Proyecto 3: Redes Neuronales

1st Noche, Claudia
Ciencia de la Computación
Universidad de Tecnología e Ingeniería
claudia.noche@utec.edu.pe
Participación: 100%

2nd Rincón, Yamileth
Ciencia de la Computación
Universidad de Tecnología e Ingeniería
yamileth.rincon@utec.edu.pe
Participación: 100%

3rd Quispe, Sebastian
Ciencia de la Computación
Universidad de Tecnología e Ingeniería
sebastian.quispe.b@utec.edu.pe
Participación: 100%

4th Flores, Carlos
Ciencia de la Computación
Universidad de Tecnología e Ingeniería
carlos.flores.p@utec.edu.pe
Participación: 100%

I. Repo

Link al repositorio en GitHub

II. Introducción

En el ámbito de la inteligencia artificial, la aplicación de algoritmos de aprendizaje automático ha surgido como un propulsor de innovaciones significativas. Entre las diversas metodologías para abordar problemáticas relacionadas con la inteligencia artificial, el Multilayer Perceptron (MLP) destaca como una herramienta fundamental. Este modelo, fundamentado en la simulación del funcionamiento cerebral humano, aspira a reproducir un proceso de aprendizaje basado en neuronas artificiales interconectadas.

Este proyecto se propone explorar el potencial inherente de la aplicación del MLP, sobre una base de datos que almacena información derivada de imágenes mamarias de un conjunto de mujeres. En este dataset, se registran casos de tumores mamarios malignos y benignos.

La finalidad principal de este proyecto radica en la implementación de una MLP con el propósito de clasificar de manera efectiva esta información y anticipar con precisión la naturaleza del tumor presente.

III. Dataset

La base de datos objeto de estudio alberga información detallada sobre 569 pacientes, de los cuales 357 han sido diagnosticados con tumores benignos, mientras que los restantes 212 presentan tumores malignos en la mama. La estructura de la base de datos consta de diversas columnas, donde la primera corresponde al identificador único del paciente, la segunda a una variable categórica representada por la letra 'M' para maligno y 'B' para benigno, y las siguientes columnas contienen valores numéricos que representan mediciones y características específicas del tumor o quiste.

La representación detallada de la diversidad de diagnósticos y las características asociadas a cada paciente proporciona una base sólida para explorar la capacidad de estos algoritmos en la clasificación precisa de tumores mamarios.

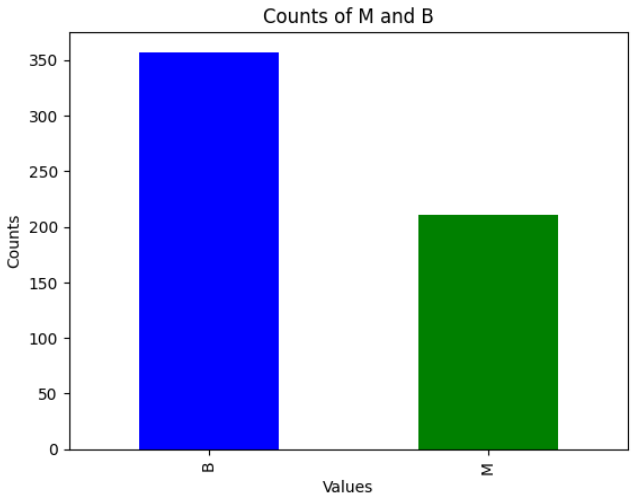

Fig. 1: Distribución de la Data

IV. Explicación de los Modelos

A. Multi-layer Perceptron

Es un modelo de aprendizaje profundo, se destaca por su capacidad para aprender representaciones complejas de datos a través de capas ocultas. La estructura está compuesta de una capa de entrada, una o más capas ocultas y una capa de salida, como se puede observar en la figura 2, donde cada una realiza operaciones específicas.

