Improving Multi-class Text Classification with Naive Bayes

by

Jason D. M. Rennie

B.S. Computer Science
Carnegie Mellon University, 1999

Submitted to the Department of Eleectrical Engineering and Computer Science in
Partial Fufillment of the Requirements for the Degree of

Master of Science in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
September 2001

Signature of Author . . . .
Department of Electrical Engineering and Computer Science
September 10, 2001

Certified by: . . . .
Tommi Jaakkola
Assistant Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: . . . .
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Improving Multi-class Text Classification with Naive Bayes

by

Jason D. M. Rennie

Submitted to the Department of Electrical Engineering and Computer Science
on September 10, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

There are numerous text documents available in electronic form. More and more are becoming available every day. Such documents represent a massive amount of information that is easily accessible. Seeking value in this huge collection requires organization; much of the work of organizing documents can be automated through text classification. The accuracy and our understanding of such systems greatly influences their usefulness. In this paper, we seek 1) to advance the understanding of commonly used text classification techniques, and 2) through that understanding, improve the tools that are available for text classification. We begin by clarifying the assumptions made in the derivation of Naive Bayes, noting basic properties and proposing ways for its extension and improvement. Next, we investigate the quality of Naive Bayes parameter estimates and their impact on classification. Our analysis leads to a theorem which gives an explanation for the improvements that can be found in multiclass classification with Naive Bayes using Error-Correcting Output Codes. We use experimental evidence on two commonly-used data sets to exhibit an application of the theorem. Finally, we show fundamental flaws in a commonly-used feature selection algorithm and develop a statistics-based framework for text feature selection. Greater understanding of Naive Bayes and the properties of text allows us to make better use of it in text classification.

Thesis Supervisor: Tommi Jaakkola
Title: Assistant Professor of Electrical Engineering and Computer Science

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There are numerous text documents available in electronic form. More are becoming available constantly. The Web itself contains over a billion documents. Millions of people send e-mail every day. Academic publications and journals are becoming available in electronic form. These collections and many others represent a massive amount of information that is easily accessible. However, seeking value in this huge collection requires organization. Many web sites offer a hierarchically-organized view of the Web. E-mail clients offer a system for filtering e-mail. Academic communities often have a Web site that allows searching on papers and shows an organization of papers. However, organizing documents by hand or creating rules for filtering is painstaking and labor-intensive. This can be greatly aided by automated classifier systems. The accuracy and our understanding of such systems greatly influences their usefulness. We aim 1) to advance the understanding of commonly used text classification techniques, and 2) through that understanding, to improve upon the tools that are available for text classification.

Naive Bayes is the de-facto standard text classifier. It is commonly used in practice and is a focus of research in text classification. Chakrabarti *et al.* use Naive Bayes for organizing documents into a hierarchy for better navigation and understanding of what a text corpus has to offer [1997]. Frietag and McCallum use a Naive Bayes-like model to estimate the word distribution of each node of an HMM to extract information from documents [1999]. Dumais *et al.* use Naive Bayes and other text classifiers to automate the process of text classification [1998]. That Naive Bayes is so commonly used is an important reason to gain a better understanding of it. Naive Bayes is a tool that works well in particular cases, but it is important to be able to identify when it is effective and when other techniques are more appropriate. A thorough understanding of Naive Bayes also makes it easier to extend Naive Bayes and/or tune it to a particular application.

There has been much work on Naive Bayes and text classification. Lewis gives a review of the use of Naive Bayes in information retrieval [Lewis, 1998]. Unlike text classification, information retrieval practitioners usually assume independence between features and ignore word frequency and document-length information. The multinomial model used for text classification is different and must be treated as such. Domingos and Pazzani discuss conditions for when Naive Bayes is optimal for classification even when its probability assessments are incorrect [Domingos and Pazzani, 1996]. Domingos and Pazzani clarify this point and show simple cases of when Naive Bayes is optimal for classification. Analysis of Naive Bayes like the work of Domingos and Pazzani is important, but little such work exists. Berger and Ghani individually ran experiments using ECOC with Naive Bayes. Both found that they were able to improve performance over regular Naive Bayes [Berger, 1999; Ghani, 2000]. But, neither adequately explains why regular Naive Bayes performs poorly compared to ECOC. Yang and Pedersen conduct an empirical study of feature selection methods for text classification [Yang and Pedersen, 1997]. They give an evaluation of five different feature selection techniques and provide some analysis of their differences. But, there is still need for better understanding of what makes a good feature selection method. Yang and Pedersen say that common terms are informative for text classification, but there are certainly other factors at work.

The application of Naive Bayes to multiclass text classification is still not well understood. An important factor affecting the performance of Naive Bayes is the quality of the parameter estimates. Text is special since there is a large number of features (usually 10,000 or more) and many features that provide information for classification will occur only a handful of times. Also, poor estimates due to insufficient examples in one class can affect the classifier as a whole. We approach this problem by analyzing the bias and variance of Naive Bayes parameter estimates.

Naive Bayes is suited to perform multiclass text classification, but there is reason to believe that other schemes (such as ECOC and multiclass boosting) can yield improved performance using Naive Bayes as a component. Regular Naive Bayes can be more efficient than these schemes, so it is important to understand when they improve performance and when they merely add inefficient baggage to the multiclass system. We show how ECOC can yield improved performance over regular Naive Bayes and give experimental evidence to back our claims.

The multitude of words that can be found in English (and other languages) often drives practitioners to reduce their number through feature selection. Feature selection can also improve generalization error by eliminating features with poor parameter estimates. But, the interaction between feature selection algorithms and Naive Bayes is not well understood. Also, commonly used algorithms have properties that are not appropriate for multiclass text classification. We point out these flaws and suggest a new framework for text feature selection.

Chapter 2

Naive Bayes

When someone says "Naive Bayes," it is not always clear what is meant. McCallum and Nigam clarify the picture by defining two different Naive Bayes "event models" and provide empirical evidence that the multinomial event model should be preferred for text classification [1998]. But, there are multiple methods for obtaining the parameter estimates. In the interest of clarity, we carefully step through the multinomial derivation of Naive Bayes and distinguish between variations within that model. We also present a fully Bayesian derivation of Naive Bayes, that, while not new, has yet to be advertised as an algorithm for text classification. Through a careful presentation, we hope to clarify the basis of Naive Bayes and to give insight into how it can be extended and improved.

To simplify our work, we assume that for each class, $c \in \{1, \dots, m\}$, there is an (unknown) parameter vector, θ^c, which generates documents independently. Some documents are observed as being part of a particular class (known as training documents and designated with D^c); others are test documents. This model is depicted in figure 2-1. We further assume that the generation model is a multinomial and ignore document length concerns.

2.1 ML Naive Bayes

One formulation of Naive Bayes is to choose the parameters that produce the largest likelihood for the training data. One then makes predictions using the estimated parameter vector, $\hat{\theta}^c$. This method has obvious flaws and includes strong assumptions about the generation of data. For example, any feature that does not occur in the training data for a class is assumed to not occur in any document generated by that class. However, this method, known as Maximum Likelihood (ML) can be effective in practice and is efficient to implement. It is used regularly in other domains. We call the multinomial version of this ML Naive Bayes.

The ML parameter for class c is

$$\hat{\theta}^c = \mathrm{argmax}_\theta \, p(D^c|\theta). \tag{2.1}$$

D^c is the training data for class c and θ^c is the class c parameter vector for a multi-

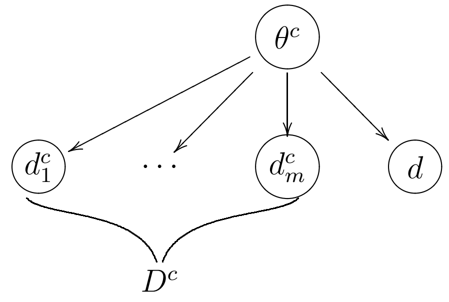


Figure 2-1: A graph of the independence relations between variables in classification. θ is the class multinomial parameter vector. D^c is the set of training documents for class c. d is test document to be classified.

nomial model. $p(D^c|\theta)$ is a multinomial likelihood,

$$p(D^c|\theta) = \frac{N^c!}{\prod_k N_k^c!} \prod_k \theta_k^{N_k^c}. \tag{2.2}$$

We use N_k^c to notate the number of times word w_k occurs in the class c training data ($N^c = \sum_k N_k^c$). θ_k is the k^{th} component of the multinomial parameter vector and is the probability that word w_k will appear as a single event of a multinomial trial. The ML estimate based on D^c (the $\hat{\theta}^c$ that maximizes $p(D^c|\hat{\theta}^c)$) is $\hat{\theta}_k^c = \frac{N_k^c}{N^c}\ \forall k$.

For ML Naive Bayes, we assume that our estimated parameter vector, $\hat{\theta}^c$, is the vector that generated D^c; we use $\hat{\theta}^c$ to assess whether a test document, d, was generated from class c. Since we infer a parameter vector, any prediction made about a test document, d, only implicitly depends on the training data, D^c; the setting of θ^c in figure 2-1 bottlenecks information that D^c may provide about d.

The Bayes optimal decision rule for classification is

$$\hat{H}(d) = \text{argmax}_c\, p(c|D, d) = \text{argmax}_c\, p(d|\hat{\theta}^c)p(c). \tag{2.3}$$

$D = \{D^1, \dots, D^m\}$ is the set of all training data. If our class prior, $p(c)$, is uniform, our classification rule simply chooses the class for which the test document is most likely,

$$\hat{H}(d) = \text{argmax}_c\, p(d|\hat{\theta}^c) = \text{argmax}_c \prod_k \left(\frac{N_k^c}{N^c}\right)^{f_k}. \tag{2.4}$$

f_k notates the number of times word w_k occurs in d. This decision rule is augmented for text classification because $p(d|\hat{\theta}^c) = 0$ when $f_k > 0$ and $N_k^c = 0$. To ensure that this cannot happen, the training data counts are supplanted with fictitious counts. The rationale for adding these counts varies. Using a fictitious count of a_k for word

w_k ($a = \sum_k a_k$), we arrive at the modified decision rule,

$$\hat{H}(d) = \text{argmax}_c \prod_k \left(\frac{N_k^c + a_k}{N^c + a} \right)^{f_k}. \tag{2.5}$$

Uniform fictitious counts ($a_i = a_j \; \forall i, j$) across all words are often used. A common choice is $a_k = 1$.

2.2 MAP Naive Bayes

ML Naive Bayes leaves something to be desired because it does not include the framework to explain the fictitious counts. As a result, we do not know what the fictitious counts represent. We would like to know what assumptions about parameter estimation underpins their inclusion in the decision rule. For this, we turn to a generalization of ML estimation, Maximum A Posteriori (MAP) estimation. MAP estimation produces the "fictitious counts" thorough a particular choice of parameter prior distribution. Except for the change in the way we estimate parameters, MAP Naive Bayes is identical to ML Naive Bayes. We still select a "best" parameter vector, $\hat{\theta}^c$ and use that vector for classification.

For MAP estimation, we estimate the parameter vector according to

$$\hat{\theta}^c = \text{argmax}_\theta \, p(\theta|D^c) = \text{argmax}_\theta \, p(D^c|\theta)p(\theta), \tag{2.6}$$

where $p(\theta)$ is the parameter prior term. MAP estimation is a generalization of ML estimation; ML is MAP with $p(\theta) = C$ (C is the appropriate constant). We choose the Dirichlet as the general form of the prior. It has hyper-parameters$\{\alpha_k\}$, $\alpha_k > 0$ ($\alpha = \sum_k \alpha_k$). The density of the Dirichlet is

$$p(\theta) = \text{Dir}(\theta|\{\alpha_k\}) = \frac{\Gamma(\alpha)}{\prod_k \Gamma(\alpha_k)} \prod_k \theta_k^{\alpha_k - 1}. \tag{2.7}$$

$\Gamma(x)$ is the Gamma function. It satisfies $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(1) = 1$. $\Gamma(n+1) = n!$ for $n \in \{0, 1, 2, \dots\}$. A valuable property of the Dirichlet is that it is the the conjugate prior to the multinomial distribution. This makes the posterior distribution Dirichlet,

$$p(\theta|D^c) = \frac{p(D^c|\theta)p(\theta)}{p(D^c)} = \text{Dir}(\theta|\{N_k^c + \alpha_k\}). \tag{2.8}$$

Setting $\theta_k = \frac{N_k^c + \alpha_k - 1}{N^c + \alpha - V}$ maximizes this expression (for $\alpha_k \geq 1$). V is the size of the vocabulary. Setting $\alpha_k = a_k + 1$ gives us the "fictitious counts" in equation 2.5 without any ad hoc reasoning. The MAP derivation makes clear that the fictitious counts represent a particular prior distribution on the parameter space. In particular, the common choice of $a_k = 1 \; \forall i$ represents a prior distribution in which more uniform parameters (e.g. $\theta_k = \frac{1}{V} \; \forall i$) are preferred.

2.3 Expected Naive Bayes

The MAP NB decision rule is commonly used, but it is sometimes derived in a different way [Chakrabarti *et al.*, 1997] [Ristad, 1995]. Instead of maximizing some aspect of the data, an expected value of the parameter is used,

$$\hat{\theta}_k^c = E[\theta_k^c | N_k^c] = \int \theta p(\theta | N_k^c) d\theta = \int \theta \frac{p(N_k^c|\theta)p(\theta)}{p(N_k^c)} d\theta. \tag{2.9}$$

$\hat{\theta}_k^c$ is the estimate of the parameter θ_k^c. N_k^c is the number of times word w_k appears in class c training documents. With a uniform prior, we get the MAP NB decision rule with $a_k = 1 \; \forall k$,

$$E[\theta_k^c | N_k^c] = \frac{N_k^c + 1}{N^c + V}. \tag{2.10}$$

V is the size of the vocabulary. Maximizing the posterior with a prior that prefers uniform parameters ($\alpha_k = 2 \; \forall k$) gives us the same parameter estimates as when a uniform prior and expected values are used.

2.4 Bayesian Naive Bayes

MAP Naive Bayes chooses a particular parameter vector, $\hat{\theta}^c$, for classification. This simplifies the derivation, but bottlenecks information about the training data for classification. An alternative approach is to use a distribution of parameters based on the data. This complicates the derivation somewhat since we don't evaluate $p(d|c, D)$ as $p(d|\hat{\theta}^c)$. Instead, we integrate over all possible parameters, using $p(\theta|D^c)$ as our belief that a particular set of parameters generated D^c.

As in ML & MAP Naive Bayes, we start with the Bayes optimal decision rule,

$$\hat{H}(d) = \operatorname{argmax}_c p(c|D, d) = \operatorname{argmax}_c p(d|c, D)p(c). \tag{2.11}$$

We expand $p(d|c, D)$ to

$$p(d|c, D) = \int p(d|\theta)p(\theta|D^c) d\theta. \tag{2.12}$$

$p(d|\theta) = \frac{f!}{\prod_k f_k!} \prod_k \theta_k^{f_k}$ is the multinomial likelihood. We expand the posterior via Bayes' Law, $p(\theta|D^c) = \frac{p(D^c|\theta)p(\theta)}{p(D^c)}$ and use the Dirichlet prior, as we did with MAP Naive Bayes. This gives us a Dirichlet posterior,

$$p(\theta|D^c) = \text{Dir}(\theta|\{\alpha_i + N_i^c\}) = \frac{\Gamma(\alpha + N^c)}{\prod_i \Gamma(\alpha_i + N_i^c)} \prod_i \theta_i^{N_i^c + \alpha_i - 1}. \tag{2.13}$$

Substituting this into equation 2.11 and selecting $p(c) = \frac{1}{m}$, we get

$$\hat{H}(d) = \text{argmax}_c \frac{\Gamma(\alpha + N^c)}{\prod_i \Gamma(\alpha_i + N_i^c)} \frac{\prod_i \Gamma(N_i^c + \alpha_i + f_i)}{\Gamma(N^c + \alpha + f)}. \tag{2.14}$$

This fully Bayesian derivation is distinct from the MAP and ML derivations, but it shares similarities. In particular, if we make the approximations

$$\frac{\Gamma(\alpha + N^c)}{\Gamma(\alpha + N^c + f)} \approx \frac{1}{(\alpha + N^c)^f} \quad \text{and} \quad \frac{\prod_i \Gamma(\alpha_i + N_i^c + f_i)}{\prod_i \Gamma(\alpha_i + N_i^c)} \approx \prod_i (\alpha_i + N_i^c)^{f_i}, \tag{2.15}$$

we get a decision rule very similar to that of MAP Naive Bayes,

$$\hat{H}(d) = \text{argmax}_{c \in C} \prod_k \left(\frac{\alpha_k + N_k^c}{\alpha + N^c} \right)^{f_k}. \tag{2.16}$$

The lone difference is that MAP Naive Bayes uses different Dirichlet hyper-parameters to achieve this rule.

Bayesian Naive Bayes is distinct from MAP Naive Bayes in its decision rule. As shown above, modifications can be made to make the two identical, but those modifications are not generally appropriate. In fact, the modifications exhibit the differences between MAP and Bayesian Naive Bayes. Compared to MAP, Bayesian Naive Bayes over-emphasizes words that appear more than once in a test document. Consider binary (+1,−1) classification with $N^{+1} = N^{-1}$. Let d be a test document in which the word w_k appears twice. The contribution for w_k in MAP Naive Bayes is $(a_k + N_k^c)^2$; the similar contribution for w_k in Bayesian Naive Bayes is $(\alpha_k + N_k^c)(\alpha_k + N_k^c + 1)$. The Bayesian term is larger even though other terms are identical. The difference is greater for a word that occurs more frequently.

2.5 Bayesian Naive Bayes Performs Worse In Practice

On one of the two data sets that we tried, we found that Bayesian NB (with a Dirichlet prior and $\alpha_k = 1\ \forall k$) performed worse than MAP NB (using a Dirichlet prior and $\alpha_k = 2\ \forall k$). This is not a sign that the Bayesian derivation is bad—far from it. The poor empirical performance is rather an indication that the Dirichlet prior is a poor choice or that the Dirichlet hyper-parameter settings are not well chosen. Well estimated hyper-parameters, or a different prior, such as the Dirichlet process, may yield better performance for Bayesian NB [Ferguson, 1973]. We show the empirical difference and give statistics exhibiting the conditions where classification differences occur.

We conducted classification experiments on the 20 Newsgroups and Industry Sector data sets. Table 2.1 shows empirical test error averaged over 10 test/train splits. See appendix A for a full description of the data sets and the preparations used for

Industry Sector	Training examples per class				
	52	20	10	3	1
MAP	0.434	0.642	0.781	0.910	0.959
Bayesian	0.486	0.696	0.825	0.932	0.970

20 Newsgroups	Training examples per class				
	800	250	100	30	5
MAP	0.153	0.213	0.305	0.491	0.723
Bayesian	0.154	0.211	0.302	0.490	0.726

Table 2.1: Shown are results of Naive Bayes multi-class classification using Bayesian and MAP NB on the 20 Newsgroups and Industry Sector data sets. Errors are the average of 10 trials. The differences in the 20 Newsgroups results are not statistically significant. Bayesian NB has higher error rates on the Industry Sector data set.

Industry Sector	Correct label	
	Bayesian	MAP
Max. term freq.	19.4	29.3

20 Newsgroups	Correct label	
	Bayesian	MAP
Max. term freq.	6.43	17.0

Table 2.2: Shown are maximum term frequencies of test documents when the two classification algorithms disagree. The “Bayesian” column gives the maximum frequency, averaged over the test documents, when Bayesian NB gives the correct label and MAP NB does not. “MAP” gives the statistic for the case that the MAP NB label is correct and the Bayesian NB label is wrong. Of the disagreements, MAP is correct when the most frequent word occurs often; Bayesian is correct when the most frequent word occurs less often.

each. The techniques perform equally well on the 20 Newsgroups data set. Although there are differences in the way each technique classifies documents, those differences do not result in large differences in error. Also neither technique consistently outperforms the other as we vary the amount of training data.

