SEMINARIO DE SOLUCION DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL 2

MAESTROS:

DIEGO ALBERTO OLIVA NAVARRO

DIEGO CAMPOS PENA

ALUMNO:

EDUARDO BLANCO GONZALEZ

Practica: 1

OBJETIVO:

Ejercicio 1: Realice un programa que permita el entrenamiento y prueba de un perceptrón simple con una cantidad variable de entradas. El programa debe realizar lo siguiente:

• Lectura de los patrones de entrenamiento (entradas y salidas) desde un archivo en formato texto separado por comas.

• Selección del criterio de finalización del entrenamiento.

• Selección del número máximo de épocas de entrenamiento.

• Selección de la tasa de aprendizaje.

• Prueba del perceptrón entrenado en datos reales.

Una vez que se generó el programa, debe ser probado considerando lo siguiente:

1. Problema XOR. Los patrones para este problema son los puntos (1,1), (1,-1), (-1,1), (-1,-1).

Además, deben considerarse alteraciones aleatorias (< 5%). Se debe generar un set de entrenamiento y otro de prueba. Utilizar los archivos XORtrn.csv y XORtst.csv

2. Mostrar gráficamente los patrones utilizado y la recta que los separa.

Perceptron

El perceptrón ocupa un lugar especial en el desarrollo histórico de las redes neuronales: fue la primera red neuronal descrita algorítmicamente. Su invención por Rosenblatt, un psicólogo, inspiró a ingenieros, físicos y matemáticos por igual a dedicar sus esfuerzos de investigación a diferentes aspectos de las redes neuronales en las décadas de 1960 y 1970. Además, es verdaderamente notable encontrar que el perceptrón (en su forma básica según se describe en este capítulo) es tan válido hoy como lo fue en 1958, cuando se publicó por primera vez el artículo de Rosenblatt sobre el perceptrón.

En los primeros años formativos de las redes neuronales (1943-1958), varios investigadores se destacan por sus contribuciones pioneras:

• McCulloch y Pitts (1943) por introducir la idea de redes neuronales como máquinas de computación.

• Hebb (1949) por postular la primera regla para el aprendizaje autoorganizado.

• Rosenblatt (1958) por proponer el perceptrón como el primer modelo para el aprendizaje con un maestro (es decir, aprendizaje supervisado).

El perceptrón de Rosenblatt se construye alrededor de una neurona no lineal, es decir, el modelo de neurona de McCulloch-Pitts. Recordamos que este modelado neuronal consiste en un combinador lineal seguido por un limitador duro (que realiza la función de signo). El nodo sumador del modelo neuronal calcula una combinación lineal de las entradas aplicadas a sus sinapsis, e incorpora un sesgo aplicado externamente. La suma resultante, es decir, el campo local inducido, se aplica a un limitador duro.

CODIGO:

3.py > ...

```python
import random
import numpy as np
import csv
import matplotlib.pyplot as plt

class Perceptron:
    def __init__(self, num_entradas, tasa_aprendizaje=0.1, max_epocas=1000):
        self.num_entradas = num_entradas
        self.tasa_aprendizaje = tasa_aprendizaje
        self.max_epocas = max_epocas
        self.pesos = np.random.rand(num_entradas + 1)  # +1 para el sesgo

    def predict(self, entrada):
        suma_ponderada = np.dot(self.pesos[1:], entrada) + self.pesos[0]  # Producto punto + sesgo
        return 1 if suma_ponderada > 0 else 0

    def train(self, datos_entrenamiento):
        for _ in range(self.max_epocas):
            errores = 0
            for entrada, salida in datos_entrenamiento:
                prediccion = self.predict(entrada)
                error = salida - prediccion
                if error != 0:
                    errores += 1
                    self.pesos[1:] += self.tasa_aprendizaje * error * np.array(entrada)
                    self.pesos[0] += self.tasa_aprendizaje * error
            if errores == 0:
                break

def leer_datos(archivo):
```

```python
def leer_datos(archivo):
    datos = []
    with open(archivo, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            entrada = list(map(float, row[:-1]))
            salida = int(row[-1])
            datos.append((entrada, salida))
    return datos

def mostrar_datos(datos, titulo):
    plt.figure()
    plt.title(titulo)
    for entrada, salida in datos:
        color = 'red' if salida == 1 else 'blue'
        plt.scatter(entrada[0], entrada[1], c=color)
    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.grid(True)
    plt.show()

# Lectura de datos de entrenamiento y prueba
datos_entrenamiento = leer_datos('XORtrn.csv')
datos_prueba = leer_datos('XORtst.csv')

# Mostrar datos de entrenamiento y prueba
mostrar_datos(datos_entrenamiento, 'Datos de Entrenamiento')
mostrar_datos(datos_prueba, 'Datos de Prueba')
```

```python

    # Entrenamiento del perceptrón
    perceptron = Perceptron(num_entradas=2)
    perceptron.train(datos_entrenamiento)

    # Prueba del perceptrón entrenado
    print("Predicciones después del entrenamiento:")
    for entrada, _ in datos_prueba:
        prediccion = perceptron.predict(entrada)
        print(f"Entrada: {entrada}, Predicción: {prediccion}")
```

