Simulación del Perceptrón con Interfaz Gráfica – MESA Python

Introducción

El perceptrón es uno de los algoritmos más básicos de la inteligencia artificial y el aprendizaje automático. Se trata de un modelo matemático inspirado en las neuronas del cerebro humano. Su objetivo es clasificar datos en dos categorías (por ejemplo, puntos de dos colores) mediante una línea recta llamada frontera de decisión

En este trabajo se implementó un modelo del perceptrón en Python utilizando el framework Mesa 1.1.1 para simular agentes y un entorno visual con Tkinter y Matplotlib
La aplicación cuenta con una interfaz gráfica interactiva, en la que el usuario puede ajustar la tasa de aprendizaje y el número máximo de iteraciones para observar cómo el perceptrón aprende en tiempo real

Funcionamiento del Perceptrón

El perceptrón realiza su tarea mediante el siguiente proceso:

1. **Inicialización de pesos y sesgo (bias) con valores aleatorios**
   Estos parámetros determinan la orientación y posición de la línea que separa las clases
2. **Predicción:**
   Para cada punto de entrenamiento (x, y), el perceptrón calcula:

$$activación = w_1 x + w_2 y + b$$

Si la activación es mayor o igual que 0, el punto se clasifica como 1; si no, como -1

3. **Ajuste de pesos:**
   Cuando el modelo se equivoca, actualiza sus parámetros:

$$w_i = w_i + tasa\_de\_aprendizaje \times error \times x_i$$

Este ajuste reduce el error en la siguiente iteración

4. **Entrenamiento repetido:**
   El modelo repite el proceso durante varias **épocas (iteraciones)** hasta clasificar correctamente todos los puntos o alcanzar el número máximo de iteraciones
5. **Visualización:**
   En la simulación, los puntos correctamente clasificados se muestran en **verde**, los incorrectos en **rojo**, y la **línea verde** representa la frontera de decisión que se actualiza en tiempo real

Implementación del Modelo

- **Entorno y agentes:**

- Se implementó un modelo *Mesa* con agentes de tipo DataPointAgent (los puntos de datos) y PerceptronAgent (el perceptrón que aprende)
- Cada punto tiene coordenadas (x, y) y una etiqueta (-1 o 1), generadas aleatoriamente, pero separables por una línea
- El perceptrón recorre los puntos, ajusta los pesos y actualiza el gráfico en cada paso

- **Interfaz gráfica:**

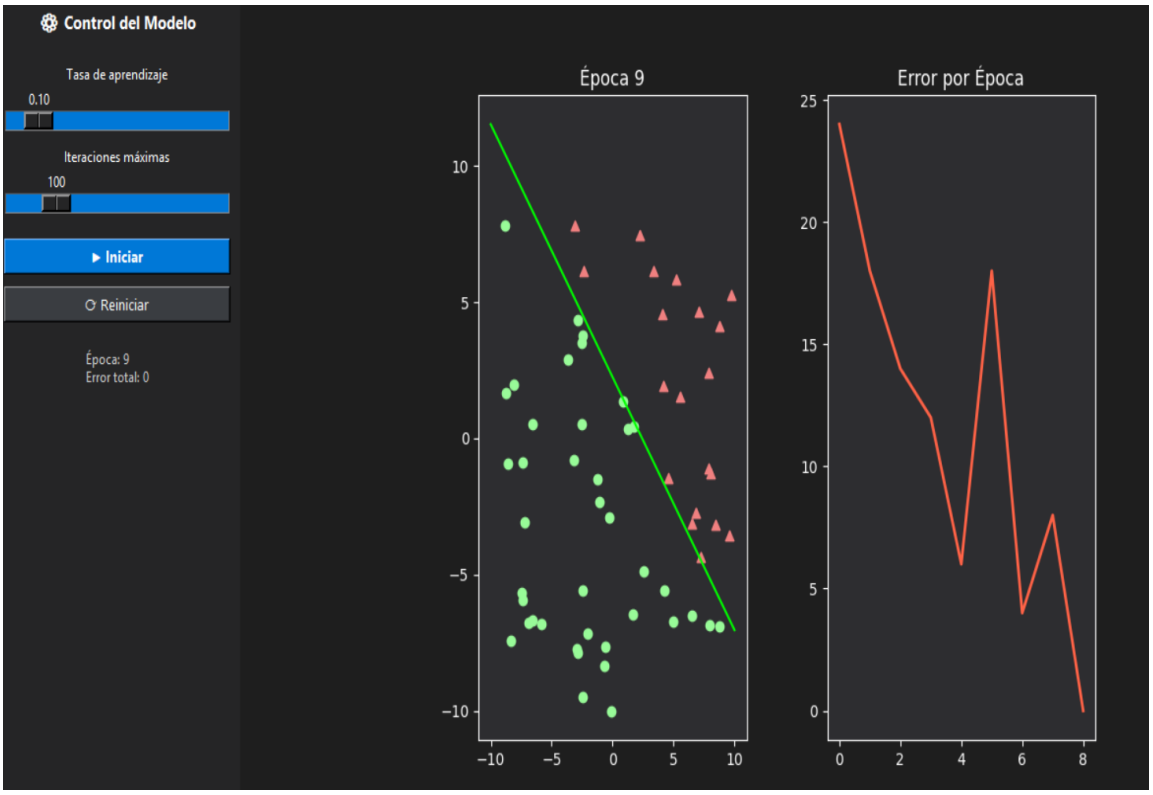
- Construida con **Tkinter**, con un diseño renovado distinto al original
- Incluye:
    - **Sliders** para modificar la tasa de aprendizaje y el número de iteraciones
    - **Botones** para iniciar o reiniciar la simulación
    - Dos gráficos:
        1. Distribución de puntos y línea de decisión
        2. Evolución del error por época

