El perceptrón, su entrenamiento y Retropropagación

The perceptron, its training and Backpropagation

Autor: Jorge Alexander Osorio López
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: alexander.osorio@utp.edu.co

***Resumen*— Este documento inicialmente define al perceptrón, comparándolo con una neurona biológica. Luego, se definen sus componentes y los usos que tiene. A continuación, se explica por qué se debe entrenar un perceptrón, cómo hacerlo y un procedimiento que se puede utilizar para entrenar un solo perceptrón. Finalmente, se describe la importancia de utilizar el algoritmo de retropropagación cuando se está entrenando un conjunto de neuronas y se presenta su pseudocódigo.**

***Palabras clave*— perceptrón, redes neuronales, inteligencia artificial, retropropagación, neurona, entrenamiento, algoritmo, modelo.**

***Abstract*— This document initially defines the perceptron, comparing it to a biological neuron. Then, its components and the uses it has are defined. The following explains why a perceptron should be trained, how to do it, and a procedure that can be used to train a single perceptron. Finally, the importance of using the backpropagation algorithm when training a set of neurons is described and its pseudocode is presented.**

***Key Word*— perceptron, neuronal networks, artificial intelligence, backpropagation, neuron, training, algorithm, model.**

I. INTRODUCCIÓN

La inteligencia artificial es un campo que ha estado tomando mucha fuerza en los últimos años, ésta consiste en hacer posible que las máquinas adquieran habilidades y capacidades propias del cerebro humano empleando el aprendizaje profundo y el procesamiento del lenguaje natural. En pocas palabras la IA aprende gracias a los datos y logra clasificar la información para alcanzar ciertos objetivos [1].

Si hablamos de inteligencia artificial, debemos hablar también de las redes neuronales artificiales y una de ellas es el perceptrón, el objetivo de este artículo es dar a entender lo que es, su entrenamiento y además explicar el algoritmo de backpropagation.

1.1 PERCEPTRÓN

El modelo biológico más simple de un perceptrón es una neurona. La neurona es una célula especializada y caracterizada por poseer una cantidad indefinida de canales de entrada llamados dendritas y un canal de salida llamado axón. Las dendritas operan como sensores que recogen información de la región donde se hallan y la derivan hacia el cuerpo de la neurona que reacciona mediante una sinapsis que envía una respuesta hacia el cerebro, esto en el caso de los seres vivos.

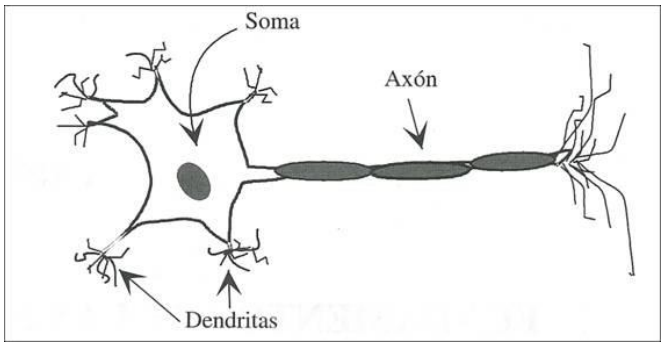

Figura 1. Estructura de una neurona biológica típica.

Una neurona sola y aislada carece de razón de ser. Su labor especializada se torna valiosa en la medida en que se asocia a otras neuronas, formando una red. Normalmente, el axón de una neurona entrega su información como señal de entrada a una dendrita de otra neurona y así sucesivamente. El perceptrón capta la señal en adelante se extiende formando una red de neuronas [2].

De manera similar tenemos entonces al perceptrón, éste es la forma más simple de una red neuronal usada para la clasificación de un tipo especial de patrones, los linealmente separables (es decir, patrones que se encuentran a ambos lados de un hiperplano).

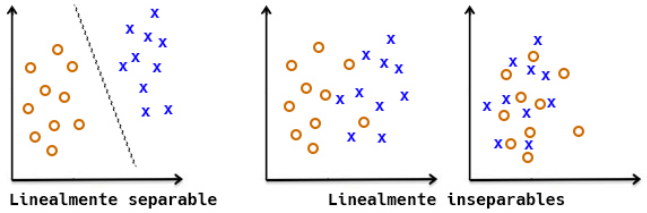

Figura 2. Comparación entre linealmente separable y linealmente inseparable.