FUNCION

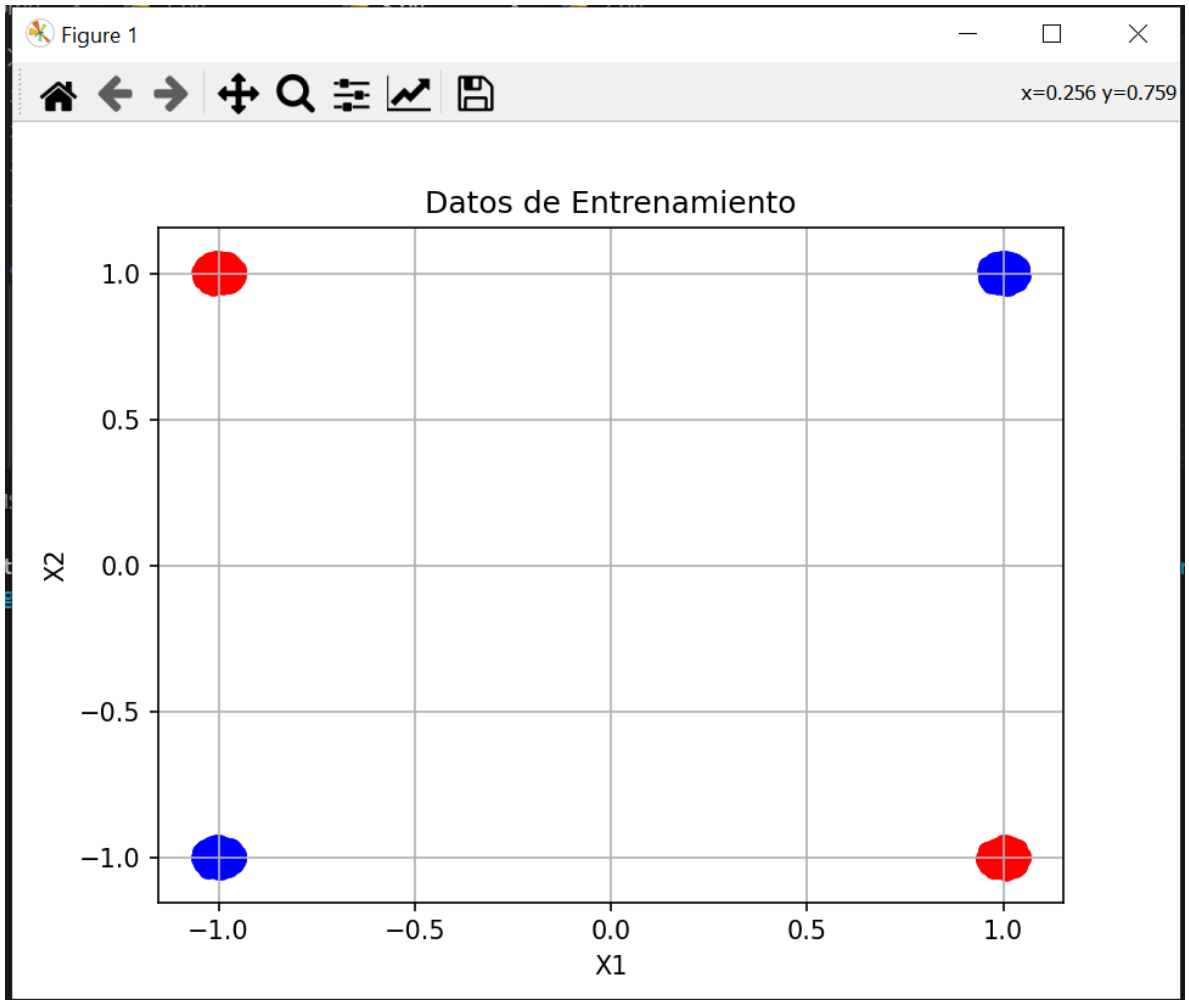


Figure 1
x=0.256 y=0.759
Datos de Entrenamiento
1.0
0.5
0.0
−0.5
−1.0
X2
−1.0
−0.5
0.0
0.5
1.0
X1

