Reconocimiento de patrones numéricos con red neuronal

Recognition of numerical patterns with neural network

Jhon Edinson Acevedo Rojas, Mauricio Marín Orozco, Juan Sebastian Herrera Giraldo

*Risaralda, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: juseherrera@utp.edu.co

***Resumen*— Mediante los algoritmos de “redes neuronales, redes neuronales backpropagation, perceptron”, se logra crear un programa capaz de reconocer números del cero al nueve (0-9). Este entrenamiento da como resultado que tan parecido es el patrón de entrada con el patrón de salida.**

***Palabras clave—*Red Neuronal, Backpropagation, perceptron, capas ocultas, capas de entrada, capas de salida.**

***Abstract*— through algorithms “neural network, backpropagation neural network, perceptron”, is able to create a program that can recognize numbers from zero to nine (0-9). This training result in so similar is the input pattern to the output pattern.**

***Key Word* —** **Neural Network, Backpropagation, hidden layers, input layers, output layers, perceptron**

1. INTRODUCCIÓN

Las redes neuronales fueron una simulación abstracta de los sistemas nerviosos biológicos. Ya que estos tienen dendritas, axón y el cuerpo de la neurona. Las redes neuronales artificiales tienen las siguientes características:

* Auto-Organización y adaptabilidad: utilizan algoritmos de aprendizaje y auto-organización, por lo que ofrecen mejores posibilidades de procesado robusto y adaptativo.
* Procesado no Lineal: Aumenta la capacidad de la red para aproximar funciones, clasificar patrones y aumentar su inmunidad frente al ruido.
* Procesado paralelo: Normalmente se usa un gran número de nodos con alto nivel de interconectividad.

1. CONTENIDO

Como se ha dicho nuestro objetivo es entrenar una red que reconozca patrones numéricos con y sin ruido.

1. Algoritmo de entrenamiento Red Neuronal Backpropagation.

El lenguaje utilizado para la creación de esta red neuronal es Python 3.x.

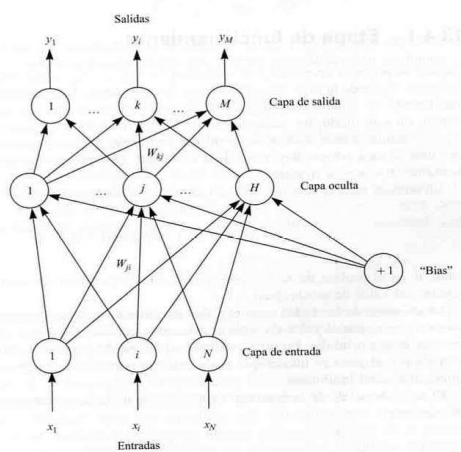


Figura 1.

Basados en la figura 1 se debe tener bien claro cuáles van a ser las entradas de nuestra red.

Una vez hecho esto se calcula todas las entradas para las capas ocultas, lo cual debe hacerse teniendo en cuenta los pesos y todas las entradas. Visto de una manera formal se puede ver:



Donde (tetha j) es el umbral de la neurona que se considera como un peso asociado ficticio a la salida de 1

Wji son los pesos.

Xi son las entradas.

El valor de la salida de la neurona de las capas ocultas se le aplica una función F(t). Visto formalmente



Una vez realizado este paso, se repite el cálculo de pesos, esta vez entre la capa oculta y la capa de salida, e incluimos la función F(t), la manera formal es la siguiente.



Y otra vez se debe calcular una función F(t) para la salida de la red.

.

1. Minimizar Errores de la Red Backpropagation

En el tema pasado se vio una parte importante del algoritmo de entrenamiento, una vez entendido esto se debe minimizar el error. Para ello se tiene en cuenta la salida deseada de la neurona ante la presentación del patrón.



Para calcular el error hacemos uso del vector director gradiente el cual es la derivada respecto a los pesos, como el vector gradiente toma la dirección más rápida al error y este se busca es minimizar, entonces se le da dirección negativa.



.

Para calcular los errores de manera iterativa se aplica regla de la cadena al gradiente y añadir una taza de aprendizaje n.



Para mayor información visitar la siguiente página web [1] Introducción a las redes neuronales.

1. Reconocimiento de patrones numéricos.

Visto el algoritmo de backpropagation sabemos que lo que hace es encontrar el mínimo error y reducir el ruido. En la figura 2. Se explica un tipo de ruido

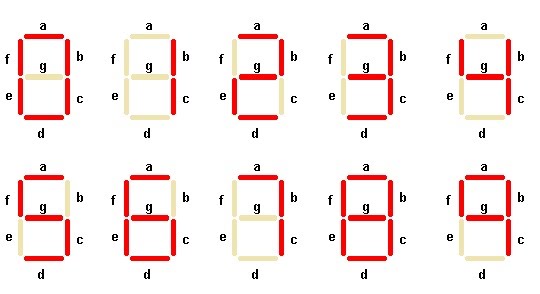


Figura 2.

El ruido se puede definir como aquello que interfiere con la forma original.

Un claro ejemplo puede ser el número nueve (9), como se observa en la figura 2. El nueve está conformado por las siguientes entradas [a, b, c, d, e, f, g], donde 1 significa encendido y 0 significa apagado.

El número nueve representado en forma de entrada es [1, 1, 1, 0, 0, 1, 1], pero si se añade ruido a la imagen se puede decir que el nueve también es [1, 1, 1, 1, 0, 1, 1] esto también debe generar como salida un nueve (9), y no solo para el número nueve, si nuestra entrada es de mayor cantidad pueden existir cientos de ruidos que hagan un poco complicado el aprendizaje.

1. CONCLUSIONES

El aprendizaje de una red neuronal puede tener tantas entradas como capas ocultas y salidas, aunque es un buen método de aprendizaje, este tiene cierto nivel de errores.

Este aprendizaje sirve tiene muchas aplicaciones, una de ellas es el famoso juego triki (tic tac toe), donde usted cada vez que juegue con la maquina ella va entrenado, hasta el punto de que piensa por ella misma.

A diferencia de la red neuronal con el algoritmo minimax en el triki, es evidente que tarda más con el algoritmo minimax debido a que ella genera un árbol con muchos nodos sobre la siguiente jugada, en cambio con la red neuronal es cuestión de entrenarla.

REFERENCIAS

1. http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf
2. <https://sites.google.com/site/electronicabasicayavanzada/_/rsrc/1439837814885/home/digitales-basico/0-9-en-display-7-segmentos-con-dipswitch/numeros%200-9.jpg>.