Centro de Enseñanza Técnica Industrial

Plantel Colomos

Ingeniería en Desarrollo de Software

Inteligencia Artificial

REPORTE DE PRÁCTICA

IDENTIFICACIÓN DE LA PRÁCTICA

Práctica	**4**	**Nombre de la práctica**	**El perceptrón**
Fecha	**01/04/2025**	**Nombre del profesor**	**Alma Nayeli Rodríguez Vázquez**
Nombre del estudiante		**Jesús Alberto Aréchiga Carrillo**	

OBJETIVO

El objetivo de esta práctica consiste en implementar el método del perceptrón para clasificación binaria.

PROCEDIMIENTO

Realiza la implementación siguiendo estas instrucciones.

Implementa el método del perceptrón en Python. Para ello, considera los siguientes requerimientos:

- Utiliza el set de datos del archivo “dataset_Perceptron.csv”.
- Utiliza los siguientes valores para los parámetros iniciales:
  `a=0.03, b=-0.5, W=[0.2,0.5]`
- Reporta el valor final de `W` y `b`. Además, la clasificación estimada para los siguientes datos de prueba:

  Dato de prueba 1: `x1=34.6237, x2=78.0247`
  Dato de prueba 2: `x1=60.1826, x2=86.3086`

- Comprueba tus resultados con los siguientes:

```
W1= 0.40913, W2= 0.45486, b= 0.13

Dato de prueba 1: x1=34.6237, x2=78.0247. Clase correcta: clase 0.
Predicción: clase 0

Dato de prueba 2: x1=60.1826, x2=86.3086. Clase correcta: clase 1.
Predicción: clase 1
```

Inteligencia Artificial

IMPLEMENTACIÓN

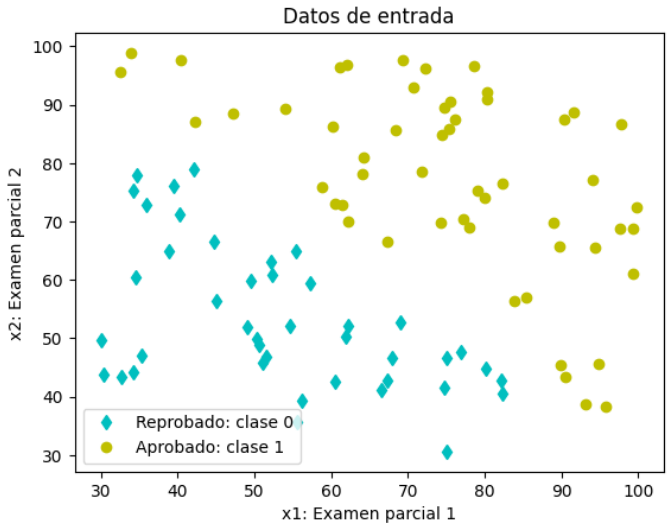
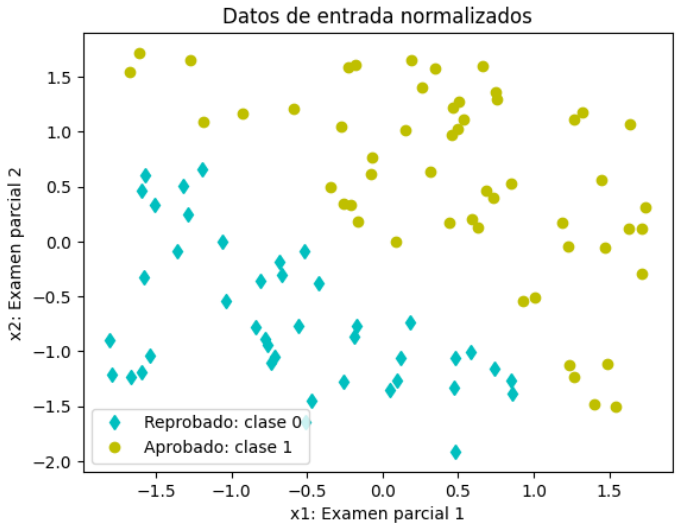
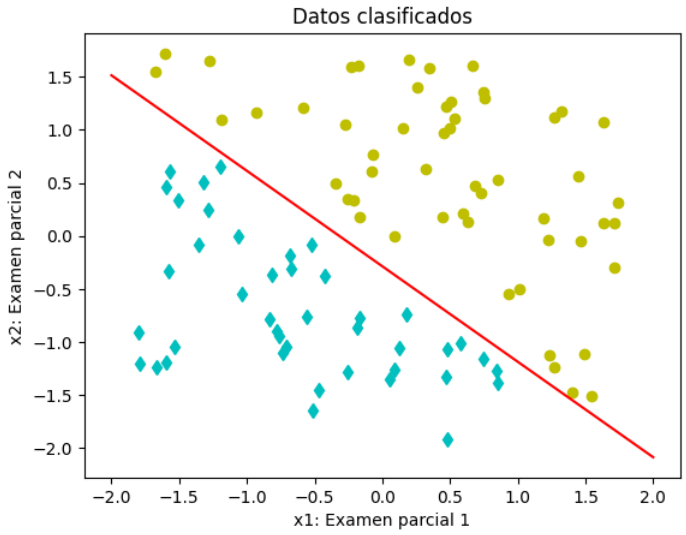
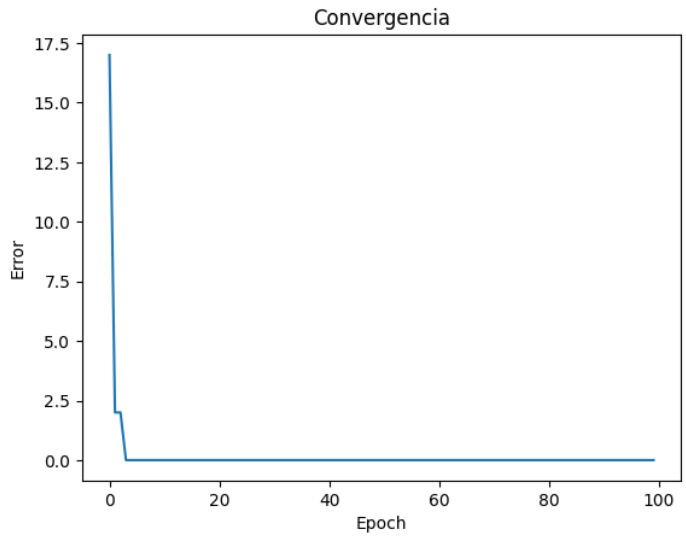
Agrega el código de tu implementación aquí.

