Introducción

El aprendizaje supervisado es una de las ramas más importantes del aprendizaje automático, y dentro de ella, la red neuronal con backpropagation es una de las técnicas más utilizadas y poderosas para abordar una amplia variedad de problemas de clasificación y regresión.

En este trabajo, exploraremos la implementación de una red neuronal con backpropagation en el lenguaje de programación Python. Este enfoque nos permitirá comprender en profundidad cómo funciona este algoritmo fundamental de aprendizaje automático, desde los conceptos teóricos hasta la aplicación práctica.

El objetivo principal de este trabajo es proporcionar una visión general del proceso de codificación de una red neuronal con backpropagation en Python. Comenzaremos explicando los fundamentos teóricos detrás de la red neuronal y el algoritmo de backpropagation, incluyendo conceptos como las capas de la red, las funciones de activación, la propagación hacia adelante y hacia atrás, y la actualización de los pesos y sesgos.

Nos sumergiremos en la implementación práctica, utilizando la biblioteca NumPy para la manipulación eficiente de matrices y vectores. Veremos cómo crear una clase de red neuronal que pueda entrenarse utilizando el algoritmo de backpropagation en conjuntos de datos de entrenamiento, y cómo utilizar esta implementación para realizar predicciones en nuevos datos.

Este trabajo proporcionará una base sólida para comprender y codificar redes neuronales con backpropagation en Python, lo que permitirá a los lectores desarrollar su experiencia en aprendizaje automático y aplicar estos conocimientos a una variedad de problemas del mundo real.

Objetivos

- Desarrollar una aplicación basada en aprendizaje supervisado para resolver problemas sencillos mediante backpropagation.

- Permitir la carga previa de conjuntos de patrones de entrada y salida desde archivos para su procesamiento.

- Implementar una red neuronal capaz de procesar cualquier cantidad de entradas y salidas, lo que garantiza la flexibilidad y la adaptabilidad de la aplicación.

- Incluir la funcionalidad de generar y visualizar la gráfica del error de la iteración durante el proceso de simulación, facilitando el análisis y la optimización del entrenamiento de la red.

Análisis del problema

La problemática consiste en desarrollar una aplicación que utilice técnicas de aprendizaje supervisado backpropagation para resolver problemas simples. La aplicación debe ser capaz de cargar conjuntos de datos previamente almacenados en archivos y utilizarlos para entrenar y simular una red neuronal artificial. La red neuronal debe ser capaz de procesar cualquier cantidad de entradas y salidas, y durante el proceso de simulación, la aplicación debe mostrar una gráfica que represente el error de la iteración para facilitar el análisis y seguimiento del proceso de entrenamiento.

Requerimientos Funcionales:

- **Carga de Datos desde Archivos:** La aplicación debe permitir cargar conjuntos de datos de entrenamiento desde archivos previamente almacenados en el sistema. Estos archivos contendrán patrones de entrada y salida etiquetados que se utilizarán para entrenar la red neuronal.

- **Establecer los parámetros de entrenamiento:** La aplicación debe permitir establecer los parámetros de entrenamiento. Los parámetros clave a configurar en el proceso de entrenamiento son:

    - ***Número de Iteraciones:*** Representa la cantidad de veces que el conjunto de entrenamiento completo se procesa durante el entrenamiento. Un número suficiente de iteraciones garantiza que la red neuronal tenga la oportunidad de converger hacia una solución óptima. Sin embargo, un número excesivo de iteraciones puede llevar al sobreajuste del modelo a los datos de entrenamiento.

    - ***Tasa de Aprendizaje:*** Controla la magnitud de los ajustes realizados en los pesos de la red durante cada paso de entrenamiento. Una tasa de aprendizaje más alta puede acelerar la convergencia del modelo, pero también puede provocar oscilaciones o divergencias. Por otro lado, una tasa de aprendizaje más baja puede garantizar una convergencia más estable, pero el entrenamiento puede ser más lento.

    - ***Tolerancia al Error:*** Especifica la precisión deseada o la cantidad aceptable de error que se puede tolerar durante el entrenamiento. Durante el proceso de entrenamiento, la red intentará minimizar este error, ajustando los pesos y los sesgos de las neuronas para reducir la discrepancia entre las predicciones y las salidas reales. Una vez que el error alcanza o se acerca al valor de tolerancia especificado, el entrenamiento puede detenerse, ya que es poco probable que la red mejore significativamente más allá de este punto.

- **Diseño de la red neuronal:** La aplicación hará uso de una red neuronal de aprendizaje supervisado backpropagation. Una red neuronal con backpropagation es un tipo de red neuronal artificial que utiliza el algoritmo de retropropagación (backpropagation) para entrenarse. Es una de las técnicas más comunes y poderosas en el campo del aprendizaje automático supervisado, particularmente en problemas de clasificación y regresión.

  - **Backpropagation**: Es un algoritmo de aprendizaje que se utiliza para entrenar redes neuronales mediante el cálculo del gradiente de la función de pérdida con respecto a los pesos de la red, y luego ajustando los pesos en la dirección opuesta al gradiente para minimizar la pérdida. Este proceso se realiza en dos pasos:

    a) ***Propagación hacia adelante:*** Los datos de entrada se propagan a través de la red capa por capa, y se calculan las salidas de cada neurona utilizando una función de activación. La salida final se compara con las salidas reales para calcular la pérdida.

    b) ***Retropropagación***: Se calculan los gradientes de la función de pérdida con respecto a los pesos de la red utilizando el algoritmo de la cadena de reglas. Luego, estos gradientes se utilizan para ajustar los pesos de la red mediante métodos de optimización como el descenso de gradiente.

  - **Funciones de Activación*:*** Son funciones matemáticas que se aplican a la salida de cada neurona en una red neuronal. Introducen no linealidades en la red, lo que permite que la red aprenda patrones complejos en los datos. Para el diseño de nuestra red utilizaremos las siguientes funciones de activación:

    a) ***Función de activación Sigmoide****:* La función sigmoide es una función logística que toma un valor real y lo "aplana" en el rango de 0 a 1. Se define como:

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

    Esta función es útil en problemas de clasificación binaria, donde la salida de la red necesita ser mapeada a una probabilidad.

    b) ***Función de activación Tangente Hiperbólica (Tanh):*** La función tangente hiperbólica es similar a la función sigmoide, pero mapea los valores en un rango entre -1 y 1. Se define como:

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

    Es comúnmente utilizada en redes neuronales debido a su naturaleza simétrica y su capacidad para manejar datos con valores negativos.

c) ***Función de activación Seno:*** La función seno mapea los valores de entrada a los valores del seno del ángulo correspondiente. Se define como:

$$sen(x)$$

Aunque menos común en redes neuronales, puede ser útil en casos donde la periodicidad de los datos es importante.

