Reporte: A01 El Perceptrón

Aldo Luna

February 6, 2026

1 Perceptron

```
# -*- coding: utf-8 -*-
"""L01_03_Perceptron.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/github/Dr-Carlos-Villasenor/Clase_Aprendizaje_Profundo/blob/main/L01_03_Perceptron.ipynb

# El perceptrón y las compuertas lógicas
## Dr. Carlos Villaseñor

Paso 1. Corre la siguiente casilla para importar la paquetería necesaria.
"""

import numpy as np
import matplotlib.pyplot as plt

"""Paso 2. Modifica la siguiente clase para que tenga un método de entrenamiento con el algoritmo del perceptrón."""

class Perceptron:

  def __init__(self, n_inputs, learning_rate):
    self.w = - 1 + 2 * np.random.rand(n_inputs)
    self.b = - 1 + 2 * np.random.rand()
    self.eta = learning_rate

  def predict(self, X):
    _, p = X.shape
    y_est = np.zeros(p)
    for i in range(p):
      y_est[i] = np.dot(self.w, X[:,i])+self.b
      if y_est[i] >= 0:
        y_est[i]=1
      else:
        y_est[i]=0
    return y_est

  def fit(self, X, Y, epochs=50):
    _, p = X.shape
    for _ in range(epochs):
      for i in range(p):
        # Escribe las ecuaciones del perceptrón
        y_est = self.predict(X[:,i].reshape(-1,1))
        self.w += self.eta * (Y[i] - y_est.item()) * X[:,i]
        self.b += self.eta * (Y[i] - y_est.item())

"""Paso 3. Instancía la siguiente función para poder dibujar la línea que representa el Perceptrón."""

# Función para dibujar superficie de desición
def draw_2d_percep(model):
  w1, w2, b = model.w[0], model.w[1], model.b
  plt.plot([-2, 2],[(1/w2)*(-w1*(-2)-b),(1/w2)*(-w1*2-b)],'--k')
  plt.show()
```

```
54
55 """Paso 4. Corre el siguiente código para comprobar que la neurona es capaz de aprender
       la compuerta OR"""
56
57 # Instanciar el modelo
58 model = Perceptron(2, 0.1)
59
60 # Datos
61 X = np.array([[0, 0, 1, 1],
62               [0, 1, 0, 1]])
63 Y = np.array( [0, 1, 1, 1])
64
65 # Entrenar
66 model.fit(X,Y)
67
68 # Predicción
69 model.predict(X)
70
71 # Primero dibujemos los puntos
72 _, p = X.shape
73 for i in range(p):
74   if Y[i] == 0:
75     plt.plot(X[0,i],X[1,i], 'or')
76   else:
77     plt.plot(X[0,i],X[1,i], 'ob')
78
79 plt.title('Perceptrón')
80 plt.grid('on')
81 plt.xlim([-2,2])
82 plt.ylim([-2,2])
83 plt.xlabel(r'x1')
84 plt.ylabel(r'x2')
85
86 draw_2d_percep(model)
87
88 """Paso 5. Realiza el paso anterior con pero con los datos de la compuerta AND y de la
       compuerta XOR"""
89
90 # Instanciar el modelo
91 model = Perceptron(2, 0.1)
92
93 # Datos
94 X = np.array([[0, 0, 1, 1],
95               [0, 1, 0, 1]])
96 Y = np.array( [0, 0, 0, 1])
97
98 # Entrenar
99 model.fit(X,Y)
100
101 # Predicción
102 model.predict(X)
103
104 # Primero dibujemos los puntos
105 _, p = X.shape
106 for i in range(p):
107   if Y[i] == 0:
108     plt.plot(X[0,i],X[1,i], 'or')
109   else:
110     plt.plot(X[0,i],X[1,i], 'ob')
111
112 plt.title('Perceptrón')
113 plt.grid('on')
114 plt.xlim([-2,2])
115 plt.ylim([-2,2])
116 plt.xlabel(r'x1')
117 plt.ylabel(r'x2')
118
119 draw_2d_percep(model)
120
121 # Instanciar el modelo
122 model = Perceptron(2, 0.1)
123
124 # Datos
```

```
X = np.array([[0, 0, 1, 1],
              [0, 1, 0, 1]])
Y = np.array( [0, 1, 1, 0])

# Entrenar
model.fit(X,Y)

# Predicción
model.predict(X)

# Primero dibujemos los puntos
_, p = X.shape
for i in range(p):
  if Y[i] == 0:
    plt.plot(X[0,i],X[1,i], 'or')
  else:
    plt.plot(X[0,i],X[1,i], 'ob')

plt.title('Perceptrón')
plt.grid('on')
plt.xlim([-2,2])
plt.ylim([-2,2])
plt.xlabel(r'x1')
plt.ylabel(r'x2')

draw_2d_percep(model)

"""Paso 6. ¿Que diferencia puedes notar entre el aprendizaje de la compuerta AND y la
    compuerta XOR?: Escribe aquí tu respuesta

El perceptron es incapaz de aprender el patron presentado por la compuerta OR ya que una
     linea recta no puede separar valores 0 de los 1 debido al acomodo diagonal que
    presentan, por lo que el error calculado en la funcion de costo nunca converge a
    cero.

Respuesta:
"""
```

