Linear Regression Using TensorFlow

Javier A. Diaz Velazquez

Colorado State University Global

CSC580: Applying Machine Learning and Neural Networks

Dr. Issa

August 6, 2023

Linear Regression Model

The primary purpose of the assignment is to create a linear regression model with TensorFlow in which the algorithm has to comply with the following parameters; 101 fixed seeds for TensorFlow and NumPy, generate random data for training the model and adding some noise to the linear data. In addition, those initial parameters need to be plotted using the "matplotlib. Pyplot" library to visualize the initial data points generated.

Parameters Completed

The rest of the parameters required to complete the algorithm were to complete a TensorFlow model by creating a placeholder that splits the training data into a 2d matrix with X and Y axes which is necessary for feeding the training data into the optimizer during the learning process. After declaring the two variables for the weights and bias initialized randomly, hyperparameters were configured at a learning rate of 0.001 with 1000 training steps (Epochs). Additionally, after defining the hyperparameters, the next step was to define the hypothesis (prediction), the cost function, and the optimizer. The following step consisted of implementing the training process pipeline for the TensorFlow session in which the epochs and the training costs, weight, and bias needed to be printed out in the console. Finally, once these processes were achieved, the final task to be implemented was to plot the fitted line on top of the original data points.

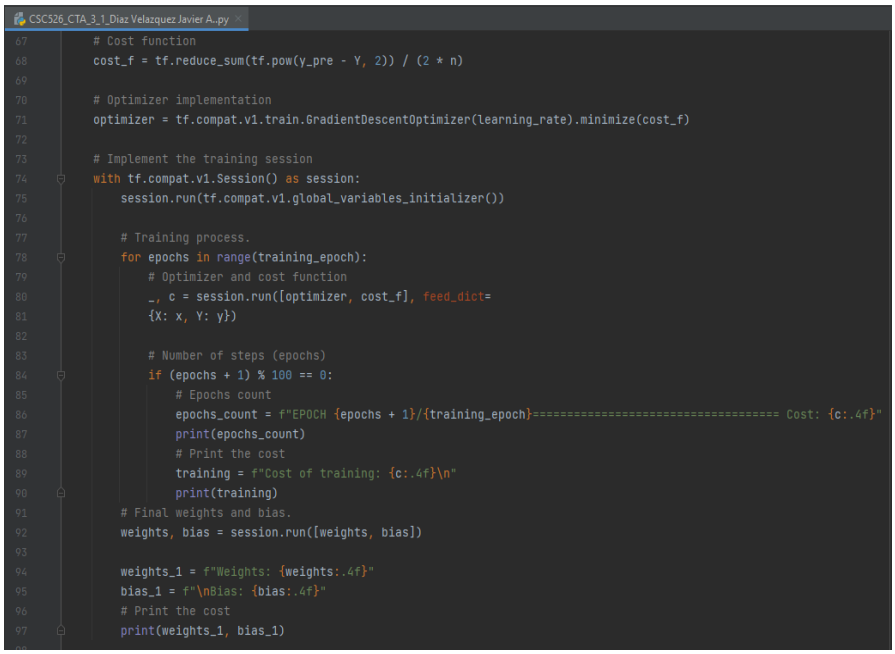
The implementation of the code, the execution, and the plotting graph are demonstrated in the images below.

