

Reporte: A01 El Perceptrón

Aldo Luna

February 13, 2026

1 Perceptron

```
1 # -*- coding: utf-8 -*-
2 """L01_03_Perceptron.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/github/Dr-Carlos-Villasenor/
8     Clase_Aprendizaje_Profundo/blob/main/L01_03_Perceptron.ipynb
9
10 # El perceptrón y las compuertas lógicas
11 ## Dr. Carlos Villaseñor
12
13 Paso 1. Corre la siguiente casilla para importar la paquetería necesaria.
14 """
15 import numpy as np
16 import matplotlib.pyplot as plt
17
18 """Paso 2. Modifica la siguiente clase para que tenga un método de entrenamiento con el
19     algoritmo del perceptrón."""
20
21 class Perceptron:
22
23     def __init__(self, n_inputs, learning_rate):
24         self.w = - 1 + 2 * np.random.rand(n_inputs)
25         self.b = - 1 + 2 * np.random.rand()
26         self.eta = learning_rate
27
28     def predict(self, X):
29         _, p = X.shape
30         y_est = np.zeros(p)
31         for i in range(p):
32             y_est[i] = np.dot(self.w, X[:,i])+self.b
33             if y_est[i] >= 0:
34                 y_est[i]=1
35             else:
36                 y_est[i]=0
37         return y_est
38
39     def fit(self, X, Y, epochs=50):
40         _, p = X.shape
41         for _ in range(epochs):
42             for i in range(p):
43                 # Escribe las ecuaciones del perceptrón
44                 y_est = self.predict(X[:,i].reshape(-1,1))
45                 self.w += self.eta * (Y[i] - y_est.item()) * X[:,i]
46                 self.b += self.eta * (Y[i] - y_est.item())
47
48 """Paso 3. Instancia la siguiente función para poder dibujar la línea que representa el
49     Perceptrón."""
50
51 # Función para dibujar superficie de desición
52 def draw_2d_percep(model):
53     w1, w2, b = model.w[0], model.w[1], model.b
54     plt.plot([-2, 2], [(1/w2)*(-w1*(-2)-b), (1/w2)*(-w1*2-b)], '--k')
55     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

```

```

125 X = np.array([[0, 0, 1, 1],
126               [0, 1, 0, 1]])
127 Y = np.array([0, 1, 1, 0])
128
129 # Entrenar
130 model.fit(X,Y)
131
132 # Predicción
133 model.predict(X)
134
135 # Primero dibujemos los puntos
136 _, p = X.shape
137 for i in range(p):
138     if Y[i] == 0:
139         plt.plot(X[0,i],X[1,i], 'or')
140     else:
141         plt.plot(X[0,i],X[1,i], 'ob')
142
143 plt.title('Perceptrón')
144 plt.grid('on')
145 plt.xlim([-2,2])
146 plt.ylim([-2,2])
147 plt.xlabel(r' $x_1$ ')
148 plt.ylabel(r' $x_2$ ')
149
150 draw_2d_percep(model)
151
152 """Paso 6. ¿Que diferencia puedes notar entre el aprendizaje de la compuerta AND y la
153     compuerta XOR?: Escribe aquí tu respuesta
154
155 El perceptron es incapaz de aprender el patron presentado por la compuerta OR ya que una
156     linea recta no puede separar valores 0 de los 1 debido al acomodo diagonal que
157     presentan, por lo que el error calculado en la funcion de costo nunca converge a
158     cero.
159
160 Respuesta:
161 """