Listing 1: l01_03_perceptron.py

1.1 Resultados Gráficos

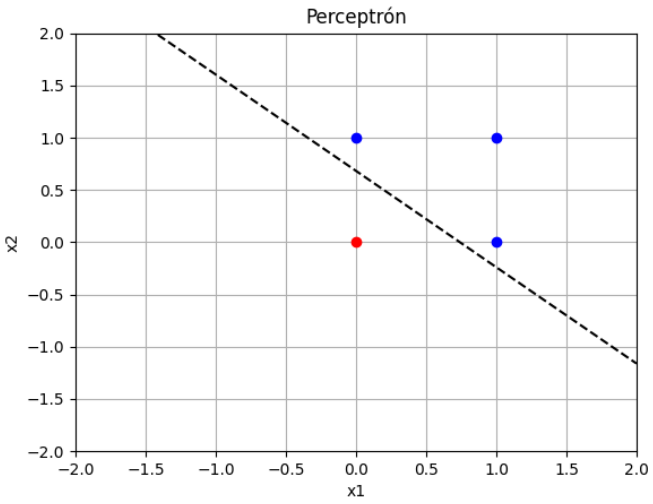

Figure 1: Compuerta OR

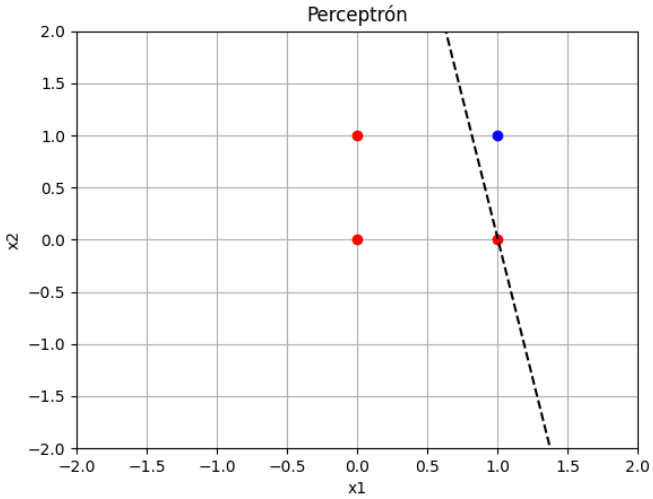

Figure 2: Compuerta AND

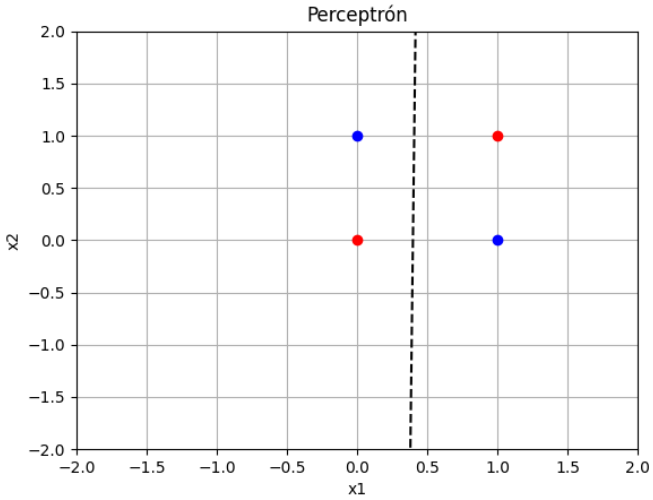

Figure 3: Compuerta XOR

2 Predicción de Sobrepeso

```
# -*- coding: utf-8 -*-
"""L01_04_Predicción_de_sobrepeso.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/github/Dr-Carlos-Villasenor/
    Clase_Aprendizaje_Profundo/blob/main/L01_04_Predicci%C3%B3n_de_sobrepeso.ipynb

# Predicción de sobrepeso con una neurona
## Dr. Carlos Villaseñor

Paso 1. Corre la siguiente casilla para importar la paquetería necesaria.
"""

import numpy as np
import matplotlib.pyplot as plt

"""Paso 2. En la siguiente cassilla de código, pega toda la clase del Perceptron."""
```

```
19
20 class Perceptron:
21
22   def __init__(self, n_inputs, learning_rate):
23     self.w = - 1 + 2 * np.random.rand(n_inputs)
24     self.b = - 1 + 2 * np.random.rand()
25     self.eta = learning_rate
26
27   def predict(self, X):
28     _, p = X.shape
29     y_est = np.zeros(p)
30     for i in range(p):
31       y_est[i] = np.dot(self.w, X[:,i])+self.b
32       if y_est[i] >= 0:
33         y_est[i]=1
34       else:
35         y_est[i]=0
36     return y_est
37
38   def fit(self, X, Y, epochs=50):
39     _, p = X.shape
40     for _ in range(epochs):
41       for i in range(p):
42         # Escribe las ecuaciones del perceptrón
43         y_est = self.predict(X[:,i].reshape(-1,1))
44         error = (Y[i] - y_est.item())
45         self.w += self.eta * error * X[:,i]
46         self.b += self.eta * error
47         print(y_est, error, self.w, self.b)
48
49 """Paso 3. Instancía la siguiente función para poder dibujar la línea que representa el
     Perceptrón."""
50
51 # Función para dibujar superficie de desición
52 def draw_2d_percep(model, xmin, xmax):
53   w1, w2, b = model.w[0], model.w[1], model.b
54   plt.plot([xmin, xmax],[(1/w2)*(-w1*xmin-b),(1/w2)*(-w1*xmax-b)],'--k')