Básicamente, consiste de una neurona con pesos sinápticos y umbral ajustables. El algoritmo usado para ajustar los parámetros libres de esta red neuronal apareció por primera vez en un procedimiento de aprendizaje desarrollado por Rosenblatt (1958) para su modelo de perceptrón del cerebro. En realidad, Rosenblatt demostró que, si los patrones usados para entrenar el perceptrón son sacados de dos clases linealmente separables, entonces el algoritmo del perceptrón converge y toma como superficie de decisión un hiperplano entre estas dos clases. La prueba de convergencia del algoritmo es conocida como el teorema de convergencia del perceptrón.

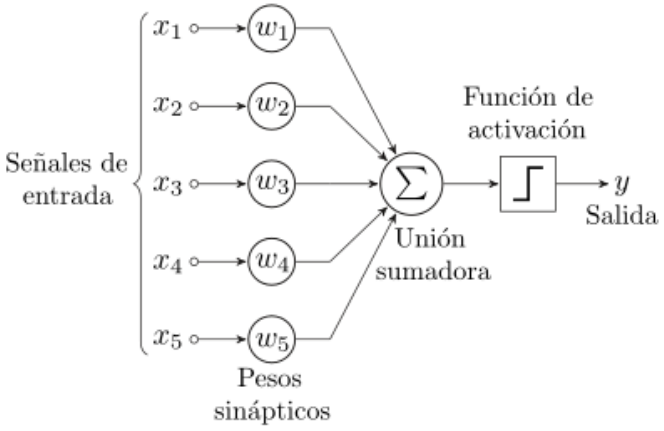

Figura 3. Diagrama de un perceptrón con cinco señales de entrada.

El perceptrón de una capa mostrado en la figura 3 tiene sólo una neurona. Dicho perceptrón está limitado a realizar clasificación de patrones con sólo dos clases. Expandiendo la capa de salida del perceptrón para incluir más que una neurona, podemos realizar dicha clasificación con más de dos clases. Sin embargo, las clases tendrían que ser linealmente separables para que el perceptrón trabaje correctamente.

El modelo de una neurona consiste en un combinador lineal seguido de un limitador. El nodo suma del mismo mide una combinación lineal de las entradas aplicadas a sus sinapsis y, además, representa un umbral aplicado externamente. La suma resultante es aplicada a un limitador. Así, de acuerdo a esto, la neurona produce una salida igual a +1 si la entrada del limitador es positiva, y -1 si es negativa.

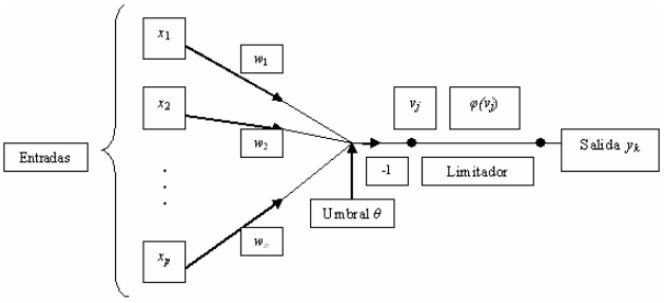

Figura 4. Modelo de un perceptrón simple

En la figura anterior, los pesos sinápticos son denotados por w1, w2, …, wp. Asimismo, las entradas aplicadas al perceptrón son denotadas por x1, x2, …, xp. El umbral externo es θ. Del modelo podemos ver que la salida del combinador lineal (es decir, la entrada del limitador) es la siguiente:

$$v = \sum_{i=1}^{p} w_i x_i - \theta$$

El propósito del perceptrón es clasificar un conjunto de estímulos aplicados externamente x1, x2, …, xp en una de dos clases, llamadas L1 o L2: La regla de decisión para la clasificación es asignar el punto representado por las entradas x1, x2, …, xp a la clase L1 si la salida del perceptrón es +1 y a la clase L2 si es -1.

Para adquirir capacidad de penetración en el comportamiento de un clasificador de patrones, es frecuente dibujar un mapa de las regiones de decisión en el espacio de señal p-dimensional abarcado por las p variables de entrada x1, x2, …, xp. En el caso de un perceptrón elemental, hay dos regiones de decisión separadas por un hiperplano, el cual está definido por la siguiente ecuación:

$$\sum_{i=1}^{p} w_i x_i - \theta = 0$$

En la siguiente figura se muestra para el caso de dos variables de entrada x1 y x2, para las que la región de decisión toma la forma de una línea recta. Un punto (x1,x2) que se encuentre por encima de la línea divisoria se asigna a la clase L1, y un punto (x1,x2) que se encuentre por debajo de dicha línea se asigna a la clase L2. Notemos, asimismo, que el efecto del umbral es desplazar la región de decisión del origen [3].

