

Universidad de San Carlos de Guatemala

Centro Universitario de Occidente

Ingeniería en Ciencias y Sistemas

Laboratorio Inteligencia Artificial 1



Manual Tecnico

202031288 Jhony Roel Fuentes Lopez

Introducción

El presente manual técnico documenta el desarrollo de una aplicación para la clasificación de cáncer de mama (benigno o maligno) utilizando un perceptrón, un modelo básico de redes neuronales artificiales. Este proyecto tiene como objetivo implementar una solución de aprendizaje automático que permita clasificar muestras del dataset "Breast Cancer Wisconsin" (disponible en el repositorio de UCI Machine Learning) en función de sus características clínicas, como el radio medio, la textura media, entre otras.

Marco Teórico

Perceptrón:

En 1957 Frank Rosenblatt inventó el perceptrón en el laboratorio aeronáutico de Cornell. Basándose en los primeros conceptos de neuronas artificiales, propuso la “regla de aprendizaje del perceptrón”.

Un perceptrón es una neurona artificial, y, por tanto, una unidad de red neuronal. El perceptrón efectúa cálculos para detectar características o tendencias en los datos de entrada.

Se trata de un algoritmo para el aprendizaje supervisado de clasificadores binarios. Ese algoritmo es el que permite que las neuronas artificiales aprendan y traten los elementos de una serie de datos.

El perceptrón desempeña un papel esencial en los proyectos de Machine Learning. Se utiliza en gran medida para clasificar datos, o como algoritmo que permite simplificar o supervisar las capacidades de aprendizaje de los clasificadores binarios.

Dataset: Breast Cancer Wisconsin

El dataset "Breast Cancer Wisconsin" contiene 569 muestras con 30 características cada una, extraídas de imágenes digitalizadas de biopsias de mama. Las etiquetas son binarias: 0 (benigno) y 1 (maligno). Para este proyecto, se seleccionan dos características (como "radius1" y "texture1") para entrenar el perceptrón y visualizar los resultados en un espacio bidimensional.

Herramientas y Tecnologías Utilizadas

- **Lenguaje de Programación:** Python 3.12.0, seleccionado por su amplio soporte para bibliotecas de aprendizaje automático y visualización.
- **Bibliotecas de Python:**
 - **NumPy (1.26.4):** Para operaciones numéricas y manejo de arreglos multidimensionales, como los cálculos del perceptrón y la normalización de datos.
 - **Matplotlib (3.8.0):** Para generar gráficos, incluyendo scatter plots, curvas de error y fronteras de decisión.
 - **Tkinter (incluido en Python 3.12.0):** Para crear la interfaz gráfica de usuario (GUI) que permite interactuar con el sistema.

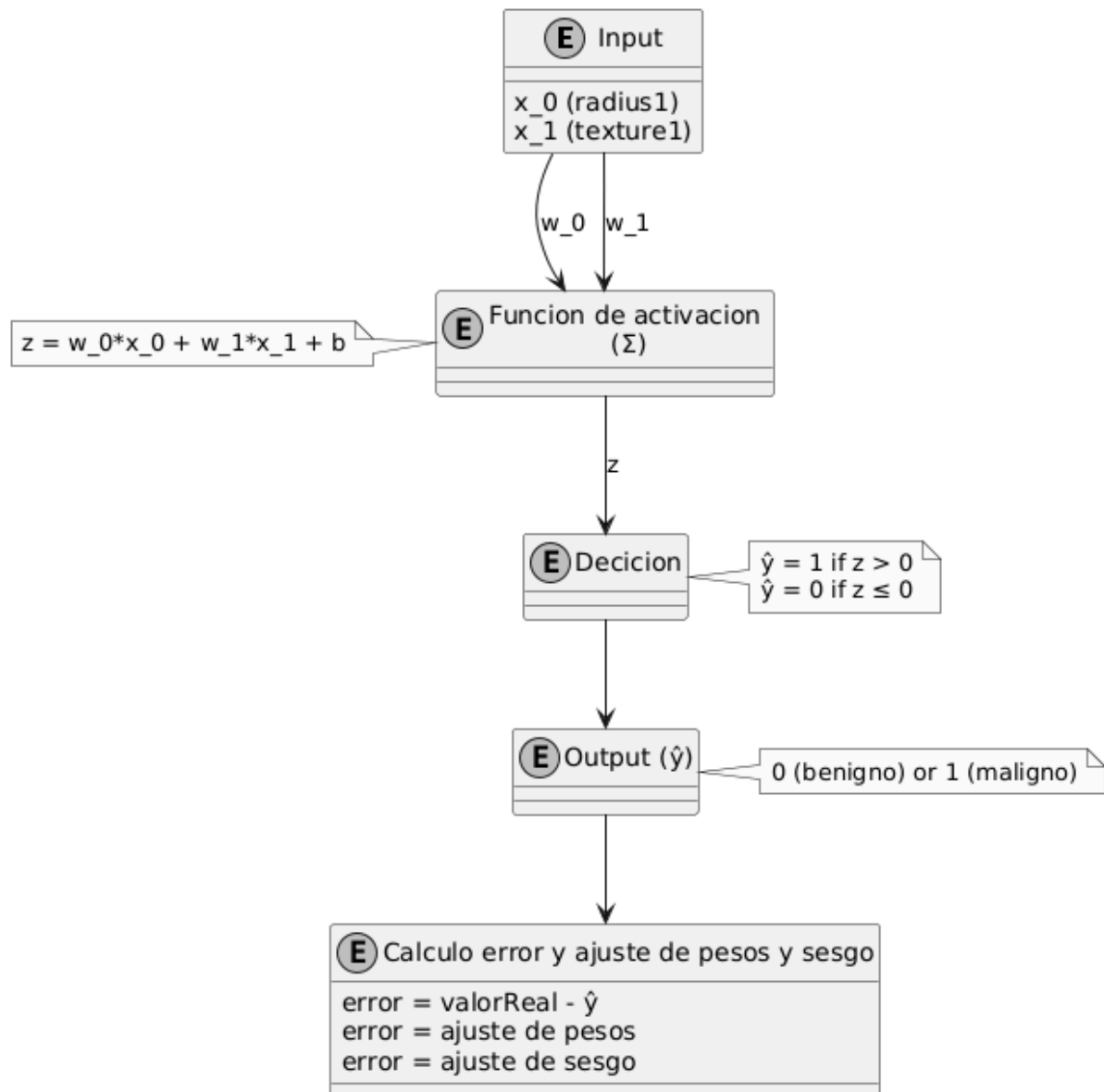
- **CSV (incluido en Python 3.12.0):** Para leer el archivo wdbc.data que contiene el dataset.
- **Dataset:** Breast Cancer Wisconsin (Diagnostic) Data Set, obtenido del repositorio de UCI Machine Learning.
- **Entorno de Desarrollo:** El código fue desarrollado y probado en un entorno local con Python 3.12.0, compatible con sistemas operativos como Windows, macOS y Linux.
- **Dependencias Adicionales:**
 - Se requiere que el archivo wdbc.data esté en el mismo directorio que la aplicación para cargar los datos correctamente.

