Reconocimiento de letras manuscritas con redes neuronales artificiales

Tercera y cuarta entrega

Edson Raul Cepeda Marquez
Sonia Alejandra Treviño Rivera

1 de junio de 2020

1. Resumen

Este proyecto consiste en una aplicación web que implementa una red neuronal artificial capaz de reconocer letras manuscritas. La aplicación web en cuestión está desarrollada con tres tecnologías que se conectan y permiten su uso:

1. **Angular**: Un framework de código libre desarollado por Google que facilita el desarrollo de aplicaciones web de una sola página (SPA: Single Page Applicaction). Tiene la ventaja de separar completamente el frontend y el backend lo que permite seguir facilmente el patrón de diseño MVC (Modelo-Vista-Controlador).
2. **Express**: Un framework de desarrollo web que utiliza el entorno de ejecución Node JS y facilita la creación de REST APIs.
3. **Python**: Un lenguaje de programación interpretado orientado a objetos. Tiene la ventaja de ser simple, lo que permite escibir código legible. Tiene una amplia comunidad de desarrollo y una gran cantidad de librerias para todo tipo de tareas.

En la figura 1 se muestra un diagrama que representa el flujo de información entre las tres tecnologías. Se puede observar que la información fluye en ambas direcciones.

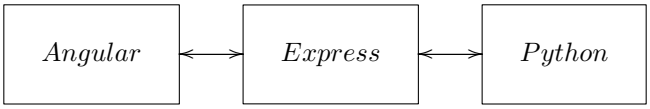

Figura 1: Flujo de información de la aplicación.

La aplicación web además de contener este proyecto, tiene otras secciones asignadas para proyectos futuros. Para este proyecto la sección importante es “Imchar”. Esta sección de la aplicación web contiene una cuadricula con la que se puede interactuar a traves del ratón.

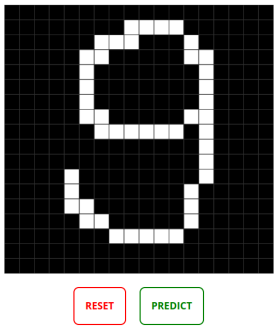

Figura 2: Cuadricula de la aplicación.

Esta es la puerta para el uso de la red neuronal artificial. En esta cuadricula es posible dibujar letras.
Al dibujar una letra en la cuadricula y presionar el botón de “Predeict” la aplicación web realiza una petición al servidor de la REST API de express, la cual recibe el estado de la cuadricula y sus dimensiones.
Posteriormente el servidor de express ejecuta un proceso asincrono de python el cual ejecuta un script que lleva a cabo una transformación de los datos enviados en la petición
Por ultimo los datos transformados de la cuadricula son procesados por la red neuronal previamente entrenada, y devuelve una respuesta al frontend convertida en el caracter que la red neuronal predice (Figura 3).

Figura 3: Respuesta de la red neuronal, también incluye el botón “Reset” para borrar la cuadricula.

Para el desarrollo de la red neuronal artificial no se utilizo ninguna librería, todo código correspondiente al algoritmo de aprendizaje fue escrito desde cero.
Así mismo la red neuronal que procesa la información fue entrenada con un conjunto de datos creado desde cero, para esto se implemento una ruta oculta (figura 4) en la aplicación web en la que es posible utilizar la cuadricula para generar un conjunto de entranamiento.

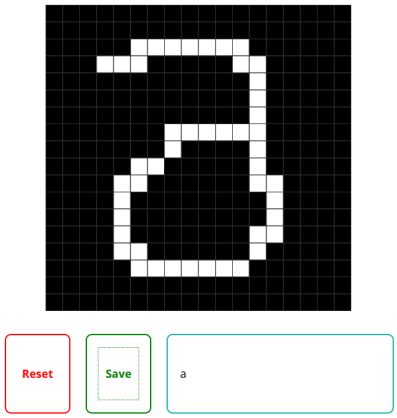

Figura 4: Ruta oculta para la generación de datos de entrenamiento

Los resultados de entrenar la red neuronal con estos datos son que, para las 26 letras del abecedario en ingles por lo menos 21 son predichas correctamente. Las letras que presentan más problemas para ser predichas correctamente son aquellas que comparten cierta similaridad. La red neuronal fue entrenada multiples veces para poder llegar a una configuración de hiperparametros con un mejor rendimiento.

