Empirical/Programming Assignment 2

Due date: Monday, 10/16
Please note new submission times: electronic submission is due by 12:00 noon of the due date. Paper copies are due by the beginning of class.

1 Introduction

In this assignment you will implement the Naive Bayes algorithm with smoothing and evaluate it using cross validation on the task of sentiment analysis (as in identifying positive/negative product reviews).

2 Text Data for Sentiment Analysis

We will be using the “Sentiment Labelled Sentences Data Set”\(^1\) that includes sentences labelled with sentiment (1 for positive and 0 for negative) extracted from three domains imdb.com, amazon.com, yelp.com. These form 3 datasets for the assignment.

Each dataset is given in a single file, where each example is in one line of that file. Each such example is given as a list of space separated words, followed by a tab character (\(\backslash t\)), followed by the label, and then by a newline (\(\backslash n\)). Here is an example from the yelp dataset:

Crust is not good. 0

The data, which is hosted by the UCI machine learning repository, is linked through the course web page.

3 Implementation

3.1 Naive Bayes for Text Categorization

In this assignment you will implement “Naive Bayes for text categorization” as discussed in class. In our application every “document” is one sentence as explained above. The description in this section assumes that a dataset has been split into separate train and test sets.

Given a training set for Naive Bayes you need to parse each example and record the counts for class and for word given class for all the necessary combinations. These counts constitute the learning process since they determine the prediction of Naive Bayes.

Now, given the test set, you parse 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.

Important points for prediction: 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).

\(^1\)https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences
3.2 Smoothing

During prediction use the formula from class for smoothing. Recall that we are using the feature of type “token in document is word \(w\)”, so that each “token feature” has as many values as words in the vocabulary (all words in training files). In particular to estimate \(p(w|c)\) for word \(w\) and class \(c\) use \(\frac{\#(w \cap c) + m}{\#(c) + mV}\) where \(m\) is a parameter, \(V\) is the vocabulary size, \(\#(w \cap c)\) is the number of word tokens in examples of class \(c\) that are the word \(w\), \(\#(c)\) is the number of word tokens in examples of class \(c\).

For example, if \(m = 1\) and \(V = 100\), and out of 10000 word locations in examples of class \(c\) the word \(w\) appeared 100 times, the probability is estimated to be \(\frac{100 + 1}{10000 + 100}\).

3.3 Cross Validation

Implement code to read and parse a dataset and prepare it for 10-fold stratified cross validation. The pseudo-code for generating such folds is given in the lecture slides, and briefly explained here. The simplest way to get random folds is to randomly permute the order of examples and then split the permuted indices sequentially. To get stratified folds you need to do this for each class separately and then put together the different portions. For example, assume the initial classes are 1, \ldots , 10 (positive examples) and 11, \ldots , 20 (negative examples) and that our permutations are \([3, 2, 4, 5, 7, 8, 1, 9, 6, 10]\) and \([17, 13, 14, 18, 11, 15, 12, 16, 19, 20]\). Then for 2-fold cross validation we produce the folds \([3, 2, 4, 5, 7, 17, 13, 14, 18, 11]\) and \([8, 1, 9, 6, 10, 15, 12, 16, 19, 20]\). In \(k\)-fold cross validation we generate \(k\) train/test splits where the \(i\)th split is given by training on all but the \(i\)th portion and testing on the \(i\)th portion.

It is important to randomize the ordering in the final training sets in case the algorithm is sensitive to example ordering. While Naive Bayes is not sensitive to ordering, our learning curve experiment of the next section is sensitive. For example, if we use initial portions of the fold \([3, 2, 4, 5, 7, 17, 13, 14, 18, 11]\) in that order it will only include positive examples, which will skew the results.

3.4 Learning Curves with Cross Validation

Learning curves evaluate how the predictions improve with increasing train set size. To measure this with cross validation we follows the following procedure. First generate the folds for cross validation, call these \(\text{train}_i, \text{test}_i\), for \(i = 1, \ldots , k\). Say \(\text{train}_i\) has \(N\) examples. Then use subsamples of \(\text{train}_i\) (you can use an initial portion if the data was randomized) of sizes \(0.1N, 0.2N, \ldots , 0.9N, N\) as train sets and evaluate the prediction on \(\text{test}_i\) measuring the accuracy in each case. Repeat this for each \(i\) and then calculate the average and standard deviation for each size. This constitutes the learning curve.

4 Experiments

Once all the above is implemented please run the following tests and report the results. You are requested to run Naive Bayes many times on combinations of datasets, parameters, and folds. With a good implementation the overall time should not be high. But please plan ahead to make sure you can complete the assignment on time.

- For each of the 3 datasets run stratified cross validation to generate learning curves for Naive Bayes without smoothing \((m=0)\) and with Laplace Smoothing \((m=1)\). For each dataset, plot averages and standard deviations of the accuracy as a function of train set size (for \(m = 0\) and \(m = 1\) together in the same plot). What observations can you make about the results?

- Run stratified cross validation for 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. What observations can you make about the results?
5 Programming Language and use of Libraries/Modules

You may write your program in any language as long as your code runs on our server homework.eecs.tufts.edu. You are of course allowed to use basic I/O, math library, randomization and other basic language facilities. But you should write your own code for all the portions described above.

6 Submitting your assignment

6.1 What to submit

You should submit the following items both electronically and in hardcopy:

- *All your code* for data processing, learning algorithms, experiments, test programs, etc. 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 compile and run your code.

  Your submitted code should assume that the data files are in the same directory as the program. There is no need to submit the original data files. There is no need to submit any intermediate files or results. These should be generated when your program is run.

- A short report with the results and plots as requested and a discussion with your observations from these plots. Please make sure that your discussion responds to questions posed in the assignment text.

6.2 When and How to Submit

- Please submit electronically using provide by 12:00 noon of the due date. Put all the files mentioned in the previous subsection into a zip or tar archive (no RAR please). For example call it myfile.zip. Then submit using provide comp135 pp2 myfile.zip.

- Please submit a printed copy (of everything) by the beginning of class (4:30pm).

6.3 Grading

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