INFO 284, Spring 2018, First Obligatory Group Assignment

Naive Bayes for Sentiment Analysis.

Deadline for feedback March 19, 2018. Final deadline June 1, 2018 (Inspera)

Outline

The goal of this group project assignment is to obtain practical knowledge of the naive Bayes algorithm by creating a classifier, **without using the scikit-learn library**, that can predict whether a movie review is positive or negative. This is a common text categorisation task which is known as *sentiment analysis*. The words used in a review provide excellent cues for the sentiment of the reviewer towards the movie. For example, words such as *love*, *fantastic*, and *avoid*, *awful*, *disappointed* are very informative in predicting the following reviews:

+ ...I loved this. And didn't it look fantastic?! I love these afternoon popcorn movies...

- ...Avoid this super duper awful movie...if you watched it you will be disappointed...

Dataset

A dataset of movie reviews (similar to the ones above) is provided together with their associated binary sentiment labels. It contains 50,000 reviews split evenly into 25k train and 25k test sets.

In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets.

The data can be found in a folder data.zip. The folder contains the following files: imdb.vocab, imdbEr.txt, README, test (folder), train (folder). The test folder contains: labeledBow.feat, neg (folder), pos(folder), urls_neg.txt, urls_pos.txt. The train folder contains: labeledBow.feat neg (folder), pos(folder), unsup(folder), unsupBow.feat, urls_neg.txt, urls_pos.txt, urls_unsup.txt.

Discrete Naive Bayes

We want a classifier that can calculate the probability that a review is positive or negative, i.e., given a new review $\mathbf{x}$, estimate $P(y = 1|\mathbf{x})$ and $P(y = 0|\mathbf{x})$. These are known as the posterior probabilities of the positive ($y = 1$) and negative ($y = 0$) class. To calculate the posterior probability for y, we use the Bayes rule:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}.$$

Note that a review $\mathbf{x}$ is a vector consisting of attributes $w_i, i = 1, \ldots, n$, $\mathbf{x} = w_1, \ldots, w_n$. Here, we will use the words w_i contained in a review $\mathbf{x}$ as attributes, which is a representation known as the *unigram* model.

In the naive Bayes model, each attribute w_i is *conditionally independent* from any other attribute in the review $\mathbf{x}$. This means that the presence of one word in a review has no effect on the probability of a different word being present in the same review.

Taking into account attributes and their conditional independence, we can reformulate the Bayes rule for the positive class as follows:

$$P(y = 1|\mathbf{x}) = \frac{\big(\prod_{i=1}^{n} P(w_i|y = 1)\big)P(y = 1)}{\big(\prod_{i=1}^{n} P(w_i|y = 1)\big)P(y = 1) + \big(\prod_{i=1}^{n} P(w_i|y = 0)\big)P(y = 0)}.$$

Notice in the equation above that the order of words in a review is disregarded, and they are treated simply as a set of unordered attributes. This is known as the *bag of words* model of documents.

Finally, since many probability values will be multiplied in the previous equation, to avoid rounding errors, we will use the logaritham of the posterior probability:

$$\log(P(y = 1|\mathbf{x})) = \\ \log\Big(\frac{\big(\prod_{i=1}^{n} P(w_i|y = 1)\big)P(y = 1)}{\big(\prod_{i=1}^{n} P(w_i|y = 1)\big)P(y = 1) + \big(\prod_{i=1}^{n} P(w_i|y = 0)\big)P(y = 0)}\Big).$$

Objectives

You will need to estimate two sets of parameters from the training set to build your classifier. The first is the prior probabilities of the class $P(y = y_k)$. For each class $y_k \in \{0, 1\}$, this results to the number of reviews labeled with the particular class from the total number of reviews in the training set.

The second set of parameters to estimate are the likelihood probabilities $P(w = w_i|y = y_k)$, for each word w_i in the the vocabulary V (which consists of a set of unique words from all reviews in the training set). The likelihood probability of the word w_i given class y_k is the number of times w_i occurs in a review with label y_k from the total number of reviews in the training set labeled with y_k.

What to so

- Write a Python code for the described classifier without using the **scikit-learn library**. The scikit-learn library is in whole off limits.
- Try out your classifier on the test set and calculate the error rate that you observe.
- Create an evaluation (prediction) function that can be used on the command line to evaluate a short review that you can write yourself.
- When a review contains a word that is not in the dictionary, the posterior class probabilities are zero.

  To overcome the problem you can use a smoothing technique known as Laplace smoothing and calculate the likelihood probabilities as:

$$\hat{P}(w = w_i|y = y_k) = \frac{\#R\{w = w_i \wedge y = y_k\} + 1}{\#R\{y = y_k\} + |V|},$$

  where the $\#R\{\}$ operator denotes the number of elements in the training set of reviews R that satisfy the constraint in the brackets, and $|V|$ is the cardinality of the vocabulary.

What to submit

All documents should be compressed in a zip file.

- The code in a digital form. Do not submit the data sets.
- A small text file on how to run the code. A necessary condition for the assignment to be admissible is that the examiner can run the code (regardless of operating system). If the examiner is unable to run the code 0 points will be given for the entire assignment!

- Between 1000 - 5000 characters of report on the performance of your classifier, the error rates for the training and test data sets, and an explanation of that performance and how you can improve it.