TALLER DE INTELIGENCIA ARTIFICIAL

DOCENTE

TONNY JIMENEZ

ANIBAL DAVID FUENTES PRIETO

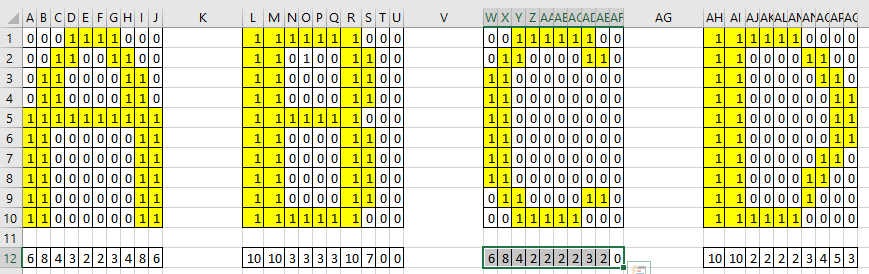
UNIVERSIDAD POPULAR DEL CESAR INGENIERIA DE SISTEMAS VALLEDUPAR, CESAR

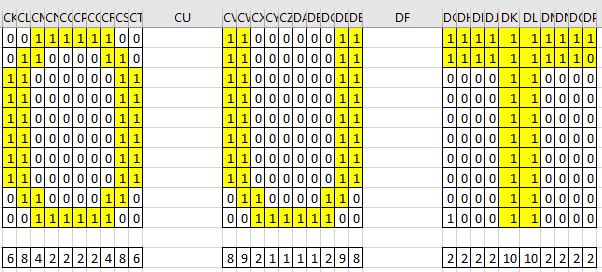
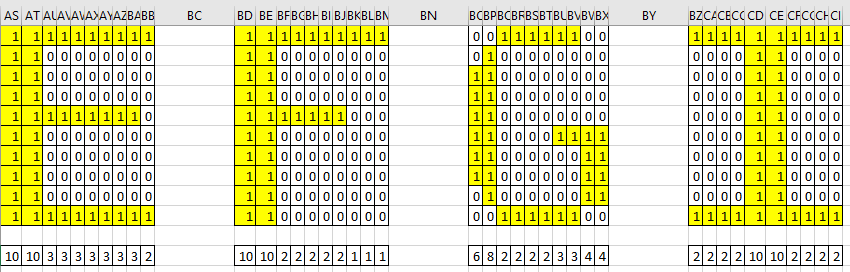
2025

# Punto 1:Red Neuronal al reconocimiento de la letra „T“\*

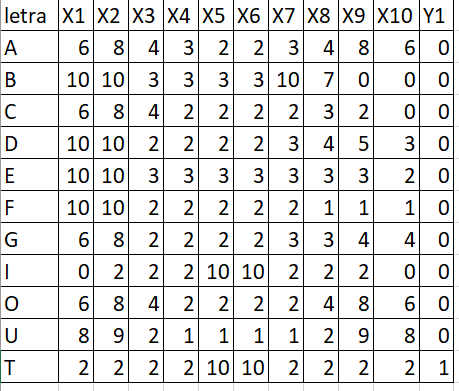
Bueno lo primero que hacemos es crear las letras (A,B,C,D,E,F,G,I,O,U,T) en las matrices de 10x10, con sus receptivas sumas para cada coluda, como se puede

apreciar en las imágenes:





Después de esto creamos nuestro datase



Ya a partir de esta información cree un programa en Matlab que me permita reconocer que letra

es y si es una vocal.

# procesamiento de Datos

Cada letra se representa como una matriz binaria de tamaño 10×1010×10, donde: 1 = Píxel activo (parte de la letra).

0 = Píxel inactivo (fondo).

Se calcula el vector suma por columnas Ejemplo:

* La letra T en una matriz 10×10 tiene un vector suma típico: ST= [2,2,2,2,10,10,2,2,2,2]

# Arquitectura de la Red Neuronal

La red implementada sigue una estructura feedforward con:

# Capa de entrada:

10 neuronas (una por cada valor del vector suma).

# Capas ocultas:

Configurables (1 a 3 capas).

Cada capa puede tener distintas funciones de activación (tansig, logsig, purelin, poslin, satlin).

