**Assignment 6**

**Big Data Analytics**

MET CS777

Faculty - Farshid Alizadeh-Shabdiz, PhD, MBA

## **1 Description**

In this assignment, you will be implementing a regularized, logistic regression to classify text documents.

You will be asked to perform three subtasks:

1. data preparation,  
2. learning (which will be done via gradient descent) and 3. evaluation of the learned model.

## **2 Data**

You will be dealing with a data set that consists of around 170,000 text documents (7.6 million lines of text), and a test/evaluation data set that consists of 18,700 text documents (almost exactly one million lines of text). All but around 6,000 of these text documents are Wikipedia pages; the remaining documents are descriptions of Australian court cases and rulings. At the high level, your task is to build a classifier that can automatically figure out whether a text document is an Australian court case or not.

We have prepared three data sets for your use.

1. The Training Data Set (1.9 GB of text). This is the set you will use to train your logistic regression model.
2. The Testing Data Set (200 MB of text). This is the set you will use to evaluate your model.
3. The Small Data Set (37.5 MB of text). This is for you to use for training and testing of your model locally, before you try to do anything in the cloud.

|  |  |
| --- | --- |
| Google Cloud | |
| Small data set (37.5MB) | gs://metcs777-fa/SmallTrainingData.txt |
| Large training data (1.9GB) | gs://metcs777-fa/TrainingData.txt |
| Test data set (200MB) | gs://metcs777-fa/TestingData.txt |

Table 1: Data set on Google Cloud Storage - URLs

Some Data Details to Be Aware Of. You should download and look at the SmallTrainingData.txt file before you begin. You’ll see that the contents are sort of a pseudo-XML, where each text document begins with a < doc id = ... > tag and ends with < /doc >.

Note that all of the Australia legal cases begin with something like < doc id = “AU1222” ... >. Doc id for an Australian legal case always starts with AU. You will be trying to figure out if the document is an Australian legal case by looking only at the contents of the document and document Id.

Assignment Tasks  
3.1 Task 1 : Data Preparation (2 points)

First, you need to write Spark code that builds a dictionary that includes the 20,000 most frequent words in the training corpus. This dictionary is essentially an RDD that has words as the keys, and the relative frequency position of the word as the value. For example, the value is zero for the most frequent word, and 19,999 for the least frequent word in the dictionary.

Next, you will convert each of the documents in the training set to a TF (“term frequency”) vector that has 20,000 entries. For example, in a particular document, the entry in the 177th value in this vector is a double that captures the frequency of the 177th most common word in this document. Likewise, the first entry in the vector is a double that captures the frequency of the most common word in the corpus in this document.

Then create the TF-idf matrix based on top-20k words similar to the previous assignments.

To get credit for this task, give us the average TF value of the words “applicant”, “and”, “attack”, “protein”, and “court” for court documents and wikipedia documents (Average on Documents). You need to have a print out in your code for these outputs.

This would be then 4 numbers for Wikipedia documents, and 4 numbers for the court cases. Print these values for the **large training data set**.

Report how long the task takes to run.

Note-1: You need to compute the TF-idf vector of the whole data set including training and test data.

Note-2: To implement your script on your laptop you can set the dimension size to smaller number and then changed it later when you run on the cloud.

3.2 Task 2 - Learning the Model (10 Points)

You will then use a gradient descent algorithm to learn a logistic regression model that can decide whether a document is describing an Australian court case or not. Your model should use L2 regularization; you can change the regularization parameter to determine who the parameter impacts the results and get a sense of the extent of the regularization. We have enough data that you might find that the regularization not that important.

I am going to ask that you not just look up the gradient descent algorithm on the Internet and implement it. Start with the LLH function from class, and write down the gradient descent algorithm.

You should run your gradient descent until the L2 norm of the difference in the parameter vector across iterations is very small.

Once you have completed this task, you will get credit by printing out the five words with the largest regression coefficients. These five words are most strongly related with an Australian court case.

Report how long the task takes to run.

Note-3: Remember that the problem is a classification problem with 1 for Australian court cases and 0 otherwise.

Note-4: Cache the important RDDs. You do not want to re-compute the RDDs each time you make a pass through the data during learning. You can create one script for the multiple assignment tasks with different outputs.

Note-5: You pass the whole data in each iteration and not a small sample of it.

Note-6: In general, when debugging your code, the first step is to verify that the loss function is correctly decreasing as the learning progresses as expected.