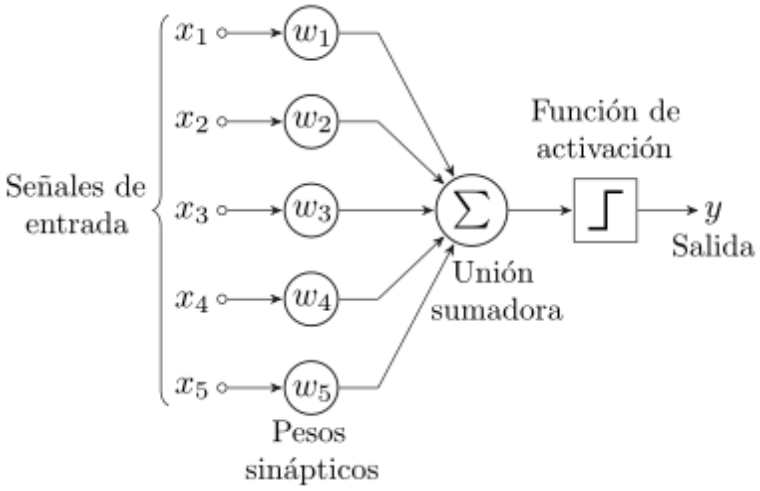
d) ***Función de activación Lineal:*** La función lineal simplemente devuelve el mismo valor de entrada sin modificarlo. Se define como:

$$f(x) = x$$

Aunque es la función más simple, se utiliza en ciertas situaciones, como en la capa de salida de una red para problemas de regresión, donde la salida deseada no tiene restricciones en el rango de valores.

Estas funciones de activación son componentes importantes en el diseño y entrenamiento de redes neuronales, ya que introducen no linealidades en la red, permitiendo que la red aprenda y represente relaciones complejas en los datos. La elección de la función de activación adecuada depende del problema específico y del comportamiento deseado de la red.

Una red neuronal con backpropagation es un modelo de aprendizaje automático supervisado que utiliza un algoritmo de retropropagación para entrenar una red neuronal artificial ajustando iterativamente los pesos de la red para minimizar la pérdida entre las salidas predichas y las salidas reales en un conjunto de datos de entrenamiento.



- **Entrenamiento de la Red Neuronal:** Una vez que los datos de entrenamiento se cargan en la aplicación, se debe permitir entrenar una red neuronal utilizando estos datos. El proceso de entrenamiento se basará en técnicas de aprendizaje supervisado, donde la red aprenderá a partir de ejemplos etiquetados, minimizando el error entre las salidas predichas y las salidas reales.

  - **Inicialización de la Red Neuronal:** Define la arquitectura de la red neuronal, incluyendo el número de capas, el número de neuronas en cada capa y las funciones de activación. Inicializa los pesos y sesgos de la red de manera aleatoria o utilizando algún método específico.

  - **Propagación hacia Adelante (Forward Propagation):** Para cada ejemplo de entrenamiento en el conjunto de datos:

    a) Propaga los datos de entrada hacia adelante a través de la red neuronal.
    b) Calcula las salidas de cada neurona en la red utilizando las funciones de activación.
    c) Calcula la función de pérdida entre las salidas predichas y las salidas reales.

  - **Retropropagación del Error (Backpropagation):** Calcula el gradiente de la función de pérdida con respecto a los pesos de la red utilizando el algoritmo de backpropagation:

    a) Calcula el gradiente de la función de pérdida con respecto a las salidas de la última capa de la red.
    b) Propaga hacia atrás el gradiente a través de la red, calculando los gradientes de la función de pérdida con respecto a las salidas de cada capa y los pesos y sesgos de cada neurona.
    c) Utiliza los gradientes calculados para actualizar los pesos y sesgos de la red utilizando un algoritmo de optimización, como el descenso de gradiente.

  - **Actualización de los Pesos y Sesgos:** Utiliza el algoritmo de optimización y los gradientes calculados para actualizar los pesos y sesgos de la red neuronal. Esto ajusta los parámetros de la red para reducir la función de pérdida en el conjunto de datos de entrenamiento.

  - **Repetición del Proceso:** Repite los pasos 2 a 4 para un número específico de épocas o hasta que se alcance algún criterio de convergencia, como una pérdida mínima o una mejora mínima en la pérdida entre épocas consecutivas.

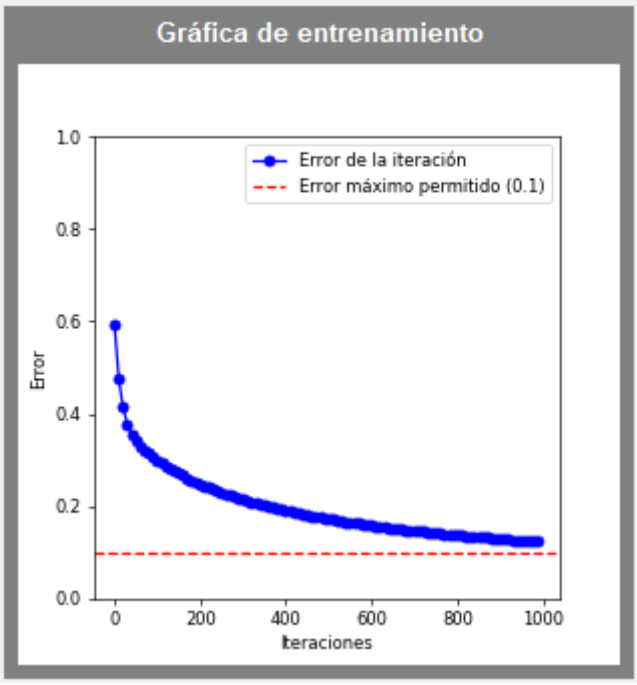
  - **Validación y Evaluación:** Después del entrenamiento, evalúa el rendimiento de la red neuronal utilizando un conjunto de datos de validación o prueba para verificar su capacidad de generalización.