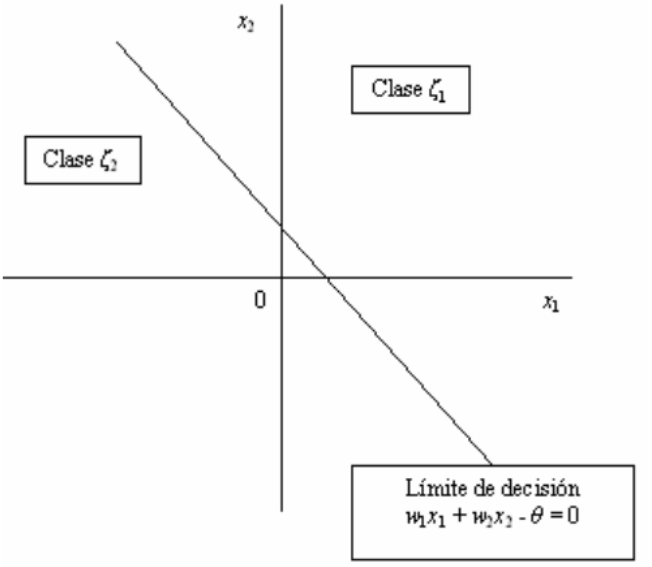

Figura 5. Regiones de decisión de un problema de clasificación de patrones de dos clases.

1.2 ENTRENAMIENTO DEL PERCEPTRÓN

El objetivo que persigue el entrenamiento de una red neuronal es básicamente es de establecer unos valores para el vector de pesos con los cuales error cometido al evaluar los ejemplos de entrenamiento sea mínimo. Una vez calculados estos pesos la red estará lista para ser probada con otros patrones de test con los cuales no ha sido entrenada. El objetivo de esta nueva comprobación es que hemos de probar como se comporta la red cuando las entradas son distintas a las usadas para el entrenamiento.

Al entrenar una red neuronal con unos determinados ejemplos de entrenamiento e intentar minimizar el error a unas cuotas ínfimas, corremos el riesgo de especializar demasiado nuestra red, la cual se comportará de una manera optima con los ejemplos con los cuales ha sido entrenada, pero para ejemplos con los que no se ha entrenado se pueden producir errores considerables.

Todo esto nos lleva la conclusión de que hay que minimizar el error, pero no hay que intentar rizar el rizo, ya que la especialización conlleva a una pérdida de generalización. Si este método de validación falla debemos optar por la opción de volver a entrenar la red desde el principio, pero con unos pesos iniciales aleatorios distintos a los del entrenamiento anterior.

Si los resultados han sido correctos, nuestra red estará lista para funcionar y podremos asegurar que para nuevas entradas desconocidas para la red esta dará resultados más o menos correctos. Aunque la fase de entrenamiento sea lenta y tediosa, una vez finalizada esta ya no hay que volver a entrenar, y además, como el cálculo de los resultados que ofrece la red es lineal, estos se obtienen de una manera muy rápida.

El entrenamiento de las redes neuronal se realiza mediante unos algoritmos de entrenamiento que se basan siempre en intentar buscar los pesos de las neuronas que ofrezcan mejores resultados. Más adelante se hablará del algoritmo de retropropagación.

Antes de comenzar a entrenar la red, lo primero que debemos hacer es definir la condición de parada. Esta condición puede ocurrir por varios motivos:

- Se ha alcanzado una cuota de error que consideramos suficientemente pequeña.
- Se ha llegado a un número máximo de iteraciones que hemos definido como tope para el entrenamiento.
- Se ha obtenido un error cero.
- Se ha llegado a un punto de saturación en el que por más que entrenemos ya no conseguimos reducir más el error.

Para comprender como se realiza el entrenamiento de una red de neuronas, lo primero que debemos de hacer es comprender el entrenamiento de una sola neurona, para luego acoplar varios entrenamientos individuales en uno colectivo para toda la red.

Para el entrenamiento de un perceptrón lo primero que debemos hacer es inicializar aleatoriamente su vector de pesos asociados, y después ir actualizando este para conseguir mejores resultados. Para cada iteración del algoritmo se actualizará el vector de pesos:

$$\Delta w_i = \eta(t - o)x_i$$

De manera que siendo n un parámetro al que denominaremos tasa de entrenamiento con un valor positivo y que hará la función de hacer que la convergencia sea más o menos rápida. Es recomendable que la tasa de entrenamiento sea pequeña (entre 0.1 y 0.2) para que los resultados sean correctos. Esta forma de recalcular los pesos se basa en el gradiente descendente de la función de error.

