Perceptrón & Backpropagation

Perceptron & Backpropagation

Autor: Carlos Miguel Rodriguez Botero, Dorian Felipe Marín Quintero.
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: carlos.rodriguez1@utp.edu.co, dorian.marin@utp.edu.co

***Resumen*— En este documento hablaremos y conoceremos el algoritmo del perceptrón ya que como lo veremos a continuación es un algoritmo que nos permite observar patrones con redes neuronales por la cual consta de entradas, un umbral y unas salidas y son patrones que se encuentran en ambos lados de un hiperplano. Por ello vamos a conocer mucho más a fondo este algoritmo tan útil y a la vez tan necesario para la programación y la inteligencia artificial, por la cual se basa este algoritmo para su implementación. Muy importante saber su entrenamiento y el backpropagation por la cual funciona más rápido y es la descripción de varias redes neuronales y esto hace posible la resolución de problemas y es muy importante para el aprendizaje en redes neuronales.**

Palabras clave*— *perceptrón, algoritmo, patrones, redes, neurona, hiperplano, programación, inteligencia artificial, implementación, conocimiento, problemas.

***Abstract*— In this document we will talk and get to know the perceptron algorithm since, as we will see below, it is an algorithm that allows us to observe patterns with neural networks by which it consists of inputs, a threshold and outputs and they are patterns that are found on both sides of a hyperplane. For this reason, we are going to learn much more about this algorithm, so useful and once so necessary for programming and artificial intelligence, on which this algorithm is based for its implementation. Very important to know its training and the backpropagation by which it works faster and is the description of several neural networks and this makes problem solving possible and is very important for learning in neural networks.**

***Key Word*— perceptron, algorithm, patterns, networks, neuron, hyperplane, programming, artificial intelligence, implementation, knowledge, problems.**

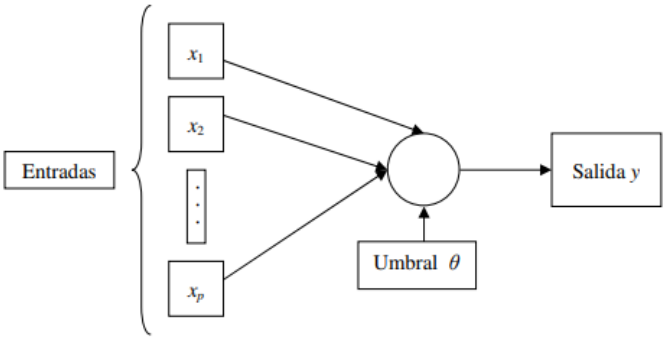
I. INTRODUCCIÓN

El perceptrón 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). 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.

Por otro lado, el backpropagation es un algoritmo que se introdujo originalmente en la década de 1970, pero no tomó mayor relevancia sino hasta que se publicó un famoso artículo en 1986 por David Rumelhart, Geoffrey Histon y Ronald Williams. En dicha publicación se describen varias redes neuronales en las que la retropropagación sirve de una forma más eficaz en comparación a estudios que se habían realizado anteriormente. Se podría decir que el algoritmo de backpropagation es el pilar del aprendizaje en redes neuronales.

I.1 Perceptrón

En la siguiente imagen se muestra el ejemplo de un perceptrón por la cual es representada por una neurona.



El perceptrón de una capa 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.

Cuando los elementos de diferentes categorías pueden ser separados por una sola línea sin que estos elementos se mezclen entre sí. El perceptrón puede ser visto como una red neuronal artificial de una sola neurona (por eso se dice que el perceptrón es unicapa).

A continuación, te presentamos la separación lineal del grupo azul y del grupo rojo, como se dijo arriba, el perceptrón solamente puede hacer esta división, por ello no es tan exacto, pues aún cabe la posibilidad de que elementos de un grupo crucen la línea del grupo contrario.

Clasificación a partir de los datos.

El perceptrón de una sola capa es una red neuronal artificial con limitaciones, pues solo puede clasificar de manera lineal, lo que causa que en algunos casos la separación por categorías no sea precisa.

Entrenamiento:

El entrenamiento del perceptrón no es más que determinar los pesos sinápticos y el umbral que mejor hagan que la entrada se ajuste a la salida. Para la determinación de estas variables, se sigue un proceso adaptativo. El proceso comienza con valores aleatorios y se van modificando estos valores según la diferencia entre los valores deseados y los calculados por la red.

