

Assignment 3

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

GitHub Classroom Invitation Link

<https://classroom.github.com/a/W1bmATsM>

1 Description

In this assignment, you will implement Batch Gradient Descent to fit a line into two-dimensional data set. You will implement a set of Spark jobs that will learn parameters for such lines from the New York City Taxi trip reports in 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 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 paid to the taxis).

2 Taxi Data Set - Same data set as Assignment 1

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

The data set is in Comma Separated Volume Format (CSV). When you read a line and split it by a comma sign “,” you will find the string array with a length of 17. With the index number starting 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 in 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

You can use the following PySpark Code to clean up the data.

```
def isfloat(value):
    try:
        float(value)
        return True

    except:
        return False

def correctRows(p):
    if(len(p)==17):
        if(isfloat(p[5]) and isfloat(p[11])):
            if(float(p[5])!=0 and float(p[11])!=0):
                return p

testDataFrame = spark.read.format('csv').\
    options(header='false', inferSchema='true', sep =",").\
    load(testFile)

testRDD = testDataFrame.rdd.map(tuple)
taxilinesCorrected = testRDD.filter(correctRows)
```

In addition to the above filtering, you should remove all of the rides that have a total amount larger than 600 USD and less than 1 USD. You can preprocess the data, clean it and store it in your own cluster storage. To avoid additional computation in each run.

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 Google Storage:

<https://storage.googleapis.com/met-cs-777-data/taxi-data-sorted-small.csv.bz2> and the whole dataset (8GB) <https://storage.googleapis.com/met-cs-777-data/taxi-data-sorted-large.csv.bz2>

When running your code on the cluster, you can access the data sets using the following internal URLs:

| | Google Cloud |
|----------------|---|
| Small Data Set | gs://met-cs-777-data/taxi-data-sorted-small.csv.bz2 |
| Large Data Set | gs://met-cs-777-data/taxi-data-sorted-large.csv.bz2 |

Table 2: Data set on Google Cloud Storage - URL

4 Assignment Tasks

4.1 Task 1: Simple Linear Regression (4 points)

We want to find a simple line to our data (distance, money). Consider a Simple Linear Regression model given in equation (1). The solutions for the m slope of the line and y-intercept are calculated based on equations (2) and (3).

$$Y = mX + b \quad (1)$$

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

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

Implement a PySpark Job that calculates the exact answers for the parameters m and b . The line slope 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 a longer time, for example, on a cluster with 12 cores in total, it takes more than 30 min computation time.

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

In this task, you should implement 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

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 recommended values of the important setup parameters:

- Initialize all parameters with 0
- Start with a very small learning rate and then increase it to speed up the learning. For example, you can start with learningRate=0.000001 and try bigger rates until the process is still converging. **Note:** The optimal learning rate for the big dataset can be different from the optimal learning rate for the small dataset
- Maximum number of iterations should be num_iteration=50 iterations

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.
- Comment on how you can interpret the parameters of the model. What is the meaning of m and b in this case

Note: You might write some code for the iteration of gradient descent in PySpark that can work perfectly 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 a single process, while on a cluster, it runs on multiple processes (shared-nothing processes). You need to be careful to reduce all of the jobs/processes to be able to update the variables, otherwise, each process 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 four variables to predict the 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 3: Taxi Data Set fields

- Initialize all parameters with 0
- Start with a very small learning rate and then increase it to speed up the learning. For example, you can start with learningRate=0.000000001 and try bigger rates until the process is still converging. **Note:** The optimal learning rate for the big dataset can be different from the optimal learning rate for the small dataset
- Maximum number of iterations should be 50, num_iteration=50
- Use NumPy Arrays to calculate the gradients - Vectorization
- Implement the "**Bold Driver**" technique to change the learning rate dynamically. (3 points of 8 points)
- You need to find parameters that will lead to convergence of the optimization algorithm.
- Print out the costs in each iteration
- Print out the model parameters in each iteration
- Comment on how you can interpret the parameters of the model. What is the meaning of m_i and b in this case

5 Important Considerations

5.1 Machines to Use

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

As a proposal for this assignment, you can use the e2-standard-4 machines on the Google Cloud, one for the Master node and two for worker nodes. You will have three machines with a total of 12 vCPU and 48GB RAM. 100 GB of disk space will be enough.

