El vector inicial $\vec{W}_i$ y las todas las entradas $\vec{\xi}^{\alpha}$ $(\alpha = 1, 2, \ldots, p)$ viven en el espacio R^N.

Una vez que le presetamos todos los elementos del conjunto de entrenamiento calculamos la función *Ec*. Una forma de cambiar las sinapsis y el umbral de forma tal que podamos disminuir el valor del error *Ec* es usar el método de descenso por el gradiente. Para ello hacemos:

$$\begin{aligned} \vec{W}_i^{nuevo} &= \vec{W}_i^{anterior} + \Delta \vec{W}_i \\ \mu_i^{nuevo} &= \mu_i^{anterior} + \Delta \mu_i \end{aligned}$$

donde

$$\Delta \vec{W}_i = -\eta \nabla E_c \qquad \Delta \mu_i = -\eta \frac{\partial Ec}{\partial \mu_i}$$

Les dejo la forma de los incrementos, en sus componentes:

$$\frac{\partial Ec}{\partial W_{ik}} = -\sum_{\alpha=1}^{p} [\zeta_i^\alpha - O_i^\alpha]\, g'(h_i^\alpha - \mu_i)\, \xi_k^\alpha$$

Llamamos

$$\delta_i^\alpha = [\zeta_i^\alpha - O_i^\alpha]\, g'(h_i^\alpha - \mu_i)$$

con lo cual

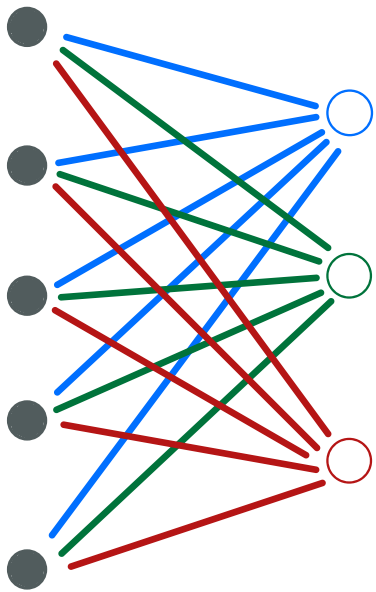
$$\Delta W_{ik} = \eta \sum_{\alpha=1}^{p} \delta_i^\alpha\, \xi_k^\alpha \qquad \Delta\mu_i = -\eta \sum_{\alpha=1}^{p} \delta_i^\alpha$$

Para la función sigmoide se cumple que

$$g'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = g(x)\,(1 - g(x))$$

lo cual nos ahorra mucho esfuerzo numérico, pues la función sigmoidea requiere muchos cálculos.

Ahora podemos volver a nuestra red de una capa oculta con M neuronas en la capa de salida.



La forma de la función *Ec* es la suma de los errores para cada neurona:

$$Ec = \sum_{\alpha=1}^{p} \sum_{i=1}^{M} \left(\zeta_i^{\alpha} - g(\vec{W}_i^{\alpha} \cdot \vec{\xi^{\alpha}} - \mu_i) \right)^2$$

Los procedimientos son los mismos y las expresiones de los incrementos solo agregan la suma sobre las neuronas de salida:

$$\Delta W_{ik} = +\eta \sum_{\alpha=1}^{p} \sum_{i=1}^{M} \delta_i^{\alpha} \, \xi_k^{\alpha} \, ,$$

$$\Delta \mu_i = -\eta \sum_{\alpha=1}^{p} \sum_{i=1}^{M} [\zeta_i^{\alpha} - {}_i^{\alpha}] \, .$$