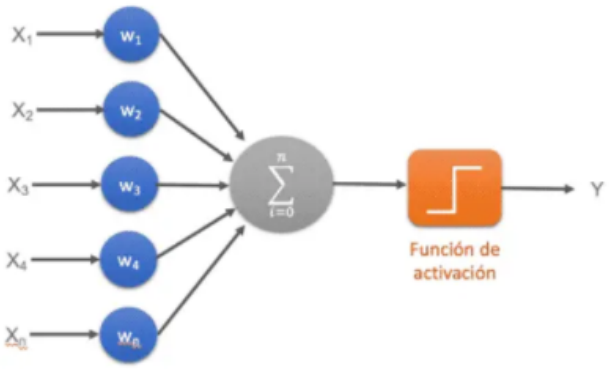
En resumen, el perceptrón aprende de manera iterativa siguiendo estos pasos:

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 umbral

Arquitectura



Este es un perceptrón simple como en el ejemplo anterior

Conjunto de entradas x1,…xn

- Representan las entradas de la red neuronal.

Pesos sinápticos w1,…wn

- Cada entrada tiene un peso que se va ajustando de forma automática a medida que la red neuronal va aprendiendo.

Función de agregación, Σ

- Realiza el sumatorio de todas las entradas ponderadas por sus pesos.

Función de activación, F

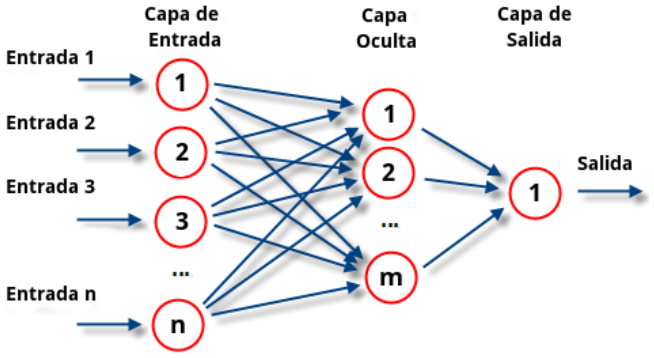
- Se encarga de mantener el conjunto de valores de salida en un rango determinado, normalmente (0,1) o (-1,1)
- Existen diferentes funciones de activación que cumplen este objetivo, la más habitual es la función sigmoide.

Salida, Y

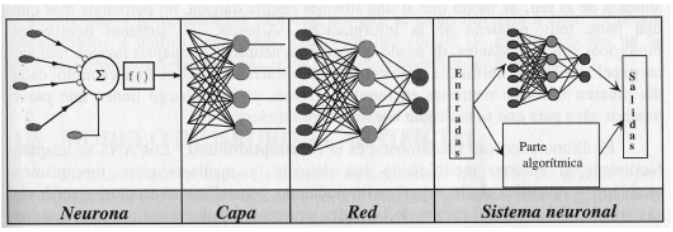
- Representa el valor resultante tras pasar por la red neuronal.

En la siguiente figura se muestra otro ejemplo de perceptrón con unas entradas, un umbral y unos pesos con unas salidas.

Pero en esta figura se muestra es un perceptrón con multicapa.

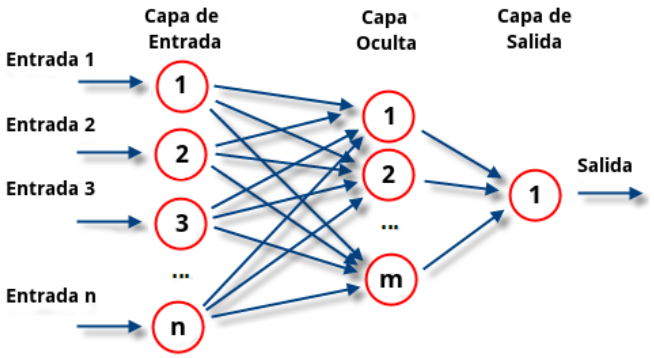


En la siguiente figura se muestra el sistema global de procesos de una red neuronal.