CSC526_CTA_3_1_Diaz Velazquez Javier A..py

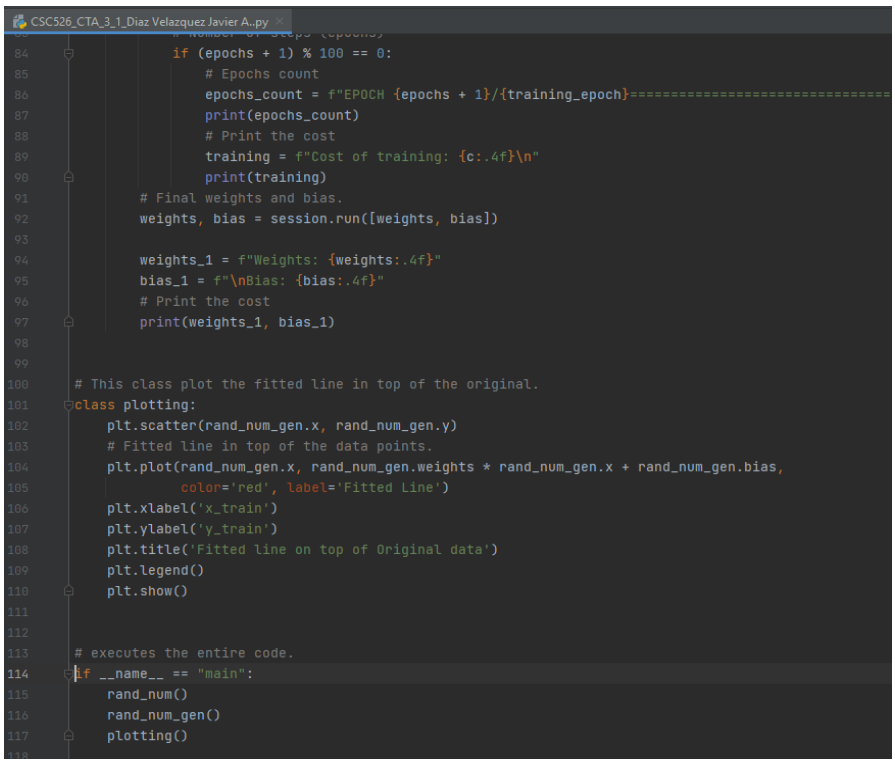
```python
import numpy as npy
import tensorflow as tf
import matplotlib.pyplot as plt
from os import system, name


# This class config houses all the variables for the clr_t() function.
class config:
    sys1 = 'nt'
    sys2 = 'clear'
    sys3 = 'cls'


# This function clears the console terminal.
def clr_T():
    if name == config.sys1:
        _ = system(config.sys3)
    else:
        _ = system(config.sys2)


# This class defines the fixed seed needed for random numbers prediction.
class rand_num:
    npy.random.seed(101)
    tf.compat.v1.set_random_seed(101)


# This class generates random training data for the regression model.
class rand_num_gen:
    clr_T()
    # Generates random linear data.
    x = npy.linspace(0, 50, 50)
    y = npy.linspace(0, 50, 50)
    # Adding noise to the random linear data.
    x += npy.random.uniform(-4, 4, 50)
    y += npy.random.uniform(-4, 4, 50)
```

