Entrenamiento Del Perceptrón

Red Neuronal

# Perceptrón Training Neural Network

Autor 1: Julian Osorio Salazar

*Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: julianos@utp.edu.co

***Resumen*—** Una de las características más significativas de las redes neuronales es su capacidad para aprender a partir de alguna fuente de información interactuando con su entorno. En 1958 el psicólogo Frank Ronsenblant desarrolló un modelo simple de neurona basado en el modelo de McCulloch y Pitts que crearon en el año 1943 y en una regla de aprendizaje basada en la corrección del error. A este modelo le llamó Perceptrón. Una de las características que más interés despertó de este modelo fue su capacidad de aprender a reconocer patrones. El Perceptrón puede ser visto como una neurona artificial y está constituido por un conjunto de sensores de entrada que reciben los patrones de entrada a reconocer o clasificar y es un algoritmo capaz de generar un criterio para hacer la selección de un grupo, entre un grupo de elementos más grandes. Por si solo un perceptrón no tiene mucha utilizad, puesto que es como una neurona. Su capacidad y funcionalidad se genera cuando se relacionan con más para generar una red. dejó de ser considerado como una técnica útil, al igual que otros tipos de redes neuronales, cuando Marvin Minsky publicó un artículo en el cual expresaba las limitantes de estas técnicas.

***Palabras clave—***Aprendizaje, Patrones, Entrada, Salida, Inteligencia Artificial, Acción, Reacción, Incertidumbre, Verdadero, Falso, Red Neuronal, Perceptrón, Neurona, Grupos, Entrenamiento, Clasificación.

***Abstract*—** One of the most significant characteristics of neural networks is their ability to learn from some source of information interacting with their environment. In 1958 the psychologist Frank Ronsenblant developed a simple neuron model based on the model of McCulloch and Pitts that they created in 1943 and on a learning rule based on error correction. This model was called Perceptron. One of the characteristics that most aroused this model was its ability to learn to recognize patterns. The Perceptron can be seen as an artificial neuron and is constituted by a set of input sensors that receive the input patterns to be recognized or classified and is an algorithm capable of generating a criterion to make the selection of a group, among a group of Larger elements. In itself only one perceptron does not have much use, since it is like a neuron. Its capacity and functionality are generated when they are related to more to generate a network. it ceased to be considered a useful technique, like other types of neural networks, when Marvin Minsky published an article in which he expressed the limitations of these techniques.

***Key Word* —** Learning, Patterns, Input, Exit, Artificial Intelligence, Action, Reaction, Uncertainty, True, False, Neural Network, Perceptron, Neuron, Groups, Training, Classification

1. INTRODUCCIÓN

La red tipo Perceptrón fue inventada por el psicólogo Frank Rosenblatt en el año 1957. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general, sin entrar en mayores detalles con respecto a condiciones específicas y desconocidas para organismos biológicos concretos. Rosenblatt creía que la conectividad existente en las redes biológicas tiene un elevado porcentaje de aleatoriedad, por lo que se oponía al análisis de McCulloch Pitts en el cual se empleaba lógica simbólica para analizar estructuras bastante idealizadas. Rosenblatt opinaba que la herramienta de análisis más apropiada era la teoría de probabilidades, y esto lo llevó a una teoría de separabilidad estadística que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias.

