

Homework 6: Naive Bayes for Sentiment Analysis

Yannet Interian

Due Aug 17th 2014

1 Introduction

In this project we consider the problem of classifying a document by the “sentiment.” You are going to write a classifier that predicts whether a movie review is positive or negative using a dataset set of movie reviews. Download the data for the project at <http://www.cs.cornell.edu/people/Pabo/movie-review-data/>. Click on “polarity dataset v2.0”.

1.1 Naive Bayes for document classification

Given a document d and a class c , one approach to text classification is to assign to a given document the class

$$c^* = \operatorname{argmax}_c P(c|d)$$

We can substitute $P(c|d)$, using Bayes’ theorem with $P(c|d) = \frac{P(c)P(d|c)}{P(d)}$, getting the formula $c^* = \operatorname{argmax}_c \frac{P(c)P(d|c)}{P(d)}$. Note that for a fixed document d , $P(d)$ is a constant so the following formula is equivalent:

$$c^* = \operatorname{argmax}_c P(c)P(d|c)$$

We can represent a document d as a sequence of words $d = (w_1, w_2, \dots, w_m)$. To estimate $P(d|c)$, Naive Bayes assumes that each word is conditionally independent given the class.

$$c^* = \operatorname{argmax}_c P(c) \prod_{i=1}^m P(w_i|c)$$

If M is the number of unique words in d and $n_i(d)$ is the count of word w_i in d , we can write the following equivalent formula.

$$c^* = \operatorname{argmax}_c P(c) \prod_{i=1}^M P(w_i|c)^{n_i(d)} \quad (1)$$

1.2 Underflow Prevention: log space

Multiplying lots of probability can result in floating-point underflow. It is better to sum logs of probabilities instead of multiplying probabilities. Using $\log(xy) = \log(x) + \log(y)$, we can write (1) as:

$$c^* = \operatorname{argmax}_c \log(P(c)) + \sum_{i=1}^M n_i(d) \cdot \log(P(w_i|c)) \quad (2)$$

1.3 Parameter Estimation with add 1 smoothing

The maximum likelihood estimator for $P(w|c)$ is $\frac{\text{count}(w,c)}{\text{count}(c)} = \frac{\text{counts } w \text{ in class } c}{\text{counts of words in class } c}$. This estimation of $P(w|c)$ could be problematic since it would give us probability 0 for documents with unknown words. A common way of solving this problem is to use Laplace smoothing. Let V be the set of words in the training set, add a new element UNK (for unknown) to the set of words. Define

$$P(w|c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V| + 1}$$

In particular, any unknown word will have probability $\frac{1}{\text{count}(c) + |V| + 1}$. What is $P(c)$ for this dataset?

1.4 Evaluation

Use 3-fold cross-validation for evaluating your results. For each class, randomly divide your reviews into 3 subsets. Train and test your results three times. Each time, one of the three subsets is used as the “test set” and the other two subsets are put together to form a “training set.” Compute the average error across all three trials.

1.5 Evaluation Metric

$$\text{Accuracy of the testing set} = 100 \cdot \frac{\text{Number of test docs classified correctly}}{\text{Total number of test documents}}$$

2 Naive Bayes for Sentiment Analysis

Name your code *naive-bayes.py*. **Implement a Naive Bayes algorithm** for the sentiment analysis project in python using the following steps:

1. Randomly divide your data in 3 parts.
2. For each iteration i
 - (a) Use the i part for testing and the rest for training.
 - (b) Use the training set to estimate the parameters $P(w|c)$ and $P(c)$ as described in section 1.3.
 - (c) For every document in the testing set use equation (2) to compute $P(c|d)$ and predict the class c^* .
 - (d) Compute the accuracy of the testing set as described in section 1.5.
3. Compute the average accuracy as your performance metric.
4. Given a directory with text files “my_directory” (neg and pos are inside of my_directory). I should be able to run your code using the command line:

```
python naive-bayes.py -d my_directory
```

5. Print results for each iteration, output the key metrics as in the example below.

```
iteration 1:
num_pos_test_docs:333
num_pos_training_docs:667
num_pos_correct_docs:267
num_neg_test_docs:331
num_neg_training_docs:669
num_neg_correct_docs:261
accuracy:79%
iteration 2:
...
iteration 3:
...
ave_accuracy:80.3%
```