BigML Assignment 2: Small Memory Footprint Streaming Naive Bayes

Due Tuesday, February 4 11:59pm via Blackboard

January 28, 2013

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**.

Bin Zhao (binzhao@cs.cmu.edu) is the contact TA for this homework. Please post clarification questions to the Google Group:
`machine-learning-with-large-datasets-10-605-in-spring-2013`

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.

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 a compressed archive (zip, tar, etc) of your code. With it, include a makefile so that

- The command *make demo* will build and test a classifier using the training and test files for the "abstract.tiny" dataset.
- The command *make test TESTFILE="test.txt"* builds a classifier using the "abstract.tiny" training file, and then classifies the documents in the file provided (test.txt in this example).

Your classification code should print out the classification results, with format the same as Assignment 1.

In addition to your code and a makefile, please include a pdf document with:

1. The output of your "make demo" command, as well as the percent correct on the other five test sets (using classifiers trained with the corresponding train set).
2. Answers to the following questions:
   (a) Compare and discuss the performance for the full links vs full abstract data.
   (b) For the last assignment we were classifying into the top level of the Reuter's hierarchy, and we asked you how you would extend the Naive Bayes Algorithm to handle news articles with multiple labels. The Reuter's hierarchy is actually several levels deep, and the links denote an *is-a* relationship. For example, Figure 1 shows the *Markets* subtree. The tree shows that an article about Metals Trading is an article about the Commodity Market, which is an article about Markets. The full list of class codes can be found at `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a02-orig-topics-hierarchy/rcv1.topics.hier.orig`.

i. Consider the meaning of multiple labels in a hierarchy. An article cannot be about Metals trading if it is not also about the Commodity Market. How can you extend your algorithm to ensure such inconsistencies in classification don't occur?

ii. As the RCV1 class labels become more specific, there are fewer training instances, and the word counts can become more sparse. So, while you may not have seen word **A** in a *Forex Markets* article, you may have seen it in a *Money Markets* article. How can you leverage the information from the articles from parent classes to improve your smoothing algorithm for children classes? (Hint: It may be wise to reflect upon the lecture of January 24, and specifically the strategy for affect/effect classification when we encountered a new unseen word context at test time.)

BONUS question:

1. Using a local copy of the RCV1.small_train.txt file from Assignment 1, compare the performance of creating your Naive Bayes feature dictionary for last assignment and this assignment. Time all parts of the dictionary creation (including, for example, sorting and combining counts for your Assignment 2 solution). Average over 10 calls. Please include in your write up the commands you used to do this comparison.

5 Marking breakdown

- Code correctness and makefile functionality [**60 points**].
- 1 [**10 points**]
- 2a [**10 points**]
- 2b i [**10 points**]
- 2b ii [**10 points**]
- Bonus question: [**10 points**]

Figure 1: The *Markets* subtree of the RCV1 hierarchy.

6 Hints/Good to know

Unix sort can handle very large files. This is helpful when you need to collect together the counts for a particular key. To ensure sort behaves properly, you should set the following environment variable:

```
LC_ALL='C'
```

Get it to sort using tabs by setting this flag:

```
-t $'\t'
```

And, when files are large it may be a good idea to tell sort where to store its temporary files:

```
-T /some/dir
```