Proyecto 4

Inteligencia Artificial

1st Jeffrey Orihuela
Computer Science
UTEC
Lima, Perú
jeffrey.orihuela@utec.edu.pe

2nd Juan Vargas
Computer Science
UTEC
Lima, Perú
juan.vargas@utec.edu.pe

***Index Terms*—Multilayer Perceptron, Machine learning**

Links:https://github.com/amaru0601/multi-layer-perceptron/

I. Introducción

Las redes neuronales artificiales estan inspiradas en el cerebro. No es una comparación perfecta pero existen neuronas, activaciones y mucha interconectividad. Una neurona por sí sola no puede resultar de mucha utilidad pero combinada con varias de ellas forman una estructura potente que resulta ser muy aplicable en muchos métodos de *machine learning* para clasificación de imágenes, video, etc. El presente informe describe el alcance de una red neuronal multi-capa para la detección de imágenes del lenguaje de señas.

II. Explicación

A. Arquitectura

El presente trabajo implementa una capa oculta y una capa de salida. Las imágenes son representadas por un vector de 784 unidades. Debido a que nuestro dataset lleva unas 27455 imágenes representamos nuestra matriz de entrada de dimensión 27455x784. La capa oculta es representada por una matriz de pesos de 60x784 donde la primera dimensión de la matriz representa el número de neuronas y las columnas el número de *features*. Por último, tenemos nuestra matriz de salida de dimensión 25x60 donde el número de filas representa las neuronas de salida y el número de columnas representa la conexión con las neuronas que preceden a esta última capa. Notar que el número 25 se escogió por los 25 tipos de letras que existen en el *dataset*.

B. Funciones de Activación

En nuestro trabajo hemos implementado las siguientes funciones:

1) Función Tangente Hiperbólica:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

2) Función ReLU:

$$ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

C. Función de Costo

La función de costo con la que hemos trabajado es la función sigmoide de esta forma:

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

III. Experimentos

Para el entrenamiento de nuestro modelo utilizamos 10 épocas y trabajamos con 27455 imágenes para el *train set* y 7172 imágenes para el *test set*.

Nuestra primera implementación tuvo la siguiente configuración para 1 capa oculta y 1 capa de salida, ambas con las dimensiones previamente mencionadas para cada una. Se utilizó como función de activación la tangente hiperbólica y se definió el *learning rate* de 0.01 logrando un resultado de 90% de precisión para el *train set* y 71% de precisión para el *test set*.

Nuestra segunda implementación tuvo la misma configuración que la anterior con la diferencia de aumentar el *learning rate* a 0.1 donde se obtuvo un un resultado de 98% de precisión para el *train set* y 75% de precisión para el *test set* mejorando la precisión del modelo.

Nuestra tercera experimentación tuvo la misma configuración que la segunda implementación con la diferencia de cambiar la función de activación de tangente hiperbólica a una ReLU. Los resultados que obtuvimos fue de 98% de precisión para el *train set* y 75% de precisión para el *test set*.

Por último, modificamos nuestra configuración con 3 capas ocultas y la función de activación ReLU para cada una de ellas, el *learning rate* se mantuvo en 0.1 obtuviendo un 99% de precisión para el *train set* y 73% de precisión para el *test set*.

IV. Conclusiones

- El modelo logra una precisión del 99% con el *train set*
- Nuestro modelo implementó una función sigmoide para el cálculo del costo en la última capa, no demostró precisión con los datos de prueba
- Superar la precisión del 75% en el *test set* requiere operaciones convolucionales, debido a que la imágen de manos contiene mucha información e induce al *overfitting* para nuestro modelo.