Cada neurona del MLP realiza una suma ponderada de sus entradas, seguida de la aplicación de una función de activación no lineal. Esta operación se expresa matemáticamente como $a = f(\sum (w_i * x_i) + b)$, donde w_i son los pesos, x_i son las entradas, b es el sesgo (bias) y f la función de activación.

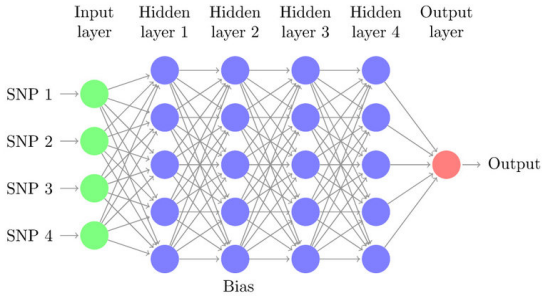

Fig. 2: Estructura general de MLP [4]

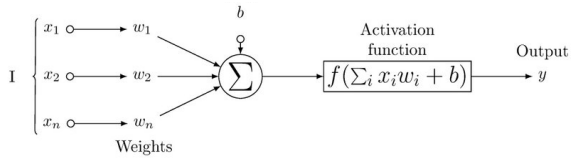

Fig. 3: Funcionamiento de las neuronas [4]

En el proceso de entrenamiento del MLP implica la retropropagación del error a través de la red para ajustar los pesos y los sesgos. En la figura 14 muestra este proceso, destacando la comparación de la salida de nuestra red neuronal con el objetivo, dónde por consiguiente se ajustan los pesos en función al error.

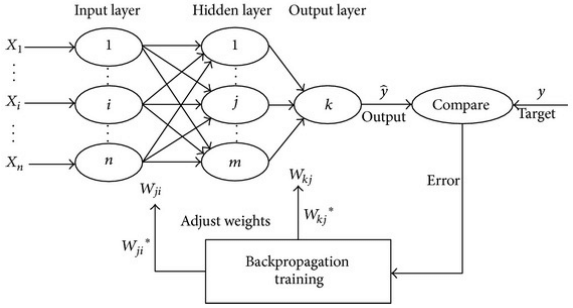

Fig. 4: Técnica de retropropagación [5]

B. Softmax Function

La función softmax es una operación matemática, comúnmente utilizada en la capa de salida de una red neuronal, en nuestro caso el MLP. Convierte un vector de puntuaciones brutas o logits en probabilidades, donde cada puntuación representa la probabilidad de pertenencia a la clase correspondiente. Dónde, dado un vector de puntuaciones brutas o logits $z = [z_1, z_2, ..., z_n]$, la función se define de la siguiente manera:

Aquí, e es la base del logaritmo natural, y n es el número de clases. La función softmax exponencia cada puntuación bruta y la normaliza por la suma de las puntuaciones exponenciadas, asegurando que las probabilidades resultantes sumen 1.

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Fig. 5: Función Softmax

C. Activation functions

Un concepto importante en el aprendizaje automático son las funciones de activación, las cuales cobran mayor importancia en temas de Deep Learning. Estas funciones determinan si una neurona debe activarse o no e introducen una transformación no lineal a una red neuronal. Tienen como propósito principal convertir una señal de entrada de una neurona y producir una salida para alimentar a la siguiente neurona en la capa siguiente. Hay principalmente 4 funciones principales que se van a explorar en este proyecto; Sigmoid, Tanh, ReLU y Softmax.

1) ***Sigmoid****:* La función Sigmoid, también conocida como función logística, transforma cualquier entrada real en un rango entre 0 y 1. Su forma de S curva la hace útil para problemas de clasificación binaria, donde se busca predecir una salida de dos posibles categorías.

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

Fig. 6: Función de Sigmoid [1]

Ventajas:

1) No lineal.
2) Activación analógica, a diferencia de una función de paso.
3) Gradiente suave.
4) Apropiada para clasificación.
5) Salida siempre en rango (0,1), evitando problemas de explosión de activación.