This is not the case for the Industry Sector data set. The differences in error between MAP and Bayesian NB are larger and one-sided; MAP NB has lower error at all levels of training data. Additional analysis shows that in cases where Bayesian and MAP NB don’t agree, there is a distinct difference in the frequency of words in the test document. When MAP produces the correct label, the word with the largest term frequency occurs more often than the word with the largest term frequency in documents that Bayesian labels correctly. The same trend is seen in the 20 Newsgroup results, but it does correlate with difference in error. Table 2.2 summarizes these statistics.

Since we use $\alpha_k = 1 \; \forall k$ for Bayesian NB, any word that does not occur often in a class of training data will be over-emphasized in the classification output (compared to MAP NB). But, our choice of $\{\alpha_k\}$ corresponds to a prior. $\alpha_k = 1$ corresponds to a preference for uniform parameter vectors—vectors where all words have the same probability. This isn't a reasonable prior for English or other languages. A more appropriate prior would cause only novel words to be over-emphasized.

The poor performance by Bayesian NB is not a fault of the classification algorithm, but rather a sign that our choice of prior or model is poor. The Bayesian derivation provides us with a classification rule that directly incorporates information from the training data and may be more sensitive to our choice of prior. Future work to better our choice of model and prior should improve the performance of Bayesian NB.

2.6 Naive Bayes is a Linear Classifier

MAP Naive Bayes is known to be a linear classifier. In the case of two classes, +1 and −1, the classification output is

$$h(d) = \log \frac{p(d|\hat{\theta}^{+1})p(+1)}{p(d|\hat{\theta}^{-1})p(-1)} \tag{2.17}$$

$$= \log \frac{p(+1)}{p(-1)} + \sum_k f_k \left(\log \frac{a_k + N_k^{+1}}{a + N^{+1}} - \log \frac{a_k + N_k^{-1}}{a + N^{-1}} \right) = b + \sum_k w_k f_k, \tag{2.18}$$

where $h(d) > 0$ corresponds to a +1 classification and $h(d) < 0$ corresponds to a −1 classification. We use w_k to represent the linear weight for the k^{th} word in the vocabulary. This is identical in manner to the way in which logistic regression and linear SVMs score documents. Logistic regression classifies according to

$$p(y = +1|x, w) = g(b + \sum_k w_k x_k), \tag{2.19}$$

where $p(y = +1|x, w) > 0.5$ is a +1 classification and $p(y = +1|x, w) < 0.5$ is a −1 classification. $g(z) = (1 + \exp(-z))^{-1}$. Similarly, the linear SVM classifies according to $h(x) = b + \sum_k w_k x_k$, assigning class +1 for $h(x) > 0$ and class −1 for $h(x) < 0$. Hence, all three algorithms are operationally identical in terms of how they classify documents. The only difference is in the way in which their weights are trained.

This similarity extends to multi-class linear classifiers. Softmax is the standard extension of logistic regression to the linear case. Given a multi-class problem with classes $\{1, \ldots, m\}$, Softmax computes

$$z_i = b_i + \sum_k w_k^i x_k \tag{2.20}$$

for each class and assigns probabilities $p(y = i|x, w) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$. The class with the largest z_i and hence the largest probability is declared the label for example x.

(a)

percentile	min. posterior
0%	0.05012
11%	0.96486
22%	0.99987
33%	1.00000
44%	1.00000
55%	1.00000
66%	1.00000
77%	1.00000
88%	1.00000
99%	1.00000

(b)

percentile	# digits
11%	1
16%	2
20%	3
24%	4
28%	5
31%	6
35%	7
38%	8
40%	9

Table 2.3: Shown are $max_c p(c|D,d)$ values produced by MAP Naive Bayes on 20 Newsgroup data. (a) shows the smallest value at each of 11 percentile levels. Naive Bayes produced a value of 1 on a majority of the test data. (b) shows the percentile at which rounding any posterior to the given number of digits would produce a value of 1. The posteriors tend to 1 rapidly.

Similarly the MAP Naive Bayes decision rule is

$$\hat{H}(d) = \text{argmax}_i\, p(d|\hat{\theta}^{c_i}) \tag{2.21}$$

$$= \text{argmax}_i \sum_k \log p(c_i) + f_k \log \frac{a_k + N_k^i}{a + N^i} = \text{argmax}_i \left(b_i + \sum_k w_k^i f_k \right). \tag{2.22}$$

Hence, Naive Bayes and Softmax are operationally identical. The extension of the linear SVM to multi-class also shares this form. The only distinction between these algorithms is in the way their weights are trained.

2.7 Naive Bayes Outputs Are Often Overconfident

Consider a pair of unfair coins. Each comes up heads 60% of the time. When we count only the times that both coins show the same side, heads appears 69% of the time. Coins which marginally show heads 90% of the time are heads 99% of the time when both coins show the same side. Consider casting a spell over our 90% heads coins so that the second coin always lands on the same side as the first. If we now model the two coins as being independent and observe a large number of flips, we would estimate that when both coins land on the same side, heads shows 99% of the time. In fact, the probability of such an event is only 90%. The same effect occurs in MAP Naive Bayes.

It is rare that words serve as exact duplicates of each other, such as in our coin example. However, distinguishing between 20 classes requires a mere 2 word vocabulary and 5 terms per document for correct classification; all remaining information about the class variable is either noisy or redundant. Text databases frequently have 10,000 to 100,000 distinct vocabulary words; documents often contain 100 or more terms. Hence, there is great opportunity for duplication.

To get a sense of how much duplication there is, we trained a MAP Naive Bayes model with 80% of the 20 Newsgroups documents. We produced $p(c|d, D)$ (posterior) values on the remaining 20% of the data and show statistics on $\max_c p(c|d, D)$ in table 2.3. The values are highly overconfident. 60% of the test documents are assigned a posterior of 1 when rounded to 9 decimal digits. Unlike logistic regression, Naive Bayes is not optimized to produce reasonable probability values. Logistic regression performs joint optimization of the linear coefficients, converging to the appropriate probability values with sufficient training data. Naive Bayes optimizes the coefficients one-by-one. It produces realistic outputs only when the independence assumption holds true. When the features include significant duplicate information (as is usually the case with text), the posteriors provided by Naive Bayes are highly overconfident.

Chapter 3

Analysis of Naive Bayes Parameter Estimates

Having an understanding of how MAP Naive Bayes parameter estimates affect classification is important. The quality of the parameter estimates directly affects performance. We show that Naive Bayes estimates are consistent; we then investigate their behavior for finite training data by analyzing their bias and variance. The bias in the estimate is a direct product of the prior and tends monotonically toward zero with more training data. The variance peaks when a word is expected to occur 1-2 times in the training data and falls off thereafter. This analysis shows that insufficient training examples in one class can negatively affect overall performance. The variance as a whole is the sum of the variances of the individual components. If a single class variance is large, the overall variance is also high.

3.1 Consistency

MAP Naive Bayes estimates a vector of parameters, $\hat{\theta}^c$ for the multinomial model. Each individual parameter, $\hat{\theta}^c_k$, is the estimated probability of word w_k appearing in a particular position of a class c document. Let $\{\alpha_k\}$ be the parameters of the Dirichlet prior ($\alpha = \sum_k \alpha_k$), $a_k = \alpha_k - 1$ ($a = \sum_k a_k$) and let N^c_k be the number of occurrences of word w_k in the training documents ($N^c = \sum_k N^c_k$). Then the MAP estimate for w_k is

$$\hat{\theta}^c_k = \frac{a_k + N^c_k}{a + N^c}. \tag{3.1}$$

A basic desirable property of parameter estimates is consistency, or the convergence of the estimates to the true values when the amount of data used to make the estimates grows large. Cover and Thomas describe the method of types as a way to describe properties of empirical distributions [Cover and Thomas, 1991]. Let X be a multinomial random variable with parameters $\{\theta^c_k\}$. Let p_X represent the distribution of the parameters. Let p_Y represent the empirical distribution when N^c samples are taken from X resulting in counts of $\{N^c_k\}$. Then, our MAP estimates are $\hat{\theta}^c_k = \frac{a_k+N^c_k}{a+N^c}$.

The probability of observing such counts and hence the probability of making such estimates is

$$p(\hat{\theta}^c|\theta^c) = \frac{N^c!}{\prod_k N_k^c!} \prod_k (\theta_k^c)^{N_k} = \frac{N^c!}{\prod_k N_k^c!} 2^{-N(H(p_Y)+D(p_Y||p_X))}, \tag{3.2}$$

The mean of our estimate is

$$\overline{\hat{\theta}_k^c} = \frac{a_k + N^c\theta_k^c}{a + N^c}, \tag{3.3}$$

which goes to θ_k^c as $N^c \to \infty$. The variance of our estimate is

$$\sigma_{c,k}^2 = \frac{N^c\theta_k(1-\theta_k^c)}{(a+N^c)^2}, \tag{3.4}$$

which goes to zero as $N^c \to \infty$. Hence, MAP estimates are consistent; in the limit, they are unbiased and have zero variance. So, as the size of the observed data grows large, our estimates converge to the true parameters.

3.2 Bias

Since we never have infinite training data in practice, it is more important to understand the behavior of estimates for finite training data. For a particular number of observed words, N^c, the bias in the estimate for word w_k is

$$\text{bias}(\hat{\theta}_k^c) = \frac{a_k + N^c\theta_k^c}{a + N^c} - \theta_k^c = \frac{a_k - a\theta_k^c}{a + N^c} \tag{3.5}$$

Hence, for words where $\theta_k^c > \frac{a_k}{a}$, the expected estimate is smaller and for $\theta_k^c < \frac{a_k}{a}$, the expected estimate is larger than the true value. This is a natural consequence of the choice of a Dirichlet prior. Also, bias lessens as the amount of training data grows large.

