Using Asymmetric Distributions to Improve Classifier Probabilities: A Comparison of New and Standard Parametric Methods

Paul N. Bennett

April 9, 2002
CMU-CS-02-126

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

For many discriminative classifiers, it is desirable to convert an unnormalized confidence score output from the classifier to a normalized probability estimate. Such a method can also be used for creating *better* estimates from a probabilistic classifier that outputs poor estimates. Typical parametric methods have an underlying assumption that the score distribution for a class is symmetric; we motivate why this assumption is undesirable, especially when the scores are output by a classifier. Two asymmetric families, an asymmetric generalization of a Gaussian and a Laplace distribution, are presented, and a method of fitting them in expected linear time is described. Finally, an experimental analysis of parametric fits to the outputs of two text classifiers, naïve Bayes (which is known to emit poor probabilities) and a linear SVM, is conducted. The analysis shows that one of these asymmetric families is theoretically attractive (introducing few new parameters while increasing flexibility), computationally efficient, and empirically preferable.

Email: pbennett+@cs.cmu.edu

Keywords: calibration, well-calibrated, reliability, posterior, text classification, cost-sensitive learning, active learning, post-processing, probability estimates

1 Introduction

Classifiers that give probability estimates are more flexible in practice than those that give only a simple classification or even a ranking. Probability estimates can be used in a Bayesian risk model (Duda et al., 2001) to make cost-sensitive decisions (Zadrozny & Elkan, 2001), for combining decisions (Bourlard & Morgan, 1990), and for active learning (Lewis & Gale, 1994; Saar-Tsechansky & Provost, 2001). However, a probability estimate must have stronger constraints than simply falling in the interval $[0, 1]$ to be useful. They must be "good" in some sense.

Calibration formalizes the concept that probabilities emitted by a classifier adhere to a fixed standard. A classifier is said to be *well-calibrated* if as the number of predictions goes to infinity the predicted probability goes to the empirical probability (DeGroot & Fienberg, 1983). Occasionally "calibration" is used loosely in the literature to indicate a method generates good probability estimates (see *Performance Measures* below).

Focus on improving probability estimates has been growing in the machine learning literature. Zadrozny and Elkan (2001) provide a corrective measure for decision trees (termed *curtailment*) and a non-parametric method for recalibrating naïve Bayes. Our work provides parametric methods applicable to naïve Bayes which complement the non-parametric methods they propose when data scarcity is an issue. In addition, their non-parametric methods reduce the resolution of the scores output by the classifier, but the methods here do not have such a weakness since they are continuous functions.

There is a variety of other work that this paper extends. Lewis and Gale (1994) use logistic regression to recalibrate naïve Bayes though the quality of the probability estimates are not directly evaluated; they are simply used in active learning. Platt (1999) uses a logistic regression framework that models noisy class labels to produce probabilities from the raw output of an SVM. His work showed that this post-processing method not only can produce probability estimates of similar quality to regularized likelihood kernel methods, but it also tends to produce sparser kernels. Finally, Bennett (2000) obtained moderate gains by applying Platt's method to the recalibration of naïve Bayes but also found there were more problematic areas than when this method was applied to SVMs.

Recalibrating poorly calibrated classifiers is not a new problem. Lindley et al. (1979) first proposed the idea of recalibrating classifiers, and DeGroot and Fienberg (1983; 1986) gave the now accepted standard formalization for the problem of assessing calibration initiated by others (Brier, 1950; Winkler, 1969).

2 Problem Definition & Approach

2.1 Problem Definition

The general problem we are concerned with is highlighted in figure 1. A classifier produces a prediction about a datapoint and gives some score $s(d)$ indicating the strength of its decision that the datapoint belongs to the *positive* class. We assume throughout there are only two classes: the positive and the negative class ('+' and '-' respectively). [1]

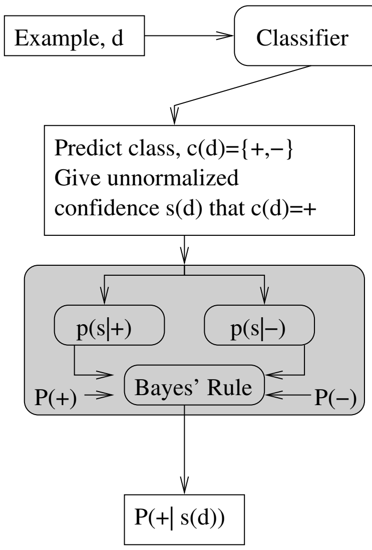

Figure 1: We are concerned with how to perform the box highlighted in grey. The internals are for one type of approach.

Since we are concerned with using methods that will also work acceptably when there is little data, we focus on parametric methods. There are two general types of parametric approaches. The first of these tries to fit the posterior function directly, *i.e.* there is one function estimator that performs a direct mapping of the score s to the probability $P(+|s(d))$. The second type of approach breaks the problem down as shown in the grey box of figure 1. An estimator for each of the class-conditional densities (*i.e.* $p(s|+)$ and $p(s|-)$) is produced, then Bayes' rule and the class priors are used to obtain the estimate for $P(+|s(d))$.

[1] When the original n classes are mutually exclusive, the binary classifiers' predictions must be combined into one final prediction (and the separate probability estimates must be normalized). In the experiments below, we deal only with the case when the original n classes are not mutually exclusive (*i.e.* an example may belong to more than one class).

2.2 Motivation for Asymmetric Distributions

Most of the previous parametric approaches to this problem[2] either directly or indirectly (when fitting only the posterior) correspond to fitting Gaussians to the class-conditional densities; they differ only in the criterion used to estimate the parameters. We can visualize this as depicted in figure 2. Since increasing s usually indicates (when the classifier has good accuracy) increased likelihood of belonging to the positive class, then the rightmost distribution usually corresponds to $p(s|+)$.

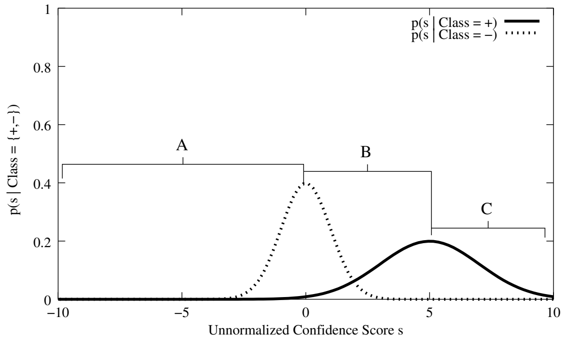

Figure 2: Typical View of Class Discrimination based on Gaussians

However, using standard Gaussians fails to capitalize on a basic characteristic commonly seen. Namely, if we have a raw output score that can be used for discrimination, then the empirical behavior between the modes (label B in figure 2) is often very different than that outside of the modes (labels A and C in figure 2). Intuitively, the area between the modes corresponds to the *hard* examples, which are difficult for this raw output score to distinguish, while the areas outside the modes are the extreme examples that are usually easily distinguished. This suggests that we may want to uncouple the scale of the outside and inside segments of the distribution (as depicted in figure 3).

As a result, an asymmetric distribution may be a more appropriate choice for application to the raw output score of a classifier. Note that the asymmetric distributions depicted in figure 3 are able to place the estimated mode much more closely to the true mode because it can separately allocate its outside and inside mass; whereas the symmetric form shifts the mode toward the long tail of the outside mass.

Ideally (*i.e.* perfect classification) there will be some scores θ_- and θ_+ such that all examples with score greater than θ_+ are positive and all examples with scores less then θ_- are negative. Furthermore, no examples fall between θ_- and θ_+. The distance $\mid \theta_- - \theta_+ \mid$ corresponds to the margin in some classifiers, and an attempt is often made to maximize this quantity. Perfect classification corresponds to using two very asymmetric distributions, but in this case, the probabilities are actually one and zero and many methods will work for typical purposes.

[2] A notable exception is (Manmatha et al., 2001) which uses a mixture model.

