Investigación Redes Neuronales: Detección de números a mano alzada

Jesus David Mena, *Estudiante, Universidad Tecnologica de Pereira,*
Stiven Cardona Monsalve, Estudiante, Universidad Tecnologica de Pereira,

***Resumen*—En el documento a continuación se planteará de forma resumida La forma como se desarrollo un sistema que reconozca números digitados a mano, para ello se utilizaran redes neuronales, y los conocimientos acumulados en la asignatura de Computación Blanda.**

I. Introducción

En el campo del aprendizaje automático o machine learning, una de las tareas más importantes es la clasificación de nuevos datos. Para esto se han creado diversos algoritmos entre ellos los árboles de decisión, regresiones lineales, o redes neuronales que con sus diferentes arquitecturas nos ayudan a solucionar uno o varios tipos de problemas. Entre los diferentes tipos de redes neuronales nos encontramos con 3 que están siendo altamente usadas, las redes neuronales fully connected, que constan de redes cuya arquitectura se basa de una capa tras otra en donde todas las neuronas de una capa se conectan con todas las neuronas de la capa siguiente, y así consecutivamente hasta el final; en segundo lugar tenemos las redes neuronales convolutivas que implementan un conjunto de algoritmos que las hacen muy fuertes en el campo del reconocimiento de imágenes, pues todo su funcionamiento se basa en aplicar convoluciones sobre una entrada, luego reducirla imagen o dato mediante filtros, volver a aplicar convoluciones y consecutivamente hacer lo mismo hasta obtener una salida que se pueda pasar a una red fully connected, para que sea posteriormente clasificada; y por ultimo el tercer tipo de red, es la red neuronal recurrente, su fuerte principal es al momento de analizar series de datos que varían en el tiempo o tienen una secuencia, todo lo que corresponde a audios, texto y señales en general se aconseja trabajar bajo este tipo de red, pues la característica principal de este tipo de estructuras es que la red se implementó con el objetivo de que tuviese memoria.

II. Red neuronal Fully connected

Campo que se empezó a estudiar en 1936 por el científico Alan Turing, pues fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación, pero fue si no posteriormente por W. McCulloch y W. Pitts, el primero un neurofisilogo y el segundo un matemático, que en 1943 lanzaron la teora que trataba la forma de trabajar de las neuronas en su articulo ”Un Calculo Lógico de la Inminente Idea de la Actividad Nerviosa”, y fue también cuando modelaron mediante circuitos eléctricos la primera red neuronal simple. Con el paso del tiempo en tareas cuyo objetivo principal era la clasificación se desarrollaron este tipo de redes cuyas capas se conectaban totalmente unas entre otras como se puede observar en la figura 1.

Universidad Tecnológica de Pereira

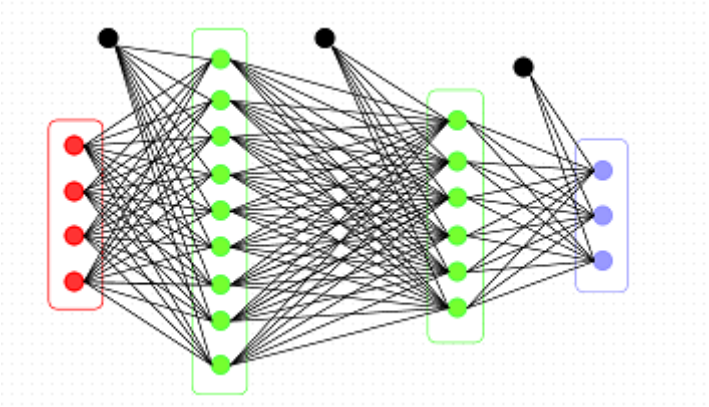

Figura 1. Titulo dela figura 1. Estructura de red con conexiones fully connected.

III. Detección de números

Para el entrenamiento de la red neuronal que se encargaría de detectar los números escritos a mano alzada, se uso la estructura de red fully connected con la siguiente estructura, una capa de entrada, dos capas ocultas y una capa de salida; las capas ocultas tienen la primera 128 neuronas y la segunda 512, mientras que la capa de entrada tiene 784, y la de salida 10 neuronas. La longitud de la capa de entrada corresponde a coger las imágenes del (mnist data set Figura 2), cuyas dimensiones son de $28x28$, cada una de estas imágenes eran cargadas en forma de matriz, con las mismas dimensiones, cuyos valores correspondían a solo uno de los canales de la imagen ”Teniendo en cuenta que las imágenes eran cargadas en formato RGB”, no tuvimos problemas en seleccionar uno u otro, debido a que las imágenes eran a blanco y negro. Posterior mente la matriz era aplanada y convertida en un vector de 784 posiciones que se pasaba a la capa de entrada, para luego proseguir con el fordward en la red.

III-A. Prepocesamiento de datos

Con ánimos de mejorar el aprendizaje en el modelo, a todas las imagines del set de datos se les hizo es respectivo pre procesamiento que consintió en normalizar la imagen, y todos los colores que sabemos van de 0 a 255 en RGB, volvimos todos esos valores, números entre 0 y 1. Esto ayuda

Figura 2. Titulo dela figura 2. Set de datos mnist para entrenar la red.