- **Simulación de la Red Neuronal:** Después de entrenar la red neuronal, la aplicación debe permitir simular su funcionamiento utilizando nuevos conjuntos de datos. Durante la simulación, la red deberá procesar las entradas proporcionadas y generar salidas predichas. Estas predicciones se compararán con las salidas reales para evaluar el rendimiento de la red.

- **Gráfica del Error de la Iteración:** Durante el proceso de entrenamiento y simulación, la aplicación debe mostrar una gráfica que represente el error de la iteración. Esta gráfica proporcionará una visualización del progreso del entrenamiento y permitirá identificar cualquier tendencia o patrón en el error a lo largo del tiempo.

  En el eje x de la gráfica tendremos el número de iteraciones o épocas que se ha realizado durante el entrenamiento de la red neuronal. Cada iteración representa un ciclo completo a través de tu conjunto de datos de entrenamiento. En el eje y tendremos el valor del error, que representa una métrica de pérdida utilizada para evaluar el rendimiento de la red neuronal.

  La gráfica mostrará cómo cambia el error a lo largo del tiempo durante el entrenamiento. Idealmente, se espera que el error disminuya a medida que la red neuronal aprendiera de los datos de entrenamiento y se ajustara a las características del problema. Sin embargo, es posible que se presenten fluctuaciones en el error debido a la variabilidad en los datos o algoritmos de optimización utilizados.



Requerimientos No Funcionales:

- **Interfaz de Usuario Intuitiva:** La aplicación debe contar con una interfaz de usuario fácil de usar que permita a los usuarios cargar datos, configurar el entrenamiento de la red y visualizar los resultados de manera clara y comprensible.

- **Eficiencia Computacional:** El proceso de entrenamiento y simulación de la red neuronal debe ser eficiente en términos de uso de recursos computacionales, para garantizar un rendimiento óptimo incluso con grandes conjuntos de datos.

- **Escalabilidad:** La aplicación debe ser capaz de manejar conjuntos de datos de cualquier tamaño, así como ajustar la arquitectura de la red neuronal para adaptarse a diferentes problemas y requisitos de rendimiento.

El desarrollo de esta aplicación requerirá la implementación de algoritmos y técnicas de aprendizaje automático, así como una sólida comprensión de los principios detrás de las redes neuronales artificiales y el aprendizaje supervisado. Además, se necesitarán habilidades de programación y diseño de software para crear una aplicación robusta y funcional que cumpla con los requisitos especificados.

Manual de usuario

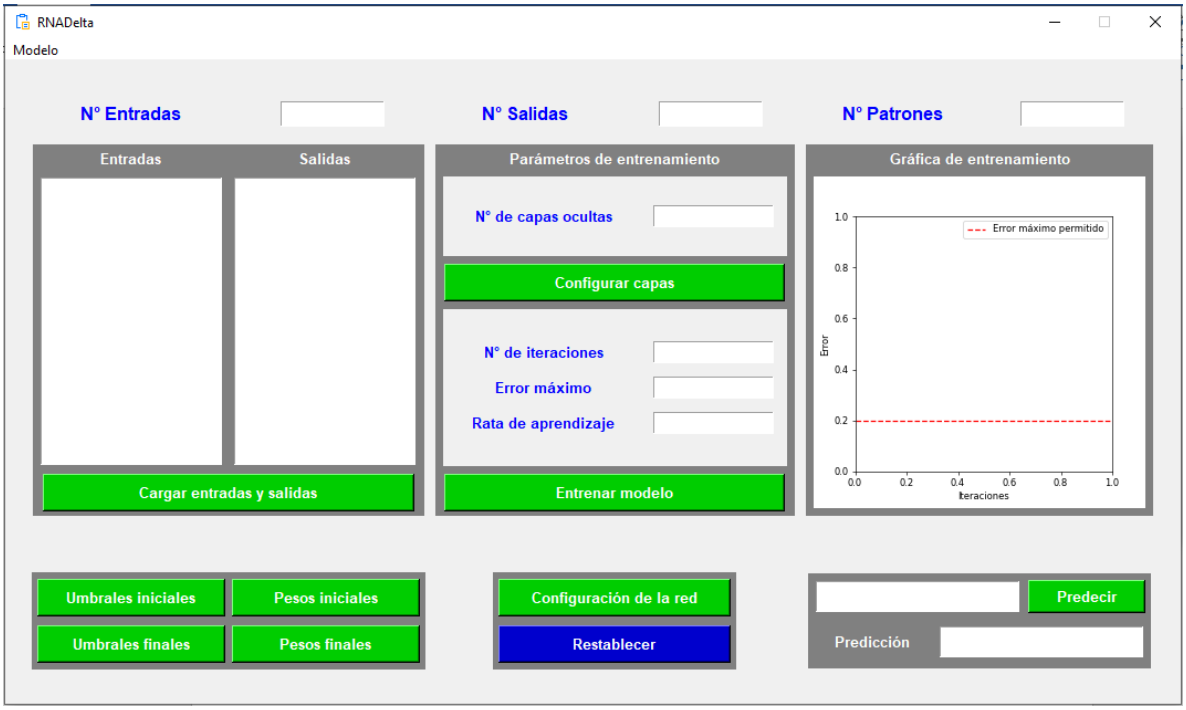
Esta aplicación está diseñada para proporcionar una herramienta intuitiva y poderosa para entrenar y simular redes neuronales artificiales de aprendizaje supervisado Backpropagation.

El aprendizaje supervisado es una técnica fundamental en el campo del aprendizaje automático, donde se enseña a un modelo a realizar predicciones basadas en ejemplos de entrada y salida conocidos. En el caso de las redes neuronales, este proceso implica ajustar los pesos de las conexiones entre las neuronas para minimizar la diferencia entre las predicciones del modelo y las salidas reales.

Esta aplicación ofrece una interfaz fácil de usar que permite cargar conjuntos de datos de entrenamiento y ajustar los parámetros de entrenamiento. Además, proporciona herramientas de visualización para analizar el progreso del entrenamiento y entender mejor el comportamiento de la red neuronal durante la simulación.