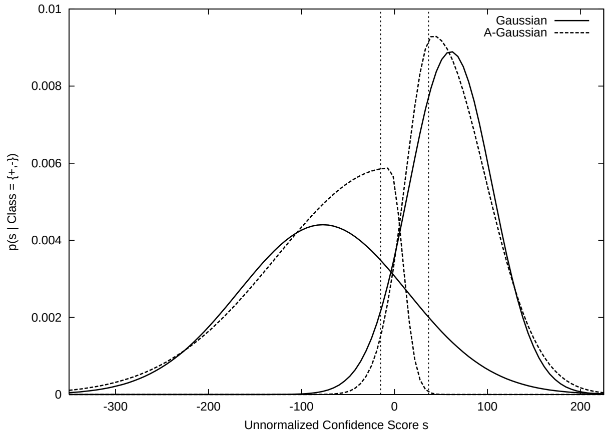

Figure 3: Gaussians vs. Asymmetric Gaussians. A Shortcoming of Symmetric Distributions — The vertical lines show the modes as estimated nonparametrically.

Practically, some examples will fall between θ_- and θ_+, and it is often important to estimate the probabilities of these examples well (since they correspond to the "hard" examples). Justifications can be given for both why you may find more and less examples between θ_- and θ_+ than outside of them, but there are few empirical reasons to believe that the distributions should be symmetric.

A natural first candidate for an asymmetric distribution is to generalize a common symmetric distribution, e.g. the Laplace or the Gaussian. An asymmetric Laplace distribution can be achieved by placing two exponentials around the mode in the following manner:

$$p(x \mid \theta, \beta, \gamma) = \begin{cases} \frac{\beta\gamma}{\beta+\gamma} \exp\left[-\beta\left(\theta - x\right)\right] & x \leq \theta \\ & (\beta, \gamma > 0) \\ \frac{\beta\gamma}{\beta+\gamma} \exp\left[-\gamma\left(x - \theta\right)\right] & x > \theta \end{cases} \tag{1}$$

where θ, β, and γ are the model parameters. θ is the mode of the distribution, β is the inverse scale of the exponential to the left of the mode, and γ is the inverse scale of the exponential to the right of the mode. We will use the notation $\Lambda(X \mid \theta, \beta, \gamma)$ to refer to this distribution.

We can create an asymmetric Gaussian in the same manner:

$$p(x \mid \theta, \sigma_l, \sigma_r) = \begin{cases} \frac{2}{\sqrt{2\pi}(\sigma_l+\sigma_r)} \exp\left[-\frac{(x-\theta)^2}{2\sigma_l^2}\right] & x \leq \theta \\ & (\sigma_l, \sigma_r > 0) \\ \frac{2}{\sqrt{2\pi}(\sigma_l+\sigma_r)} \exp\left[-\frac{(x-\theta)^2}{2\sigma_r^2}\right] & x > \theta \end{cases} \tag{2}$$

where θ, σ_l, and σ_r are the model parameters. To refer to this asymmetric Gaussian, we use the notation $\Gamma(X \mid \theta, \sigma_l, \sigma_r)$.

These distributions allow us to fit our data with much greater flexibility at the cost of only fitting six parameters. We could instead try mixture models for each component or other extensions, but most other extensions require at least as many parameters (and can often be more computationally expensive). In addition, the motivation above should provide significant cause to believe the underlying distributions actually behave in this way. Furthermore, this family of distributions can still fit a symmetric distribution, and finally, in the empirical evaluation, evidence is presented that demonstrates this behavior.

To the author's knowledge, neither family of distributions has been previously used in machine learning. Both are termed generalizations of an Asymmetric Laplace in (Kotz et al., 2001), but we refer to them as described above to reflect the nature of how we derived them for this task.

3 Estimating the Parameters of the Asymmetric Distributions

This section develops the method for finding maximum likelihood estimates (MLE) of the parameters for the above asymmetric distributions. In order to find the MLEs, we have two choices: (1) use numerical estimation to estimate all three parameters at once (2) fix the value of θ, and estimate the other two (β and γ or σ_l and σ_r) given our choice of θ, then consider alternate values of θ. Because of the simplicity of analysis in the latter alternative, we choose this method.

3.1 Asymmetric Laplace MLEs

For $\boldsymbol{D} = \{x_1, x_2, \ldots, x_N\}$ where the x_i are *i.i.d.* and $X \sim \Lambda(X \mid \theta, \beta, \gamma)$, the likelihood is $\prod_i^N \Lambda(X \mid \theta, \beta, \gamma)$. Now, we fix θ and compute the maximum likelihood for that choice of θ. Then, we can simply consider all choices of θ and choose the one with the maximum likelihood (or equivalently the loglikelihood) over all choices of θ.

The complete derivation of the following solution is given in appendix A. We define the following values:

$$\begin{aligned} N_l &= |\{x \in \boldsymbol{D} \mid x \leq \theta\}| & N_r &= |\{x \in \boldsymbol{D} \mid x > \theta\}| \\ S_l &= \sum_{x \in \boldsymbol{D} \mid x \leq \theta} x & S_r &= \sum_{x \in \boldsymbol{D} \mid x > \theta} x \\ D_l &= N_l\theta - S_l & D_r &= S_r - N_r\theta. \end{aligned}$$

Note that D_l and D_r are the sum of the absolute differences between the x belonging to the left and right halves of the distribution (respectively) and θ. Finally the MLEs for β and γ for a fixed θ are:

$$\beta_{MLE} = \frac{N}{D_l + \sqrt{D_r D_l}} \quad \gamma_{MLE} = \frac{N}{D_r + \sqrt{D_r D_l}}. \tag{3}$$

These estimates are not wholly unexpected since we would obtain $\frac{N_l}{D_l}$ if we were to estimate β independently of γ. The elegance of the formulae is that the estimates will tend to be symmetric only insofar as the data dictate it (i.e. the closer D_l and D_r are to being equal, the closer the resulting inverse scales).

By continuity arguments, when $N = 0$, we assign $\beta = \gamma = \epsilon_0$ where ϵ_0 is a small constant that acts to disperse the distribution to a uniform. Similarly, when $N \neq 0$ and $D_l = 0$, we assign $\beta = \epsilon_{\text{inf}}$ where ϵ_{inf} is a very large constant that corresponds to an extremely sharp distribution (*i.e.* almost all mass at θ for that half). $D_r = 0$ is handled similarly.

Assuming that θ falls in some range $[\phi, \psi]$ dependent upon only the observed datapoints, then this alternative is also easily computable. Given N_l, S_l, N_r, S_r, we can compute the posterior and the MLEs in constant time. In addition, if the scores are sorted, then we can perform the whole process quite efficiently. Starting with the minimum $\theta = \phi$ we would like to try, we loop through the scores once and set N_l, S_l, N_r, S_r appropriately. Then we increase θ and just step past the scores that have shifted from the right side of the distribution to the left. Assuming the number of candidate θs are $O(n)$, this process is $O(n)$, and the overall process is dominated by sorting the scores, $O(n \log n)$ (or expected linear time). Simple C code implementing this algorithm is given in appendix B.

There is no need to let θ_+ be less than θ_- for this problem. Enforcing this makes estimating the parameters for both distributions expected time $O(N_+ N_-)$. When enforcing this, one can easily make the additional constraint that if there are ties (generally unlikely), prefer the estimate with higher value for $\theta_+ - \theta_-$. However, enforcing these constraints is rarely needed in practice (since classifiers are attempting to separate the data); in addition, it is usually preferable to represent the fact that the classifier score is reversed (*i.e.* lower scores tend to mean membership in positive class).

3.2 Asymmetric Gaussian MLEs

For $\boldsymbol{D} = \{x_1, x_2, \ldots, x_N\}$ where the x_i are *i.i.d.* and $X \sim \Gamma(X \mid \theta, \sigma_l, \sigma_r)$, the likelihood is $\prod_i^N \Gamma(X \mid \theta, \beta, \gamma)$. The MLEs can be worked out similar to the above.

We assume the same definitions as above (the complete derivation is given in appendix C), and in addition, let:

$$S_{l^2} = \sum_{x \in \boldsymbol{D} | x \leq \theta} x^2 \qquad S_{r^2} = \sum_{x \in \boldsymbol{D} | x > \theta} x^2$$

$$D_{l^2} = S_{l^2} - S_l\theta + \theta^2 N_l \qquad D_{r^2} = S_{r^2} - S_r\theta + \theta^2 N_r.$$

The analytical solution for the maximum likelihood estimates for a fixed θ is:

$$\sigma_{l,MLE} = \sqrt{\frac{D_{l^2} + D_{l^2}^{2/3} D_{r^2}^{1/3}}{N}} \qquad (4)$$

$$\sigma_{r,MLE} = \sqrt{\frac{D_{r^2} + D_{r^2}^{2/3} D_{l^2}^{1/3}}{N}}. \qquad (5)$$

By continuity arguments, when $N = 0$, we assign $\sigma_r = \sigma_l = \epsilon_{\text{inf}}$, and when $N \neq 0$ and $D_{l^2} = 0$ (resp. $D_{r^2} = 0$), we assign $\sigma_l = \epsilon_0$ (resp. $\sigma_r = \epsilon_0$).

