Perceptron y backpropagation

Perceptron and backpropagation

Autor 1: Sebastián Mora Carmona
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: sebastian.mora@utp.edu.co
Autor 2: Federico Idárraga Cardenas
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: federico.idarraga@utp.edu.co

***Resumen*— Este documento presenta un resumen de los principales contenidos de el perceptrón y backpropagation aquí pasaremos por su explicación, veremos las partes que los componen explicando cada una de ellas y viendo como trabaja la función de activación, la suma ponderada y el entrenamiento en general de tanto el perceptrón y el backpropagation. Por último veremos aplicaciones para las cuales son usados estos algoritmos.**

***Palabras clave*— *perceptrón, backpropagation,* machine learning, inteligencia artificial**

***Abstract*— This document presents a summary of the main contents of the perceptron and backpropagation, here we will go through their explanation, we will see the parts that compose them, explaining each one of them and seeing how the activation function works, the weighted sum and the training in general of both the perceptron and backpropagation. Finally we will see applications for which these algorithms are used.**

***Key Word*— perceptron, backpropagation, machine learning, artificial intelligence**

I. INTRODUCCIÓN

La redes neuronales artificiales se han convertido en una herramienta indispensable en los sistemas de información modernos estando ya presentes en nuestro dia a dia de forma transparente, por lo que se hace necesario tener un conocimiento básico de su funcionamiento.

En este documento se presenta una explicación breve y práctica de dos de los pilares fundamentales de las redes neuronales, permitiendo que el lector pueda obtener de este conocimientos básicos de su funcionamiento ,su teoria subyacente y su campos de aplicación.

I.1 PERCEPTRÓN

Un perceptrón es un clasificador lineal (binario). Además, se utiliza en aprendizaje supervisado. Ayuda a clasificar los datos de entrada dados. Se puede visualizar en [1]

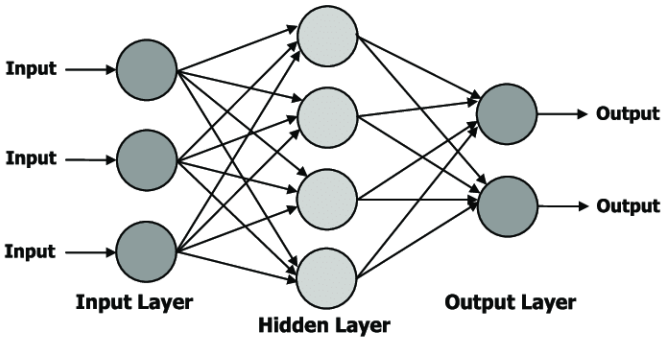

FIGURA 1

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.

Partes del Perceptrón

1. Valores de entrada o capa de entrada

Son los datos de entradas que entran al sistema y que serán clasificados en la capa de salida. Contiene los datos con los que se entrenará su modelo. Cada neurona en la capa de

entrada representa un atributo único en su conjunto de datos (por ejemplo, altura, color de cabello, etc.).

2. Pesos y sesgo

Podemos interpretar que cada peso, wi, representa la influencia relativa de la entrada por la cual se multiplica, xi. A menudo al término b se le llama sesgo (en inglés bias), ya que controla qué tan predispuesta está la neurona a disparar un 1 o un 0 independiente de los pesos. Un sesgo alto hace que la neurona requiera una entrada más alta para generar una salida de 1. Un sesgo bajo lo hace más fácil.

3. Suma Ponderada

Es la suma neta que entra a la función de activación y debe cumplir para generar la salida.

4. Función de activación

La función de activación devuelve una salida que será generada por la neurona dada una entrada o conjunto de entradas. Cada una de las capas que conforman la red neuronal tienen una función de activación que permitirá reconstruir o predecir. Además, se debe considerar que en la red neuronal se usará una función no lineal debido a que le permite al modelo adaptarse para trabajar con la mayor cantidad de datos.
Las funciones de activación se dividen en dos tipos como: lineal y no lineal.

- Función Lineal [2]

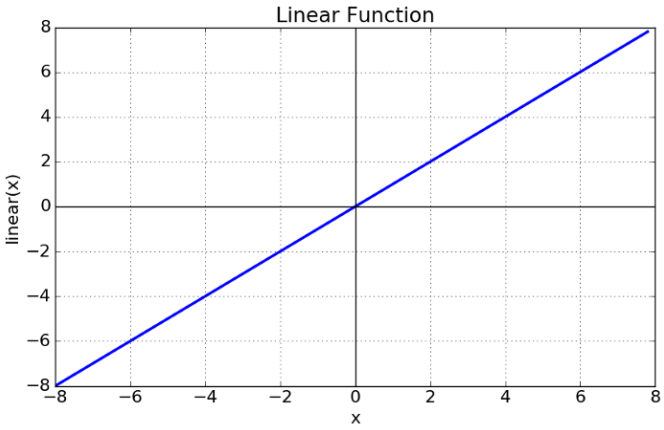

FIGURA 2

Esta función también conocida como identidad, permite que lo de la entrada sea igual a la salida por lo que si tengo un red neuronal de varias capas y aplicó función lineal se dice que es una regresión lineal.

- Funciones no Lineales

- Función Umbral [3]

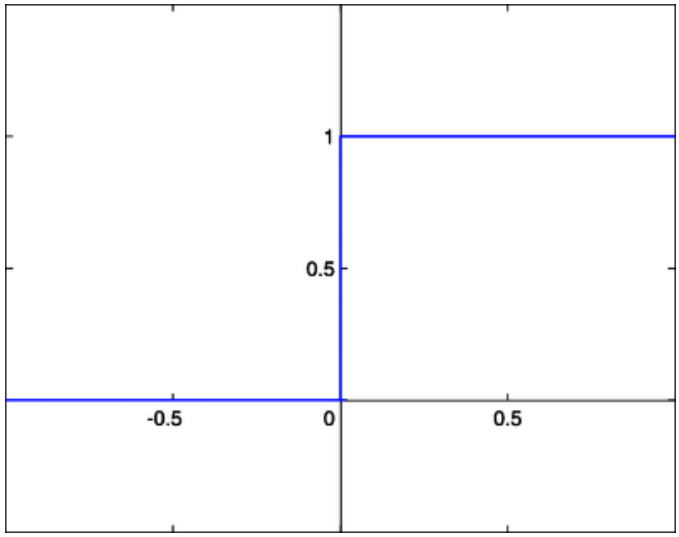

FIGURA 3

Esta función también conocida como escalón, indica que si la x es menor que cero la neurona va a ser cero pero cuando es mayor igual a cero dará como salida igual 1.

- Función sigmoide [4]

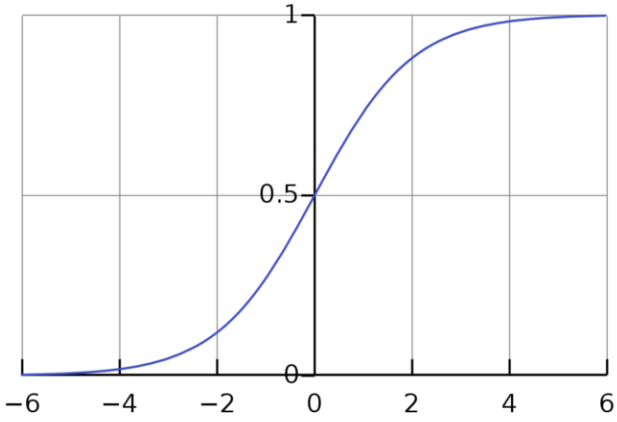

FIGURA 4

Esta función también conocida como función logística, está en un rango de valores de salida entre cero y uno por lo que la salida es interpretada como una probabilidad.

- Función tangente hiperbólica [5]

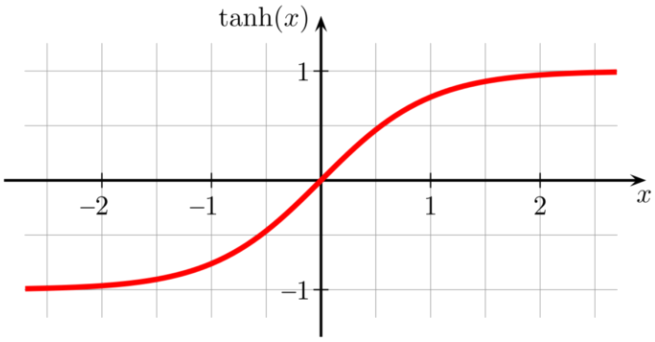

FIGURA 5

Esta función de activación llamada tangente hiperbólica tiene un rango de valores de salida entre -1 y 1.

- Función ReLu [6]

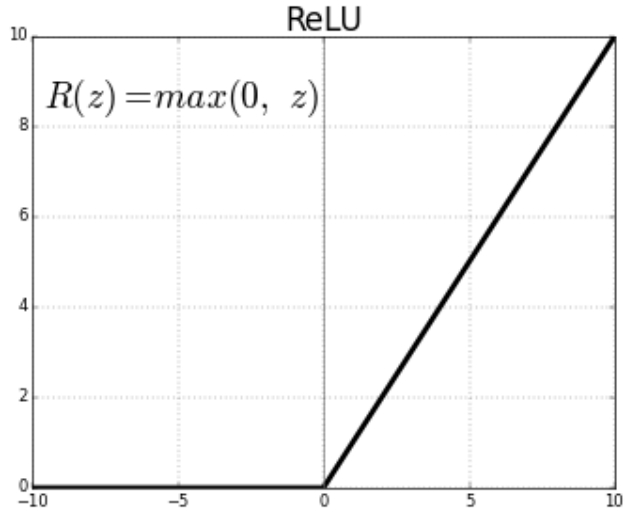

FIGURA 6

Está función es la más utilizada debido a que permite el aprendizaje muy rápido en las redes neuronales. Si a esta función se le da valores de entrada muy negativos el resultado es cero pero si se le da valores positivos queda igual y además el gradiente de esta función será cero en el segundo cuadrante y uno en el primer cuadrante.

- Función softmax [7]

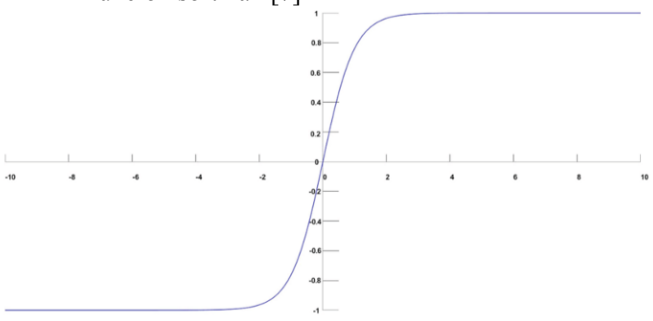

FIGURA 7

Esta función se usa para clasificar data, por ejemplo si le damos de entrada la imagen de una fruta y se solicita saber el tipo de fruta a que pertenece, aplicando softmax la red nos dará la probabilidad de que puede ser 0.3 o 30% melón, 0.2 o 20% sandía y 0.5 o 50% papaya, por lo que nos el resultado será el que tenga mayor probabilidad y cabe recalcar que la suma de estas probabilidades será igual a 1.

Podemos visualizar las partes del perceptrón de las que estábamos hablando en [8]

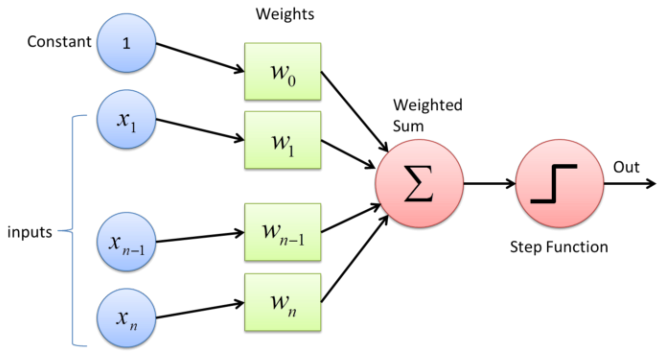

FIGURA 8

Pasos del Perceptrón

1. Todas las entradas x se multiplican por sus pesos w. Lo llamamos K. [9]

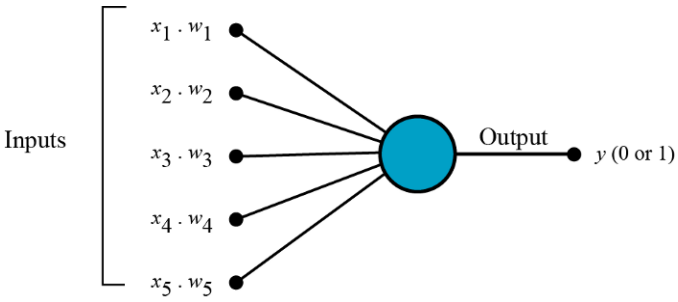

FIGURA 9

2. Sumamos todos los valores multiplicados y lo llamamos suma ponderada.

3. Aplicamos nuestra suma ponderada a la función de activación. Ejemplo [10]

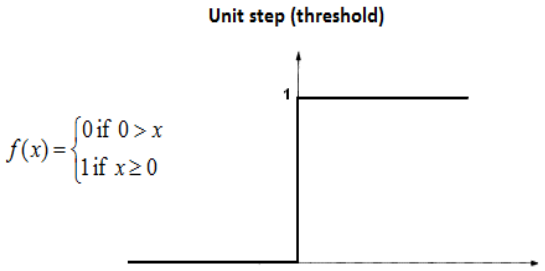

FIGURA 10

Uso del Perceptrón

El perceptrón se usa generalmente para clasificar los datos en dos partes. Por lo tanto, también se conoce como clasificador binario lineal, como dijimos anteriormente. Podemos ver una imagen de este clasificador binario lineal en [11]

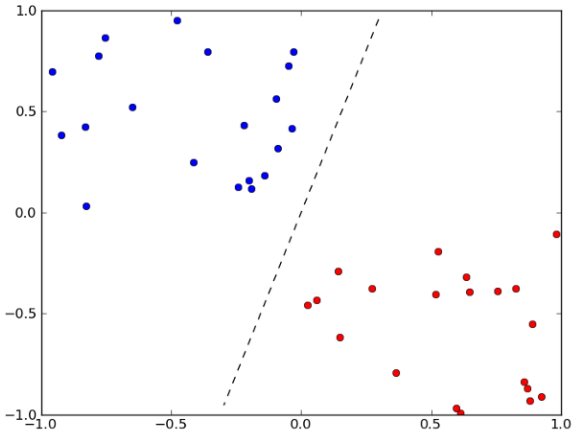

FIGURA 11

I.2 BACKPROPAGATION

Conforme el uso de las redes neuronales se ha ido extendiendo a diferentes campos de la ciencia y la ingeniería la confianza en los modelos se hace fundamental. El factor más importante para lograr un modelo confiable es el entrenamiento.

El entrenamiento de las redes neuronales está asociado al "backpropagation" que en la práctica es el proceso mediante el cual se ajustan los pesos de las neuronas basado en la tasa de error [12].

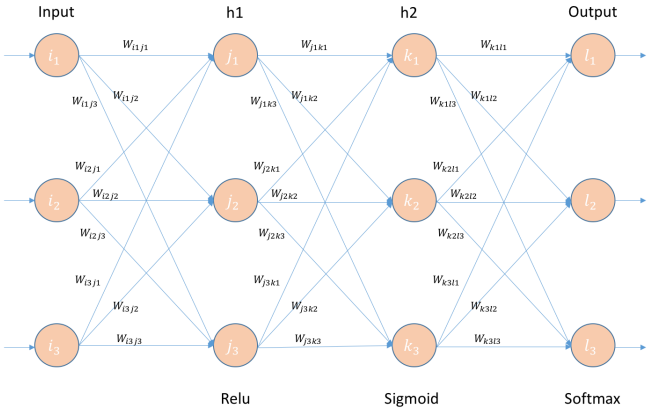

FIGURA 12

Los Componentes del Modelo

1. función XOR por medio de una red neuronal

Se quiere entrenar una red neuronal que pueda llevar a cabo la función XOR con dos neuronas de entrada y tres ocultas [13].

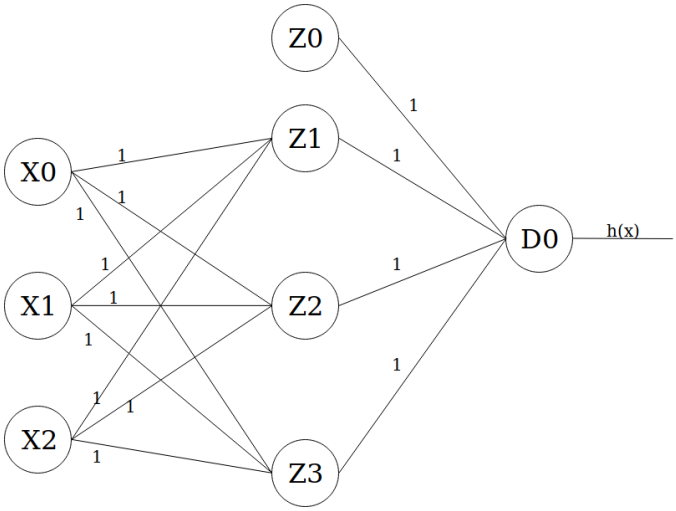

FIGURA 13

2. Función de activación y Función de hipótesis

Se requiere un función de activación que determine la activación en cada neurona para ello se usará la función de activación identidad:

$$f(a) = a$$

También se necesita un función que determine cuál será la entrada de la función de activación:

$$h(X) = W0.X0 + W1.X1 + W2.X2$$

3. Funcion de perdida.

Se escoge como función de pérdida la regresión logística [14]

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

FIGURA 14

Más adelante se usará el descenso gradiente por lotes para determinar la dirección en la cual ajustar los pesos para obtener la menor pérdida, tendremos una tasa de aprendizaje de 0.1 y los valores de los pesos serán 1 inicialmente.

4. Propagación hacia adelante.

Se ingresan ciertos valores una capa hacia otra, esto se realiza en dos pasos:

- Obteniendo la suma ponderada de una unidad particular utilizando h(x).
- Ingresando el valor obtenido en el paso anterior por la función de activación f(a) y usando el resultado de este como entrada para los nodos conectados en la siguiente capa.

En los nodos X0, X1, X2, Z0 no se realizan los anteriores pasos ya que no tiene un capa anterior.

5. Ingresando una muestra de entrada

Unidad Z1:

$$h(x) = W0.X0 + W1.X1 + W2.X2 = 1.1 + 1.0 + 1$$
$$z = f(a) = a \Rightarrow z = f(1) = 1$$

Este mismo proceso se realiza en cada unidad:

Unidad Z2:

$$h(x) = 1$$
$$z = 1$$

Unidad z3:

$$h(x) = 1$$
$$Z = 1$$

Unidad D0:

$$h(x) = W0.Z0 + W1.Z1 + W2.Z2 + W3.Z3 = 1.1 + 1.1 + 1.1$$
$$h(x) = 4$$

$$z = f(4) = 4$$

La función Z de activación de la unidad final (D0) es el resultado de todo el modelo, por lo tanto el modelo predijo 1 de el input {0,0}, Calculando la pérdida:

$$perdida = 0 - 4$$
$$perdida = -4$$

Entrenando el modelo

De acuerdo con este ejemplo se tiene un modelo muy poco preciso y esto se atribuye al hecho de que los pesos no han sido ajustados.
El Backpropagation trata de alimentar la pérdida hacia atrás de tal manera que se puedan ajustar los pesos basados en esto. Se usará la función de optimización de descenso de gradiente para encontrar los pesos

1. Derivadas Parciales,

Cuando se realiza la propagación hacia adelante se usaron la siguientes funciones:

$$f(a) = a$$

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

Por lo tanto la propagación hacia atrás se llevará a cabo con la derivadas parciales de estas funciones:

$$f'(a) = 1$$

$$J'(W) = Z.\ delta$$

Donde Z es el valor obtenido por la función de activación del paso adelante,mientras que delta es la pérdida de la unidad en la capa.

2. Calculando los deltas

Se necesita calcular la pérdida de cada nodo en la red neuronal, esto con el fin de obtener información de cuáles nodos son los mayores responsables de la pérdida general del modelo.

Se tiene la siguiente fórmula para calcular el delta de cada unidad:

$$delta_0 = w.\ delta_1\ .\ f'(z)$$

donde delta0, w, f'(z) son de la misma unidad, mientras que delta1 es la pérdida en el otro lado del enlace ponderado como se muestra en la figura [15].

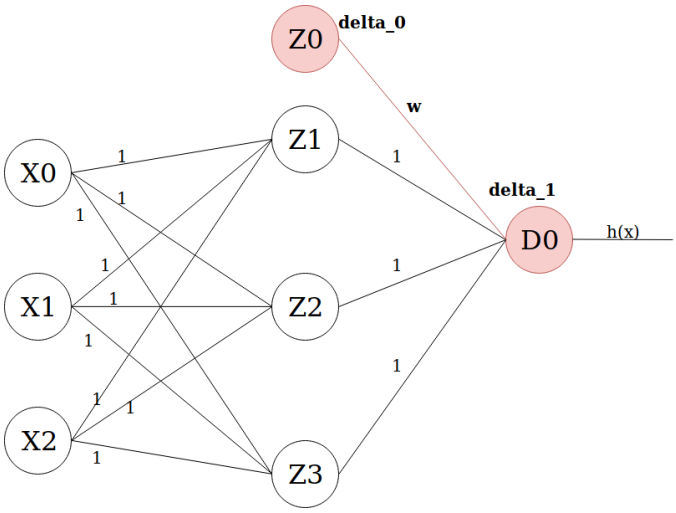

FIGURA 15

Calculando lo delta de cada nodo:

