PERCEPTRON Y BACKPROPAGATION

Perceptron and Backpropagation

Autores: Elsa Carolina Peláez Sampredo, Evelyn Rodriguez Cardona, Jaime Andres Valencia Gaviria

***Resumen*— En este paper se abordará el tema red de aprendizaje supervisado backpropagation. entenderemos qué es y cómo se comporta la unidad fundamental de la red neuronal perceptrón.**

Las Redes Neuronales Artificiales son las que actualmente están causando un mayor impacto, debido a su extraordinaria aplicación práctica, ya que son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. La evolución de la investigación en redes neuronales desde los años 50 a nuestros días ha estado condicionada por dos grandes acontecimientos: el abandono de esta línea de investigación en la segunda mitad de los 60 debido a las limitaciones observadas en la red Perceptrón simple y la aparición del algoritmo, denominado backpropagation error (propagación del error hacia atrás) o simplemente backpropagation, que permite modificar las conexiones de arquitecturas multiestrato.

***Palabras clave*— perceptrón, neurona, función de activación, retropropagación.**

***Abstract*— In this paper we are going to talk about the backpropagation type of supervised learning network. We will understand what it is and how the fundamental unit of the perceptron neural network behaves.**

Artificial Neural Networks are the ones that are currently causing a greater impact, due to their extraordinary practical application, since they are a model to find that combination of parameters and apply it at the same time. The evolution of neural network research from the 1950s to the present day has been conditioned by two major events: the abandonment of this line of research in the second half of the 1960s due to the limitations observed in the simple perceptron network, and the appearance of the algorithm known as backpropagation error (error propagation backwards) or simply backpropagation, which makes it possible to modify the connections of multi-layer architectures.

***Key Word*— perceptron, neuron, activation function, backpropagation.**

INTRODUCCIÓN

El concepto de redes neuronales suele ser explicado fácilmente cuando se relaciona con el aprendizaje humano. Cuando un niño va a aprender a identificar los colores, primero debe mirar el color y luego con indicaciones, aprende las características de ese color, esto le permite identificar cada uno de ellos en el futuro sin tener que recibir nuevamente la información. Esto mismo sucede con las redes neuronales, estas pueden tomar decisiones una vez han sido entrenadas con información previa. Si se desea que identifique colores como se mencionaba anteriormente, se debe suministrar información que permita identificar cada color tal cual como lo haría un niño en su aprendizaje.

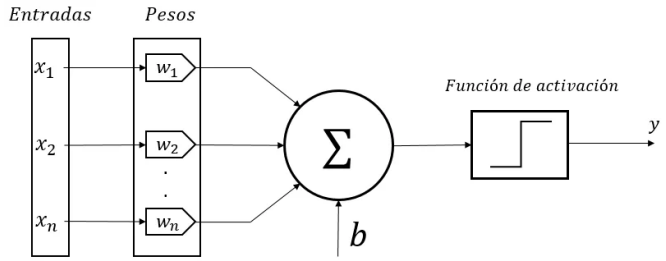
Para comprender mejor el funcionamiento de una red neuronal, este artículo se enfoca en el análisis del perceptrón puesto que es uno de los modelos más simples de una red neuronal.

PERCEPTRON

Un perceptrón es un algoritmo usado para el aprendizaje supervisado de clasificadores binarios. Los clasificadores binarios son los que deciden si una entrada, generalmente representada por una serie de vectores, pertenece a una clase en específico.

Un perceptrón es una red neuronal de una sola capa. Consiste en cuatro partes principales, que incluyen valores de entrada, pesos y tendencias, suma ponderada y una función de activación.

El objetivo final de un perceptrón es identificar las entradas que intervienen en él. Para identificar si las características son verdaderas o falsas.



Componentes principales del perceptrón

- Entrada: Se entienden como x1, x2. x3, …, xn. Todas estas entradas denotan valores del perceptrón de características y la ocurrencia total de las características.