CSC526_CTA_3_1_Diaz Velazquez Javier A..py

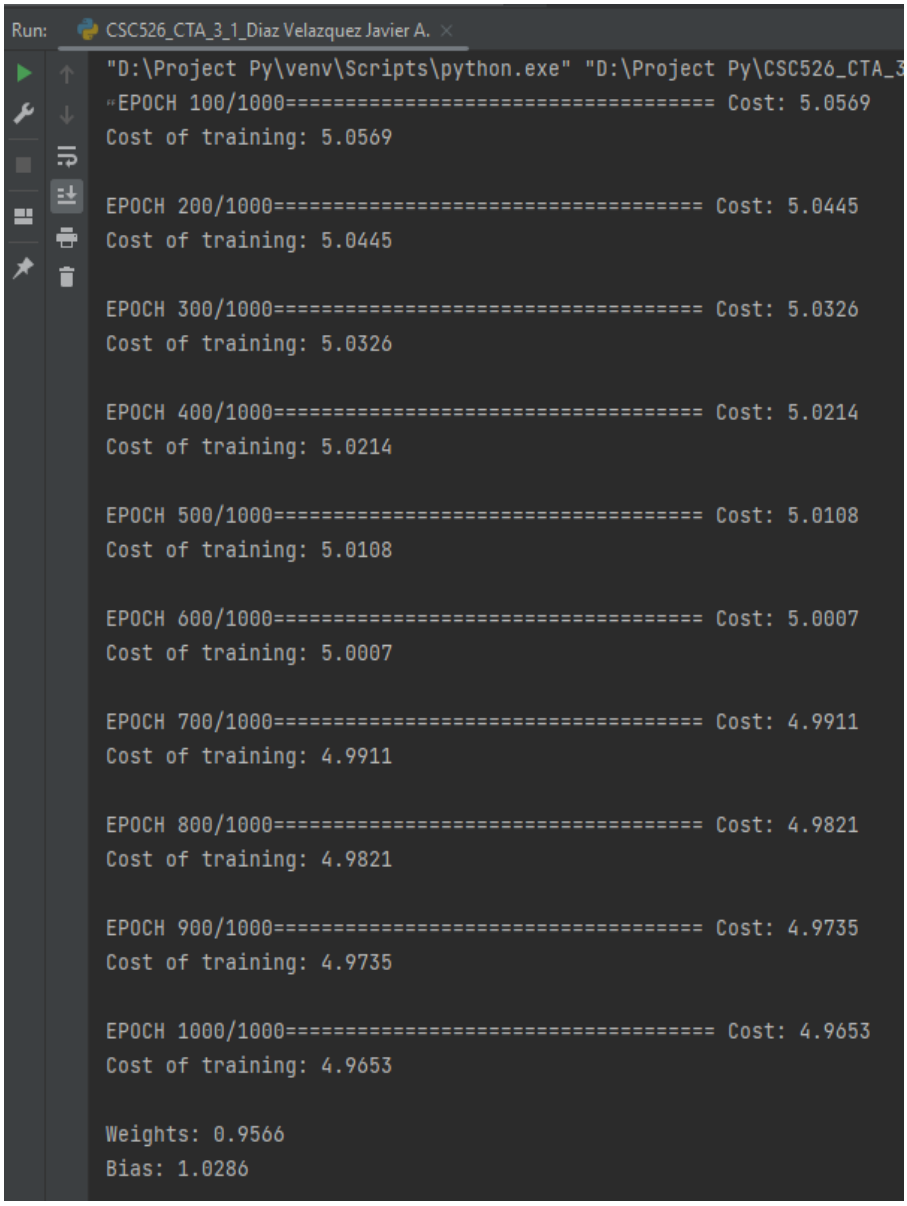
```python

    n = len(x)  # Number of data points.

    # Plot the training data.
    plt.scatter(x, y, label='Training Data', color='red')

    plt.xlabel('Input Data')
    plt.ylabel('Target Data')
    plt.title('Training Data Plot')

    plt.legend()
    plt.show()

    tf.compat.v1.disable_eager_execution()  # This line disables the eager execution which is by default in TF v2.

    # Defines the placeholder.
    X = tf.compat.v1.placeholder(tf.float32)
    Y = tf.compat.v1.placeholder(tf.float32)

    # Declare two trainable TensorFlow variables for the weights and bias.
    weights = tf.Variable(npy.random.randn(), dtype=tf.float32)
    bias = tf.Variable(npy.random.randn(), dtype=tf.float32)

    # Hyperparameters of the model.
    learning_rate = 0.001
    training_epoch = 1000

    # hypothesis
    y_pre = tf.add(tf.multiply(X, weights), bias)

    # Cost function
    cost_f = tf.reduce_sum(tf.pow(y_pre - Y, 2)) / (2 * n)

    # Optimizer implementation
    optimizer = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost_f)

```

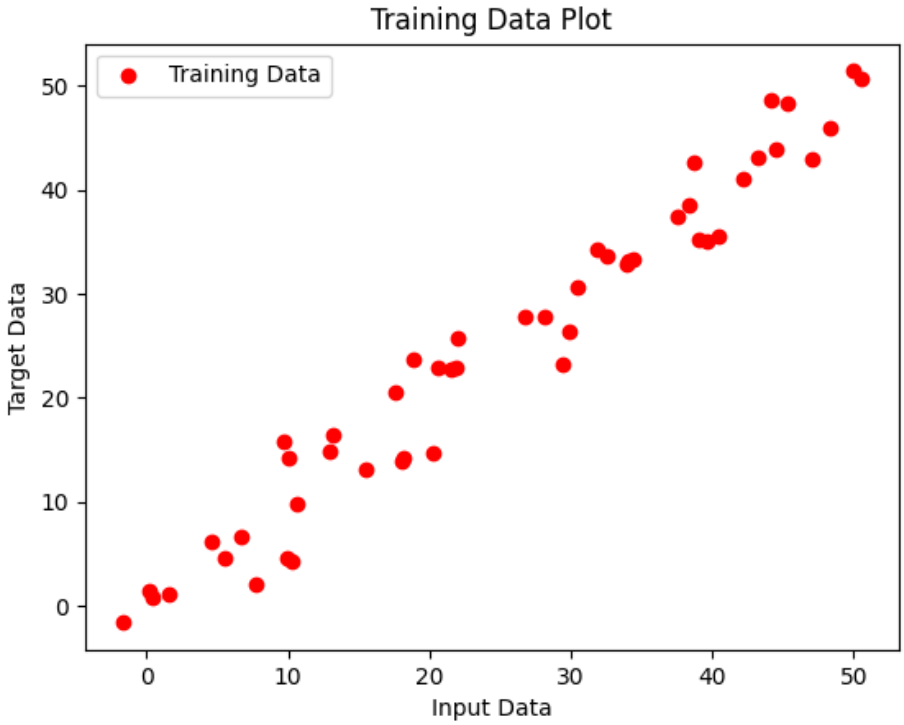


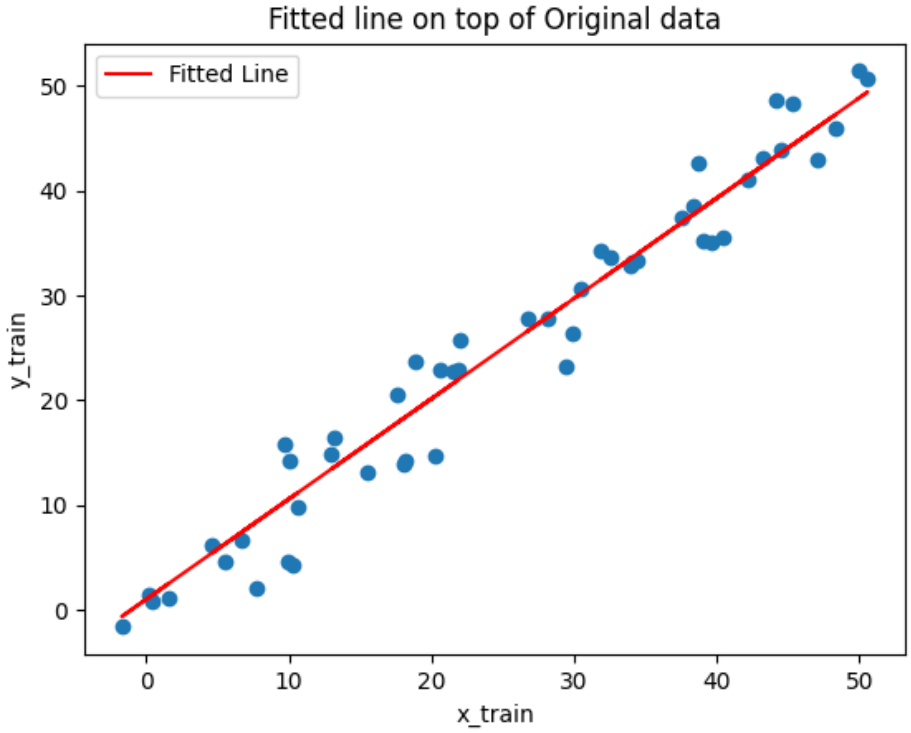
```
CSC526_CTA_3_1_Diaz Velazquez Javier A..py
67      # Cost function
68      cost_f = tf.reduce_sum(tf.pow(y_pre - Y, 2)) / (2 * n)
69
70      # Optimizer implementation
71      optimizer = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost_f)
72
73      # Implement the training session
74      with tf.compat.v1.Session() as session:
75          session.run(tf.compat.v1.global_variables_initializer())
76
77          # Training process.
78          for epochs in range(training_epoch):
79              # Optimizer and cost function
80              _, c = session.run([optimizer, cost_f], feed_dict=
81              {X: x, Y: y})
82
83              # Number of steps (epochs)
84              if (epochs + 1) % 100 == 0:
85                  # Epochs count
86                  epochs_count = f"EPOCH {epochs + 1}/{training_epoch}================================== Cost: {c:.4f}"
87                  print(epochs_count)
88                  # Print the cost
89                  training = f"Cost of training: {c:.4f}\n"
90                  print(training)
91          # Final weights and bias.
92          weights, bias = session.run([weights, bias])
93
94          weights_1 = f"Weights: {weights:.4f}"
95          bias_1 = f"\nBias: {bias:.4f}"
96          # Print the cost
97          print(weights_1, bias_1)
```



```
CSC526_CTA_3_1_Diaz Velazquez Javier A..py
84              if (epochs + 1) % 100 == 0:
85                  # Epochs count
86                  epochs_count = f"EPOCH {epochs + 1}/{training_epoch}================================
87                  print(epochs_count)
88                  # Print the cost
89                  training = f"Cost of training: {c:.4f}\n"
90                  print(training)
91          # Final weights and bias.
92          weights, bias = session.run([weights, bias])
93
94          weights_1 = f"Weights: {weights:.4f}"
95          bias_1 = f"\nBias: {bias:.4f}"
96          # Print the cost
97          print(weights_1, bias_1)
98
99
100 # This class plot the fitted line in top of the original.
101 class plotting:
102     plt.scatter(rand_num_gen.x, rand_num_gen.y)
103     # Fitted line in top of the data points.
104     plt.plot(rand_num_gen.x, rand_num_gen.weights * rand_num_gen.x + rand_num_gen.bias,
105              color='red', label='Fitted Line')
106     plt.xlabel('x_train')
107     plt.ylabel('y_train')
108     plt.title('Fitted line on top of Original data')
109     plt.legend()
110     plt.show()
111
112
113 # executes the entire code.
114 if __name__ == "main":
115     rand_num()
116     rand_num_gen()
117     plotting()
```


Run:
CSC526_CTA_3_1_Diaz Velazquez Javier A.
"D:\Project Py\venv\Scripts\python.exe" "D:\Project Py\CSC526_CTA_3
EPOCH 100/1000========== Cost: 5.0569
Cost of training: 5.0569
EPOCH 200/1000========== Cost: 5.0445
Cost of training: 5.0445
EPOCH 300/1000========== Cost: 5.0326
Cost of training: 5.0326
EPOCH 400/1000========== Cost: 5.0214
Cost of training: 5.0214
EPOCH 500/1000========== Cost: 5.0108
Cost of training: 5.0108
EPOCH 600/1000========== Cost: 5.0007
Cost of training: 5.0007
EPOCH 700/1000========== Cost: 4.9911
Cost of training: 4.9911
EPOCH 800/1000========== Cost: 4.9821
Cost of training: 4.9821
EPOCH 900/1000========== Cost: 4.9735
Cost of training: 4.9735
EPOCH 1000/1000========== Cost: 4.9653
Cost of training: 4.9653
Weights: 0.9566
Bias: 1.0286


Training Data Plot
Training Data
Target Data
Input Data
0
10
20
30
40
50


Fitted line on top of Original data
Fitted Line
y_train
x_train
0
10
20
30
40
50