Assignment 10: Naive Bayes

10601a: Machine Learning (Spring 2014)

TA : Seth Flaxman *(sflaxman@andrew.cmu.edu)*

Assigned: April 3, 2014 at 1:30pm
Due: April 9, 2014 at 11:59pm
This assignment is current as of 20:11 on Thursday 3rd April, 2014.

Policy on Collaboration among Students

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 must contain a file named collaboration.txt where you will answer the following questions**:

- 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. All violations (even first one) will carry severe penalties, up to failure in the course, and will in addition always be reported to the university authorities.

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 strictly enforce this policy, and 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 above.

1 Building a Naive Bayes Classifier

Your task for this problem is to implement a Naive Bayes classifier to classify a political blog as being "liberal" or "conservative". After implementing the basic algorithm, you will be asked to extend it in a few ways and analyze its behavior. The data set to be used for this assignment is a set of self-identified liberal and conservative blogs[1]. We have rearranged and preprocessed the data slightly so as to make it easier to do this assignment. The data set is available on autolab. The dataset contains a total of 120 blogs, among which 56 were identified by their author as being "liberal", with the remaining 64 considered by their author to be "conservative". Each blog is stored in a separate file, within which each line is a separate word. Files with a name of the form "lib*.txt" are liberal blogs, and files with a name of the form "con*.txt" are conservative.

Implement the Naive Bayes algorithm, using smoothing, as shown in Table 6.2 on page 183 of Tom Mitchell's textbook. Your program will take a set of labeled training examples in split.train and a set of test examples split.test, and classify them using a Naive Bayes classifier. You can use python or Java. Assume that all data files are in the same directory as your program. Don't upload the data when you upload your program to autolab. Your program should output predicted labels for the test data, one per line, in the order they are listed in split.test, and calculate the accuracy on the test dataset. You should ignore case—treat "President" and "president" as the same word type. Do not worry about any other type of preprocessing—leave non-alphabetic symbols as they are.

```
$ python nb.py split.train split.test
L
L
...
C
C
C
L
Accuracy: 0.8056
```

Use 4 digits after the decimal place, (i.e. use "%.04f").

Tips:

- Before you start coding, take a look at all the questions below so you can think about how you want to structure your code.
- Beware of numerical underflow due to products of very small numbers. If $p_1 = 1 \times 10^{-10}$, $p_2 = 2 \times 10^{-20}$ and $p_3 = 1 \times 10^{-30}$ then $p_1 \cdot p_2 \cdot p_3 = 2 \times 10^{-60}$. In Java, the minimum value a float can take is 1.4×10^{-45}, so if you simply multiplied very small floats together you would get 0, which would be very bad. A simple solution is to transform probabilities with log and use addition in place of multiplication: $\log(p_1 \cdot p_2 \cdot p_3) = \log(p_1) + \log(p_2) + \log(p_3)$. If you need to report a probability, use exponentiation.

[1] http://politicalbloglistings.blogspot.com/, accessed 31 October 2008

2 Interpreting the output

Write a program topwords.py / topwords.java to take a training dataset as input and print out the top 20 words with the highest word probabilities in the liberal category as well as in the conservative category (i.e., $\hat{p}(w|C_{lib})$ and $\hat{p}(w|C_{cons})$, where C_{lib} is the liberal class and C_{cons} is the conservative class). The format should be one word per line, sorted with the highest probability first. Print the probabilities with 4 digits after the decimal place (i.e. use "%.04f"). Output the top 20 liberal words and probabilities first, then print a blank line, then print the top 20 conservative words and probabilities:

```
$ python topwords.py split.train
liberalword1 .0911
liberalword2 .0505
...
liberalword20 .0011

conservativeword1 .1013
conservativeword2 .0905
...
conservativeword20 .0021
```

In topwords.txt, answer the following question: Do the two lists look different? Are there any overlapping words? In general, what kind of words are they?

3 Stop words

It is general practice to preprocess datasets and remove words like "the", "a", "of", etc. before training a classifier. Rather than prespecifying a list of stop words, we can simply exclude the N most frequent words. Write a new classifier nbStopWords.py based on nb.py which additionally takes a parameter N and excludes the N most frequent words from its vocabulary before training the classifier. Here is the syntax for $N = 10$; the output should look like the output from nb.py:

```
python nbStopWords.py split.train split.test 10
```

Investigate various settings for N—what values seem to improve the classifier?—and put your observations in nbStopWords.txt.

4 Smoothing

As discussed in section 6.9.1.1 on page 179 of Mitchell, estimating probabilities based on observed fractions can cause problems when observed counts are small or 0. We will investigate various approaches to this. Following Mitchell's notation on p. 179 and p. 183 we have:

$$P(w_k \mid v_j) \leftarrow \frac{n_k + 1}{n + |\text{Vocabulary}|}$$

In this question, we will consider:

$$P(w_k \mid v_j) \leftarrow \frac{n_k + q}{n + q \cdot |\text{Vocabulary}|}$$

Write a program smoothing.py based on nb.py which additionally takes a single parameter q.

Here is the syntax for $q = 1$; the output should be identical to the output from nb.py:

```
python smoothing.py split.train split.test 1
```

Try your program with $q = 0, .1, .5, 1, 5$—what values seem to improve the classifier?—and put your observations in smoothing.txt.

5 Log Odds

Write a program topwordsLogOdds.py based on topwords.py to print out the top 20 words with the highest log-odds ratio for each class, i.e. $\log \frac{\hat{p}(w|C_{lib})}{\hat{p}(w|C_{cons})}$ and $\log \frac{\hat{p}(w|C_{cons})}{\hat{p}(w|C_{lib})}$. Assume the same input and output format as in topwords.py. Print the log-odds with 4 digits after the decimal place (i.e. use "%.04f"). Use natural log (log base e):

```
python topwordsLogOdds.py split.train
```

In topwordsLogOdds.txt answer the following: What kind of words did you find? Are there any overlapping words between the two lists? How are these words different from what you printed out in the previous question?

6 Autolab Submission

Submit a .tgz containing your source code, written assignments, and a file collaboration.txt). You can create that file by running `tar -cvf hw10.tgz *.py *.txt`. **DO NOT** put the above files in a folder and then tar gzip the folder. You must submit this file to the "homework10" link on Autolab.