Red Neuronal, el perceptrón multicapa.

Autor 1: Santiago Londoño Autor 2: David Cediel, Autor 3: Juan Pablo Narváez
Facultad de ingenierías, Universidad Tecnológica de Pereira, Pereira, Colombia

***Resumen*—Una red neuronal es un modelo inspirado en el comportamiento del cerebro humano, este modelo es usado en el campo del machine learning para realizar clasificación o regresión sobre un conjunto de datos, existen varios tipos de redes neuronales, pero el más común y el que se va a tratar en este artículo es el perceptrón multicapa.**

***Palabras clave*— Red, neurona, optimización, perceptrón, gradiente, descenso, capas, época, activación, función, mínimos cuadrados, error.**

***Abstract*— A neural network is a model inspired by the behavior of the human brain, this model is used in the field of machine learning to perform classification or regression on a set of data, there are several types of neural networks, but the most common and the one that is going to deal with in this article is the multilayer perceptron.**

***Key Word* — Network, neuron, optimization, perceptron, gradient, descent, layers, epoch, activation, function, least squares, error.**

I. INTRODUCCIÓN

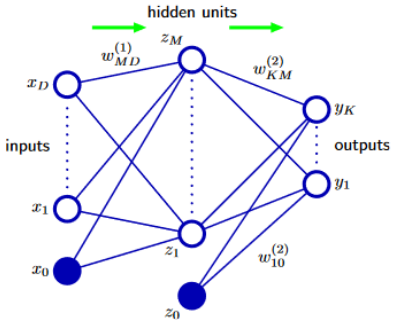
Las redes neuronales han tomado fuerza en el campo del machine learning en la última década, desplazando así modelos muy importantes tales como los modelos Bayesianos.

En este artículo se explicará cómo está compuesta una red neuronal, y cual es el proceso mediante el cual puede realizar la clasificación o regresión a partir de un conjunto de datos, luego de esto, se explicará el código realizado para crearla.

II. CONTENIDO

A. Qué es una red Neuronal

Elementos de la red neuronal:
La red neuronal esta compuesta de unidades básicas llamadas **neuronas**, estas neuronas se organizan en **capas**, donde, en una red típica, la salida de cada neurona de una capa se convierte en una entrada para cada neurona de la capa siguiente.



Img 1: Arquitectura de una red neuronal [1]

Dentro de la neurona ocurre un proceso de **combinación lineal** donde se operan todas las entradas que esta tiene (al igual que en el perceptrón), cada entrada tiene asociado un **peso** al cual se le denomina W, este será el parámetro que se debe optimizar para poder conseguir así minimizar el error.

$$a_j = \sum_{i=1}^{D} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)},$$

Ec 1: Combinación lineal que ocurre en la neurona j de la capa 1 [1]

Luego de realizar la combinación lineal de las entradas, el resultado es pasado por una **función de activación** para así lograr que, si el conjunto de datos es no linealmente separable en el dominio de entrada, pueda serlo luego de pasar por esta función, también es usada para denotar una salida a partir del vector de entrada.

$$z_j = h(a_j).$$

Ec 2: Salida de la neurona j pasada por la función de activación [1]

Fecha de Recepción: 2 octubre 2019
Fecha de Aceptación:

Luego de pasar por esta función, el resultado (Z) será usado como entrada para las neuronas de la siguiente capa, donde ocurrirá el proceso de combinación lineal de nuevo.

$$a_k = \sum_{j=1}^{M} w_{k,j}^{(2)} z_j + w_{k,0}^{(2)},$$

Ec 3: Combinación lineal dentro de la neurona k de la capa 2 [1]

Otro elemento importante en la red neuronal el valor bias, este es un valor que existe en cada capa de la red y representa un valor de sesgo, en este elemento no ocurre ninguna combinación lineal, sino que es un valor concreto, normalmente 1.

Existen tres tipos de capa en una red neuronal:

1. Capa de entrada: Es la capa que está ubicada al inicio de la red, recibe la información de los atributos que tiene el conjunto de datos
2. Capas ocultas: son las capas en las cuales se realiza constantemente una combinación entre las entradas para luego así pasar por la función de activación, su nombre deriva del hecho que estas operaciones no son visibles en el resultado y no tienen contacto con el entorno, solo con ellas mismas.
3. Capa de salida: Es la capa ubicada al final de la red, esta tiene los resultados los cuales servirán para optimizar la red por medio del algoritmo del descenso del gradiente si aun está en fase de entrenamiento o para realizar una predicción. Es de importancia recalcar que, si el problema que se está tratando es un problema de clasificación, a esta salida se la aplica una función sigmoide para así, a partir de un umbral, predecir a qué clase pertenece el parámetro de entrada.

$$\text{Regresión} \rightarrow y_k = a_k.$$
$$\text{Clasificación} \rightarrow y_k = \sigma(a_k).$$

Ec 4: Aplicación o no de la función sigmoide a la salida dependiendo del problema [1]

Entrenamiento de la red neuronal

En el caso de la red neuronal asó como en muchos modelos de machine learning, el entrenamiento de un modelo para que realice una predicción se realiza minimizando una función de error, esta depende de si el problema es de clasificación o de regresión.

La función de error para el modelo de regresión es la función de error cuadrático medio.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{w}, \mathbf{x}_n) - \mathbf{t}_n\|^2.$$

Ec 5. Función de error para regresión [1]

Para ello, se deriva la función de error con respecto a W, para ejemplificarlo se considera un ejemplo con dos capas ocultas.

$$\frac{dE(\mathbf{w})}{d\mathbf{w}} = \left[\left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_1} \right)^{\top} \left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_2} \right)^{\top} \right]^{\top},$$

Ec 6: Derivada de la función de error [1]

En la ecuación mostrada anteriormente, W1 y W2 son vectores con los pesos de cada una de las entradas para cada neurona de la capa correspondiente.

Con el algoritmo de backpropagation, la derivada de la función de error se puede calcular de una manera óptima, la cual queda resumida en las siguientes fórmulas

$$\delta_k = y_k - t_k.$$

Ec 7: Error imputado para las neuronas de la última capa [1]

$$\delta_j = (1 - z_j^2) \sum_{k=1}^{K} w_{k,j}^{(2)} \delta_k.$$

Ec 8: Error imputado para las neuronas de la capa oculta [1]

$$\frac{\partial E_n}{\partial w_{j,i}^{(1)}} = \delta_j x_i, \quad \frac{\partial E_n}{\partial w_{k,j}^{(2)}} = \delta_k z_j.$$

Ec 7: Error imputado para las neuronas de la primera y segunda capa [1]

Con estos resultados, es posible actualizar los pesos para así minimizar la función de error

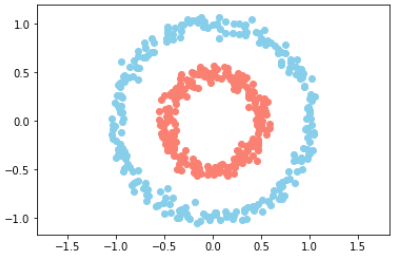
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

Ec 8: Actualización de los pesos de cada entrada [1]

B. Explicación del código.

Para realizar la red neuronal se utilizaron 3 librerías principalmente: numpy, la cual nos permitió usar funciones matemáticas; sklearn, de la cual se obtuvieron los datos para el entrenamiento y la predicción y por último matplotlib, la cual fue usada para graficar varios elementos.

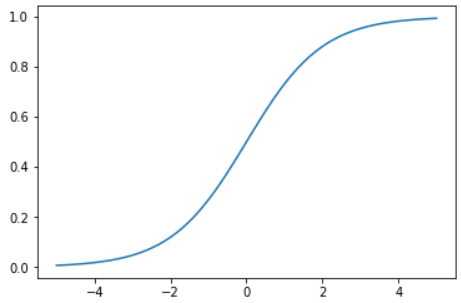
Se empezó sacando el data set, para que fuera un elemento retador, se eligió un conjunto de datos no linealmente separables.



