REDES NEURONALES 2024

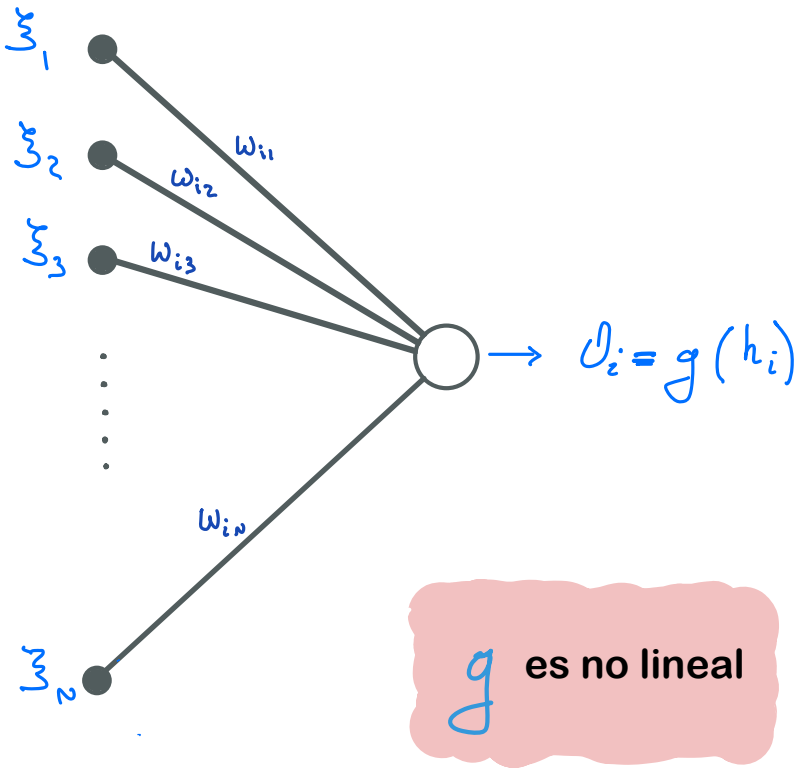
Clase 17 parte 2

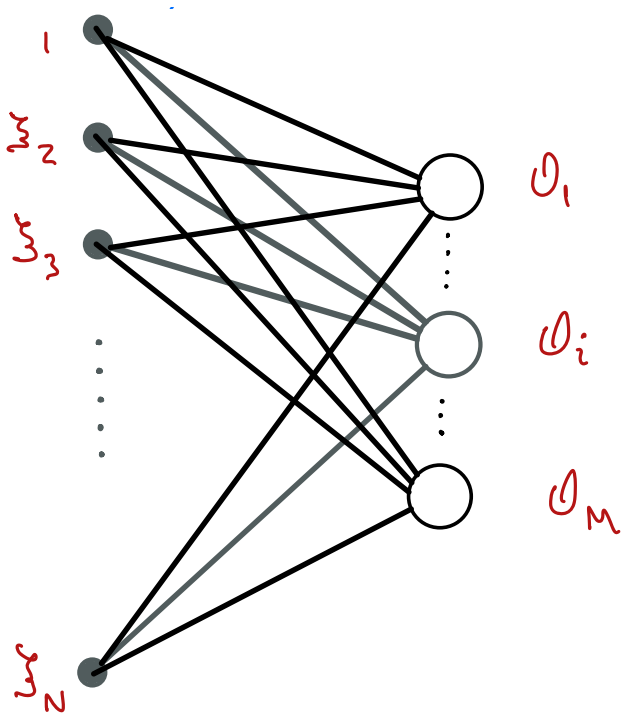
Jueves 17 de octubre 2024

FAMAF, UNIVERSIDAD NACIONAL DE CÓRDOBA

INSTITUTO DE FÍSICA ENRIQUE GAVIOLA (UNC-CONICET)