Again, the same computational complexity analysis applies to estimating these parameters. Appendix D gives C code implementing this algorithm.

4 Experimental Analysis

4.1 Methods

For each of the methods that use a class prior, we use a smoothed add-one estimate, i.e. $P(c) = \frac{|c|+1}{N+2}$ where N is the number of datapoints. For methods that fit the class-conditional densities, $p(s|+)$ and $p(s|-)$, the resulting densities are inverted using Bayes' rule as described above.

For recalibrating a classifier (*i.e.* correcting poor probability estimates output by the classifier), it is usual to use the log-odds of the classifier's estimate as $s(d)$. The log-odds are defined to be $\log \frac{P(+|d)}{P(-|d)}$. The normal decision threshold (minimizing error) in terms of log-odds is at zero (*i.e.* $P(+|d) = P(-|d) = 0.5$).

As discussed in (Lindley et al., 1979), the log-odds is useful since it scales the outputs to a space $[-\infty, \infty]$ where normal (and similar distributions) are applicable. Lewis and Gale (1994) give a more motivating viewpoint that we can see fitting the log-odds as a dampening effect (correcting for the inaccurate independence assumption) and a bias correction (for possibly inaccurate estimates for the priors). We note that in general fitting the log-odds can serve to boost or dampen the signal from the original classifier as the data dictate.

4.1.1 Gaussians

A Gaussian is fit to each of the class-conditional densities, using the usual maximum likelihood estimates. This method is denoted in the tables below as *Gauss*.

4.1.2 Asymmetric Gaussians

An asymmetric Gaussian is fit to each of the class-conditional densities using the maximum likelihood estimation procedure described above. Intervals between adjacent scores are divided by 10 in testing candidate θs, *i.e.* 8 points between actual scores occurring in the data set are tested. This method is denoted as *A. Gauss* below.

4.1.3 Laplace Distributions

Even though Laplace distributions are not typically applied to this task, we also tried this method to isolate why benefit is gained from the asymmetric form. The usual MLEs were used for estimating the location and scale of a classical symmetric Laplace distribution as described in Kotz et al. (2001). We denote this method as *Laplace* below.

4.1.4 Asymmetric Laplace Distributions

An asymmetric Laplace is fit to each of the class-conditional densities using the maximum likelihood estimation procedure described above. As with the asymmetric Gaussian, intervals between adjacent scores are divided by 10 in testing candidate θs. This method is denoted as *A. Laplace* below.

4.1.5 Logistic Regression

This method is the first of two methods we evaluated that directly fit the posterior, $P(+|s(d))$. Both methods restrict the set of families to a two-parameter sigmoid family; they differ primarily in their model of class labels. As opposed to the above methods, one can argue that an additional boon of these methods is they completely preserve the ranking given by the classifier. When this is desired, these methods may be more appropriate. The previous methods will mostly preserve the rankings, but they can deviate if the data dictate it. Thus, they may model the data behavior better at the cost of departing from a monotonicity constraint in the output of the classifier.

Lewis and Gale (1994) use logistic regression to recalibrate naïve Bayes for subsequent use in active learning. The model they use is:

$$P(+|s(d)) = \frac{\exp(a + b\, s(d))}{1 + \exp(a + b\, s(d))}. \qquad (6)$$

Instead of using the probabilities directly output by the classifier, they use the loglikelihood ratio of the probabilities, $\log \frac{P(d|+)}{P(d|-)}$, as the score $s(d)$. Instead of using this below, we will use the log-odds ratio. This does not effect the model as it simply shifts all of the scores by a constant determined by the priors.

We refer to this method as *LogReg* below.

4.1.6 Logistic Regression with Noisy Class Labels

Platt (1999) proposes a framework that extends the logistic regression model above to incorporate noisy class labels and uses it to produce probability estimates from the raw output of an SVM classifier.

This model differs from the *LogReg* model only in how the parameters are estimated. The parameters are still fit using maximum likelihood estimation, but a model of noisy class labels is used in addition to allow for the possibility that the class was mislabeled. The noise is modeled by assuming there is a finite probability of mislabeling a positive example and of mislabeling a negative example; these two noise estimates are determined by the number of positive examples and the number of negative examples (using Bayes' rule to infer the probability of incorrect label).

Even though the performance of this model would not be expected to deviate much from *LogReg*, we evaluate it for completeness. We refer to this method below as *LR+Noise*.

4.2 Data

We examined the above methods on several "real world" text corpora, including the *MSN Web Directory, Reuters*, and *TREC-AP* data sets. Since the categories under consideration in the experiments are not mutually exclusive, the classification was done by training n binary classifiers, where n is the number of classes.

4.2.1 MSN Web Directory

The MSN Web Directory is a large collection of heterogeneous web pages (from a May 1999 web snapshot) that have been hierarchically classified. We used the same train/test split of 50078/10024 documents as that reported in Dumais and Chen (2000).

The MSN Web hierarchy is a 7-level hierarchy, but we have restricted our analysis to the 13 top-level categories. The class proportions in the training set vary from 1.15% to 22.29%. In the testing set, they range from 1.14% to 21.54%. The classes are general subject categories such as *Health & Fitness* and *Travel & Vacation*. Human indexers assign the documents to zero or more categories. There are approximately 130K binary decisions made over the test documents (*i.e.* 13 classes times 10024 test documents).

For the experiments below, only the top 1000 words with highest mutual information for each class were used (Duda et al., 2001); approximately 195K words appear in at least 3 training documents (those occurring in less than 3 were removed).

4.2.2 Reuters

The Reuters 21578 corpus (Lewis, 1997) contains Reuters news articles from 1987. For this data set, we used the ModApte standard train/test split of 9603/3299 documents (8676 unused documents). The classes are economic subjects (*e.g.*, "acq" for acquisitions, "earn" for earnings, etc.) that human indexers decided applied to the document; a document may have multiple subjects. There are actually 135 classes in this domain (only 90 of which occur in the training and testing set); however, we only examined the 10 most frequent classes (similar to (Dumais et al., 1998; Joachims, 1998; McCallum & Nigam, 1998; Platt, 1999)) as we believed we had a significant enough variation of class frequency over all the corpora used.[3]

The class proportions in the training set vary from 1.88% to 29.96%. In the testing set, they range from 1.7% to 32.95%. There are approximately 33K binary decisions to be made over the test set.

For the experiments below we used only the top 300 words with highest mutual information for each class; approximately 15K words appear in at least 3 training documents.

4.2.3 TREC-AP

The TREC-AP corpus is a collection of AP news stories from 1988 to 1990. We used the same train/test split of 142791/ 66992 documents that was used in (Lewis et al., 1996). As described in (Lewis & Gale, 1994) (see also (Lewis, 1995)), the categories are defined by keywords in a keyword field. The title and body fields are used in the experiments below.

The frequencies of the 20 classes are the same as those reported in (Lewis et al., 1996). The class proportions in the training set vary from 0.06% to 2.03%. In the testing set, they range from 0.03% to 4.32%. There are approximately 1.3M binary decisions to be made over the test set.

For the experiments described below, we use only the top 1000 words with the highest mutual information for each class; approximately 123K words appear in at least 3 training documents.

4.3 Classifiers

We selected two classifiers for evaluation. A linear SVM classifier which is a discriminative classifier that does not

[3]A separate comparison over all 90 categories in a slightly non-standard version of Reuters (Yang & Liu, 1999) was conducted that compared *LogReg*, *LR+Noise*, and *A. Laplace*. That evaluation also supported the claims made here.

normally output probability values, and a naïve Bayes classifier whose probability outputs are typically poor (Bennett, 2000; Domingos & Pazzani, 1996) but can be improved (Bennett, 2000; Zadrozny & Elkan, 2001).

4.3.1 SVM

For linear SVMs, we use the *Smox* toolkit which is based on Platt's **S**equential **M**inimal **O**ptimization algorithm. The features were represented as continuous values. We used the raw output score of the SVM as $s(d)$ since it has been shown to be appropriate before (Platt, 1999). The normal decision threshold (assuming we are seeking to minimize errors) for this classifier is at zero.

