CS294-1 A1: Naive Bayesian Classifier

Di Wang and Liwen Sun

1 Introduction

We implement a naive bayesian classifier in the context of document sentiment analysis. Given a document collection of movie reviews, each of which is labeled as either “positive” or “negative”, our task is to train a naive bayesian classifier based on word occurrences that can determine whether a given document belongs to the positive or negative class.

Settings. Our codes were written in Scala and compiled under Simple Build Tool (SBT). The programs were run on Mac OS. We test the effectiveness of our implementation in various aspects. If not mentioned explicitly, we adopt the following default settings. We report macroaveraged F1 measures, which were further averaged by ten-fold cross validations. We consider both “Bernoulli” and “Multinomial” models. We use as features all the words preprocessed by stemming and stop-word elimination. Later discussions may unveil why certain default choices are made.

In the reminder of the report, our study of the naive bayesian classifier has the following components:

- Model Selection
- Ten-fold Cross Validation
- Tokenizing
- Smoothing Factor
- Feature Selection

2 Model Selection

We implemented both Bernoulli and multinomial models for our task. Since the two models have different strengths and preferred situations, it is hard to say which one is better than the other. The Bernoulli model works best on shorter documents with fewer features,
and the multinomial model is more scalable with the size of documents and the number of features. In the following sections, we show the effectiveness comparisons of the two models in different settings.

3 Ten-fold Cross Validation

To assess the effectiveness of our classification algorithm, we partition the labeled data into two parts, one used as training set and the other as testing set. During the large variability of this one-round testing, we perform a ten-fold cross validation. In detail, we randomly partition the data into ten parts, and iteratively use one of them as testing set and the rest nine parts as training set.

To illustrate, we show an example of the ten-round results in Table 1. We can see that there are indeed some variation of the performance but the variance of ten rounds is in an acceptable range (i.e., $10^{-4}$). In other parts of the report, we only show the average of the ten-fold cross validation.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>mean</th>
<th>var</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-</td>
<td>0.83</td>
<td>0.82</td>
<td>0.77</td>
<td>0.79</td>
<td>0.84</td>
<td>0.78</td>
<td>0.79</td>
<td>0.84</td>
<td>0.82</td>
<td>0.81</td>
<td>0.81</td>
<td>$6.8 \times 10^{-4}$</td>
</tr>
<tr>
<td>Bi-</td>
<td>0.85</td>
<td>0.85</td>
<td>0.80</td>
<td>0.80</td>
<td>0.85</td>
<td>0.81</td>
<td>0.82</td>
<td>0.83</td>
<td>0.85</td>
<td>0.83</td>
<td>4.8 $\times 10^{-4}$</td>
<td></td>
</tr>
</tbody>
</table>

4 Tokenizing

Tokenizing is a preprocessing step, which extracts the words, or tokens, from the document texts, and then pass the tokens as features for later use. While it is an important step to identify the meaningful words that really are an indicator for a certain class, such as “good” and “glad” in the positive class, removing the noisy tokens is also essential, i.e., those have little to do with the classes, such as stop words (e.g., “the” and “a”).

We use two extra steps after extracting the words, namely, stemming and stop-word elimination. In Figure 2, we show the effectiveness ($F1$ score) of our algorithm with four alternatives, where Stem stands for ply stemming, Stop stands for only eliminating stop words, Stop+Stem indicates using both, and None means using neither. We make several interesting observations. First, the Bernoulli model is more sensitive to the token noises, i.e., Bernoulli model receives a better performance gain by removing stop words and stemming than multinominal model. This is because the Bernoulli model only considers the presence of words, and stemming, for example, can turn the score of an important signal
(e.g., convert “loving” to “love”) from zero to one, which may lead to a much better result. The multinomial model, on the other hand, takes into account the frequency of words, and refinement on the tokenizing step may only change the frequency of certain words, or the score of features, marginally. Second, the Bernoulli model, after removing the noises, can do much better a job than the multinomial model. This is probably due to the fact that most documents in our data are of a small length.

5 Smoothing Factor

One inherent issue of the naive bayesian classifier is that if a token does not appear in a class, the prior probability of this token in the class is set to be zero, which can be inaccurate and makes the calculation problematic. Fortunately this can be easily alleviated by introducing a smoothing factor $\alpha$. However, it is not trivial to set an appropriate value for $\alpha$, although $\alpha = 1$ was a commonly used one. In Figure 2, we show the effectiveness of Bernoulli and multinomial models by varying $\alpha$. Both models reach the performance peak at around the point $\alpha = 2$ to 3. This is a fortunate result, as we can easily make choices of $\alpha$ from the curve. We also observe that Bernoulli consistently outperforms multinomial with different $\alpha$.

6 Feature Selection

We examine the effect of selecting features. We use the frequency-based feature selection, in which frequently occurred features in a class have higher scores. Note that for multinomial we count the frequency as the number of occurrences in the class, while for Bernoulli we
count the number of documents that contain the feature. In Figure 3 we pick the top-$k$ frequent features with a varying $k$. We find that the performance is not maximized with a very small $k$, e.g., $k < 1000$. It becomes stable after $k$ reaches a considerable size.

**Term Weights.** In the Naive Bayes model, the $\hat{p}(t|c)$ is proportional to the number of times term $t$ appears in all reviews of class $c$, so the terms with the highest weights are those that appear most often in the training documents with the specific label. We count the terms in positive and negative reviews respectively, and it is not surprising to find out that the terms with highest weights are words such as "'a'"", "'the'"", "'and'"", "'of'"", "'to'"", "'in'" etc. Those terms are not very indicative in terms of classification. With stemming and stop word removing, we see some meaningful words in the top weighted terms, including "'good'"", "'like'"", "'character'"", "'really'"", "'best'"", "'little'" etc., but we see little difference between the set of top weighted terms for the two different classes. One way to improve the situation is to use feature selection based on mutual information measure or $\chi^2$-squared measure, which will pick out the features that are most indicative for each class.

![Figure 2: Effect of $\alpha$](image-url)
Figure 3: Effect of $\alpha$