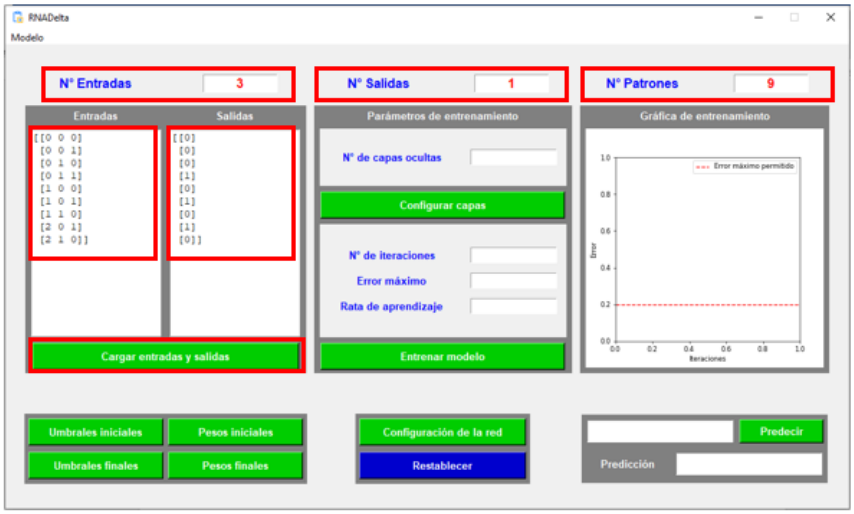
A lo largo de este manual, se proporcionarán instrucciones detalladas sobre cómo utilizar cada función de la aplicación, así como recomendaciones y mejores prácticas para obtener los mejores resultados en sus tareas de entrenamiento y simulación de redes neuronales.

Ventana de la aplicación



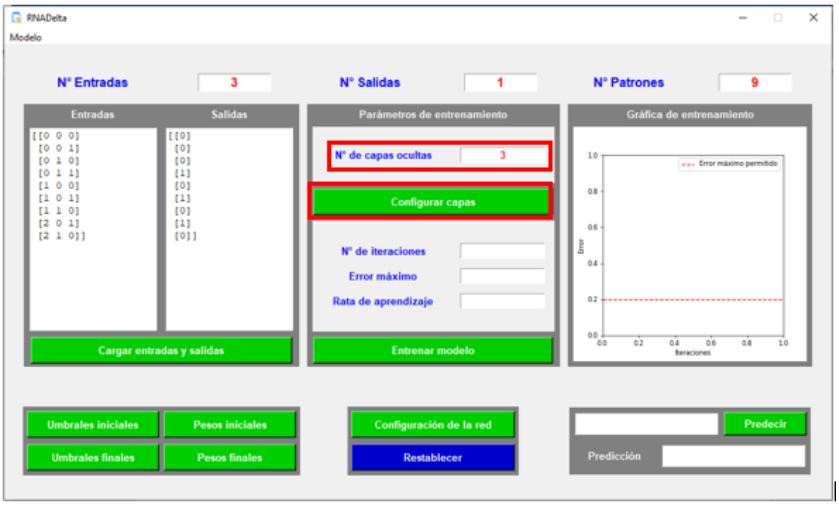
Cargar entradas y salidas

El botón cargar entradas y salidas nos permite cargar el conjunto de datos que tengamos disponibles previamente para el posterior entrenamiento de la red neuronal. Una vez cargado el conjunto de datos: las entradas, las salidas serán desplegadas en el área superior al botón. También se desplegará el número de entradas, el número de salidas y el número de patrones del conjunto de datos.



Configurar capas

El botón configurar capas nos permite establecer la cantidad de neuronas de cada capa, asi como también el tipo de función de activación que tendrá cada capa. Antes de oprimir el botón se debe ingresar el número de capas ocultas que tendrá el modelo.



Después de oprimir el botón se desplegará una ventana que nos permitirá elegir los parámetros antes mencionados: el número de neuronas de cada capa y su respectiva función de activación, incluyendo la capa de salida. Después de haber ingresado el conjunto de valores que son solicitados, se puede proceder a registrarlos.

Entrenar modelo

Esta opción nos permite realizar el entrenamiento del modelo por medio de los parámetros exigidos por la interfaz. La aplicación tomará los parámetros ingresados por el usuario y entrenará el modelo basado en esos parámetros usando el banco de datos cargado previamente.

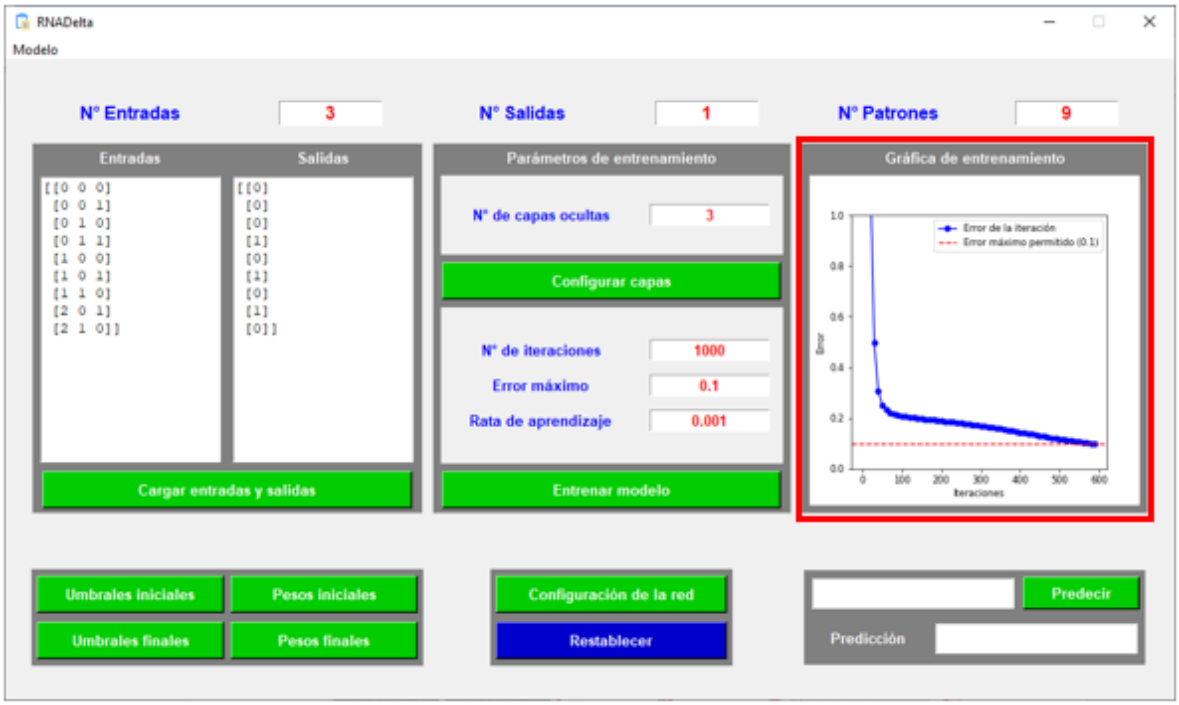
- **N° de iteraciones:** El número de iteraciones representa la cantidad de veces que se procesa todo el conjunto de datos de entrenamiento durante el entrenamiento de la red. Cada iteración consiste en un paso hacia adelante y un paso hacia atrás a través de la red, seguido de la actualización de los pesos basada en el error cometido. Un número adecuado de iteraciones garantiza que la red tenga la oportunidad de converger hacia una solución óptima sin sobreajustarse a los datos de entrenamiento.

