# Implementación de un perceptrón multicapa para calsificación de tumores*

Mauricio Pinto Larrea
*(dept. Computer Science)*
*UTEC*
Lima, Peru
mauricio.pinto@utec.edu.pe

Francesco Ucelli Meneses
*(dept. Computer Science)*
*UTEC*
Lima, Peru
francesco.uccelli@utec.edu.pe

Juan Manuel Navarro Nieto
*(dept. Computer Science)*
*UTEC*
Lima, Peru
juan.navarro@utec.edu.pe

## I. Introducción

Para este proyecto, se implementó una red neuronal Multilayer Perceptron (MLP) con el objetivo de clasificar tipos de tumores en una Base de Datos de la Universidad de Wisconsin. Este dataset contiene información de un grupo de mujeres que presentan un tumor en la mama tanto benigno como maligno. El objetivo en si es implementar un MLP en lenguaje c++ que pueda predecir si un nuevo paciente presenta un tumor maligno o no, además de realizar pruebas con diferentes parámetros y funciones de activación. Para esta implementación nos basamos en la explicación dada en [1].

## II. Explicación

### A. Arquitectura de la Red

Un MLP es una red neuronal artificial profunda compuesta por mas de un perceptron. Esta compuesta por una capa de entrada que recibe las señales del dataframe, una de salida que toma una decisión o predicción sobre la entrada, y entre ambas un numero arbitrario de capas ocultas que son el motor computacional. Un MLP con capas ocultas tiene la capacidad de aproximarse a cualquier función continua. Para este caso, tenemos la opción de definir qué estructura tendrá nuestra red en tanto los parámetros de cada capa como el numero de capas.

Los MLPs se aplican usualmente a problemas de aprendizaje supervisados. Se entrenan en un conjunto de pares de entrada-salida y aprenden a modelar la correlación entre esas. El entrenamiento implica ajustar los parámetros (weights y biases) del modelo para minimizar el error. En este caso, hemos definido la red de tal manera que recibe tantas columnas como hay en el dataset (una observación) y devuelve dos valores que pasan a representar probabilidades de que esa observación corresponda a un tumor benigno(0) o maligno(1). La arquitectura presentada en la imagen y descrita a continuación es arbitraría, el objetivo principal de este trabajo es experimentar con las características de la red para entender como adaptar esta mejor a la tarea dada. Por esto, constantemente se han ido modificando parámetros como el número de capas, el numero de épocas, los valores de input y output de cada capa, etc.

### B. Backpropagation

Con el objetivo de minimizar el error que genera la red al ser entrenada, se usa el algoritmo de backpropagation que consiste básicamente en utilizar métodos que direccionen la red hacia el mínimo error a lo largo de un número determinado de iteraciones. Consiste en lo siguiente: Primero, se ”encasilla” la capa de cálculo de error y de esta se obtiene un primer valor de ”sensibilidad”. Este valor de sensibilidad, que define cuanto cambian los pesos de la red, se calcula a continuación recursivamente en función del mismo valor de la capa anterior (se recorre de atrás hacia adelante). Finalmente, se calcula la gradiente del error (para orientarnos hacia el mínimo) y se actualizan los pesos en función de este y de las sensibilidades.

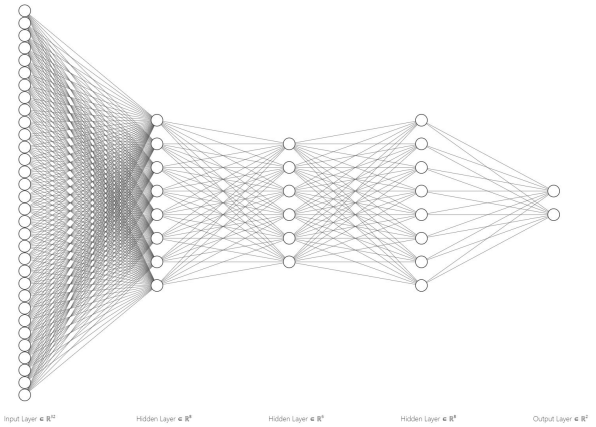

Fig. 1. Esquema de arquitectura de la red [3]

Al final de este proceso, tenemos una red que se actualiza cada vez que calcula un error, de esta manera por cada iteración este error se va reduciendo y se consiguen buenos resultados.

### C. Funciones de activación y pérdida

Para las funciones de activacion, utilizaremos las siguientes funciones:

- Sigmoide: una función real diferenciable, acotada que se define para todos los valores de entrada reales y tiene una derivada no negativa en cada punto y exactamente un punto de inflexión. Tiene la forma:

$$S(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

- Tanh: la función tangente hiperbólica, que es el análogo hiperbólico de la función circular tangente utilizada en trigonometría. Tiene la forma:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2}$$

- ReLU: una función lineal por partes que generará la entrada directamente si es positiva; de lo contrario, generará cero. Tiene la forma:

$$relu(x) = max(0, x) \tag{3}$$

En el caso de las funciones de perdida utilizaremos Softmax, la cual es una generalización de la función logística a múltiples dimensiones. A esta le añadiremos Cross Entropy la cual mide el rendimiento de un modelo de clasificación cuya salida es un valor de score que nos indica el desempeño de la red.