Desventajas:

1) Hacia los extremos, la función tiene poca respuesta a cambios en X.
2) Problema de ”vanishing gradients”.
3) Salida no centrada en cero, dificultando optimización.
4) Saturación, que puede anular gradientes.
5) Puede llevar a que la red aprenda lentamente o se detenga, dependiendo del caso y hasta que los límites de valores de punto flotante afecten cálculos de gradientes.

2) ***Tanh****:* Se considera una función mejorada de la función Sigmoid. La función tangente hiperbólica, o Tanh, tiene un rango de valores de -1 a 1. Se utiliza principalmente para la clasificación entre dos clases.

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e6{-}2x}$$

Fig. 7: Función de Tanh [1]

Ventajas:

1) El gradiente es más fuerte para tanh que para la sigmoidal (las derivadas son más pronunciadas).

Desventajas:

1) Problema de ”vanishing gradients”.

*3) **ReLU**:* El uso más extendido en el aprendizaje profundo es la función de activación ReLU (Rectified Linear Unit). Es no lineal, lo que facilita la retropropagación de errores y permite múltiples capas de neuronas activadas por la función ReLU.

$$R(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases}$$

Fig. 8: Función de ReLU [1]

Ventajas:

1) Evita y corrige el problema del ”vanishing gradient”.
2) Menos costosa computacionalmente que tanh y sigmoid, ya que involucra operaciones matemáticas más simples.
3) No activa todas las neuronas al mismo tiempo, lo que hace que la red sea eficiente y fácil de computar.

Desventajas:

1) Limitada a capas ocultas de un modelo de red neuronal.
2) Algunos gradientes pueden ser frágiles durante el entrenamiento y ”morir”, lo que puede resultar en neuronas inactivas.
3) En la región (x¡0) de ReLU, el gradiente es 0, lo que puede llevar a ”neuronas muertas” que no responden a variaciones en el error/entrada.
4) El rango de ReLU es [0, inf), lo que puede hacer que la activación se dispare, presentando el riesgo de explosión de activación.

V. Implementación

1) Funciones de Activación:
   Sigmoide: `sigmoide`
   Tangente hiperbólica (tanh): `tanh`
   ReLU: `relu`
   Capa de salida: `softmax`

2) Inicialización de la Red: Parámetros como número de características, capas ocultas, neuronas por capa y función de activación.
3) Pesos Iniciales: Pesos aleatorios para las capas ocultas y de salida.
4) Método de Entrenamiento `train`: Retropropagación durante un número especificado de épocas.
5) Actualización de Pesos: Utiliza el gradiente descendente con un parámetro de aprendizaje `alpha`.
6) Propagación Hacia Adelante `forward`: Calcula las salidas de cada capa oculta y de la capa de salida.
7) Backward Propagation `backward`: Calcula los gradientes y ajusta los pesos mediante *Backward Propagation*.
8) Función de Pérdida `calculate_loss`): Utiliza el error cuadrático medio (MSE) para evaluar la pérdida.
9) Predicciones `predict` Realiza predicciones basadas en la red entrenada.
10) Visualización de Pérdida: Método textttplot_loss_curve para graficar la curva de pérdida durante el entrenamiento.

VI. Resultados

Se llevaron a cabo experimentos utilizando una implementación en Python de una Red Neuronal Multicapa (MLP), empleando diversas funciones de activación. Simultáneamente, se generó una representación gráfica de los resultados de la función de pérdida (loss) utilizando MSE. Cada experimento consistió en **500 épocas**, con **2 capas ocultas** y **1 capa de salida**.