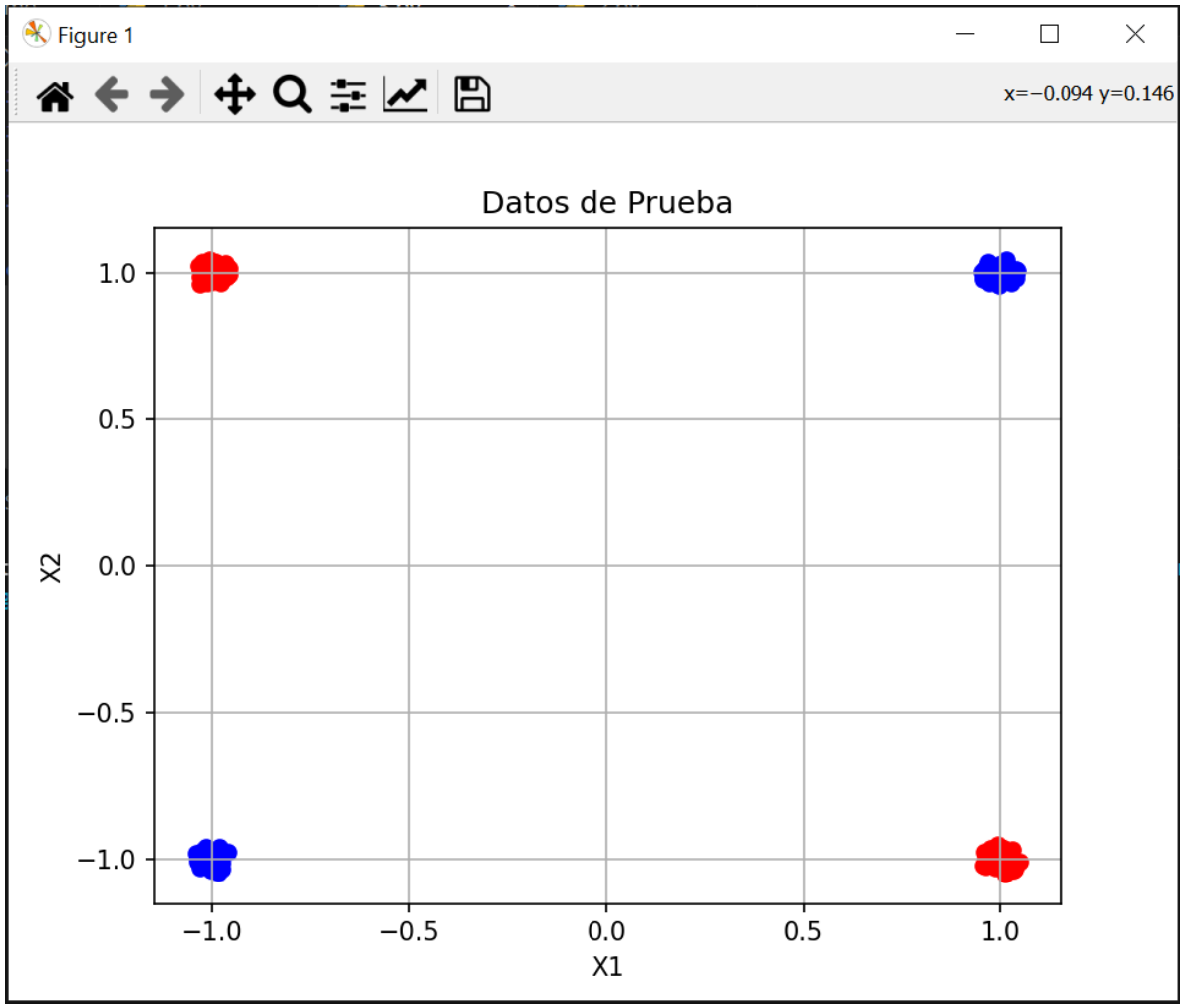


Figure 1
x=−0.094 y=0.146
Datos de Prueba
1.0
0.5
0.0
−0.5
−1.0
X2
−1.0
−0.5
0.0
0.5
1.0
X1