4.3.2 Naïve Bayes

The *naïve Bayes* classifier model is a multinomial model (McCallum & Nigam, 1998). We smoothed word and class probabilities using a Bayesian estimate (with the word prior) and a Laplace m-estimate, respectively. We use the log-odds estimated by the classifier as $s(d)$. The normal decision threshold is at zero.

4.4 Performance Measures

We use log-loss (Good, 1952) and squared error (Brier, 1950; DeGroot & Fienberg, 1986) to evaluate the quality of the probability estimates. DeGroot and Fienberg (1983) show how these two scoring rules can be broken down into a sum of two terms — one corresponding to calibration (as defined above) and another to *refinement*. It is beyond the scope of this article to delve into these issues, but these scoring rules have been typically used to assess the quality of probability estimates without breaking down the score into its component parts. In the literature achieving a better score according to these rules has sometimes been loosely termed improving "calibration" but actually meaning the overall quality was improved (via improving one or both of the components).

For a datum d with class $c(d) \in \{+, -\}$ (*i.e.* the data have known labels and not probability values), log-loss is defined as $\delta(c(d), +) \log P(+|d) + \delta(c(d), -) \log P(-|d)$ where δ is the Kronecker delta function. The squared error is $\delta(c(d), +)(1 - P(+|d))^2 + \delta(c(d), -)(1 - P(-|d))^2$.

We report the sum of these measures as well as their averages, average log-loss and mean squared error (MSE).

In addition, we also compare the error rate of the classifiers at their default thresholds and with the probabilities. This gives an idea of how the probability estimates have improved with respect to the decision threshold $P(+|d) = 0.5$.[4]

[4] We note that this measure should not be used alone when judging the quality of a probability distribution since this measure only indicates whether the estimates tend to the correct side of $P(+|d) = 0.5$.

We use a a standard paired sign-test (Yang & Liu, 1999) to determine statistical significance in the difference of all measures. Only pairs that the methods disagree on are used in the sign test. This test compares pairs of scores from two systems with the null hypothesis that the number of items they disagree on are binomially distributed (*i.e.* each system does better about half the time they disagree). We use a significance level of $p = 0.01$.

4.5 Experimental Methodology

In order to generate the scores that each method uses to fit its probability estimates, we use five-fold cross-validation on the training data. We note that even though it is computationally efficient to perform leave-one-out cross-validation for the naïve Bayes classifier, this may not be desirable in all cases since the distribution of scores can be biased as a result (*i.e.* the class-conditional densities might show a larger separation than would be expected in held-out data). Of course, as with any application of n-fold cross-validation, it is also possible to bias the results by holding n too low and underestimating the performance of the final classifier.

4.6 Results

The results for recalibrating naïve Bayes are given on the *left* of tables 1 and 2. For producing probabilistic outputs for SVMs, the results are given on the *right* of tables 1 and 2.

We can break things down as the sign test does and just look at wins and losses on the items that the methods disagree on. Looked at in this way only two methods (*naïve Bayes* and *A. Gauss*) ever have more pairwise wins than *A. Laplace*; those two sometimes have more pairwise wins on log-loss and squared error even though the total never wins (*i.e.* they are dragged down by heavy penalties). The reasons for this behavior is discussed below.

In addition, this comparison of pairwise wins means that for those cases where *LogReg* and *LR+Noise* have better scores than *A. Laplace*, it would not be deemed significant by the sign test at any level since they do not have more wins. For example, of the 130K binary decisions over the MSN Web dataset, *A. Laplace* had approximately 101K pairwise wins versus *LogReg* and *LR+Noise*.

No method ever has more pairwise wins than *A. Laplace* for the error rate comparison nor does any method every achieve a better total number.

In order to give the reader a better sense of the behavior of these methods, figures 4-12 show the fits produced by these methods versus the actual data behavior (as estimated nonparametrically using a fixed width kernel) for class *Earn* in Reuters. Figures 4-6 show the class-conditional densities. Figures 7-9 show the estimations of

Table 1: Results for naïve Bayes (*left*) and SVM (*right*). The best entry for a corpus is in bold. Entries that are statistically significantly better than all other entries are underlined. A † denotes the method is significantly better than all other methods except for *naïve Bayes*. A ‡ denotes the entry is significantly better than all other methods except for *A. Gauss* (and *naïve Bayes* for the table on the left).

	Log-loss	Error^2	Errors
MSN Web			
Gauss	-60656.41	10503.30	10754
A.Gauss	-57262.26	8727.47	9675
Laplace	-45363.84	8617.59	10927
A.Laplace	-36765.88	**6407.84**†	**8350**
LogReg	-36470.99	6525.47	8540
LR+Noise	**-36468.18**	6534.61	8563
naïve Bayes	-1098900.83	17117.50	17834
Reuters			
Gauss	-5523.14	1124.17	1654
A.Gauss	-4929.12	652.67	888
Laplace	-5677.68	1157.33	1416
A.Laplace	**-3106.95**‡	**554.37**‡	**726**
LogReg	-3375.63	603.20	786
LR+Noise	-3374.15	604.80	785
naïve Bayes	-52184.52	1969.41	2121
TREC-AP			
Gauss	-57872.57	8431.89	9705
A.Gauss	-66009.43	7826.99	8865
Laplace	-61548.42	9571.29	11442
A.Laplace	-48711.55	**7251.87**‡	**8642**
LogReg	**-48250.81**	7540.60	8797
LR+Noise	-48251.51	7544.84	8801
naïve Bayes	-1903487.10	41770.21	43661

	Log-loss	Error^2	Errors
MSN Web			
Gauss	-54463.32	9090.57	10555
A. Gauss	-44363.70	6907.79	8375
Laplace	-42429.25	7669.75	10201
A. Laplace	-31133.83	**5003.32**	**6170**
LogReg	**-30209.36**	5158.74	6480
LR+Noise	-30294.01	5209.80	6551
Linear SVM	N/A	N/A	6602
Reuters			
Gauss	-3955.33	589.25	735
A. Gauss	-4580.46	428.21	532
Laplace	-3569.36	640.19	770
A. Laplace	-2599.28	412.75	**505**
LogReg	-2575.85	**407.48**	509
LR+Noise	**-2567.68**	408.82	516
Linear SVM	N/A	N/A	516
TREC-AP			
Gauss	-54620.94	6525.71	7321
A. Gauss	-77729.49	6062.64	6639
Laplace	-54543.19	7508.37	9033
A. Laplace	-48414.39	**5761.25**‡	**6572**‡
LogReg	-48285.56	5914.04	6791
LR+Noise	**-48214.96**	5919.25	6794
Linear SVM	N/A	N/A	6718

the posterior, (*i.e.* $P(Earn|s(d))$). Figures 10-12 show the estimations of the log-odds, (*i.e.* $\log \frac{P(Earn|s(d))}{P(\neg Earn|s(d))}$). The differences between *LogReg* and *LR+Noise* were not visible to the eye in these graphs; thus only one line is shown for both. In order to help the reader quantify the differences in these fits, we present a detailed breakdown of log-loss and squared error for class *Earn* in tables 3 and 4.

4.7 Discussion

We start by noting several points of interest observable in figures 4-11 and then move on to more general observations. First, the training and test distributions in figure 4 are clearly different. The training distribution in both cases (for recalibrating naïve Bayes and producing probabilistic outputs for SVMs) is harder to separate than the test distribution. There are two primary reasons why this might be the case. The first is that Reuters is a time sequence of news stories, and the train/test split is a split at one point in time. Therefore, the actual distribution might drift. The second possible explanation is that using 5-fold cross-validation might be underestimating the performance of the final classifier. While few details are observable in figures 7-9, it should give the reader a general sense of the behavior of the posterior of these functions. Finally, in figure 11, one potential benefit of the *A. Laplace* method over *LogReg* and *LR+Noise* is demonstrated. Logistic regression corresponds to a line in this space (thus the name), but this can overconstrain it at times. Whereas, *A. Laplace* corresponds to a piecewise linear function of three line segments (the hinges occur at the modes of the class-conditional densities). This allows *A. Laplace* to find a better fit in these cases.

