250B Problem Set 2

William Fedus

January 2015

1 Text Classification via Multinomial Naive Bayes

Using the *20 Newsgroups* data set we train a multinomial Naive Bayes classifier to predict the class designation j of a particular document composed of n unique features $x_1, \ldots, x_n$ where $n = |V|$ which is the size of the Vocabulary V.

1.1 Probabilistic Approach

At an abstract level, to classify a document, we use Bayes rule to calculate the probability that a particular document of n words belongs to class j, we evaluate

$$P(j|x_1, \ldots, x_n) = \frac{P(j)P(x_1, \ldots, x_n|j)}{P(x_1, \ldots, x_n)} \tag{1}$$

Equation 1 is difficult to evaluate, but we can greatly simplify the conditional probability via the naive assumption that the probability of a given feature x_i occurring is conditionally independent on all prior and subsequent words given the document class j, or restated mathematically,

$$P(x_i|j, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|j) \tag{2}$$

this assumption simplifies Equation 1 to the following form

$$P(j|x_1, \ldots, x_n) = \frac{P(j)\prod_{i=1}^{n} P(x_i|j)}{P(x_1, \ldots, x_n)} \tag{3}$$

but since $P(x_1, \ldots, x_n)$ is a constant given a particular inputted document, we can classify it to class j by choosing the argument j which maximizes the numerator of Equation 3, this corresponds to the *maximum a posterior* rule,

$$\hat{j} = \text{argmax}_j \left[P(j) \prod_{i=1}^{n} P(x_i|j) \right] \tag{4}$$

where $P(j)$ is the fraction of documents belonging to class j and $P(x_i|j)$ is the probability of feature x_i given class j. However, the multiplication of many probabilities can lead to underflow issues in the computation.

1.2 Multinomial Naive Bayes Model

Now, in the case of using a Multinomial model, each feature x_i now corresponds to the frequency that word i was observed in a particular document. Our probability distribution for a particular class j over documents now is of the form in Equation 5

$$p(x_i, \ldots, x_n|j) = \frac{m!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} P(x_i|j)^{x_i} \tag{5}$$

where $m = \sum_{i=1}^{n} x_i$ is the length of the particular document, $P(x_i|j)$ is the probability of word i occurring in class j and x_i is the frequency that word i appears in that document. Now with a probability distribution defined over each document, we can use Bayes rule analagous to the prior section. However, the multiplication of many probabilities may lead to underflow issues in large documents so in prevent this issue, we instead seek the argmax over log() of the posterior. This yields the classification rule given by Equation 6

$$\hat{j} = \text{argmax}_j \left[\log(P(j|x_i, \ldots, x_n)\right] \propto \log\left(P(j) \prod_{i=1}^{n} P(x_i|j)^{x_i}\right) \tag{6}$$

then using rules of logarithms, we may simplify the following classifier into a *linear* classifier in log-space.

$$\hat{j} = \text{argmax}_j \left[\log P(j) + \sum_{i=1}^{n} x_i \log P(x_i|j)\right] \tag{7}$$

This however has the issue that if a particular word i is found in a document, that has never appeared in a document of class j before, the entire probability will be evaluated to 0. To adjust for this, we use a smoothing parameter. In particular, our estimate for $\hat{P}(x_i|j)$ will be the following

$$\hat{P}(x_i|j) = \frac{\sum_{x \in T} x_i + \alpha}{\sum_i \sum_{x \in T} x_i + \alpha n} \tag{8}$$

where α is the smoothing parameter, $\sum_{x \in T} x_i$ is the number of times word i is found across training set documents of class j and $\sum_i \sum_{x \in T} x_i$ is the total number of features in class j. Note that if we were to set $\alpha = 0$, then we would simply recover the frequency estimate for $\hat{P}(x_i|j)$.

1.3 Classification Accuracy of Model

We test the performance of this model on the *20 Newsgroups* dataset. In this instance, we feed in the data via sklearn datasets module, which has a pre-processed version of the data. In order to prevent overly optimisitic prediction accuracy, we strip out the headers and footers for the datasets. Additionally, we manually set the vocabulary with $|V| = 61188$ provided with the assignment so

that the classifier does not automatically generate a vocabulary from the provided corpus. To also ensure that certain documents do not get misclassified, we employ Laplace smoothing with $\alpha = 1.0$.

We evaluate the Multinomial Naive Bayes model both on classification accuracy as well as F1 score, which is a weighted average of both precision and recall. After training on the Train partition, we achieve the following results on the Test partition in Table 1.

Measure	Performance
Accuracy	0.725
F1 Score	0.706

Table 1: Base classification performance of the Multinomial Naive Bayes model.

1.4 Improvement of Model

Our performance is quite good, but we can seek to improve via new tests. In our earlier example, we simply considered the count frequency of words to determine the probability distribution over the vocabulary V that models a specific class. However, a query of the words with the highest $\hat{P}(x_i|j)$ are primarily low-information words, typically termed 'stop-words', including words of the set {the, of, is, it, in, and, that, this}. To improve performance immediately, we strip out these stop-words from our vocabulary to achieve the performance on Test data in Table 2.

Measure	Performance
Accuracy	0.751
F1 Score	0.734

Table 2: Classification performance of the Multinomial Naive Bayes model after stop-words are removed from the vocabulary.

By removing these words, we find a several percentage point improvement over the base case and if we query the most highest $\hat{P}(x_i|j)$ words, we find the words to be intuitively far more indicative of a document class. For instance, for 'talk.religionn.misc' we find the top-10 words with the highest $\hat{P}(x_i|j)$: {bible christian know think just article writes jesus people god}.

As an additional way to improve performance, we can use term frequency-inverse document frequency (TF-IDF) which is a numerical statistic that attempts to quantify the importance of a particular word to a document class. When we apply this, we find the results in Table 3.

We have increased the performance of the classifier nearly 7% from the baseline through a couple of simple data transformations. This analysis has also been

Measure	Performance
Accuracy	0.777
F1 Score	0.768

Table 3: Classification performance of the Multinomial Naive Bayes model after stop-words are removed from the vocabulary and we use term frequency-inverse document frequency (TF-IDF).

careful to prevent leakage of information from headers and footers, so that the model is reflective of true text classification.

2 Classification with an abstain option

We wish to create a classifier $h(x) : \mathcal{X} \to \{0, 1, \text{abstain}\}$ by minimizing the risk for a particular x value. Specifically, our classifier will be the following,

$$h(x) = \text{argmin}\left[c_{01}\eta(x), c_{10}(1 - \eta(x)), \theta\right] \tag{9}$$

where $c_{01} = 1$ is the cost of predicting 0 when the actual value is 1, $c_{10} = 1$ is the cost of predicting 0 when the actual value is 1, and $\theta \in [0, \frac{1}{2}]$ is the cost associated with abstaining. Therefore, given these values, for any value of x, we choose the argument that yields the minimal cost, or as shown in Equation 10

$$h(x) = \text{argmin}\left[\eta(x), (1 - \eta(x)), \theta\right] \tag{10}$$