2. Introducción

El problema que se intenta resolver al diseñar este sistema, es el de reconocer texto manuscrito para su digitalización. Se diseña una red neuronal convolucional que es entrenada con un conjunto de datos que contiene pequeñas imagenes de letras del abecedario en ingles. En la sección 1 de este documento, se indica en que consiste el proyecto.
La teoría necesaria para la realización de este proyecto se encuentra en la sección 3
En la sección 4 se revisa la implementación a detalle del sistema al igual que el diseño de la red neuronal. Por utlimo en la sección 5 se incluye una conclusión de lo que fue el proyecto incluyendo una introspección del equipo y trabajo a futuro. Todas la referencias utilizadas para este proyecto se incluyen en la sección de referencias.

3. Marco teórico

Una red neuronal artificial es un modelo matemático y un algoritmo de aprendizage automático inpirado en el funcionamiento de las neuronas biologicas que se encuentran en el cerebro. El elemento básico de una red neuronal es la neurona. Una neurona se compone de cuatro partes fundamentales:

1. **Dendrita**: Son ramificaciónes que conectan con el el cuerpo celular de una neurona, es la parte encargada de transportar los impulsos electricos.
2. **Cuerpo celular**: Es la parte de la neurona encargada de procesar la suma de los impulsos electricos recibidos de las dentritas. Posteriormente, se decide si disparara otra señal electrica de salida por el axon mediante un umbral.
3. **Axon**: Son las ramificaciones finales de la neurona, transportan el impulso de salida de la neurona tras haber sido procesado en el cuerpo celular.
4. **Sinapsis**: El punto de contacto entre el axon de una neurona y la dendrita de otra neurona.

Es la organización de las neuronas y la fuerza de las sinapsis lo que establece la función principal de una neurona. En la figura 5 se puede observar cada una de las partes de la neurona.