- Pesos: Valores que se planifican a lo largo de la sesión de preparación del perceptrón. Los pesos ofrecen un valor preliminar en el inicio del aprendizaje del algoritmo. Con la ocurrencia de cada inexactitud de entrenamiento, los valores de los pesos se actualizan. Estos se representan principalmente como w1, w2, w3, … wn.

- Suma ponderada: Es la proliferación de cada valor de entrada o característica asociada con el valor de paso correspondiente.

- Función de activación: Esta función toma un único número y realiza una determinada operación matemática fija sobre él. Existen varias funciones de activación, la más común es la Sigmoide.
- Salida: Al pasarse la suma ponderada a la función de activación, se obtiene un valor después de dicho cálculo, obteniendo así como resultado la salida.

ENTRENAMIENTO

La regla de aprendizaje del perceptrón establece que el algoritmo aprende automáticamente los coeficientes de peso óptimos. Luego, las características de entrada se multiplican con estos pesos para determinar si una neurona se activa o no.

El perceptrón recibe múltiples señales de entrada, y si la suma de las señales de entrada excede un cierto umbral, emite una señal o no devuelve una salida. En el contexto de aprendizaje supervisado y la clasificación, esto puede ser utilizado para predecir la clase de una muestra.

Algoritmo de aprendizaje:

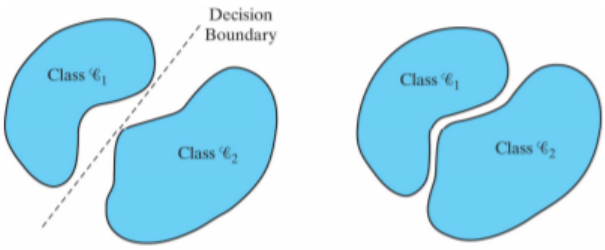
- Si la salida es correcta, los pesos quedan tal cual.

- Si la unidad de salida incorrectamente a cero, se añade el vector de entrada al vector de pesos.

- Si la unidad de salida incorrecta da uno, se resta el valor de entrada del vector de pesos.

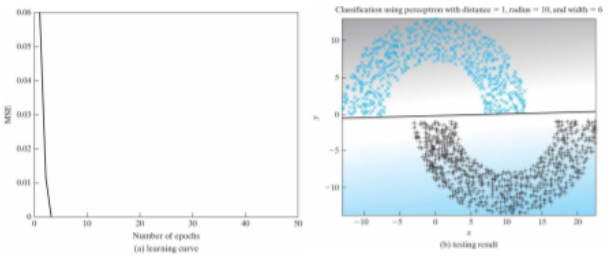
El algoritmo de aprendizaje del perceptrón garantiza encontrar un conjunto de pesos que proporcione la respuesta correcta si tal conjunto existe.

El perceptrón es un modelo de clasificación lineal, por tanto será capaz de clasificar correctamente los ejemplos de entrada siempre que las clases sean linealmente separables.

