Coursework **Fall 2021**
Machine Learning (B555) **Roni Khardon**
Department of Computer Science, IU Bloomington

Programming Project 1

This assignment is due by Monday 9/20, 11:59pm via Canvas.

You can write your code in any programming language so long as we are able to test it on SICE servers (python and matlab should be ok; please ask about other options). We plan to run some or all or submitted code for further testing and validation.

You may use standard I/O, math, and plotting libraries. However, other than these, please write all the code yourself without referring to special libraries or modules, i.e., no scikit, no pandas, no other data processing libraries etc.

1 Overview

In this assignment you will implement the Naive Bayes algorithm with maximum likelihood and MAP solutions 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 `(\t)`, followed by the label, and then by a newline `(\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.

[1] https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

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 (for both maximum likelihood and MAP solutions).

Now, given the test set, you parse each example, calculate the scores for each class 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. However, note that unless your programming environment handles infinity natively, will need to handle $\ln(0) \triangleq -\infty$ as a special case in your code.

Important point 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).

3.2 Maximum Likelihood and MAP Solutions

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). The maximum likelihood (and MAP) estimates of parameters are given by the solution for a Discrete distribution (with a Dirichlet prior) for its parameters.

The maximum likelihood estimate of $p(w|c)$ for word w and class c is $\frac{\#(w \wedge c)}{\#(c)}$ where $\#(w \wedge c)$ is the number of word tokens in examples of class c that are the word w and $\#(c)$ is the number of word tokens in examples of class c.

If we use a prior with parameter vector where all entries are equal to $m+1$, that is, $(m+1)\mathbf{1}$, the effect is that of adding a pseudo count of m to all entries. In this case, the MAP estimate of $p(w|c)$ is $\frac{\#(w \wedge c)+m}{\#(c)+mV}$ where V is the vocabulary size and other parameters are as above. For example, if $m = 1$ and $V = 1000$, and out of 10000 word locations in examples of class c the word w appeared 100 times, the probability is estimated to be $\frac{100+1}{10000+1000}$.

This estimate is often referred to as "smoothing" in the literature because it smoothes out the maximum likelihood values and avoids 0/1 extreme solutions. Note that the maximum likelihood solution is simply the special case of smoothing with m=0.

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 ith split is given by training on all but the ith portion and testing on the ith 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 train$i$, test$i$, for $i = 1, \ldots, k$. Say train$i$ has N examples. Then use subsamples of 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 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 reasonable implementation the overall run time should not be high ($\leq$ 1min). 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 with $m = 0$ and with $m = 1$. For each dataset, plot averages of the accuracy and standard deviations (as error bars) as a function of train set size. It is insightful to put both $m = 0$ 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 accuracy and standard deviations as a function of the smoothing parameter. What observations can you make about the results?

Submission

Please submit two separate items via Canvas:

(1) A zip file `pp1.zip` with all your work and the report. The zip file should include: (1a) Please write a report on the experiments, their results, and your conclusions as requested above. Prepare a PDF file with this report. (1b) Your code for the assignment, including a README file that explains how to run it. When run your code should produce all the results and plots as requested above. Your code should assume that the data files will have names as specified above and will reside in sub-directory pp1data/ of the directory where the code is executed. We will read your code as part of the grading – please make sure the code is well structured and easy to follow (i.e., document it as needed). This portion can be a single file or multiple files.

(2) One PDF "printout" of all contents in 1a,1b: call this `YourName-pp1-everything.pdf`. One PDF file which includes the report, a printout of the code and the README file. We will use this file as a primary point for reading your submission and providing feedback so please include anything pertinent here.

Grading

Your assignment will be graded based on (1) the clarity of the code, (2) its correctness, (3) the presentation and discussion of the results, (4) our ability to test the code.