Several more general observations come from examining the performance of these methods over the various corpora. The first is that *A. Laplace*, *LR+Noise*, and *LogReg*, quite clearly outperform the other methods. There is usually little difference between the performance of *LR+Noise* and *LogReg* (both as shown here and on a decision by decision basis), but this is unsurprising since *LR+Noise* just

Table 2: Averages for calibrating naïve Bayes (*left*) and SVM (*right*). The best entry for a corpus is in bold. Entries that are statistically significantly better than all other entries are underlined. A † denotes the method is significantly better than all other methods except for *naïve Bayes*. A ‡ denotes the entry is significantly better than all other methods except for *A. Gauss* (and *naïve Bayes* for the table on the left).

	Avg LL	MSE	Error
MSN Web			
Gauss	-0.4655	0.0806	0.0825
A.Gauss	-0.4394	0.0670	0.0742
Laplace	-0.3481	0.0661	0.0839
A.Laplace	-0.2821	**0.0492**†	**0.0641**
LogReg	-0.2799	0.0501	0.0655
LR+Noise	**-0.2799**	0.0501	0.0657
naïve Bayes	-8.4328	0.1314	0.1369
Reuters			
Gauss	-0.1674	0.0341	0.0501
A. Gauss	-0.1494	0.0198	0.0269
Laplace	-0.1721	0.0351	0.0429
A. Laplace	**-0.0942**‡	**0.0168**‡	**0.0220**
LogReg	-0.1023	0.0183	0.0238
LR+Noise	-0.1023	0.0183	0.0238
naïve Bayes	-1.5818	0.0597	0.0643
TREC-AP			
Gauss	-0.0432	0.0063	0.0072
A. Gauss	-0.0493	0.0058	0.0066
Laplace	-0.0459	0.0071	0.0085
A. Laplace	-0.0364	**0.0054**‡	**0.0065**
LogReg	**-0.0360**	0.0056	0.0066
LR+Noise	-0.0360	0.0056	0.0066
naïve Bayes	-1.4207	0.0312	0.0326

	Avg LL	MSE	Error
MSN Web			
Gauss	-0.4179	0.0698	0.0810
A. Gauss	-0.3404	0.0530	0.0643
Laplace	-0.3256	0.0589	0.0783
A. Laplace	-0.2389	**0.0384**	**0.0473**
LogReg	**-0.2318**	0.0396	0.0497
LR+Noise	-0.2325	0.0400	0.0503
Linear SVM	N/A	N/A	0.0507
Reuters			
Gauss	-0.1199	0.0179	0.0223
A. Gauss	-0.1388	0.0130	0.0161
Laplace	-0.1082	0.0194	0.0233
A. Laplace	-0.0788	0.0125	**0.0153**
LogReg	-0.0781	**0.0124**	0.0154
LR+Noise	**-0.0778**	0.0124	0.0156
Linear SVM	N/A	N/A	0.0156
TREC-AP			
Gauss	-0.0408	0.0049	0.0055
A. Gauss	-0.0580	0.0045	0.0050
Laplace	-0.0407	0.0056	0.0067
A. Laplace	-0.0361	**0.0043**‡	**0.0049**‡
LogReg	-0.0360	0.0044	0.0051
LR+Noise	**-0.0360**	0.0044	0.0051
Linear SVM	N/A	N/A	0.0050

adds noisy class labels to the *LogReg* model. With respect to the three different measures, *LR+Noise* and *LogReg* tend to perform slightly better (but never significantly) at some tasks with respect to log-loss and squared error. However, *A. Laplace* always produces the least number of errors for all of the tasks, though at times the degree of improvement is not significant.

The basic observation made about naïve Bayes in previous work is that it tends to produce estimates very close to zero and one (Bennett, 2000; Lewis & Gale, 1994). This means if it tends to be right enough of the time, it will produce results that do not appear significant in a sign test that ignores size of difference (as the one here). The totals of the squared error and log-loss bear out the previous observation that "when it's wrong it's *really* wrong".

There are several interesting points about the performance of the asymmetric distributions as well. First, *A. Gauss* performs poorly because (similar to naïve Bayes) there are some examples where it is penalized a large amount. This behavior results from a general tendency to perform like the picture shown in figure 3 (note the crossover at the tails). While the asymmetric Gaussian tends to place the mode much more accurately than a symmetric Gaussian, its asymmetric flexibility combined with its distance function causes it to distribute too much mass to the outside tails while failing to fit around the mode accurately enough to compensate. Figure 3 is actually a result of fitting the two distributions to real data (an excerpt of figure 5). As a result, at the tails there can be a large discrepancy between the likelihood of belonging to each class. Thus when there are no outliers the *A. Gauss* can perform quite competitively, but when there is an outlier the *A. Gauss* can be penalized quite heavily for it. There are enough such cases overall that it seems clearly inferior to the top three methods.

However, the asymmetric Laplace places much more emphasis around the mode because of the different distance function (think of the "sharp peak" of an exponential). As a result most of the mass stays centered around the modes, while the asymmetric parameters still allow more flexibility

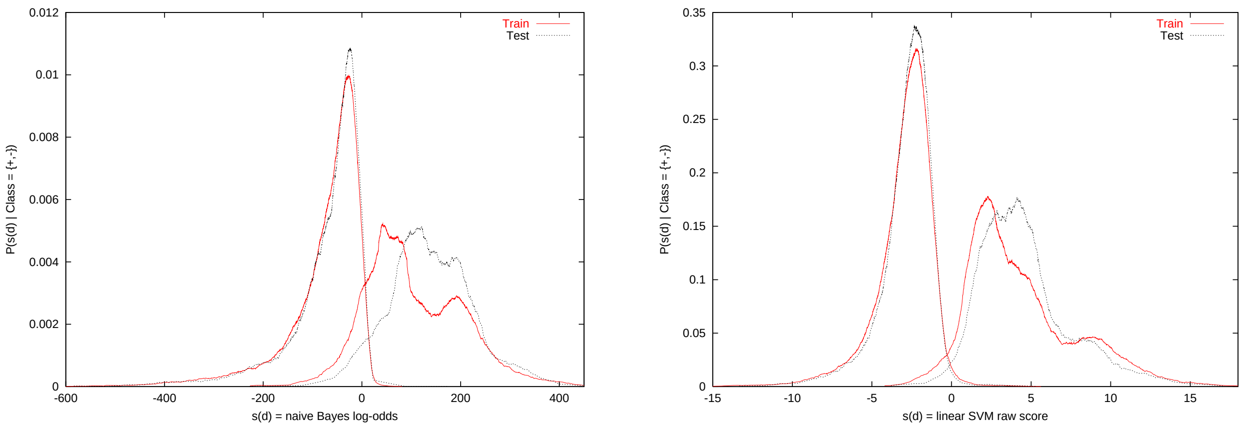

Figure 4: Nonparametric estimation (using a fixed width kernel, *i.e.* data centered bin) of class conditional score densities in the training and the test set for class *Earn* in Reuters. The *positive* class (*i.e. Earn*) is the distribution on the right in each graph, and the *negative* class (*i.e.* ¬*Earn*) is that on the left in each graph.

Table 3: Detailed results from one binary classification task for recalibrating naïve Bayes. The task is discrimination of class *Earn* in the Reuters corpus.

	Over Training		**Over Testing**	
	Error^2	MSE	Error^2	MSE
Gauss	686.45	0.0715	143.57	0.0435
A. Gauss	392.14	0.0408	78.05	0.0237
Laplace	835.05	0.0870	223.98	0.0679
A. Laplace	378.26	0.0394	70.93	0.0215
LogReg	433.44	0.0451	91.36	0.0277
LR+Noise	434.30	0.0452	91.55	0.0278

	Over Training		**Over Testing**	
	Log-loss	Avg LL	Log-loss	Avg LL
Gauss	-3501.49	-0.3636	-862.07	-0.2613
A. Gauss	-2361.48	-0.2459	-747.71	-0.2266
Laplace	-4259.04	-0.4435	-1261.26	-0.3823
A. Laplace	-2176.02	-0.2266	-492.50	-0.1493
LogReg	-2684.12	-0.2795	-595.72	-0.1806
LR+Noise	-2684.17	-0.2795	-596.39	-0.1808

than the standard Laplace in fitting the data.

We could extend the significance tests here with a Wilcox signed rank test (which is an extension of the sign test to consider size of win as well by ranking the absolute differences) (DeGroot, 1989). Though the expectation is that this would change few of the comparisons except those against *naïve Bayes* (since all of the methods post very large pairwise wins against it).