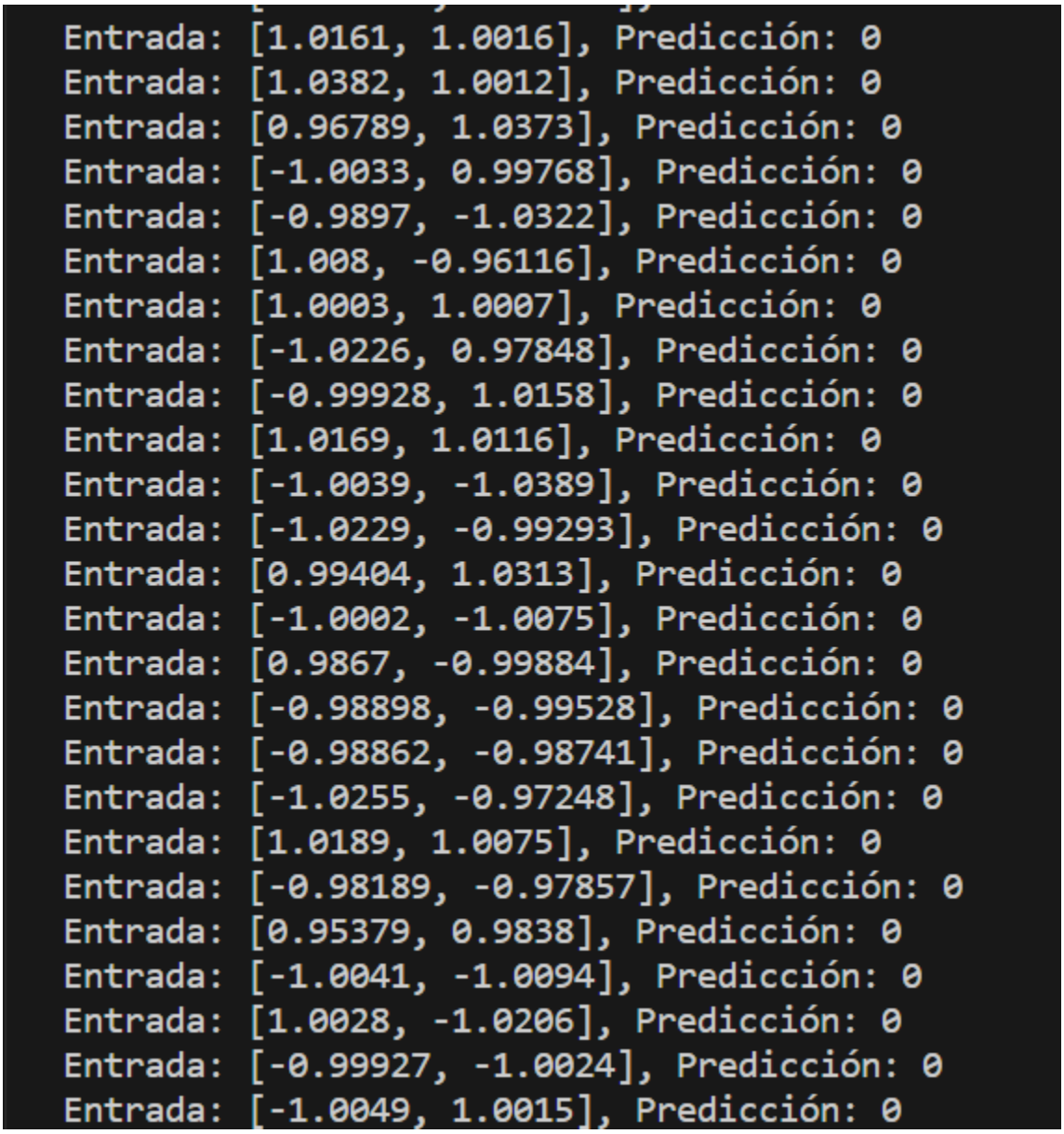
```
Predicciones después del entrenamiento:
Entrada: [1.0145, 1.045], Predicción: 0
Entrada: [-0.98456, 1.01], Predicción: 0
Entrada: [-1.0008, -1.0004], Predicción: 0
Entrada: [-0.98678, -1.0454], Predicción: 0
Entrada: [-0.9766, -1.0326], Predicción: 0
Entrada: [0.97443, 1.0293], Predicción: 0
Entrada: [-1.0005, 0.9742], Predicción: 0
Entrada: [-1.0017, 0.99955], Predicción: 0
Entrada: [1.0419, 1.0094], Predicción: 0
Entrada: [-1.0238, 0.99646], Predicción: 0
Entrada: [-1.022, -0.99921], Predicción: 0
Entrada: [-1.0417, -0.9805], Predicción: 0
Entrada: [-0.97597, 1.0136], Predicción: 0
Entrada: [-0.9627, 1.0086], Predicción: 0
Entrada: [-0.98026, 0.98706], Predicción: 0
Entrada: [0.99096, 0.99962], Predicción: 0
Entrada: [0.96733, 1.0086], Predicción: 0
Entrada: [-0.99959, 0.99233], Predicción: 0
Entrada: [1.0391, -1.0227], Predicción: 0
Entrada: [-0.99903, -1.0024], Predicción: 0
Entrada: [-1.0034, 1.0245], Predicción: 0
Entrada: [1.027, 0.96597], Predicción: 0
Entrada: [-1.0129, 0.96639], Predicción: 0
Entrada: [0.99988, 1.0312], Predicción: 0
Entrada: [-1.0066, 1.0312], Predicción: 0
Entrada: [-0.95836, 0.99843], Predicción: 0
Entrada: [-0.97595, 0.97542], Predicción: 0
Entrada: [0.98515, 0.9966], Predicción: 0
Entrada: [1.0093, 0.97741], Predicción: 0
Entrada: [1.0126, 0.99643], Predicción: 0
Entrada: [0.97379, 1.0033], Predicción: 0
Entrada: [0.97067, -1.0068], Predicción: 0
Entrada: [-0.99733, -1.004], Predicción: 0
Entrada: [1.0008, -0.99102], Predicción: 0
Entrada: [0.99871, 1.0012], Predicción: 0
```

```
Entrada: [0.99871, 1.0012], Predicción: 0
Entrada: [0.97203, 0.96929], Predicción: 0
Entrada: [1.0004, 1.0029], Predicción: 0
Entrada: [1.0156, -0.99238], Predicción: 0
Entrada: [-1.0173, -0.96178], Predicción: 0
Entrada: [0.99966, 0.99375], Predicción: 0
Entrada: [1.0056, 0.97917], Predicción: 0
Entrada: [1.0232, -1.0325], Predicción: 0
Entrada: [-0.99849, -0.99734], Predicción: 0
Entrada: [0.97619, 1.0147], Predicción: 0
Entrada: [-0.97848, 1.0068], Predicción: 0
Entrada: [-1.0002, 1.0011], Predicción: 0
Entrada: [1.0039, -0.96532], Predicción: 0
Entrada: [-0.99799, 1.0021], Predicción: 0
Entrada: [0.96307, 0.98147], Predicción: 0
Entrada: [0.99713, 0.99697], Predicción: 0
Entrada: [-1.0305, 1.0114], Predicción: 0
Entrada: [-1.0081, -1.0119], Predicción: 0
Entrada: [-1.009, -1.0016], Predicción: 0
Entrada: [0.95604, 0.97864], Predicción: 0
Entrada: [1.01, -1.0489], Predicción: 1
Entrada: [1.0306, -0.96726], Predicción: 0
Entrada: [1.0251, -1.037], Predicción: 0
Entrada: [-0.99938, -0.99813], Predicción: 0