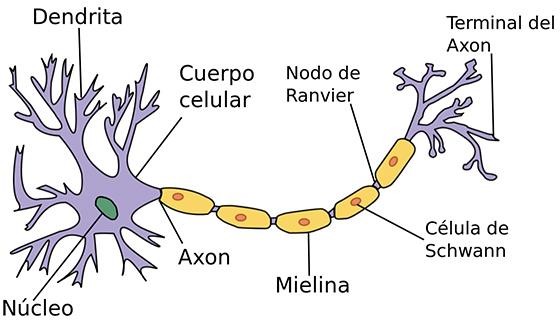
El Perceptrón era inicialmente un dispositivo de aprendizaje, en su configuración inicial no estaba en capacidad de distinguir patrones de entrada muy complejos, sin embargo, mediante un proceso de aprendizaje era capaz de adquirir esta capacidad. En esencia, el entrenamiento implicaba un proceso de refuerzo mediante el cual la salida de las unidades A se incrementaba o se decrementaba dependiendo de si las unidades A contribuían o no a las respuestas correctas del Perceptrón para una entrada dada. Se aplicaba una entrada a la retina, y el estímulo se propagaba a través de las capas hasta que se activase una unidad de respuesta. Si se había activado la unidad de respuesta correcta, se incrementaba la salida de las unidades A que hubieran contribuido. Si se activaba una unidad R incorrecta, se hacía disminuir la salida de las unidades A que hubiesen contribuido.

Mediante estas investigaciones se pudo demostrar que el Perceptrón era capaz de clasificar patrones correctamente, en lo que Rosenblatt denominaba un entorno diferenciado, en el cual cada clase estaba formada por patrones similares. El Perceptrón también era capaz de responder de manera congruente frente a patrones aleatorios, pero su precisión iba disminuyendo a medida que aumentaba el número de patrones que intentaba aprender.

1. DESARROLLO DEL TEMA

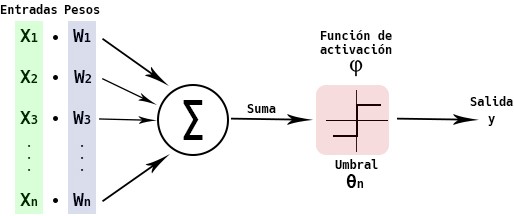
¿qué podríamos hacer con una única neurona? Para averiguarlo, primero tendríamos que saber cómo funciona.

El modelo biológico de una neurona tiene esta pinta:

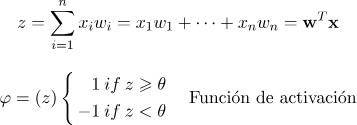


Resulta que en 1943, McCulloch y Pitts desarrollaron un modelo matemático que emula el comportamiento de una neurona. En el modelo biológico, las dentrinas tienen la función de recibir impulsos. En función de esos impulsos, se transmite una señal electroquímica a otras neuronas a través del Axón y sus terminales. La neurona McCulloch-Pitts (MCP) emula este comportamiento. El modelo recibe unas entradas, que llevan unos pesos asociados que simulan la intensidad de los impulsos. El valor de dichas señales se suma y, si dicha suma supera un cierto valor umbral, se produce una salida. El umbral actúa sobre una función de activación, que decidirá entre dos valores el valor de la salida, dependiendo si la suma de las señales supera o no dicho valor umbral.

La cual se puede ver claramente en el siguiente modelo:



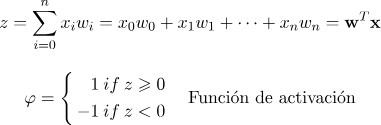
Mediante algunas ecuaciones se puede evidenciar las bases matemáticas seguidas:



Con una pequeña transformación logramos pasar el umbral θ al otro lado de la ecuación, lo cual simplifica las cosas si definimos lo siguiente:

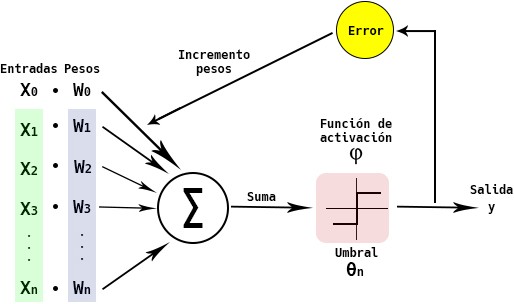


Obteniendo la siguiente ecuación:



En 1957, Frank Rosenblatt puso nombre a lo que, en Machine Learning, podríamos llamar una neurona artificial. La llamó Perceptrón. Basándose en el modelo MCP, desarrolló un algoritmo matemático que, simulando el comportamiento de las neuronas, tiene la habilidad de aprender. Lo que consigue aprender dicho algoritmo son los pesos adecuados para que la salida sea la correcta. Y como al final está decidiendo entre dos valores, podemos decir que el perceptrón está clasificando en dos clases los datos.

