Primer avance de proyecto final: clasificador de caracteres japoneses escritos a mano

Adrián Chouza, Raul Hernandez, Jose Jimenez
Aprendizaje de máquina
Profesor Ulises Orozco

Abstract—En este trabajo se presenta el primer avance del problema de clasificar imágenes de caractéres de higarana (japonés) escritos a mano a partir de redes neuronales. Se explican las bases matemáticas de las redes neuronales y la lógica de su funcionamiento. Además se describe la manera en la que será resuelto el problema, desde la investigación del tema hasta el desarrollo de algoritmos. En este primer avance se programa la manera de almacenar los datos en el formato deseado en el lenguaje de programación Python, para ser procesado en un futuro.

Index Terms—Caracteres japoneses, reconocimiento de imagen, Python, redes neuronales, neurona

1 Descripción del problema

En este trabajo se resolverá el problema de reconocimiento de imágenes mediante el uso de redes neuronales. Específicamente, se tomará la tarea de tomar imágenes en blanco y negro de caractéres japoneses de higarana escritos a mano y clasificarlas dependiendo el que representan. Este *dataset* fue tomado de un repositorio en Github [1], formado de 50 figuras distintas con 20 muestras de cada una.

2 Objetivo

Este proyecto tiene como objetivo construir una red neuronal para clasificar un *dataset*. En este caso, clasificar caracteres japoneses del hiragana. Para ello, se explicarán las bases matemáticas de una red neuronal y se aplicaran estos conocimientos a través de la librería `Keras` de Python para leer y procesar las imágenes.

3 Marco teórico

3.1 Neurona artificial

Las redes neuronales son un modelo computacional inspirado por la forma en que las neuronas biológicas procesan información en el cerebro. Las partes importantes de esta son las dendritas, el axón y las terminales. En una neurona artificial, las dendritas son las entradas a procesar, el axón es la función o transformación que se le aplicará a las entradas y las terminal es el resultado. Para explicar la red neuronal es necesario entender cada una de sus partes, llamadas neuronas.

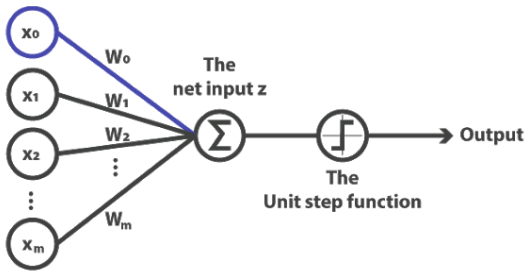

Fig. 1. Neurona artificial. [2]

En la figura 3.1 se pueden observar las similitudes mencionadas entre la neurona artificial y biológica. Esta consiste de entradas x_i y pesos w_i, los cuales son sumados al entrar en la neurona:

$$e = \sum_{i=1}^{m} w_i x_i = w_0 x_0 + w_1 x_1 + ... + w_m x_m$$

Normalmente $x_0 = 1$, representando un cezgo, o bien, el término constante en una regresión lineal como analogía. las entradas x_i representan la salida de otras neuronas o las entradas iniciales de un modelo, mientras que los pesos, la importancia de cada una de las entradas. Este peso, como se explicará más adelante, se modificará para ajustar el modelo. Después, se elige una función de activación f y se calcula la salida de la neurona como $f(e)$. Entre las funciones de activación que existen se encuentran [2]:

- sigmoid: transforma a un valor dentro del rango [0,1]
$$\sigma(x) = \frac{1}{1+e^{-x}}$$
- tanh: transforma a un valor dentro del rango [-1,1]
$$tanh(x) = 2\sigma(2x) - 1$$
- ReLU (Rectified Linear Unit): reemplaza valores negativos por 0
$$f(x) = \max(0, x)$$

Varias neuronas se combinan entre si para crear una red neuronal artificial, las cuales pueden consistir de una sola capa o varias.

3.2 Red Neuronal Artificial

En una red neuronal se utilizan diferentes tipos de neuronas, sin embargo, todas son iguales entre sí, lo que cambia es su posición en la red. Estos tres tipos son:

- Neurona de entrada: estos conectan el mundo con el modelo, introduciendo los datos a la red. No se realiza ninguna transformación en estos datos. Se encuentran en la primera capa de la red.
- Neurona escondida: estos se conectan a los nodos de entrada, otros nodos escondidos o a los nodos de salida y se encuentran en la capa escondida, entre la de entrada y salida. Una red neuronal puede tener varias capas escondidas.
- Neurona de salida: estos se encuentran en la capa de salida, y representan el resultado del modelo. Están conectados solo a los nodos de la última capa escondida y son los responsables de brindar información de vuelta al mundo real.

