COMP 135 – Machine Learning – Fall 2016

Empirical/Programming Assignment 2

Due date: Wednesday, 10/19 (by the beginning of class, both paper and electronically)

In this assignment you will implement two variants of the Naive Bayes algorithm with smoothing as applied to text categorization.

1 Text Categorization with Naive Bayes

We will be using a subset of the 20 newsgroups dataset[1] that has already been pre-processed for the assignment. The data files are accessible through the course web page.

More specifically, each dataset for this assignment includes text documents containing articles from two different categories. The task is to identify the category of each document. We have two datasets of similar structure but different contents. The first dataset includes articles that were posted on newsgroups `comp.sys.mac.hardware` and `comp.sys.ibm.pc.hardware`. Each document is a message posted to one of the groups. Our class labels `Yes` and `No` capture membership in these groups (Yes corresponds to ibm). Similarly the second dataset includes articles that were posted on newsgroups `rec.sport.baseball` and `rec.sport.hockey`. Our class labels are again `Yes` and `No` (Yes corresponds to baseball). Our goal is to learn a classifier that, when given a new article, assigns it to the correct group.

1.1 The Data Files

For each dataset, we have roughly 1200 examples and each example resides in a separate text file. When you unzip the data provided on the web page you will find files in subdirectories `ibmmac/` and `sport/`. To help you in your task we have "cleaned" the text files (removed non-alphabet characters).

The class label of each example is indicated in a separate "index" file. We have already split the intended data into a *training* portion and a *test* portion, and each is given in a separate index file (`index_train` and `index_test` respectively. Here are a few lines from the index file to illustrate the syntax.

```
589|Yes|
590|Yes|
591|Yes|
592|No|
593|No|
```

589, 590 etc are the file names of examples and `Yes` and `No` are the labels.

We have also provided `index_train_short` which you can use while developing and debugging your program to reduce run time. However, the experiments requested in the report are with the full training index.

1.2 Two Variants of Naive Bayes for Text

In this assignment we are asking you to implement two variants. The first variant, **Type1**, is exactly as in the slides. In this case a document has as many features as the number of words in it and a "token feature" has as many possible values as words in the vocabulary (all words in training files).

The second variant, **Type2**, is a simple bag of words representation. In this case the number of features is the number of words in the vocabulary. Each feature is binary, having value 1 when the corresponding word appears in the document and value 0 otherwise.

To illustrate, if the vocabulary is {`day, night, desk, chair, any`} and the document is `night desk any day` then Type1 represents the document as having 4 features with values `f1=night, f2=desk, f3=any, f4=day` and Type2 represents the document as having 5 features (implicitly ordered as in the vocabulary) with values 11101.

[1] http://qwone.com/~jason/20Newsgroups/

2 Implementation

2.1 Naive Bayes for Text Categorization

In this assignment you may assume separate index files for train and test. Given a training set index file for Naive Bayes you need to parse each file and record the counts for documents, words, classes as needed by the two variants. These counts constitute the learning process since they determine the prediction of Naive Bayes.
Now, given the test index file, you read each example, calculate the scores for each class, (possibly with smoothing) and test the prediction. Note, that products of small numbers (probabilities) will quickly lead to underflow problems. Due to that you should work with sum of log probabilities instead of product of probabilities. Recall that $a \cdot b \cdot c > d \cdot e \cdot f$ iff $\log a + \log b + \log c > \log d + \log e + \log f$ so that working with the logarithms is sufficient.
If a word in a test example did not appear in the training set at all (i.e. in any of the classes), then simply skip that word when calculating the score for this example. However, if the word did appear with some class but not the other then use the counts you have (zero for one class but non zero for the other).
Your implementation should support both Type1 and Type 2 variants and the experiments will compare their performance.

2.2 Smoothing

During prediction use the formula from class for smoothing $\frac{\#(w \wedge c)+m}{\#(c)+mV}$ where m is a parameter. The meaning of the other three terms is different for Type1 and Type2 since they have different features.
For Type1 each "token feature" has as many values as words in the vocabulary (all words in training files). $\#(w \wedge c)$ is the number of word tokens in documents of class c that are the word w, $\#(c)$ is the number of word tokens in documents of class c, and V is the vocabulary size. For example, if $m = 1$ and $V = 1000$, and out of 10000 word locations in documents of class c the word w appeared 100 times, the probability is estimated to be $\frac{100+1}{10000+1000}$.
For Type2 each feature is Binary i.e., it has two possible values. $\#(w \wedge c)$ is the number of documents in class c that include the word w, $\#(c)$ is the number of documents in class c, and V=2. For example, if $m = 1$ and there are 500 documents of class c of which 200 include the word w, the probability that the feature corresponding to w has value 1 is estimated to be $\frac{200+1}{500+2}$ and the probability that the value is 0 is estimated to be $\frac{300+1}{500+2}$.
In the evaluation requested below, you should run Naive Bayes with smoothing parameter $m = 0, 0.1, 0.2, \ldots, 0.9$ and $1, 2, 3, \ldots, 10$ (i.e., 20 values overall). Plot the cross validation performance as a function of the smoothing parameter.

2.3 Producing Learning Curves

Learning curves evaluate how the predictions improve with increasing train set size. In our context we keep the test set fixed but train with random portions of increasing size of the training set. In the evaluation requested below please evaluate learning performance for 10 different training set sizes $0.1N, 0.2N, \ldots, 0.9N, N$.
To simplify your work we have already randomized the order of examples in the training index so that you can obtain the corresponding random portion by simply using the first K examples in the training index.
The accuracy in these runs, plotted again train set size, constitutes the learning curve.

3 Evaluation

Once all the above is implemented please run the following tests and report the results. We are aware that this is a good number of runs of the Naive Bayes algorithm over train/test configurations. Please allocate enough time so that your code can complete these runs.

- For each of the two datasets (ibm/mac and sport) and types (Type1 and Type2) run the code to generate learning curves for Naive Bayes without smoothing (m=0) and with Laplace Smoothing (m=1). This produces 8 plots overall and it is useful to put the 4 plots for each dataset together.

  Please discuss the results: what observations can you make about the performance of the different algorithms and settings?

- Now use the full train sets and for each of the two datasets (ibm/mac and sport) and types (Type1 and Type2) run the code to evaluate performance as a function of m. This produces 4 plots overall and it is useful to put the 2 plots for each dataset together.

  Please discuss the results: what observations can you make about the performance of the different algorithms and settings?

4 Submitting your assignment

- You should submit the following items both electronically and in hardcopy:
  (1) All your code for data processing, learning algorithms, test program, and the experiments. Please write clear code and document it as needed.

  Please make sure that your code runs on *homework.eecs.tufts.edu*. Please include a README file with instructions how to compile and run your code to reproduce the results of experiments. If this is nontrivial please include a script to run your code.
  (2) A short report with the results and plots as requested and a discussion with your observations from these plots. Please make sure to address the questions posed above in your discussion.

- **Please submit a hardcopy** in class.

- **Please submit electronically using provide** by 4:30 (class time). Put all the files from the previous item into a zip or tar archive (no RAR please). For example call it `myfile.zip`. Then submit using `provide comp135 pp2 myfile.zip`.

Your assignment will be graded based on the **code**, its **clarity, documentation and correctness**, the **presentation** of the results/plots, and their **discussion**.