Reporte: A02 Neurona Lineal y logística

Aldo Luna

February 12, 2026

l02_01_neurona_lineal_(SGD).ipynb

Neurona Lineal

Dr. Carlos Villaseñor

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

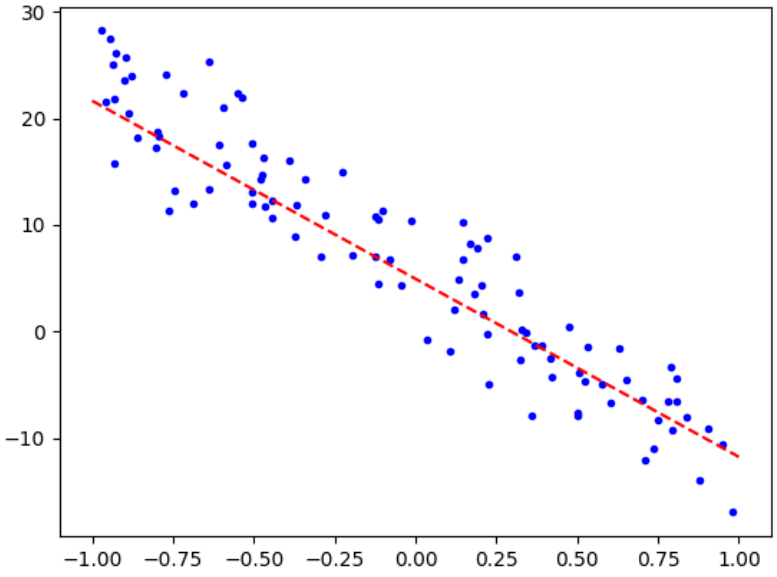
```
import numpy as np
import matplotlib.pyplot as plt
```

Paso 2. Revisa el siguiente código de la neurona lineal y completa las líneas que faltan

```
class Linear_Neuron:

  def __init__(self, n_inputs, learning_rate=0.1):
    self.w = - 1 + 2 * np.random.rand(n_inputs)
    self.b = - 1 + 2 * np.random.rand()
    self.eta = learning_rate

  def predict(self, X):
    Y_est = np.dot(self.w, X) + self.b
    return Y_est

  def train(self, X, Y, epochs=50):
    _, p = X.shape
    for _ in range(epochs):
        for i in range(p):
            y_est = self.predict(X[:,i])
            # Completa las siguientes dos lineas
            self.w += self.eta * (Y[:,i] - y_est) * X[:,i]
            self.b += self.eta * (Y[:,i] - y_est)
```

Paso 3. Corre el siguiente ejemplo

```
# Ejemplo
p =100
x = -1 + 2 * np.random.rand(p).reshape(1,-1)
y = -18 * x + 6 + 3 * np.random.randn(p)
plt.plot(x,y,'.b')


neuron = Linear_Neuron(1, 0.1)
neuron.train(x,y, epochs=100 )

# Dibujar línea
xn = np.array([[-1, 1]])
plt.plot(xn.ravel() ,neuron.predict(xn),'--r')
plt.show()
```



l02_02_neurona_lineal_(BGD).ipynb

Neurona Lineal

Dr. Carlos Villaseñor

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

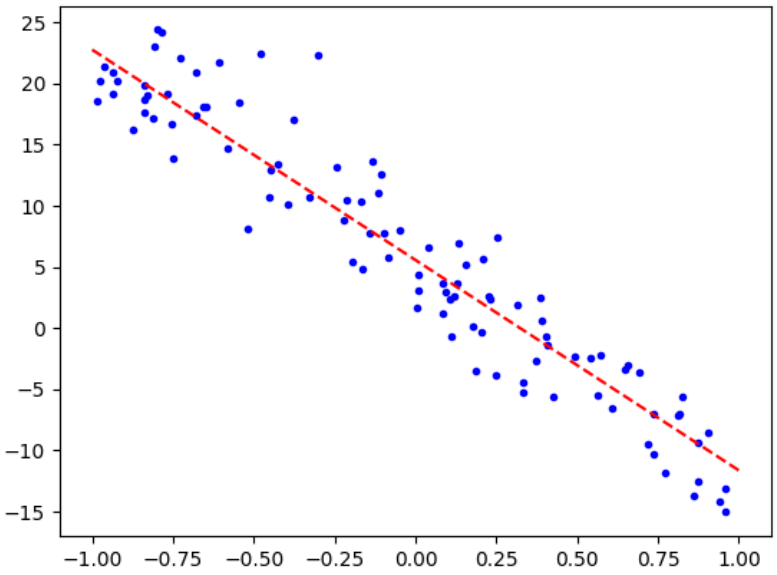
```
import numpy as np
import matplotlib.pyplot as plt
```

