**Build a Linear Regression Model for a given dataset using Gradient Descent**

(NumPy and data visualization packages are allowed.)

(Existing Linear Regression Model from any library/package is NOT allowed!)

1. Import numpy and matplotlib.pyplot first in your python code

Installation could be done by pip3 (e.g. “pip3 install numpy”)

e.g.

*import numpy as np*

*import matplotlib.pyplot as plt*

1. **Generate data samples**: generate 10 pair of data samples that satisfying

Y=2\*X + 50 + a small random number

e.g.

*import numpy as np*

*﻿x\_data = np.array([35., 38., 31., 20., 22., 25., 17., 60., 8., 60.])*

*y\_data = 2\*x\_data+50+5\*np.random.random(10) # updated! Thanks~*

X could be within any range, stored as an array (using numpy.array)

1. **Plot the landscape of the loss function. (2pts)**

For example, the following code will print the landscape of Z[i][j]. Z is the loss function value of different *ww* and *bb* values.

X\_data and y\_data are the arrays of your samples.

﻿e.g.

*bb = np.arange(0,100,1) #bias*

*ww = np.arange(-5, 5,0.1) #weight*

*Z = np.zeros((len(bb),len(ww)))*

*for i in range(len(bb)):*

*for j in range(len(ww)):*

*b = bb[i]*

*w = ww[j]*

*Z[j][i] = 0*

*for n in range(len(x\_data)):*

*Z[j][i] = …*

*Z[j][i] = …*

1. Build a linear regression model that minimizing the loss for the given dataset using “gradient descent” algorithm introduced in lecture. **(4pts)**

-

Randomly pick some weights to start the “gradient descent” process.

e.g.

*b = 0 # initial b*

*w = 0 # initial w*

**Explain how your gradient descent process was terminated (e.g. by testing convergence or finishing certain number of iterations) and explain all threshold values you used in your report. (1pt)**

1. Test different values of the learning rate “lr” and different number of iterations/convergence threshold values.

**Explain how these values affect your program in your report. (1pts)**

e.g.

*lr = 0.0001 # example learning rate*

*iteration = 10000 # example iteration number*

1. **Track the change of the weight values (w and b) from each iteration and plot all the values out. (2pts)**

e.g.

*# Store parameters for plotting*

*﻿b\_history = [b]*

*w\_history = [w]*

*# model by gradient descent*

*#…*

*b\_history.append(b)*

*w\_history.append(w)*

*#...*

*﻿plt.plot(b\_history, w\_history, 'o-', ms=3, lw=1.5,color='black')*

*Example track change figure:*

**

1. **Bonus (up to 2pts)**

**Compare the prediction result using your model with the given target values (Y values)**

**Or**

**Any other type of model performance testing.**

*--------------------------------------------------------------------------------------------------------------------*

**Submit your report and your code in two different files.**

**Please include the figure of your plot in your report.**

**e.g.**

**Assignment1\_FengJiang.doc/.pdf (this is the report)**

**Assignment1\_FengJiang.py (this is the code. only “.py” files accepted.)**