Assignment 4

MET CS 777 - Big Data Analytics
Classification - Logistic Regression with Gradient Descent
(20 points)

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

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 (this is 7.6 million lines of text in all), and a test/evaluation data set that consists of 18,700 text documents (almost exactly one million lines of text in all). 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 highest level, your task is to build a classifier that can automatically figure out whether a text document is an Australian court case. 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.

	Amzon AWS
Small Training Data Set (37.5 MB of text)	s3://metcs777/SmallTrainingData.txt
Large Training Data Set (1.9 GB of text)	s3://metcs777/TrainingData.txt
Test Data Set (200 MB of text)	s3://metcs777/TestingData.txt

Table 1: Data set on Amazon AWS - URLs

	Google Cloud Storage
Small Training Data Set (37.5 MB of text)	gs://metcs777/SmallTrainingData.txt
Large Training Data Set (1.9 GB of text)	gs://metcs777/TrainingData.txt
Test Data Set (200 MB of text)	gs://metcs777/TestingData.txt

Table 2: 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" ... > that is, the 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.

3 Assignment Tasks

3.1 Task 1: Data Preparation and Model Training (10 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 the word as the key, 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 a particular document, the entry in the 177th value in this vector is a double that tells us the frequency, in this document, of the 177th most common word in the corpus. Likewise, the first entry in the vector is a double that tells us the frequency, in this document, of the most common word in the corpus.

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. You can just print these values for the large training data set.

Notes:

- You need to compute the TF-idf vector of the whole data set including training and test data.
- 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.
- You can write one script for the Task 1 and Task 2 that runs both tasks together.

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 l_2 regularization; you can play with in things a bit to determine the parameter controlling the extent of the regularization. We will have enough data that you might find that the regularization may not be too 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 l_2 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. That is, those five words that are most strongly related with an Australian court case.

Notes:

- Remember that the problem is a classification problem 1 for Australian court cases, 0 otherwise.
- When you debug your code. In general, the first debugging step is to verify that the loss function is correctly decreasing as the learning progresses, as expected.
- Start from a very small learning rate, if the GD works well you can change it.
- The dataset is an unbalanced data set which means that you have less AU cases than Wiki Cases in your data. Think how what would be good initialization for this case.
- 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.
- You can implement the mini-batch GD or full-batch
- The key thing is that each entry in the RDD you build 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 lines. If you look at the class slides θ_i is the result of the dot product between the i^{th} data point and the vector of regression coefficients.
- First implement your logistic regression without Regularization. Then, implement it with regularization, do some tests runs and find a good regularization value.
- Maximum 100 iterations is enough, you do not need to waste lots of Cloud Computation time.
- Optional: In your feature matrix, It make sense to store only the non-zero entries in this vector (working with Sparse representation), just to keep the RAM memory requirements low. However, this is very complex to implement.

3.2 Task 2 - Evaluation of the learned Model (6 Points)

Now that you have trained your model, it is time to evaluate it. Here, you will use your model to predict whether or not each of the testing points correspond to Australian court cases. To get credit for this task, you need to compute for us the F1 score obtained by your classifier.

Also, I am going to ask you to actually look at the text for three of the false positives that your model produced (that is, Wikipedia articles that your model thought were Australian court cases). Write paragraph describing why you think it is that your model was fooled. Were the bad documents about Australia? The legal system?

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

3.3 Task 3 - Using Spark Machine Learning Library (4 Points)

In this task, you can use the Spark MLib or Spark ML library to train your Logistic regression model.

`http://spark.apache.org/docs/latest/ml-classification-regression.html#binomial-logistic-regression`
`http://spark.apache.org/docs/latest/mllib-linear-methods.html#classification`

Use one of the libraries:

- from pyspark.ml.classification import LogisticRegression
- from pyspark.mllib.classification import LogisticRegressionWithLBFGS, LogisticRegressionModel

To create and train your logistic regression model.

Present the evaluation results after 100 iterations. You can assume and document any other model parameters that you need.

4 Important Considerations

4.1 Machines to Use

One thing to be aware of is that you can choose virtually any configuration for your Cloud Cluster (Amazon or Google Cloud) - 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 3 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.

4.2 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.

4.3 Turnin

Create a single document that has results for all three tasks.

Also for each task, for each Spark job you ran, include a screen shot of the Spark History.

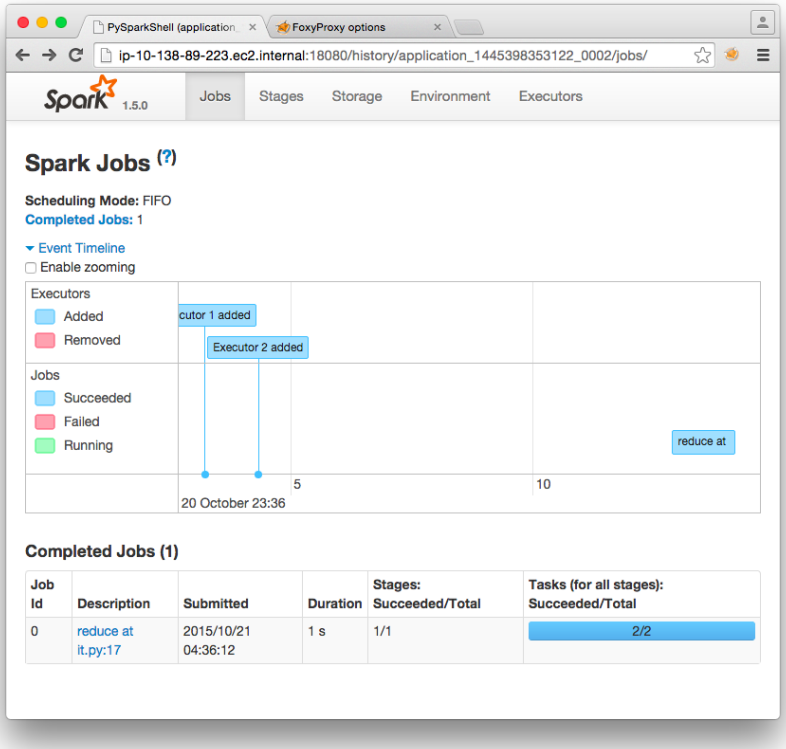

Figure 1: Screenshot of Spark History

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