Perceptrón multicapa

El perceptrón multicapa es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple). El perceptrón multicapa puede estar totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa "i" es entrada de todas las neuronas de la capa "i+1", mientras que en el segundo cada neurona de la capa "i" es entrada de una serie de neuronas (región) de la capa "i+1".

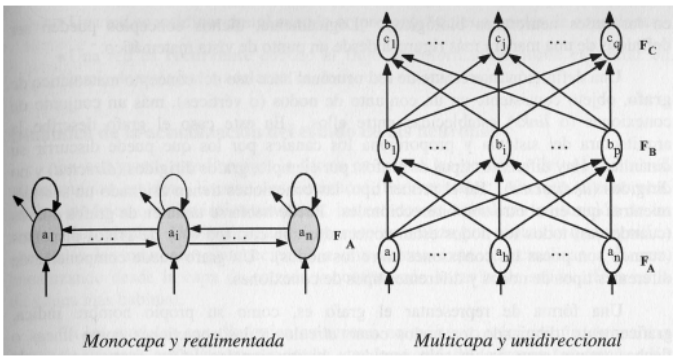


Las capas pueden clasificarse en tres tipos:

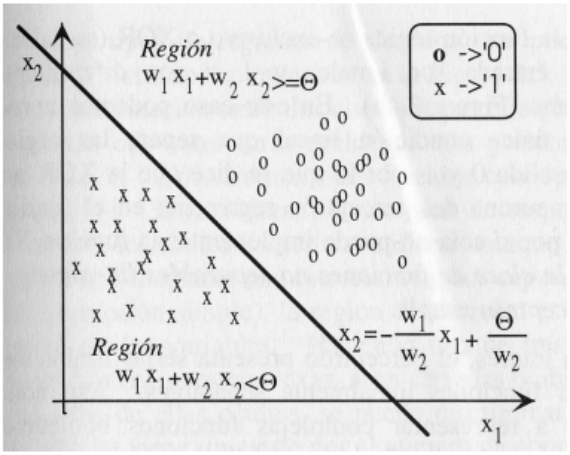
- Capa de **entrada**: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas **ocultas**: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- Capa **de salida**: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red

Definición de una red neuronal artificial

A continuación en la siguiente figura se muestran las diferentes arquitecturas de redes neuronales.



El perceptrón simple presenta grandes limitaciones, ya que tan sólo es capaz de representar funciones linealmente separables. Basándose en este hecho, Minsky y Papert (1969) publicaron un trabajo exponiendo las limitaciones del perceptrón simple, como consecuencia del cual muchos de los recursos que se venían dedicando a las redes neuronales se desviaron a otros campos de la inteligencia artificial.



En la anterior figura se muestra la región de decisión correspondiente a un perceptrón simple o de una sola capa con dos neuronas de entrada.

Arquitectura	Región de decisión	Ejemplo 1: XOR	Ejemplo 2: clasificación	Regiones más generales
Sin capa oculta	Hiperplano (dos regiones)			
Una capa oculta	Regiones polinomiales convexas			
Dos capas ocultas	Regiones arbitrarias			

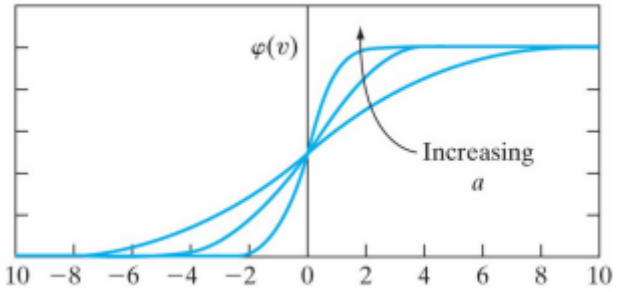
En la figura anterior tenemos regiones de decisión obtenidas para el perceptrón simple (arriba), el perceptrón multicapa con una capa oculta (en medio) y el perceptrón multicapa con dos capas ocultas (abajo).

I.2 Backpropagation

Dos elementos a tener en cuenta en esté algoritmo son los pesos y los sesgos para calcular los costes. Esto se puede encontrar haciendo uso de una expresión para la derivada parcial $\frac{\delta C}{\delta w}$ de la función de costes C con respecto a cualquier peso (w) en la red. Algo supremamente interesante al hacer uso de esta expresión es que podemos observar la forma en que nos brinda información detallada sobre cómo al cambiar los pesos y los sesgos se cambia el comportamiento general de la red.