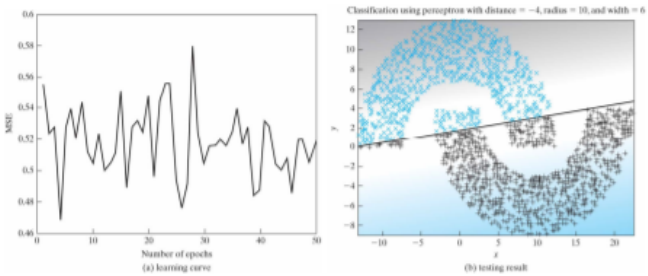


Clases linealmente separables

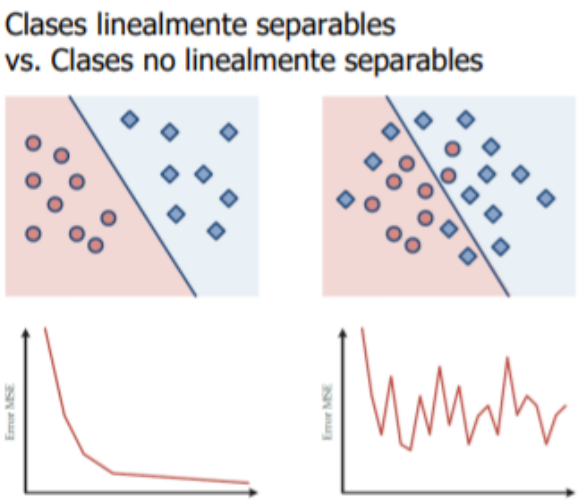
Clases linealmente no separables



Clases linealmente separables



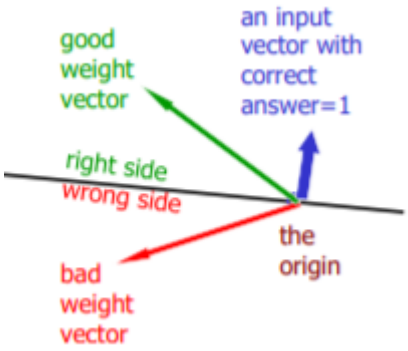
Clases linealmente no separables



Interpretación geométrica del algoritmo de aprendizaje:

Espacios de pesos:

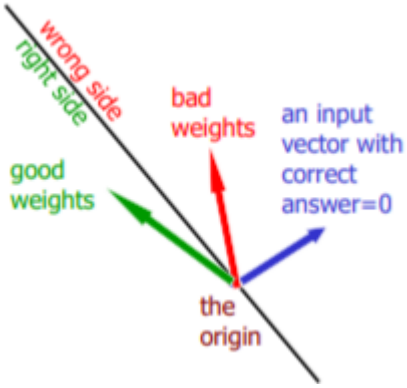
- Una dimensión por cada peso.
- Cada punto en el espacio representa un valor particular para el conjunto de pesos.
- Cada caso de entrenamiento corresponde a un hiperplano que pasa por el origen (tras eliminar el umbral e incluirlo como un peso más).



Cada caso define un hiperplano:

- Hiperplano perpendicular al vector de entrada
- Los pesos deben quedar a un lado del hiperplano.

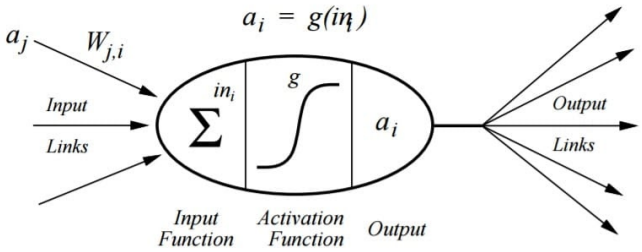
Si el producto escalar del vector de entrada con el vector de pesos es negativo, la salida será la equivocada.



FUNCIONES DE ACTIVACIÓN

Sigmoide

El diagrama a continuación nos muestra un perceptrón con un sigmoide como función de activación.



$$a_i = g(\sum_j W_{j,i} a_j)$$

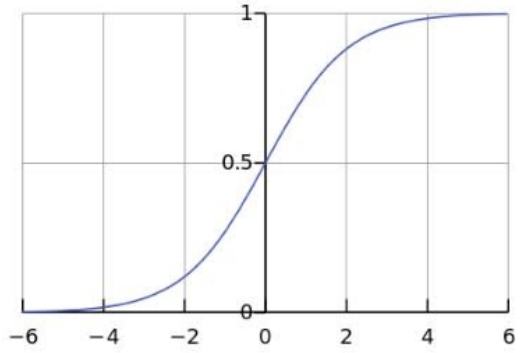
Una función sigmoide es una función matemática con una curva sigmoide (Curva “S”). Es un caso especial de esta función lógica y está definido por la siguiente función:

$$\phi_{logistic}(z) = \frac{1}{1+e^{-z}}$$

Aquí, el valor de z es:

$$z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x$$

- Curva sigmoide



Esta tiene un valor de probabilidad entre 0 y 1.
Es útil como función de activación cuando se está interesado en el mapeo de probabilidad en lugar de valores precisos del parámetro de entrada t.