EL PERCEPTRÓN SIMPLE CON SALIDA NO LINEAL





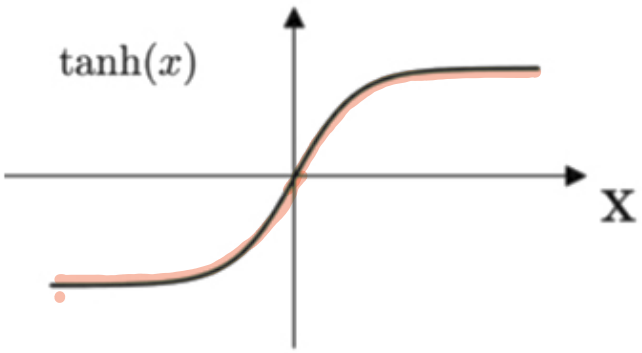
En la ante últimas clase analizamos el perceptrón simple con salida binaria y vimos que con él podemos *clasificar*, bajo ciertas condiciones, en dos categorías. Esto es posible cuando el conjunto de entrenamiento es *linealmente separable*. Vimos que existe un algoritmo llamado *REGLA DEL PERCEPTRÓN* para encontrar parámetros (sinapsis y umbral) adecuados.

En la última clase analizamos el caso de un perceptrón simple con una salida lineal. Vimos que este perceptrón nos devuelve un número real y por lo tanto se usa para hacer una regresión, o sea, para hacer una estimación de la salida cuando la entrada no está en el conjunto de entrenamiento. Así pudimos definir una función *Error Cuadrático Medio E* que es un ejemplo de muchas funciones loss que podemos usar para mensurar el error que comete el perceptrón simple con salida lineal. Esto nos permitió encontrar un buen conjunto de parámetros (sinapsis y umbrales) usando el Método de Descenso por el Gradiente (MDG), el cual es *determinista*.

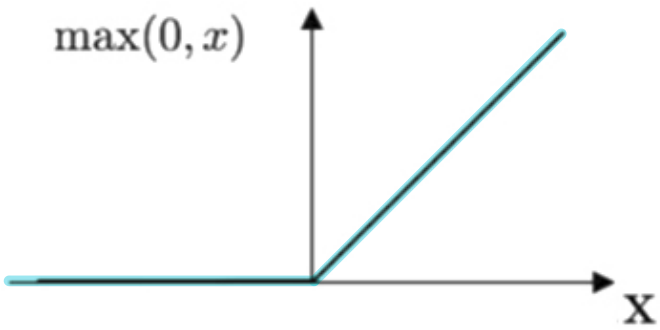
Hoy analizaremos el caso más general en el cual el perceptrón simple tiene una salida *NO LINEAL a la que denominamos g(z)*. Este es el caso más rico que veremos, y puede atenderer a muchas de nuestras necesidades. A medida que vayamos creando ensamblados más y más grandes de neuronas, preservaremos estas funciones no lineales.

FUNCIONES DE ACTIVACIÓN NO LINEALES

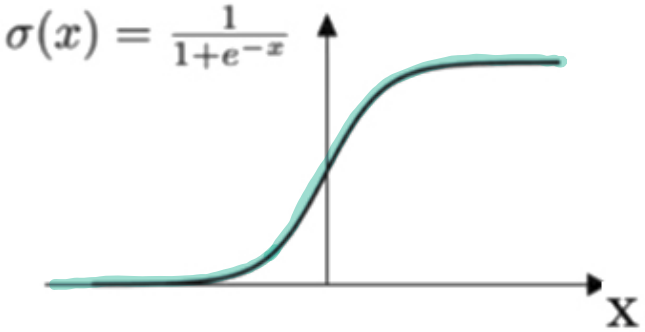
Tanh



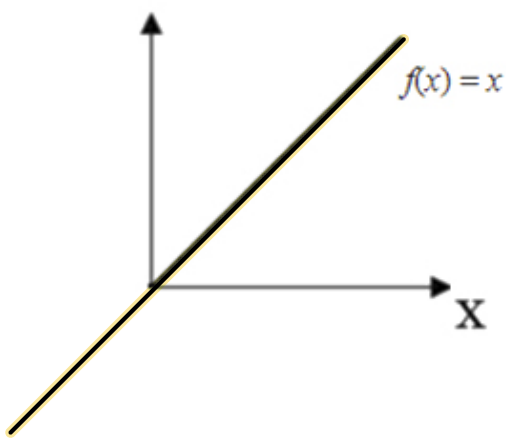
ReLU



Sigmoid



Linear



Ahora comenzaremos a implementar con mucho detalle el *Algoritmo de Descenso por el Gradiente* (ADG) que es el método que hoy permite implementar todas las formas de aprendizaje automático que utilizamos con éxito.

