Perceptrón, su entrenamiento y el backpropagation

DAVID CAMILO LADINO BERNAL Y SEBASTIAN ALDANA SOLARTE.
UNIVERSIDAD TECNOLÓGICA DE PEREIRA. INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

Resumen

Se describen el modelo, la historia, las propiedades y el algoritmo de aprendizaje del Perceptrón para la resolución de problemas de clasificación, también se presentan ejemplos resueltos, así como las limitaciones de este tipo de redes.

Abstract

The model, the history, the properties and the learning algorithm of the Perceptron are described for solving classification problems, solved examples are also presented, as well as the limitations of this type of networks.

Keywords: Perceptrón, rule training.

1. Historia

En 1943, Warren McCulloch y Walter Bates introdujeron una de las primeras neuronas artificiales. La característica principal de su modelo de neuronas es que la suma ponderada de las señales de entrada se compara con un umbral para identificar las neuronas de salida. Cuando la suma es mayor o igual que el límite, la salida está en 1. Cuando la suma es menor que el límite, la salida está en 0.

A finales de la década de 1950, Frank Rosenblatt y otros investigadores desarrollaron una clase de redes neuronales llamadas perceptrones. Las neuronas de estas redes son similares a las de McCulloch y Bates.

Los perceptrones son limitados. Estas limitaciones fueron publicadas en el libro The Perceptrons de Marvin Minsky y Seymour Papert. Han demostrado que la red Perceptrón no puede realizar algunas funciones básicas. Fue solo en la década de 1980 que estas limitaciones fueron superadas por redes perceptivas avanzadas (de múltiples capas) combinadas con reglas de aprendizaje.

2. Introducción

El algoritmo Perceptrón fue publicado en 1957 por Frank Rosenblatt. El objetivo del Perceptrón es encontrar un hiperplano capaz de analizar con precisión un conjunto de datos separables linealmente que, una vez que se obtiene el hiperplano, se puede utilizar para la clasificación binaria. Aunque Perceptrón es un algoritmo de aprendizaje muy simple, entender cómo funciona es necesario para aprender otros métodos más complejos como los vectores virtuales o las redes neuronales artificiales.

3. Perceptrón Simple

El perceptrón simple es un modelo neuronal unidireccional, compuesto por dos capas de neuronas, una de entrada y otra de salida . La operación de una red de este tipo, con n neuronas de entrada y m neuronas de salida, se puede expresar de la siguiente forma:

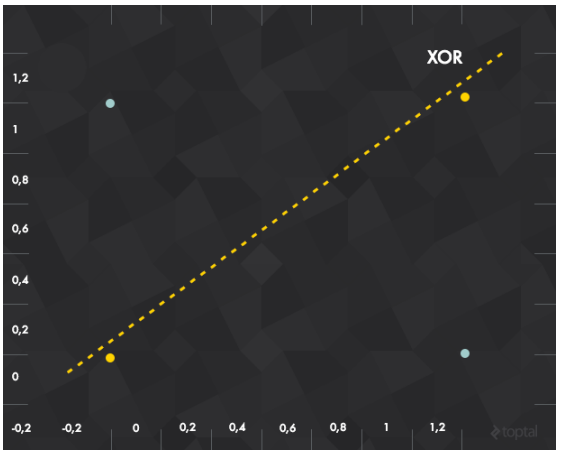
$$y_i = f(\sum_{j=1}^{n} w_{ij}x_j - \theta_i), \forall_i, 1 \leq i \leq m$$

Las neuronas de entrada no realizan ningún cómputo, únicamente envían la información (en principio consideraremos señales discretas {0, 1}) a las neuronas de salida. La función de activación de las neuronas de la capa de salida es de tipo escalón. Así, la operación de un perceptrón simple puede escribirse:

$$y_i(t) = H(\sum_{j=1}^{n} w_{ij}x_j - \theta_i), \forall_i, 1 \leq i \leq m$$

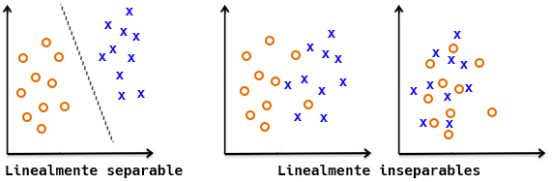
H: *función escalón de Heaviside*

El enfoque del perceptrón simple del aprendizaje profundo tiene un gran inconveniente: solo puede aprender funciones linealmente separables. ¿Qué importancia tiene este defecto? Tome XOR, que es una función relativamente simple, y notará que no se puede ordenar con un delimitador lineal (observe el intento fallido a continuación):



Por ende se tiene en consideración las siguientes dos condiciones:

1. - El conjunto de datos tiene que ser linealmente separable.