# Capa de salida:

2 neuronas con función logsig (salida en [0, 1]):

**Salida 1:** Probabilidad de ser T. **Salida 2:** Probabilidad de no ser T. **Proceso de entrenamiento**

El entrenamiento utiliza descenso de gradiente con alguna de las siguientes variantes:

* **trainscg** (Gradiente conjugado escalado) → Eficiente en memoria.
* **trainlm** (Levenberg-Marquardt) → Rápido, pero consume más recursos.
* **trainbr** (Bayesiano) → Bueno para evitar sobreajuste.
* **trainrp** (Resiliente) → Adapta tasas de aprendizaje por peso.

# División de Datos

70% Entrenamiento → Ajuste de pesos.

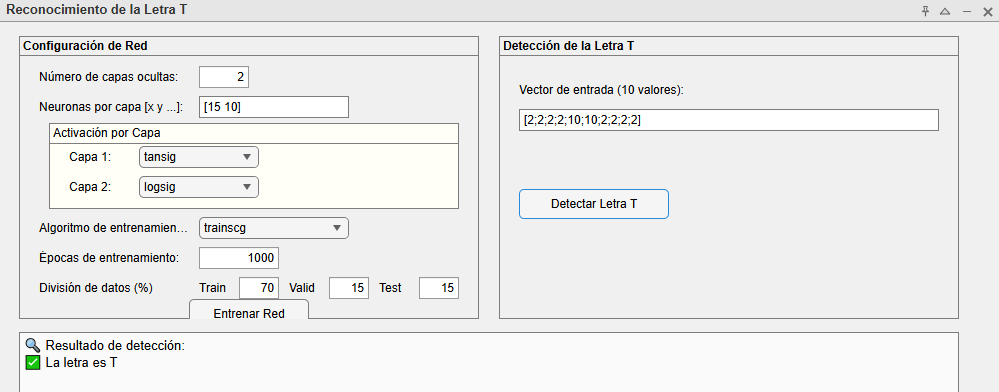
15% Validación → Parada temprana (evitar sobreajuste). 15% Prueba → Evaluación final.

todos estos datos los podemos modificar en la interfaz que creamos

Clasificación Binaria

La salida de la red la interpretamos de forma binaria:

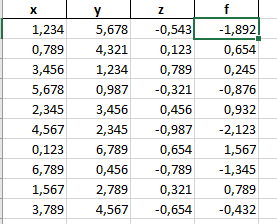
|  |  |
| --- | --- |
| **salida** | **clasificación** |
| (1) | Es la letra T |
| (0) | No es la letra T |



# Punto 2:

Bueno lo primero que hacemos es analizar 10 patrones y crear nuestro dataset





Donde x, y, z son las entrada y f la salida.

Y a partir de esta información cree un programa en Matlab para poder reconocer los 10 patrones que tengo en mi datase.

¿Cómo funciona?

# Carga del Dataset

Se carga automáticamente un archivo Excel (Dataset\_Fxyz.xlsx) que contiene los datos de entrada (x, y, z) y la salida esperada F.