Sigmoid

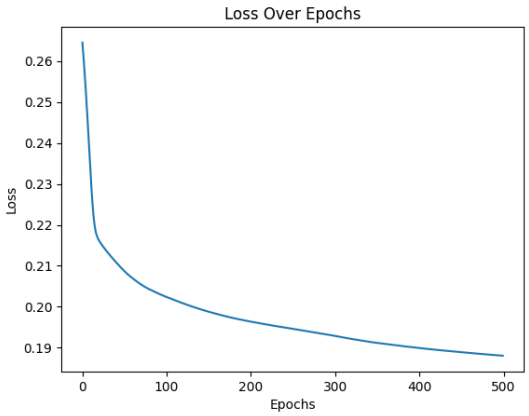

Fig. 9: Loss con Sigmoid

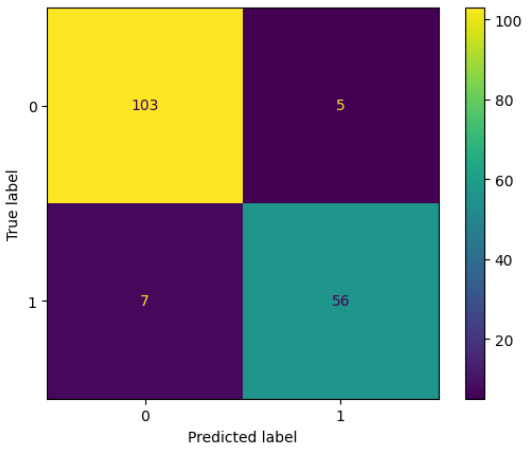

Fig. 10: Matriz de confusión con Sigmoid

- **Pérdida Final:** 0.1847, epoch 500

- **Matriz de Confusión:**

$$\begin{bmatrix} 103 & 5 \\ 7 & 56 \end{bmatrix}$$

- **Comentarios:**
  - La pérdida obtenida indica un rendimiento moderadamente bueno en la tarea.
  - La matriz de confusión sugiere un número relativamente bajo de falsos positivos y falsos negativos, indicando un equilibrio en la clasificación.

Tanh

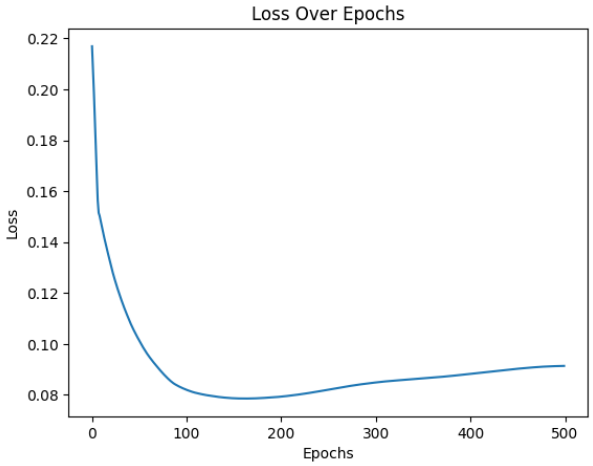

Fig. 11: Loss con Tanh

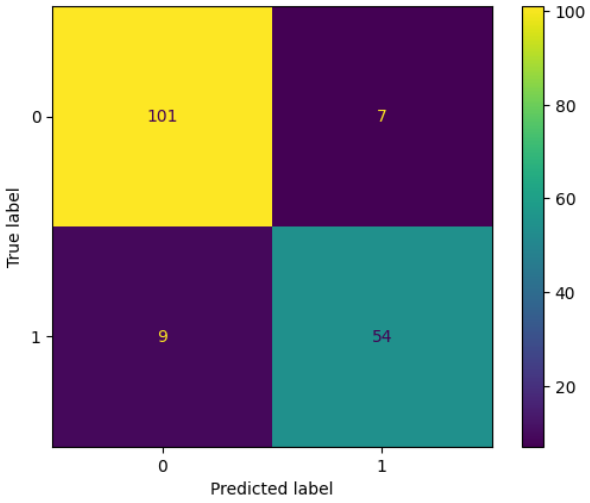

Fig. 12: Matriz de confusión con Tanh

- **Pérdida Final:** Aumenta después de la época 118, alcanzando 0.10
- **Matriz de Confusión:**

$$\begin{bmatrix} 101 & 7 \\ 9 & 54 \end{bmatrix}$$

- **Comentarios:**
  - El aumento de la pérdida después de la época 118 podría indicar posiblemente un sobreajuste o un problema en la convergencia.
  - La matriz de confusión sugiere un aumento en falsos positivos y falsos negativos después de esa época, lo que respalda la observación de la pérdida.