Finally, we can make a few observations about the usefulness of the various performance metrics. First, log-loss only awards a finite amount of credit as the degree to which something is correct improves (*i.e.* there are diminishing returns as it approaches zero), but it can infinitely penalize for a wrong estimate. Thus, it is possible for one outlier to skew the totals, but misclassifying this example may not matter for any but a handful of actual utility functions (ones with *extremely* high skew) used in practice. Secondly, squared error has a weakness in the other direction. That is, its penalty and reward are bounded in $[0, 1]$, but if the number of errors are small enough, it is possible for a method to appear better when it is producing what we generally consider unuseful probability estimates. For example, consider a method that only estimates probabilities as zero or one (which naïve Bayes tends to but doesn't quite reach if you use smoothing). This method could win according to squared error, but with just one error it would never perform better on log-loss than any method that assigns some non-zero probability to each outcome. For these reasons, we recommend that neither of these are used in isolation as they each give slightly different insights to the quality of the estimates produced. These observations are straightforward from the metric definitions but are underscored by the evaluation.

5 Future Work

A promising extension to the work presented here is a hybrid distribution of a Gaussian (on the outside slopes) and

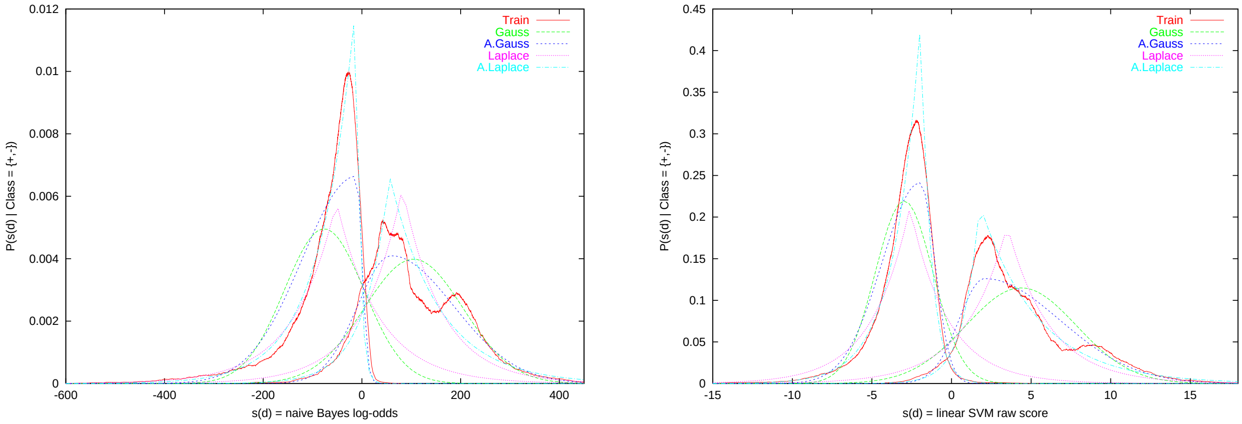

Figure 5: Estimated class conditional score densities of various methods versus the nonparametric density of the training data for class *Earn* in Reuters.

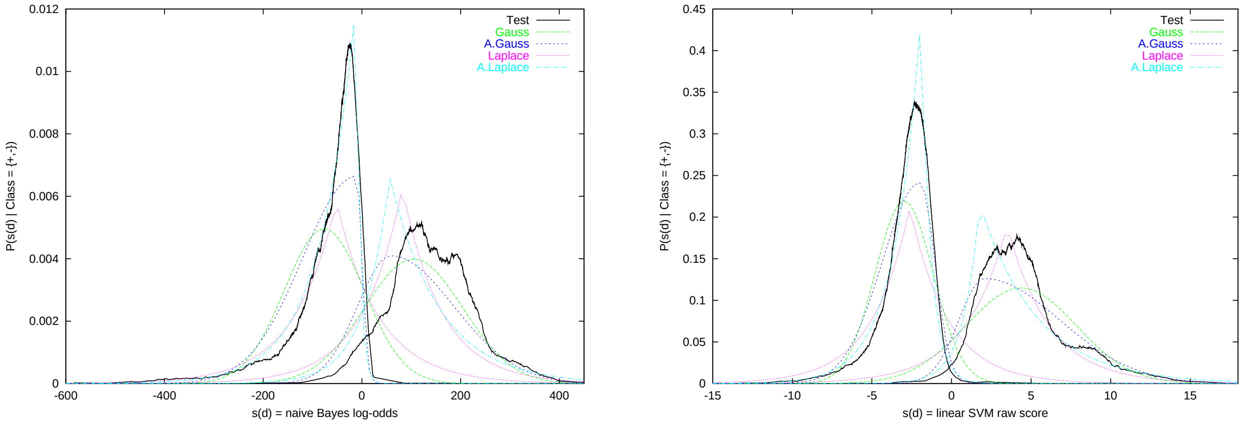

Figure 6: Estimated class conditional score densities of various methods versus the nonparametric density of the testing data for class *Earn* in Reuters.

exponentials (on the inner slopes). From the empirical evidence presented in (Platt, 1999), the expectation is that such a distribution might allow more emphasis of the probability mass around the modes (as with the exponential) while still providing more accurate estimates toward the tails. Comparing it to a mixture model, such as that used in (Manmatha et al., 2001) for combining search engine output, may also provide useful insights.

Finally, extending these methods to the outputs of other discriminative classifiers is an open area. We are currently evaluating the appropriateness of these methods for the output of a *voted perceptron* (Freund & Schapire, 1999). By analogy to the log-odds, the operative score that appears promising is $\log \frac{\text{weight perceptrons voting } +}{\text{weight perceptrons voting } -}$.

6 Summary and Conclusions

We have reviewed a wide variety of parametric methods for producing probability estimates from the raw scores of a discriminative classifier and for recalibrating an uncalibrated probabilistic classifier. In addition, we have introduced two new families that attempt to capitalize on the asymmetric behavior that tends to arise from learning a discrimination function. We have given an efficient way to estimate the parameters of these distributions.

While these distributions attempt to strike a balance between the generalization power of parametric distributions and the flexibility that the added asymmetric parameters give, the asymmetric Gaussian appears to have too great of an emphasis away from the modes. In striking con-

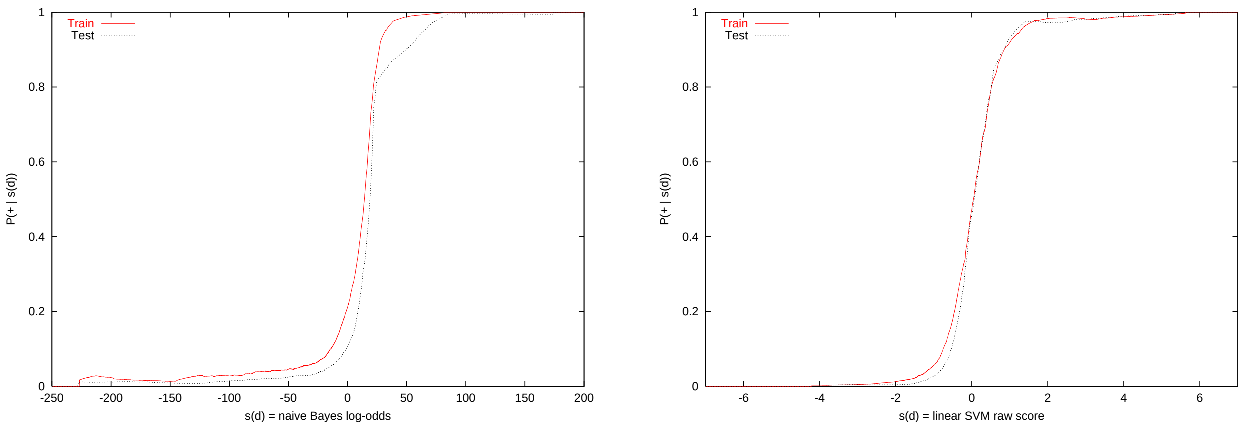

Figure 7: Nonparametric estimation of posterior (using Bayes' rule to invert densities in figure 4) in the training and the test set for class *Earn* in Reuters.