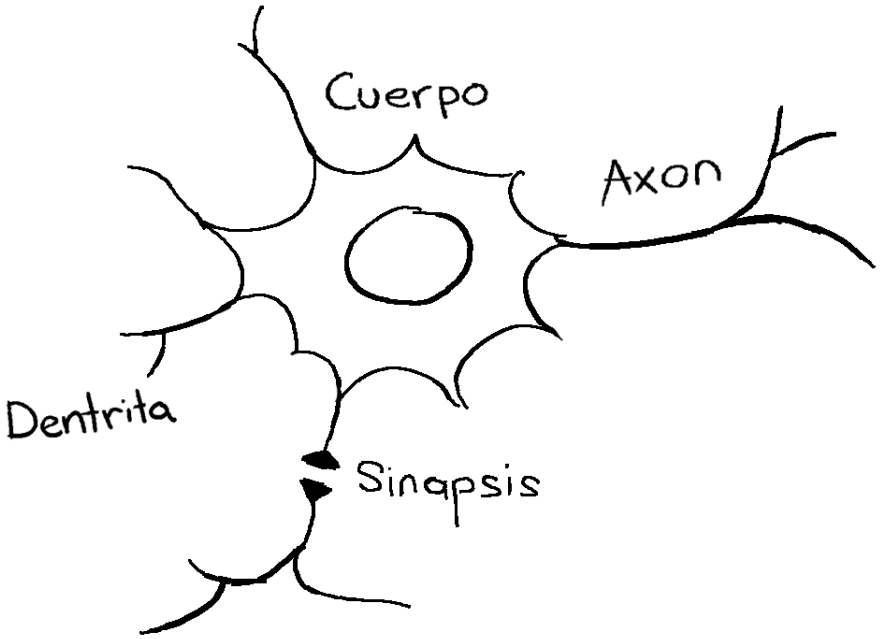

Figura 5: Neurona biologica

Con esto es posible construir un modelo matemático de una neurona artificial (figura 6). En este modelo se tiene un escalar p que corresponde a la señal del impulso electrico recibido de otras neuronas, p es multiplicado por w que corresponde a la fuerza de la sinapsis. Este producto wp es enviado a una sumatoria y se incluye una entrada de sesgo b con un valor de 1. La salida de la sumatoria n va a una función de activación f lo cual produce un escalar de salida a.

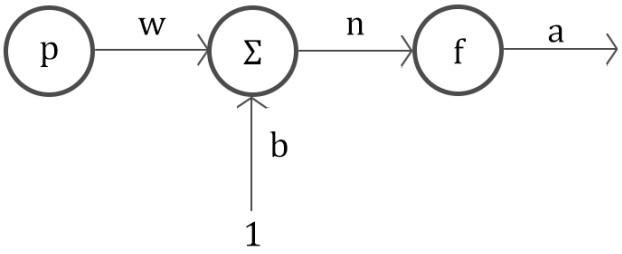

Figura 6: Modelo matemático de neurona artificial.

De esta manera la salida de la neurona es calculada de la forma $a = f(wp + b)$.
La organización de multiples capas y multiples neuronas es lo que le da el poder a este algoritmo. A esto se le conoce como **red neuronal profunda**. Una red neuronal es también conocida como un aproximador de funciones universal, puesto es capaz de aproximar cualquier función teniendo la configuración adecuada.
El modelo matematico de una capa de una red neuronal profunda es, un vector **p** de tamaño R como entrada,un determinado número de neuronas S, seguido de una matriz de pesos **W** con un tamaño $S \times R$ y un vector **b** de sesgo de tamaño $S \times 1$. La operación **Wp + b** es enviada a una sumatoria que resulta en un vector de resultados **n** de tamaño $S \times 1$ y que se dirige a una función de activación f que da una vector **a** de tamaño $S \times 1$ como salida. La salida de una capa es la entrada de la otra. Para definir la arquitectura de una red neuronal se necesita definir una serie de parametros:

1. Número de capas
2. Número de neuronas por capa
3. Tasa de aprendizage
4. Funciones de activación de las neuronas
5. Función de error de la red

A estos se les conoce como **hiperparametros de la red**. Una red neuronal profunda (figura 7) recibe su nombre del hecho que tiene una cantidad considerable de capas intermedias llamadas **capas ocultas**.

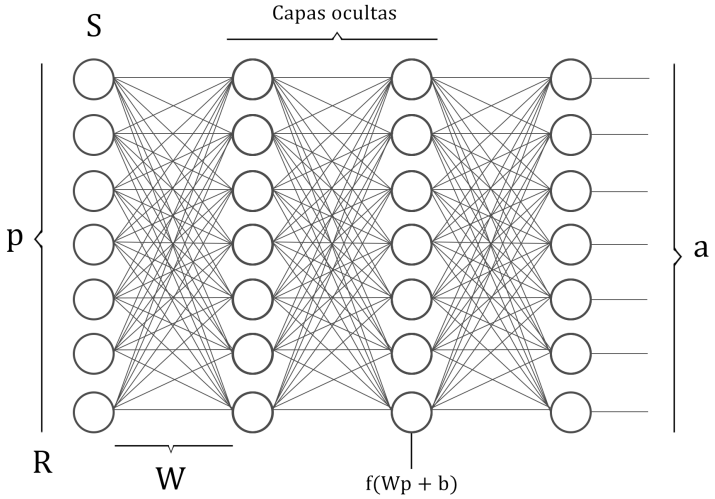

Figura 7: Diagrama representativo de una red neuronal profunda

Al definir el número de neuronas de una capa también se tiene que escoger una función de activación adecuada. Existen distintas funciones de activación que son comúnmente usadas en las redes neuronales. Las tres más comúnes son las siguientes:
Función sigmoidal o logistica: Esta función se encuentra en el rango de 0 a 1. Es muy útil para modelos que requieren predecir una probabilidad.

