Exercise: Basic Linear Regression

Overview:

In this exercise, you will implement linear regression and get to see it work on data.

List of files for this assignment:

ex1.py - main entry script to execute the program for this assignment.
ex1modules.py - the script that needs to be filled in by you for this assignment. It includes functions to compute the cost of linear regression, run gradient descent, and compute the normal equations.

What you should do:

ex1modules.py contains functions that are not yet implemented. Your task is to implement the functions by filling in "YOUR CODE HERE" sections. After you have finished filling in your code, activate Miniconda, change directory to where your ex1.py is located, then type in following command and press Enter:

```
python ex1.py
```

This will run the ex1.py Python script. During the execution, you will see output text results in console and graphic results in a separate window. If your implementation is correct, the graphic results will be similar to what is shown on the "Sample Results" section of this instructions sheet.

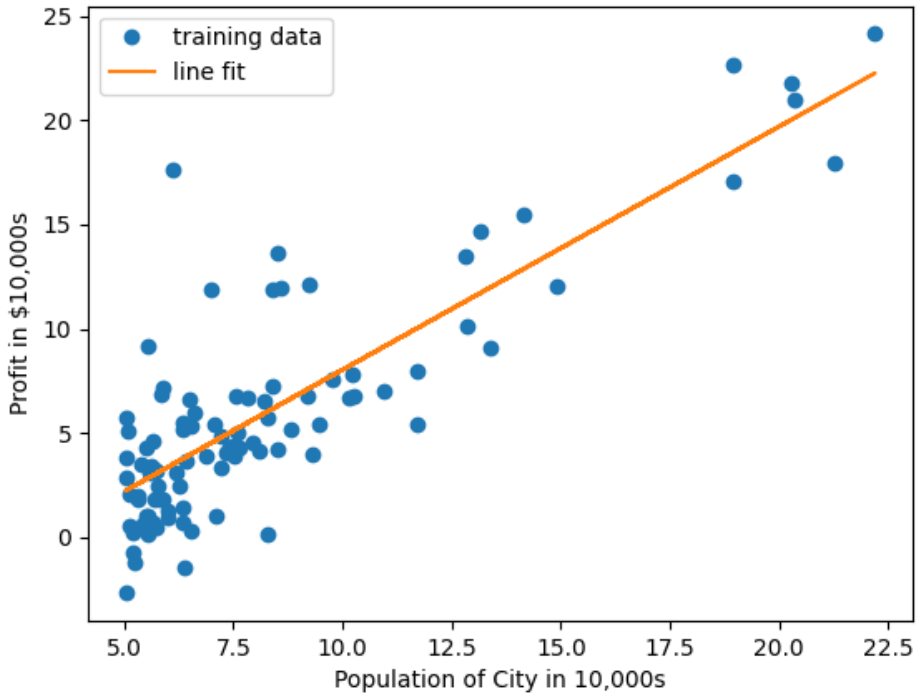
The details can be explained as follows:

[Section 1: Linear regression with one variable]
Suppose you are the CEO of a restaurant franchise (e.g. Paik Jongwon) and are considering different cities for opening a new outlet. The chain already has food trucks in various cities, and you have data for profits and populations from the cities.
The file ex1data1.txt contains the dataset for our linear regression problem. The first column is the population of a city with units in 10,000 persons and the second column is the profit of a food truck in that city with units in 10,000,000 Won (currency unit in Korea). A negative value for profit indicates a loss. We are going to process this data and visualize it using ex1.py Python script.
We start by loading the training data (assigned to variables x and y in ex1.py) as follows:

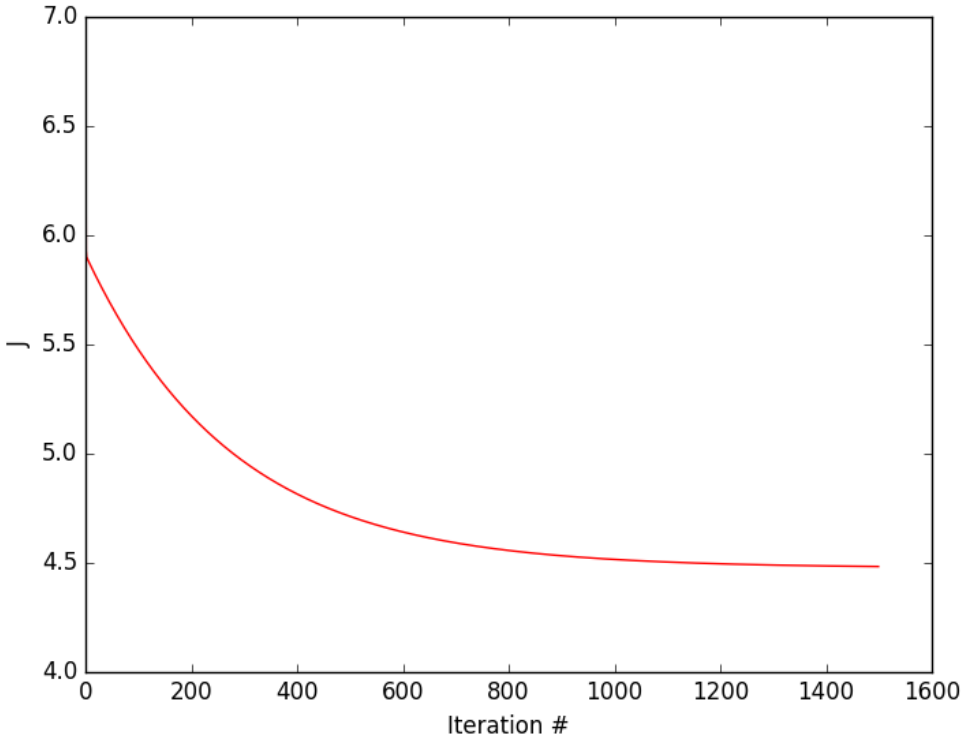
```
x, y = np.loadtxt('ex1data1.txt', delimiter=',', unpack=True)
x = np.matrix(x).T
y = np.matrix(y).T
```

We then compute cost function and perform gradient descent. Here, you should implement computeCost() and gradientDescent() functions in ex1modules.py. Note that for this assignment, gradientDescent() not only should do gradient descent, but also should compute the value of cost function (via invoking computeCost() function) at the end of every gradient descent iteration. This is because, although irrelevant to the optimization process itself, we want to see if the cost function value actually decreases at each iteration.

After implementing computeCost() and gradientDescent() function, the training data and the best fit linear graph of the given training set can be visualized within the rectangular 2D plot space. Your job is to fill in the code in ex1.py to do this. You should give the proper labels to the figure axes. You should also provide the legend that describes what is shown in the plot space. Successful implementation of computeCost() and gradientDescent(), along with correct choice of function(s) for plotting the best fit graph, results in figure that is at least as similar as follows:



Now you should include lines of code that visualizes J as a function of iteration number to ensure that J decreases by iteration. You should also give the proper labels to the figure axes. Correct choice of function(s) for plotting J results in something like this:



Now that you have seen the cost function value properly decreases by iteration, let's plot this as a function of thetas. As stated in class, there are two ways to do this: the 2D-way or 3D-way. To do the 2D-way means to draw a contour plot. To do the 3D-way means to draw a surface plot.

[Section 2: Linear regression with multiple variables]

In this part, you will use ex1data2.txt to do linear regression with multiple variables to predict the prices of houses. ex1data2.txt contains a training set of housing prices in Portland, Oregon.
- The first column is the size of the house (in square feet).
- the second column is the number of bedrooms,
- and the third column is the price of the house.

The initialization part of ex1_multi.py script has been set up to help you step through this exercise. Note that it performs feature scaling and mean normalization (in a single line of code) to make gradient descent converge much more quickly.

Note that this is a multivariate linear regression problem. Make sure your code supports any number of features and is well-vectorized. Looking back at your implemenations of computeCost() and gradientDescent() for Section 1 of this assignment, if the functions already support multiple variables, no further modification is needed. However, if the functions only

support single variable, you should better change that to a more general implementation for linear regression with multiple variables.

Using this normal equation will get an exact solution in one calculation. Complete the normalEqn() function in ex1modules.py to use the equation to calculate parameter vector.