Assignment 3

MET CS 777 - Big Data Analytics
Gradient Descent
(20 points)

GitHub Classroom Invitation Link
`https://classroom.github.com/a/DEne63c6`

1 Description

In this assignment you will implement Batch Gradient Descent to fit a line into a two dimensional data set. You will implement a set of Spark jobs that will learn parameters for such line from the New York City Taxi trip reports in the Year 2013. The dataset was released under the FOIL (The Freedom of Information Law) and made public by Chris Whong (https://chriswhong.com/open-data/foil_nyc_taxi/). See the Assignment 1 for details about this data set.

We would like to train a linear model between travel distance in miles and fare amount (the money that is paid to the taxis).

2 Taxi Data Set - Same data set as Assignment 1

This is the same data set as use for the Assignment 1. Please have a look on the table description there.

The data set is in Comma Separated Volume Format (CSV). When you read a line and split it by comma sign ”,” you will the an string array with length of 17. With index number started from zero, we need for this assignment to get index 5 trip_distance (trip distance in miles) and index 11 fare_amount (fare amount in dollars) as stated on the following table.

index 5 (this our X-axis)	trip_distance	trip distance in miles
index 11 (this our Y-axis)	fare_amount	fare amount in dollars

Table 1: Taxi Data Set fields

Data Clean-up Step

- Remove all taxi rides that are less than 2 min or more than 1 hours.
- Remove all taxi rides that have ”fare_amount” less than 3 dollar or more than 200 dollar
- Remove all taxi rides that have ”trip_distance” less than 1 mile or more than 50 miles
- Remove all taxi rides that have ”tolls_amount” less than 3 dollar.

You can also preprocess the data and store it in your own cluster storage.

3 Obtaining the Dataset

Small data set. (93 MB compressed, uncompressed 384 MB) for implementation and testing purposes (roughly 2 million taxi trips). This is available at Amazon S3:
`https://s3.amazonaws.com/metcs777/taxi-data-sorted-small.csv.bz2`

You can download or access the data sets using the following internal URLs:

	Google Cloud
Small Data Set	gs://metcs777/taxi-data-sorted-small.csv.bz2
Large Data Set	gs://metcs777/taxi-data-sorted-large.csv.bz2

Table 2: Data set on Google Cloud Storage - URLs

	Amzon AWS
Small Data Set	s3://metcs777/taxi-data-sorted-small.csv.bz2
Large Data Set	s3://metcs777/taxi-data-sorted-large.csv.bz2

Table 3: Data set on Amazon AWS - URLs

4 Assignment Tasks

4.1 Task 1 : Simple Linear Regression (4 points)

We want to find a simple line to our data (distance, ”fare_amount”) and use it to predict ”fare_amount” from the travel distance.

Consider a Simple Linear Regression model given in equation (1). What are the regression coefficient for your model?

The solutions for m slope of the line and y-intercept is calculated based on the equations (2) and (3).

$$Y = mX + b \tag{1}$$

$$\hat{m} = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2} \tag{2}$$

$$\hat{b} = \frac{\sum_{i=1}^{n} {x_i}^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n \sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2} \tag{3}$$

Implement a PySpark Job that calculates the exact answers for the parameters m and b. The slope of the line is the parameter m and b is the y-intercept of the line.

Run your implementation on the large data set and report the computation time for your Spark Job for this task. You can find the time for the completion of your Job on the Cloud System. You find there a place that reports the time for you.

Note on Task 1: Execution of this task on Large data set depending on your implementation can take longer time, for example on a cluster with 12 cores in total, it takes more than 40 min computation time.

4.2 Task 2 - Find the Parameters using Gradient Descent (8 Points)

In this task, you should implement the batch gradient descent to find the optimal parameters for our Simple Linear Regression model.

- You should load the data into spark cluster memory as RDD or Dataframe
- Start with all parameters set to 0.1 and do 100 iterations.

Cost function will be then

$$L(m, b) = \sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

Partial Derivatives to update the parameters m and b

$$\frac{\partial}{\partial m} = \frac{2}{N}\sum_{i=1}^{n} -x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial b} = \frac{2}{N}\sum_{i=1}^{n} -(y_i - (mx_i + b)$$

Here is a list of important setup parameters:

- Set your initial learning rate to be learningRate=0.0001 and change it if needed.
- You can implement the bold driver to improve your learning rate.
- Maximum number of iteration should be 100 iterations, num_iteration=100

Run your implementation on the large data set and report the computation time for your Spark Job for this task. Compare the computation time with the previous tasks.

- Print out the costs in each iteration
- Print out the model parameters in each iteration

Note: You might write some code for the iteration of gradient descent in PySpark that can work perfect on your laptop but does not run on the clusters (AWS/Google Cloud). The main reason is that on your laptop it is running in single process while on a cluster it runs on multiple processes (shared-nothing processes). You need to be careful to reduce all of jobs/processes to be able to update the variables, otherwise each processes will have its own variables.