ReLU

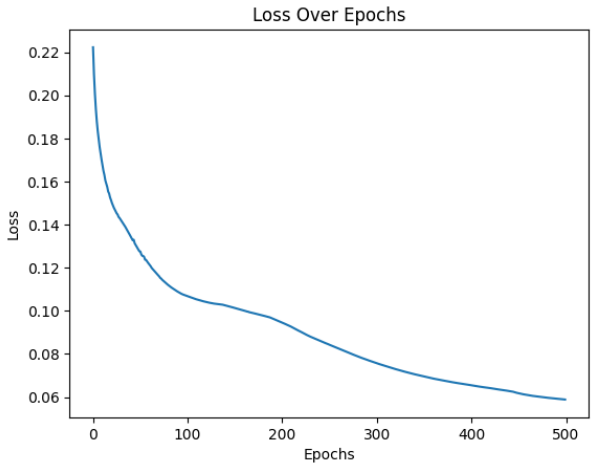

Fig. 13: Loss con ReLU

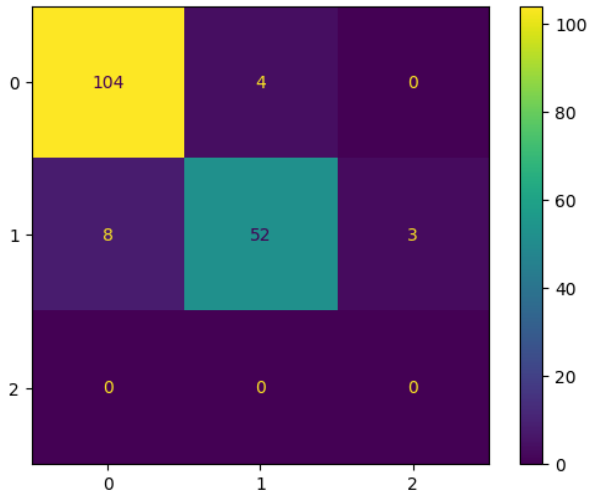

Fig. 14: Matriz de confusión con ReLU

- **Pérdida Final:** 0.0615, epoch 500
- **Matriz de Confusión:**

$$\begin{bmatrix} 104 & 4 \\ 8 & 52 \end{bmatrix}$$

- **Comentarios:**
  - La pérdida más baja indica un rendimiento sólido en la tarea de regresión.
  - La matriz de confusión sugiere un menor número de falsos positivos y falsos negativos en comparación

con Sigmoid y Tanh, indicando una mejor capacidad de clasificación.

VII. Conclusiones

En resumen, los resultados indican que la función ReLU exhibe el mejor rendimiento para la clasificación en esta base de datos. Esta función ofrece la pérdida más baja y consistente, junto con una matriz de confusión que refleja una mayor capacidad para clasificar de manera precisa. Aunque Sigmoid presenta un rendimiento moderado, su pérdida es ligeramente más alta en comparación con ReLU. Tanh muestra un comportamiento peculiar con un aumento en la pérdida después de cierto punto, indicando posibles problemas de convergencia o sobreajuste.

Estos resultados pueden atribuirse tanto a la implementación específica de cada función de activación como a la idoneidad de cada función para una base de datos diversa como la evaluada.

Como trabajo a futuro, se propone la prueba de la implementación sobre distintas bases de datos con el fin de comprobar cuál es el rendimiento de cada red neuronal en distintas situaciones.

References

[1] (NN), Activation functions and when to use them. Dive in Data Science.
[2] Wood T. (NN), Softmax Function. DeppAI org.
[3] scikit-learn Org, Neural network models (supervised)
[4] A Guide for Using Deep Learning for Complex Trait Genomic Prediction - Scientific Figure on ResearchGate.
[5] Presenting a new method to improve the detection of micro-seismic events - Scientific Figure on ResearchGate.