# Construcción de la Red Neuronal

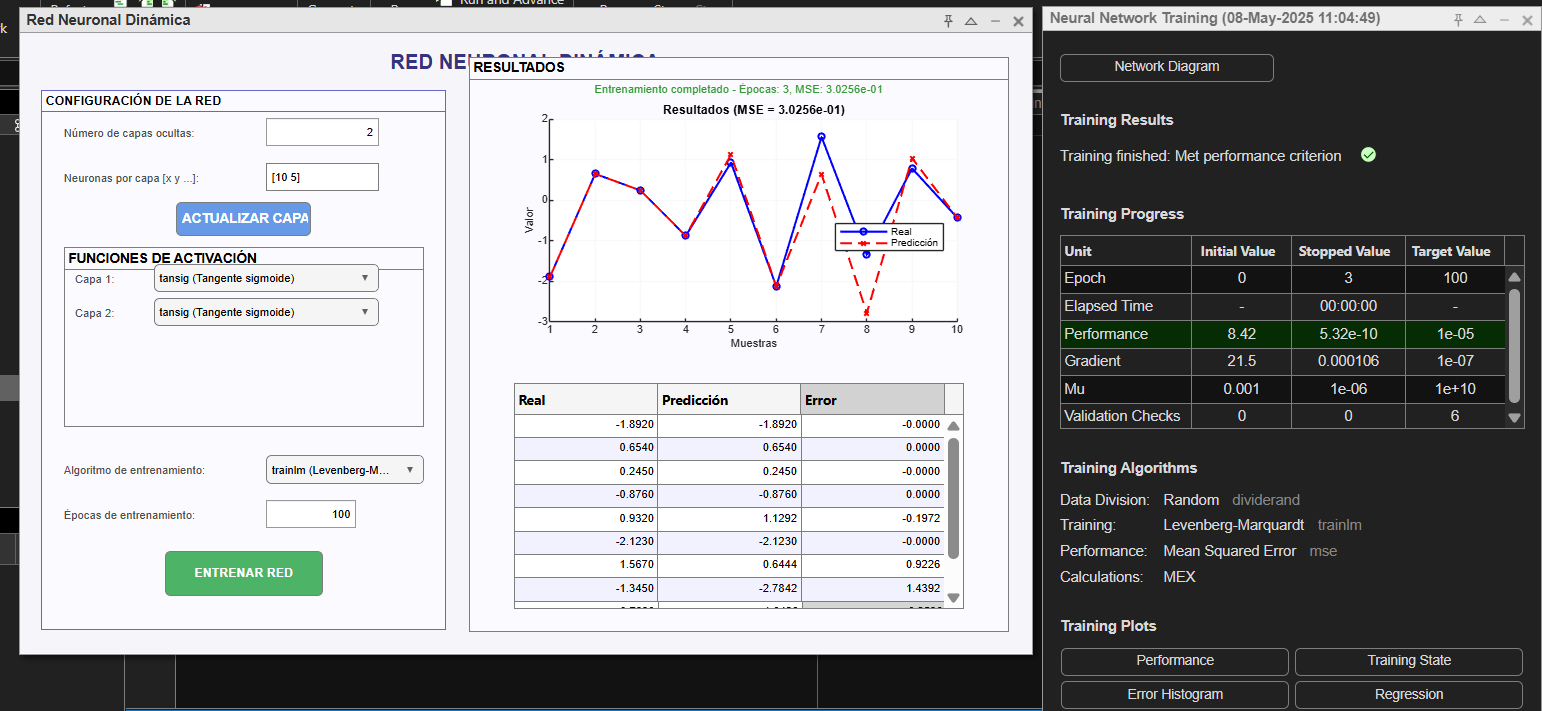
Con base en los valores introducidos en la GUI, se construye una red con el número de capas, neuronas y funciones de activación seleccionadas.

# Entrenamiento de la Red

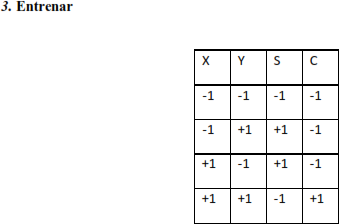
La red se entrena usando el algoritmo elegido. Se divide el dataset en conjuntos de entrenamiento, validación y prueba (70%-15%-15%).

# Evaluación y Visualización

Una vez entrenada, la red genera predicciones sobre los datos de entrada. Estas predicciones se comparan con los valores reales en un gráfico, y se muestra el MSE.



# Punto 3

****

Bueno lo primero que hacemos en este caso en interpretar cuales son las entradas y salidas ya que el ejercicio no lo especifica.

Calcificamos los datos de la siguiente manera

**Entradas:** X, Y

# Salida esperada: S C

**Los valores son binarios**: -1 o +1

Luego creamos nuestro dataset en Excel para obtener nuestros datos directamente en el código que realizamos en Matlab



Ya teniendo toda esta información, no es más que realizar el código correspondiente.

Realizamos una interfaz intuitiva para el usuario que está dividida en dos sesiones principales

# Panel de Configuración de Red

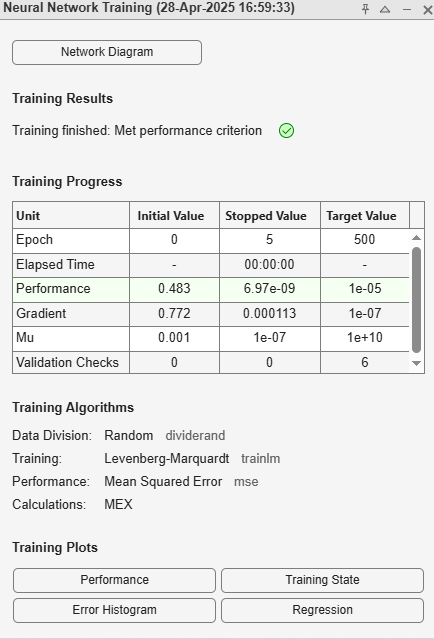
Este panel permite al usuario definir la arquitectura de la red neuronal. Incluye los siguientes componentes:

