BigML Assignment 1B: Small Memory Footprint Streaming Naive Bayes

Due Tuesday, January 27 11:59pm via Autolab

January 22, 2015

1 Important Note

This assignment is the second of three that use the Naive Bayes algorithm. You will be expected to reuse the code you develop for this assignment for a future assignment, and **you are expected to use Java for this assignment**.

Rahul Goutam (rgoutam@cs.cmu.edu) and Yipei Wang (yipeiw@cmu.edu) are the contact TAs for this homework. Please post clarification questions to Piazza.

2 Naive Bayes with limited memory

The algorithm for this assignment is exactly the same as for assignment 1, but now we will constrain the size of the memory footprint. In your makefile for this assignment, all Java commands must be followed with -Xmx128m which limits the size of the Java heap space. Also, there will be NO required tokenizer in the assignment, so feel free to use your own. In this homework, you are allowed to write the temporary counts to disk to make your streaming counts fit in the memory.

3 The Data

For this assignment, we will be classifying Wikipedia articles by the languages in which they are also available. The data appears at `/afs/cs.cmu.edu/project/bigML/dbpedia/`

The data format is the same as for Assignment 1. There is one instance per line. The first token of each line is the (comma separated) list of class names , then a tab, then the data. Please do multi-label learning and evaluation as described for Assignment 1.

There are 6 data sets for this assignment. Again, they are in increasing size so that you can debug your code on smaller data. The files that start with *abstract* include Wikipidia text. The files that start with *links* are the outlinks from each wikipedia page.

```
abstract.small.test
abstract.small.train
abstract.test
abstract.train
abstract.tiny.test
abstract.tiny.train
links.small.test
links.small.train
links.test
links.train
links.tiny.test
links.tiny.train
```

4 Deliverables

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).

The training code NBTrain.java should be able to run without the out-of-memory issue, using the command:

```
cat train.txt | java -Xmx128m NBTrain
```

This will output the streaming counts for the NB model, in the tab-separated two column form that was specified in the previous homework. The test code provide a one-per-line prediction for each test case, so the full streaming command is:

```
cat train.txt | java -Xmx128m NBTrain | sort -k1,1 | \
java -Xmx128m MergeCounts | java -Xmx128m NBTest test.txt
```

Here, MergeCounts.java is a function that you need to write to merge the counts of multiple occurrences of the same hash key. The test code produces the following output:

```
Best Class<tab>LogProbability
```

Here, the Best Class is the class with the maximum log probability.

$$ln(p(Y = y)) + \sum_{w_i} ln(p(W = w_i|Y = y)) \tag{1}$$

Note that we will be using natural logarithm. Here's an example of the output format:

```
CCAT -1042.8524
GCAT -4784.8523
...
```

You are also required to submit a report that answers the following questions[5 points each] :

1. What is the purpose of smoothing ? Compare the classifier performance on abstract.small dataset with and without smoothing in terms of correctness and efficiency (time taken).

2. Report the total number of tokens for all documents with label y for all labels on the abstract.train dataset using the tokenizer provided in HW1A.

3. For the streaming version Naive Bayes, we can use a buffer to store the event counter in memory to reduce the message passing cost. (refer to Jan20th slide p58). Please do the following experiments. Run your training pipeline of streaming NB on RCV1 small dataset, and measure the message passing size with different buffer size: 10, 100, 1000 and 10000. (hint: You can redirect your result from NBTrain to a file and list the filesize for measuring the message passing cost).

4. We have ignored the computation of vocabulary size required for smoothing in the implementation of streaming Naive Bayes (You may have done this in memory or used some approximation). How would you use the stream-and-sort design to compute the vocabulary size ? Briefly, in english, describe each step of the algorithm. Please do not include pseudo code or java code.

You should tar gzip the following items into hw2.tgz and submit to the homework 2 assignment via Autolab:

1. NBTrain.java

2. MergeCounts.java

3. NBTest.java

4. Any other auxiliary functions

5. report.pdf

Tar gzip the files directly using tar -cvf hw2.tgz *.java report.pdf. Do NOT put the above files in a folder and then tar gzip the folder. You do not need to upload the saved temporary files. Please make sure your code is working fine on linux.andrew.cmu.edu machines before you submit.

5 Submission

You must submit your homework through Autolab via the Homework2: Small Memory Streaming Naive Bayes link. In this homework, we provide an additional tool called Homework2-validation.

Homework2-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 linux.andrew.cmu.edu machines. This is basically a Autolab debug mode. There will be NO feedback on your performance in this mode. You have unlimited amount of submissions here. To avoid Autolab queues on the submission day, the validation link will be closed 24 hours prior to the official deadline. If you have received a score of 1000, this means that you code has passed the validation.

Homework2 Small Memory Streaming Naive Bayes: This is where you should submit your validated final submission. You have a total of 5 possible submissions. Your performance will be evaluated, and feedback will be provided immediately.

6 Grading breakdown

- Memory Usage : 25
- Runtime : 25
- Validation of testing code : 5
- Final test performance : 25
- Report : 20