Week #4 Assignment – Satish Ramachandran

Problem 1

```
'''
Week4 - Assignment problem #1

'''

import math
import matplotlib.pyplot as plt

x = 5
y = 5

learning_rate = 0.01
epsilon = 0.000001
iteration = 0

'''
Function is:
 f(x,y) = z = -1 * math.sqrt(25 - (x - 2) ** 2 - (y - 3) **2)
'''
def func_z(x, y):
    return (-1 * math.sqrt(25 - (x - 2) ** 2 - (y - 3) **2))

'''
Partial derivative w.r.t x
'''
def dz_dx(x, y):
    return ((2 * (x - 2)) / (math.sqrt((25 - (x - 2) ** 2 - (y - 3) ** 2))))

'''
Partial derivative w.r.t y
'''
def dz_dy(x, y):
    return ((2 * (y - 3)) / (math.sqrt((25 - (x - 2) ** 2 - (y - 3) ** 2))))
```

```
while True:
    iteration = iteration + 1
    plt.plot(x, y, 'o')
    new_x = x - learning_rate * dz_dx(x,y)
    new_y = y - learning_rate * dz_dy(x,y)
    if (abs(x - new_x) < epsilon) and (abs(y - new_y) < epsilon):
        print("Solution reached..")
        x = new_x
        y = new_y
        plt.plot(x, y, 'o')
        plt.waitforbuttonpress()
        print('New x and y less than epsilon')
        break
    # More improvements could be made
    x = new_x
    y = new_y

print('Solution reached after ' + str(iteration) + ' adjustments')
print('Value of x is: ' + str(x))
print('Value of y is: ' + str(y))
print('Value of z is: ' + str(func_z(x, y)))
```

```
(base) satishramac-a01:Week4 satishramach$ python Problem1.py
Solution reached..
New x and y less than epsilon
Solution reached after 2310 adjustments
Value of x is: 2.000248177318635
Value of y is: 3.0001654515457563
Value of z is: -4.999999991103381
```

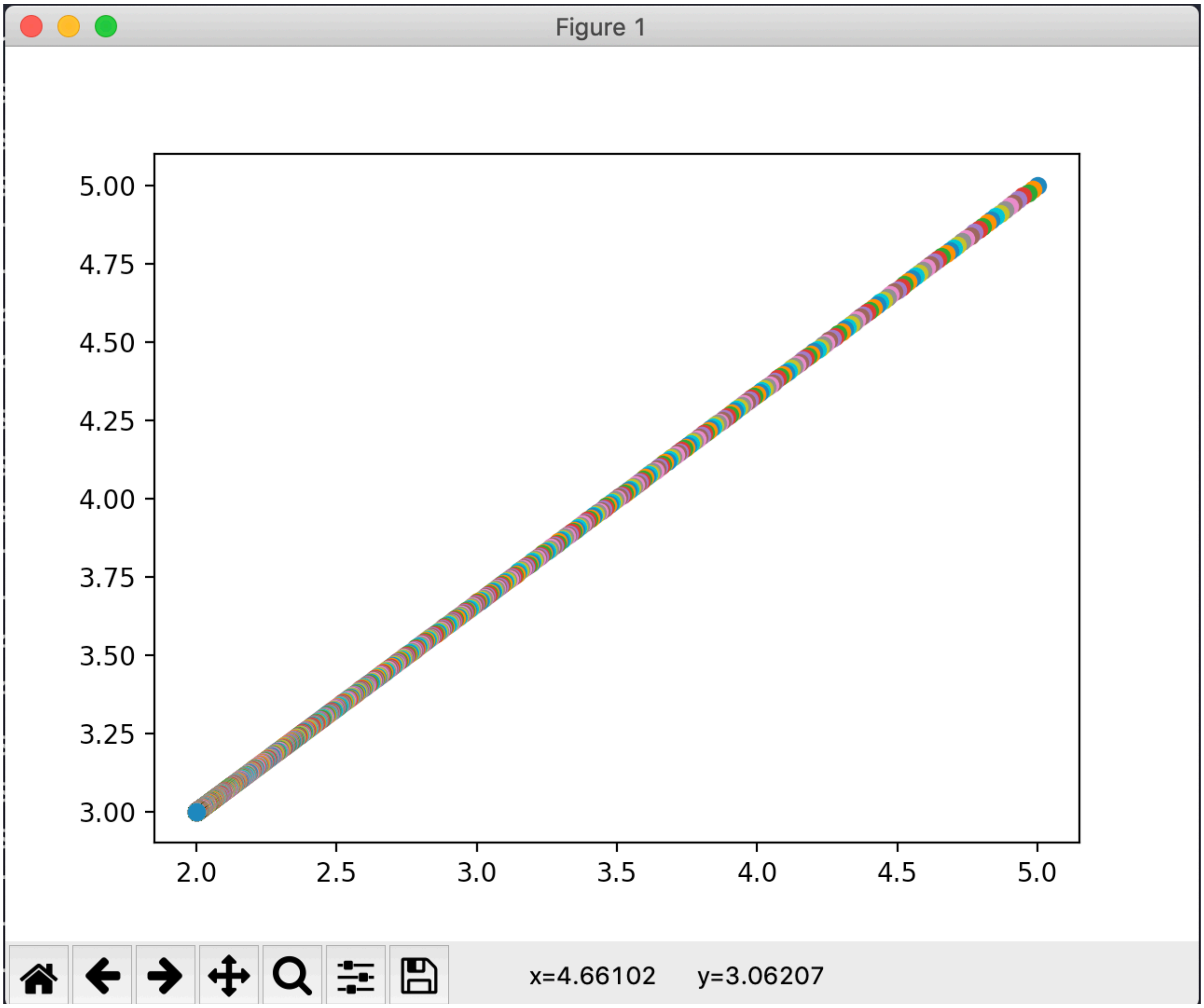
Figure 1
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.0
2.5
3.0
3.5
4.0
4.5
5.0
x=4.66102 y=3.06207

Problem 2

```
'''
Week 4 - Problem 2
Solution using Scikit-learn and hand-coded Gradient descent method

'''
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn import preprocessing

### Generate the data ###
def generate_data(random_seed, n_samples):
    train_x = np.linspace(0,20,n_samples)
    train_y = 3.7 * train_x + 14 + 4 * np.random.randn(n_samples)
    print("X data")
    print("------")
    print("Size: " + str(np.shape(train_x)))
    print(train_x)
    print("Y data")
    print("------")
    print("Size: " + str(np.shape(train_y)))
    print(train_y)
    return(train_x, train_y)

### SciKit Learn method
def scikit_method(x_data, y_data):
    print("Using SciKit learn..")
    linear_reg = linear_model.LinearRegression()
    print("Dimensions: X: " + str(x_data.ndim) + ", Y: " + str(y_data.ndim))
    # IMPORTANT: LinearRegression expects a 2-D array. So, add a dimension using
    # reshape()
    linear_reg.fit(x_data.reshape(-1,1), y_data.reshape(-1,1))
    print("Slope : " + str(linear_reg.coef_))
    print("Intercept: " + str(linear_reg.intercept_))
    return (linear_reg.coef_, linear_reg.intercept_)
```

```
### Handcoded Gradient descent method
'''
Partial differentiation w.r.t slope
'''
def dy_dslope(slope, intercept, x_data, y_data):
   return (-2 * sum((y_data - slope * x_data - intercept) * x_data))

'''
Partial differentiation w.r.t intercept
'''
def dy_dintercept(slope, intercept, x_data, y_data):
   return (-2 * sum((y_data - slope * x_data - intercept)))

'''
Gradient Descent method
'''
def gradient_descent(x_data, y_data, learn_rate, epochs):
   print("Gradient Descent method..")
   slope = 0
   intercept = 0

   for iteration in range(epochs):
      #print(iteration)
      slope = slope - learn_rate * dy_dslope(slope, intercept, x_data, y_data)
      intercept = intercept - learn_rate * dy_dintercept(slope, intercept, x_data,
y_data)

   return (slope, intercept)

'''
Function to descale a min_max scaled data
'''
def deScale_y(y_orig, scaled_y):
   result_y = []
   min_y = min(y_orig)
   max_y = max(y_orig)
   for each_scaled_y in scaled_y:
```

```
        result_y.append(each_scaled_y * (max_y - min_y) + min_y)

    return result_y


train_x_actual, train_y_actual = generate_data(42, 30)
train_x = preprocessing.minmax_scale(train_x_actual)
train_y = preprocessing.minmax_scale(train_y_actual)

s_slope, s_intercept = scikit_method(train_x, train_y)
gd_slope, gd_intercept = gradient_descent(train_x, train_y, 0.001, 4000)

print("RESULTS" + "\n" + "-------")
print("SciKit Learn : slope: " + str(s_slope) + " intercept: " + str(s_intercept))
sci_slope = s_slope[0][0]
sci_intercept = s_intercept[0]
result_scaled_y = train_x * sci_slope + sci_intercept
sci_descaled_computed_y = deScale_y(train_y_actual, result_scaled_y)

print("Gradient Descent : slope: " + str(gd_slope) + " intercept: " +
str(gd_intercept))
result_scaled_y = train_x * gd_slope + gd_intercept
gd_descaled_computed_y = deScale_y(train_y_actual, result_scaled_y)

# Plot the results
plt.plot(train_x_actual, train_y_actual, 'o')
plt.plot(train_x_actual, sci_descaled_computed_y, '-o')
plt.plot(train_x_actual, gd_descaled_computed_y, '-o')
plt.show()
plt.waitforbuttonpress()
```

```
(base) satishramac-a01:Week4 satishramach$ python Problem2.py
X data
------
Size: (30,)
[ 0.          0.68965517  1.37931034  2.06896552  2.75862069  3.44827586
  4.13793103  4.82758621  5.51724138  6.20689655  6.89655172  7.5862069
  8.27586207  8.96551724  9.65517241 10.34482759 11.03448276 11.72413793
 12.4137931  13.10344828 13.79310345 14.48275862 15.17241379 15.86206897
 16.55172414 17.24137931 17.93103448 18.62068966 19.31034483 20.        ]
Y data
------
Size: (30,)
[10.1324881  20.27247899 19.56040352 19.8758327  27.9244679  24.10339172
 29.2153935  33.98646013 31.00014387 30.36080068 39.15754639 46.14557143
 47.74767228 49.16766352 47.77319689 53.89202668 55.98438066 47.71154659
 62.14662761 60.55171094 64.61373962 66.50456287 77.37409401 66.81942334
 74.68099732 82.00284451 76.03000291 87.25920073 82.03237915 81.20239725]
Using SciKit learn..
Dimensions: X: 1, Y: 1
Slope : [[0.9426086]]
Intercept: [0.05218696]
Gradient Descent method..
RESULTS
-------
SciKit Learn : slope: [[0.9426086]] intercept: [0.05218696]
Gradient Descent : slope: 0.9426085769850121 intercept: 0.05218697401882757
```

As can be seen, the SciKit derived values and Gradient Descent values are almost the same.

This is the plot of the deScaled values of Y based on the computed slope and intercept. Since both the values are extremely close, the plot line seems like just one.

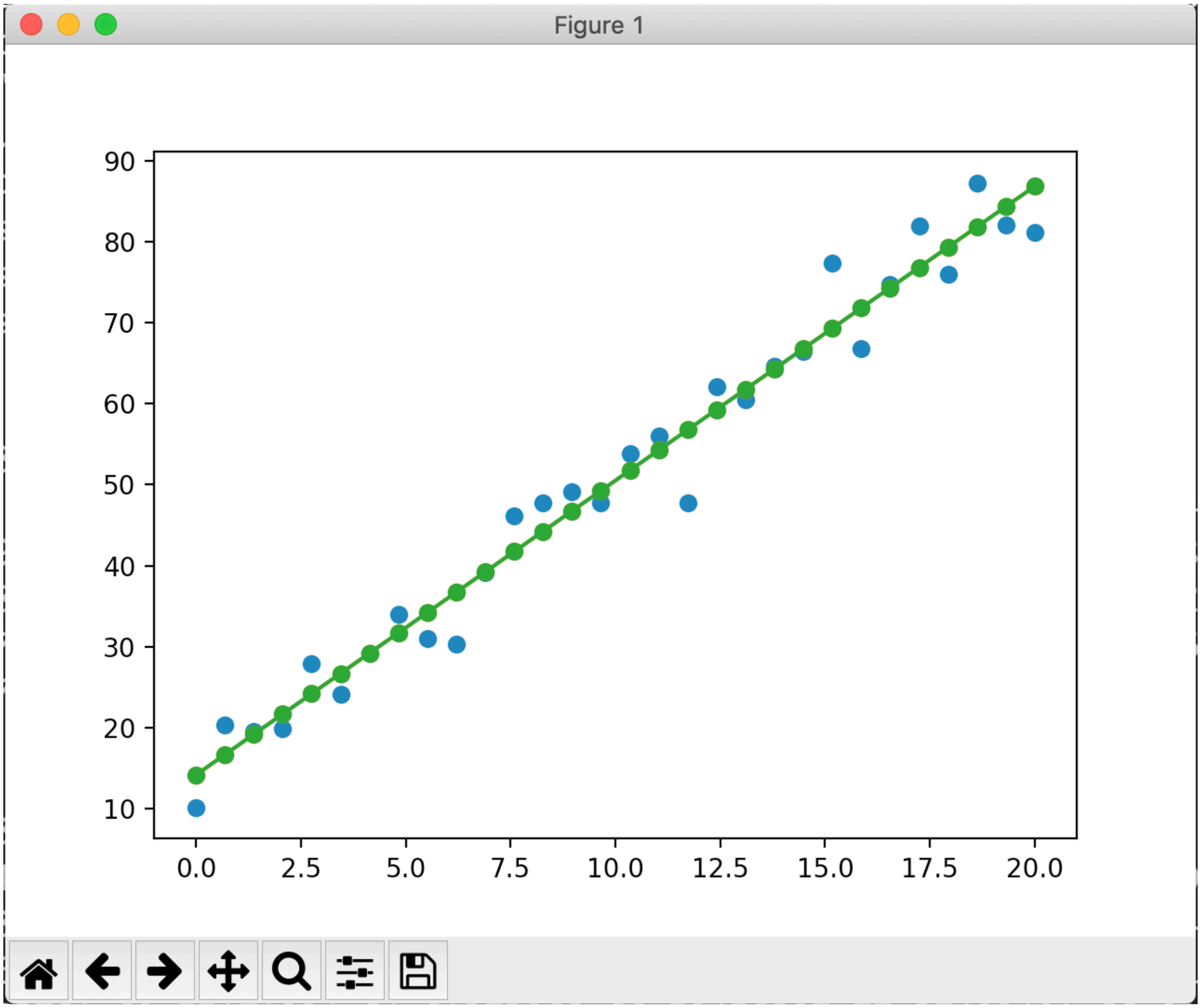
Figure 1
90
80
70
60
50
40
30
20
10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0