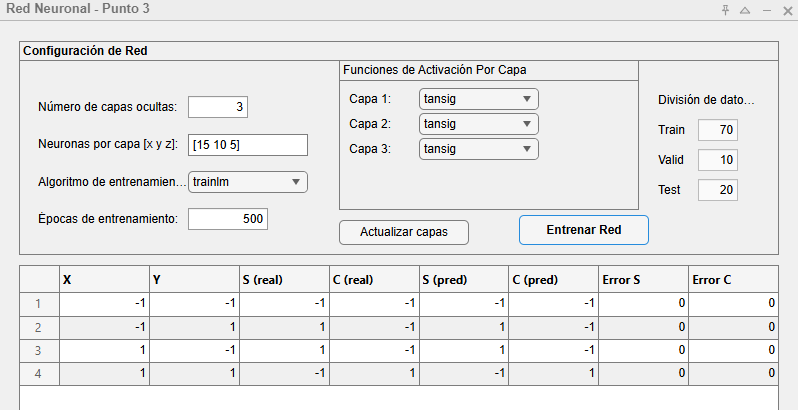
* + Número de capas ocultas
  + Neuronas por capa
  + Funciones de activación por capa
  + Algoritmo de entrenamiento
  + Épocas de entrenamiento
  + División de datos

# Tabla de Resultados

En la parte inferior de la interfaz, se muestra una tabla numérica con los resultados obtenidos tras el entrenamiento. La tabla incluye:

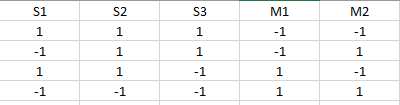
* + Las entradas originales (X, Y)
  + La salida esperada (S, C)
  + La salida predicha por la red
  + El error (Esperado − Predicción)





**Punto 4:** Considere la tabla, en esta aparecen los estados de los sensores (S1, S2, S3) de un mini robot y los estados de sus motores (M1, M2), las demás condiciones no importan.

Bueno como ya sabemos cuáles son las salidas y entradas lo primero que hacemos en crear nuestro dataset para llamar los datos directamente desde ahí.



Tomamos S1, S2, S3 como entradas y salidas M1 Y M2, ya teniendo estos valores claros y almacenados previamente, pasamos a construir nuestro algoritmo en Matlab.

Componentes y funcionamiento

# Panel de Configuración de Red

Este panel permite al usuario establecer libremente los parámetros más relevantes de la red neuronal:

# Número de capas ocultas

El usuario selecciona cuántas capas ocultas incluir en la red

# Número de neuronas por capa

Se introduce como un vector (ejemplo: [10 5]).

# Funciones de activación por capa

De forma dinámica, se generan listas desplegables donde el usuario elige una función de activación para cada capa.

# Algoritmo de entrenamiento

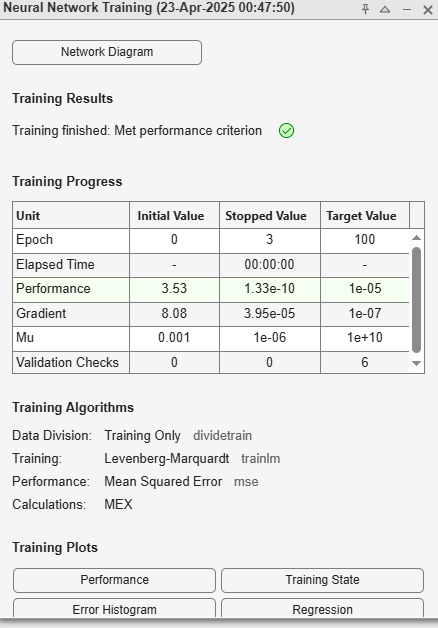
El usuario puede seleccionar entre varios algoritmos para el ajuste de pesos de la red