```

Listing 1: l01-03.perceptron.py

1.1 Resultados Gráficos

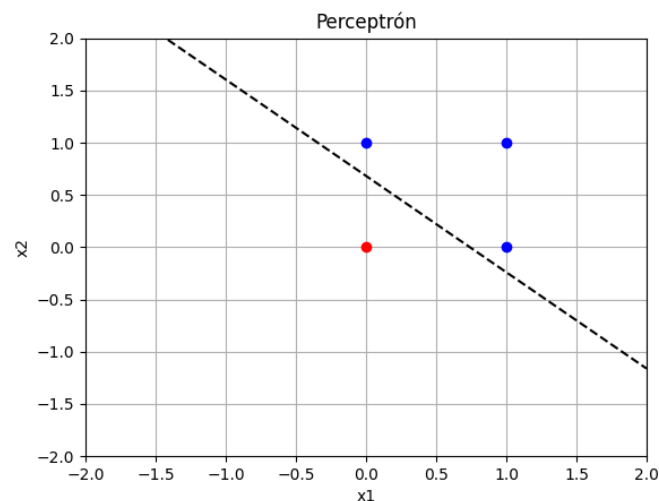


Figure 1: Compuerta OR

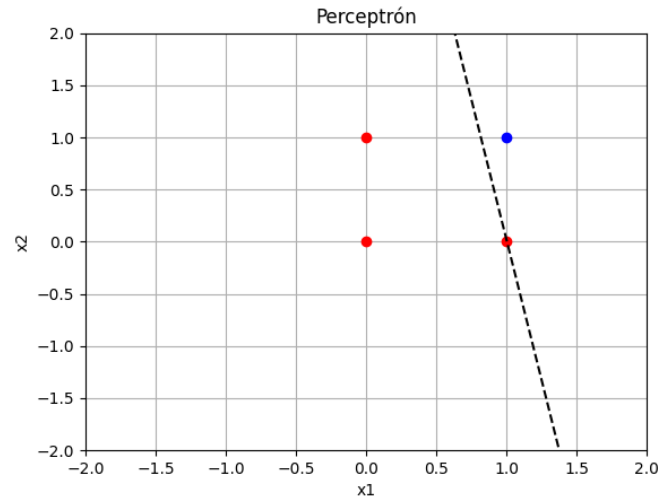


Figure 2: Compuerta AND

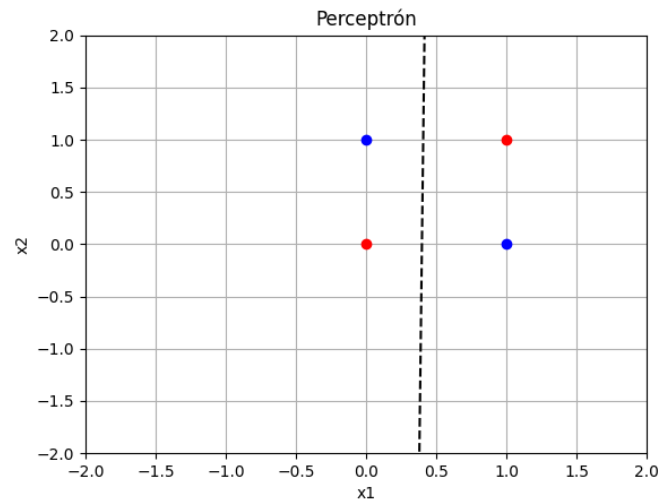


Figure 3: Compuerta XOR

2 Predicción de Sobre peso

```

1 # -*- coding: utf-8 -*-
2 """L01_04_Predicción_de_sobrepeso.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/github/Dr-Carlos-Villasenor/
8     Clase_Aprendizaje_Profundo/blob/main/L01_04_Predicci%C3%B3n_de_sobrepeso.ipynb
9
10 # Predicción de sobre peso con una neurona
11 ## Dr. Carlos Villaseñor
12
13 Paso 1. Corre la siguiente casilla para importar la paquetería necesaria.
14 """
15 import numpy as np
16 import matplotlib.pyplot as plt
17
18 """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. Instancia la siguiente función para poder dibujar la línea que representa el
50 Perceptrón."""
51
52 # Función para dibujar superficie de desición
53 def draw_2d_percep(model, xmin, xmax):
54     w1, w2, b = model.w[0], model.w[1], model.b
55     plt.plot([xmin, xmax],[(1/w2)*(-w1*xmin-b),(1/w2)*(-w1*xmax-b)], '--k')
56
57 """Paso 4. Completa el siguiente código para generar datos de personas con y sin
58 sobrepeso. Considera a personas con un peso mínimo de 40Kg y un peso máximo de 120,
59 de igual manera, una altura mínima de 1 metro y una máxima de 2.2 metros."""
60
61 p = 100
62
63 # Crear datos
64 X = np.zeros((2,p))
65 Y = np.zeros(p)
66 for i in range(p):
67     # masa aleatoria
68     X[0,i] = np.random.randint(40, 120)
69
70     # estatura aleatoria
71     X[1,i] = np.random.uniform(1, 2.2)
72
73     imc = X[0,i] / X[1,i]**2
74
75     if imc >= 25:
76         Y[i]=1
77     else:
78         Y[i]=0
79
80 """Paso 5. Entrena el modelo y dibuja los resultados"""
81
82 # Crear y entrenar modelo
83 model = Perceptron(2, 0.1)
84 model.fit(X, Y, epochs=60)
85
86 # Dibujar resultados
87 plt.figure()
88 for i in range(p):
89     if Y[i] == 0:
90         plt.plot(X[0,i],X[1,i], 'or')

