**Problem#1**

Dataset:

Generate a synthetic dataset using the following Python code.

---------------------------------------------------------------

RANDOM\_SEED = 42

tf.set\_random\_seed(RANDOM\_SEED)

import numpy as np

import matplotlib.pyplot as plt

n\_samples = 30

train\_x = np.linspace(0,20,n\_samples)

train\_y = 3.7 \* train\_x + 14 + 4 \* np.random.randn(n\_samples)

plt.plot(train\_x, train\_y,'o')

---------------------------------------------------------------

The value of RANDOM\_SEED can be any integer. However, choose the value of RANDOM\_SEED as 42. This will allow all students to get identical datasets. The ‘train\_x’ is the predictor variable and ‘train\_y’ is the response variable.

Model Building:

First, compute the regression equation between ‘train\_x’ and ‘train\_y’ variables using Scikit- Learn package. Next use TensorFlow software to compute the regression equation by minimizing the cost function. Use the following ‘cost’ and ‘optimization’ functions.

* • 𝐶𝑜𝑠𝑡 𝐹𝑢𝑛𝑐𝑡𝑖𝑜𝑛 = Σ (𝑦𝑐𝑜𝑚𝑝𝑢𝑡𝑒𝑑 − 𝑦𝑡𝑟𝑎𝑖𝑛 )2 𝑛1
* • 𝑂𝑝𝑡𝑖𝑚𝑖𝑧𝑎𝑡𝑖𝑜𝑛 𝐹𝑢𝑛𝑐𝑡𝑖𝑜𝑛 = 𝐺𝑟𝑎𝑑𝑖𝑒𝑛𝑡 𝐷𝑒𝑠𝑐𝑒𝑛𝑡

Iterate the optimization operation till the cost is minimized. Make sure that the regression equation computed by TensorFlow matches with the regression equation computed by Scikit- Learn. (Note: the value of the slope and intercept will not match exactly – it will be approximately equal)

During the iteration, print the value of ‘cost’, ‘slope’, and ‘intercept’ frequently. Make sure that the value of ‘cost’ variable is decreasing as the number of iterations are increasing. Since the synthetic data is generated using the slope value of 3.7, intercept of 14 and some random noise, your answer for slope and intercept values should be approximately 3.7 and 14.

Hyper Parameters:

There are 2 hyper parameters in this problem. The first one is learning rate (which varies from 0.1 to 0.00001), and other one is epochs (which varies from 10 to 100,000). You must vary both the hyper parameters to make sure that the slope and intercept values converge to a stable value. 2

**Problem#2**

In problem#1 you used TensorFlow software directly to compute the regression equation. In problem#2 you will repeat problem#1. Instead of using TensorFlow directly, use Keras software to interface with TensorFlow. Final regression equation computed in problem#1 would be same as the equation computed in problem#2.

**Problem#3**

**Data Source**

Analyze the data source in ‘kc-house-data.csv’ file. This data source is a part of databases available in the public domain. This file contains 21,613 observations of real-estate properties of King county in Washington state. The data for the following 21 variables is provided.

1. id

2. date

3. price

4. bedrooms

5. bathrooms

6. sqft\_living

7. sqft\_lot

8. floors

9. waterfront

10. view

11. condition

12. grade

13. sqft\_above

14. sqft\_basement

15. yr\_built

16. yr\_renovated

17. zipcode

18. latitude

19. longitude

20. sqft\_living15

21. sqft\_lot15

Response Variable: price

Predictor Variables:

1. bedrooms

2. sqft\_living

Compute the regression equation between the predictor variables and response variables using Scikit-Learn package. Next use TensorFlow software to compute the regression equation by minimizing the cost function. Use the following ‘cost’ and ‘optimization’ functions.

* • 𝐶𝑜𝑠𝑡 𝐹𝑢𝑛𝑐𝑡𝑖𝑜𝑛=Σ(𝑦𝑐𝑜𝑚𝑝𝑢𝑡𝑒𝑑−𝑦𝑡𝑟𝑎𝑖𝑛)2𝑛1
* • 𝑂𝑝𝑡𝑖𝑚𝑖𝑧𝑎𝑡𝑖𝑜𝑛 𝐹𝑢𝑛𝑐𝑡𝑖𝑜𝑛=𝐺𝑟𝑎𝑑𝑖𝑒𝑛𝑡 𝐷𝑒𝑠𝑐𝑒𝑛𝑡

Make sure that the regression equation computed by TensorFlow matches with the regression equation computed by Scikit-Learn. 3

If your Neural Network doesn’t converge, scale all the predictor variables and response variable between 0 – 1. The method (Python code) to scale the variables between 0 and 1 is given in homework#2.

**Problem#4**

In problem#3 you used TensorFlow software directly to compute the regression equation. In problem#4 you will repeat problem#3. Instead of using TensorFlow directly, use Keras software to interface with TensorFlow. Final regression equation computed in problem#3 would be same as the equation computed in problem#4.