**ID5030 Assignment 1**

**Linear Regression, Gradient Descent**

Due 30/01/2018

Estimated Time : 8-10 hours

This assignment involves programming linear regression using gradient descent from ground up and testing it on a given data set. We have broken up the problem into several parts. Please answer all of them and include a short report within a .docx file as with Assignment 0.

**NOTE** : This assignment is a bit longer than what we expect others to be in order to compensate for the missed class on Jan 26.

(15-30 mins) **Initial work** : Look up the packages Numpy, Pandas and Matplotlib within python. In this assignment, these are useful, respectively, for numerical computation (arrays, matrix operations, etc), reading from other datasets, and plotting.

1. Linear Regression with gradient descent : Implement linear regression **using batch gradient descent** in Python. Have a strategy for systematically testing your code
   1. **Simplest case** : Start with one, non-bias feature. That is, a single independent variable. Generate training and test data yourselves by using a pre-determined hypothesis function (can you randomize this?). Ensure that you use both cases with and without noise (**Estimated Time** : 45 mins)
   2. **Multiple features** : Once you are sure the one-feature case works, extend your code to the multiple-feature case. Test this too as above. (**Estimated Time** : 45 mins)
   3. **Reading inputs**: Now ensure that your code can read from a .csv file (use Pandas) and figure out the number of features, examples, etc by itself. That is, the user should be asked only to provide the .csv file, the rest of the job is done by your code (**Estimated Time** : 30 mins)
   4. **Report**: Write a short report for this part including the following : (**Estimated Time** : 30 mins)
      1. Your most general code
      2. Why you are sure your code works. That is, what test cases did you use and why are they general?
      3. What were the convergence criterion and learning rate you used? Did you experiment with the learning rate? Do you have any comments on this?
2. Stochastic gradient descent and variants : As discussed in the class, batch gradient descent is only one of many options available to find the optimum. We will experiment with some variants in this part. For this part of the assignment, use the general, multiple features code you developed in part 1. Read and experiment with the following variants:
   1. Stochastic gradient descent (plain, as discussed in class)
   2. Stochastic gradient descent with momentum
   3. Adagrad
   4. RMSprop

**Reading** : Read about the variations of the stochastic gradient descent here <https://en.wikipedia.org/wiki/Stochastic_gradient_descent>. You need not read the whole document. Just reading about the above variants is sufficient. Note that this page uses mildly different notation from us. You should be able to figure it out. (**Estimated Time** : 30 mins)

**Programming :** Now program all these variants, one by one. It would be ideal to modify your original linear regression code so that the gradient descent function is separate. This way, you can simply choose the module separately. (**Estimated Time :** 1**-**2 hours)

**Report writing** (**Estimated Time** : 1 hour) : For this question, report the following **for *each* variant** used above.

1. Write down the formulations of the variant in a notation which is consistent with what we are using for our class.
2. Include the code for the specific modules you used for this variant. How did you test whether your code gives the correct output?
3. Include a plot for convergence of the loss function with number of iterations for this variant
4. Write briefly what you think the relative merits and demerits of this variant are.

3. Employing your code on real data : We will now use your code above to predict and report on a real case. **This part involves primarily running your code and reporting on the data**. Use any gradient descent algorithm you wish. Download your data from <https://goo.gl/YQTW2m>. The data comprises of 7 independent variables (X) and one output (regression target-Y). In this dataset the "NaN" is used to indicate presence of incomplete data. Incomplete data is quite common in real life data sets. Now finish the following tasks and report on them:

* 1. Without incomplete data : Initially, you can discard the incomplete data points and use the rest to perform linear regression. Read the provided dataset using an in-built function in the python package "Pandas". Remove rows from the data that comprises of incomplete values to obtain your training data set. Train a linear regressor with 70 % of the training data.
     1. **Report** the mean squared error achieved on the remaining 30 % of the data along with the vector of weight coefficients.[¶](about:blank)
     2. **Plot** your estimated values and the ground truth as a function of any two independent variables (**Estimated Time** : 1 hour)
  2. Including incomplete data : In the previous part, we discarded entries from the dataset that had incomplete values. However, this practice may not be the best choice in a situation where data collection is a difficult task.
     1. **Discuss** **briefly**, the different ways in which you can think of estimating the missing values.
     2. Repeat Task (a) with all the data provided after estimating the missing values. Evaluate two different methods of replacing missing data and report your results like in Task 1 for both. (**Estimated Time** : 1-2 hours)