Resultados Obtenidos

Ejemplo 1:

Tasa de aprendizaje: 0.10
Iteraciones máximas: 100

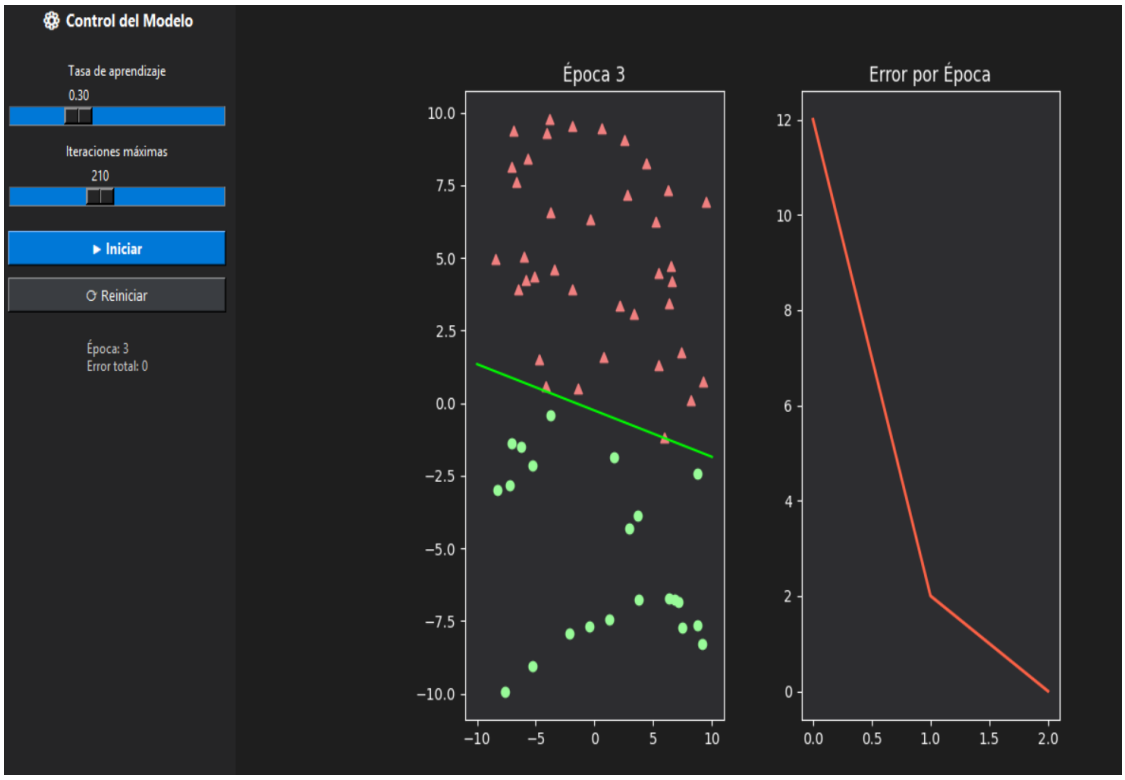
En este caso, el aprendizaje fue lento pero estable. El modelo necesitó varias épocas para ajustar correctamente la frontera de decisión, pero logró clasificar todos los puntos (error final = 0)
La línea verde se estabilizó con una pendiente negativa que separa correctamente ambas clases