- **Error máximo:** El error máximo, también conocido como tolerancia al error, especifica la precisión deseada o la cantidad aceptable de error que se puede tolerar durante el entrenamiento. Durante el proceso de entrenamiento, la red intentará minimizar este error, ajustando los pesos y los sesgos de las neuronas para reducir la discrepancia entre las salidas predichas y las salidas reales. Una vez que el error alcanza o se acerca al valor máximo definido, el entrenamiento puede detenerse, ya que es poco probable que la red mejore significativamente más allá de este punto.

- **Rata de aprendizaje:** La tasa de aprendizaje es un parámetro que controla la magnitud de los ajustes realizados en los pesos de la red durante cada paso de entrenamiento. Una tasa de aprendizaje más alta puede permitir que la red converja más rápidamente, pero también puede provocar oscilaciones o divergencias en el proceso de optimización. Por otro lado, una tasa de aprendizaje más baja puede garantizar una convergencia más suave y estable, pero el entrenamiento puede ser más lento. En la práctica, encontrar el valor óptimo de la tasa de aprendizaje es crucial para garantizar un entrenamiento eficiente y efectivo de la red neuronal.

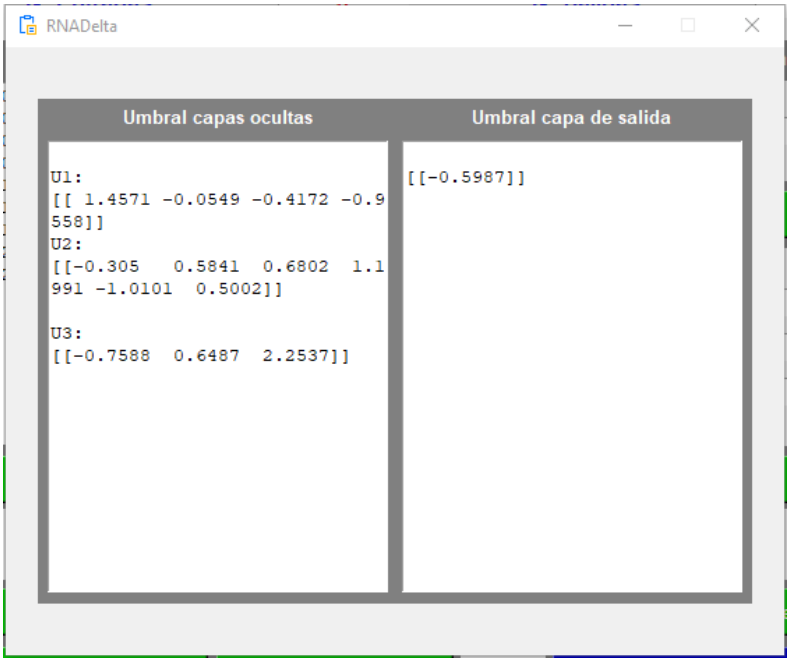
Gráfica de entrenamiento

Una vez realizado el entrenamiento del modelo, se desplegará la gráfica correspondiente al entrenamiento del modelo. Esta gráfica proporcionará una visualización del progreso del entrenamiento y permitirá identificar cualquier tendencia o patrón en el error a lo largo del tiempo.



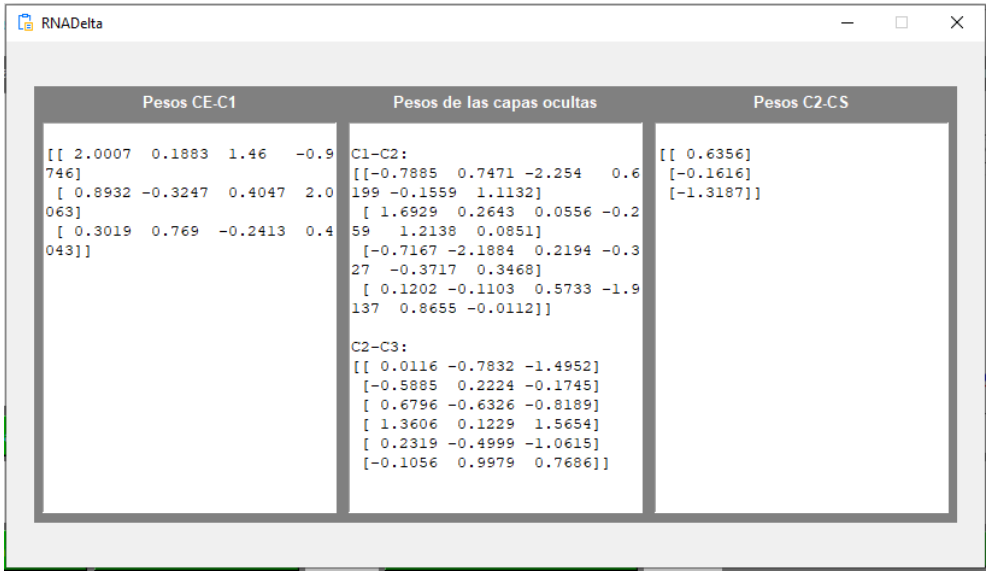
Umbrales iniciales

Una vez realizado el entrenamiento del modelo, este botón nos permitirá mostrar en pantalla los umbrales iniciales (escogidos aleatoriamente), que fueron utilizados para el entrenamiento del modelo de red neuronal.