$$f(x) = \frac{1}{1 + \exp^{-} x} \tag{1}$$

Función tangente hiperbolica: Tambíen es una función logistica pero mejorada, el rango de esta función esta entre -1 a 1. Las ventajas de usar esta función es que penaliza fuertemente las entradas negativas. Es comúnmente usada para la clasificación de dos clases.

$$f(x) = \frac{1 - \exp^{-} 2x}{1 + \exp^{-} 2x} \tag{2}$$

ReLU (Unidad lineal rectificadora): Esta función es la más usada en las capas ocultas de las redes profundas puesto que evita el problema del desvanecimiento del gradiente que presentan las dos funciones anteriores. Esta función tiene un rango de 0 a infinito. Las entradas iguales o menores a cero se hacen cero, por lo que en algunos casos esto impide que se aprenda bien de los datos. Existen modificaciones de esta función que pretenden arreglar este problema como **Leaky ReLU** o **Randomized ReLU**.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x >= 0 \end{cases} \tag{3}$$

Las tres funciones anteriores son facílmente diferenciables lo que las hace adecuadas para el algoritmo de aprendizaje. El algoritmo más utilizado es el de propagación hacía atras o mejor conocido en ingles como **Backpropagation**.
Backpropagation es un algoritmo de aprendizaje que tiene como objetivo minimizar una función de error mediante el uso del **descenso del gradiente**. El entrenamiento de una red neuronal con este algoritmo consta de tres etapas:

1. **Feedfoward**: Esta etapa consiste en propagar las entradas hacia delante en la red para obtener un resultado, este resultado se compara con el resultado esperado de la entrada y se calcula un error.
2. **Backpropagation**: Esta etapa consiste en calcular cuanto cambia el error si se ajustan cada uno de los pesos de la red neuronal. Se calcula una sensibilidad para los pesos derivando parcialmente la función de error respecto a cada peso.
3. **Actualización de pesos**: Usando las sensibilides calculadas en la etapa anterior se realiza una actualización en los valores de los pesos.

Este procedimiento es repetido multiples veces y con distintas entradas, estas son las **iteraciones** del entrenamiento. De esta manera la red es alimentada con suficientes ejemplos como para generalizar correctamente y reducir el error lo mejor posible. Existen distintas architecturas que resuelven distintos tipos de problemas. Dos de las más comunes son:

1. **Red neuronal convolucional**: es usada comúnmente para la clasificación de imagenes. Esta red neuronal consta de una serie de capas de extracción de caracteristicas que utiliza tecnicas de visión computacional. Las salidas de estas capas son conectadas con una serie de capas finales densas (totalmente conectadas) en las que se lleva a cabo el aprendizage.
2. **Red neuronal recurrente**: es usada ampliamente en el campo de **procesamiento de lenguaje natural**, estas redes neuronal reciben su nombre debido a que existe una retroalimentación entre las capas.

Las redes neuronales como algoritmos de aprendizage automatico supervisado, realizan el proceso de aprendizage con conjuntos de datos previamente etiquetados. La calidad de este conjunto de datos refleja la calida de las predicciones de una red neuronal.
Otro de los factores que influyen en el rendimiento de la red neuronal, es la configuración de los hiperparametros. Existen problemas que requieren más o menos neuronas, otros que requieren más capas o menos capas. Es por esto que los hiperparametros de una red neuronal también pueden ser configurados automaticamente con el uso de otros algoritmos tales como **el algoritmo genetico**.

4. Desarrollo

5. Conclusion