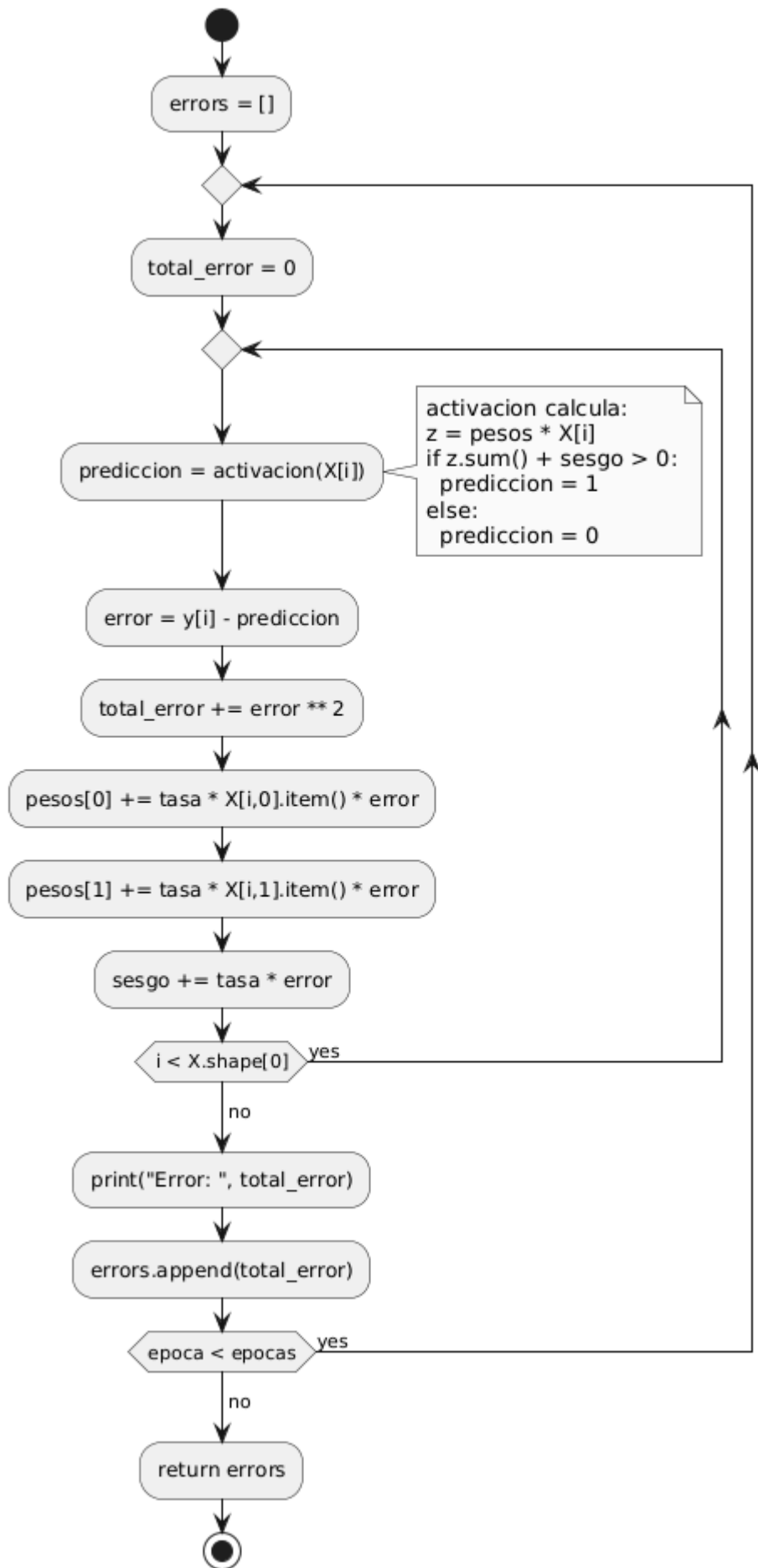
Clase perceptron:



Entrenamiento del perceptrón flujo:


Este flujo se ejecuta $n \cdot t$ veces donde n es el número de épocas y t es el número de datos que se especifica en la interfaz de usuario ,por cada par de datos (input) se pasan a la función activación, este se encargará de asignarle un valor según los pesos y el sesgo, y después tomará una decisión si es maligno o benigno, este será nuestro output, después de esto se calculará el error (si hay), y se ajustarán los pesos y el sesgo.





CLASE clasificación Cáncer:

Esta clase se encarga de crear la interfaz gráfica y de manejar los datos seleccionados por el usuario.

 clasificacionCancer
<ul style="list-style-type: none">□ root: tk.Tk□ X: np.ndarray□ y: np.ndarray□ datosNormlaizados: np.ndarray□ datosPruebagrafica: np.ndarray□ solucionPrueba: np.ndarray□ solucionEntreno: np.ndarray□ datosEntreno: np.ndarray□ datosPrueba: np.ndarray□ perceptron: Perceptron□ errors: list□ feature_names: list□ seleccionX: tk.IntVar□ seleccionY: tk.IntVar□ tasa: tk.DoubleVar□ epoca: tk.IntVar□ fig: matplotlib.figure.Figure□ ax1: matplotlib.axes.Axes□ ax2: matplotlib.axes.Axes□ canvas: FigureCanvasTkAgg
<ul style="list-style-type: none">● <code>__init__(root: tk.Tk)</code>● <code>load_data()</code>● <code>normalizar()</code>● <code>create_gui()</code>● <code>plot_data()</code>● <code>entrenar()</code>● <code>update_plots()</code>

Requerimientos del Sistema

- **Sistema Operativo:** Compatible con Windows, macOS o Linux.
- **Python:** Versión 3.12.0 instalada.
- **Bibliotecas:**
 - NumPy 1.26.4: `pip install numpy==1.26.4`
 - Matplotlib 3.8.0: `pip install matplotlib==3.8.0`
 - Tkinter: Incluido con Python 3.12.0.
- **Archivo de Datos:** Descargar `wdbc.data` desde [Descargar wdbc.data](#) y colocarlo en el mismo directorio que la aplicación .

Instrucciones de Uso

1. **Preparación:**
 - Asegúrate de tener Python 3.12.0 instalado.
 - Instala las bibliotecas requeridas ejecutando los comandos de instalación mencionados.
 - Descarga el archivo `wdbc.data` y colócalo en el directorio del script.
2. **Ejecutar la Aplicación:**

WINDOWS

- clonar el repositorio o descarga el repositorio:
<https://github.com/Jhony-sudo2/RedesNeuronales>
- Ingresa a la carpeta `./dist`
- Ejecuta el archivo `main.exe`

LINUX /MAC

- Abre la carpeta donde descargaste el repositorio.
- Abre una terminal en ese directorio y ejecuta.

`python main.py.`

- Se abrirá una ventana gráfica con los controles de la aplicación.

3. **Interacción con la Interfaz:**

- Selecciona las características para los ejes X e Y (por ejemplo, `"radius_mean"` y `"texture_mean"`).
- Haz clic en "Visualizar Datos" para ver un scatter plot de los datos originales.
- Configura los parámetros: tasa de aprendizaje η , número de épocas, y porcentaje de datos de entrenamiento.
- Haz clic en "Entrenar Perceptrón" para entrenar el modelo y visualizar:
 - Un gráfico de error vs. épocas.

- La frontera de decisión con los datos de prueba.

4. **Interpretación de Resultados:**

- La exactitud en los datos de prueba se muestra en un mensaje emergente.
- Los gráficos permiten analizar visualmente la separación entre clases (benigno y maligno).