Ejemplo 2:

Tasa de aprendizaje: 0.30
Iteraciones máximas: 210

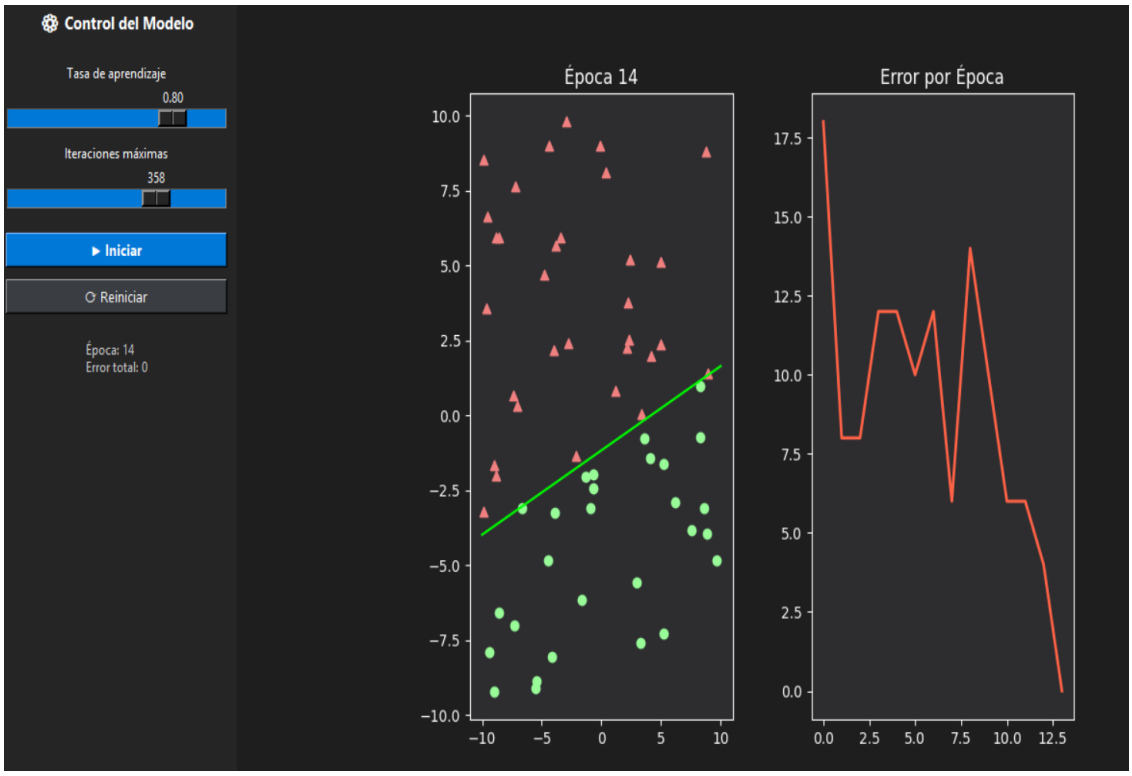
Con una tasa de aprendizaje mayor, el modelo **converge más rápido** (en pocas épocas). Sin embargo, los saltos en el gráfico de error muestran que las actualizaciones son más bruscas
A pesar de ello, también logra clasificar correctamente todos los puntos en menos iteraciones



Ejemplo 3:

Tasa de aprendizaje: 0.80
Iteraciones máximas: 358

En este experimento, la tasa de aprendizaje alta provoca un comportamiento **más oscilante**, lo que se refleja en el gráfico de error
Aunque el modelo eventualmente converge, la línea de separación fluctúa más antes de estabilizarse



Conclusiones

- El perceptrón permite aprender una frontera lineal que separa correctamente los datos cuando son linealmente separables.
- La tasa de aprendizaje influye directamente en la velocidad y estabilidad del entrenamiento:
    - Tasa baja = aprendizaje lento pero estable.
    - Tasa alta = aprendizaje rápido, pero menos estable.
- La simulación con Mesa permitió visualizar el proceso en tiempo real, reforzando la comprensión de cómo el perceptrón ajusta sus parámetros para reducir el error