3.3 Variance

The variance of a parameter estimate yields little insight into the effect estimates have on classification. Since Naive Bayes is a linear classifier, a more useful variance quantity to examine is the variance of each individual term in the classification output. Let f_k be the frequency of word k in the test document ($f = \sum_k f_k$). Then

$$z_c = -f\log(a + N^c) + \sum_k f_k \log(a_k + N_k^c) \tag{3.6}$$

is the classification score for class c. The assigned class is the one with the largest score. The individual terms of the sum are independent (assuming N^c to not be

a) b)

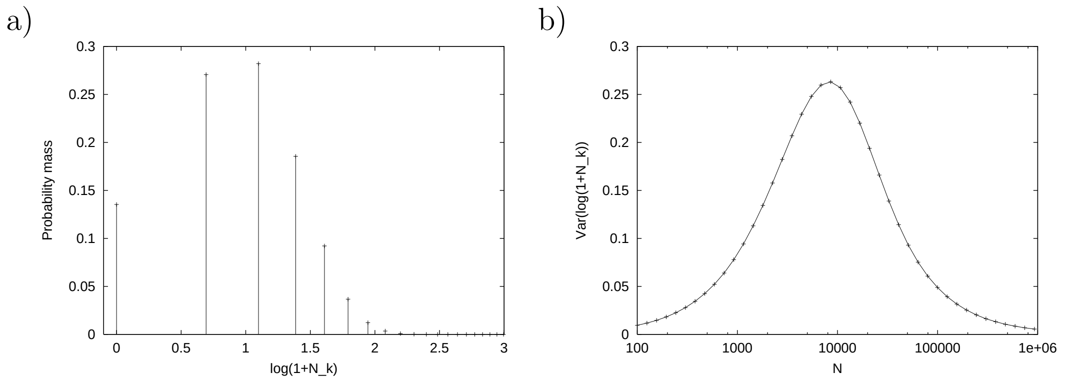


Figure 3-1: (a) is a plot of the pmf of $\log(1+N_k)$ for $\theta_k^c = 0.0002$ where $N = 10000$. (b) plots the variance of $\log(1+N_k)$ for $\theta_k^c = 0.0002$ as we vary N. Note the x-axis log scale. $\text{var}(\log(1+N_k))$ peaks when the word is expected to occur 1-2 times. (b) is representative of all θ_k^cs. The plot of $\text{var}(\log(1+N_k))$ peaks near $\theta_k^c = 1/N$ and has the same shape as the one shown.

fixed), so

$$\text{var}(z_i) = \sum_k f_k^2 \text{var}(\log(a_k + N_k^i)). \tag{3.7}$$

We assume the $\{f_k\}$ to be fixed and that the $\{N_k^c\}$ may vary. The variance of an individual term is

$$\text{var}(\log(a_k + N_k^i)) = E[(\log(a_k + N_k^i)^2] - E[\log(a_k + N_k^i)]^2. \tag{3.8}$$

Treating each N_k as a binomial with parameter θ_k^c, we get

$$E[\log(a_k + N_k^i)] = \sum_n \log(a_k + n) \binom{N}{n} (\theta_k^c)^n (1 - \theta_k^c)^{(N-n)}. \tag{3.9}$$

Although equation 3.9 is not difficult to compute, we approximate N_k as a Poisson with $\lambda = \theta_k^c$ and use Stirling's formula for $n!$ to arrive at

$$E[\log(a_k + N_k^i)] = \frac{N\theta_k^c \log(2)}{\exp(N\theta_k^c)} + \sum_{n=2}^{N} \frac{\log(a_k + n)}{\sqrt{2\pi n}} \exp(n(1 + \log(N\theta_k^c) - \log n) - N\theta_k^c). \tag{3.10}$$

We use this formula for the graphs that we present. The Poisson approximation is good for $\theta_k^c << 1$, which is generally the case in text.

Figure 3-1 shows plots of the the pmf and variance for a word with $\theta_k^c = 0.0002$. $\text{var}(\log(1+N_k))$ is maximized when w_k is expected to occur 1-2 times in the training data. This does not incorporate f_k; a word that occurs 1-2 times in the training data

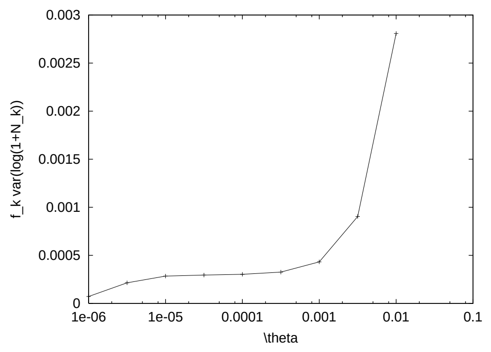


Figure 3-2: Shown is the per-word variance contribution to the classification output for $N = 1000000$, $f = 300$ and various values of θ_k^c . We assume that $f_k = f\theta_k^c$. Although $\mathrm{var}(\log(1 + N_k))$ is largest for $\theta_k^c = 0.000001$, larger values of θ_k^c yield larger per-word variance contributions.

for class c is unlikely to occur in test documents generated from class c. However, figure 3-1 does give us the ability to compare variances across classes. Let $\theta_k^{+1} = 0.02$ and $\theta_k^{-1} = 0.0002$ be the true parameters for w_k for the classes $+1$ and -1. If the training data for both classes consists of 10,000 words, $N^{+1} = N^{-1} = 10,000$ then the w_k contribution to the variance of the classification output will be much greater for class -1 than for class $+1$.

Figure 3-2 shows the variance contribution of individual tokens assuming that $f_k = f\theta_k^c$. Words with the largest θ_k^c contribute the largest variance to the classification output. $f_k \approx f\theta_k^c$ is only reasonable for class-independent words and for test documents drawn from class c. Words with large θ_k^c values often contribute the greatest amount of variance to classification outputs, but, a word with small θ_k^c can easily contribute a great deal of variance if w_k occurs frequently in the test document.

We can glean from figure 3-1 the effect of additional training data on classification. It is widely believed that additional training data improves classification. The plot of the variance of $\log(1 + N_k)$ shows that for every word, there is a point after which the variance contribution for that word diminishes with additional training data. Once that point is passed for most words, the overall variance in the classification output decreases monotonically. Before this point, output variance may increase with additional training data, but when the amount of training data is relatively small, bias is a significant factor. For $N = 1000$ and a word with $\theta_k^c = 0.00002$, the estimate may be $\hat{\theta}_k^c = 0.0001$, five times the actual parameter value. When the amount of training data is very small, bias plays a greater role in affecting classification performance. Our analysis of variance shows that after a point variance decreases monotonically for each word. This lessening of variance contributes to improved classification as the number of training examples increases.

category	word	log-odds ratio	$\hat{\theta}_k$
alt.atheism	atheism	0.013	0.0040
comp.graphics	jpeg	0.037	0.0073
comp.os.ms-windows.misc	windows	0.043	0.020
comp.sys.ibm.pc.hardware	scsi	0.033	0.012
comp.sys.mac.hardware	mac	0.024	0.012
comp.windows.x	window	0.024	0.0091
misc.forsale	sale	0.018	0.0076
rec.autos	car	0.043	0.017
rec.motorcycles	bike	0.045	0.010
rec.sport.baseball	baseball	0.016	0.0057
rec.sport.hockey	hockey	0.037	0.0078
sci.crypt	clipper	0.033	0.0058
sci.electronics	circuit	0.010	0.0031
sci.med	patients	0.011	0.0029
sci.space	space	0.035	0.013
soc.religion.christian	god	0.035	0.018
talk.politics.guns	gun	0.028	0.0094
talk.politics.mideast	armenian	0.039	0.0057
talk.politics.misc	stephanopoulos	0.024	0.0034
talk.religion.misc	god	0.011	0.011

Table 3.1: For each category in the 20 Newsgroups dataset, the word with the highest log odds ratio. A larger score indicates a word which is commonly found in the specified category, but rarely found in other categories. Words with high log odds ratios are good discriminants for the one vs. all problem.

3.4 The Danger of Imbalanced Class Training Data

An observation we can make from figure 3-1 is that classes with little observed training data (e.g. 5 documents of 200 words each, $N = 1000$) yield high-variance outputs. Few words that are useful for classification have $\theta_k^c > 0.01$. Table 3.1 gives a list of frequent, class-predictive words for the 20 Newsgroups data set. It gives a sense of the frequency with which words occur. The table shows the word with the greatest log-odds ratio for each class in the 20 Newsgroups data set. We define a log-odds ratio as

$$\text{LogOdds}(w_k|c_i) = p(w_k|c_i)\log\frac{p(w_k|c_i)}{p(w_k|\neg c_i)} = \theta_k^i \log\frac{\theta_k^i}{\sum_{j\neq i}\theta_k^j}. \tag{3.11}$$

Words with high log-odds ratio occur unusually frequently in class i and occur often within that class.

For $N = 1000$, words with $\theta_k^c \in (0.01, 0.0001)$ correspond to $\text{var}(\log(1 + N_k)) \geq$

0.05, all relatively large variances. In contrast, when $N = 10000$, $\text{var}(\log(1 + N_k)) < 0.01$ for $\theta_k^c = 0.01$. Larger amounts of observed data yield even smaller variances for words that occur frequently. Hence, if one class has little training data, its variance may be much greater than other classes.

Theorem 3.4.1 *Consider a two-class* ($+1$, -1) *classification problem. Let* $z_{+1}(d) = \log p(d|\hat{\theta}^{+1})p(+1)$ *and* $z_{-1}(d) = \log p(d|\hat{\theta}^{-1})p(-1)$*. Assume that* $var(z_{+1}(d)) > var(z_{-1}(d))$*. Then* $2var(z_{+1}(d)) > var(h(d)) > var(z_{+1}(d))$*.*

Proof: $h(d) = \log p(d|\hat{\theta}^{+1})p(+1) - \log p(d|\hat{\theta}^{-1})p(-1)$ (as given in equation 2.18). Since the two terms are independent, the variance of $h(d)$ is the sum of the variances of the two terms. $\square$

If one class has much higher variance than other classes, that variance will dominate the variance of the overall classification outputs. Ample training data will yield estimates that contribute little variance to the overall output; a dearth of examples in one class will contribute great variance. Hence, the performance of a Naive Bayes classifier can easily be dictated by the class with the smallest number of examples. The benefit that Naive Bayes receives from additional training data is marginal if the data is not distributed evenly across the classes.