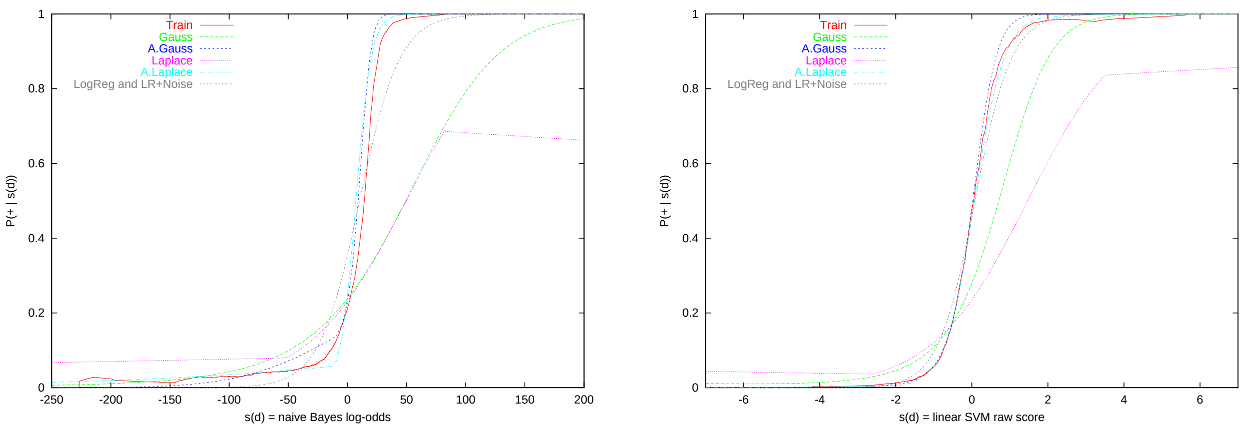

Figure 8: Estimated posteriors of various methods versus the nonparametric estimation of the posterior of the training data for class *Earn* in Reuters.

trast, the asymmetric Laplace distribution appears to be preferable over several large text domains and a variety of performance measures to the primary competing parametric methods, though comparable performance is sometimes achieved with one of two varieties of logistic regression. Given the ease of estimating the parameters of this distribution, it is a good first choice for producing quality probability estimates.

Acknowledgements

We are grateful to Francisco Pereira for the sign test code, Anton Likhodedov for logistic regression code, and John Platt for the code support for the linear SVM classifier toolkit *Smox*. Also, we sincerely thank Chris Meek and John Platt for the very useful advice provided in the early stages of this work. Thanks also to Jaime Carbonell and John Lafferty for their useful feedback on the final versions of this paper.

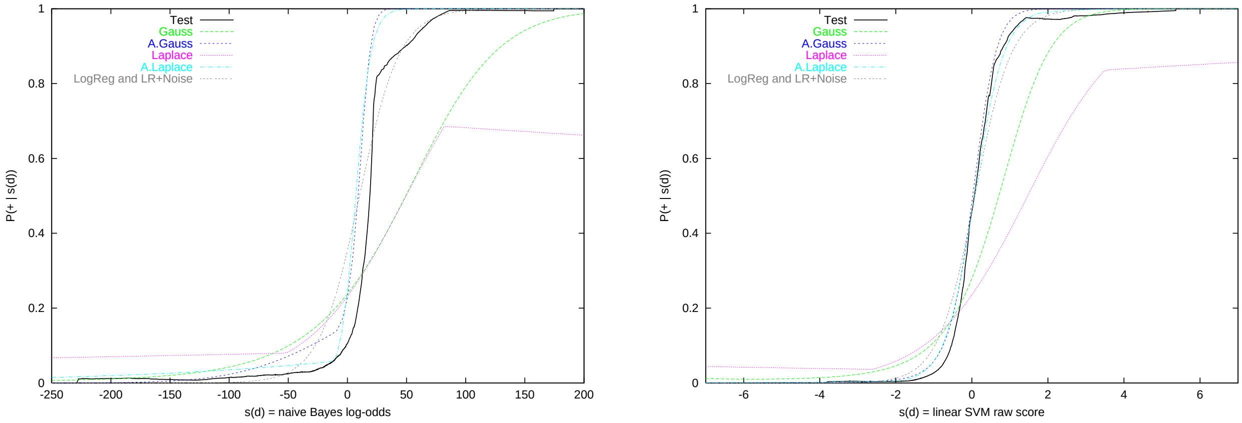

Figure 9: Estimated posteriors of various methods versus the nonparametric estimation of the posterior of the testing data for class *Earn* in Reuters.

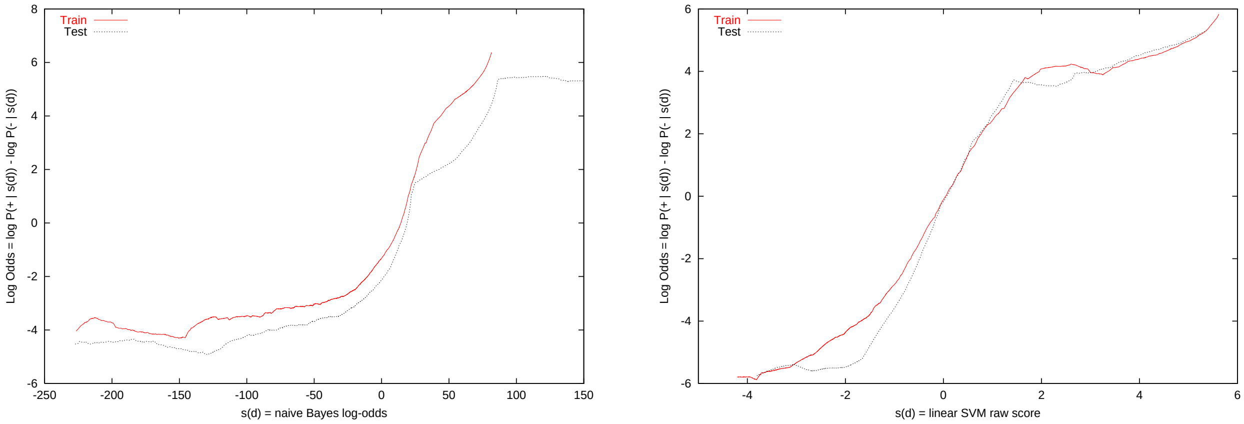

Figure 10: Nonparametric estimation of log-odds in the training and the test set for class *Earn* in Reuters.

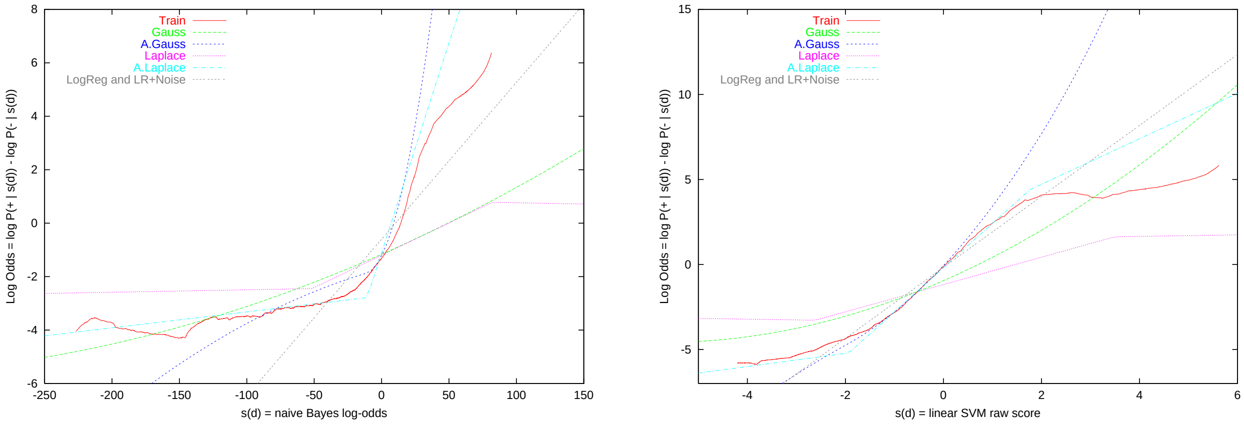

Figure 11: Estimated log-odds of various methods versus the nonparametric estimation of the log-odds of the training data for class *Earn* in Reuters.