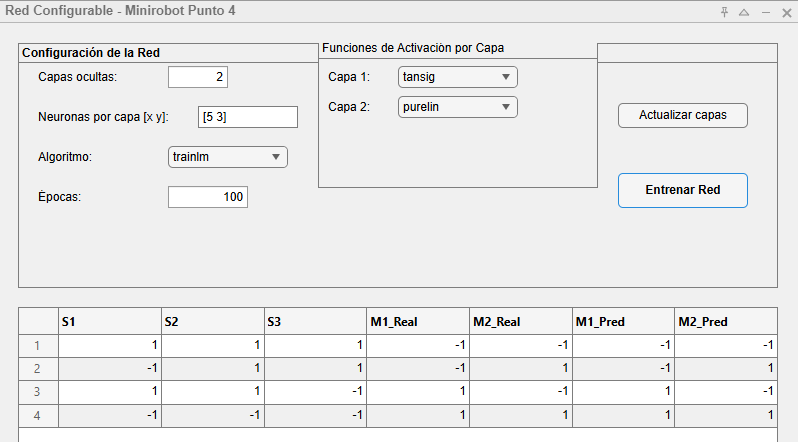
# Número de épocas

Define cuántas iteraciones se realizan durante el entrenamiento.

# Tabla de Resultados

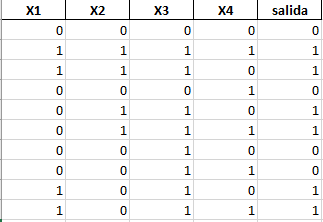
Luego del entrenamiento, se muestran en una tabla con los resultados obtenido, comparando las salidas deseados con las esperaddas, como en la siguiente imagen.





**Punto 5:** Esta red neuronal, está entrenada para diagnosticar un resfriado a partir de los síntomas que se le indiquen

Bueno lo primero que hacemos es crear nuestro dataset en Excel, para identificar cuáles son nuestras entradas y salidas



Las primeras 3 columnas representan los síntomas (entradas). La última columna representa la salida, siendo:

1 → paciente con resfriado. 0 → paciente sin resfriado.

# Configuración de la red neuronal

La interfaz permite al usuario definir varios parámetros personalizables:

* Número de capas ocultas.
* Número de neuronas por capa
* Funciones de activación para cada capa
* Algoritmo de entrenamiento
* Cantidad de épocas de entrenamiento.

# ¿Cómo evaluación de síntomas?

En la parte inferior de la interfaz se incluyen cuatro casillas de verificación:

* Dolor de cabeza
* Fiebre
* Tos
* Dolor de rodilla

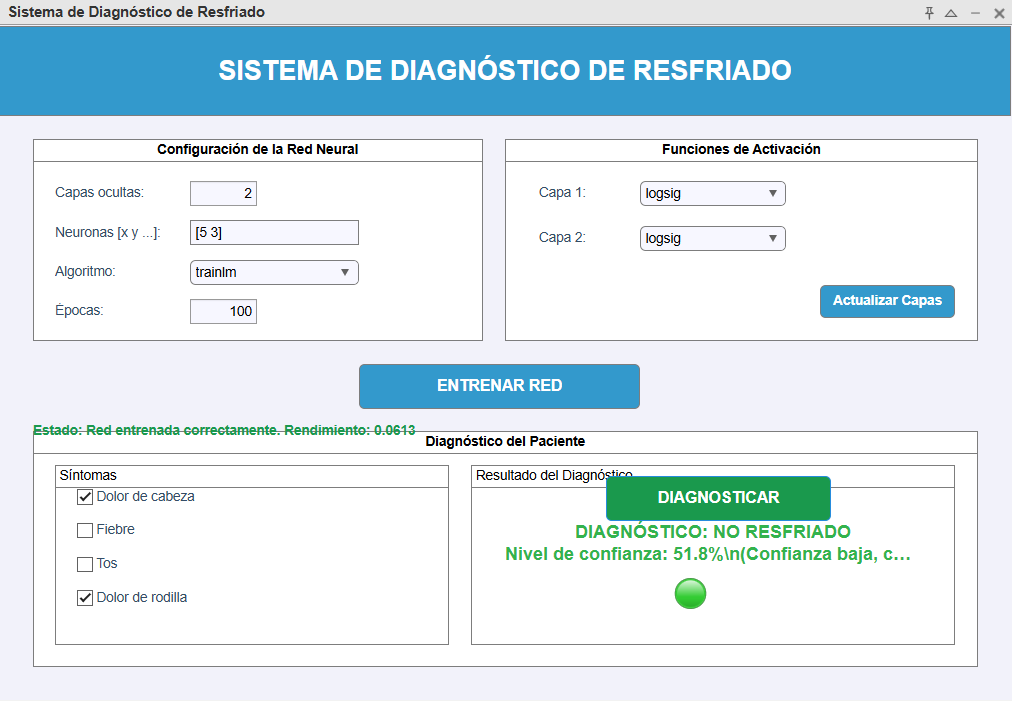
Cada una representa una característica binaria (0 o 1) de entrada. Al presionar el botón “Predecir”, se realiza lo siguiente:

Se capturan los valores de los síntomas seleccionados. Se pasan como entrada a la red neuronal entrenada.