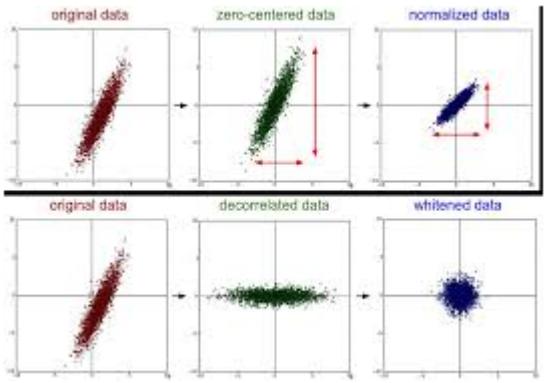

Figura 3. Titulo dela figura 3. Set de datos mnist para entrenar la red.

a que el espacio de solución al momento de aplicar los algoritmos de optimización tengan un espacio de soluciones mas limitado, como se puede ver en la figura 3, las imágenes azules corresponden al los valores normalizados.

Y posteriormente se hizo un procedimiento que suele ser llamado one hot encoding, que basicamente consiste en si tengo N categorias en mi conjunto de datos, todos los englobo y clasifico mediante un vector de N posiciones, en donde cada posición corresponde a cada una de las categorías, por ejemplo:

$$1 = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$$
$$5 = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$$
$$onehotencoding$$

Ahora bien dado que vamos a clasificar 10 diferentes números que corresponden a los digitos del 0 al 9, nuestro vector generado tiene 10 posiciones, y esto explica porque la estructura de nuestra red tiene 10 neuronas de salida.

III-B. Estructura de la red

Ya especificamos que la red tendría una estructura compuesta por 784 neuronas en la capa de entrada, 128 en la primera capa oculta, 512 en la siguiente capa, y 10 neuronas en la ultima capa, vale la pena aclarar que las dos capas ocultas manejan una función de activación sigmoide, aunque perfectamente podíamos utilizar la función Relu y obtener

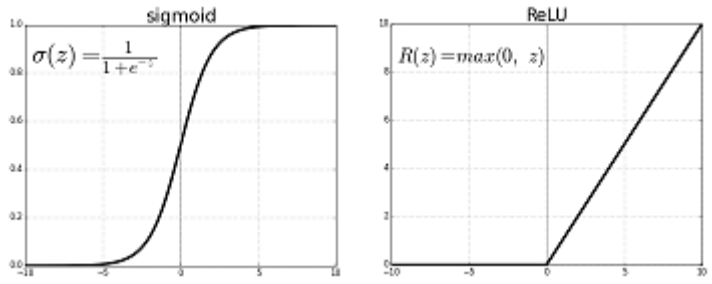

Figura 4. Titulo dela figura 4. Funcion sigmoid y funcion Relu.

resultados similares, estando cada una representada por las ecuaciones que se observan en la figura 4.

En cuanto a la ultima capa la función de activación utilizada fue la $softmax$, para manejar las salidas midiendo la probabilidad de cada valor, y así al final solo escoger la probabilidad mas alta, y ese valor utilizar como predicción.

III-C. Entrenamiento

El entrenamiento de la red se hizo mediante el algoritmo de optimización Adam, el cual es en realidad un algoritmo SGD (Stochastic Gradient Descent) en el que el gradiente utilizado en cada iteración se actualiza desde el anterior utilizando una técnica basada en momentos. Adam define su actualizacion mediante el siguiente algortimo:

$$\Delta_k = \mathbf{x}_k - \mathbf{x}_{k-1}$$

Entonces, $\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta_k$. En el algoritmo GD, esta actualización fue simplemente $\Delta_{k+1} = -\alpha_k G_J(\mathbf{x}_k)$.

Por otro lado la perdida o el error lo calculamos teniendo en cuenta el $categorical_crossentropy$ la cual mide el número promedio de bits necesarios para identificar un evento extraído del conjunto, si se utiliza un esquema de codificación optimizado para una distribución de probabilidad "no natural"q, en lugar de la distribución "verdadera"p, la cual en ultimas nos ayuda mucho cuando manejamos datos categóricos.

Al final lo que se hizo fue programar el entrenamiento para hacerlo 200 veces, con las imágenes del conjunto de aprendizaje, y luego calculamos el % de acierto en un conjunto de prueba, obteniendo al final un porcentaje de $98{,}4\,\%$.

IV. Conclusiones

Para concluir es bueno aclarar que el objetivo era poder reconocer números digitados a mano, pero se implemento usando el navegador web y dibujando con el cursor. El acierto depende de varios factores pero los resultados obtenidos cumplen con las expectativas planteadas.

Referencias

[1] $https : //www.quora.com/Can - you - explain - basic - intuition - behind - ADAM - a - method - for - stochastic - optimiza$
[2] $https : //en.wikipedia.org/wiki/Cross_entropy$
[3] Documentación de tensorflow, y Keras.