Lo primero que debemos hacer es asignar valores a cada una de las ***N*** sinapsis (una reservada para el umbral) de las ***M*** neuronas de salida. Estos son los parámetros que debemos mover para encontrar un buen mínimo de la función error ***E***.

¿Cuántos parámetros son? ***N x M***

A modo de ejemplo, Chat GPT-4 tiene 100.000.000.000.000, o sea, cien millones de millones de sinapsis y umbrales. Es importante que sepan que las funciones de activación son casi todas ReLU y que los parámetros (sinapsis y umbrales) se fijan por medio del gradiente por el descenso. El tipo de arquitectura es mucho más complejo, con muchas capas y muchísimas neuronas por capa, pero la esencia es la misma.

Salvo casos muy particulares y poco frecuentes, los valores iniciales de los ***NxM*** parámetros, sinapsis y umbrales, serán elegidos al azar, como variables independientes (uñas de otras). A modo de una posible elección entre muchas posibles, asumamos que los elegimos con una distribución normal (campana de Gauss) de valor medio cero y desviación estándar uno

Una vez que hemos asignado los valores iniciales de las sinapsis, incluidos los umbrales (los parámetros), sin perder de vista que las entradas no son neuronas, vamos a presentarle secuencialmente todas las entradas del conjunto de entrenamiento (que tiene p entradas), uno tras otro, y veremos qué obtenemos en cada una de las M neuronas de

Comenzamos con el primer elemento del conjunto de entrenamiento.

$$\bar{\xi}^1 \longrightarrow \bar{O}^1$$

Con esto calculamos el error debido a este ejemplo

$$E^1(w_1, w_2, \ldots, w_{N+1}) = \frac{1}{2}\sum_{i=1}^{N}\left(\xi_i^1 - O_i^1\right)^2$$

Vamos pasando por cada uno de los p elementos del conjunto de entrenamiento y así obtendremos el error total.

Repetimos esto pasando por todos los elementos del conjunto de entrenamiento.

Ahora podemos aplicar el método de descenso por el gradiente (MDG) con pocas variaciones si lo comparamos con el caso lineal. En particular podemos usar la misma definición de *Error Cuadrático Medio E(**W**)*, sin importar cual es la imagen de la función de salida *g(z)*.

$$E(\bar{w}) = \sum_{M=1}^{P} \frac{1}{2} \sum_{i=1}^{M} \left(\zeta_i^M - O_i^M \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{M} \sum_{M=1}^{P} \left(\zeta_i^M - g\left(h_i^M \right) \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{M} \sum_{M=1}^{P} \left(\zeta_i^M - g\left(\sum_{k=1}^{N} w_{ik} \xi_k^M \right) \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{M} \sum_{M=1}^{P} \left(\zeta_i^M - g\left(\bar{w}_i \cdot \bar{\xi}^M \right) \right)^2$$

Ahora podemos calcular cada componente *ik* del gradiente:

$$\frac{\partial E}{\partial w_{ik}} = -\frac{1}{2} \sum_{\mu=1}^{P} 2 \left(\xi_i^\mu - g\left(\sum_{k=1}^{N} w_{ik} \xi_i^\mu\right)\right) \left(g'(h_i^\mu)\right) \frac{d h_i^\mu}{d w_{ik}}$$

$$\boxed{\frac{\partial E}{\partial w_{ik}} = -\sum_{\mu=1}^{P} \left(\xi_i^\mu - O_i^\mu\right) g'(h_i^\mu) \xi_k^\mu}$$

Esta componente del gradiente es casi idéntica a la obtenida en la clase pasada para la salida lineal. La única diferencia es la aparición de la derivada de la función no lineal de activación *g'(h)*.

$$\boxed{\frac{\partial E}{\partial w_{ik}} = -\sum_{\mu=1}^{P} \left(\xi_i^\mu - O_i^\mu\right) \xi_k^\mu}$$

caso lineal

Pero la derivada de la función identidad es uno, con lo cual la fórmula obtenida con la neurona no lineal incluye al caso lineal.

