Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación
Seminario de Solución de Problemas de Inteligencia Artificial II
Profesor: Campos Peña Diego

Lepiz Hernández Manuel Alejandro
Código: 214797998

Práctica 1.

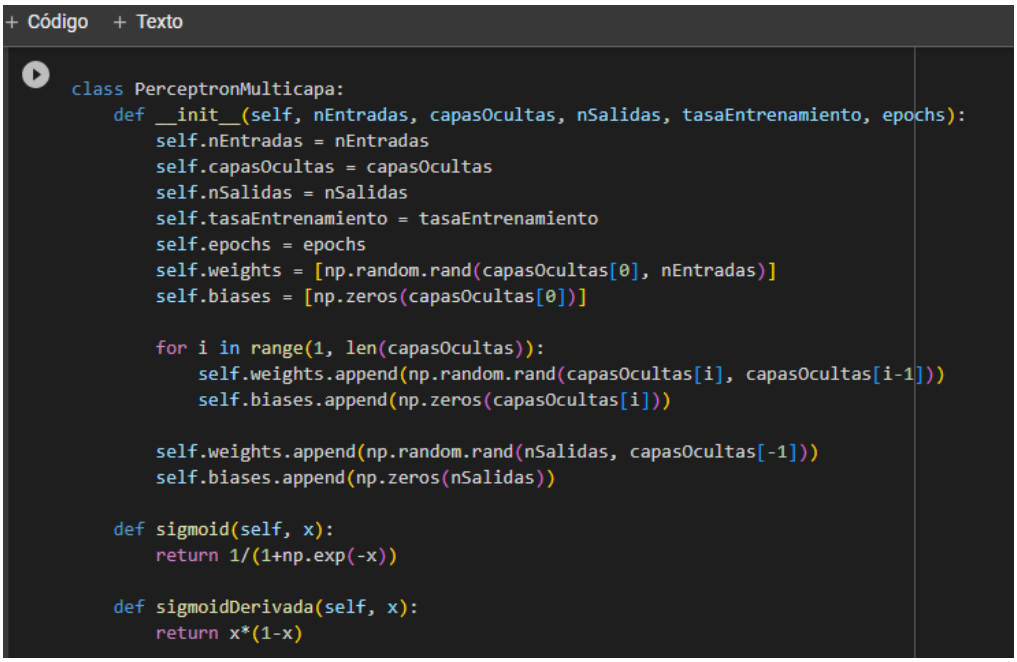
Ejercicio 3.

Introducción

Se realizará en Python la implementación del Algoritmo de retropropagación, usando un archivo .csv para la entrada de datos y un perceptrón multicapa para la resolución del algoritmo.

