7 | Perceptrón, unidad fundamental de las redes neuronales

Luis Alfredo Lizárraga Santos

Objetivo

Conocer el funcionamiento de la unidad elemental con la cual se construyen las redes neuronales: el perceptrón, y lograr implementar un perceptrón simple que aprenda las operaciones `AND` y `OR` para tres variables.

Introducción

"Una red neuronal se puede definir como un modelo de razonamiento basado en el cerebro humano" (Negnevitsky 2005, pág. 166). Basándonos en el libro de Negnevisky explicaremos cómo funciona el cerebro, cómo las Redes Neuronales modelan el cerebro, cómo aprenden y finalizaremos con el Perceptrón (Negnevitsky 2005, cap. 6).

Redes Neuronales Artificiales

Sabemos que el cerebro consiste en un conjunto densamente interconectado de unidades básicas de procesamiento, llamadas neuronas. El cerebro humano contiene cerca de 86 mil millones de neuronas (Herculano-Houzel 2009) y entre 100 y 500 billones de conexiones (Drachman 2005), llamadas **sinapsis**.

A pesar de que cada neurona tiene una estructura muy simple, un conjunto (aunque sea pequeño) de estos elementos constituye un poder de procesamiento enorme. Una neurona está constituida por un cuerpo celular, llamado **soma**, un conjunto de fibras llamadas **dendritas**, y una única fibra larga llamada **axón**. La figura 7.1 representa dos neuronas conectadas.

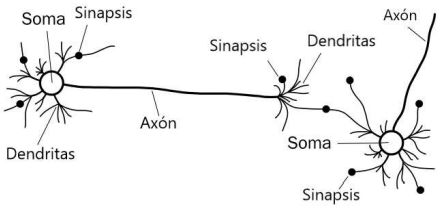

Figura 7.1 Partes de una neurona biológica. (Negnevitsky 2005, pág. 166)

Señales se propagan de una neurona a otra por medio de reacciones electroquímicas complejas. Substancias químicas que se liberan desde las sinapsis causan un cambio en el potencial eléctrico del cuerpo celular de la neurona. Cuando este potencial sobrepasa su umbral, una señal eléctrica, llamada **potencial de acción**, se manda a través del axón. Este pulso se dispersa y eventualmente llega a las sinapsis, haciendo que estas incrementen o decrementen su potencial. Pero el descubrimiento más interesante es que las neuronas exhiben **plasticidad**.

Esta plasticidad permite que las conexiones hacia las neuronas que conducen a la "respuesta correcta" se vean fortalecidas, mientras que las conexiones que llevan a la "respuesta equivocada" sean debilitadas. Como resultado, las redes neuronales tienen la habilidad de aprender mediante la experiencia.

El cerebro, muchas neuronas conectadas

Las neuronas de la red neuronal artificial se conectan entre sí usando conexiones con un peso dado, pasando señales de una neurona a otra. Cada neurona recibe un número de señales de entrada a través de sus conexiones, pero sólo produce una salida, como se muestra en la figura 7.2. Esta señal de salida se transmite por la conexión saliente de la neurona (lo equivalente al axón en la neurona biológica). Esta conexión saliente, a su vez, se separa en varias ramas que transmiten la misma señal. Estas ramas terminan como conexiones de entrada de otras neuronas en la red.

Aprendizaje

Las neuronas se conectan mediante enlaces que tienen un peso asociado a ellos. Estos pesos son los medios para guardar información a largo plazo. Expresan la importancia de cada entrada de la neurona. Entonces, las redes neuronales "aprenden" por medio de ajustes a estos pesos.

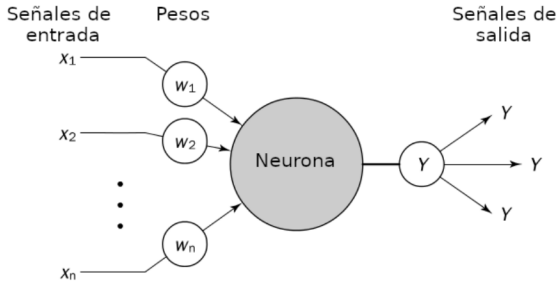

Figura 7.2 Diagrama de un perceptrón. (Negnevitsky 2005, pág. 168)

Neuronas Artificiales

Cálculo de la salida

La neurona calcula la suma de las señales de entrada multiplicadas por el peso de su conexión con la neurona siguiente (combinación lineal) y compara este resultado con un umbral θ. Si el total es menor que el umbral, la salida de la neurona es (-1) , si es mayor entonces la neurona se activa y su salida es (1) . A lo descrito anteriormente se le llama función de activación.

Perceptrón

Un perceptrón es la forma más simple de una red neuronal, ya que representa a una sola neurona.