Paso 2. Revisa el siguiente código de la neurona lineal y completa las líneas que faltan

```
class Linear_Neuron:

  def __init__(self, n_inputs, learning_rate=0.1):
    self.w = - 1 + 2 * np.random.rand(n_inputs)
    self.b = - 1 + 2 * np.random.rand()
    self.eta = learning_rate

  def predict(self, X):
    Y_est = np.dot(self.w, X) + self.b
    return Y_est

  def train(self, X, Y, epochs=50):
    _, p = X.shape
    for _ in range(epochs):
        Y_est = self.predict(X)
        # Completa las siguientes líneas de código
        self.w += self.eta/p * np.dot((Y-Y_est), X.T).ravel() # (Y-Yest) @
        self.b += self.eta/p * np.sum(Y-Y_est)
```

Paso 3. Corre el siguiente ejemplo

```
# Ejemplo
p =100
x = -1 + 2 * np.random.rand(p).reshape(1,-1)
y = -18 * x + 6 + 3 * np.random.randn(p)
plt.plot(x,y,'.b')


neuron = Linear_Neuron(1, 0.1)
neuron.train(x,y, epochs=100 )

# Dibujar línea
xn = np.array([[-1, 1]])
plt.plot(xn.ravel() ,neuron.predict(xn),'--r')
plt.savefig("output.png")
plt.show()
```



l02_03_neurona_logistica.ipynb

Neurona Logística

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. Revisa el siguiente código de la neurona lineal y completa las líneas que faltan

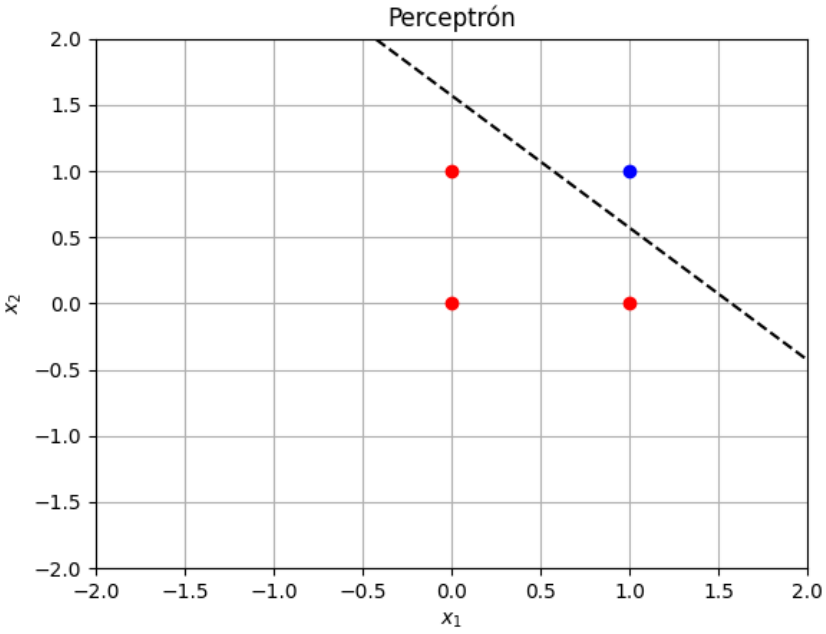
```
class Logistic_Neuron:

    def __init__(self, n_inputs, learning_rate=0.1):
        self.w = - 1 + 2 * np.random.rand(n_inputs)
        self.b = - 1 + 2 * np.random.rand()
        self.eta = learning_rate

    def predict_proba(self, X):
        Z = np.dot(self.w, X) + self.b
        Y_est = 1/(1+np.exp(-Z))
```

```
        return Y_est

    def predict(self, X, umbral=0.5):
        Y_est = self.predict_proba(X)
        return 1 * (Y_est > umbral)

    def train(self, X, Y, epochs=100):
        p = X.shape[1]
        for _ in range(epochs):
            Y_est = self.predict_proba(X)
            diff = Y - Y_est
            self.w += (self.eta/p) * np.dot((diff), X.T).ravel()
            self.b += (self.eta/p) * np.sum(diff)
```