- Softmax : softmax es un método para convertir los valores de output de la red, que son números "cualquiera" en su representación a probabilidades. Es de la forma:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{4}$$

  Donde una x corresponde a una observación y la sumatoria del denominador al término normalizador.
- Cross entropy loss : Cross entropy loss es la función de error más básica para clasificación binaria con clases mutuamente excluyentes, consiste en diferenciar las probabilidades predecidas de las etiquetas reales de manera eficiente. Tiene la forma:

$$-(y \log(p) + (1 - y) \log(1 - p)) \tag{5}$$

  Donde p corresponde una sola predicción y "y" al valor real de la etiqueta.

## III. Dataset

El dataset proveído para el proyecto es de una base de datos de la Universidad de Wisconsin. Como mencionamos previamente, este contiene imágenes de un grupo de mujeres que presentan un tumor en la mama, dándole como tarea a nuestro MLP identificar si el tumor es maligno o benigno.

Este dataset fue luego preprocesado por nosotros, cambiando los valores de malignos a "1" y benignos a "0" a manera de tener un datatype estandarizado sobre esta. Finalmente, los training y testing sets fueron divididos utilizando 70 y 30 por ciento de la data respectivamente.

## IV. Experimentación

Para la experimentación, se utilizó el dataset proporcionado con información sobre tumores benignos y malignos. Se dividió la data en un 70% para entrenamiento y un 30% para hacer pruebas. Se utilizó una red de 5 capas, 3 capas ocultas con números variables de neuronas. Se realizaron pruebas con cada una de las funciones de activación previamente mencionadas. A continuación, observamos la variación del error con cada una:

1) Sigmoid:

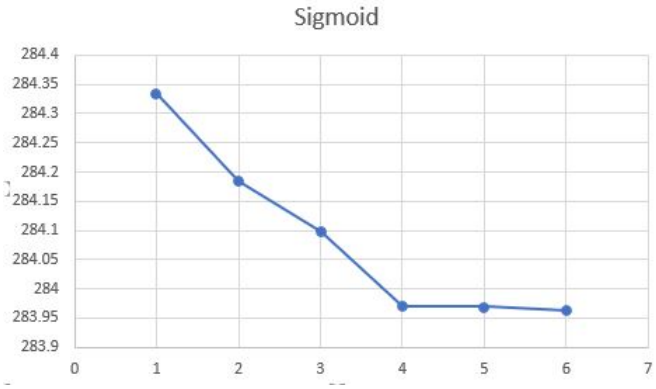


Fig. 2. Variación del error con la función Sigmoid

2) TanH:

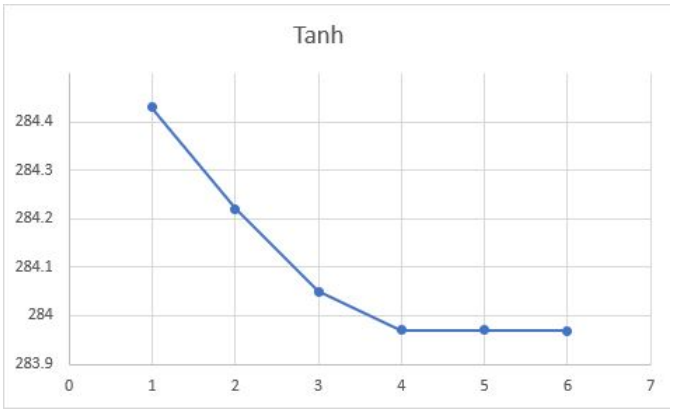


Fig. 3. Variación del error con la función TanH

3) ReLU: No se pudo obtener valores.

## V. Conclusiones

Una de las posibles mejoras que se han notado durante el desarrollo del trabajo está relacionada con el uso de optimizadores de descenso de gradiente. Utilizar uno de estos métodos en un futuro podrían ayudar a conseguir un mejor score y hacer que la red se adapte más rápido. Por otro lado, en términos de implementación, fue complicado cumplir con todas las optimizaciones para ir mejorando el desempeño en el tiempo dado. Esta implementación puede ser vista como una base eficiente sobre la cual se pueden construir optimizaciones para obtener resultados prominentes manteniendo buenos tiempos de ejecución y complejidad de memoria.

## VI. Referencias


1 Sathyanarayana, Shashi. (2014). A Gentle Introduction to Backpropagation. Numeric Insight, Inc Whitepaper.
2 https://scriptreference.com/neural-networks-from-scratch/neural-network-gradient-descent
3 http://alexlenail.me/NN-SVG/index.html
4 https://patrickhoo.wixsite.com/diveindatascience/single-post/2019/06/13/activation-functions-and-when-to-use-them

5 https://deepai.org/machine-learning-glossary-and-terms/softmax-layer