Entrada: [1.0199, 1.0133], Predicción: 0
Entrada: [0.96171, -1.0257], Predicción: 0
Entrada: [1.0291, 1.0106], Predicción: 0
Entrada: [1.0171, 0.98937], Predicción: 0
Entrada: [0.99663, 0.95812], Predicción: 0
Entrada: [-0.99787, 1.0167], Predicción: 0
Entrada: [-0.99584, -1.0131], Predicción: 0
Entrada: [0.98482, -1.0289], Predicción: 0
Entrada: [-1.0074, -1.0298], Predicción: 0
Entrada: [1.0036, 0.98929], Predicción: 0
Entrada: [1.0392, 0.9841], Predicción: 0
Entrada: [0.97038, 0.99223], Predicción: 0
```

```
Entrada: [-1.0074, -1.0298], Predicción: 0
Entrada: [1.0036, 0.98929], Predicción: 0
Entrada: [1.0392, 0.9841], Predicción: 0
Entrada: [0.97038, 0.99223], Predicción: 0
Entrada: [-1.0003, 0.9993], Predicción: 0
Entrada: [-0.97733, -1.0122], Predicción: 0
Entrada: [-1.0059, -0.99967], Predicción: 0
Entrada: [0.9524, 1.0042], Predicción: 0
Entrada: [-0.99499, 1.0369], Predicción: 0
Entrada: [-1.0021, -1.0136], Predicción: 0
Entrada: [-0.99393, 1.0266], Predicción: 0
Entrada: [0.99058, 1.0076], Predicción: 0
Entrada: [-0.97939, 0.9682], Predicción: 0
Entrada: [0.97028, 0.97582], Predicción: 0
Entrada: [-0.96046, -0.97613], Predicción: 0
Entrada: [1.0028, 1.0218], Predicción: 0
Entrada: [-0.95812, 1.0156], Predicción: 0
Entrada: [1.0011, -0.99662], Predicción: 0
Entrada: [0.98374, 0.99504], Predicción: 0
Entrada: [-1.0267, 1.0022], Predicción: 0
Entrada: [0.9836, -0.98333], Predicción: 0
Entrada: [0.97305, -0.9631], Predicción: 0
Entrada: [1.0175, 1.0052], Predicción: 0
Entrada: [1.0309, -1.0307], Predicción: 0
Entrada: [-0.98385, -0.95758], Predicción: 0
Entrada: [1.0041, -0.9642], Predicción: 0
Entrada: [1.002, -0.9911], Predicción: 0
Entrada: [-0.9698, 1.0124], Predicción: 0
Entrada: [-0.99868, -0.99267], Predicción: 0
Entrada: [-1.0246, 1.0387], Predicción: 0
Entrada: [-1.0054, 1.0073], Predicción: 0
Entrada: [-0.98991, -1.0437], Predicción: 0
Entrada: [-0.98323, -0.98736], Predicción: 0
Entrada: [0.99094, 0.96282], Predicción: 0
Entrada: [-1.0179, -0.95805], Predicción: 0
Entrada: [1.0005, 0.99944], Predicción: 0
```

```
Entrada: [1.0035, -0.9937], Predicción: 0
Entrada: [-1.0322, 0.96329], Predicción: 0
Entrada: [0.97306, -1.0142], Predicción: 0
Entrada: [1.009, 1.0212], Predicción: 0
Entrada: [1.0, 1.0117], Predicción: 0
Entrada: [0.97828, 0.97704], Predicción: 0
Entrada: [-0.99097, 1.0389], Predicción: 0
Entrada: [-1.0011, -1.0051], Predicción: 0
Entrada: [-0.99796, 1.0105], Predicción: 0
Entrada: [-1.0044, -1.0356], Predicción: 0
Entrada: [0.96224, -1.0012], Predicción: 0
Entrada: [-1.0279, -0.98428], Predicción: 0