Paso 3. Corre el siguiente ejemplo

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

neuron = Logistic_Neuron(2, 1)
neuron.train(X,Y)
print(neuron.predict_proba(X))
print(neuron.predict(X))

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')

# 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'$x_1$')
plt.ylabel(r'$x_2$')
draw_2d_percep(neuron)
plt.savefig("output.png")
plt.show()
```

```
[0.00805077 0.14813155 0.14824399 0.78853753]
[0 0 0 1]
```



l02_04_Regla_delta.ipynb

Regla Delta

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. Corre las siguientes funciones

```
def linear(z, derivative=False):
    a = z
    if derivative:
        da = np.ones(z.shape)
        return a, da
    return a
```

```
def logistic(z, derivative=False):
    a = 1/(1 + np.exp(-z))
    if derivative:
        da = a * (1 - a)
        return a, da
    return a


def tanh(z, derivative=False):
    a = np.tanh(z)
    if derivative:
        da = (1 - a) * (1 + a)
        return a, da
    return a


def relu(z, derivative=False):
    a = z * (z >= 0)
    if derivative:
        da = np.array(z >= 0, dtype=float)
        return a, da
    return a
```

Paso 3. Revisa el siguiente código de la regla delta

```
class neuron:

    def __init__(self, n_inputs,
                 activation_funtion=linear, learning_rate=0.1):
        self.w = - 1 + 2 * np.random.rand(n_inputs)
        self.b = - 1 + 2 * np.random.rand()
        self.eta = learning_rate
        self.f = activation_funtion

    def predict(self, X):
        Z = np.dot(self.w, X) + self.b
        return self.f(Z)

    def train(self, X, Y, L2=0, epochs=1000):

        p = X.shape[1]
        for _ in range(epochs):

            # Propagation ------------------------------------------------------
            Z = np.dot(self.w, X) + self.b
            Yest, dY = self.f(Z, derivative=True)
```

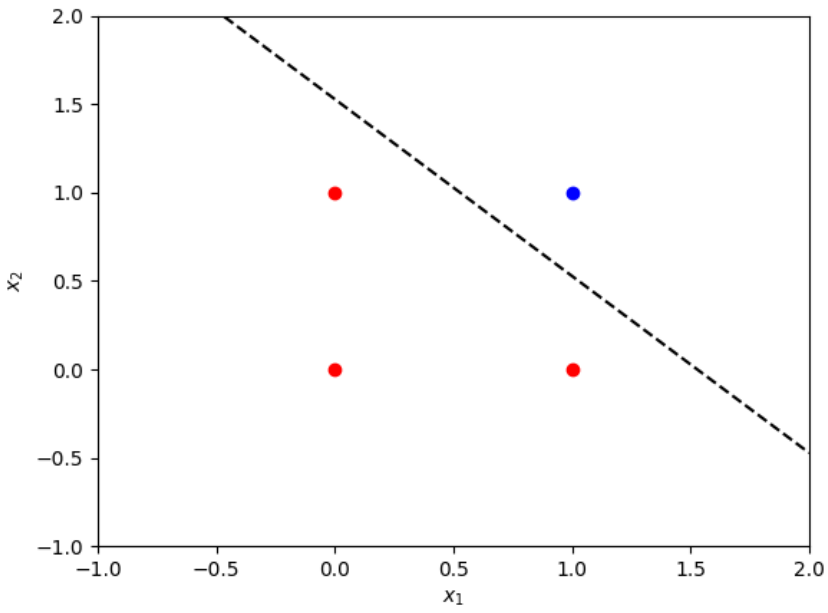
```
            # Training ------------------------------------------------------


            # Calculate local gradient
            lg = (Y - Yest) * dY

            # Update parameters
            self.w +=  (self.eta/p) * np.dot(lg, X.T).ravel()
            self.b += (self.eta/p) * np.sum(lg)
```

Paso 3. Corre este primer ejemplo

