Improving the Performance of Naive Bayes for Text Classification

Yirong Shen and Jing Jiang

CS224N Spring 2003

Abstract

We seek to improve the performance of the naive Bayes classifier by adding a low-dimension "discriminative model" component to the "generative model" naive Bayes. Focusing on the problem of text classification, we show that our method results in reduced test error in experiments as well as improved class posteriors that leads to better accuracy/coverage curves.

1 Introduction

The naive Bayes classifier is a simple generative classification algorithm that has reasonably good performance. However, there are often situations where discriminative algorithms, such as logistic regression, achieves lower generalization error [2] [3]. In addition, it has been observed that when naive Bayes is asked to make predicitions, it almost always gives class posteriors very close to 0 or 1. This overconfidence of naive Bayes makes it unsuitable for applications that require smoother class posteriors.

In this work, we make a simple modification to naive Bayes to improve its performance and make it output more accurate class posterior probabilities. Naive Bayes is trained as a "generative" model that fits the distribution of the data instances given the class label; our method adds a small number of parameters that are trained like a "discriminative model," and fits the distribution of the class label given the instance. Focusing on the problem of text classification, we demonstrate that adding this discriminative component

to naive Bayes significantly lowers the test error and leads to much improved accuracy/coverage curves.

Our method is also straightforward to apply to non-text applications of naive Bayes, and to settings where other generative models are used for classification.

2 The Algorithms

In this section, we briefly review the multinomial naive Bayes classifier applied to text categorization [1], and then describe our method for modifying it to obtain better posteriors.

2.1 Naive Bayes Classification

Let $\mathcal{Y} = \{0, 1, \ldots, |\mathcal{Y}|\}$ be the set of possible labels for documents, and $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ be a dictionary of words. A document of N words is represented by the vector $X = (X_1, X_2, \ldots, X_N)$ of length N. The i-th word in the document is $X_i \in \mathcal{W}$. Note that N can vary for different documents. The multinomial naive Bayes model assumes that the label Y is chosen from some prior distribution $p(Y = \cdot)$, the length N is drawn from some distribution $p(N = \cdot)$ independently of the label, and each word X_i is drawn independently from some distribution $p(W = \cdot|Y)$ over the dictionary. Thus, we have

$$p(X = x, Y = y) = p(Y = y)p(N = n)\prod_{i=1}^{n} p(W = x_i|Y = y) \qquad (1)$$

Since the length n of the document does not depend on the label and therefore does not play a significant role in our subsequent derivations, we will drop the p(N=n) from our equations.

Now let there be a training set $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{m}$ of m examples. The generative naive Bayes classifier uses S to calculate estimates $\hat{p}(X|Y)$ and $\hat{p}(Y)$ of the probabilities $p(X|Y)$ and $p(Y)$. The estimated probabilities are typically calculated in a straightforward way from counts from training data, with Laplace or some other smoothing. To classify a new document, it is straightforward to see using Bayes rule that the estimated class posterior

probabilities are

$$\hat{p}(Y = y_0|X = x) = \frac{\hat{p}(X = x|Y = y_0)\hat{p}(Y = y_0)}{\sum_{y=1}^{|\mathcal{Y}|} \hat{p}(X = x|Y = y)p(Y = y)} \tag{2}$$

where

$$\hat{p}(X = x|Y = y) = \prod_{i=1}^{n} \hat{p}(W = x_i|Y = y) \tag{3}$$

The predicted class for the document is then just $\arg\max_{y \in \mathcal{Y}} \hat{p}(Y = y|X = x)$.