¿En qué consiste el algoritmo de Rosenblatt? Al perceptrón se le pasa un conjunto de datos de entrenamiento. La regla de aprendizaje del perceptrón consiste en ir añadiendo un incremento a los pesos de las entradas hasta que las salidas obtenidas por el perceptrón coincidan con las salidas esperadas del conjunto de datos de entrenamiento. Prueba y error.



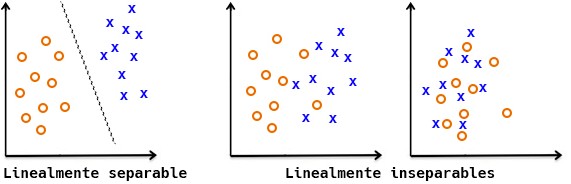
Para comenzar con el algoritmo, se deben inicializar todos los pesos en 0,con esta inicialización lo que se hace es calcular la salida para cada una de las entradas y se va modificando el valor de las pesos siguiendo la ecuación:



Donde se tiene que:

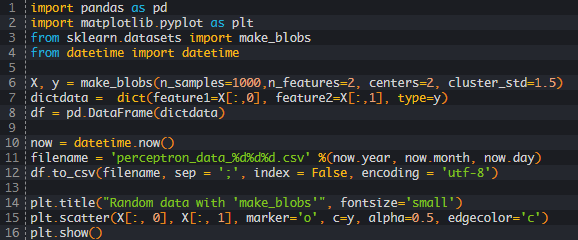
* δ sería la salida esperada del conjunto de datos de entrenamiento
* y sería la salida calculada por la neurona.
* η sería la tasa de aprendizaje, con un valor entre 0 y 1.

Para que este perceptrón pueda converger a un resultado esperado, el conjunto de entradas debe ser linealmente separable, lo que significa deben estar de la siguiente manera:



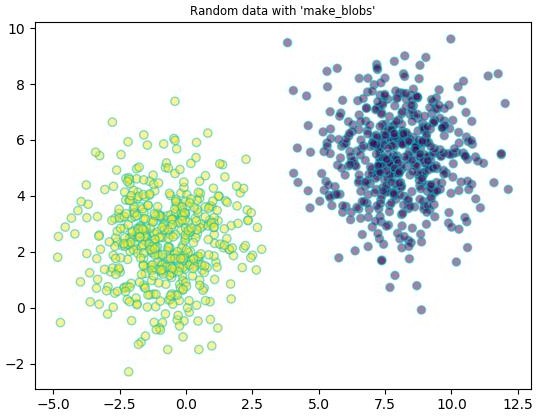
Además de esta condición la tasa de aprendizaje debe ser pequeña para que se pueda utilizar de manera adecuada el perceptrón.

Pero para verlo con más claridad como funcionaria en la realidad, se mostrará un ejemplo del entrenamiento realizado en Python haciendo uso de la librería de Machine Learning: scikit-learn. Y más concretamente la función make\_blobs, para generar datos aleatorios para clasificación.

