En tensorflow no es dataframe sino arrays

train\_x = np.array([[400, 180, 200], [430, 140, 305], [405, 255, 300],

                [180, 180, 180], [220, 100, 160], [405, 255, 300],

                [500, 350, 440], [1500, 900, 200], [1500, 900, 900],

                [1000, 1000, 1000]], dtype=float)

train\_y = np.array([4.20, 4.85, 6, 3.50, 2.70, 6.50, 11, 20.5, 39.8, 35.3], dtype=float)

import numpy as np

#import tensorflow as tf

import matplotlib.pyplot as plt

import tensorflow.compat.v1 as tf

tf.disable\_v2\_behavior()

# Data Random

np.random.seed(101)

from numpy.random import seed

seed(1)

from tensorflow import random

random.set\_seed(101)

#tensorflow.random.set\_seed(101)

#set\_random\_seed(101)

x = np.linspace(0, 50, 50)

y = np.linspace(0, 50, 50)

x += np.random.uniform(-4, 4, 50)

y += np.random.uniform(-4, 4, 50)

n = len(x)

|  |  |
| --- | --- |
| #Visualizar  plt.scatter(x, y)  plt.xlabel('x')  plt.xlabel('y')  plt.title("Training Data")  plt.show() |  |

X = tf.placeholder("float")

Y = tf.placeholder("float")

# 2 variables de tensorFlow

# pesos W, y sesgo b

# inicializadas random

W = tf.Variable(np.random.randn(), name = "W")

b = tf.Variable(np.random.randn(), name = "b")

#hiperparametros del modelo,

learning\_rate = 0.01

training\_epochs = 1000

# Hipotesis, fx de costo, optimizador

y\_pred = tf.add(tf.multiply(X, W), b)

cost = tf.reduce\_sum(tf.pow(y\_pred-Y, 2)) / (2 \* n)

optimizer = tf.train.GradientDescentOptimizer(learning\_rate).minimize(cost)

# No implementaremos el Optimizador de descenso de gradiente manualmente,

#ya que está integrado en Tensorflow. Después de eso,

#Inicializaremos las Variables.

init = tf.global\_variables\_initializer()

with tf.Session() as sess:

    sess.run(init)

    for epoch in range(training\_epochs):

        for (\_x, \_y) in zip(x, y):

            sess.run(optimizer, feed\_dict = {X : \_x, Y : \_y})

        if (epoch + 1) % 50 == 0:

            c = sess.run(cost, feed\_dict = {X : x, Y : y})

            print("Epoch",(epoch + 1), ": cost =", c, "W =", sess.run(W), "b =", sess.run(b))

    training\_cost = sess.run(cost, feed\_dict ={X: x, Y: y})

    weight = sess.run(W)

    bias = sess.run(b)

**Salida:**

Epoch: 50 cost = 5.8868036 W = 0.9951241 b = 1.2381054

Epoch: 100 cost = 5.7912707 W = 0.99812365 b = 1.0914398

Epoch: 150 cost = 5.7119675 W = 1.0008028 b = 0.96044314

Epoch: 200 cost = 5.6459413 W = 1.0031956 b = 0.8434396

Epoch: 250 cost = 5.590799 W = 1.0053328 b = 0.7389357

Epoch: 300 cost = 5.544608 W = 1.007242 b = 0.6455922

Epoch: 350 cost = 5.5057883 W = 1.008947 b = 0.56222

Epoch: 400 cost = 5.473066 W = 1.01047 b = 0.48775345

Epoch: 450 cost = 5.4453845 W = 1.0118302 b = 0.42124167

Epoch: 500 cost = 5.421903 W = 1.0130452 b = 0.36183488

Epoch: 550 cost = 5.4019217 W = 1.0141305 b = 0.30877414

Epoch: 600 cost = 5.3848577 W = 1.0150996 b = 0.26138115

Epoch: 650 cost = 5.370246 W = 1.0159653 b = 0.21905091

Epoch: 700 cost = 5.3576994 W = 1.0167387 b = 0.18124212

Epoch: 750 cost = 5.3468933 W = 1.0174294 b = 0.14747244

Epoch: 800 cost = 5.3375573 W = 1.0180461 b = 0.11730931

Epoch: 850 cost = 5.3294764 W = 1.0185971 b = 0.090368524

Epoch: 900 cost = 5.322459 W = 1.0190892 b = 0.0663058

Epoch: 950 cost = 5.3163586 W = 1.0195289 b = 0.044813324

Epoch: 1000 cost = 5.3110332 W = 1.0199214 b = 0.02561663

Ahora veamos el resultado.

predictions = weight \* x + bias

print("Training cost =", training\_cost, "Weight =", weight, "bias =", bias, '\n')

**Salida:**

Training cost = 5.3110332 Weight = 1.0199214 bias = 0.02561663

Tenga en cuenta que en este caso tanto el peso como el sesgo son escalares.

Esto se debe a que hemos considerado solo una variable dependiente en nuestros datos de entrenamiento.

Si tenemos m variables dependientes en nuestro conjunto de datos de entrenamiento, el Peso será un vector de m dimensiones mientras que el sesgo será un escalar.

Finalmente, trazaremos nuestro resultado.

|  |  |
| --- | --- |
| plt.plot(x, y, 'ro', label ='Original data')  plt.plot(x, predictions, label ='Fitted line')  plt.title('Linear Regression Result')  plt.legend()  plt.show() | Chart, scatter chart  Description automatically generated |