4.3 Task 3 - Fit Multiple Linear Regression using Gradient Descent (8 Points)

We would like to learn a linear model with 4 variables to predict total paid amounts of Taxi rides. The following table describes the variables that we want to use.

index 4 (1st independent variable)	trip_time_in_secs	duration of the trip
index 5 (2nd independent variable)	trip_distance	trip distance in miles
index 11 (3rd independent variable)	fare_amount	fare amount in dollars
index 12 (4th independent variable)	tolls_amount	bridge and tunnel tolls in dollars
index 16 (y-axis, dependent variable)	total_amount	total paid amount in dollars

Table 4: Taxi Data Set fields

- Initialize all parameters with 0.1
- Set your learning rate to be learningRate=0.001
- Maximum number of iteration should be 100, num_iteration=100
- **Use Vectorizatoin for this task. We will not accept your solution when you write duplicated code. It should include vectorization.**

- **Implement ”Bold Driver” technique to dynamically change the learning rate.** (2 points of 8 points)
- Print out the costs in each iteration
- Print out the model parameters in each iteration

5 Important Considerations

5.1 Machines to Use

One thing to be aware of is that you can choose virtually any configuration for your EMR cluster - you can choose different numbers of machines, and different configurations of those machines. And each is going to cost you differently!

Pricing information is available at: `http://aws.amazon.com/elasticmapreduce/pricing/`

Since this is real money, it makes sense to develop your code and run your jobs locally, on your laptop, using the small data set. Once things are working, you'll then move to Amazon EMR.

We are going to ask you to run your Spark jobs over the “real” data using 3 machines with **4 cores and 8GB RAM each** as workers. This provides 4 cores per machine (16 cores total) so it is quite a bit of horsepower. On the Google cloud take 4 machines with 4 cores and 8 GB of memory.

As you can see on EC2 Price list , this costs around 50 cents per hour. That is not much, but **IT WILL ADD UP QUICKLY IF YOU FORGET TO SHUT OFF YOUR MACHINES**. Be very careful, and stop your machine as soon as you are done working. You can always come back and start your machine or create a new one easily when you begin your work again. Another thing to be aware of is that Amazon charges you when you move data around. To avoid such charges, do everything in the ”N. Virginia” region. That's where data is, and that's where you should put your data and machines.

- You should document your code very well and as much as possible.
- You code should be compilable on a unix-based operating system like Linux or MacOS.

5.2 Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between ”cheating” and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

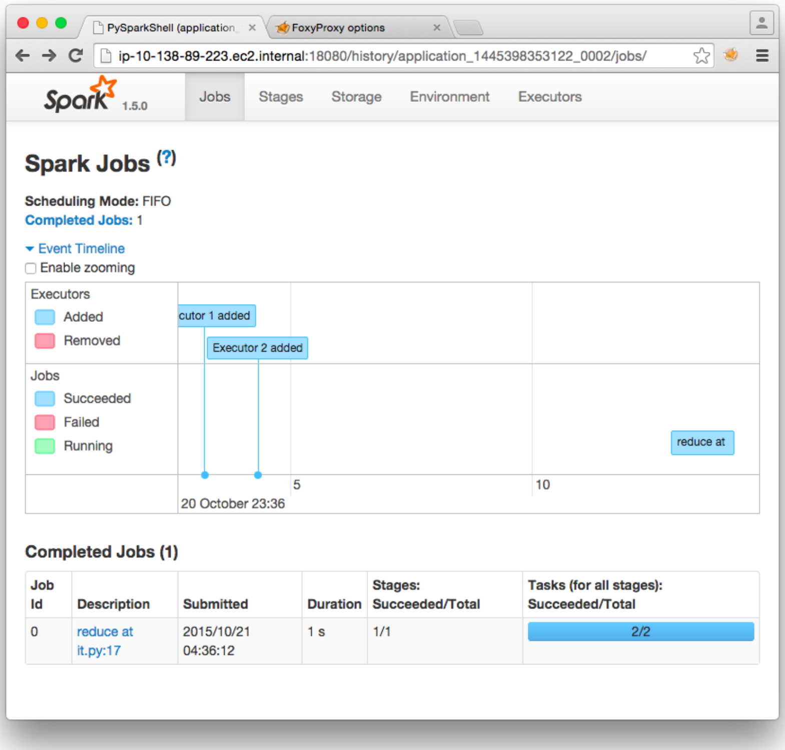
The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way—visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**. As far as going to the web and using Google, we will apply the **”two line rule”**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the ”two line rule” inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.

5.3 Turnin

Create a single document that has results for all three tasks. For each task, copy and paste the result that your last Spark job wrote to Amazon S3. Also for each task, for each Spark job you ran, include a screen shot of the Spark History.



Spark Jobs (?)

Scheduling Mode: FIFO
Completed Jobs: 1

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	reduce at it.py:17	2015/10/21 04:36:12	1 s	1/1	2/2

Figure 1: Screenshot of Spark History

Please zip up all of your code and your document (use .zip only, please!), or else attach each piece of code as well as your document to your submission individually.

Please have the latest version of your code on the GitHub. Zip the files from GitHub and submit as your latest version of assignment work to the Blackboard. We will consider the latest version on the Blackboard but it should exactly match your code on the GitHub