La función de error mide de una manera numérica la diferencia existente entre la salida ofrecida por la red y la salida correcta. Se calcula de la siguiente manera:

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

Donde D es el conjunto de ejemplos de entrenamientos, td es la salida correcta y od es la salida calculada por la red.

El método del gradiente descendente se basa en buscar la dirección en la que una pequeña variación del vector de pesos hace que el error decrezca más rápidamente. Para encontrar

esta dirección usamos la que nos marca el gradiente de la función de error con respecto al vector de pesos.

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_o}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

1.3 RETROPROPAGACIÓN

Cuando necesitamos representar problemas complejos, no nos basta únicamente con un simple perceptrón, sino que necesitamos una red de perceptrones interconectados entre ellos.

Para el entrenamiento de una red hemos de tener en cuenta que la salida de cada neurona no va a depender únicamente de las entradas del problema, sino que también depende de las salidas que ofrezcan el resto de las neuronas. Por este mismo motivo también podemos afirmar que el error cometido por una neurona no solo va a depender de que sus pesos sean los correctos o no, sino que dependerá del error que traiga acumulado del resto de neuronas que le precedan en la red.

Para controlar el error cometido hemos de redefinir la función de error, de tal forma que la nueva función de error es la siguiente:

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in salidas} (t_{kd} - o_{kd})^2$$

Donde w es el vector de pesos, D es el conjunto de ejemplos de entrenamiento, d es un ejemplo de entrenamiento concreto, salidas es el conjunto de neuronas de salida, k es una neurona de salida, tkd es la salida correcta que debería dar la neurona de salida k al aplicarle a la red el ejemplo de entrenamiento d y okd es la salida que calcula la neurona de salida k al aplicarle a la red el ejemplo de entrenamiento d.

Dentro de una red tenemos varios tipos de neuronas:

- Neuronas tipo nin. Serán las neuronas de entrada a la red.
- Neuronas tipo h. Serán las neuronas internas de la red.
- Neuronas tipo nout. Serán las neuronas de salida de la red.

Nótese que el número de neuronas de entrada a la red ha de ser igual al número de entradas del problema a resolver, y que es recomendable que el número de neuronas de salida sea igual al número de patrones distintos entre los que pretendamos clasificar los ejemplos de entrenamiento, así, un determinado ejemplo de entrenamiento pertenecerá al patrón que indique la neurona de salida cuya salida se aproxime más a una cota que nosotros definamos.

Cuando haya varias neuronas con un valor de aproximación a la cota establecida que sea muy similar, podremos decir que "lo más probable" es que la neurona sea de uno de esos patrones, sin embargo, cuando solo haya una salida que se aproxime claramente más que el resto a la cota clasificaremos ese ejemplo dentro del patrón que indique esa neurona.

Tenemos a continuación el pseudocódigo del algoritmo de retropropagación, teniendo en cuenta que n es la tasa de entrenamiento, nin el número de neuronas de entrada de la red, nhidden el número de neuronas internas de la red y nout el número de neuronas de salida de la red.

Retropropagación(ejemplos, n, nin, nhidden, nout)
    Crear la red con el número de neuronas indicado.
    Inicializar todos los pesos de todas las neuronas a un valor aleatorio.
    Hasta que se cumpla la condición de parada hacer:
        Propagar las entradas a través de la red y obtener las salidas.
        Propagar los errores de atrás hacia adelante:
            Para cada neurona de salida k calcular:

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

            Para cada neurona intermedia h calcular:

$$\delta_k \leftarrow o_k(1 - o_k) \sum_{k \in salidas} w_{kh}\delta_k$$

            Actualizar cada peso Wji de la red:

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

            Con:

$$\Delta w_{ji} = \eta(\delta_j)x_{ji}$$

Fin Retropropagación [4].

REFERENCIAS

Referencias en la Web:

[1] https://www.crehana.com/co/blog/desarrollo-web/que-es-perceptron-algoritmo/#que-es-perceptron

[2] https://es.wikipedia.org/wiki/Perceptr%C3%B3n

[3] http://bibing.us.es/proyectos/abreproy/11084/fichero/Memoria+por+cap%C3%ADtulos+%252FCap%C3%ADtulo+4.pdf+#:~:text=El%20perceptr%C3%B3n%20es%20la%20forma,ambos%20lados%20de%20un%20hiperplano).&text=La%20prueba%20de%20convergencia%20del,teorema%20de%20convergencia%20del%20perceptr%C3%B3n.

[4] http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm#entrenamiento