El perceptrón consiste en un combinador lineal seguido de una función de activación. Se le aplica la función umbral a la suma con pesos de las entradas menos el umbral, la cual produce una salida positiva si la suma es positiva, o una salida negativa si la suma es negativa. El objetivo del perceptrón es clasificar entradas en una de dos clases. Esto se expresa con la siguiente función:

$$Y = f\left(\sum_{i=1}^{n} x_i w_i - \theta\right)$$

(Negnevitsky 2005, pág. 169)

Aprendizaje para clasificación

Esto se logra haciendo pequeñas modificaciones a los pesos de las entradas para reducir las diferencias entre la salida obtenida y la salida deseada. El proceso de actualización de pesos es bastante simple. Como queremos obtener (o aproximarnos a) una salida

deseada, tenemos que iterar sobre un conjunto de entrenamiento (de tamaño m) las veces necesarias, modificando los pesos de las conexiones, hasta minimizar el error en la salida. Si para el ejemplar de entrenamiento j, la salida es $Y(j)$ y la salida deseada es $Y_d(j)$, entonces la función de error está dada por:

$$e(j) = Y_d(j) - Y(j)$$

(Negnevitsky 2005, pág. 171)

Si $e(j)$ es positivo, se necesita incrementar $Y(j)$, pero si es negativo, se necesita disminuirlo. Tomando en cuenta que cada entrada contribuye $x_i(j) \times w_i(j)$ a la entrada total $X(j)$, nos damos cuenta que si el valor de entrada $x_i(j)$ es positivo, aumentar su peso $w_i(j)$ incrementa la salida del perceptrón, mientras que si $x_i(j)$ es negativa, un aumento en su peso $w_i(j)$ disminuye el valor de salida del perceptrón. Por lo tanto se puede establecer la siguiente regla de aprendizaje:

$$w_i(j+1) = w_i(j) + \alpha \times x_i(j) \times e(j)$$

(Negnevitsky 2005, pág. 171)

donde α es la **taza de aprendizaje**, una variable no negativa menor a 1.

Resumen del algoritmo de aprendizaje

En pocas palabras, se necesitan cuatro pasos para que un perceptrón aprenda a clasificar las entradas (Negnevitsky 2005, pág. 172):

1. **Inicialización:** Fijar los pesos iniciales $w_1, w_2, ..., w_n$ y el umbral θ a números aleatorios en el rango $[-0.5, 0.5]$ (recomendado).

2. **Activación:** Para un ejemplar j, activar el perceptrón aplicando las entradas $x_1(j), x_2(j), ..., x_n(j)$.

   $$Y(j) = f\left(\sum_{i=1}^{n} x_i(j) w_i(j) - \theta\right)$$

   donde n es el número de entradas y f es la función de activación.

3. **Entrenamiento de pesos:** Se actualizan los pesos del perceptrón:

   $$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

   donde $\Delta w_i(t)$ es la corrección del peso al tiempo t para el ejemplar j, que se calcula de la siguiente forma:

   $$\Delta w_i(t) = \alpha \times x_i(j) \times e(j)$$

   El umbral θ se queda fijo en el valor que le fue asignado al inicio.

4. **Iteración:** Repetir los pasos 2 y 3 para cada ejemplar del conjunto de entrenamiento hasta minimizar lo mejor posible el error. Si el resultado no es satisfactorio, continuar ejecutando otra vez desde el primer ejemplar.

Desarrollo e implementación

Deberán crear dos perceptrones, uno que aprenda la operación `AND` y otro la operación `OR`, ambas de tres variables. Cada neurona tendrá cuatro entradas, tres para las entradas de la operación lógica y una para el sesgo θ.

x_1	x_2	x_3	$Salida$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(a) Operación AND

x_1	x_2	x_3	$Salida$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(b) Operación OR

Su aplicación deberá permitir elegir distintos conjuntos de entrenamiento, en particular, tienen que especificar al menos 5 conjuntos de entrenamiento :

1. `[[0,0,0,0],[1,1,1,1]]` para AND y OR
2. El conjunto que contenga todas las combinaciones posibles de entradas
3. Y tres conjuntos más con elementos distintos, tomados de las tablas mostradas anteriormente.

Con el formato:

$$[x_1, x_2, x_3, Salida]$$

Esto para que se den una idea ustedes de lo importante que son los conjuntos de entrenamiento de un perceptrón. También tienen que mostrar el proceso de entrenamiento, y una vez entrenado el perceptrón, debe permitir hacer consultas de estas operaciones lógicas.

Para probar su perceptrón, deberán utilizar el siguiente conjunto de entradas:

$$[[0,0,0],[1,0,1],[0,0,1],[1,1,1]]$$

PD: Se les recomienda que no se compliquen y usen la función escalón (*step function*) como función de activación.

Requisitos y resultados

Entreguen el código donde implementaron el perceptrón y su entrenamiento. Generen un reporte con observaciones de cómo se comporta el perceptrón con cada uno de los conjuntos de entrenamiento. Lo podrán programar en `Java` o `Python`. No olviden comentar su código.