Practica 1. Ejercicio 1

Morales Zepeda Ivan Yutlanih

Sem. de Sol. de Problemas de Inteligencia Artificial II

Sección: D05

Contenido

Introducción.

El código implementa un perceptrón simple, una unidad básica en las redes neuronales, para realizar operaciones lógicas como OR y XOR. En general, un perceptrón es un modelo simple que puede aprender a realizar clasificaciones lineales.

Desarrollo.

La implementación de este perceptrón se puede dividir en 3 fases principales:

1. **Definición de Funciones:**
    a. La función *step_function* define una función de activación de escalón que devuelve el signo de la entrada, utilizada para la predicción del perceptrón.
    b. La función *predict* utiliza los pesos y el sesgo para predecir la salida del perceptrón aplicando la función de activación de escalón.
    c. La función *read_data* lee los datos desde un archivo CSV y devuelve las entradas y salidas por separado.
    d. La función *train_perceptron* implementa el entrenamiento del perceptrón utilizando la regla de actualización de pesos y sesgo basada en el error.
    e. La función *test_perceptron* predice las salidas del perceptrón para nuevos datos de entrada utilizando los pesos y el sesgo entrenados.
    f. La función *_plot* se encarga de visualizar los datos de entrada, las clases y la frontera de decisión aprendida por el perceptrón.
2. **Entrenamiento y Prueba:**

    Se especifican los parámetros de entrenamiento como la tasa de aprendizaje, el número máximo de épocas y el criterio de convergencia. Los pesos y el sesgo se inicializan de manera aleatoria. En la ejecución se utiliza un bucle while para iterar hasta que se cumpla el criterio de convergencia o se alcance el número máximo de épocas. Dentro del bucle, se itera sobre los

patrones de entrada, calculando la predicción del perceptrón y actualizando los pesos y el sesgo según el error.

Se prueba el perceptrón con datos de prueba utilizando la función *test_perceptron*.

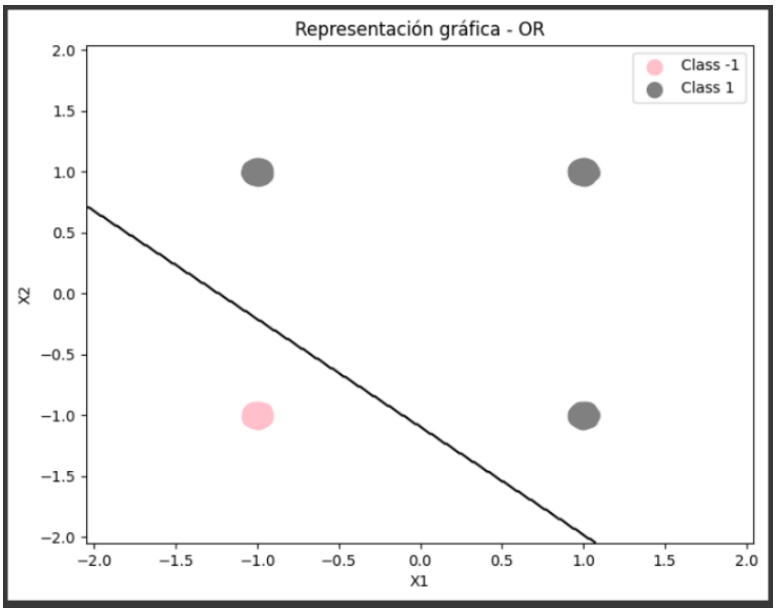
3. **Visualización Gráfica:**
   La función _plot se encarga de visualizar gráficamente los datos de entrenamiento, las clases y la frontera de decisión aprendida por el perceptrón. Se utilizó Matplotlib para generar un gráfico que muestra los puntos de las clases y la frontera de decisión.

Además, para la práctica, se realizó un pequeño script para leer los datos provenientes de un archivo .csv.

- **Lectura de Datos:**
  Las rutas de los archivos de datos se definen según el tipo de operación lógica (OR o XOR). Los datos de entrenamiento y prueba se leen utilizando la función *read_data* *(funcionalidad en:* **1. Definición de Funciones***)*.

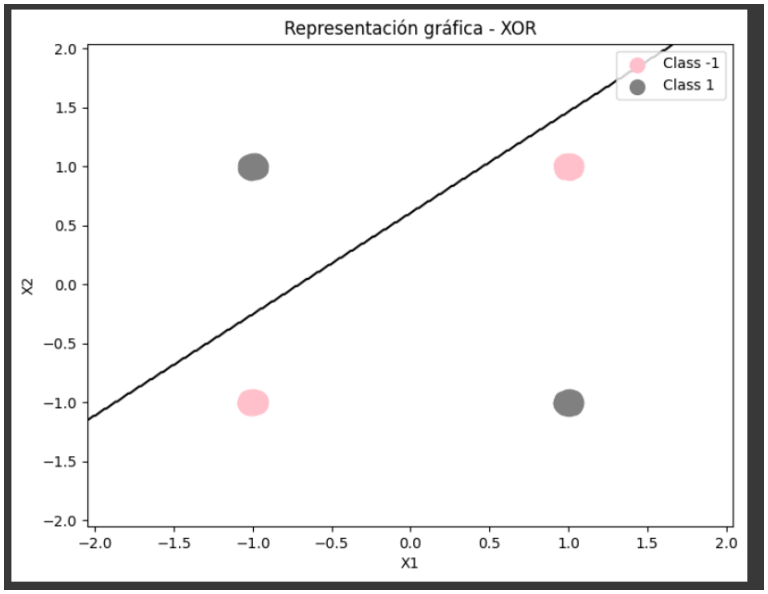
Resultados.

Representación gráfica - OR:



En este gráfico, se muestran las clases: “Class -1” (puntos rosados) y “Class 1” (puntos grises). La línea diagonal negra es la frontera de decisión del perceptrón. La clasificación parece ser efectiva, ya que la frontera de decisión separa las dos clases de manera razonable.

Representación gráfica - XOR (según la descripción):



En este resultado se esperaría ver una frontera de decisión más compleja debido a la naturaleza no lineal del problema XOR. La frontera de decisión debería ser una curva o una combinación de líneas rectas que separe las dos clases correctamente. Sin embargo, el perceptrón simple solo es capaz de clasificar problemas de naturaleza lineal.

En general, parece que el perceptrón está funcionando correctamente para clasificar los datos en operaciones lineales. La visualización de las fronteras de decisión es útil para comprender cómo el modelo está tomando decisiones. Por otra parte, al presentarse problemas de naturaleza no lineal, este perceptrón no es la opción indicada.

Conclusiones.

En conclusión, el código proporciona una implementación básica de un perceptrón y demuestra su capacidad para aprender y realizar operaciones lógicas. Sin embargo, es importante destacar que este perceptrón puede no ser suficiente para problemas más complejos (como el XOR), especialmente aquellos que no son linealmente separables. La eficacia del perceptrón depende de la naturaleza del problema y la capacidad del modelo para representar la relación entre las entradas y las salidas. Para tareas más desafiantes, se necesitarían enfoques más avanzados y arquitecturas de redes neuronales más complejas. Este código proporciona una introducción útil al concepto básico de un perceptrón y su aplicación en problemas sencillos de clasificación.

Enlace al repositorio de GitHub.

Link: IvanYMz/SSPIA2-Tareas (github.com)