2.2 Modifications to Naive Bayes

We assume that every input document X can be naturally divided into C components $X^1, X^2, \ldots, X^C$. For example, if the documents are USENET postings, then they can be considered to consist of a subject line and a body component. Note that C could be just 1. The components are of variable lengths $N_1, N_2, \ldots, N_C$. We replace the document probability in equation 3 with

$$\prod_{c=1}^{C} \hat{p}(X^c = x^c|Y = y)^{\beta_c/N_c} = \prod_{c=1}^{C} (\prod_{i=1}^{N_c} \hat{p}(W = x_i^c|Y = y))^{\beta_c/N_c} \tag{4}$$

Here, $\beta = (\beta_1, \beta_2, \ldots, \beta_C)$ is a constant vector that will be determined later. With this modification, our "class posteriors", which we denote by $\hat{p}_\beta$ to make explicit the dependence on β, become

$$\hat{p}_\beta(Y = y_0|X = x) = \frac{\prod_{c=1}^{C} \hat{p}(X^c = x^c|Y = y_0)^{\beta_c/N_c} \hat{p}(Y = y_0)}{\sum_{y=1}^{|\mathcal{Y}|} \prod_{c=1}^{C} \hat{p}(X^c = x^c|Y = y)^{\beta_c/N_c} \hat{p}(Y = y)} \tag{5}$$

There are several reasons for breaking the document into components and adding the exponent β_c/N_c to the conditional component probabilities. One justification is that some parts of the document may have stronger dependence on the label than other parts. Hence it is reasonable to add some type of "weighting" factor to take this into account. Another reason is that the independence assumption for naive Bayes is too strong. With a document of length N, the classifier "assumes" that it has N completely independent pieces of evidence supporting its conclusion about the document's

label. Putting N_c in the denominator of the exponent as a normalization factor can be viewed as a way of counter-acting the overly strong independence assumptions.

It remains to specify how β is chosen. Given a training set of m labelled documents $\{x^{(i)}, y^{(i)}\}_{i=1}^m$ (which may be the same training set used to estimate the class priors and word probabilities), a very natural criterion is to choose β to maximize the loglikelihood of the training data's labels:

$$\beta = \arg\max_{\hat{\beta}} \sum_{i=1}^{m} \log \hat{p}_{\hat{\beta},i}(Y = y^{(i)}|X = x^{(i)}) \tag{6}$$

where $\hat{p}_{\hat{\beta},i}(Y = y^{(i)}|X = x^{(i)})$ is as given in equation 5, except that the class priors and word generation probabilities were estimated with the i-th sample of the training data held out.

Now specializing to the two-class case (i.e. $\mathcal{Y} = \{0, 1\}$), we can obtain, after some arithmetic manipulations, the following expression for $\hat{p}_\beta(Y = 1|X = x)$:

$$\hat{p}_\beta(Y = 0|X = x) = \frac{1}{1 + \exp(a + \beta_1 b_1 + \ldots + \beta_C b_C)} \tag{7}$$

where $a = \log \frac{\hat{p}(Y=1)}{\hat{p}(Y=0)}$ and $b_c = (\log \frac{\hat{p}(X^c=x^c|y=1)}{\hat{p}(X^c=x^c|y=0)})/N_c$. With this formulation for $\hat{p}_\beta(Y = 1|X = x)$, we see the maximization problem in 8 is very similar to the form of the objective of logistic regression, the only difference being that in this case a is calculated from the estimated class priors. To make the parallels to logistic regression complete, we define $b_0 = 1$, $\tilde{\beta} = (\beta_0, \beta_1, \ldots, \beta_C)$, and

$$\tilde{p}_{\tilde{\beta}}(Y = 0|X = x) = \frac{1}{1 + \exp(\tilde{\beta}^T b)}$$

where $\tilde{\beta}$ is chosen as follows:

$$\tilde{\beta} = \arg\max_{\tilde{\beta}'} \sum_{i=1}^{m} \log \tilde{p}_{\tilde{\beta},i}(Y = y^{(i)}|X = x^{(i)}) \tag{8}$$

The predicted label for a new document under this scheme is just

$$\arg\max_{y \in \{0,1\}} \tilde{p}_{\tilde{\beta}}(Y = y|X = x)$$

We call this scheme the *normalized hybrid* algorithm. For the sake of comparison, we will also consider a related scheme in which the exponents for