Es utilizado para entrenar redes neuronales multicapa. Un aspecto a tener en cuenta es que exige que la función de activación de las neuronas sea derivable y creciente. Normalmente se usan:

- Sigmoide



- Logística

$$\varphi(v) = \frac{1}{1+e^{-av}}$$

- Tangente Hiperbólica

$$\varphi(v) = \tanh v = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}}$$

El objetivo principal de backpropagation es de minimizar el error cuyo dominio es el espacio de los pesos de las conexiones. Esté es descrito por la siguiente ecuación:

$$E_p = 1/2 \sum_k (y_{pk} - d_{pk})^2$$

[3]

¿Cómo funciona el algoritmo?

- Presenta un patrón de entrada a la red.
- Propagada dichas entradas hasta la capa de salida..
- Calcula el error de la capa de salida.
- Propaga dicho error hacia las neuronas ocultas, es decir, hacía atrás.
- Cambia los pesos de las conexiones.
- Repite el ciclo N iteraciones para reducir el error cuadrático medio del conjunto entero y de esta forma llegar a la solución más óptima.
- Se da fin a las iteraciones cuando se llega al error mínimo o al cumplir las iteraciones previamente establecidas.

Pasos del Algoritmo:

1. Inicializar aleatoriamente los pesos.
2. Escoger aleatoriamente un patrón de entrada X.
3. Propagar la señal hacía adelante
4. Calcular el error de la capa de salida con la siguiente ecuación.

$$\delta_i(t) = f'(x_i(t)) * (d_i - y_i)$$

5. Propagar dicho error hacia las neuronas ocultas (de derecha a izquierda), con la siguiente ecuación.

$$\delta_j(t) = f'(x_i(t)) * (\sum_i w_{ij} \delta_i(t))$$

6. Actualizar los pesos, con la siguiente ecuación

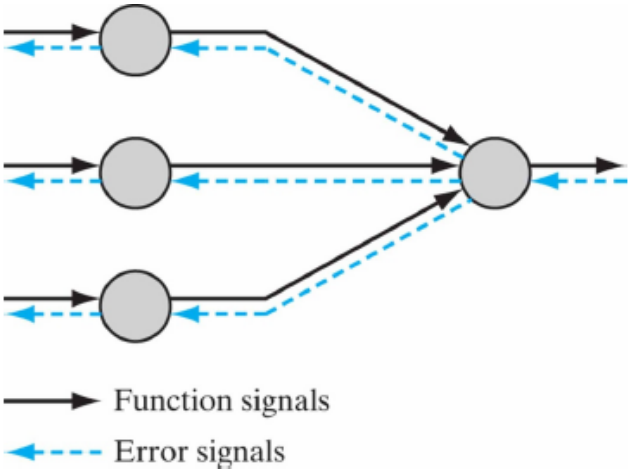
$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t+1)$$

donde,

$$\Delta w_{ji}(t+1) = [\eta \delta_j y_i(t) + \alpha \Delta w_{ji}(t)]$$

Tomar otro patrón de entrada [paso 2] y hacer las mismas iteraciones una y otra vez hasta alcanzar el error deseado.

Un punto supremamente importante es que a medida que la red neuronal se entrena, esta va tomando características del espacio total de la entrada, lo que implica que cuando reciba un patrón que se asemeje a las características anteriormente mencionadas, pues la neurona responderá con una salida activa, de lo contrario, al no encontrar características anteriormente obtenidas, descarta ese patrón para la salida.



REFERENCIAS

Perceptron

http://bibing.us.es/proyectos/abreproy/11084/fichero/Memoria+por+cap%C3%ADtulos+%252FCap%C3%ADtulo+4.pdf+

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

https://www.diegocalvo.es/perceptron/

https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa

http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

https://empresas.blogthinkbig.com/como-funciona-el-algoritmo-backpropagation-en-una-red-neuronal/

Backpropagation

https://elvex.ugr.es/decsai/computational-intelligence/slides/N2%20Backpropagation.pdf

https://www.youtube.com/watch?v=boP3O89rErA

https://disi.unal.edu.co/~lctorress/RedNeu/RNA005c.pdf

http://medicinaycomplejidad.org/pdf/redes/Backpropagation.pdf