55
56
57 """Paso 4. Completa el siguiente codigo para generar datos de personas con y sin
     sobrepeso. Considera a personas con un peso mínimo de 40Kg y un peso máximo de 120,
     de igual manera, una altura mínima de 1 metro y una máxima de 2.2 metros."""
58
59 p = 100
60
61 # Crear datos
62 X = np.zeros((2,p))
63 Y = np.zeros(p)
64 for i in range(p):
65   # masa aleatoria
66   X[0,i] = np.random.randint(40, 120)
67
68   # estatura aleatoria
69   X[1,i] = np.random.uniform(1, 2.2)
70
71   imc = X[0,i] / X[1,i]**2
72
73   if imc >= 25:
74     Y[i]=1
75   else:
76     Y[i]=0
77
78 """Paso 5. Entrena el modelo y dibuja los resultados"""
79
80 # Crear y entrenar modelo
81 model = Perceptron(2, 0.1)
82 model.fit(X, Y, epochs=60)
83
84 # Dibujar resultados
85 plt.figure()
86 for i in range(p):
87   if Y[i] == 0:
88     plt.plot(X[0,i],X[1,i], 'or')
```

```
  else:
    plt.plot(X[0,i],X[1,i], 'ob')

plt.title('Predicción de Sobrepeso (Datos Originales)')
plt.grid('on')
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')

draw_2d_percep(model, 40, 120)
plt.savefig('sobrepeso_raw.png')
plt.show()

"""Paso 6. Repite el paso anterior pero ahora normaliza los datos"""

# Normalizat datos
X[0,:] = (X[0,:] - 40)/80
X[1,:] = (X[1,:] - 1)/1.2

# Crear y entrenar modelo
model = Perceptron(2, 0.1)
model.fit(X, Y, epochs=60)

# Dibujar resultados
plt.figure()
for i in range(p):
  if Y[i] == 0:
    plt.plot(X[0,i],X[1,i], 'or')
  else:
    plt.plot(X[0,i],X[1,i], 'ob')

plt.title('Predicción de Sobrepeso (Datos Normalizados)')
plt.grid('on')
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')

draw_2d_percep(model, 0, 1)
plt.savefig('sobrepeso_norm.png')
plt.show()
```

Listing 2: l01_04_predicción_de_sobrepeso.py

2.1 Resultados Gráficos

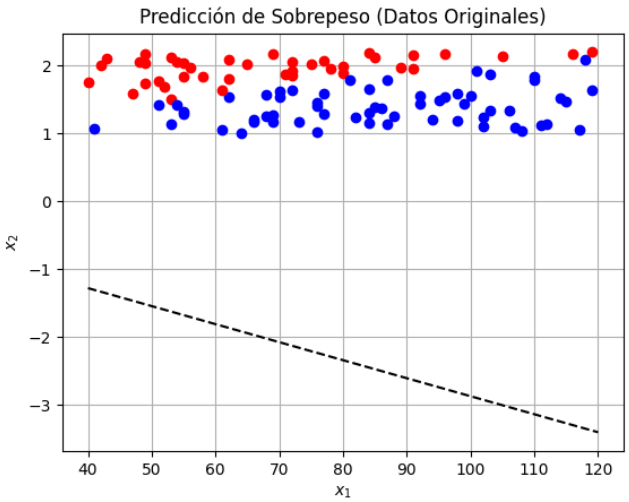

Figure 4: Clasificación con datos originales

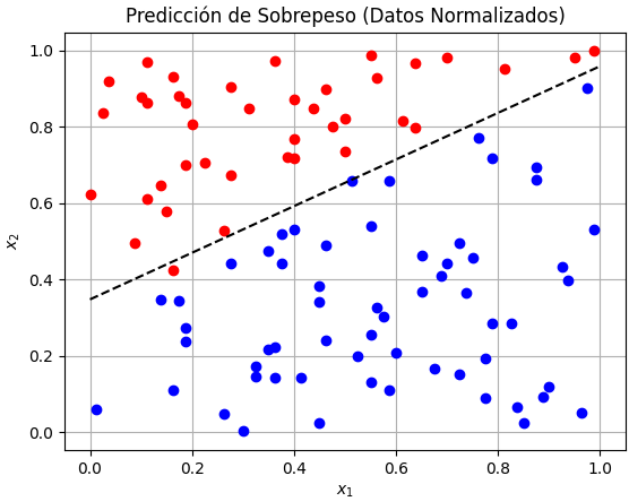

Figure 5: Clasificación con datos normalizados