CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS (CUCEI)

DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Carrera: Ingeniería en Computación

Nombre Materia: Seminario de Solución de Problemas de IA ll

Profesor: Valdez López Julio Esteban

SECCIÓN: D02

Nombre alumno: López Arellano Ricardo David

CODIGO: 217136143

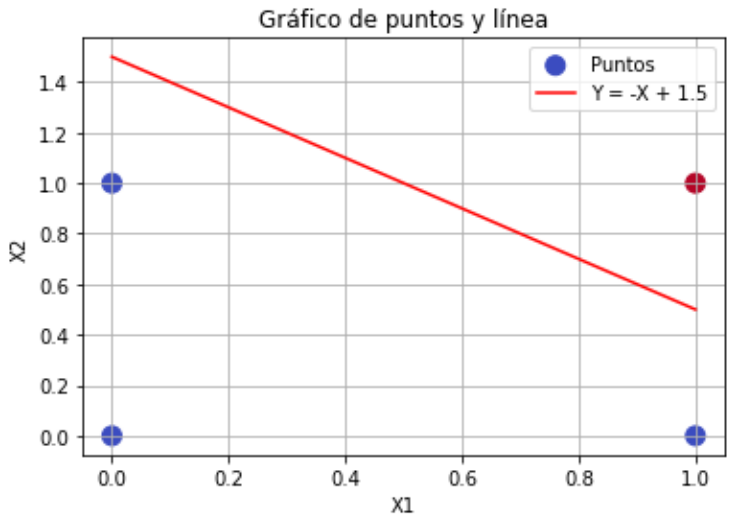
Practica 1

Fecha de entrega: 06/09/2023

CÓDIGO 1:

```
import matplotlib.pyplot as plt
import numpy as np
# Puntos
points = np.array([[0, 0, 0],
[0, 1, 0],
[1, 0, 0],
[1, 1, 1]])

x = points[:, 0:2]
y = points[:, 2]
# Línea
m = -1
c = 1.5
x_line = np.array([0, 1])
y_line = m * x_line + c
# Crear la figura y los ejes
plt.figure()
plt.scatter(x[:, 0], x[:, 1], c=y, cmap='coolwarm', marker='o', s=100,
label='Puntos')
plt.plot(x_line, y_line, color='red', label='Y = -X + 1.5')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Gráfico de puntos y línea')
plt.legend()
plt.grid(True)
plt.show()
```

RESULTADO:

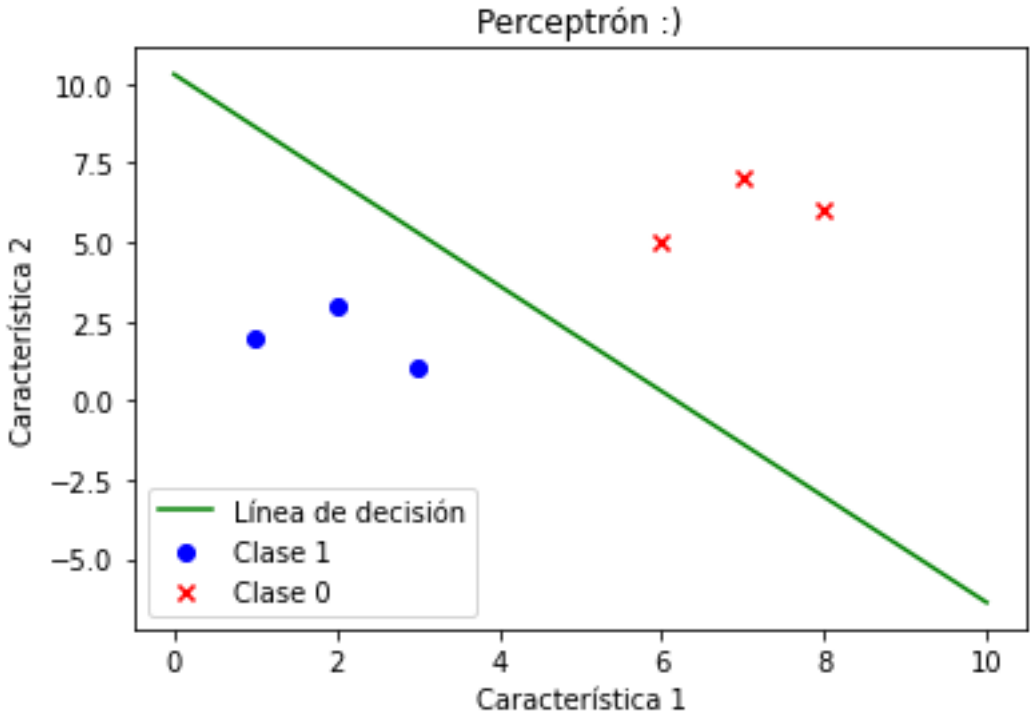


CÓDIGO 2:

```
import numpy as np
import matplotlib.pyplot as plt

# Definir la función de activación (función escalón)
def activation_function(x):
    return 1 if x >= 0 else 0

# Clase para la neurona Perceptrón
class Perceptron:
    def __init__(self, num_inputs, learning_rate=0.1):
        self.weights = np.random.rand(num_inputs)
        self.bias = np.random.rand()
        self.learning_rate = learning_rate

    def predict(self, inputs):
        weighted_sum = np.dot(self.weights, inputs) + self.bias
        return activation_function(weighted_sum)

    def train(self, training_data, epochs):
        for _ in range(epochs):
            for inputs, target in training_data:
                prediction = self.predict(inputs)
                error = target - prediction
                self.weights += self.learning_rate * error * inputs
                self.bias += self.learning_rate * error

# Datos de entrenamiento (dos clases)
training_data = [
    (np.array([1, 2]), 1),
    (np.array([2, 3]), 1),
    (np.array([3, 1]), 1),
    (np.array([6, 5]), 0),
    (np.array([7, 7]), 0),
    (np.array([8, 6]), 0)
]

# Crear un perceptrón con 2 entradas
perceptron = Perceptron(num_inputs=2)

# Entrenar el perceptrón durante 100 épocas
perceptron.train(training_data, epochs=100)

# Crear un rango de valores para la línea de decisión
x = np.linspace(0, 10, 100)
y = (-perceptron.weights[0] * x - perceptron.bias) /
perceptron.weights[1]
```

```
# Graficar los resultados
plt.plot(x, y, label="Línea", color='green')
plt.scatter(*zip(*[data[0] for data in training_data if data[1] == 1]),
color='b', marker='o', label='Clase 1')
plt.scatter(*zip(*[data[0] for data in training_data if data[1] == 0]),
color='r', marker='x', label='Clase 0')
plt.xlabel('Característica 1')
plt.ylabel('Característica 2')
plt.legend()
plt.title('Perceptrón ☺)
plt.show()
```

RESULTADO:



Los pesos los puse random asi que en cada ejecución saldrán diferentes resultados…