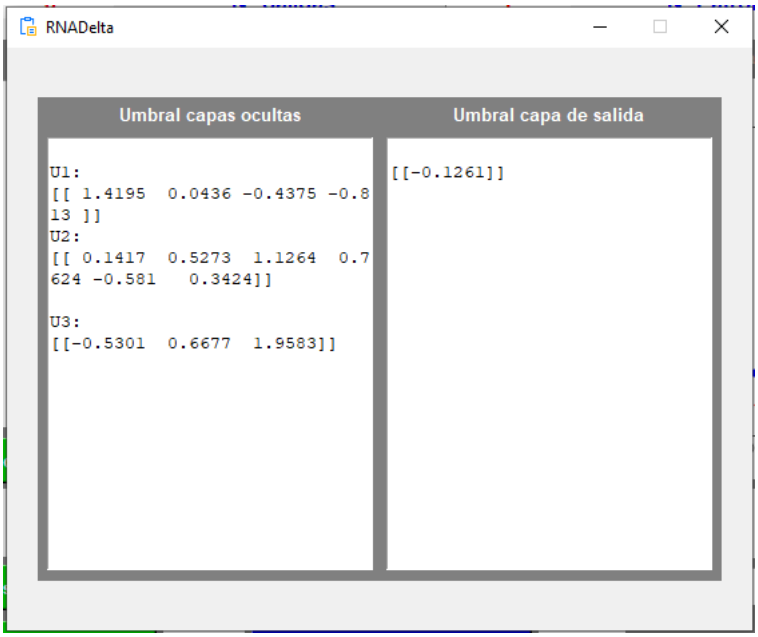
Pesos iniciales

Una vez realizado el entrenamiento del modelo, este botón nos permitirá mostrar en pantalla los pesos iniciales (escogidos aleatoriamente), que fueron utilizados para el entrenamiento del modelo de red neuronal.



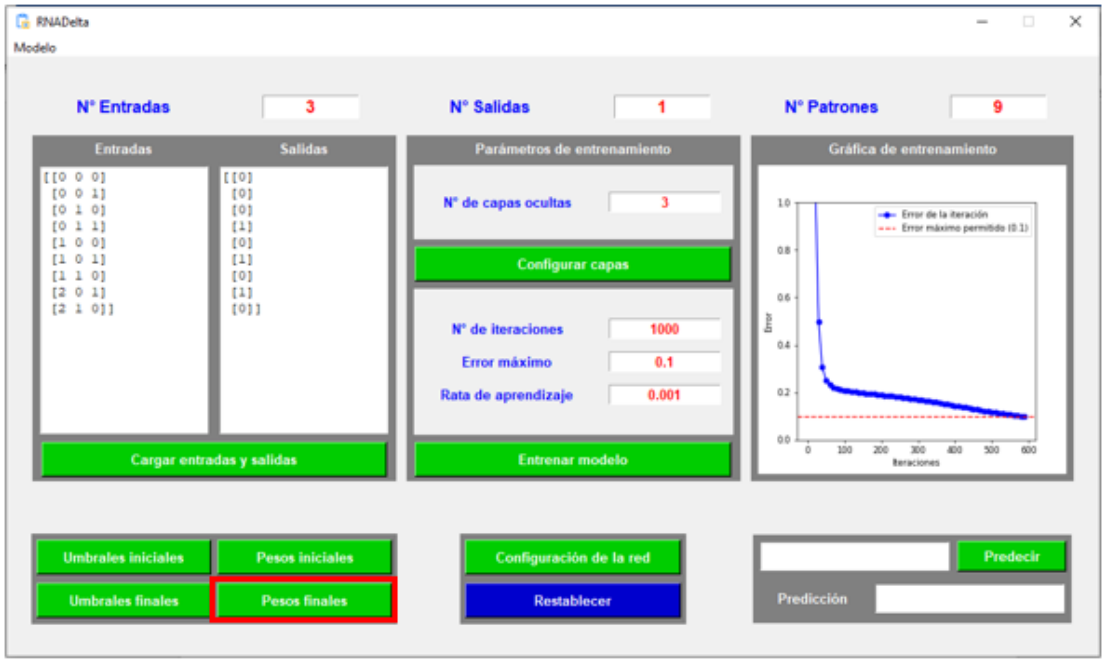
Umbrales Finales

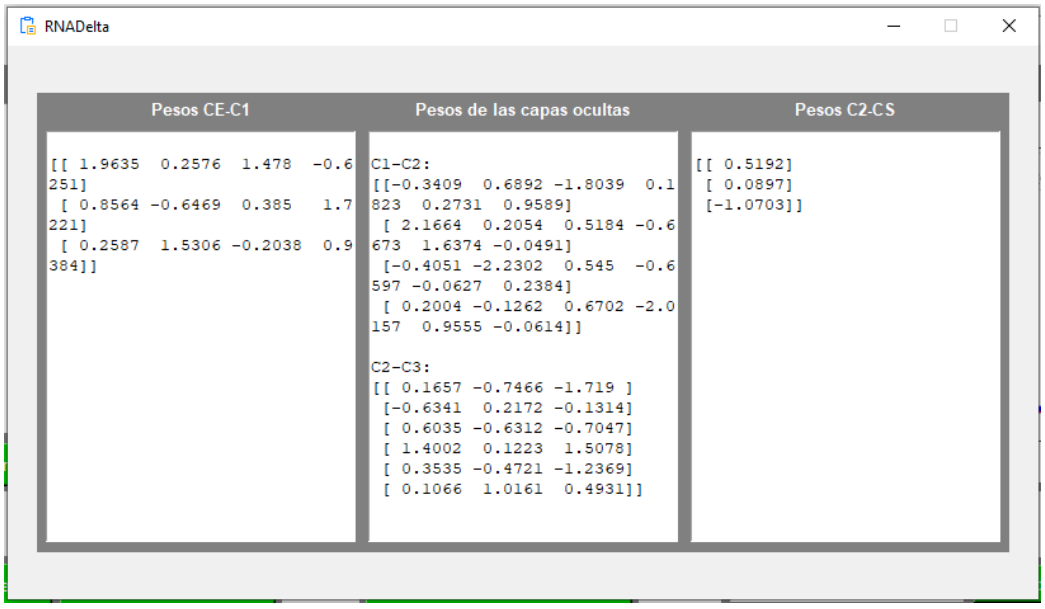
Una vez realizado el entrenamiento del modelo, este botón nos permitirá mostrar en pantalla los umbrales finales, que resultaron del entrenamiento del modelo de red neuronal (cuando el modelo ya ha alcanzado el error mínimo o el máximo de iteraciones).