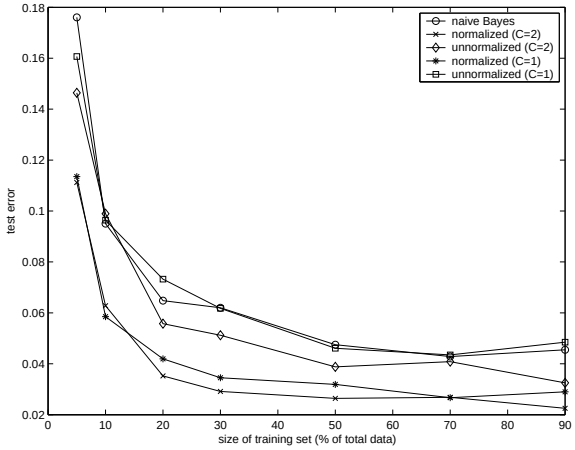

Figure 1: Test error vs training size on the newsgroups rec.sport.baseball and rec.sport.hockey

$\hat{p}(X^c = x^c|Y = y)$ in equation 4 is not normalized by N_c. In other words, we replace β_c/N_c by just β_c. We refer to this scheme as the *unnormalized hybrid* algorithm.

3 Experiments and Discussion

We now describe the results of two sets of experiments testing the effectiveness of our methods. The first directly measures the test error as a function of the training set size. The second examines accuracy/coverage curves, which shows how good an algorithm is at picking the documents on which it is most likely to make correct predictions. All experiments were run using varies pairs of newsgroups from the 20newsgroups data set of USENET news postings. When parsing this data, we skip everything in the USENET headers except the subject line; numbers and email addresses are replaced by special tokens NUMBER and EMAILADDR; tokens are formed after stemming using the Porter Stemmer on contiguous alphabetic characters.

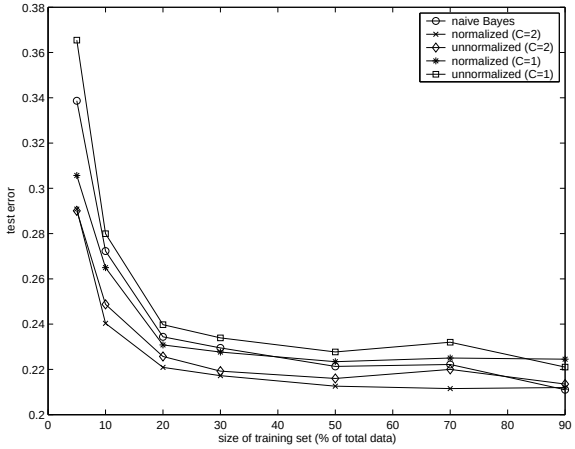

Figure 2: Test error vs training size on the newsgroups alt.atheism and talk.religion.misc

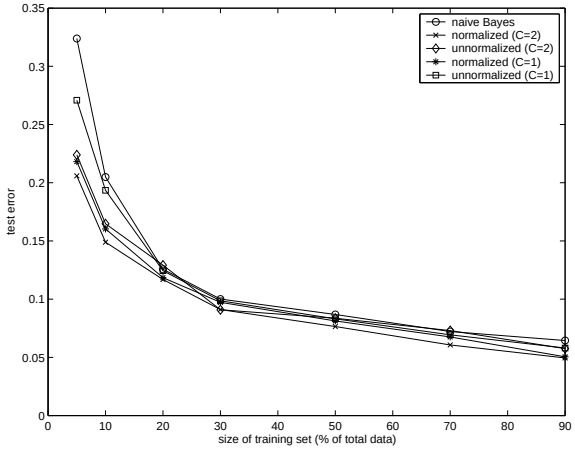

Figure 3: Test error vs training size on the newsgroups comp.sys.ibm.pc.hardware and comp.sys.mac.hardware