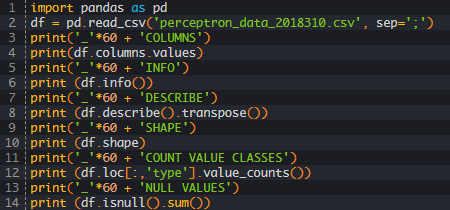


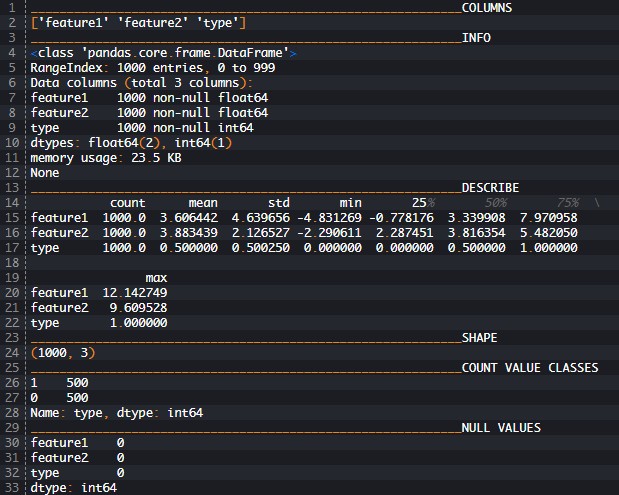
Con éste código generamos un dataset con 1000 muestras, dos características y dos clusters, con una desviación típica de los clusters de 1.5. El código te hace una gráfica del dataset, donde se puede comprobar si son linealmente independientes. Además, guarda los datos en un archivo csv.

La gráfica que genera es la siguiente:



En las siguiente imágenes se aprecia los datos que lanza el programa:





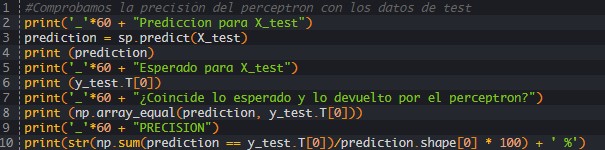
Para pobar el correcto funcionamiento del percetron se divide el dataset en dos conjuntos, uno para el entrenamiento y otro para comprobar que la salida sea correcta utilizando la función train\_test\_split.

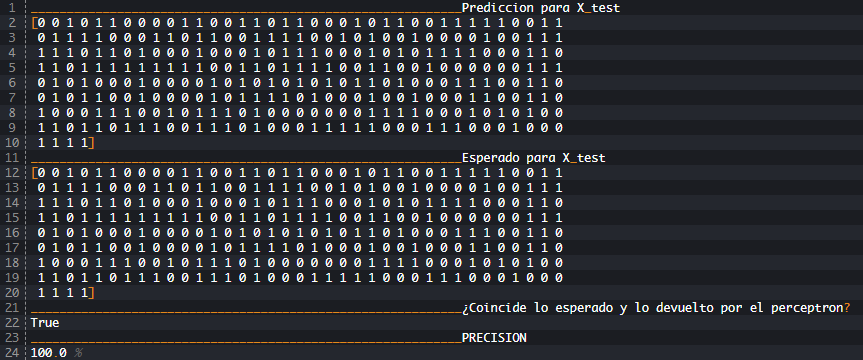


Luego se crea una clase perceptrón la cual incluirá tres métodos:

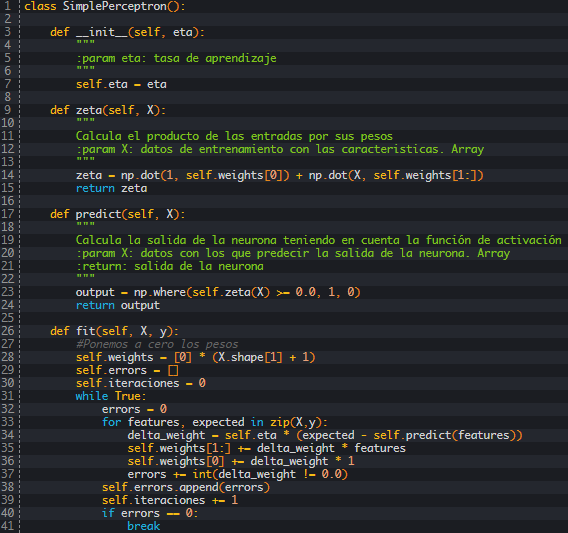
* zeta: con el que se calculan los productos de las entradas por sus pesos.
* predict: con el que se calcula la salida de la neurona a partir de los datos devueltos por zeta y teniendo en cuenta la función de activación. En nuestros datos de entrenamiento tenemos dos estados, dos clases, 1 y 0. Por lo tanto nuestra función de activación devolverá un 1 si z es mayor o igual a uno y 0 en el resto de casos.
* fit: donde se producen las iteraciones sobre el conjunto de los datos, contando el número de iteraciones, el número de errores en cada iteración y se van actualizando los pesos hasta llegar a cero errores.