```
X = np.array([[0, 0, 1, 1],
              [0, 1, 0, 1]])
Y = np.array([[0, 0, 0, 1]])
model = neuron(2, logistic, 1)
print(model.predict(X))
model.train(X,Y, epochs=1000)
print(model.predict(X))

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

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.xlim([-1,2])
plt.ylim([-1,2])
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
```

```
[0.47411663 0.58514899 0.28278102 0.38151207]
[0.00315057 0.12033617 0.12033252 0.85550921]
```

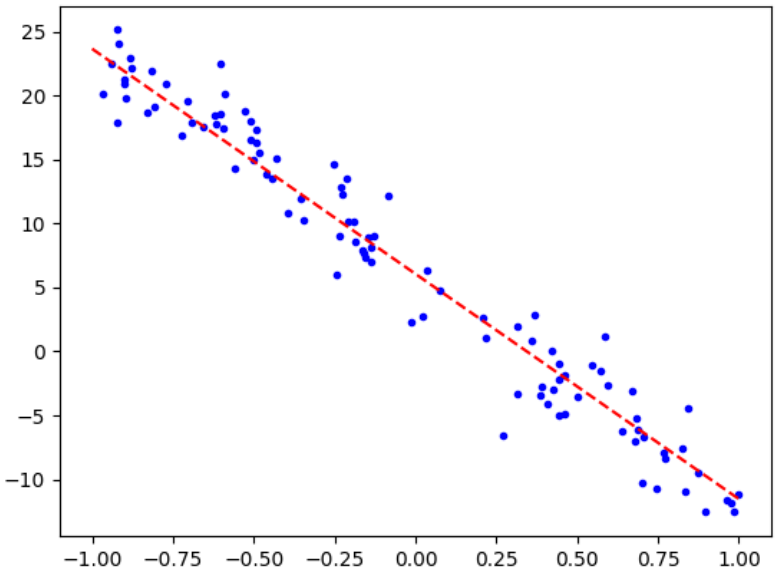
```
Text(0, 0.5, '$x_2$')
```



Paso 4. Corre este segundo ejemplo

```
p = 100
x = -1 + 2 * np.random.rand(p).reshape(1,-1)
y = -18 * x + 6 + 2.5 * np.random.randn(p)
plt.plot(x,y,'.b')
model = neuron(1, linear, 0.1)
model.train(x, y, epochs=100)
xn = np.array([[-1, 1]])
plt.plot(xn.ravel() ,model.predict(xn),'--r')
```

```
[<matplotlib.lines.Line2D at 0x7029c6152890>]
```



neurona_log_dataset.py

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

class Logistic_Neuron:

    def __init__(self, n_inputs, learning_rate=0.1):
        self.w = - 1 + 2 * np.random.rand(n_inputs)
        self.b = - 1 + 2 * np.random.rand()
        self.eta = learning_rate

    def predict_proba(self, X):
        Z = np.dot(self.w, X) + self.b
        Y_est = 1/(1+np.exp(-Z))
        return Y_est

    def predict(self, X, umbral=0.5):
        Y_est = self.predict_proba(X)
        return 1 * (Y_est > umbral)

    def train(self, X, Y, epochs=100):
        p = X.shape[1]
        for _ in range(epochs):
            Y_est = self.predict_proba(X)
            diff = Y - Y_est
            self.w += (self.eta/p) * np.dot((diff), X.T).ravel()
```

```
            self.b += (self.eta/p) * np.sum(diff)

if __name__ == '__main__':

    # Cargar datos
    df = pd.read_csv('cancer.csv')

    # Preprocesamiento
    X = df.drop(['Class'], axis=1).values
    Y = df['Class'].values

    # Transponer
    X = X.T

    neuron = Logistic_Neuron(X.shape[0], 0.1)
    neuron.train(X,Y, epochs=2000)

    def predict_measure(X, Y):
        return np.sum(neuron.predict(X) == Y)

    print("Prediction measure:", predict_measure(X,Y),"of", Y.shape
    [0])
    print("Accuracy:", np.mean(neuron.predict(X) == Y))

    # Prediction measure: 661 of 683
    # Accuracy: 0.9677891654465594
```

Listing 1: neurona_log_dataset.py

Resultados:
Prediction measure: 661 of 683
Accuracy: 0.9677891654465594