Inteligencia Artificial
Práctica 9: Redes Neuronales

Verónica Esther Arriola Ríos
Pedro Rodríguez Zarazúa
Luis Alfredo Lizárraga Santos

Fecha de entrega: Miércoles 27 de Abril de 2016

1. Objetivo

Conocer el funcionamiento de las redes neuronales y logran implementar un perceptron simple que "aprenda" las operaciones `AND` y `OR`.

2. Introducción

Una red neuronal se puede definir como un modelo de razonamiento basado en el cerebro humano.

Sabemos que el cerebro consiste de un conjunto densamente interconectado de unidades básicas de procesamiento, llamadas neuronas. El cerebro humano hace uso de cerca de 60 mil millones de neuronas, y unas 79 billones de conexiones, 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 1 representa una red neuronal.

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.

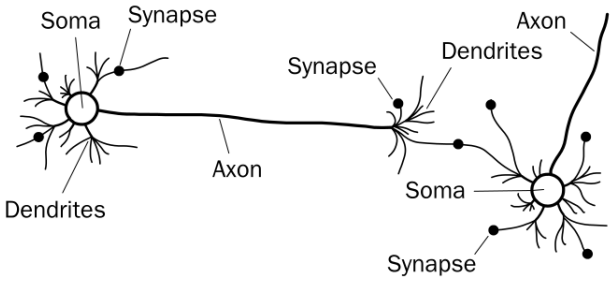

Figura 1: Red neuronal biológica.

3. Redes Neuronales Artificiales

3.1. ¿Cómo modelan al cerebro?

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. 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.

3.2. ¿Cómo aprenden?

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.

4. Neuronas Artificiales

4.1. ¿Cómo se determina la salida?

La neurona calcula la suma de las señales de entrada y su peso, 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

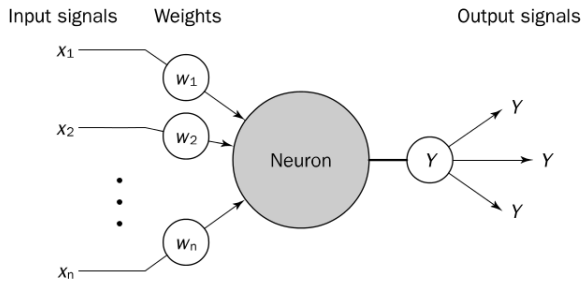

Figura 2: Diagrama de una neurona.

llama función de activación.

Hay varias funciones que se pueden llegar a utilizar, la figura 3 presenta algunas de ellas.

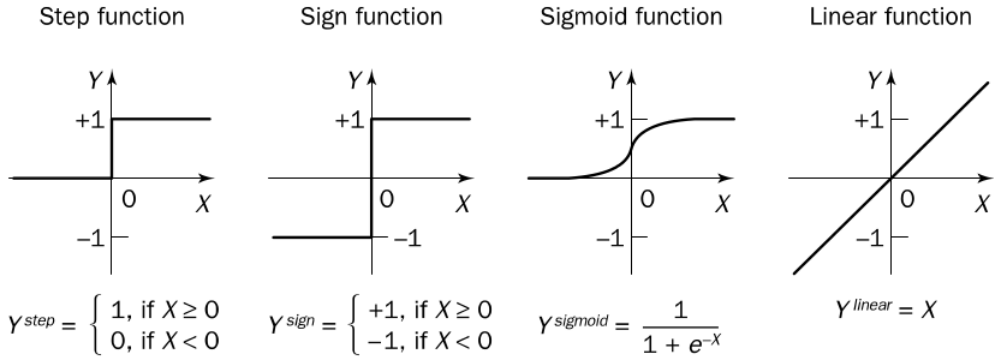

Figura 3: Funciones de activación de una neurona.

5. Perceptrón

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

El perceptrón consiste de un combinador lineal seguido de una función de limitación dura (*hard limiter*). Se le aplica la función de limitación a la suma con pesos de las entradas, 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.

5.1. Pero, ¿cómo aprende a clasificar?

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. Si en la iteración n, la salida es $Y(n)$ y la salida deseada es $Y_d(n)$, entonces la función de error está dada por:

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

donde n representa el n-ésimo ejemplar de entrenamiento.

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

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

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

6. Resumen del algoritmo de aprendizaje

En pocas palabras, se necesitan cuatro pasos para que un perceptron "aprenda" a clasificar las entradas:

1. **Inicialización:** Fijar los pesos iniciales $w_1, w_2, ..., w_j$ y el umbral θ a números aleatorios en el rango $[-0{,}5, 0{,}5]$ (recomendado).

2. **Activación:** Activar el perceptrón aplicando las entradas $x_1(n), x_2(n), ..., x_j(n)$ y la salida deseada $Y_d(n)$. Calcular la salida real en la iteración $n = 1$

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

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

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

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

   donde $\Delta w_i(n)$ es la corección del peso en la iteración n, que se calcula de la siguiente forma:

$$\Delta w_i(n) = \alpha \times x_i(n) \times e(n)$$

4. **Iteración:** aumentar n en uno, y repetir desde el paso 2 hasta converger (hasta que de la respuesta correcta).

7. Desarrollo e implementación

Deberán crear dos perceptrones, uno que "aprenda" la operación `AND` y otro la operación `OR`, cada una de cuatro entradas (tres para la compuerta y el sesgo).

Su aplicación deberá permitir elegir distintos conjuntos de entrenamiento, en particular, tienen que especificar al menos 5 conjuntos de entrenamiento, de los cuales dos deberán ser:

1. `[[0,0,0,0],[1,1,1,1]]` para AND y OR
2. El conjunto que contenga todas las combinaciones posibles de entradas

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 tiene que mostrar el proceso de entrenamiento, y una vez entrenado el perceptrón, debe permitir hacer consultas de estas operaciones lógicas.
Al final, generen un reporte con observaciones de cómo se comporta el perceptrón con cada uno de los conjuntos de entrenamiento.

PD: Se les recomienda que no se compliquen y usen la función "step" como función de activación.

Lo podrán programar en `Java` o `Python`. No olviden comentar su código.

8. Notas adicionales.

La práctica es individual, anexen a su código un archivo readme.txt con su nombre completo, número de cuenta, número de la práctica y cualquier observación o notas adicionales (posibles errores, complicaciones, opiniones, críticas de la práctica o del laboratorio, cualquier comentario relativo a la práctica).
Pueden agregar cualquier biblioteca extra, sólo asegurense de que se encuentre bien comentada.

Compriman la práctica en un solo archivo (.zip, .rar, .tar.gz) con la siguiente estructura:

- ApellidopaternoNombreNúmerodepráctica.zip (por ejemplo: LizarragaLuis09.zip)
  - ApellidopaternoNombreNúmerodepráctica (por ejemplo: LizarragaLuis09)
    - src

        - ◇ rNeuronal(.java / .py)
    - readme.txt

La práctica se entregará en la página del curso en la plataforma AVE Ciencias.
O por medio de correo electrónico a `luislizarraga@ciencias.unam.mx` con asunto `Práctica09[IA 2016-2]`

La fecha de entrega es hasta el día Miércoles 27 de Abril a las 23:59:59 hrs.

Referencias

[1] Michael Nengnevitsky *Artificial Intelligence: A guide to Intelligent Systems*, 2005 : Pearson Education Limited, Essex, England .