$$delta\_D0 = perdida\_total = -4$$
$$delta\_Z0 = W\ .\ delta\_D0\ .\ f'(Z0) = 1\ .\ (-4)\ .\ 1 = -$$
$$delta\_Z1 = W\ .\ delta\_D0\ .\ f'(Z1) = 1\ .\ (-4)\ .\ 1 = -$$
$$delta\_Z2 = -4$$
$$delta\_Z3 = -4$$

3. Actualizando los pesos

Ahora solo queda actualizar los pesos de la red neuronal siguiendo la fórmula de descenso gradiente por lotes:

$$W = W - alpha\ .\ J'(W)$$

Donde W es el peso en cuestión, alfa es la tasa de aprendizaje (es decir, 0.1 en nuestro ejemplo) y J '(W) es la derivada parcial de la función de costo J (W) con respecto a W.

Usando la función de la derivada parcial de la pérdida:

$$J'(w) = Z.delta$$
$$W = W - alpha.Z.delta$$

El cálculo de los nuevos pesos se haría de la siguiente forma:

W10 := W10 - alpha . Z_X0 . delta_Z1
= 1 - 0.1 . 1 . (-4) = 1.4
W20 := W20 - alpha . Z_X0 . delta_Z2
= 1 - 0.1 . 1 . (-4) = 1.4
. . . . .
. . . . .
. . . . .
W30 := 1.4
W11 := 1.4
W21 := 1.4
W31 := 1.4
W12 := 1.4
W22 := 1.4
W32 := 1.4
V00 := V00 - alpha . Z_Z0 . delta_D0
= 1 - 0.1 . 1 . (-4) = 1.4
V01 := 1.4
V02 := 1.4
V03 := 1.4

Se debe destacar que el modelo no estaría entrenado apropiadamente hasta este momento ya que se propagó solamente sobre una muestra del set de entrenamiento, esto se debe realizar una y otra vez por todas las muestras para obtener una mejor precisión.

I.3 CONCLUSIONES

Se ha demostrado que con inteligencia artificial se pueden realizar clasificaciones realmente interesantes como lo veíamos con el perceptrón que entre sus aplicaciones en multicapa podemos resolver problemas de asociación de patrones, segmentación de imágenes, compresión de datos, etc.

Las redes neuronales presentan grandes ventajas frente a otros métodos para la resolución de problemas de la ingeniería, podríamos decir que es la nueva ingeniería teniendo en cuenta que la filosofía de la ingeniería es la resolución rápida y efectiva de problemas las redes neuronales cumplen la mejor función ya que su inspiración es en el cerebro humano mismo lo que nos facilita entenderlo gracias a su analogía. Aparte de esto es fácil de implementar en lenguajes como python con librerías como tensor flow y keras.
Al igual como estuvimos explicando backpropagation tiene increíbles uso o aplicaciones como lo pueden ser:

- La red neuronal está entrenada para enunciar cada letra de una palabra y una oración.
- Se utiliza en el campo del reconocimiento de voz.
- Se utiliza en el campo del reconocimiento facial y de caracteres.

Que son aplicaciones muy importantes en un mundo en el cuál tenemos una increíble identidad digital y qué mejor manera de proteger nuestra información que con nuestra cara o voz. Ya es completamente normal que nuestros celulares tengan

reconocimiento facial al instante. La base de toda esta tecnología proviene de aquí.

REFERENCIAS

Referencias en la Web:

[1]
https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

[2]
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[3]
http://bibing.us.es/proyectos/abreproy/11084/fichero/Memoria+por+cap%C3%ADtulos+%252FCap%C3%ADtulo+4.pdf+#:~:text=El%20perceptr%C3%B3n%20es%20la%20forma,ambos%20lados%20de%20un%20hiperplano).&text=La%20prueba%20de%20convergencia%20del,teorema%20de%20convergencia%20del%20perceptr%C3%B3n.

[4]
https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html

[5]
https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da

[6]
https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb

[7]
https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7