Chapter 4

Error-correcting Output Coding

Error-correcting output coding (ECOC) is an approach for solving multiclass categorization problems originally introduced by Dietterich and Bakiri [1991]. It reduces the multiclass problem to a group of binary classification tasks and combines the binary classification results to predict multiclass labels. Others have experimentally shown that ECOC can improve text classification with Naive Bayes [Ghani, 2000] [Berger, 1999]. Here, we give detailed results on the 20 Newsgroups and Industry Sector data sets. We explain how our parameter estimate analysis predicts the success and failure of (MAP) Naive Bayes and its use in conjunction with ECOC. Certain ECOC classifiers outperform Naive Bayes. The performance of the binary classifiers in the ECOC scheme has a great impact on multiclass performance. Those that perform well do not suffer from too few examples and have relatively good binary performance. Additionally, we experiment with a linear loss function and find that it yields performance comparable to that of the best non-linear loss function that we tried. This is evidence that text classification using a bag-of-words representation is a linear problem. Note that throughout this section when we say "Naive Bayes," we are referring to MAP Naive Bayes with Dirichlet hyper-parameters $\alpha_k = 2 \ \forall k$.

4.1 Introduction

R is the *code matrix*. It defines the data splits which the binary classifier is to learn. $R_{i\cdot}$ is the i^{th} row of the matrix and defines the *code* for class i. $R_{\cdot j}$ is the j^{th} column of the matrix and defines a split for the classifier to learn. $R \in \{-1,+1\}^m \times \{-1,+1\}^l$ where m is the number of classes and l is the number of partitionings (or length of each code). In a particular column, $R_{\cdot j}$, -1 and $+1$ represent the assignment of the classes to one of two partitions. For this work, we use three different matrices, the one-vs-all (OVA) matrix, where each column has one $+1$ and is otherwise filled with -1 entries, the Dense matrix, where entries are independently determined by flipping a fair coin, assigning $+1$ for heads and -1 for tails and BCH codes, a matrix construction technique that yields high column- and row-separation [Ghani, 2000]. We use the BCH codes that Ghani has made available on-line at http://www.cs.cmu.edu/$\sim$rayid/ecoc.c

Let $(f_1, \ldots, f_l)$ be the classifiers trained on the partitionings indicated in the code matrix. Furthermore, let $g : \Re \rightarrow \Re$ be the chosen loss function. Then, the multiclass classification of a new example, x is

$$\text{argmin}_{c \in \{1,\ldots,m\}} \sum_{i=1}^{l} g(f_i(x) R_{ci}). \tag{4.1}$$

Allwein *et al.* give a full description of the code matrix classification framework and give loss functions for various models [2000]. We use "hinge" loss, $g(z) = (1 - z)_+$, for the SVM, since that is the loss function for which the SVM is optimized. Unlike the SVM, Naive Bayes does not optimize a loss function. However, we find that the hinge loss function yields lower error than the 0/1 and logistic loss functions, so we use the hinge loss for our Naive Bayes ECOC classifier as well.

4.2 Additive Models

ECOC resides within a greater class of models known as additive models. An additive model for classification has the form

$$\text{argmin}_{c \in \{1,\ldots,m\}} \sum_{i=1}^{l} w_i f_{ic}(x), \tag{4.2}$$

where $f_{ic}(x)$ is an arbitrary function of the data and the w_i are weights. ECOC uses uniform ($w_i = 1 \; \forall i$) weights. The name comes from the fact that the final output is determined by a (weighted) summing of outputs of possibly non-linear functions. All algorithms which determine their final output by voting fall into this class of algorithms. In fact, an effective way to make use of a collection of experts is to have them vote. This is very similar to how ECOC works. It creates a handful of experts, each of which specializes at partitioning the set of classes in a particular way. ECOC allows each expert to vote for the set of classes within which it believes the example to lie. With non-binary loss functions, these votes are weighted by the confidence of the expert. The additive aspect imposes a linear constraint on the final output. This restricts the expressiveness of the output (as a function of the experts), but also tempers the final output. However, there is no overall constraint on the expressiveness of the classifier (so long as the experts are sufficiently expressive).

4.2.1 The relation to boosting

Some algorithms, such as logistic regression, softmax, the linear SVM, its multiclass extension and MAP Naive Bayes are trivially additive models because they are linear classifiers. The loss function in ECOC may make it non-linear even when the individual classifiers are linear. Another model which is a non-linear additive model is boosting. Boosting shares a similarity with ECOC: it is composed of separately-trained binary classifiers. The original formulation of boosting, AdaBoost, was designed to perform only binary classification [Freund and Schapire, 1999]. AdaBoost composes binary classifiers which are experts at different parts of the example space by training each classifier with a different weighted set of examples. In the multiclass case, the creation of experts can be done by partitioning according to class and/or weighting the individual examples. ECOC only specifies a partitioning according to class, whereas multiclass boosting schemes (such as AdaBoost.OC and AdaBoost.ECC) specify partitionings of both the classes and the example space [Freund and Schapire, 1996] [Guruswami and Sahal, 1999]. Multiclass boosting and ECOC are closely related: multiclass boosting is an extension of ECOC. Multiclass boosting specifies a particular binary learner (although the underlying weak learner is unspecified) and imposes weights on the loss output of each binary learner. Also, multiclass boosting algorithms train binary classifiers as a function of previous classifiers. This is not usually done with ECOC. However, a main thrust of Boosting is its creation of various meaningful binary sub-problems. In the multiclass case, ECOC does this by partitioning examples according to class. The classes give meaningful locations in which to draw boundaries. It is not clear that multiclass boosting schemes offer any advantage over a strong binary classifier being used with ECOC.

4.3 The Support Vector Machine

The Support Vector Machine is a classifier, originally proposed by Vapnik, that finds a maximal margin separating hyperplane between two classes of data [1995]. There are non-linear extensions to the SVM, but Yang found the linear kernel to outperform non-linear kernels in text classification. In our own informal experiments, we also found that linear performs at least as well as non-linear kernels. Hence, we only present linear SVM results. We use the SMART 'ltc' transform and use the SvmFu package for running experiments [Rifkin, 2000].

We introduce the SVM and show results on the SVM to contrast the Naive Bayes performance. The SVM is known to perform well in the case of imbalanced training data, whereas theorem 3.4.1 gives us reason to believe that Naive Bayes does not handle imbalanced training data well. The SVM results give us a baseline with which to grade Naive Bayes' performance.

4.4 Experiments

Table 4.1 shows the results of our ECOC experiments. Appendix A describes the preparations we used for each data set. All of our results are averaged over 10 random train/test splits of the data. The SVM consistently performs better than Naive Bayes as the binary classifier of an ECOC scheme. But, the degree of difference depends both on the matrix type and the data set.

20 Newsgroups	800		250		100		30	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA	0.131	0.146	0.167	0.199	0.214	0.277	0.311	0.445
Dense 15	0.142	0.176	0.193	0.222	0.251	0.282	0.366	0.431
BCH 15	0.145	0.169	0.196	0.225	0.262	0.311	0.415	0.520
Dense 31	0.135	0.168	0.180	0.214	0.233	0.276	0.348	0.428
BCH 31	0.131	0.153	0.173	0.198	0.224	0.259	0.333	0.438
Dense 63	0.129	0.154	0.171	0.198	0.222	0.256	0.326	0.407
BCH 63	0.125	0.145	0.164	0.188	0.213	0.245	0.312	0.390

Industry Sector	52		20		10		3	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA	0.072	0.357	0.176	0.568	0.341	0.725	0.650	0.885
Dense 15	0.119	0.191	0.283	0.363	0.461	0.542	0.738	0.805
BCH 15	0.106	0.182	0.261	0.352	0.438	0.518	0.717	0.771
Dense 31	0.083	0.145	0.216	0.301	0.394	0.482	0.701	0.769
BCH 31	0.076	0.140	0.198	0.292	0.371	0.462	0.676	0.743
Dense 63	0.072	0.135	0.189	0.279	0.363	0.453	0.674	0.745
BCH 63	0.067	0.128	0.176	0.272	0.343	0.443	0.653	0.734

Table 4.1: Above are results of multiclass classification experiments on the 20 Newsgroups (top) and Industry Sector (bottom) data sets. The top row of each table indicates the number of documents/class used for training. The second row indicates the binary classifier. The far left column indicates the multiclass technique. Entries in the table are classification error. We thank Ryan Rifkin for providing us with the SVM results.

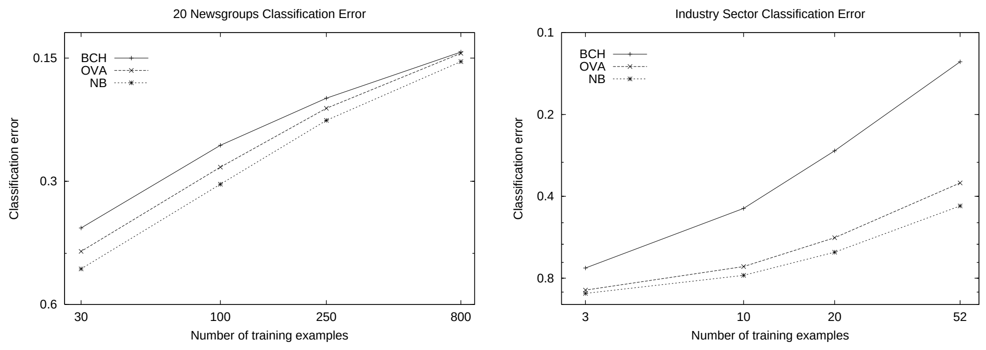


Figure 4-1: Shown are multiclass errors for three different classification algorithms. OVA refers to ECOC with the one-vs-all matrix. BCH refers to ECOC with the BCH-63 matrix. Naive Bayes is used as the binary classifier for both OVA and BCH in this plot. NB refers to regular Naive Bayes. Note that OVA and NB follow similar trends; OVA outperforms NB by a small margin. BCH greatly outperforms OVA and NB on Industry Sector but only marginally outperforms them on 20 Newsgroups. Note the log scale on both axes.

