BigML Assignment 1: Streaming Naive Bayes

Due Mon, January 27 before class (1:29pm) via Autolab
Late submission: 50% credit before Mon, January 29 1:29pm via Autolab

Policy on Collaboration among Students

These policies are the same as were used in Dr. Rosenfeld's machine learning class of 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

This assignment is the first of three that use the Naive Bayes algorithm. You will be expected to reuse the code you develop for this assignment for future assignments. Thus, the more you adhere to good programming practices now (e.g. abstraction, encapsulation, documentation), the easier the subsequent assignments will be.

Siddarth Varia (varias@cs.cmu.edu) is the contact TA for this homework. Please post clarification questions to Piazza site.

2 Naive Bayes

Much of machine learning with big data involves - sometimes exclusively - counting events. Multinomial Naive Bayes fits nicely into this framework. The classifier needs just a few counters.

For this assignment we will be performing document classification using streaming Multinomial Naive Bayes. We call it streaming because the input and output of each program are read from stdin and written to stdout. This allows us to use Unix pipe "|" to chain our programs together. For example:

```
cat train.txt | java NBTrain | java NBTest test.txt
```

The streaming formulation allows us to process large amounts of data without having to hold it all in memory.

Let y be the labels for the training documents and wi be the ith word in a document. Here are the counters we need to maintain:

(Y=y) for each label y the number of training instances of that class

(Y=*) here * means *anything*, so this is just the total number of training instances.

(Y=y,W=w) number of times token w appears in a document with label y.

(Y=y,W=*) total number of tokens for documents with label y.

The learning algorithm just increments counters:

```
for each example {y [w1,...,wN]}:
    increment #(Y=y) by 1
    increment #(Y=*) by 1
    for i=i to N:
        increment #(Y=y,W=wi) by 1
    increment #(Y=y,W=*) by N
```

You may elect to use a tab-separated format for the event counters as well: eg, a pair <event,count> is stored on a line with two tab-separated fields. Classification will take a new documents with words w1,...,wN and score each possible label y with the log probability of y (as covered in class).

For now (hint, hint), you may keep a hashtable in memory, with keys like "Y=news", "Y=sports,W=aardvark", etc. You may NOT load all the training documents in memory. That is, you must make one pass through the data to collect the count statistics you need to do classification. Then, write these counts (feature dictionary) to disk via stdout.

Important Notes:

- At classification time, use Laplace smoothing with $\alpha = 1$ as described here: `http://en.wikipedia.org/wiki/Additive_smoothing`.
- You may assume that all of the test documents will fit into memory.
- With the exception of the test set, all files should be read from stdin and written to stdout
- You must use this function to change documents into features (do not lowercase or remove stopwords):

```
static Vector<String> tokenizeDoc(String cur_doc) {
        String[] words = cur_doc.split("\\s+");
        Vector<String> tokens = new Vector<String>();
        for (int i = 0; i < words.length; i++) {
              words[i] = words[i].replaceAll("\\W", "");
                if (words[i].length() > 0) {
                        tokens.add(words[i]);
                }
        }
        return tokens;
}
```

3 The Data

For this assignment, we are using the Reuters Corpus, which is a set of news stories split into a hierarchy of categories. There are multiple class labels per document. This means that there is more than one correct answer to the question "What kind of news article is this?" For this assignment, we will ignore all class labels except for those ending in CAT. This way, we'll just be classifying into the top-level nodes of the hierarchy:

- CCAT: Corporate/Industrial

- ECAT: Economics
- GCAT: Government/Social
- MCAT: Markets

There are some documents with more than one CAT label. Treat those documents as if you observed the same document once for each CAT label (that is, add to the counters for all labels ending in CAT). If you're interested, a description of the class hierarchy can be found at `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a02-orig-topics-hierarchy/rcv1.topics.hier.orig`.

The data for this assignment is at: `/afs/cs.cmu.edu/project/bigML/RCV1`
Note that you may need to issue the command *kinit* before you can access the afs files. The format is one document per line, with the class labels first (comma separated), a tab character, and then the document. There are three file sets:

```
RCV1.full.*
RCV1.small.*
RCV1.very_small.*
```

The two file sets with "small" in the name contain smaller subsamples of the full data set. They are provided to assist you in debugging your code. Each data set appears in full in one file, and is split into a train and test set, as indicated by the file suffix.

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) along with a **report**, which should solve the following questions:

1. What changes could you make to reduce the amount of RAM required for the dictionary? (5 points)
2. Right now we're basically ignoring the fact that there are multi-labeled instances in the train/test sets. How would you extend your algorithm to enable it to predict multiple labels? (5 points)
3. Why should we use Laplace smoothing? What will happen if we don't use any smoothing? (5 points)
4. What is the relationship between Laplace smoothing and Dirichlet prior? (5 points)
5. Your answers to collaboration policy (on the first page of this handout).

The training code NBTrain.java should be able to run, using commands:

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

This will output the streaming counts for the NB model, in the tab-separated two column form:

```
Y=CCAT,W=he 3.0
Y=CCAT,W=saw 1.0
Y=CCAT,W=her 3.0
Y=CCAT,W=duck 4.0
Y=CCAT,W=or 1.0
Y=CCAT,W=* 123.0
Y=CCAT 10.0
Y=* 10.0
...
```

The test code provide a one-per-line prediction for each test case, so the full streaming command is:

```
cat train.txt | java NBTrain | java NBTest test.txt
```

which produces output in the following format:

```
Best Class<tab>Log prob
```

which is essentially the classification results, including the log probabilities of the best class y:

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

Here, Best Class is the class with the maximum log probability (as in Equation 1), and the last field is the log probability. L is the length of the testing document. Note that we're using the natural logarithm here. Here's an example of the output format:

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

You should tar gzip the following items into **hw1.tgz** and submit to the homework 1 assignment under Autolab:

- NBTrain.java
- NBTest.java

- and all other auxiliary functions you have written
- report.pdf

Tar gzip the files directly using "tar -cvf hw1.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 "Homework1: Streaming Naive Bayes" link. In this homework, we provide an additional tool called "Homework1-validation":

- Homework1-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 debugging 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. Please proceed to the Homework1: Streaming Naive Bayes link for formal submission.
- Homework1: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

You will be graded based on the correctness (25 points) and efficiency (25 points) of the training code, the validation of your test code (5 points), and your final test performance (25 points). Note that a good implementation typically has a total of 70+/80, but it is very unlikely that one will receive 80 in total. The report will be graded manually (20 points).