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

data = pd.read_csv('dataset_Perceptron.csv')

data.head()

# Separar los datos en X y Y
m,n = data.shape
array = data.values
x = array[:,0:n-1]
Y = array[:,n-1]

# Dibujar los datos para visualizarlos (grafico de dispersion)
class0 = np.argwhere(Y==0)
class1 = np.argwhere(Y==1)

class0 = class0[:, 0]
plt.plot(x[class0, 0], x[class0, 1], 'dc', label = 'Reprobado: clase 0')
plt.plot(x[class1, 0], x[class1, 1], 'oy', label = 'Aprobado: clase 1')
plt.xlabel('x1: Examen parcial 1')
plt.ylabel('x2: Examen parcial 2')
plt.legend()
plt.show()

# Normalizar datos
mean = x.mean(axis = 0)
std = x.std(axis = 0, ddof = 1)
x = (x - mean)/std

# Inicializar parametros
a = 0.03
b = -0.5
w = np.array([0.2, 0.5])
epochMax = 100

# Entrenamiento
conv = np.zeros(epochMax)
for epoch in range(epochMax):
  error = 0
  for i in range(m):
    v = np.dot(w, x[i,:]) + b
    if  v > 0:
      y = 1
    else:
      y = 0
    if Y[i] != y:
      w = w + a * x[i,:]
      b = b + a
      error += 1
```

```
  conv[epoch] = error

# Dibujar convergencia
plt.plot(conv)
plt.xlabel('Epoch')
plt.ylabel('Error')
plt.show()

#Imprimir los resultados
print('w = ', w)
print('b = ', b)

#Dibugar clasificación final
plt.plot(x[class0, 0], x[class0, 1], 'dc', label = 'Reprobado: clase 0')
plt.plot(x[class1, 0], x[class1, 1], 'oy', label = 'Aprobado: clase 1')

x1 = np.linspace(-2,2,2)
x2 = (-w[0] * x1 -b) / w[1]
plt.plot(x1, x2, 'r')

plt.xlabel('x1: Examen parcial 1')
plt.ylabel('x2: Examen parcial 2')

#Prueba 1
prueba1 = np.array([34.6237, 78.0247])
prueba1 = (prueba1 - mean) / std

v = np.dot(w, prueba1) + b
if  v > 0:
  y = 1
  print('Clase 1')
else:
  y = 0
  print('Clase 0')

#Prueba 2
prueba2 = np.array([60.1826, 86.3086])
prueba2 = (prueba2 - mean) / std

v = np.dot(w, prueba2) + b
if  v > 0:
  y = 1
  print('Clase 1')
else:
  y = 0
  print('Clase 0')
```

RESULTADOS

Agrega la(s) imagen(es) con los resultados obtenidos en Matlab en los espacios indicados.

Gráfica de los datos	Gráfica de los datos normalizados
	
Gráfica de los datos clasificados	**Gráfica de convergencia**
	

Resultados en la que se despliegue el valor de `W`, el valor de `b`, los datos de prueba, su clase correcta, y la clasificación estimada para los datos de prueba.

```
w =  [0.40912514 0.45485556]
b =  0.1300000000000002
```

```
#Prueba 1
prueba1 = np.array([34.6237, 78.0247])
prueba1 = (prueba1 - mean) / std

v = np.dot(w, prueba1) + b
if  v > 0:
  y = 1
  print('Clase 1')
else:
  y = 0
  print('Clase 0')
```

```
Clase 0
```

```
#Prueba 2
prueba2 = np.array([60.1826, 86.3086])
prueba2 = (prueba2 - mean) / std

v = np.dot(w, prueba2) + b
if  v > 0:
  y = 1
  print('Clase 1')
else:
  y = 0
  print('Clase 0')
```

```
Clase 1
```

CONCLUSIONES

Escribe tus observaciones y conclusiones.
El perceptrón es una red neuronal simple que hace como función clasificar dependiendo de los datos que se haya utilizado para entrenar. Esto genera unos pesos que van a funcionar para cuando se utilicen datos de prueba, que se logre clasificar correctamente dicho dato de prueba. El algoritmo de entrenamiento es similar al de regresión logística, cambia un poco para el perceptrón, pero sigue las bases de la regresión logística. También introduce las redes neuronales, siendo una red neuronal de sólo una neurona, que es la que va a hacer toda la función de la clasificación.