La red emite una salida entre 0 y 1.

1 indica que el paciente tiene resfriado. 0 indica que no tiene resfriado.

como se puede ver en las imágenes





Ahora realizamos el mismo punto, pero con Función de base radial, lo primero que hacemos es:

Inicialización de la red

Creamos una estructura red para almacenar:

* W: pesos
* b: sesgo
* C: centros RBF
* sigma: ancho de las funciones gaussianas
* errores: errores durante el entrenamiento

Luego definimos los centros, con una función que nos permite generar una tabla editable con los **centros RBF** aleatorios, que representan los "puntos clave" donde se centra cada función gaussiana.

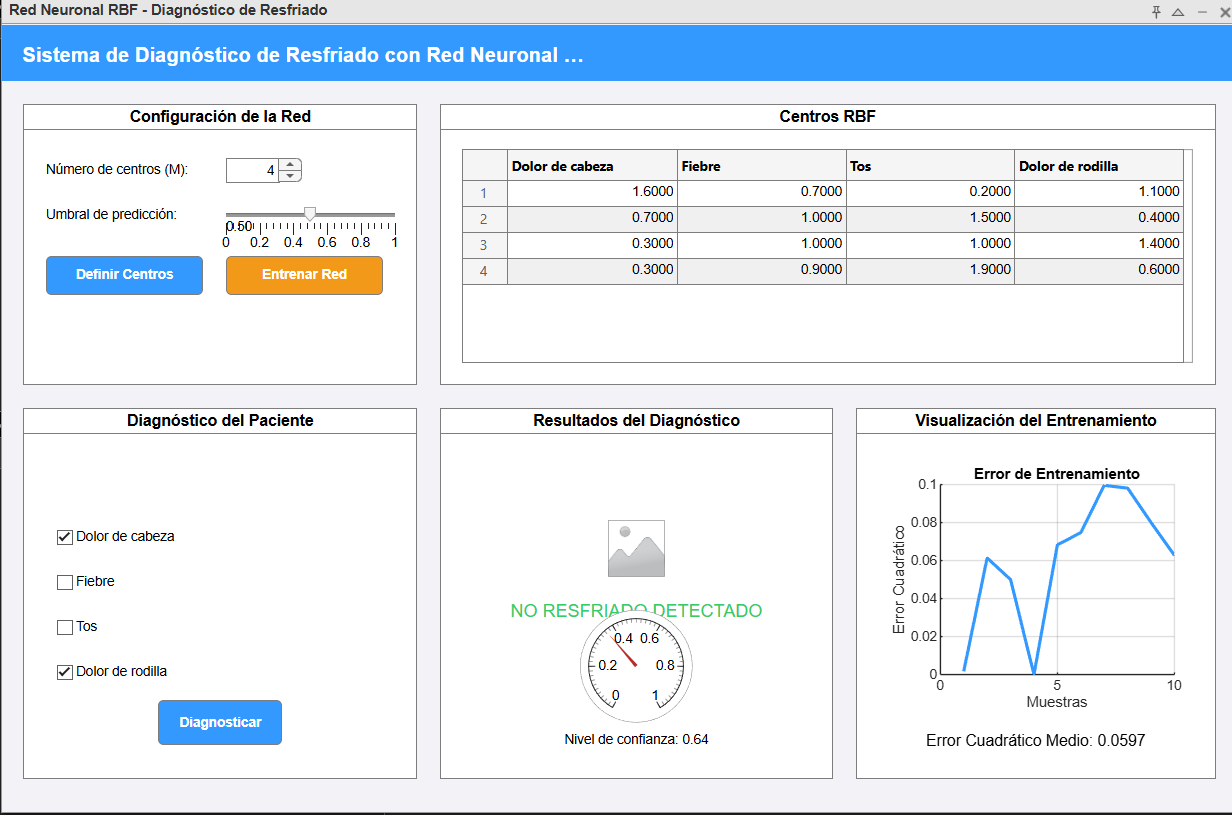
Después con la función entrenar

Realiza el entrenamiento manual de la red:

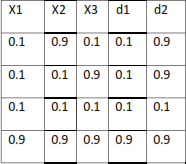
1. Obtiene los centros
2. Calcula sigma como la distancia promedio entre centros.
3. Construye la matriz de activación Φ con funciones gaussianas.
4. Calcula los pesos y sesgo usando la pseudoinversa.
5. Calcula el error cuadrático para cada muestra.
6. Muestra la gráfica del error en el panel.