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos



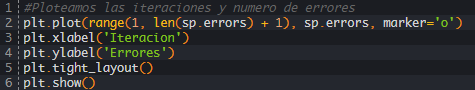


Como vemos en la imagen anterior, se evidencia que el entrenamiento ha sido correcto y obtenemos un acierto del 100%.

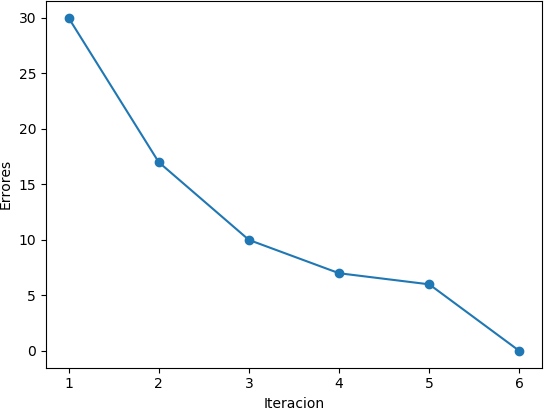
Ahora sigue la prueba, hacemos lo siguiente:



Luego se plotea las iteracciones con sus errores:



La siguiente gráfica nos muestra que se ha necesitado 6 iteraciones para que encuentre los pesos correctos:



Luego de tener el perceptrón entrenado, se le pasa el conjunto de prueba para comprobar el nivel de aciertos que obtiene:

1. CONCLUSIONES
   * Aunque el perceptrón simple tiene algunas limitaciones como se pudieron ver en el desarrollo del documento como lo son: datos deben ser linealmente separables, y tasa pequeña de aprendizaje, esto no le quita la utilidad que este posee, y el avance que significo para la época.
   * Las redes neuronales en sí, son un gran apoyo para los profesionales en el día a día, además de tener en muchos casos mayor porcentaje de acierto que las investigaciones realizadas con humanos, al ser una herramienta muy poderosa de cálculo.
   * La IA busca ofrecer soluciones algorítmicas con un coste bajo. En comparación de los modelos estadísticos clásicos, el uso de las RNA para el análisis de datos es más fácil ya que no es necesario que se tengan en cuenta los diferentes supuestos estadísticos: homogeneidad de la matriz, normalidad, mal procesamiento de los datos, etc. Además de modificar los pesos de las conexiones de forma automática y de ser un sistema tolerante a los fallos.

REFERENCIAS

<https://koldopina.com/como-entrenar-a-tu-perceptron/>

[http://www.lcc.uma.es/~munozp/documentos/modelos\_compu](http://www.lcc.uma.es/%7Emunozp/documentos/modelos_computacionales/temas/Tema4MC-05.pdf) [tacionales/temas/Tema4MC-05.pdf](http://www.lcc.uma.es/%7Emunozp/documentos/modelos_computacionales/temas/Tema4MC-05.pdf)

[http://dspace.uib.es/xmlui/bitstream/handle/11201/1126/TFG](http://dspace.uib.es/xmlui/bitstream/handle/11201/1126/TFG%20Marta%20Vidal%20Gonz%C3%A1lez.pdf?sequence=1)

[%20Marta%20Vidal%20Gonz%C3%A1lez.pdf?sequence=1](http://dspace.uib.es/xmlui/bitstream/handle/11201/1126/TFG%20Marta%20Vidal%20Gonz%C3%A1lez.pdf?sequence=1) <https://inteligenciaartificial101.wordpress.com/tag/perceptron/>

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos