BigML Assignment 3:
Naive Bayes using GuineaPig

Due: Tuesday, October 13, 23:59 via Autolab

September 29, 2015

Policy on Collaboration among Students

These policies are the same as were used in Dr. Rosenfeld's previous version of 10601 from 2013. The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone, and the student should be ready to reproduce their solution upon request. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment. Specifically, each assignment solution must start by answering the following questions in the report:

- Did you receive any help whatsoever from anyone in solving this assignment? Yes / No. If you answered 'yes', give full details: _____ (e.g. "Jane explained to me what is asked in Question 3.4")
- Did you give any help whatsoever to anyone in solving this assignment? Yes / No. If you answered 'yes', give full details: _____ (e.g. "I pointed Joe to section 2.3 to help him with Question 2".

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism. As a related point, some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be (or may have been) available online, or from other people. It is explicitly forbidden to use any such sources, or to consult people who have solved these problems

before. You must solve the homework assignments completely on your own. I will mostly rely on your wisdom and honor to follow this rule, but if a violation is detected it will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

1 Important Note

In this assignment, **you will be using python**.

This assignment is worth 100 points. In this assignment, you will have to create a Naive Bayes classifier that will run with GuineaPig on Python.

Aurick Qiao (aqiao@cs.cmu.edu), Suraj Dharmapuram (sdharmap@cs.cmu.edu), and Tian Jin (tjin1@andrew.cmu.edu) are the contact TAs for this assignment. Please post clarification questions to the Piazza, and the instructors can be reached at the following email address: *10605-Instructors@cs.cmu.edu*.

2 Introduction

In this part of the assignment, you need to implement Naive Bayes algorithm on GuineaPig. You will be implementing both training and testing phases with the assumption that any part of the data can fit in memory. Unlike the previous two assignments, the goal for the testing phase is to implement a small-memory test algorithm for Naive Bayes - i.e, testing where the vocabulary and test data do not fit into memory.

Following is a high-level sketch of the algorithm:

- Generate event counts from the training data(This phase is identical to what you have already implemented)
- Convert the event counts to key value pairs (k,v) where k is a word, and v is some representation of all the counts for that word.
- Generate requests for word counts from the test file i.e. flatten the test documents into pairs(word,docId).
- Join the flattened test documents with the reorganized event counts.
- Group the join result together so that you can classify the test documents.

For more details on this idea, you should refer to the algorithm mentioned in the second half of the lecture slides from 9/15.

2.1 Introduction to GuineaPig

GuineaPig is a lightweight python library that is similar to Pig, but is easier to learn and debug. You are required to have some working knowledge of Python and Hadoop to be able to use GuineaPig. You express your workflow using high level constructs (such as Join, Augment etc) and GuineaPig spawns off MapReduce tasks behind the scenes to do the compute. GuineaPig provides for you a layer of abstraction over bare-bones Hadoop. You can find more information about GuineaPig (including a tutorial) at `http://curtis.ml.cmu.edu/w/courses/index.php/Guinea_Pig`. It is recommended that you read this document before proceeding with the assignment. You can directly download GuineaPig at `https://github.com/TeamCohen/GuineaPig/archive/master.zip`

2.2 Local execution

In addition to providing an abstraction over Hadoop, GuineaPig programs can also be executed locally, without Hadoop. This feature makes your program much easier to debug. We recommend that you debug locally and test on Hadoop using the Stoat cluster before submitting.

2.3 Using Hadoop on the Stoat cluster

By now, you should all have access to Stoat, which already has an installation of Hadoop ready to use. You just need to copy GuineaPig, with your code, to the cluster:

```
$ scp -r /path/to/GuineaPig <username>@shell.stoat.pdl.local.cmu.edu:~
```

You can SSH to Stoat from within CMU's network:

```
$ ssh <username>@shell.stoat.pdl.local.cmu.edu
```

In order to set up GuineaPig to execute on Hadoop, follow the instructions on the wiki: `http://curtis.ml.cmu.edu/w/courses/index.php/Guinea_Pig#Using_Hadoop`.

3 The Data

We are using the same dataset as the one in Homework 2. These are articles from DBPedia, and the label is the type of the article. There are in total 20 classes in the dataset, and they are from the first level class in DBpedia ontology. For more information about this dataset, you can refer to `http://wiki.dbpedia.org/Downloads2015-04`. The data is of the format:

```
<id>    <label1>,<label2>,<label3>...    w1 w2 w3 w4...
```

The three columns are separated by tab, and the documents are preprocessed so that there are no tabs in the body.

You can find the data in `/afs/cs.cmu.edu/project/bigML/dbpedia_15fall/`. For your convenience, the data also appears on AWS at `s3://bigml15fall/dbpedia/abstract/full/` and `s3://bigml15fall/dbpedia/abstract/small/`. You do not need to copy the data to S3 by yourself.

For this homework, similar to Homework 1, feel free to play with the parameters, tokenizer and output to improve your Naive Bayes classifier.

4 Deliverables

At this point, you should be very familiar with Naive Bayes. In this assignment, you will implement Naive Bayes in GuineaPig, a library in python that works with Hadoop and allows a scripting version of streaming algorithms.

4.1 Autolab Implementation details

We have given you a starter file, nb.py, that contains some setup code. You should implement Naive Bayes in this file, and it should handle both training and testing. We will test your code with the following command:

```
python nb.py --store output \\
   --params trainFile:path/to/train,testFile:path/to/test
```

After running, you should store your final predicitons in the 'output' variable, which will then be stored in 'gpig_views/output.gp' by GuineaPig. The output file should be of the form

```
('id', 'prediction', probability)
('id', 'prediction', probability)
...
```

'output' should be python tuples, where the first element is an id of type string, the second element is prediction of type string, and the third element is a probability of type float. You don't need to worry about the ordering of the output, as long as all the information is there.

For the autograding, you don't need to worry about setting python paths. Just assume that `from guineapig import *` will work.

You should look through the GuineaPig guide at `http://curtis.ml.cmu.edu/w/courses/index.php/Guinea_Pig` The wordcount example might be a good place to start.

4.2 Report

Submit your implementations via AutoLab. You should implement the algorithm by yourself instead of using any existing machine learning toolkit. You should upload your code (including all your function files) along with a **report**, which should solve the following questions:

1. Answer the questions in the collaboration policy on page 1.

2. Recall the Phrase Finding algorithm from lecture. We want to use a dataset for unigrams of the following form:

   ```
   apple 1960 60
   apple 1970 79
   apple 1980 934
   apple 1990 3875
   banana 1960 3
   banana 1970 2
   banana 1990 10
   carrot 1970 3
   carrot 1980 484
   carrot 1990 492
   ...
   ```

   The first column contains a unigram, the second column specifies a decade, and the third column is the number of occurences of the unigram in that decade. We set the foreground corpus to be unigram counts from 1960. The background corpus is unigrams from 1970, 1980, 1990. Given this, how would you aggregate the above dataset into a file containing correct background corpus and forground corpus counts for each unigram? The output we wish to generate is of the following form:

   ```
   apple,fg=60,bg=4888
   banana,fg=3,bg=12
   carrot,fg=0,bg=976
   ...
   ```

   You do not need to write code for this, but you should describe how to generate the background and foreground counts using GuineaPig workflows.

3. Using the Stoat cluster and the abstract.smaller dataset, run your code using parallel modes 1, 2, 4, and 8, and measure the runtime of each. Using parallel mode n, Guinea Pig will use n reducers for each MapReduce task. In addition, it will use n mappers for every intermediate (uses only results from a previous MapReduce step) MapReduce

step in an execution chain. The following command will run your code using parallel mode n:

```
$ python nb.py --store output \\
--opts target:hadoop,viewdir:/user/<username>/gpig_views,parallel:<n>
```

To measure the total runtime of your program, sum together the runtimes between each `map 0% reduce 0%` and `map 100% reduce 100%` for each MapReduce task spawned. You may want to pipe the output to a file in order to save it.

(a) Plot the runtime of your experiments vs. the parallel mode used.

(b) A program is perfectly scalable when the runtime halves if the number of worker machines is doubled. List three reasons why your program might not achieve this ideal scalability.

4. GuineaPig offers you the ability to join two datasets. Being able to understand how this is implemented under the hood will help you better understand how Hadoop works and will help you debug better. In this question, you will implement the Join operation on Hadoop.

   Lets say we have two datasets containing information about people in the following format:

   Dataset1: id,name

   Dataset2: id,p1,p2,p3

   Each person is assigned an unique integer id. Dataset1 contains this id and the person's name. Dataset2 contains a person's id and a long list of integer productIds that the person has purchased.

   We want to find out the number of distinct products purchased by each person. Specifically, the output of your workflow should be of the form: name,number of distinct products This can be done using the Join and Distinct operations in GuineaPig. You need to design a Hadoop workflow to achieve the same result.

   You are NOT required to submit code for this question. We are looking for a high level description of your workflow - it suffices to specify the input-output values to each step of the Hadoop workflow.

   Hint: Think of the steps between the Map and Reduce steps in Hadoop. Think of how you could override these steps to build a scalable solution.

5. Consider the following sequence of GuineaPig commands:

```
def flat1(line):
    for i in xrange(0,len(line),2):
        yield(line[i])

def flat2(line):
    for i in xrange(1,len(line),2):
        yield(line[i])

doc = ReadLines('data.txt') | Map(by = lambda line:line.strip().split(" "))
t1 = Flatten(doc, flat1)
t2 = Flatten(doc, flat2)
t3 = Join(Jin(t1),Jin(t2)) | ReplaceEach(by=lambda(w1,w2):(w1+w2,len(w2)))
```

What do these lines of python do to 'data.txt'?

5 Submission

You must submit your homework through Autolab. You submit a tar file that contains nb.py, as well as a pdf containing your answers to the report questions.

In this homework, there will be a validation link.

- Homework3-validation: You will be notified by Autolab if you can successfully finish your job on the Autolab virtual machines. Note that this is not the place you should debug or develop your **algorithm**. All development should be done on your local computer or Stoat machines. There will be **NO** feedback on your **performance** in the validation. You have unlimited amount of submissions here. To avoid Autolab queues on the submission day, the validation link may be closed 24 hours prior to the official deadline. If you have received a score of 1000 with no errors, this means that you code has passed the validation.
- Homework3: This is where you should submit your tar ball. You have a total of **10 possible submissions**. Your score will be reported, and feedback will be provided immediately.

6 Grading

You will be graded on memory, efficiency and accuracy. You will receive 15 for memory based on your peak memory usage through your run. For efficiency, we will time your code and assign a score out of 15. For accuracy, we will be testing your implementation against a hidden training and testing set on Autolab and assigned a score out of 25. In

this assignment, we do not want you to load any of the datasets into memory. All data should be streamed through GuineaPig, and high memory usage will be penalized.