Y por último con la función predecir

* + Construye un vector de entrada con base en los síntomas.
  + Calcula activaciones con los centros y sigma.
  + Calcula la salida de la red: y\_net = φ·W + b
  + Compara con el umbral para decir si hay resfriado (1) o no (0).
  + Muestra el resultado en texto.

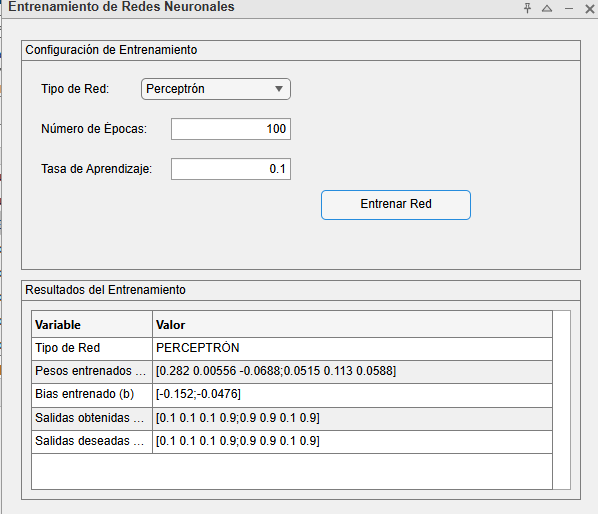


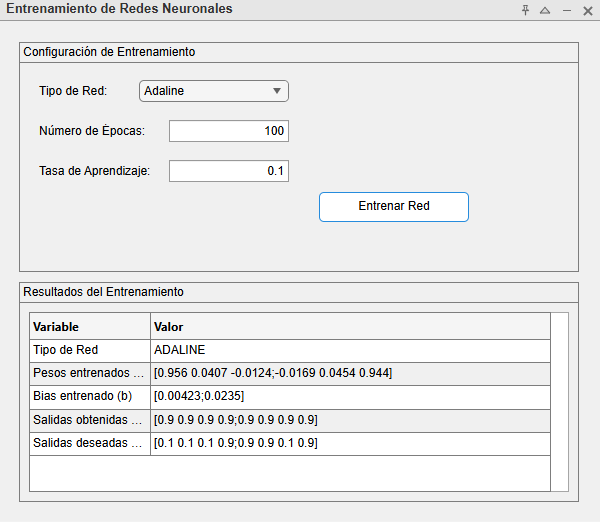
**Punto 6:** Entrenar una red perceptrón y adaline



Los pesos iniciales de la red se escogen al azar en el intervalo [-0.1, 0.1] y la rata de aprendizajes de 0.1

Bueno en este ejercicio tenemos como entadas a X1, X2, y X3. Y como salidas a d1 y d2





**Punto 7:** Este archivo contiene un conjunto de datos estructurado centrado en la detección temprana y la

progresión de la enfermedad renal crónica (ERC). Consolida datos clínicos, bioquímicos y de estilo de

vida completos de los pacientes, ofreciendo un total de 43 atributos que abarcan datos demográficos,

resultados de análisis de laboratorio, síntomas, historial médico y biomarcadores. El conjunto de datos está diseñado para servir como recurso para:

* Modelos de aprendizaje automático orientados a clasificar y predecir estadios de ERC.
* La investigación médica se centró en las correlaciones entre los parámetros del paciente y la salud

renal.

* Sistemas de apoyo a la toma de decisiones clínicas para el diagnóstico personalizado y la planificación

del tratamiento.

Incluye datos tanto categóricos como numéricos, y la columna "Objetivo" indica la presencia o clasificación

de la enfermedad renal, lo que hace que este archivo sea muy valioso tanto para tareas de aprendizaje

supervisado como para análisis de datos exploratorios.

Ya sea que sea un investigador, un científico de datos o un entusiasta de la atención médica, este conjunto

de datos le permite explorar patrones, factores de riesgo y resultados asociados con los trastornos renales

de una manera basada en datos.