4.4.1 The success and failure of Naive Bayes

Figure 4-1 compares the performance of ECOC/OVA with regular NB and ECOC/BCH. Note that across both data sets, the performance of ECOC/OVA and regular NB follows a consistent pattern across different train set sizes: regular NB consistently performs slightly worse than ECOC/OVA. This harkens back to Berger's claim that ECOC/OVA classification with Naive Bayes is very similar to regular Naive Bayes classification [Berger, 1999]. In fact, the "one" components of the binary classifiers are simply the individual components of the regular Naive Bayes classifier. OVA adds outputs to compare against (the "all"). This additional information allows OVA to outperform NB somewhat. OVA is innately tied to the performance of regular Naive Bayes. But, what causes regular Naive Bayes to perform poorly?

To understand the performance of regular Naive Bayes, we return to theorem 3.4.1. Theorem 3.4.1 gives us the intuition that a regular Naive Bayes classifier is only good as its worst component. Also, since additional training examples reduce variance in a Naive Bayes classifier, the class with the fewest examples is likely to dictate the performance of the overall classifier. Unlike 20 Newsgroups, the training data in Industry Sector is not even across classes. The class with the fewest training examples has 12. The class with the most has 52 training examples. For the "52" and "20" training levels, some classes use fewer than 52 and 20 training examples, respectively. This correlates well with the improved performance of ECOC/BCH in figure 4-1. The BCH matrix shows the greatest gains over OVA and NB when the largest number of training examples is used. This is the case where there is the largest disparity in number of training examples used for different classes and is also the case where theorem 3.4.1 is most applicable.

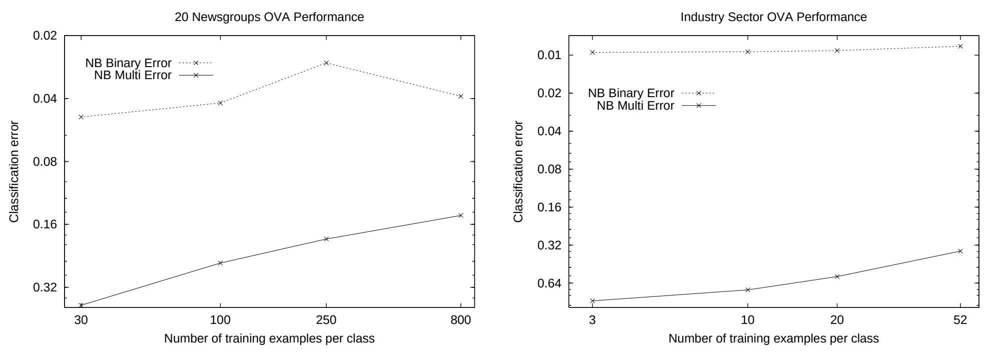


Figure 4-2: Multiclass error improves as the number of training examples increases, but binary error improves marginally for Industry Sector and degrades for 20 Newsgroups. Shown is the performance of ECOC with OVA and Naive Bayes as the binary classifier. Since the OVA binary classifiers have a lop-sided example distribution, guessing achieves a binary error of 0.05 for 20 Newsgroups and 0.01 for Industry Sector. Binary error is only loosely tied to binary classifier strength. Note the log scale on both axes.

		Guess	
		$+1$	-1
True	$+1$	tp	fn
Label	-1	fp	tn

Table 4.2: The performance of a binary classifier can be described with a 2x2 confusion matrix, as shown. Two letters describe each entry. "t" stands for true. "f" is false. "p" is positive. "n" is negative. The detection rate is tp/(tp+fn). The false alarm rate is fn/(tp+fn). The miss rate is fp/(tn+fp). ROC breakeven is the average of the alarm and miss rates when the difference between them is minimized.

4.4.2 Multiclass error is a function of binary performance

The performance of an ECOC classifier is affected by a number of factors: (1) binary classifier performance, (2) independence of the binary classifiers, and (3) the loss function. Of these, we find binary performance to be the most influential in multiclass text classification. We use error to measure multiclass performance. However, we avoid binary error as a measure of binary performance. Figure 4-2 shows why. Additional training examples yields improved multiclass error, but binary error rises and then falls using 800 training examples/class on the 20 Newsgroups data set. The OVA matrix partitions examples very unevenly, assigning most examples to a single class. Hence, error mainly judges the classifiers performance on examples of that class. A better measure is one that evenly weights performance on the two classes. We propose ROC breakeven as such a measure. Table 4.2 shows terms used to describe the output of a classifier. We define the ROC breakeven as the average of

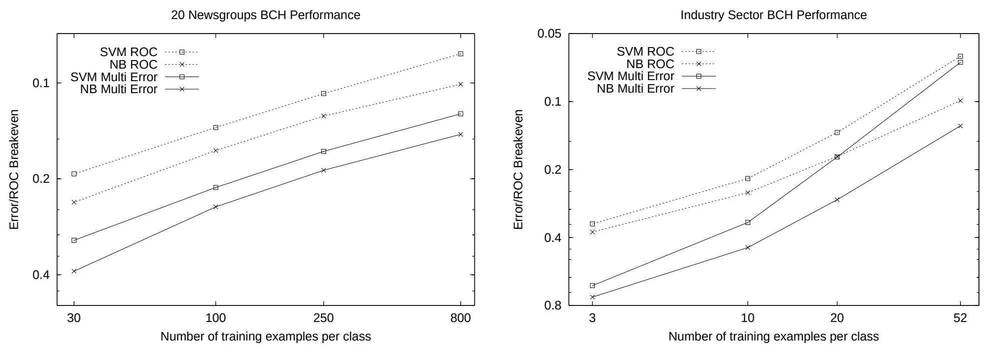
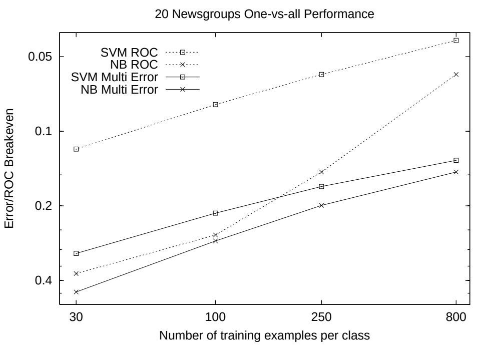


Figure 4-3: Shown is a comparison between ROC breakeven and multiclass error of ECOC using a BCH-63 matrix and the SVM and Naive Bayes as the binary classifier. We see that ROC breakeven largely dictates multiclass error. Trends in the ROC breakeven curves are reflected in the multiclass error curves. The maximum number of examples/class is used. Note the log scale on both axes.

the miss and false alarm rates at the point where the difference between false alarm rate and the miss rate is minimum. Note that unlike precision-recall breakeven, the ROC breakeven is always achievable. We achieve different rates by modifying the bias term of the classifier. ROC breakeven selects the bias such that the classifier performs as well on examples of class $+1$ as examples of class -1. ROC breakeven allows us to better judge the strength of a binary classifier when the example distribution is uneven. When the example distribution is even, ROC breakeven is nearly identical to binary error.

Figure 4-3 gives a comparison between multiclass error and ROC breakeven for ECOC classification with a BCH-63 matrix. The SVM achieves lower ROC breakeven on both data set and correspondingly achieves lower multiclass error. The figure makes the relationship between ROC breakeven and multiclass error clear. On 20 Newsgroups, there is a relatively consistent relationship between SVM and NB ROC breakeven. The gap between the two remains constant as the number of training examples increases. This is mirrored in the multiclass error. The SVM outperforms NB by a consistent margin. On Industry Sector, ROC breakeven is close at 3 training examples/class, but quickly diverges. Multiclass error shows the same pattern. SVM and NB multiclass errors are close at 3 examples/class, but at 52 examples/class, the SVM multiclass error is just over half that of the NB multiclass error. The performance of the binary classifier has great impact on the multiclass performance.

The trends seen in ECOC classification with a BCH-63 matrix are repeated in the OVA matrix results. Figure 4-4 shows these results. On Industry Sector, SVM ROC breakeven improves more quickly than NB ROC breakeven as the number of training examples increases. Multiclass error follows in suit, decreasing to an error of 0.072 at a binary ROC breakeven of 0.036. Naive Bayes lags behind with a multiclass error of 0.357 at a binary ROC breakeven of 0.282. The results on 20 Newsgroups are similar, although large differences in binary ROC have less of an effect on multiclass error.



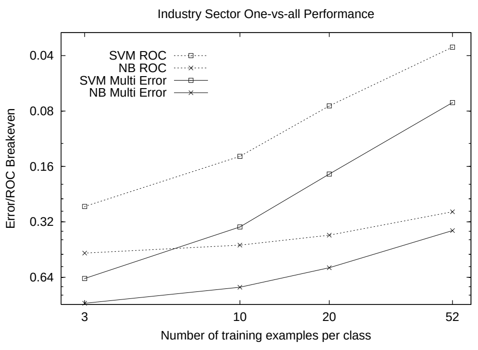


Figure 4-4: Shown is ROC breakeven and multiclass error for ECOC with the OVA matrix. Changes in ROC breakeven are directly reflected in multiclass error. Multiclass error changes gradually for 20 Newsgroups, but trends in ROC breakeven are evident in the multiclass error. The maximum number of examples/class is used. Note the log scale on both axes.

Lower ROC breakeven yields lower multiclass error and as the ROC breakevens of the SVM and NB converge, so do their multiclass errors.

The plots in figure 4-4 show that there are clearly factors other than binary performance at work. For example, an ROC breakeven of 0.282 for Naive Bayes on the Industry Sector data set (52 examples/class) yields a multiclass error of 0.357, while an ROC breakeven of 0.264 for the SVM (3 examples) yields multiclass error of 0.650. The SVM has higher multiclass error even though its ROC breakeven is lower. This is due to correlation between binary classifiers. When there are only 3 examples/class, the SVM classifiers produce identical labels more often than when more training data is available. For example, on average, a pair of 3 example SVM binary classifiers (trained using an OVA split of the data) produce the same label 99.77% of the time. The average pair of NB binary classifiers trained with 52 examples produce the same label 99.54% of the time. Greater independence between classifiers allows lower multiclass error in an ECOC scheme when the binary classifiers show higher ROC breakeven scores.

The full binary error and ROC breakeven results can be found in table 4.3. As we have seen in the figures and as can be seen in the table, ROC breakeven is well correlated with multiclass error. Other factors are at work—identical NB and SVM ROC breakevens does not yield identical multiclass errors. However, trends in ROC breakeven are clearly reflected in multiclass error. This is not the case with binary error, at least for the OVA matrix (where ROC breakeven and binary error differ). ROC breakeven is clearly a good indicator of multiclass performance as it better judges the strength of the classifier when the example distribution is skewed.

20 Newsgroups	800		250		100		30	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA/Error	0.015	0.039	0.021	0.027	0.03	0.042	0.044	0.049
OVA/ROC	0.043	0.059	0.059	0.146	0.078	0.262	0.118	0.375
BCH/Error	0.079	0.101	0.105	0.121	0.135	0.151	0.194	0.224
BCH/ROC	0.081	0.101	0.108	0.127	0.138	0.163	0.193	0.237

Industry Sector	52		20		10		3	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA/Error	0.003	0.008	0.005	0.009	0.007	0.009	0.009	0.010
OVA/ROC	0.036	0.282	0.075	0.378	0.141	0.428	0.264	0.473
BCH/Error	0.062	0.100	0.137	0.176	0.218	0.253	0.347	0.376
BCH/ROC	0.063	0.099	0.137	0.175	0.219	0.253	0.348	0.378

Table 4.3: Shown are binary errors and ROC breakeven points for the binary classifiers trained according to the matrix columns. Results for the Dense matrix are omitted since they are nearly identical to the BCH results. Table entries are averaged over all matrix columns and 10 train/test splits. Error is a poor judge of classifier strength for the OVA matrix. Error increases with more examples on 20 Newsgroups. Note that error and ROC breakeven numbers are very similar for the BCH matrix.

20 Newsgroups	Hinge	Linear
OVA/SVM	0.131	0.131
OVA/NB	0.146	0.146
BCH 63/SVM	0.125	0.126
BCH 63/NB	0.145	0.144

Industry Sector	Hinge	Linear
OVA/SVM	0.072	0.072
OVA/NB	0.357	0.357
BCH 63/SVM	0.067	0.067
BCH 63/NB	0.128	0.127

Table 4.4: Shown are multiclass errors on two data sets and a variety of ECOC classifiers. Errors are nearly identical between the hinge and linear loss functions. Although ECOC provides opportunity for non-linear decision rules through the loss function, the use of a non-linear loss function provides no practical benefit.

4.4.3 Non-linear loss does not affect ECOC performance

Another factor which can greatly impact ECOC multiclass error is the loss function. We use the hinge function for our experiments, $g(z) = (1-z)_+$, which exhibits a non-linearity at $z = 1$. Using this loss function allows ECOC to express functions that linear classifiers, such as Naive Bayes and the linear SVM, cannot express. However, the fact that ECOC is non-linear does not provide empirical benefit, at least in our experiments. Table 4.4 shows results of experiments that we ran to compare the hinge loss function to a trivial linear loss function, $g(z) = -z$. We find practically no difference in multiclass error compared to using the hinge loss function. The results we show use the maximum number of training examples (up to 52/class for Industry Sector and 800/class for 20 Newsgroups), but results are similar when fewer training examples are used. The confidence information contributed by the loss function is important for text classification, but non-linearity provides no practical benefit. The linear loss function yields a completely linear system (since both our NB and SVM classifiers are linear). This contributes evidence that text classification with bag-of-words representation is a linear problem.

Chapter 5

Feature Selection

Feature selection is an essential part of text classification. Document collections have 10,000 to 100,000 or more unique words. Many words are not useful for classification. Restricting the set of words that are used for classification makes classification more efficient and can improve generalization error. We describe how the application of Information Gain to feature selection for multiclass text classification is fundamentally flawed and compare it to a statistics-based algorithm which exhibits similar difficulties. A text feature selection algorithm should select features that are likely to be drawn from a distribution which is distant from a class-neutral distribution. Neither of the two algorithms do this. We describe a framework for feature selection that encapsulates this notion and exposes the free parameters which are inherent in text feature selection. Our framework provides a basis for new feature selection algorithms and clarifies the intent and design of such algorithms.

5.1 Information Gain

Information gain (IG) is a commonly used score for selecting words for text classification [Joachims, 1997; McCallum and Nigam, 1998; Yang and Pedersen, 1997; Mitchell, 1997]. It is derived from information theoretic notions. For each word, IG measures the entropy difference between the unconditioned class variable and the class variable conditioned on the presence or absence of the word,

$$IG = H(C) - H(C|W_k) = \sum_{c \in C} \sum_{w_k \in \{0,1\}} p(c, w_k) \log \frac{p(c|w_k)}{p(c)}. \tag{5.1}$$

This score is equivalent to the mutual information between the class and word variables, $IG = I(C; W_k)$. Hence, this score is sometimes called mutual information. The probabilities correspond to individual word occurrences. $w_k = 1$ corresponds to the occurrence of word w_k. $w_k = 0$ corresponds to the occurrence of some other word. We treat every token in the data as a binomial event and estimate the probabilities in equation 5.1 via maximum likelihood. Let f_k^c be the number of occurrences of word

w_k in class c ($f_k = \sum_c f_k^c$). Let $N^c = \sum_k f_k^c$ ($N = \sum_c N^c$). Then

$$IG = \sum_{c \in C} f_k^c/N \log \frac{f_k^c/f_k}{N^c/N} + (N^c - f_k^c)/N \log \frac{(N^c - f_k^c)/(N - f_k)}{N^c/N}. \qquad (5.2)$$

For feature selection, IG is computed for every word and words with larger scores are retained.

5.2 Hypothesis Testing

A desirable property of a feature is for its distribution to be highly dependent on the class. Words that occur independent of the class give no information for classification. A natural approach to developing a metric for filtering features is to determine whether each word has a class-independent distribution and to eliminate the word if it has such a distribution. In statistics, the problem of determining whether data is generated from a particular distribution is known as hypothesis testing. One proposes a model and parameters and ranks data according to its likelihood.

For text feature selection, we call this feature selection score HT. We consider a single word, w_k, and treat its f_k appearances in the training data as f_k draws from a multinomial where each event is a class label. Our hypothesized parameters are $\tilde{p} = \{N^c/N\}$. These parameters correspond to word occurrence being irrelevant of class, i.e. $\theta_k^1 = \cdots = \theta_k^m$ in the multinomial model. Our test statistic, which determines the ordering of data, is the difference in log-likelihoods between a maximum likelihood estimate, $\hat{p} = \{f_k^c/f_k\}$, and the hypothesized parameters,

$$HT(\hat{p}, \tilde{p}) = 2[l(\hat{p}) - l(\tilde{p})] = 2 \sum_c f_k^c \log \frac{f_k^c/f_k}{N^c/N}. \qquad (5.3)$$

$HT > 0$ always and larger HT values correspond to data that is less likely to have been generated by the proposed model. We keep words with large HT values and discard words with small HT values. Note that this score is similar to the IG score.

5.3 The Generalization Advantage of Significance Level

It is common for feature selection to be performed in terms of the number of features. For example, when using the IG score, one does not usually select an IG cutoff and eliminate all words with IG score less than that. Rather, one ranks words by their IG score and retains the top N scoring words. However, the number of words that should be retained for a particular application varies by data set. For example, McCallum and Nigam found that the best multinomial classification accuracy for the 20 Newsgroups data set was achieved using the entire vocabulary (62,000+ words) [1998]. In contrast, they found that the best multinomial performance on the "interest" category of the Reuters data set was achieved using about 50 words. An advantage of the HT score is that the number of words to be selected can be specified in terms of a significance level. Let HT_{cut} be the chosen cutoff HT score. The significance level corresponding to HT_{cut} is

$$SL = \Pr\{HT(\hat{p}, \tilde{p}) \geq HT_{\text{cut}} | \; \hat{p} \text{ is a sample estimate of } \tilde{p}\}. \tag{5.4}$$

$\tilde{p}$ is fixed; $\hat{p}$ is variable. $SL = 0.10$ selects words with empirical distributions that occur in only 10% of draws from the hypothesis distribution; selected words are atypical of the class-neutral distribution. This is more intuitive than simply selecting an HT or IG cutoff and may allow generalization across different data sets and conditions. Using significance level to choose a number of words for feature selection gives an easy-to-interpret understanding of what words are retained.

5.4 The Undesirable Properties of IG and HT

The application of IG and HT to text classification ignores critical aspects of text. Most words occur sparsely and only provide information when they occur. IG expects a word to provide information when it does not occur. Both IG and HT have a tendency to give higher scores to words that occur more often. For example, if $\tilde{p} = \{1/2, 1/2\}$, $\hat{p} = \{2/5, 3/5\}$ and $f_k = 10000$, $HT \approx 201.3$. More than 99.9% of draws from $\tilde{p}$ have a HT score less than 201.3. However, words which are devoid of class information have such empirical distributions. They are given a high score by IG and HT because they provide a significant reduction in entropy and there is little chance that they could have been drawn from the hypothesis distribution. The fact that the true distribution is probably very close to the hypothesized distribution is ignored by IG and HT. A word that occurs just a few times (e.g. $f_k = 7$) can never have a high IG or HT score because its non-occurrences provide little information and since the most extreme empirical distribution is a relatively common draw from the hypothesis distribution. For example, the chance of observing $\hat{p} = \{1, 0\}$ or $\hat{p} = \{0, 1\}$ from 7 draws of a multinomial with parameters $\tilde{p} = \{1/2, 1/2\}$ is $2/2^7 \approx 0.0156$.

The appearance of a single word can sometimes be used to predict the class (e.g. "Garciaparra" in a "baseball" document). However, a non-appearance is rarely informative (e.g. "Garciaparra" won't appear in all "baseball" documents). A text feature selection algorithm should retain words whose appearance is probably highly predictive of the class. In this sense, we want words that are discriminative.

5.5 Simple Discriminative Feature Selection

A simple score for selecting discriminative features is

$$S = \text{argmax}_c \, p(c|w_k), \tag{5.5}$$

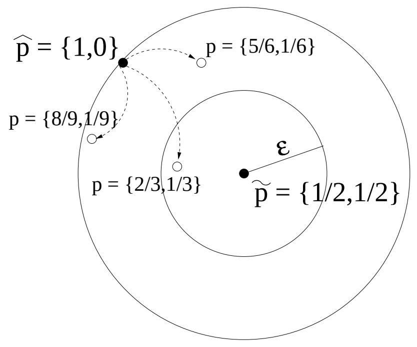


Figure 5-1: Our new feature selection framework views text feature selection as a problem of finding words (their empirical distribution represented by $\hat{p}$) which are unlikely to have a true distribution, p, within ϵ of the class independent distribution, $\tilde{p}$. The dashed arrows point to distributions from which $\hat{p}$ could have been drawn.

where $p(c|w_k)$ is the probability of the class being c given the appearance of word w_k. This gives the largest score to words which only appear in a single class. If such a word appears in a document, we know without a doubt what class that document belongs to. We cannot find $p(c|w_k)$, but we can make an estimate of it based on $\hat{p}$. A MAP estimate with Dirichlet $\{\alpha_c = 2\}$ prior gives us

$$S = \text{argmax}_c \frac{f_k^c + 1}{f_k + m}. \tag{5.6}$$

The setting of these hyper-parameters encode a preference for the uniform distribution, but there is no reason to believe that other choices are not more appropriate. The choice of prior is important as it serves as a measure of confidence for the empirical distribution. If the prior is a Dirichlet that prefers the class-neutral distribution over all others, $\{\alpha_c = cN^c/N\}$, the estimate of p for a word lies on the line connecting $\hat{p}$ and $\tilde{p}$. The prior dictates how close to $\tilde{p}$ the estimate is for a given number of draws.

5.6 A New Feature Selection Framework

The simple score we describe selects discriminative features, but is limiting as it imposes a specific distance metric. We describe a framework for text feature selection that exposes parameters of a feature selection method which are not always made explicit.

Figure 5-1 gives a visual description of this framework. To develop the framework, we extend HT in two important ways. First, we introduce an ϵ-ball around the hypothesis distribution. This serves to define distributions that are nearly class-independent. Second, we define a metric for measuring distances between distributions. This is used to determine the distribution in the ϵ-ball which is nearest to the empirical distribution. Let p_{near} be the distribution within the ϵ-ball which is nearest to $\hat{p}$. HT judges the possibility of $\hat{p}$ being drawn from $\tilde{p}$. In the new framework, we evaluate the probability of $\hat{p}$ being drawn from p_{near}. We select words that are likely to have distributions outside of the ϵ-ball—distributions which are far from the class-independent distribution. So, the new feature selection framework has as parameters

- ϵ, to define a set of distributions close to $\tilde{p}$,
- a metric, $d(p, q)$, to determine p_{near}, and
- a significance level, SL, for comparing empirical and true distributions.

As before, we use a hypothesis test score to define significance level,

$$NHT(\hat{p}, p_{\text{near}}) = 2[l(\hat{p}) - l(p_{\text{near}})]. \tag{5.7}$$

Given a cutoff choice for NHT, the significance level is defined as

$$SL = \Pr\{NHT(\hat{p}, p_{\text{near}}) > NHT_{\text{cut}} | \ \hat{p} \text{ is a sample estimate of } p_{\text{near}}\}. \tag{5.8}$$

A word ($\hat{p}$) is selected iff $NHT(\hat{p}, p_{\text{near}}) > NHT_{\text{cut}}$ where NHT_{cut} is defined by the chosen significance level and p_{near} is the distribution in the ϵ-ball that is closest to $\hat{p}$ (defined by $d(\tilde{p}, p_{\text{near}})$). Since words with empirical distributions near $\tilde{p}$ are discarded, a smaller cutoff and larger significance level can be used. Thus, NHT will include more discriminative words than IG or HT for the same number of selected features.

This new framework exposes the fundamental parameters variables in a text feature selection scheme where only word appearances are used. ϵ compensates for the fact that words are not truly drawn from a multinomial by eliminating words that are close to the class-neutral distribution. SL allows the user to select the amount of evidence required to show that a word is not drawn from a class-neutral distribution. $d(p, q)$ defines the closeness of two distributions and specifies (along with ϵ) to which distribution empirical distributions should be compared. This new framework selects words that are likely to be drawn from a discriminative distribution. Unlike HT and IG, it accounts for the fact that text is not a multinomial and empirical distributions that are close to the class-neutral distribution are unlikely to be informative with respect to the class variable.

Chapter 6

Conclusion

The focus of this thesis has been the application of Naive Bayes to multiclass text classification and has resulted in several new insights. Our parameter estimate analysis shows that Naive Bayes performs poorly when one class has relatively few examples. We also empirically showed that ECOC performance is mainly a result of binary performance. When the binary classifiers in ECOC have sufficient examples, ECOC performs much better than regular Naive Bayes. Furthermore, we showed that a commonly-used text feature selection algorithm is not good for multiclass text classification because it judges words by their non-appearances and has a bias to words that appear often. We proposed to select features by whether or not their distribution is discriminative and gave a framework which exposes the free parameters in such a scheme.

In terms of future work, the choice of the prior can greatly affect classification, especially for words with few observations, but its choice is not well understood. Better selection of the prior may lead to improved classification performance. Moreover, we along with others have observed that linear classifiers perform as well or better than non-linear classifiers on text classification with a bag-of-words representation. Determining whether this is generally true and understanding why this is the case is important. In our ECOC experiments, the performance of a particular matrix varied by data set and the amount of training data. Additional gains may be possible by developing algorithms to successively tune the columns of the ECOC matrix to the specific problem. We also envision to be able to use unlabeled data with EM to counter the limiting effect of classes with only a few labeled examples.

Appendix A

Data Sets

For our experiments, we use two different commonly used data sets [McCallum and Nigam, 1998; Slonim and Tishby, 1999; Berger, 1999; Ghani, 2000].We use McCallum's rainbow to pre-process the documents [1996].

20 Newsgroups is a data set collected and originally used for text classification by Lang [1995b] [Lang, 1995a]. It contains 19,974 non-empty documents evenly distributed across 20 categories, each representing a newsgroup. We remove all headers, UU-encoded blocks and words which occur only once in the data. The vocabulary size is 62061. We randomly select 80% of documents per class for training and the remaining 20% for testing. This is the same pre-processing and splitting as McCallum and Nigam used in their 20 Newsgroups experiments [McCallum and Nigam, 1998].

The Industry Sector data is a collection of corporate web pages organized into categories based on what a company produces or does [Nigam, 2000]. There are 9619 non-empty documents and 105 categories. We remove headers, prune stoplist words and words that occur only once. We include HTML for our experiments. However, we find that regular Naive Bayes and ECOC with OVA and Naive Bayes do better when HTML is first removed. The difference does not change our conclusions, but is of note. Our vocabulary size is 55197. We randomly select 50% of documents per class for training and the remaining 50% for testing. We create subsets of the training set to observe the effects of varying amounts of training data. This is similar to the pre-processing and splitting as Ghani used in his Industry Sector experiments [Ghani, 2000]. The only difference is that Ghani excluded HTML in his pre-processing.

Text classification experiments often include a feature selection step which may improve classification. McCallum and Nigam performed feature selection experiments on a modified version of the Industry Sector data set and the 20 Newsgroups data set; in neither case did feature selection significantly improve classification [1998]. In our own experiments, we found information gain feature selection to not improve classification. We use the full vocabulary for all of our experiments.

Bibliography

[Allwein *et al.*, 2000] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[Berger, 1999] Adam Berger. Error-correcting output coding for text classification. In *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, Stockholm, Sweeden, 1999.

[Chakrabarti *et al.*, 1997] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Using taxonomy, discriminants and signatures for navigating in text databases. In *Proceedings of the 23rd VLDB Conference*, 1997.

[Cover and Thomas, 1991] Thomas Cover and Joy Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[Dietterich and Bakiri, 1991] Tom G. Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 572–577, Anaheim, CA, 1991. AAAI Press.

[Domingos and Pazzani, 1996] Pedro Domingos and Michael Pazzani. Beyond independence: conditions for the optimality of the simple bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, 1996.

[Dumais *et al.*, 1998] Susand Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and reepresentations for text classification. In *Seventh International Conference on Information and Knowledge Management*, 1998.

[Ferguson, 1973] T. S. Ferguson. A bayesian analysis of some nonparametric problems. *Annals of Statistics*, pages 209–230, 1973.

[Freund and Schapire, 1996] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[Freund and Schapire, 1999] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[Frietag and McCallum, 1999] Dayne Frietag and Andrew McCallum. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI'99 Workshop on Machine Learning for Information Extraction*, 1999.

[Ghani, 2000] Rayid Ghani. Using error-correcting codes for text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[Guruswami and Sahal, 1999] Venkatesan Guruswami and Amit Sahal. Multiclass learning, boosting and error-correcting codes. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

[Joachims, 1997] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical report, University of Dortmund, Computer Science Department, 1997.

[Lang, 1995a] Ken Lang. 20 newsgroups. http://www.ai.mit.edu/people/jrennie/20Newsgroups, 1995.

[Lang, 1995b] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[Lewis, 1998] David D. Lewis. Naive (bayes) at forty: the independence assumption in information retrieval. In *Proceedings of the Tenth European Conference on Machine Learning*, 1998.

[McCallum and Nigam, 1998] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI-98 workshop on Learning for Text Categorization*, 1998.

[McCallum, 1996] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

[Mitchell, 1997] Tom Mitchell. *Machine Learning.* McGraw-Hill Companies, Inc., 1997.

[Nigam, 2000] Kamal Nigam. Industry sector data. http://www.cs.cmu.edu/∼TextLearning/datasets.html, 2000.

[Rifkin, 2000] Ryan Rifkin. Svmfu. http://five-percent-nation.mit.edu/SvmFu/, 2000.

[Ristad, 1995] Eric Svwn Ristad. A natural law of succession. Technical Report CS-TR-495-95, Princeton University, 1995.

[Slonim and Tishby, 1999] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *Neural Information Processing Systems 12 (NIPS-99)*, 1999.

[Vapnik, 1995] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[Yang and Pedersen, 1997] Yiming Yang and J. O. Pedersen. A comparitive study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.