Pesos finales

Una vez realizado el entrenamiento del modelo, este botón nos permitirá mostrar en pantalla los pesos finales, que resultaron del entrenamiento del modelo de red neuronal (cuando el modelo ya ha alcanzado el error mínimo o el máximo de iteraciones).





Reporte

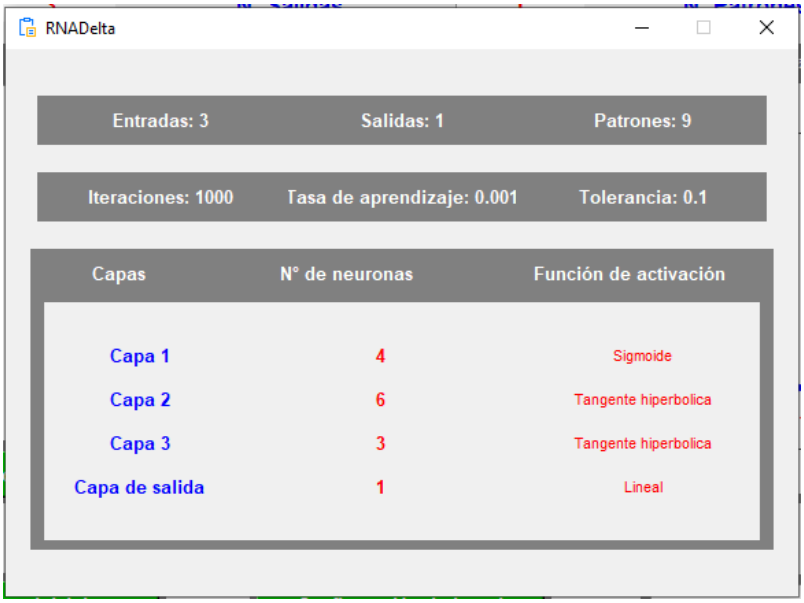
También se generará un informe breve con los datos del entrenamiento del modelo de la red neuronal, donde se especificarán: el número de entradas, número de salidas, numero de patrones, umbrales iniciales, pesos iniciales, umbrales finales, pesos finales, iteraciones alcanzadas, iteraciones totales, error máximo permitido y la rata de aprendizaje.

```
# ------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
Reporte de entrenamiento utilizando BACKPROPAGATION - 2024-05-12-18-49-48
------------------------------------------------------------------------------------------

Numero de entradas
3

Numero de salidas
1

Numero de patrones
9

Pesos iniciales
0.52
0.09
-1.07

Umbrales iniciales
-0.13
```

```
Pesos finales
1.96
0.26
1.48
-0.63
0.86
-0.65
0.38
1.72
0.26
1.53
-0.2
0.94

Umbrales finales
1.42
0.04
-0.44
-0.81

Iteraciones alcanzadas
590

Iteraciones totales
1000

Error maximo permitido
0.1

Rata de aprendizaje
0.001
```

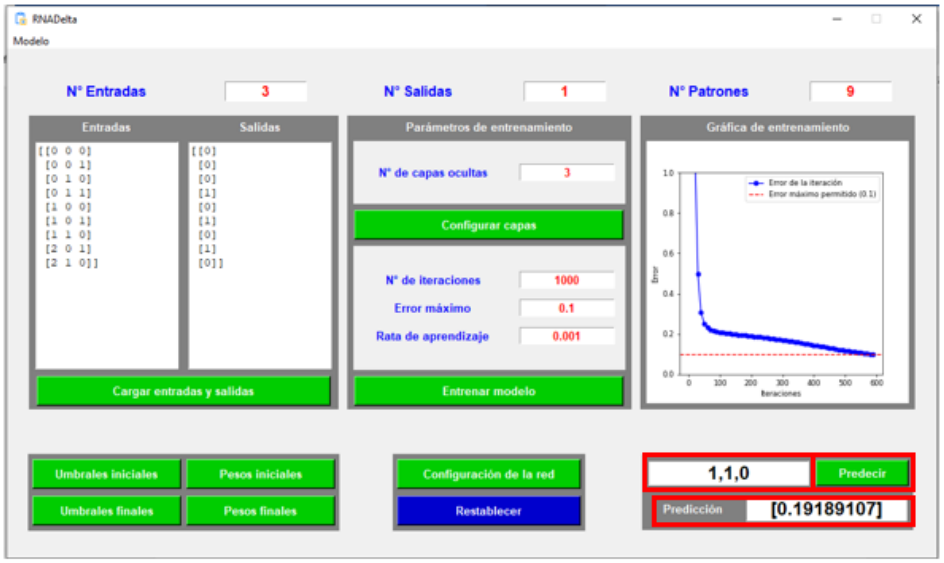
Configuración de la red

Este botón nos permitirá desplegar una ventana con la configuración que establecimos para la arquitectura del modelo de red neuronal, donde encontraremos: el número de entradas, salidas y patrones, como también las iteraciones, el error máximo y la tasa de aprendizaje. En la ventana también podremos visualizar el número de capas de la red, como la cantidad de neuronas de cada capa y su respectiva función de activación.



Predecir

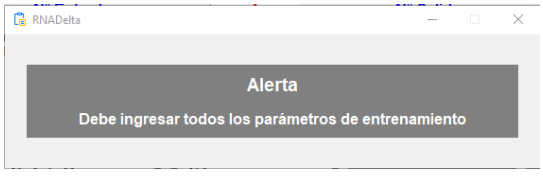
Esta opción nos permite realizar una predicción una vez que el modelo este entrenado, solo se requiere ingresar el patrón al que se requiere realizar la predicción (separados por coma cada valor). La predicción resultante se desplegará en el cuadro de texto inferior.

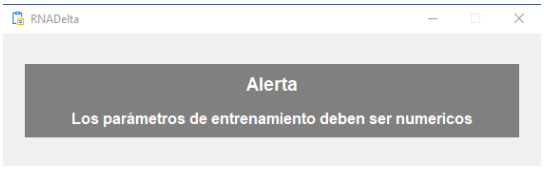


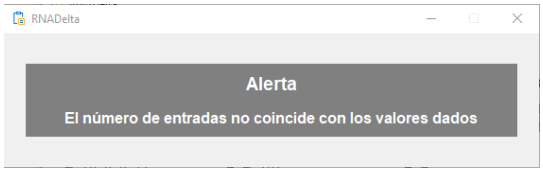
Validaciones

El programa realiza las siguientes validaciones:

- Verifica que se hayan ingresado las entradas y salidas antes de realizar el entrenamiento del modelo.
- Verifica que se hayan ingresado los parámetros de entrenamiento antes de realizar al entrenamiento del modelo.
- Verifica que los parámetros ingresados sean numéricos.
- Verifica que el patrón a predecir corresponda al número de entradas del modelo.

