Redes Neuronales

Aproximación teórica a sus aspectos fundamentales

Estructura y conceptos fundamentales sobre redes neuronales.

Sara Rincon Galeano

*Facultad de ingenierías, Universidad Tecnológica de Pereira, Pereira, Colombia*

sara.16@utp.edu.co

***Resumen*— El propósito de este documento es presentar de manera general los conceptos básicos acerca de las redes neuronales, comenzando por el perceptrón simple, como unidad fundamental.**

***Palabras clave— redes neuronales, perceptrón, capa, entrenamiento, aprendizaje autónomo, pesos, umbral, conjuntos de entrada y de salida.***

***Abstract*—The purpose of this document is to present in a general way the basic concepts about neural networks, starting with the simple perceptron, as a fundamental unit.**

***KeyWords* —neural networks, perceptron, layer, training, autonomous learning, weights, threshold, input and output sets.**

1. INTRODUCCIÓN

Warren McCulloch y Walter Pitts 2​ (1943) crearon un modelo informático para redes neuronales, que se llama lógica umbral, que se base en las matemáticas y los algoritmos. Este modelo señaló el camino para que la investigación de redes neuronales se divida en dos enfoques distintos. Un enfoque se centró en los procesos biológicos en el cerebro y el otro se centró en la aplicación de redes neuronales para la inteligencia artificial.

A finales de la década de 1940 el psicólogo Donald Hebb creó una hipótesis de aprendizaje basado en el mecanismo de plasticidad neuronal que ahora se conoce como aprendizaje de Hebb. Aprendizaje de Hebb se considera que es un "típico" de aprendizaje no supervisado y sus variantes posteriores fueron los primeros modelos de la potenciación a largo plazo. Los investigadores empezaron a aplicar estas ideas a los modelos computacionales en 1948 con la sugerencia de Turing, que el córtex humano infantil es lo que llamaba "máquina desorganizada" (también conocido como "máqina Turing Tipo B").5​6​

Farley y Wesley A. Clark7​ (1954) al principio utilizaron máquinas computacionales, que entonces se llamaban "calculadoras", para simular una red de Hebb en el MIT. Otras simulaciones de redes neuronales por computadora han sido creadas por Rochester, Holanda, Hábito y Duda (1956).8​

Frank Rosenblatt9​ 10​ (1958) creó el perceptrón, un algoritmo de reconocimiento de patrones basado en una red de aprendizaje de computadora de dos capas, que utilizaba adición y sustracción simples. Con la notación matemática, Rosenblatt también describe circuitería que no está en el perceptrón básico, tal como el circuito de o-exclusiva, un circuito que no se pudo procesar por redes neuronales antes de la creación del algoritmo de propagación hacia atrás por Paul Werbos (1975).11​

En 1959, un modelo biológico propuesto por dos laureados de los Premios Nobel, David H. Hubel y Torsten Wiesel, estaba basado en su descubrimiento de dos tipos de células en la corteza visual primaria: células simples y células complejas.12​

El primer reporte sobre redes funcionales multicapas fue publicado en 1965 por Ivakhnenko y Lapa, y se conoce como el método de agrupamiento para el manejo de datos.13​14​15​

La investigación de redes neuronales se estancó después de la publicación de la investigación de aprendizaje automático por Marvin Minsky y Seymour Papert (1969)16​, que descubrió dos cuestiones fundamentales con las máquinas computacionales que procesan las redes neuronales. La primera fue que los perceptrones básicos eran incapaces de procesar el circuito de o-exclusivo. La segunda cuestión importante era que los ordenadores no tenían suficiente poder de procesamiento para manejar eficazmente el gran tiempo de ejecución requerido por las grandes redes neuronales.

Propagación hacia atrás y el resurgimiento

Un avance clave, fue el algoritmo de propagación hacia atrás, que resuelve eficazmente el problema de o-exclusivo, y en general el problema del entrenamiento rápido de redes neuronales de múltiples capas (Werbos 1975). El proceso de propagación hacia atrás utiliza la diferencia entre el resultado producido y el resultado deseado para cambiar los "pesos" de las conexiones entre las neuronas artificiales.

Las redes neuronales, tal como se utilizan en la inteligencia artificial, han sido consideradas tradicionalmente como modelos simplificados de procesamiento neuronal en el cerebro, a pesar de que la relación entre este modelo y la arquitectura biológica del cerebro se debate; no está claro en qué medida las redes neuronales artificiales reflejan el funcionamiento cerebral.

Máquinas de soporte vectorial y otros métodos mucho más simples, tales como los clasificadores lineales, alcanzaron gradualmente popularidad en el aprendizaje automático. No obstante, el uso de redes neuronales ha cambiado algunos campos, tales como la predicción de las estructuras de las proteínas.

El problema del desvanecimiento del gradiente afecta las redes neuronales pre-alimentadas de múltiples capas, que usan la propagación hacia atrás, y también los redes neuronales recurrentes (RNNs).Aunque los errores se propagan de una capa a otra, disminuye exponencialmente con el número de capas, y eso impide el ajuste hacia atrás de los pesos de las neuronas basado en esos errores. Las redes profundas se ven particularmente afectadas.

Para vencer este problema, Schmidhuber adopta una jerarquía multicapa de redes (1992) pre entrenados, una capa a la vez, por aprendizaje no supervisado, y refinado por propagación hacia atrás.​Behnke (2003) contaba solamente con el signo del gradiente (Rprop) tratándose de problemas tales como la reconstrucción de imágenes y la localización de caras.

1. PERCEPTRÓN SIMPLE

El Perceptrón está constituido por un conjunto de sensores de entrada que reciben los patrones de entrada a reconocer o clasificar una neurona de salida que se ocupa de clasificar a los patrones de entrada en dos clases, según que la salida de la misma sea 1 (activada) o 0 (desactivada).

Supongamos que tenemos una función f de Rn en {-1,1}, que aplica un patrón de entrada x = (x1,x2,…,xn) T ∈ Rn en la salida deseada z ∈ {-1,1}, es decir, f(x) = z. La información de que disponemos sobre dicha función viene dada por p pares de patrones de entrenamiento.

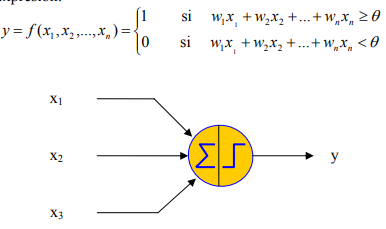
{x 1 ,z 1 }, {x2 ,z 2 },…,{ xp ,z p }

donde x i ∈Rn y f (xi ) = zi ∈{-1,1}, i=1,2,…,p.

Dicha función realiza una partición en el espacio Rn de patrones de entrada; por una parte estarían los patrones con salida +1 y por otra parte los patrones con salida −1. Por lo tanto, diremos que la función f clasifica a los patrones de entrada en dos clases. Ejemplos de funciones f de este tipo son la función lógica OR o la función par. Ahora vamos a construir un dispositivo sencillo que aprenda dicha función a partir de un conjunto conocido de patrones (relaciones) de entrenamiento. Para ello vamos a utilizar una unidad de proceso bipolar que como vimos es una función matemática con dominio el conjunto n-dimensional {-1,1}n y rango el conjunto {-1,1}, definida por la siguiente expresión:

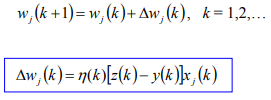


donde los parámetros w1,w2,…,wn, se llaman pesos sinápticos y son los pesos con los que se ponderan los valores de entrada x1,x2,…,xn, o argumentos de la función; la suma ponderada n u = w x + w x + ... + w x 1 1 2 2 se llama potencial sináptico y el parámetro θ se llama umbral o sesgo. También se puede expresar la función f mediante la función signo, y diremos que en este caso la función de transferencia es la función signo. Análogamente, se define una unidad de proceso binaria como una función matemática con dominio el conjunto n-dimensional {0,1}n y rango el conjunto {0,1}, definida por la siguiente expresión:



Cuando la salida de la unidad de proceso es igual a 1 se dice que dicha unidad de proceso está activada o encendida y presenta el estado 1, mientras que si su salida es igual a cero se dice que está desactivada o apagada, presentando el estado 0.

Para la determinación de los pesos sinápticos y del umbral vamos a seguir un proceso adaptativo que consiste en comenzar con unos valores iniciales aleatorios e ir modificándolos iterativamente, cuando la salida de la unidad no coincide con la salida deseada. La regla que vamos a seguir para modificar los pesos sinápticos se conoce con el nombre de regla de aprendizaje del Perceptrón simple y viene dada por la expresión:



esto nos indica que la variación del peso wj es proporcional al producto del error z() () k − y k por la componente j-ésima del patrón de entrada que hemos introducido en la iteración k, es decir, x ( ) k j . La constante de proporcionalidad η(k) es un parámetro positivo que se llama tasa de aprendizaje puesto que cuanto mayor es más se modifica el peso sináptico y viceversa. Es decir, es el parámetro que controla el proceso de aprendizaje. Cuando es muy pequeño la red aprende poco a poco. Cuando se toma constante en todas las iteraciones, η(k ) = η > 0 tendremos la regla de adaptación con incremento fijo.

Por lo tanto, esta regla de aprendizaje es un método de detección del error y corrección. Solo aprende, es decir, modifica los pesos, cuando se equivoca. Cuando tenemos un patrón que pertenece a la primera clase (z(k)=1) y no es asignado a la misma, entonces corrige el valor del peso sináptico añadiendoles una cantidad proporcional al valor de entrada, es decir lo refuerza, mientras que si el patrón de entrada no pertenece a esta clase y el Perceptrón lo asigna a ella, lo que hace es debilitar el peso restándole una cantidad proporcional al patrón de entrada . No modificaremos los pesos cuando el valor deseado coincida con la salida de la red. ahora bien, teniendo en cuenta que el sesgo se puede considerar como el peso sináptico correspondiente a un nuevo sensor de entrada que tiene siempre una entrada igual a xn+1=−1, y como peso sináptico el valor del umbral, pues w1x1 + w2 x2 +...+ wn xn ≥θ ⇔ 0 ... w1x1 + w2 x2 + + wn xn + wn+1xn+1 ≥ cuando wn+1=θ y xn+1= −1. Así, la red equivalente tendría n+1 sensores, su umbral sería siempre cero, los patrones de entrada ( n x , x ,..., x 1 2 ) serán ahora ( x1 , x2 ,..., xn ,−1), los pesos asociados (w w wn , ,..., 1 2 ) serán ( w1, w2, w3 , ,..., , wn, wn+1) con wn+1 =θ y la regla de aprendizaje:



α partir de ahora vamos a considerar el umbral como un peso sináptico más.

1. ENTRENAMIENTO DE UN PERCEPTRÓN

Acontinuaciòn se describirán los pasos fundamentales para el entrenamiento de un perceptrón:

Paso 0: Inicialización

Inicializar los pesos sinápticos con números aleatorios del intervalo [-1,1]. Ir al paso 1 con k=1

Paso 1: (k-ésima iteración)

Calcular



Paso 2: Corrección de los pesos sinápticos

Si z(k)≠y(k) modificar los pesos sinápticos según la expresión:



Paso 3: Parada

Si no se han modificado los pesos en las últimas p iteraciones, es decir,



parar. La red se ha estabilizado. En otro caso, ir al Paso 1 con k=k+1.

1. CONCLUSIONES

las redes neuronales son una idea simple pero potente, una vez entrenadas tiene una gran capacidad para realizar correctamente la tarea para la que fueron entrenadas, sin necesidad de traducir un complejo modelo del mundo real α un algoritmo, se puede hacer que ellas simulan la capacidad de aprender y reconocer patrones, como los seres humanos.

REFERENCIAS

1. <http://www.lcc.uma.es/~munozp/documentos/modelos_computacionales/temas/Tema4MC-05.pdf>
2. <https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf>