qSTP 2016
WEEK 1 ASSIGNMENT

BASED ON LINEAR REGRESSION & OCTAVE

Regular Assignment

Start Date: Tuesday 24th May 2016
Due Date: Sunday 29th May 2016 midnight

Try to write the codes in VECTORISATION formats. The videos, maybe even after 2.8 might help.

1. This assignment is similar to what is there in coursera but not the same!
2. Within this folder you should find certain files.

Example_1.m and Example_2.m are completed files with very basic scripts (just for brush up). warmUp_1.m and warmUp_2.m are there for your practice (Again you need to write very basic script for this).

3. Open file ex1.m. PLEASE NOTE THAT THIS FILE IS ALREADY COMPLETE AND YOU NEED NOT EDIT IT. After opening you may scroll through the contents. This file is linked to the remaining files i.e. computeCost.m, gradientDescent.m and plotData.m.

4. File ex1data1.txt contains data that is used by ex1.m.

Problem:

1. Introduction :

Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. The chain already has trucks in various cities and you have data for profits and populations from the cities. You would like to use this data to help you select which city to expand to next. In this part of this exercise, you will implement linear regression with one variable to predict profits for a food truck. The file ex1data1.txt contains the dataset for our linear regression problem. The first column is the population of a city and the second column is the profit of a food truck in that city. A negative value for profit indicates a loss. The ex1.m script has already been set up to load this data for you.

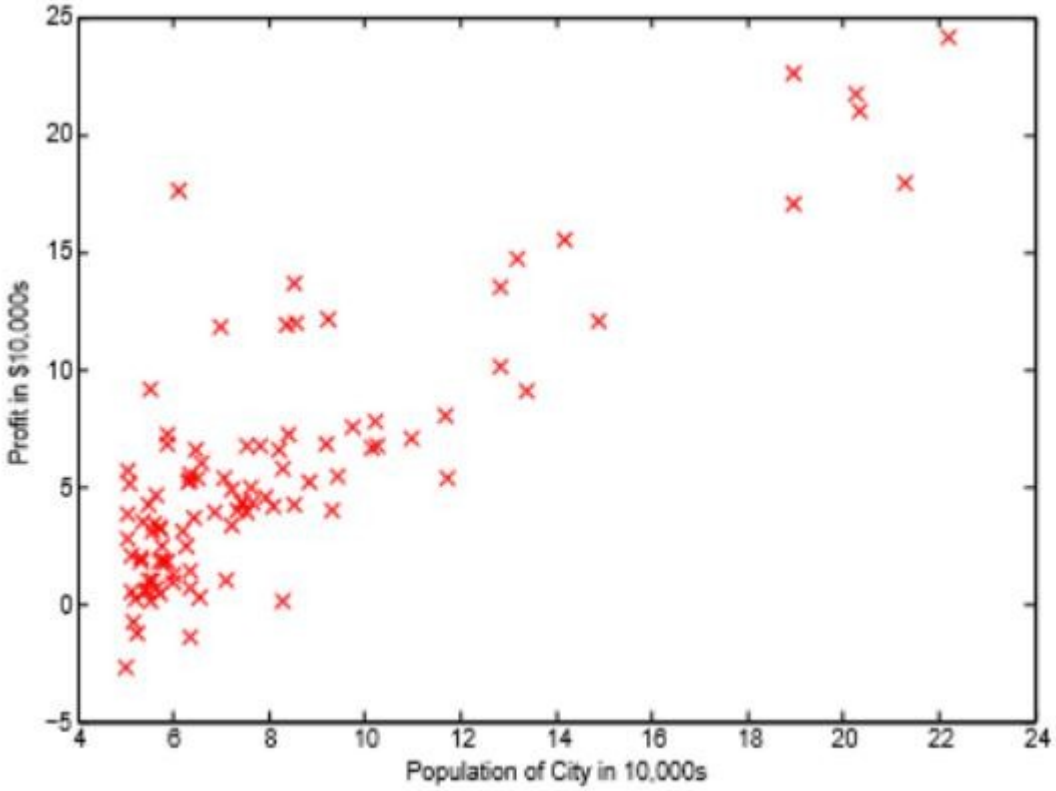
2. Plotting the Data

Before starting on any task, it is often useful to understand the data by visualizing it. For this dataset, you can use a scatter plot to visualize the data, since it has only two properties to plot (profit and population). (Many other problems that you will encounter in real life are multi-dimensional and can't be plotted on a 2-d plot.)

In ex1.m, the dataset is loaded from the data file into the variables X and y:

```
data = csvread('ex1data1.txt'); % read comma separated data
X = data(:, 1);
y = data(:, 2);
m = length(y); % number of training examples
```

Next, the script calls the plotData function to create a scatter plot of the data. Your job is to complete plotData.m to draw the plot. Now, when you continue to run ex1.m, our end result should have red “x” markers and axis labels (shown below). To learn more about the plot command, you can type help plot at the Octave command prompt or to search online for plotting documentation.



(To change the markers to red “x”, we used the option ‘rx’ together with the plot command, i.e., plot(..,[your options here],.., ‘rx’);)

3. Gradient Descent

In this part, you will fit the linear regression parameters θ to our dataset using gradient descent. The objective of linear regression is to minimize the cost function J(θ). Recall that the parameters of your model are the θj values. These are the values you will adjust to minimize cost J(θ). One way to do this is to use the batch gradient descent algorithm. With each step of gradient descent, your parameters θj come closer to the optimal values that will achieve the lowest cost J(θ).

Implementation Note: We store each example as a row in the X matrix in Octave. To take into account the intercept term (θ0), we add an additional first column to X and set it to all ones. This allows us to treat θ0 as simply another ‘feature’.

In ex1.m, we have already set up the data for linear regression. In the following lines, we add another dimension to our data to accommodate the θ0 intercept term. We also initialize the initial parameters to 0 and the learning rate alpha to 0.01.

X = [ones(m, 1), data(:,1)]; % Add a column of ones
to x theta = zeros(2, 1); % initialize fitting
parameters iterations = 1500;
alpha = 0.01;4.

4. Computing the cost J(θ)

As you perform gradient descent to learn minimize the cost function J(θ), it is helpful to monitor the convergence by computing the cost. In this section, you will implement a function to calculate J(θ) so you can check the convergence of your gradient descent implementation. Your next task is to complete the code in the file computeCost.m, which is a function that computes J(θ). As you are doing this, remember that the variables X and y are not scalar values, but matrices whose rows represent the examples from the training set. Once you have completed the function, the next step in ex1.m will run computeCost once using θ initialized to zeros, and you will see the cost printed to the screen.

You should expect to see a cost of 32.07.

5. Gradient descent

Next, you will implement gradient descent in the file gradientDescent.m. The loop structure has been written for you, and you only need to supply the updates to θ within each iteration. As you program, make sure you understand what you are trying to opti- mize and what is being updated.
Keep in mind that the cost J(θ) is parameterized by the vector θ, not X and y. That is, we minimize the value of J(θ) by changing the values of the vector θ, not by changing X or y.

Refer to the equations in this handout and to the video lectures if you are uncertain. A good way to verify that gradient descent is working correctly is to look at the value of J(θ) and check that it is decreasing with each step. The starter code for gradientDescent.m calls computeCost on every iteration and prints the cost. Assuming you have implemented gradient descent and computeCost correctly, your value of J(θ) should never increase, and should converge to a steady value by the end of the algorithm. After you are finished, ex1.m will use your final parameters to plot the linear fit. The result should look something like Figure 2: Your final values for θ will also be used to make predictions on profits in areas of 35,000 and 70,000 people. Note the way that the following lines in ex1.m uses matrix
multiplication, rather than explicit summation or loop- ing, to calculate the predictions. This is an example of code
vectorization in Octave.

```
predict1 = [1, 3.5] * theta;
predict2 = [1, 7] * theta;
```

6. Submission

After you are done with this exercise create a zip format of this folder and mail it to both Akhilesh and me any time before Monday(30th May).

Please make sure you submit on time and all the best for the rest of the course!

----------END----------