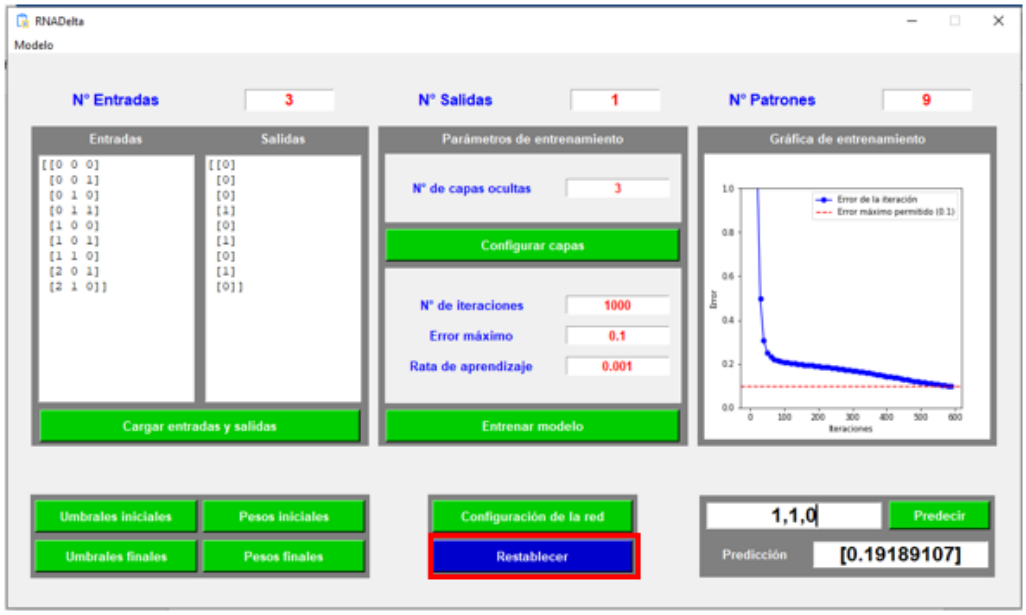


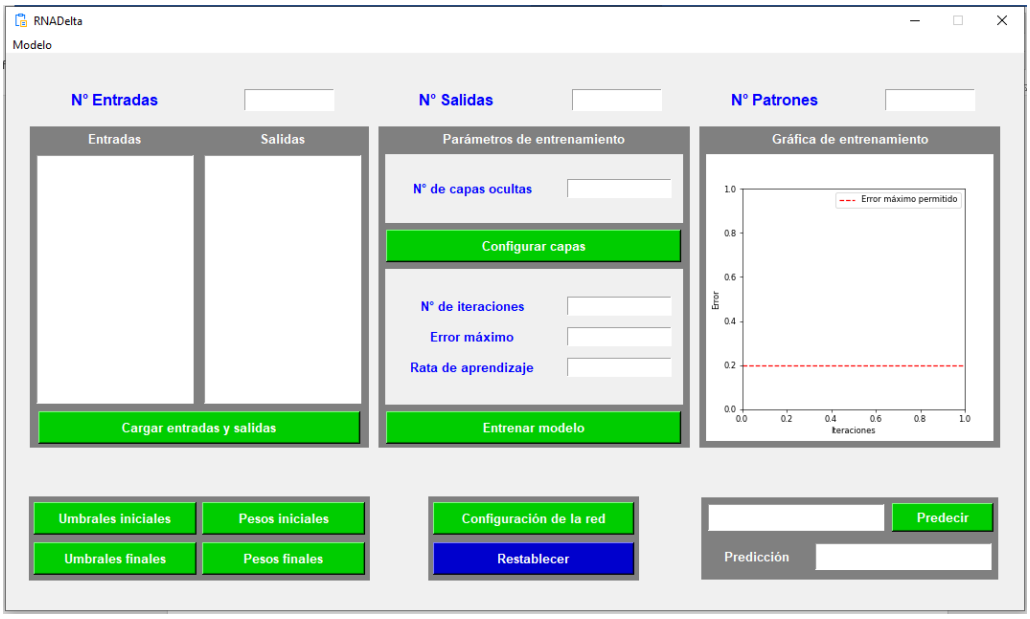




Restablecer

Esta opción nos permite restablecer todos los valores de la aplicación para realizar una nueva configuración del modelo de red neuronal. Todos los parámetros y datos del modelo serán eliminados.





Conclusión

En el transcurso del desarrollo de este informe, se presentó una descripción detallada de la aplicación propuesta, incluyendo su diseño, implementación y funcionalidades clave. Se proporcionó una explicación breve de los conceptos de aprendizaje supervisado, así como una guía paso a paso sobre cómo utilizar la aplicación para resolver problemas específicos.

Además, se presentó ejemplos de aplicación y casos de uso práctico, demostrando la eficacia y la versatilidad de la herramienta en una variedad de situaciones del mundo real. Finalmente, pudimos identificar el desarrollo de áreas futuras para mejorar y ampliar la aplicación.

Este trabajo representa un esfuerzo significativo para desarrollar una aplicación de aprendizaje supervisado centrada en la resolución de problemas sencillos, con un enfoque en la flexibilidad, la eficacia y la facilidad de uso.

Referencias bibliográficas

- Las Redes Neuronales Artificiales y su importancia como herramienta en la toma de decisiones. Villanueva Espinoza, María del Rosario. https://sisbib.unmsm.edu.pe/bibvirtualdata/tesis/basic/Villanueva_EM/enPDF/Cap5.PDF

- Procesamiento Digital de Imágenes. Pablo Roncagliolo B. (2007). http://www2.elo.utfsm.cl/~elo328/PDI21_RedesNeuronales.pdf

- Fundamentos de las Redes Neuronales. Luis Moreno, Santiago Garrido, Dorin Copaci. (2009). https://ocw.uc3m.es/pluginfile.php/143/mod_page/content/24/Fundamentos%20de%20las%20Redes%20Neuronales.%20Parte%20II%20L7.pdf