Remember to delete your cluster after the calculation is finished!!!

More information regarding Google Cloud Pricing can be found here <https://cloud.google.com/products/calculator>. As you can see average server 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 Google and Amazon charge you when you move data around. To avoid such charges, do everything in the "Iowa (us-centall)" 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.
- Your 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 ours, there is sometimes a very fine line between "cheating" and acceptable and beneficial interaction between peers. Thus, it is essential that you fully understand what is and what is not allowed in collaboration with your classmates. We want to be 100% precise, so 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

1. Fill in the results in the provided template.
2. For each task, and for each Spark job you ran, include a screenshot of the Spark History.
To demonstrate that you did execute your code on the cloud, it is important to include URLs in the screenshots. Otherwise, there is no way for us to verify if the code was executed in your cloud account.

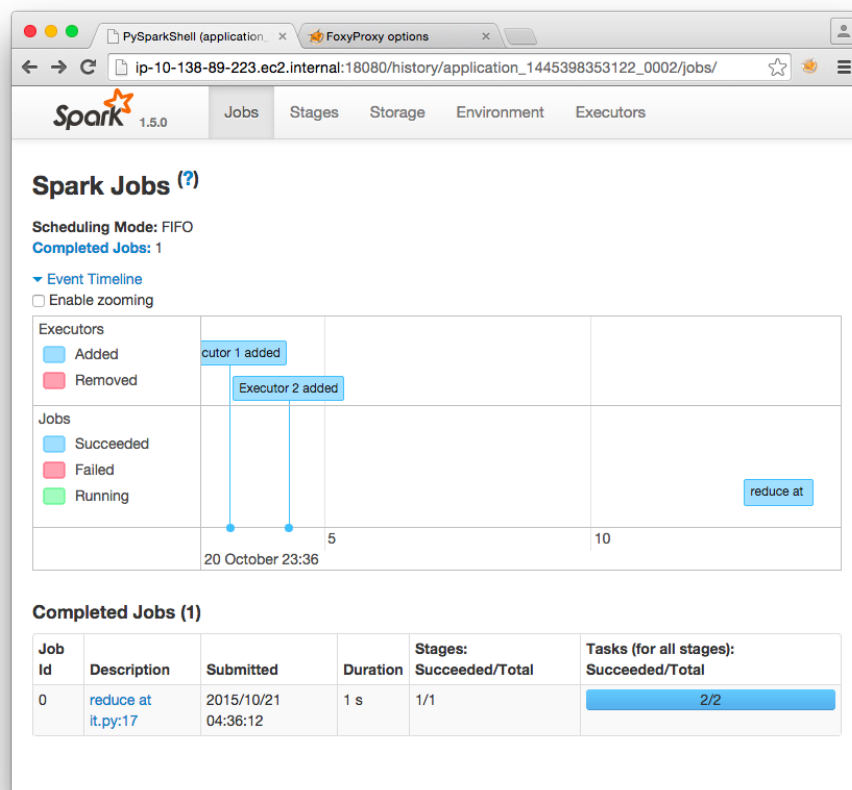


Figure 1: Screenshot of Spark History

3. Please zip up all of your code and your document (use .zip only, please!), or attach each piece of code and your document to your submission individually.
4. Please have the latest version of your code on GitHub. Zip the GitHub files and submit your latest version of assignment work to Blackboard. We will consider the latest version on the Blackboard, but it should exactly match your code on GitHub.
5. **Remember to delete your cluster after the calculation is finished!!!**