Note-7: The key thing is that each entry in the RDD should hold the ENTIRE normalized TF vector for a document (this can be stored in any reasonable data structure; it might make sense to store only the non-zero entries in a map from int to double). Then, any loop happens WITHIN the map/reduce lambdas you build.

If you look at the class slides θi is the result of the dot product between the ith data point and the vector of regression coefficients.

Note-9: First implement your logistic regression without Regularization. Then, implement it with regularization, do some tests and find a good regularization value.

3.3 Task 3 - Evaluation of the learned Model (8 Points)

Now that you have trained the model, it is time to evaluate it. You will use your model to predict which of the test docs are an Australian court case. To get credit for this task, you need to compute the F1 score obtained by your classifier.

Next step - look at the text of three false positives that your model produced (that is, Wikipedia articles that your model thought are Australian court cases). Write paragraph describing why you think your model made a mistake.

Is this related to bad documents from Australian legal system?

If you don’t have three false positives, just use the ones that you have (if any)

# 5 Important Considerations

## 5.1. Machines to Use

Be aware that you can choose virtually any configuration for your cluster - you can choose different number of machines, and different configurations of those machines. Note that each setting 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, then you’ll move to a cluster.

Run your Spark jobs over the “large” data using 3 workers machines with 4 cores and 8GB RAM each.

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.

* As you can see on the list price, costs per hour 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 cloud services charge you when you move data around. To avoid such charges, do everything in the north east region, where the data is.
* You should document your code very well and as much as possible.  
  Your code should be compiled on a Unix-based operating system like Linux or MacOS.

# **Submission Format**

Create a single document that has results for all three tasks. For each task, copy and paste the result that your last Spark job saved in the bucket. Also for each task and each Spark job include a screen shot of the **Spark History**.

Please zip 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.

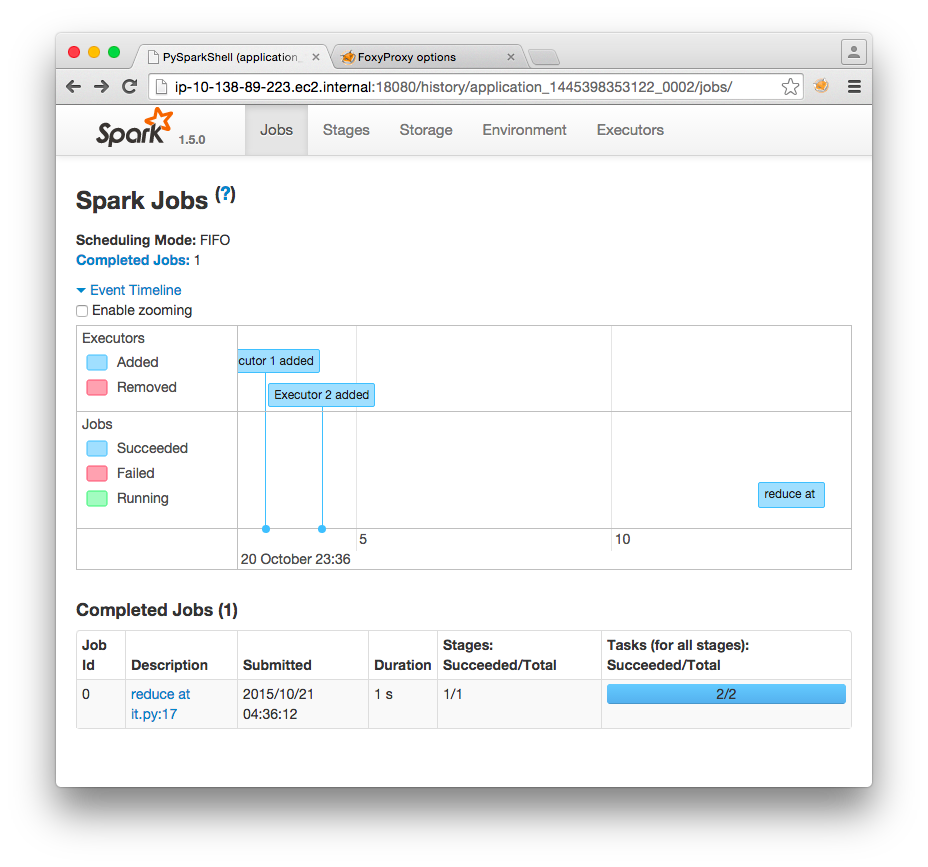


Figure 1: Screenshot of Spark History

# **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.