La salida sigmoide es cercana a cero para una entrada muy negativa. Esto puede ser un problema en el entrenamiento de redes neuronales y puede conducir a un aprendizaje lento y ocasionar que el modelo se quede estancado durante el entrenamiento.
Por tanto, la tangente hiperbólica es más preferible como una función de activación en capas ocultas de una red neuronal.

Lógica sigmoidea para datos de muestra:

```
>>> import numpy as np

>>> X = np.array([1, 1.4, 2.5]) ## first value must be 1
>>> w = np.array([0.4, 0.3, 0.5])

>>> def net_input(X, w):
...     return np.dot(X, w)
...
>>> def logistic(z):
...     return 1.0 / (1.0 + np.exp(-z))
...
>>> def logistic_activation(X, w):
...     z = net_input(X, w)
...     return logistic(z)
...
>>> print('P(y=1|x) = %.3f' % logistic_activation(X, w))
P(y=1|x) = 0.888
```

Salida:

La salida del perceptrón es 0.888, lo que indica que la probabilidad de salida para *y* sea 1.

Si la salida del sigmoide es un valor mayor a 0.5, la salida es tomada como verdadera. Como en este caso 0.888 es mayor a 0.5, la salida es verdadera.

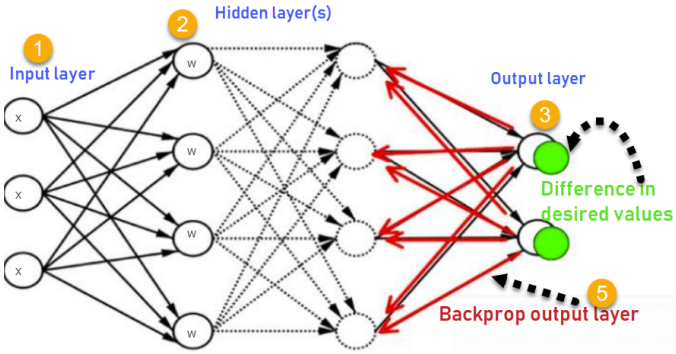
BACKPROPAGATION

Es probablemente el bloque de construcción más fundamental en una red neuronal. Fue introducido por primera vez en la década de 1960 por

La BackPropagation es un tipo de red de aprendizaje supervisado, que emplea un ciclo de propagación. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo, las neuronas de la capa oculta sólo reciben una fracción de la señal total del error basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error

que describa su contribución relativa al error total. Basándose en la señal de error percibida, donde se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.



I.5 DESCENSO PRONUNCIADO

Este es un proceso para encontrar el punto mínimo de una función de forma iterativa, por lo que se requieren de múltiples pasos para llegar a un punto de convergencia.

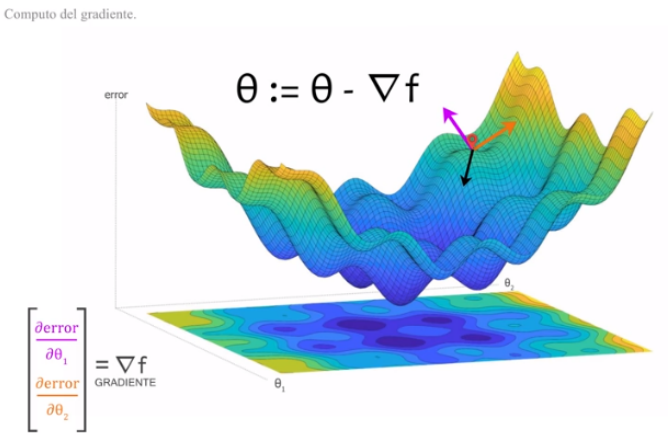
Este método es utilizado en funciones

$$f : R^n -> R$$

de múltiples variables de las cuales es muy difícil analíticamente encontrar su mínimo.