```

```

89     else:
90         plt.plot(X[0,i],X[1,i], 'ob')
91
92 plt.title('Predicción de Sobrepeso (Datos Originales)')
93 plt.grid('on')
94 plt.xlabel(r'$x_1$')
95 plt.ylabel(r'$x_2$')
96
97 draw_2d_percep(model, 40, 120)
98 plt.savefig('sobrepeso_raw.png')
99 plt.show()
100
101 """Paso 6. Repite el paso anterior pero ahora normaliza los datos"""
102
103 # Normalizat datos
104 X[0,:] = (X[0,:] - 40)/80
105 X[1,:] = (X[1,:] - 1)/1.2
106
107 # Crear y entrenar modelo
108 model = Perceptron(2, 0.1)
109 model.fit(X, Y, epochs=60)
110
111 # Dibujar resultados
112 plt.figure()
113 for i in range(p):
114     if Y[i] == 0:
115         plt.plot(X[0,i],X[1,i], 'or')
116     else:
117         plt.plot(X[0,i],X[1,i], 'ob')
118
119 plt.title('Predicción de Sobrepeso (Datos Normalizados)')
120 plt.grid('on')
121 plt.xlabel(r'$x_1$')
122 plt.ylabel(r'$x_2$')
123
124 draw_2d_percep(model, 0, 1)
125 plt.savefig('sobrepeso_norm.png')
126 plt.show()

```

Listing 2: l01_04_predicción_de_sobrepeso.py

2.1 Resultados Gráficos

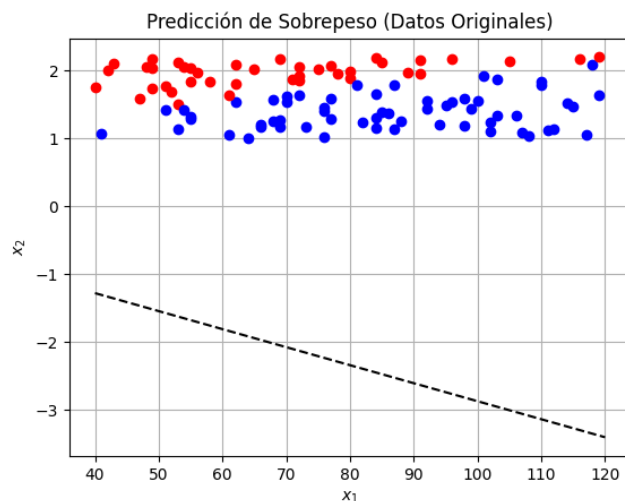


Figure 4: Clasificación con datos originales

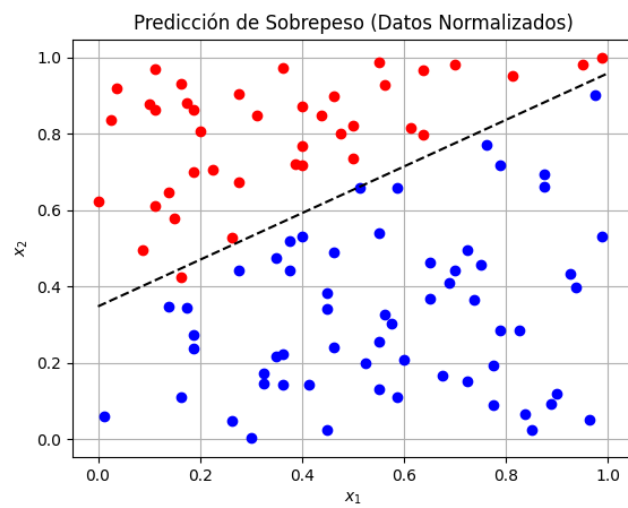


Figure 5: Clasificación con datos normalizados