Siguiendo lo hecho en la última clase redefinimos:

$$\delta_i^\mu \equiv \left[\xi_i^\mu - O_i^\mu\right] g'(h_i^\mu)$$

con lo cual

$$\frac{\partial E}{\partial w_{ik}} = -\sum_{\mu=1}^{p} \delta_i^\mu \, \xi_k^\mu$$

Cuando aplicamos el descenso por el gradiente a la sinapsis W nos queda

$$\overline{w}_i^{\text{nuevo}} = \overline{w}_i^{\text{viejo}} + \Delta \overline{w}_i$$

$$w_{ik}^{\text{nuevo}} = w_{ik}^{\text{viejo}} + \Delta w_{ik}$$

$$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = \eta \sum_{\mu=1}^{p} \delta_i^\mu \, \xi_k^\mu$$

Ahora podemos hacer lo mismo que hicimos con las neuronas de salida lineales, o sea, dado el conjunto de entrenamiento

$$\mathcal{L} = \left\{ \left(\overline{\xi}^\mu, \zeta_i^\mu \right) \right\} \qquad \mu = 1, 2, \ldots, p$$

podemos comenzar con valor de $\overline{\boldsymbol{w}}$ aleatorio y cambiarlo algorítmicamente usando el descenso por el gradiente durante tantas épocas como haga falta para al alcanzar un error E final menor que cierta tolerancia que nos imponemos de antemano.

Sobre las derivadas y su complejidad

Tangente hiperbólica

$$g(h) = \tanh(h) = \frac{\operatorname{senh}(h)}{\cosh(h)}$$

$$g'(h) = \frac{\cosh^2(h) - \operatorname{senh}^2(h)}{\cosh^2(h)} = 1 - \tanh^2(h) = 1 - g^2(h)$$

Sigmoide

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

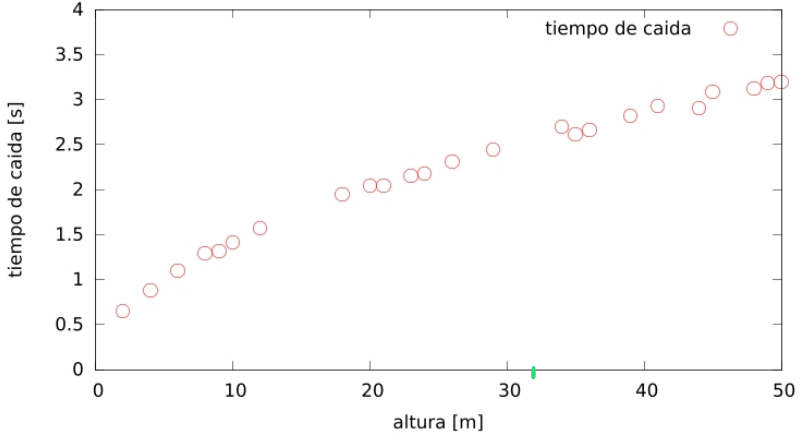
$$g'(h) = \frac{+ e^{-h}}{(1 + e^{-h})^2} = g(h)(1 - g(h))$$

Relu

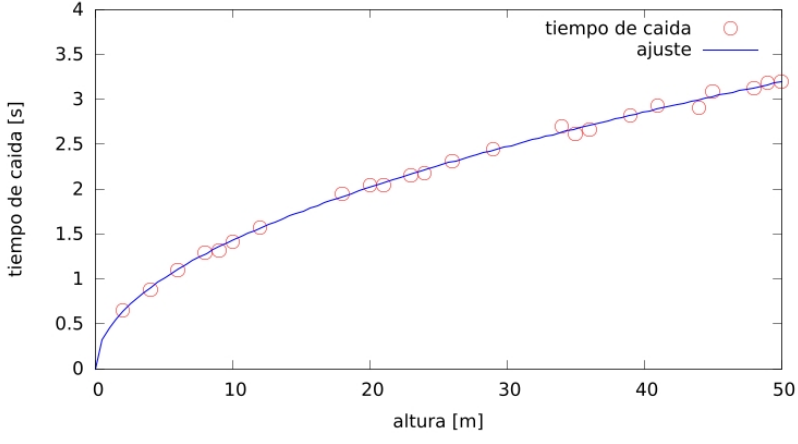
$$g(h) = \max(0, x)$$

$$g'(h) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$$

Un ejemplo simple de regresión



Un ejemplo simple de regresión



$$t = \sqrt{\frac{2\,x}{g}} \quad [s]$$

g = 9,76 m/s^2