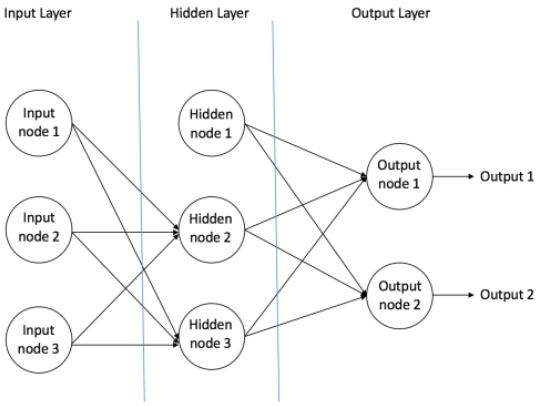

Fig. 2. Red neuronal artificial. [2]

En la figura 2 se puede observar una representación gráfica de una red neuronal y sus diferentes capas. En primera instancia se tiene la capa de entrada y varios nodos. La siguiente capa es una escondida, en las entradas de cada nodo son todos los resultados de cada neurona de la capa anterior. Puede haber cualquier cantidad de capas escondidas y por la existencia de estas ya se considera aprendizaje profundo. Por último se encuentra la capa de salida, conectada a la última capa escondida (o en su ausencia, a las entradas) y realiza la última transformación. Nótese que el primer nodo de cada capa no tiene entradas porque representa el término de cezgo.

3.3 Propagación en regreso

El entrenamiento en una red neuronal consiste en ajustar los pesos w_i hasta que se obtengan resultados deseables. Esto se realiza mediante un método llamado descenso de gradiente. Se utilizará la figura 3 para ejemplificar este algoritmo.

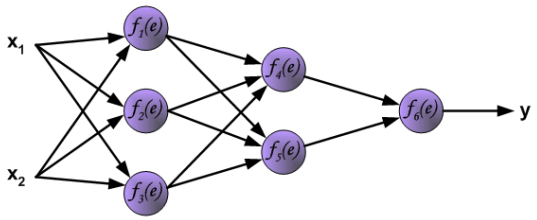

Fig. 3. Red neuronal artificial de tres capas. [3]

Una vez obtenido $\hat{y}$ (el valor estimado del modelo) se obtiene el error $\delta_6 = y - \hat{y}$ y se propaga a todos los nodos de la capa anterior, multiplicándose por el peso. Por ejemplo, $\delta_4 = w_{46}\delta_6$. Es importante aclarar la notación de que w_{xy} significa el peso del nodo x al nodo y de la figura 3 y δ_x el error en el nodo x. De esta forma se tiene el error en cada nodo de la penúltima capa y se vuelve a propagar hacia la que sigue, por ejemplo $\delta_1 = w_{14}\delta_4 + w_{13}\delta_5$. Una vez que se tienen todos los errores en cada nodo, se empiezan a corregir los pesos. Cada nodo se encarga de corregir los pesos que entran a él. Por ejemplo, $w'_{(x_1)1} = w_{(x_1)1} + \eta\delta_1 f'(e)x_1$ donde x_1 representa la entrada del nodo anterior (en este caso, las entradas del mundo externo), w'_{xy} el peso que reemplazará a w_{xy} y η la razón de aprendizaje, que es un valor en el intervalo $(0, 1]$. De esta forma se recalculan todos los pesos, lo cual puede ser hecho mediante multiplicación de matrices.

4 Metodología de la investigación

Se comenzó buscando un *dataset* del alfabeto japonés para utilizarlo como entrada a la red neuronal. Además, se profundizo más en la teoría de estas para tener más claro como se realizará el proceso de entrenamiento y de prueba.

Se investigó como manejar imágenes en el lenguaje de programación Python, así como la utilización de la librería `Keras` para el manejo de redes neuronales. En futuros avances se investigará e implementara el algoritmo necesario para entrenar la red neuronal y ponerla a prueba. Finalmente, se obtendrán y evaluaran resultados.

5 Implementación

Se diseñaron varias funciones con distintos propósitos. Hasta este punto del proyecto se han realizado tres las cuales hacen en orden: remover el nombre del caracter asociado con el archivo para utilizarlo después, convertir un arreglo multi-dimensional en un un arreglo de una sola dimensión agregándolo a una lista que incluye el nombre del carácter y finalmente una función que toma un conjunto de datos y los convierte en dichos listados de bits junto con la etiqueta de carácter. Este proceso se resume en el siguiente algoritmo:

1) Por cada imagen:
2) obtener la etiqueta del nombre del archivo
3) convertir la imagen en un vector de bits
4) agregar la etiqueta y el vector a la matriz de datos
5) Guardar la matriz de datos en un archivo csv.

De esta manera, los *headers* del archivo csv son *label*, p_1, p_2, ..., p_{6972}, donde p_i representa el color del pixel en la posición i y *label* el caracter correspondiente. Al ser una imagen de 84x83 pixeles, son 6972 en total.

REFERENCES

[1] GitHub. (2018). inoueMashuu/hiragana-dataset. [online] Available at: https://github.com/inoueMashuu/hiragana-dataset [Accessed 13 Sep. 2018].
[2] TrisZaska. (2018). Overview about Perceptron. [online] Available at: http://www.triszaska.com/2017/06/overview-about-perceptron.html [Accessed 11 Sep. 2018].
[3] The data science blog. (2018). A Quick Introduction to Neural Networks. [online] Available at: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/ [Accessed 11 Sep. 2018].