Perceptrón

Nicolás Kossacoff

Noviembre 2024

1. Introducción

Las redes neuronales nacen a principios de 1940 con McCulloch y Pitts. Ambos desarrollaron la idea de neuronas artificiales como **unidades binarias**, las cuales computan una suma ponderada de los valores de entrada y la comparan con un umbral. Si la suma es mayor al umbral, la neurona se enciende, y si es menor se apaga.

Luego, en 1947, Donald Hebb tuvo la idea de que las neuronas aprenden al modificar el peso de sus conexiones. Esto se llamó **hiper-aprendizaje** y se basa en la idea de que si dos neuronas se activan conjuntamente, el peso de su conexión se incrementa.

En 1957, Frank Rosenblatt introduce el **perceptrón**, un algoritmo de aprendizaje que intenta replicar el funcionamiento de una red neuronal. En 1969, Minsky y Papert refinaron este modelo a su versión actual.

Comentario. *Después de haber investigado un poco sobre las diferencias entre un perceptrón y una neurona artifical (o solo neuronas para simplificar), encontré que las neuronas son una* ***generalización*** *de los perceptrones. Es decir, mientras que el perceptrón propuesto inicialmente por Rosenblatt aplicaba una función indicadora como función de activación, las neuronas actuales no tienen esa limitación y pueden utilizar funciones de activación más complejas.*

Otra diferencia importante que encontré es que el perceptrón es considerado un modelo por sí solos, mientras que la neurona es considerada una unidad básica dentro de una red neuronal. Una explicación más detallada se puede encontrar aquí.

2. Perceptrón

Supongamos que tenemos el vector de entrada, $X = (x_1, x_2, \ldots, x_p)$, y el vector de pesos, $W = (w_1, w_2, \ldots, w_p)$. Un perceptrón recibe los valores de entrada junto con sus pesos y devuelve su producto interno:

$$\langle W, X \rangle = \sum_{i=1}^{p} w_i x_i \tag{1}$$

Geométricamente, este producto interno no es más que un plano en dimensión p, el cual representa la frontera de decisión que utilizamos para discriminar entre clases.

Ahora, este producto escalar se suele comparar con un umbral, b, al cual llamamos **bias**[1]. De esta manera, nuestro perceptrón se activa si (1) es mayor al umbral, y se apaga en caso contrario. Podemos entonces reescribir la Ecuación (1) de la siguiente manera:

$$\left(\sum_{i=1}^{p} w_i x_i\right) - b = \sum_{i=0}^{p} w_i x_i \tag{2}$$

donde, para simplificar las cuentas, definimos $w_0 = b$ y $x_0 = -1$.

Por último, definimos una **función de activación**, la cual nos devuelve una transformación no lineal[2] de la Ecuación (2). Matemáticamente:

$$z = f\left(\sum_{i=0}^{p} w_i x_i\right) \tag{3}$$

donde z es la salida de nuestro perceptrón.

2.1. Entrenamiento

2.1.1. Función de pérdida

La función de pérdida[3] mide que tan buena es nuestra predicción, dado los pesos actuales, con respecto a las etiquetas de nuestra muestra de entrenamiento:

$$\mathcal{L}\left((X, y); W\right)$$

donde y es la etiqueta asociada a la observación X. Si $\mathcal{L} \gg 0$, entonces nuestra predicción es muy distinta al verdadero valor de y (ground-truth), y si $\mathcal{L} \approx 0$, entonces nuestra predicción es muy parecida al valor de y.

Dicho esto, queremos encontrar los pesos W que minimizan la función de pérdida, es decir, que mejor ajustan a nuestros datos.

2.1.2. Optimización

Los pesos óptimos se obtienen a partir del siguiente problema de optimización:

$$\hat{W} = \arg\min_{W} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left((X, d)_i; W\right) \tag{4}$$

[1] La razón por la cual incluimos es porque nos permite desplazar nuestra frontera de decisión en caso de que lo necesitemos.

[2] Definimos a la la función de activación como una función no lineal porque, si no lo fuese, tendríamos un modelo lineal como resultado. Además, cuando esta función es no lineal, podemos capturar efectos no lineales o interacciones entre features que con una función lineal no podríamos.

[3] La forma funcional que toma la función de pérdida depende del tipo de problema que estemos resolviendo (i.e., clasificación o regresión). Para problemas de regresión se suele utilizar el error cuadrático medio y para problemas de clasificación se suele utilizar la entropía cruzada.

Hay muchos métodos que nos permiten minimizar la función de pérdida, pero el más utilizado es el del **gradiente descendente**.

2.2. Problemas

Como vimos anteriormente, la frontera de decisión que construye un perceptrón es lineal. Esto quiere decir que no es un modelo que nos permita resolver problemas no lineales, como es el caso de las funciones `XOR`.

Ejemplo. *La función* `XOR` *toma los siguientes valores:*

x_1	x_2	`XOR`	y
0	0	0	0
1	0	1	1
0	1	1	1
1	1	1	0

Si graficamos una frontera de decisión arbitraria para nuestro modelo, vamos a poder observar que no hay manera de separar ambas clases (positiva y negativa) de forma lineal.

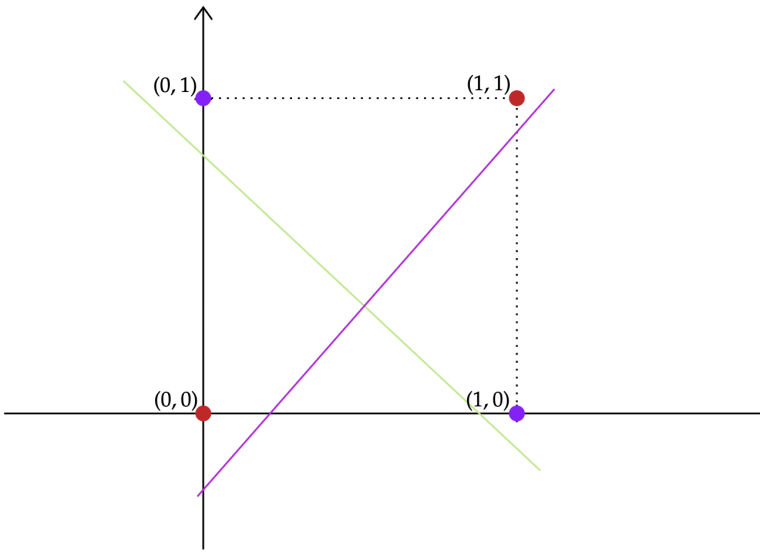

Figura 1: Podemos ver como la frontera de decisión generada por el perceptrón no logra discriminar correctamente entre clases. La frontera de color verde clasifica bien a las observaciones positivas, pero clasificando mal una de las observaciones negativas. Por otro lado, la frontera de color violeta logra clasificar bien las observaciones negativas, pero clasifica mal a una de las observaciones positivas.