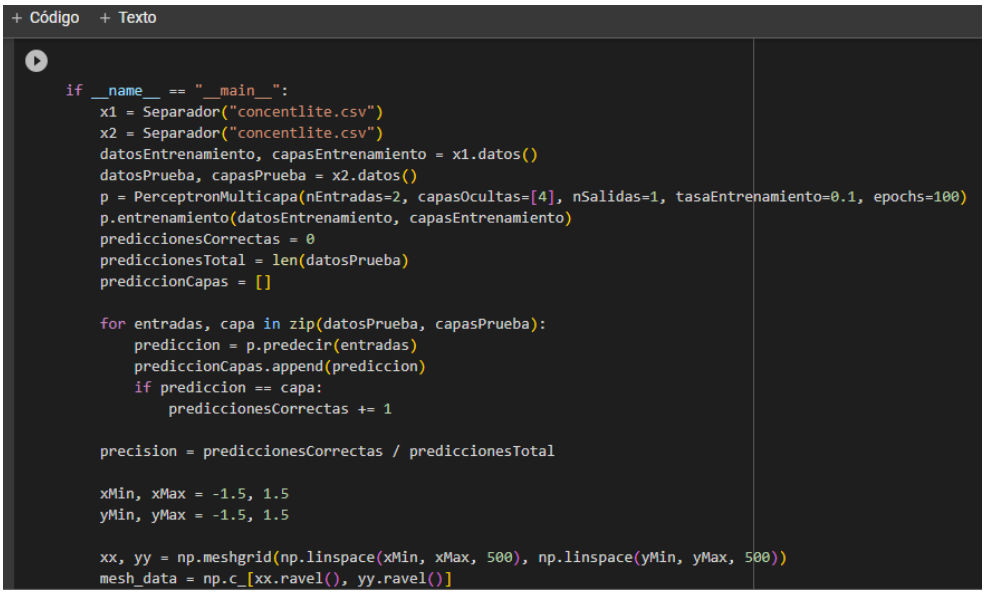
Desarrollo

Se hizo la implementación de una Clase para representar el perceptrón



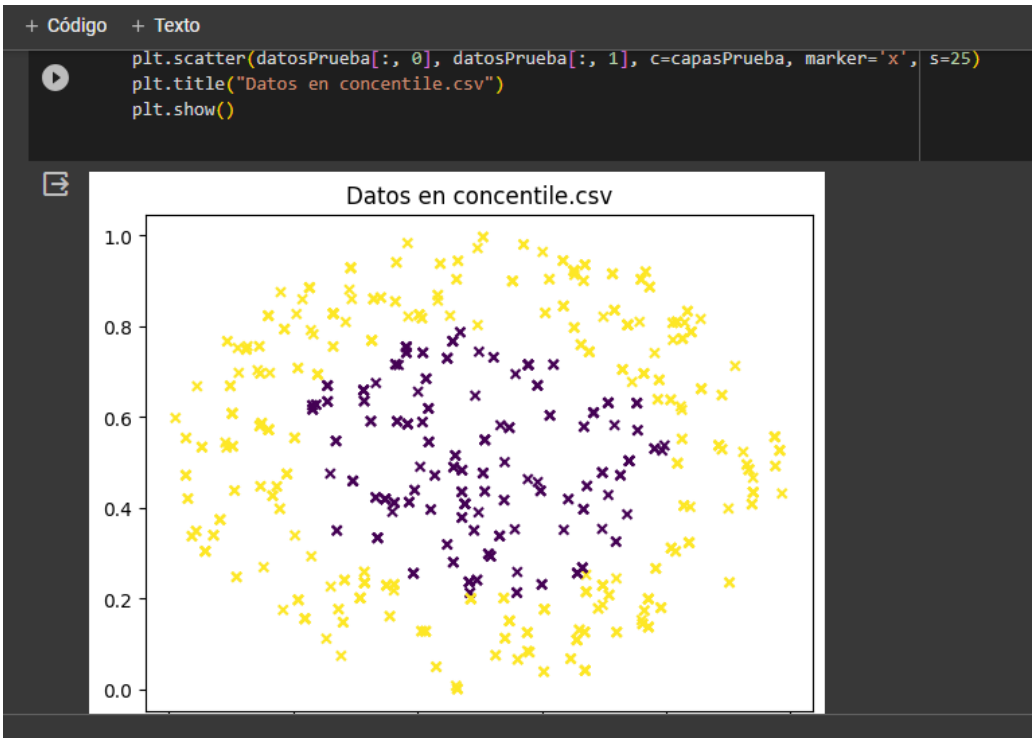
```
+ Código  + Texto

class PerceptronMulticapa:
    def __init__(self, nEntradas, capasOcultas, nSalidas, tasaEntrenamiento, epochs):
        self.nEntradas = nEntradas
        self.capasOcultas = capasOcultas
        self.nSalidas = nSalidas
        self.tasaEntrenamiento = tasaEntrenamiento
        self.epochs = epochs
        self.weights = [np.random.rand(capasOcultas[0], nEntradas)]
        self.biases = [np.zeros(capasOcultas[0])]

        for i in range(1, len(capasOcultas)):
            self.weights.append(np.random.rand(capasOcultas[i], capasOcultas[i-1]))
            self.biases.append(np.zeros(capasOcultas[i]))

        self.weights.append(np.random.rand(nSalidas, capasOcultas[-1]))
        self.biases.append(np.zeros(nSalidas))

    def sigmoid(self, x):
        return 1/(1+np.exp(-x))

    def sigmoidDerivada(self, x):
        return x*(1-x)
```

Se realizaron distintas pruebas usando el mismo set de datos, pero con distintos valores dados al perceptrón



```
+ Código  + Texto

if __name__ == "__main__":
    x1 = Separador("concentlite.csv")
    x2 = Separador("concentlite.csv")
    datosEntrenamiento, capasEntrenamiento = x1.datos()
    datosPrueba, capasPrueba = x2.datos()
    p = PerceptronMulticapa(nEntradas=2, capasOcultas=[4], nSalidas=1, tasaEntrenamiento=0.1, epochs=100)
    p.entrenamiento(datosEntrenamiento, capasEntrenamiento)
    prediccionesCorrectas = 0
    prediccionesTotal = len(datosPrueba)
    prediccionCapas = []

    for entradas, capa in zip(datosPrueba, capasPrueba):
        prediccion = p.predecir(entradas)
        prediccionCapas.append(prediccion)
        if prediccion == capa:
            prediccionesCorrectas += 1

    precision = prediccionesCorrectas / prediccionesTotal

    xMin, xMax = -1.5, 1.5
    yMin, yMax = -1.5, 1.5

    xx, yy = np.meshgrid(np.linspace(xMin, xMax, 500), np.linspace(yMin, yMax, 500))
    mesh_data = np.c_[xx.ravel(), yy.ravel()]
```

Se grafican los datos obtenidos al usar el perceptrón



Siendo bastante similar al resultado esperado

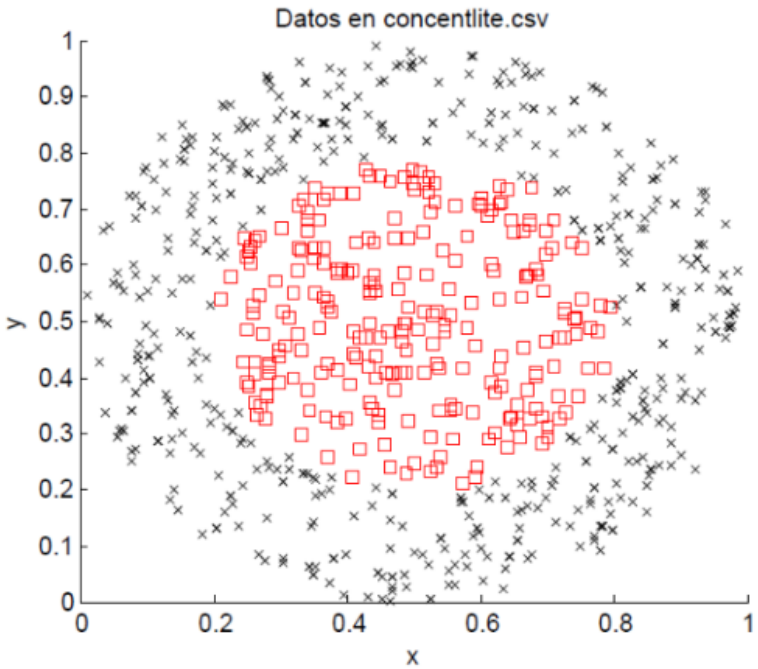


Figura 2. Distribución de clases para el dataset concentlite.

Conclusiones

El perceptrón fue capaz de hacer la separación correcta de los datos, siendo mostrados en la gráfica como sí se dividen los datos en distintas áreas.