Programming Homework, CSE 440
Spring 2019, Michigan State University,

Naive Bayes Classifier for Natural Language

Sari Saba-Sadiya
CSE 440, Spring 2019

Michigan State University
sadiyasa@msu.edu

ABSTRACT. Naive Bayes Classifiers are a powerful tools that leverage Bayesian probability to learn from labeled data and make inference. While they have been becoming less popular in the aftermath of the deep learning tsunami, they remain a favorite among many for their ease of use and their relatively transparent nature (in comparison to neural network black boxes).

KEYWORDS: Naive Bayes Classifiers, Natural Language Processing.

Prelude

This document will be censored for academic honesty concerns, an uncensored version will be made available upon request after all projects are submitted. A censored text looks like this: [redacted].

1. Introduction

In the early 2000's Dr Lillian Lee and her students at Cornell collected a database of movie reviews. This database was used to benchmark many algorithms for various natural language processing tasks. Most prominently, Dr Lee herself published a paper[1] comparing multiple algorithms for sentiment analysis from

text. One of the algorithms was a simple Naive Bayes classier that learned how to infer if a review in natural language was positive or negative with a 78.7% accuracy. In this assignment you will reproduce her results.

1.1. Grading

Before we begin, I would like to offer a few words of encouragement:

- As long as you use Bayesian inference and get an at least 75% accuracy and runs in less than 3 minutes you will get full credit!
- No monkey business! Do not use `bayespy` or any of the shelve Bayesian inference package. Also, sending your code to someone else / asking for someone else for his code will get you an academic suspension.
- My solution had less than 100 lines of code. It is not a complicated coding assignment. Just make sure to document your code.

1.2. Packages and Setup

There are two phases for this project, the training and the testing. The training phase involves reading the reviews in the `/posTrain/` and `/negTrain/` folders (which contain positive and negative reviews respectively) and calculating the conditional probabilities. The testing phase involves using these probabilities to predict the classes of the documents in `/posTest/` and `/negTest/` (again, they contain positive and negative reviews respectively) via Bayesian inference.

Considering the above, we need to begin by reading the content of all files in the directory. This is done in python in the following way:

```
for file in directory:
    with  open(file,'r') as f:
        rawText = f.read().encode('ISO-8859-1')
```

Additionally, before using the text you need to remove punctuation, and capitalization. There are multiple ways to accomplish this, I have tested the following two:

```
## with nltk
import nltk

tokenizer = nltk.tokenize.RegexpTokenizer(\
                    r'[a-zA-Z]+[\']*[a-zA-Z]+|[;!?$]')
```

At which case you can get filtered text like this:

```
rawText = f.read()
cleanText = tokenizer.tokenize(rawText)
```

Alternatively you can use the `string` module:

```
import string
tokens =[token.translate(None,string.punctuation).lower()\
            for token in tokens]
```

Go ahead and try on different strings. If you fail to get the accuracy perhaps trying to also remove numbers will further improve your classifier.

2. Training

Our final objective is to predict the class of the review $c \in \{pos, neg\}$ base on the content which is a list of words $W = w_1, w_2, \ldots, w_n$. In other words we are trying to determine which of the two probabilities $P(c = pos|W)$ and $P(c = neg|W)$ is larger. For simplicity we will assume that the words are independent of each other, as shown in class this enables us to perform key simplifications:

$$P(c = pos|W) = P(c = pos|w_1, w_2, \ldots, w_n) = \Pi_{i=1}^{n} P(c = pos|w_i)$$

This presents a practical challenge; Because we are dealing with small numbers we run the risk of the computer approximating our probabilities to zeros. To avoid this, computer scientists often use the "log-space" (remember

$log(a \cdot b) = log(a) + log(b)$) substituting mutiplications with additions:

$$P(c = pos|W) \geq P(c = neg|W) \Leftrightarrow$$
$$log(P(c = pos|W)) \geq log(P(c = neg|W)) \Leftrightarrow$$
$$log(\Pi_{i=1}^{n} P(c = pos|w_i)) \geq log(\Pi_{i=1}^{n} P(c = neg|w_i)) \Leftrightarrow$$
$$\sum_{i=1}^{n} log(P(c = pos|w_i)) \geq \sum_{i=1}^{n} log(P(c = neg|w_i))$$

Following Bayesian rules, the probabilities can be rewritten:

$$P(c = pos|w_i) = P(\blacksquare|\blacksquare\blacksquare\blacksquare) \cdot \frac{P(w_i)}{P(c = pos)}$$

Make sure you understand the previous equation! Note that in our case $P(c = pos) \approx P(c = neg)$ and hence, because we are comparing $P(c = pos|W)$ and $P(c = neg|W)$ we can disregard $P(c = pos)$ and $P(c = neg)$ in our calculations.

Considering all of the above, all we need is to calculate $P(\blacksquare|\blacksquare\blacksquare\blacksquare)$. This can be done by counting 3 measures:

- The number of appearances on of the word w in positive documents.
- The number of total word appearances in positive documents.

2.1. Smoothing

What if a certain word w only appears in negative reviews and not positive ones? You are encouraged to try to simply assign a zero probability (simply skip the word). As you might discover, the classifier performance is usually extremely low if this system is used. To address this issue, researchers often use Additive / Laplace smoothing (look at the section about applications). The intuition behind Laplace smoothing is somewhat complicated. However, luckily, the application is straight forward. Specifically instead of $\frac{x_i}{N}$ (the two counts mentioned above) we calculate $\frac{x_i+\alpha}{N+\alpha \cdot d}$ where d is the number of unique words in both negative and positive documents (not number of occurrences, read the Wikipedia if it is still confusing). the parameter α is a called the pseudo-count, please use $\alpha = 0.1$.

Problem:Laplace_Smoothing

Calculate $P(\blacksquare|\blacksquare)$ with the Laplace smoothing and name it `conditionalProbPos` and `conditionalProbNeg` for the probability for the negative classes. Specifically, `conditionalProbPos` needs to be a a dictionary of words that gives the value of the conditional probability for every word.

3. Testing

Having calculated $P(\blacksquare|\blacksquare)$ with the Laplace smoothing we can now calculate

$$P(c = pos|W) = P(c = pos|w_1, \ldots, w_n) \propto \sum_{i=1}^{n} log(P(\blacksquare|\blacksquare))$$

This enables us to decide what is the category of the review based on which of the measures $P(c = pos|W)$ and $P(c = neg|W)$ is larger.

Problem:Laplace_Smoothing

Produce a dictionary "resultsP" for the files in `/posTest/` that gives you `True` if the file was correctly labeled and `false` otherwise.

4. Conclusion

Bayesian probability is a surprisingly powerful tool that achieves competitive results on real benchmarks. Dr Lee wrote her paper at 2002, while it might not be in the cutting edge of our field anymore, it remains a testament to your ability to read, understand, and apply research papers.

REFERENCES

[1] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of EMNLP*, pp. 79–86, 2002.