Img 2: Data set usado en la red neuronal [2]

Luego de esto se creo la clase capa neuronal, la cual instanciaba el bias y los pesos de cada neurona de manera aleatoria, en esta clase se especificaba el número de neuronas de cada capa y el número de entradas de cada neurona, también se especificaba el tipo de función de activación por el cual iba a pasar la salida de las neuronas de dicha capa.

A continuación, se especificaron las funciones de activación, se codificó la función sigmoide y la función relu, aunque solo se usa la sigmoide, cada una de estas funciones con sus respectivas derivadas ya que son necesarias para el algoritmo de backpropagation



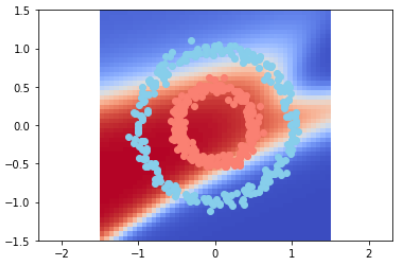
Img 3: Función de activación sigmoide [2]

Después se codificó la función de error con su respectiva derivada, también, se creó la función crear red, la cual, a partir de una topología y una función de activación instanciaba a la clase Capa neuronal creada anteriormente.

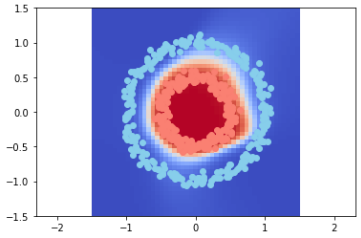
Para finalizar la creación de la red neuronal, se escogió una topología a usar.
Para el entrenamiento de la red se codificó la función [1] y [2] para realizar la combinación lineal de las entradas y pasarlo por la función de activación

Luego de esto, se calculaban los errores en cada capa y se actualizaba el valor de W y b a partir del algoritmo del descenso del gradiente.

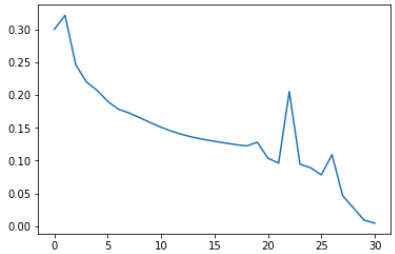
Por último, se codificó una función la cual realizara la optimización de la red neuronal hasta un valor de error aceptable, este valor se fijó en 0.005, así, la red neuronal pasaba por un número de épocas hasta llegar a un error muy bajo. En esta función, cada 50 iteraciones se graficaba la función de error y la estimación actual que estaba realizando la red, así, se podría tener una mejor idea de lo que se estaba realizando por dentro.



Img 4. Entrenamiento de la red neuronal en pocas iteraciones [2]



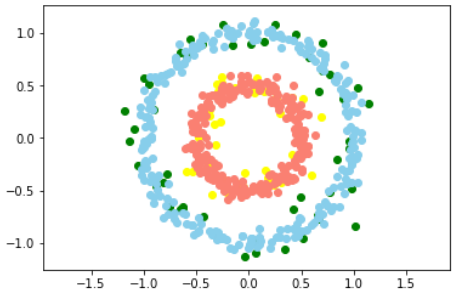
Img 5. Red neuronal entrenada [2]



Img 6. Valor de la función de error a medida de las iteraciones [2]

Se puede evidenciar que en la gráfica de la función de error hay un pico, este puede deberse a que la red atravesó un mínimo local.

Por último, se realizaron estimaciones con nuevos datos a partir de la red ya entrenada



Img 7: Estimación de nuevos datos a partir de la red entrenada [2]

III. CONCLUSIONES

Se mostraron los componentes de una red neuronal, como interactúan entre se para definir su arquitectura, luego se expuso el entrenamiento de esta y por último se explicó el código realizado para crear la red.

BIBLIOGRAFÍA

https://optimizacionheuristica.blogs.upv.es/files/2013/04/Introducci%C3%B3n-Redes-Neuronales-ArtificialesMFM.pdf

http://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/

https://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo2.html

https://es.wikipedia.org/wiki/Red_neuronal_artificial

REFERENCIAS

[1] C. M. BISHOP, Pattern Recognition and Machine Learning, Cambridge: Springer, 2006.

[2] Creación Propia