2. La tasa de aprendizaje tiene que ser suficientemente pequeña.

Si no se cumplen las dos condiciones anteriores, nuestro perceptrón seguirá intentando calcular los pesos correctos hasta el infinito, y más allá.

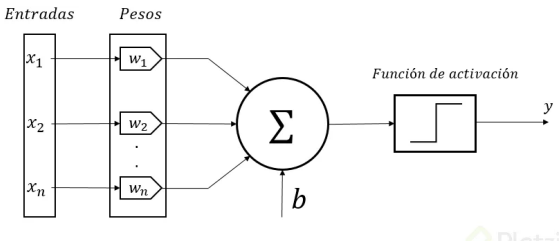
Para resolver este problema, tendremos que utilizar un perceptrón multicapa, también conocido como red neuronal feedforward: de hecho, combinaremos una

gran cantidad de estos perceptrones para crear un mecanismo de aprendizaje más sólido.

3.1 Entrenamiento

Por reglas de aprendizaje nos referimos a un procedimiento para cambiar los pesos y sesgos de una red (también conocido como algoritmo de aprendizaje). El propósito de una regla de aprendizaje es entrenar a la red para realizar una tarea. Existen varios tipos de reglas de aprendizaje para redes neuronales. Se divide en tres categorías: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado. El entrenamiento del Perceptrón no es más que determinar el peso sináptico y el umbral más adecuado con la entrada(Solo puede representar funciones lineales porque no tiene capas ocultas como la realización del Perceptrón multicapa). Para identificar estas variables, se sigue un proceso de adaptación. El proceso comienza con valores aleatorios y estos valores se modifican como función de la diferencia entre los valores deseados y los valores calculados por la red. En resumen, el perceptrón aprende de manera iterativa siguiendo estos pasos:

1. Inicializar pesos y umbrales
2. Bucle: hasta resultado de pesos sea aceptable
    - Bucle: para todos los ejemplos
    - Leer valores de entrada
    - Calcular error
    - Actualizar pesos según el error
        - Actualizar pesos de entradas
        - Actualizar el umbra



Considere el ejemplo anterior. Los perceptrones consisten en:

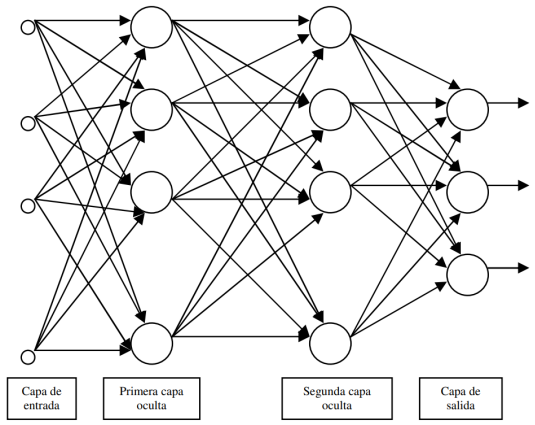
- **Entrada**: esta es la información que recibe el perceptrón. Se da un ejemplo en Learning Colors. Cuando un niño nota un color, sus ojos registran una imagen con propiedades específicas que le permiten identificar un color en particular. Cada una de estas propiedades es una entrada en el formulario.
- **Pesos**: Los valores numéricos responsables de determinar el efecto de la entrada en la salida deseada; Por ejemplo, para determinar si es probable que una persona sufra un ataque cardíaco, ingrese valores como obesidad, inactividad, diabetes, etc. Esta especificación puede indicar si una persona puede tener convulsiones, sin embargo, características como la obesidad y la diabetes aumentan el riesgo mucho más que la falta de ejercicio, por lo que es aún más importante a la hora de predecir un ataque cardíaco.
- **Bias**: este es un parámetro que tienen algunos modelos de redes neuronales, que le permite encontrar fácilmente la separación entre las capacidades de salida de la red neuronal.
- **Función de activación**: Es una función matemática que se encarga

de determinar el valor de la salida después de procesar cada entrada. En el aprendizaje de colores, se aplica al clasificar colores. En función de las características, el niño decide qué color imagina.

4. Perceptrón multicapa

La arquitectura del Perceptrón multicapa se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada, las capas ocultas y la capa de salida.

El Perceptrón multicapa define una relación entre las variables de entrada y las variables de salida de la red. Esta relación se obtiene propagando hacia adelante los valores de las variables de entrada. Para ello, cada neurona de la red procesa la información recibida por sus entradas y produce una respuesta o activación que se propaga, a través de las conexiones correspondientes, hacia las neuronas de la siguiente capa.



4.1 Entrenamiento (Perceptrón multicapa)

- **Lote**: Actualiza los pesos de las entradas solo después de que se hayan transferido todos los registros de datos de entrenamiento; Es decir, el aprendizaje en grupo utiliza información de todos los registros del conjunto de datos de entrenamiento. A menudo se prefiere el entrenamiento por lotes porque reduce directamente el error general; Sin embargo, el aprendizaje por lotes puede obligar a que los pesos se actualizan varias veces hasta que se cumpla una de las reglas de apagado y, por lo tanto, puede requerir varias lecturas de datos. Esto es útil para conjuntos de datos "más pequeños".
- **En línea:** Actualice los pesos después de cada registro de datos de entrenamiento; Es decir, la capacitación en línea solo usa información de un registro a la vez. Los entrenadores en línea reciben actualizaciones constantes de los registros y el peso hasta que se cumple la regla de parada. Si todos los registros se utilizan una vez y no se siguen las reglas de cierre, el proceso continuará reciclando los registros de datos. El entrenamiento en línea es superior al entrenamiento por lotes para conjuntos de datos “más grandes” con predictores correlacionados; Es decir, si hay muchos registros y muchas entradas, y sus valores no son independientes entre sí, la

capacitación en línea puede obtener una respuesta razonablemente más rápida que la capacitación por lotes.

- **Por mini lotes:** Divide los registros de datos de entrenamiento en grupos de tamaño parecido y actualiza las ponderaciones sinápticas tras pasar un grupo; es decir, el entrenamiento por mini lotes utiliza la información de un grupo de registros. A continuación, el proceso recicla el grupo de datos si es necesario. El entrenamiento por mini lotes ofrece una solución intermedia entre el entrenamiento por lotes y en línea, y puede ser el idóneo para conjuntos de datos de "tamaño medio". El procedimiento puede determinar automáticamente el número de registros de entrenamiento por mini lote, o bien puede especificar un entero mayor que 1 y menor o igual que el número máximo de casos para almacenar en memoria. Puede establecer el número máximo de casos a almacenar en memoria en la pestaña Opciones.

4.2 Algoritmo de optimización

- **Gradiente conjugado escalado**: Los supuestos que justifican el uso de métodos de gradiente conjugado se aplican únicamente a los tipos de entrenamiento por lotes, de modo que este método no se encuentra disponible para el entrenamiento en línea o por mini lotes.
- **Pendiente de gradiente**: Este método debe utilizarse con el entrenamiento en línea o por mini lotes; también puede utilizarse con el entrenamiento por lotes.

4.3 Opciones de entrenamiento

Las opciones de entrenamiento le permiten ajustar el algoritmo de optimización. Generalmente no tendrá que cambiar estos ajustes a menos que la red experimente problemas con la estimación.

- **Lambda inicial:** El valor inicial del parámetro lambda para el algoritmo de gradiente conjugado escalado. Especifique un número mayor que 0 y menor que 0,000001.
- **Sigma inicial:** El valor inicial del parámetro sigma para el algoritmo de gradiente conjugado escalado. Especifique un número mayor que 0 y menor que 0,0001.
- **Centro de intervalo y desplazamiento de intervalo**: El centro del intervalo (a_0) y el desplazamiento de intervalo (a) definen el intervalo [$a_0 - a$, $a_0 + a$], en el que se generan aleatoriamente vectores de ponderación cuando se utiliza el recocido simulado. El algoritmo de recocido simulado se utiliza para superar un mínimo local, con el objetivo de encontrar el mínimo global, durante la aplicación del algoritmo de optimización. Este enfoque se utiliza en la

inicialización de ponderaciones y la selección automática de arquitectura. Especifique un número para el centro de intervalo y un número mayor que 0 para el desplazamiento de intervalo.

5. Backpropagation

El algoritmo de backpropagation se introdujo originalmente en la década de 1970, pero su importancia no se apreció por completo hasta después de un artículo popular de 1986 de David Rummelhart, Jeffrey Hinton y Ronald Williams. Este artículo describe varias redes neuronales en las que la propagación inversa funciona mucho más rápido que los métodos de aprendizaje anteriores, lo que permite que la red neuronal se utilice para resolver problemas difíciles en primer lugar. Hoy en día, los algoritmos de backpropagation son herramientas para aprender redes neuronales.

La señal de error de la neurona j en el instante n se define por

$e_j(n) = d_j(n) - y_j(n)$, siendo la neurona j el nodo de salida (1.1)

Definimos el valor instantáneo del error cuadrático para la neurona j como $\frac{1}{2}e_j^2(n)$. De esta forma, el valor instantáneo $\xi(n)$ de la suma de errores cuadráticos es obtenido sumando $\frac{1}{2}e_j^2(n)$ para todas las neuronas de la capa de salida; éstas son las únicas neuronas para las que las señales de error pueden ser calculadas. La suma instantánea de errores cuadráticos de la red es escrita, por tanto, como

$$\xi(n) = \frac{1}{2}\sum_{j\epsilon C} e_j^2(n) \quad (1.2)$$

donde el conjunto C incluye a todas las neuronas de la capa de salida de la red. Denotamos por N al número total de patrones (ejemplos) contenidos en el paquete de entrenamiento. El error cuadrático medio se obtiene sumando (n) para todo n y luego normalizando con respecto al paquete de tamaño N, como mostramos a continuación:

$$\xi_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} \xi(n) \quad (1.3)$$

El algoritmo backpropagation aplica una corrección $\Delta w_{ji}(n)$ al peso sináptico $w_{ji}(n)$ que es proporcional al gradiente instantáneo. Según la regla de la cadena, podemos expresar este gradiente como sigue:

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \frac{\partial \xi(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad (1.4)$$

El gradiente $\frac{\partial \xi(n)}{\partial w_{ji}(n)}$ representa un factor que determina la dirección de búsqueda en el espacio de pesos para el peso sináptico w_{ji}. Derivando ambos lados de la ecuación (1.2) respecto a $e_j(n)$, obtenemos

$$\frac{\partial \xi(n)}{\partial e_j(n)} = e_j(n) \quad (1.5)$$

Derivando ahora a ambos lados de (1.1) respecto a $y_j(n)$, obtenemos

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \qquad (1.6)$$

Ahora, derivando la ecuación respecto a $v_j(n)$, obtenemos

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \delta_j'(v_j(n)) \qquad (1.7)$$

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \qquad (1.8)$$

A continuación, usamos la regla de la cadena para la derivada parcial $\frac{\partial \xi(n)}{\partial y_j(n)}$, y así
podemos escribir la ecuación (1.8) de forma equivalente como

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}$$

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento.

6. Bibliografía

- https://koldopina.com/como-entrenar-a-tu-perceptron/
- http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas/transparencias/Tema2%20PerceptronAdalineRN.pdf
- https://www.youtube.com/watch?v=OWgAtvadHGU
- https://platzi.com/contributions/entrenamiento-del-perceptron/
- https://www.toptal.com/machine-learning/un-tutorial-de-aprendizaje-profundo-de-perceptrones-a-redes-profundas
- https://www.youtube.com/watch?v=vr6dz5Avdqk
- https://www.uaeh.edu.mx/docencia/P_Presentaciones/huejutla/sistemas/redes_neuronales/perceptron.pdf

- http://bibing.us.es/proyectos/abrepr oy/11084/fichero/Memoria+por+ca pítulos+%252FCapítulo+5.pdf+