Entrada: [-0.96465, 0.98999], Predicción: 0
Entrada: [0.95592, -1.0227], Predicción: 0
Entrada: [0.96459, 0.98882], Predicción: 0
Entrada: [0.96945, 0.98498], Predicción: 0
Entrada: [-0.99121, 0.98879], Predicción: 0
Entrada: [1.0345, -1.0343], Predicción: 0
Entrada: [-0.9956, -0.99193], Predicción: 0
Entrada: [1.0003, 0.99954], Predicción: 0
Entrada: [1.0403, 1.0132], Predicción: 0
Entrada: [-0.98114, 1.0074], Predicción: 0
Entrada: [0.98139, -1.0102], Predicción: 0
Entrada: [-1.0025, 0.97069], Predicción: 0
Entrada: [-1.0341, -0.99184], Predicción: 0
Entrada: [-0.99049, 1.0149], Predicción: 0
Entrada: [1.0151, 1.0329], Predicción: 0
Entrada: [-1.0085, 0.99511], Predicción: 0
Entrada: [-1.0122, 1.0007], Predicción: 0
Entrada: [1.02, 1.0077], Predicción: 0
Entrada: [-1.0007, 0.99945], Predicción: 0
Entrada: [-0.96788, 1.032], Predicción: 0
Entrada: [0.95741, 1.0127], Predicción: 0
Entrada: [-1.0204, 1.0155], Predicción: 0
Entrada: [1.0185, 1.0272], Predicción: 0
Entrada: [-0.99688, -1.0043], Predicción: 0
```

```
Entrada: [-1.0319, -1.0311], Predicción: 0
Entrada: [-0.99362, -1.0039], Predicción: 0
Entrada: [-0.99525, -1.0014], Predicción: 0
Entrada: [-0.97976, 1.0189], Predicción: 0
Entrada: [-1.036, -0.98634], Predicción: 0
Entrada: [1.0048, 1.0014], Predicción: 0
Entrada: [0.96407, 1.0258], Predicción: 0
Entrada: [-1.0057, 1.048], Predicción: 0
Entrada: [-1.0004, -1.0037], Predicción: 0
Entrada: [-1.0371, -0.99751], Predicción: 0
Entrada: [-0.98958, 1.0064], Predicción: 0
Entrada: [1.0105, 0.97503], Predicción: 0
Entrada: [-0.97167, 1.0111], Predicción: 0
Entrada: [0.99157, -0.95207], Predicción: 0
Entrada: [0.97637, -0.98091], Predicción: 0
Entrada: [-1.0356, 1.0243], Predicción: 0
Entrada: [1.0486, -1.0075], Predicción: 0
Entrada: [1.0084, 0.98276], Predicción: 0
Entrada: [-1.0044, -1.0308], Predicción: 0
Entrada: [-0.96638, 1.0194], Predicción: 0
Entrada: [-1.007, -1.0097], Predicción: 0
Entrada: [1.0321, -1.0382], Predicción: 0
Entrada: [-0.99614, 0.97495], Predicción: 0
Entrada: [0.97865, 1.0135], Predicción: 0
Entrada: [0.99324, -1.0129], Predicción: 0
Entrada: [0.96536, -1.021], Predicción: 0
Entrada: [0.99687, 1.0033], Predicción: 0
Entrada: [-1.0305, 0.9891], Predicción: 0
Entrada: [0.95803, -0.9764], Predicción: 0
Entrada: [-1.0375, -1.0104], Predicción: 0
Entrada: [0.97714, -1.0098], Predicción: 0
Entrada: [1.0098, -0.98666], Predicción: 0
Entrada: [-0.99752, -0.99964], Predicción: 0
Entrada: [1.0025, 1.0009], Predicción: 0
Entrada: [0.98223, -1.017], Predicción: 0
Entrada: [-0.99495, -0.9988], Predicción: 0
```