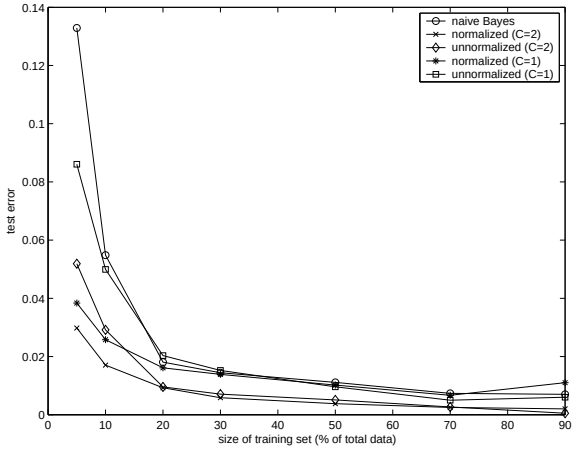

Figure 4: Test error vs training size on the newsgroups comp.graphics and talk.politics.mideast

3.1 Test Error vs Training Set Size

In this set of experiments, we compare the performance of the basic naive Bayes algorithm against both the normalized and unnormalized hybrid algorithms. For each of the hybrid algorithms, we measure performance in two cases. The first case is when each document is broken up into two components ($C = 2$), the subject line and the body. The second case is when each document is treated as just one component ($C = 1$). Each data point in this experiment is obtained by averaging the test error of 10 runs on random train-test splits.

Figures 1-4 plots the average test error of the various algorithms versus training set size for several pairs of newsgroups. We find that in every experiment, for every training size, the normalized hybrid algorithm with $C = 2$ has test error that is either the lowest or very near the lowest among all the algorithms. In particular, it almost always outperforms the basic naive Bayes algorithm. The difference in performance is especially dramatic for small training sets.

We also observe from the experiments that in general, the hybrid algorithms with $C = 2$ tends to outperform the corresponding algorithms with $C = 1$. This indicates that breaking the documents into components that are "weighted" differently does indeed contribute to improved performance.

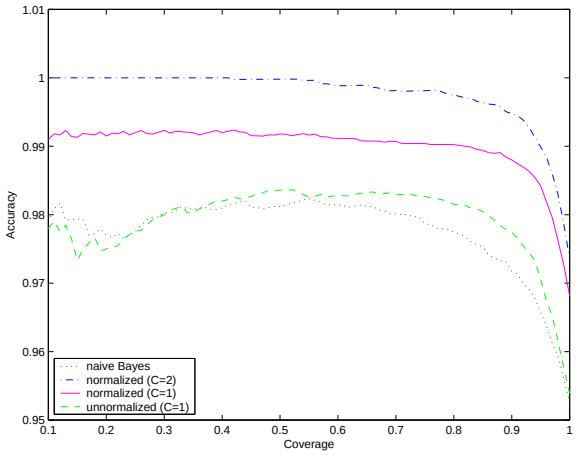

Figure 5: accuracy vs coverage on the newsgroups rec.sport.baseball and rec.sport.hockey

Similarly, we see that in general the normalized version of the hybrid algorithm tend to outperform the unnormalized version for the same value of C. This is evidence that normalization by component length also contribute to improved performance.

3.2 Accuracy vs Coverage

In the second set of experiments, we examine the accuracy/coverage curves of naive Bayes along with the normalized hybrid algorithm for $C = 1, 2$ as well as the unnormalized hybrid algorithm with $C = 1$. An accuracy/coverage curve shows the accuracy of a classifier if it is asked only to provide $x\%$ coverage – that is, if it is asked only to label the $x\%$ of the test data on which it is most confident. Accuracy/coverage curves towards the upper-right of the graph mean high accuracy even when the coverage is high, and therefore good performance. Accuracy value at coverage 100% is just the normal misclassification error.