La idea detrás de este método es modificar las variables en dirección en donde se encuentre su mínimo, con pasos óptimos, ni muy grandes para saltarse el mínimo, ni muy pequeños para que no sea muy costoso en tiempo y procesamiento.

Como se dijo anteriormente el proceso de Backpropagation es la forma en la cual las Redes Neuronales aprenden, este proceso está encargado de modificar los pesos y los términos bias de la Red Neuronal al minimizar la función de error de las predicciones de la Red, a dicha función de error se le conoce como función de costo y esta puede variar según sea el tipo de problema que resuelva la Red, como por ejemplo problemas de regresión o clasificación.



I.6 GRADIENTES DE BACKPROPAGATION

El nivel de ajuste está determinado por los gradientes de la función de costo con respecto a esos parámetros. ¿por qué calcular gradientes ?

- El gradiente de una función C (x_1, x_2,…, x_m) en el punto x es un vector de las derivadas parciales de C en x.

$$\frac{\partial C}{\partial x} = [\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}, \ldots, \frac{\partial C}{\partial x_m}]$$

Ecuación para la derivada de C en x

- La derivada de una función C mide la sensibilidad al cambio del valor de la función (valor de salida) con respecto a un cambio en su argumento x (valor de entrada) . En otras palabras, la derivada nos dice la dirección en la que va C
- El gradiente muestra cuánto debe cambiar el parámetro x (en dirección positiva o negativa) para minimizar C.
  Calcular esos gradientes ocurre usando una técnica llamada regla de cadena .
  Para un solo peso (w_jk) ^ l, el gradiente es:

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \qquad chain\ rule$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l \qquad by\ definition$$

$$m\ -\ number\ of\ neurons\ in\ l-1\ layer$$

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \qquad by\ differentiation\ (calculating\ derivati$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}a_k^{l-1} \qquad final\ value$$

Ecuaciones para la derivada de C en un solo peso (w_jk) ^ l

Se puede aplicar un conjunto similar de ecuaciones a (b_j) ^ l:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \qquad chain\ rule$$

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \qquad by\ differentiation\ (calculating\ derivativ$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}1 \qquad final\ value$$

Ecuaciones para la derivada de C en un solo sesgo (b_j) ^ l

La parte común en ambas ecuaciones a menudo se denomina "gradiente local" y se expresa de la siguiente manera:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \qquad local\ gradient$$

Ecuación para gradiente local

El "gradiente local" se puede determinar fácilmente usando la regla de la cadena.
Los gradientes nos permiten optimizar los parámetros del modelo:

$$while\ (termination\ condition\ not\ met)$$

$$w := w - \epsilon\frac{\partial C}{\partial w}$$

$$b := b - \epsilon\frac{\partial C}{\partial b}$$

$$end$$

Algoritmo para optimizar pesos y sesgos (también llamado "descenso de gradiente")

- Los valores iniciales de w y b se eligen al azar.
- Epsilon (e) es la tasa de aprendizaje . Determina la influencia del gradiente.
- w y b son representaciones matriciales de los pesos y sesgos. La derivada de C en w o b se puede calcular usando derivadas parciales de C en los pesos o sesgos individuales.
- La condición de terminación se cumple una vez que se minimiza la función de costo.

REFERENCIAS

[1] https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

[2] https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/

[3] https://elvex.ugr.es/decsai/deep-learning/slides/NN2%20Perceptron.pdf

[4] Understanding Backpropagation Algorithm https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

Videos recomendados:

[5] ¿Qué es una Red Neuronal? Parte 1 : La Neurona https://www.youtube.com/watch?v=MRIv2IwFTPg

[6] ¿Qué es una Red Neuronal? Parte 2 : La Red https://www.youtube.com/watch?v=uwbHOpp9xkc

[7] ¿Qué es una Red Neuronal? Parte 3 : Backpropagation https://www.youtube.com/watch?v=eNIqz_noix8&list=LL&index=2&t=133s