Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
Seminario de Solución de Problemas de Inteligencia Artificial II
Dr. Diego Oliva
Depto. De Ciencias Computacionales
Practica 1
Perceptrón Simple y Multicapa

Mercado Manzo Luis Alfonso

Código:212559704

Objetivo:

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.

Introducción:

Un perceptrón es un modelo de clasificación binaria que se utiliza para separar dos clases de datos mediante un hiperplano en un espacio n-dimensional. Es un tipo de neurona artificial que toma un conjunto de entradas ponderadas, las suma y luego aplica una función de activación para producir una salida. La función de activación comúnmente utilizada en un perceptrón es una función de paso que devuelve un valor de 1 si la suma de las entradas ponderadas es mayor o igual a un umbral, y 0 en caso contrario.

Codigo:

```
# -*- coding: utf-8 -*-
"""

Created on Wed Sep  6 14:30:48 2023


@author: luis mercado
"""
```

```
import numpy as np
import pandas as pd
from sklearn.linear_model import Perceptron
import matplotlib.pyplot as plt

# Lectura de los patrones de entrenamiento desde un archivo en formato texto separado por comas
datos_entrenamiento = pd.read_csv('XOR_trn.csv', header=None)

# Extracción de las entradas y las salidas de los datos de entrenamiento
entradas_entrenamiento = datos_entrenamiento.iloc[:, :-1].values
salidas_entrenamiento = datos_entrenamiento.iloc[:, -1].values

# Definición de parámetros
max_epocas = 100  # Número máximo de épocas (ajusta según tus necesidades)
tasa_aprendizaje = 0.1  # Tasa de aprendizaje (ajusta según tus necesidades)

# Creación y entrenamiento del perceptrón
perceptron = Perceptron(max_iter=max_epocas, eta0=tasa_aprendizaje)
perceptron.fit(entradas_entrenamiento, salidas_entrenamiento)

# Lectura de los patrones de prueba desde un archivo en formato texto separado por comas
datos_prueba = pd.read_csv('XOR_tst.csv', header=None)
```

```
# Extracción de las entradas y las salidas de los datos de prueba
entradas_prueba = datos_prueba.iloc[:, :-1].values
salidas_prueba = datos_prueba.iloc[:, -1].values

# Prueba del perceptrón entrenado en los datos de prueba
salidas_predecidas = perceptron.predict(entradas_prueba)

# Mostrar resultados
print('Salidas reales vs. Salidas predecidas:')
print(np.column_stack((salidas_prueba, salidas_predecidas)))

# Mostrar gráficamente los patrones y la recta que los separa
plt.scatter(entradas_entrenamiento[:, 0], entradas_entrenamiento[:, 1],
c=salidas_entrenamiento, cmap='bwr')
plt.scatter(entradas_prueba[:, 0], entradas_prueba[:, 1],
c=salidas_predecidas, cmap='bwr', marker='x')
x = np.linspace(-2, 2, 100)
y = (-perceptron.coef_[0][0] * x - perceptron.intercept_) /
perceptron.coef_[0][1]
plt.plot(x, y, color='black', linewidth=2)
plt.legend(['Recta separadora', 'Patrones de entrenamiento', 'Patrones de
prueba'])
plt.xlabel('x')
plt.ylabel('y')
plt.title('Patrones y Recta Separadora')
```

plt.show()

resultados:

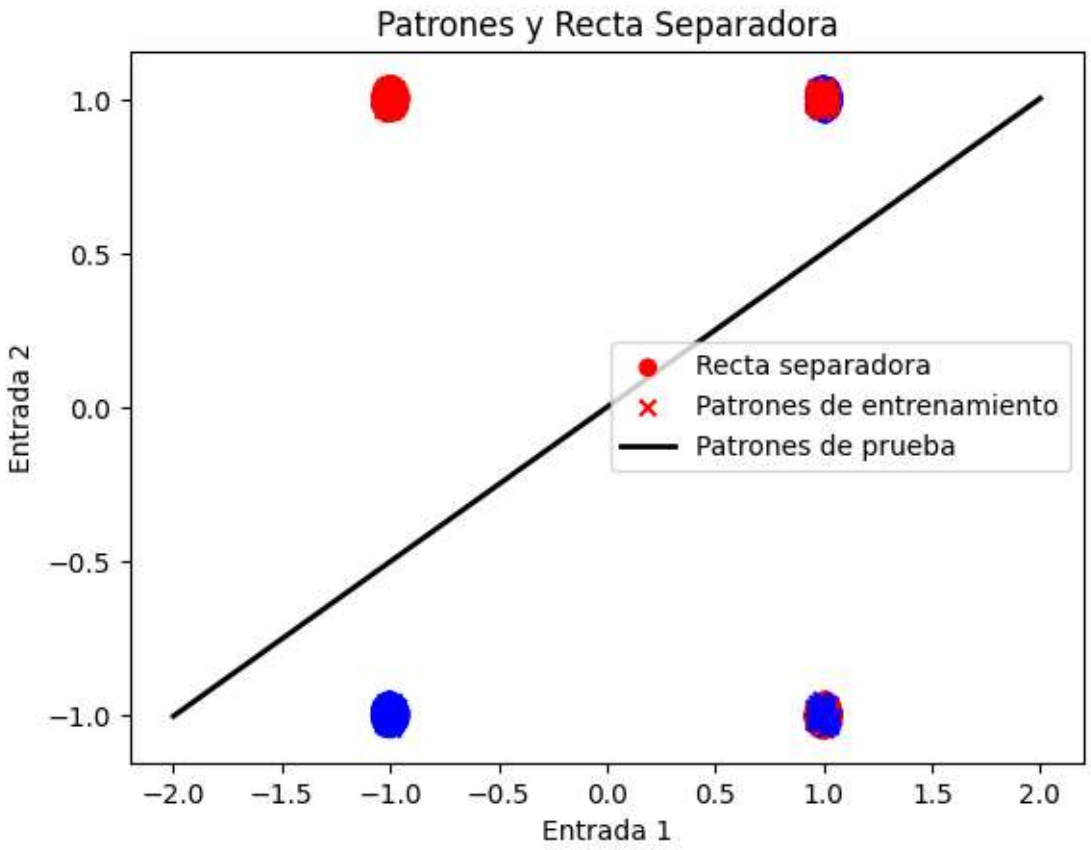
```
[[-1  1]
 [ 1  1]
 [-1 -1]
 [-1 -1]
 [-1 -1]
 [-1  1]
 [ 1  1]
 [ 1  1]
 [-1  1]
 [ 1  1]
 [-1 -1]
 [-1 -1]
 [ 1  1]
 [ 1  1]
 [ 1  1]
 [-1  1]
 [-1  1]
 [ 1  1]
 [ 1 -1]
 [-1 -1]
 [ 1  1]
 [-1  1]
 [ 1  1]
 [-1  1]
 [ 1  1]
 [ 1  1]
 [ 1  1]
 [-1  1]
 [-1  1]
 [-1  1]
 [-1  1]
 [ 1 -1]
 [-1 -1]
 [ 1 -1]
 [-1  1]
 [-1  1]
 [-1  1]
 [ 1 -1]
 [-1 -1]
 [-1  1]
 [-1  1]
 [ 1 -1]
 [-1 -1]
 [-1  1]
 [ 1  1]
 [ 1  1]
 [ 1 -1]
 [ 1  1]
```

```
[-1  1]
[-1  1]
[ 1  1]
[-1 -1]
[-1 -1]
[-1  1]
[ 1 -1]
[ 1 -1]
[ 1 -1]
[-1 -1]
[-1  1]
[ 1 -1]
[-1  1]
[-1  1]
[-1  1]
[ 1  1]
[-1 -1]
[ 1 -1]
[-1 -1]
[-1  1]
[-1  1]
[-1  1]
[ 1  1]
[-1 -1]
[-1 -1]
[-1  1]
[ 1  1]
[-1 -1]
[ 1  1]
[-1  1]
[ 1  1]
[-1  1]
[-1 -1]
[-1  1]
[ 1  1]
[ 1 -1]
[-1  1]
[ 1  1]
[ 1 -1]
[ 1 -1]
[-1  1]
[ 1 -1]
[-1 -1]
[ 1 -1]
[ 1 -1]
[ 1  1]
[-1 -1]
[ 1  1]
[ 1  1]
[-1 -1]
[-1 -1]
[-1  1]
[-1 -1]
[-1  1]
```

```
[ 1 -1]
[ 1  1]
[ 1 -1]
[-1  1]
[-1  1]
[-1  1]
[ 1  1]
[-1 -1]
[ 1  1]
[-1 -1]
[ 1 -1]
[-1 -1]
[ 1  1]
[ 1 -1]
[-1  1]
[-1  1]
[ 1  1]
[ 1 -1]
[-1 -1]
[-1  1]
[-1  1]
[ 1  1]
[ 1 -1]
[ 1  1]
[-1 -1]
[ 1  1]
[-1  1]
[ 1  1]
[ 1  1]
[-1  1]
[ 1  1]
[ 1  1]
[-1  1]
[ 1  1]
[-1  1]
[-1 -1]
[-1 -1]
[-1 -1]
[-1 -1]
[ 1  1]
[-1 -1]
[-1  1]
[-1  1]
[ 1  1]
[-1 -1]
[-1 -1]
[ 1  1]
[-1  1]
[ 1  1]
[ 1 -1]
[ 1 -1]
[ 1  1]
[ 1 -1]
[-1  1]
```

```
 [-1 -1]
 [ 1  1]
 [-1 -1]
 [ 1 -1]
 [ 1  1]
 [-1  1]
 [ 1 -1]
 [ 1 -1]
 [-1  1]
 [ 1  1]
 [ 1 -1]
 [-1 -1]
 [ 1 -1]
 [ 1 -1]
 [-1 -1]
 [-1  1]
 [ 1 -1]
 [-1 -1]
 [-1  1]
 [-1  1]
 [-1  1]
 [ 1  1]
 [-1 -1]
 [ 1 -1]
 [-1  1]
 [ 1  1]
 [ 1  1]
 [-1  1]
 [-1 -1]
 [-1 -1]
 [-1  1]
 [-1 -1]
 [ 1 -1]
 [-1 -1]
 [-1  1]
 [-1 -1]
 [-1 -1]
 [-1  1]
 [-1 -1]
 [-1  1]
 [-1 -1]
 [ 1 -1]
 [-1 -1]
 [ 1  1]]
```

Grafico:



Conclusión:

El perceptrón tiene limitaciones. Es capaz de separar clases linealmente, pero no puede manejar problemas que no son linealmente separables. Sin embargo, estas limitaciones se pueden superar mediante la utilización de redes neuronales multicapa (MLP) que constan de múltiples capas de perceptrones interconectados, lo que permite abordar problemas más complejos.

En resumen, el perceptrón es un modelo de clasificación simple que utiliza un hiperplano para separar dos clases de datos. Aunque tiene limitaciones en términos de su capacidad para resolver problemas no linealmente separables, sienta las bases para modelos de redes neuronales más avanzados.