In settings where both human and computer label documents, accuracy/coverage curves play a central role in determining how much data has to be labeled by humans. They are also indicative of the quality of a classifier's class posteriors, because a classifier with better class posteriors would be able to better judge exactly which $x\%$ of the test data it should be most

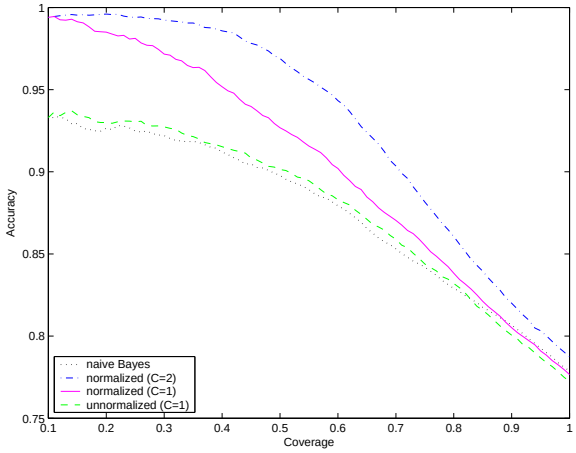

Figure 6: accuracy vs coverage on the newsgroups alt.atheism and talk.religion.misc

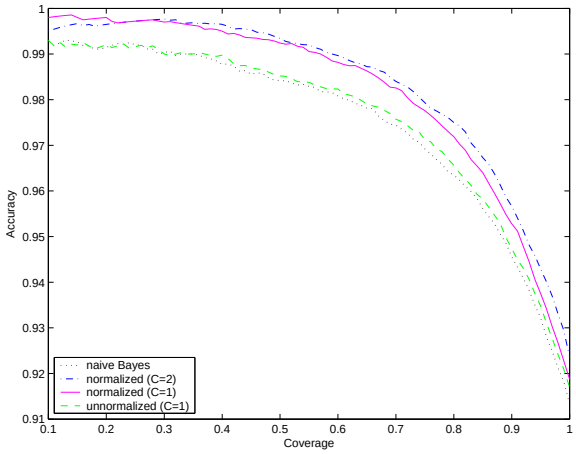

Figure 7: accuracy vs coverage on the newsgroups comp.sys.ibm.pc.hardware and comp.sys.mac.hardware

confident on, and hence achieve higher accuracy when it chooses to label that $x\%$ of the data.

Figures 5-7 shows 3 typical accuracy/coverage curves on the 20newsgroups data. Each curve is obtained by averaging the results of 10 runs on random 50%/50% train/test splits. We see that the normalized hybrid algorithm with C=2 is doing significantly better than naive Bayes, and has accuracy/coverage curves that is higher almost everywhere. For example, in figure 5, the normalized hybrid algorithm with $C = 2$ has a coverage of over 90% at 99% accuracy, while naive Bayes' coverage is 0 for the same accuracy. We also see that the unnormalized hybrid algorithm with $C = 1$ has accuracy/coverage curves that are about the same as naive Bayes, while the normalized hybrid algorithm with $C = 1$ has significantly better accuracy/coverage curves even in the cases when the test errors were comparable (figures 6 and 7). This again reinforces the importance of mitigating the overly strong independence assumptions of naive Bayes through normalization.

4 Final Remarks

The proposed normalized hybrid algorithm is simple and yet have significant performance gains over the naive Bayes classifier. Within the same framework, it is also natural to envision having more complex terms in the exponent than β_c/N_c such as $\beta_c/\sqrt{N_c} + \gamma_c$, $\beta_c/\log(N_c)$, etc.

References

[1] Andrew McCallum and Kamal Nigam. A comparison of event models for Naive Bayes text classification. AAAI-98 Workshop on Learning for Text Categorization, 1998.

[2] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. NIPS 14, 2002.

[3] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. IJCAI-99 Workshop on Machine Learning for Information Filtering, 1999.

[4] David Lewis and William Gale. A sequential algorithm for training text classifiers. Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, 1994.