CONCLUSION

El programa desarrollado es un ejemplo de implementación de un perceptrón simple para la resolución del problema XOR. El perceptrón es una red neuronal artificial de una sola capa que puede utilizarse para la clasificación de patrones linealmente separables. En este caso, se utiliza para separar los puntos del espacio bidimensional del XOR.

El objetivo principal del programa es entrenar un perceptrón utilizando un conjunto de datos de entrenamiento y luego probar su capacidad de generalización utilizando un conjunto de datos de prueba. Esto se logra siguiendo los siguientes pasos:

Lectura de datos: Se leen los patrones de entrenamiento y prueba desde archivos CSV que contienen las entradas y las salidas esperadas.

Entrenamiento del perceptrón: Se entrena el perceptrón ajustando iterativamente los pesos y el sesgo para minimizar el error entre las salidas calculadas y las salidas esperadas. Se permite al usuario seleccionar el criterio de finalización del entrenamiento, el número máximo de épocas y la tasa de aprendizaje.

Prueba del perceptrón: Una vez entrenado el perceptrón, se prueba su rendimiento utilizando el conjunto de datos de prueba. Se calcula la precisión del perceptrón en la clasificación de los datos de prueba.

Visualización: Finalmente, se muestra una representación gráfica de los datos de entrenamiento y prueba, junto con la línea que separa las clases según lo determinado por el perceptrón.

El programa ilustra el proceso de entrenamiento y prueba de un perceptrón simple y su capacidad para resolver el problema del XOR, un problema clásico en el campo de las redes neuronales artificiales.