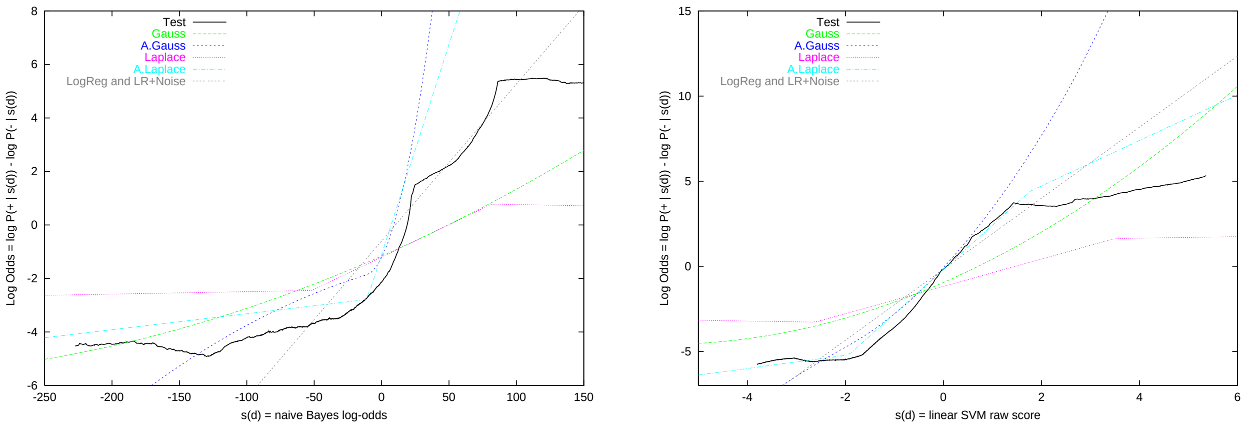

Figure 12: Estimated log-odds of various methods versus the nonparametric estimation of the log-odds of the testing data for class *Earn* in Reuters.

Table 4: Detailed results from one binary classification task for producing probabilistic outputs for an SVM. The task is discrimination of class *Earn* in the Reuters corpus.

	Over Training		**Over Testing**	
	Error^2	MSE	Error^2	MSE
Gauss	231.42	0.0241	52.82	0.0160
A. Gauss	165.27	0.0172	37.76	0.0114
Laplace	405.15	0.0422	108.95	0.0330
A. Laplace	164.72	0.0172	37.59	0.0114
LogReg	165.41	0.0172	38.61	0.0117
LR+Noise	165.66	0.0173	38.71	0.0117

	Over Training		**Over Testing**	
	Log-loss	Avg LL	Log-loss	Avg LL
Gauss	-1462.49	-0.1523	-384.73	-0.1166
A. Gauss	-1195.92	-0.1245	-393.64	-0.1193
Laplace	-2368.40	-0.2466	-717.30	-0.2174
A. Laplace	-1015.49	-0.1057	-268.56	-0.0814
LogReg	-1034.93	-0.1078	-292.21	-0.0886
LR+Noise	-1035.07	-0.1078	-292.06	-0.0885

Appendix A: Derivation of MLEs for Asymmetric Laplace Distribution

For $\boldsymbol{D} = \{x_1, x_2, \ldots, x_N\}$ where the x_i are *i.i.d.* and $X \sim \Lambda(X \mid \theta, \beta, \gamma)$, the likelihood is:

$$\prod_{i}^{N} \Lambda(X \mid \theta, \beta, \gamma). \tag{7}$$

We desire to find the maximum likelihood estimates for β, γ and θ. To do so, we fix θ and compute the maximum likelihood for that choice of θ. Then, we can simply consider all choices of θ and choose the one with the maximum likelihood (or equivalently the loglikelihood) over all choices of θ.

The loglikelihood we must compute then is:

$$\begin{aligned}
\log \prod_{i=1}^{N} \Lambda(x_i \mid \theta, \beta, \gamma) &= \sum_{i=1}^{N} \log \Lambda(x_i \mid \theta, \beta, \gamma) && (8)\\
&= \sum_{x \in \boldsymbol{D} | x \leq \theta} \log \Lambda(x_i \mid \theta, \beta, \gamma) + \sum_{x \in \boldsymbol{D} | x > \theta} \log \Lambda(x_i \mid \theta, \beta, \gamma) && (9)\\
&= \sum_{x \in \boldsymbol{D} | x \leq \theta} \left[\log \frac{\beta\gamma}{\beta+\gamma} - \beta(\theta - x)\right] + \sum_{x \in \boldsymbol{D} | x > \theta} \left[\log \frac{\beta\gamma}{\beta+\gamma} - \gamma(x - \theta)\right] && (10)\\
&= N \log \frac{\beta\gamma}{\beta+\gamma} + \sum_{x \in \boldsymbol{D} | x \leq \theta} [-\beta(\theta - x)] + \sum_{x \in \boldsymbol{D} | x > \theta} [-\gamma(x - \theta)] && (11)
\end{aligned}$$

Let $N_l = |\ \{x \in \boldsymbol{D} \mid x \leq \theta\}\ |\ ,\ N_r = |\ \{x \in \boldsymbol{D} \mid x > \theta\}\ |\ ,\ S_l = \sum_{x \in \boldsymbol{D} | x \leq \theta} x,\ S_r = \sum_{x \in \boldsymbol{D} | x > \theta} x$

$$= N \log \frac{\beta\gamma}{\beta+\gamma} - N_l\beta\theta + \beta S_l + N_r\gamma\theta - \gamma S_r \tag{12}$$

Let $D_l = N_l\theta - S_l$, $D_r = S_r - N_r\theta$

$$= N \log \frac{\beta\gamma}{\beta+\gamma} - \beta D_l - \gamma D_r \tag{13}$$

The partial derivatives are: $\frac{\partial\beta}{\partial l} = \frac{N\gamma}{\beta(\beta+\gamma)} - D_l$ and $\frac{\partial\gamma}{\partial l} = \frac{N\beta}{\gamma(\beta+\gamma)} - D_r$. We can set the derivatives to zero and solve them analytically to find for a fixed θ:

$$\beta_{MLE} = \frac{N}{D_l + \sqrt{D_r D_l}} \qquad\qquad \gamma_{MLE} = \frac{N}{D_r + \sqrt{D_r D_l}}. \tag{14}$$

We then can iterate through alternate choices for θ.

For comparison of the symmetry of this solution to the asymmetric Gaussian, we give the scale parameters (i.e. inverses of β and γ) as follows:

$$\beta^{-1}_{MLE} = \frac{D_l^{1/2}}{N^{1/2}} \cdot \frac{D_l^{1/2} + D_r^{1/2}}{N^{1/2}} \qquad\qquad \gamma^{-1}_{MLE} = \frac{D_r^{1/2}}{N^{1/2}} \cdot \frac{D_l^{1/2} + D_r^{1/2}}{N^{1/2}}. \tag{15}$$

The second part of each equation is equal to $(\beta^{-1}_{MLE} + \gamma^{-1}_{MLE})^{1/2}$.

Appendix B: C code for MLE of Asymmetric Laplace

This code is also currently available at http://www.cs.cmu.edu/˜pbennett/asymmetric/aLaplace.c .

```
/* For the data given in scores,
   finds the maximum likelihood set of parameters for
   an asymmetric Laplace family

   p(x | THETA, BETA, GAMMA) =
        if (x <= THETA) then
              (BETA * GAMMA)/(BETA + GAMMA) exp[- BETA (THETA - X) ]
        else

              (BETA * GAMMA)/(BETA + GAMMA) exp[- GAMMA (X - THETA) ]

   candidate thetas are restricted to the range [min,max]
*/
void find_aLaplace_parameters(double * scores, /* vector of scores - assumed to
                                                  be sorted least to greatest */
                              long int N, /* number of scores */
                              double min, /* min mode to try */
                              double max, /* max mode to try */
                              double defaultm, /* default mode if N == 0 */
                              int num_interval_slices, /* breaks scores[i] and
                                                          scores[i + 1] into
                                                          this many pieces to
                                                          try as candidate
                                                          modes */
                              /* next three are return values of distribution */
                              double * final_theta,
                              double * final_beta,
                              double * final_gamma) {

  double max_ln_posterior, ln_posterior;
  double max_init = 0;
  double prev_score_theta = min;
  double theta = prev_score_theta, beta, gamma;
  long int num_left = 0, num_right = N;
  double sum_left = 0, sum_right = 0;
  double diff_left, diff_right;
  int done = 0, slice_num = 0;
  long int i;

  if (N == 0) {
    /* no scores */
    *final_theta = defaultm;
    *final_beta = EPSILON_ZERO;
    *final_gamma = EPSILON_ZERO;
    return;
  }

  /* loop through and get the sum of all scores */
  for (i = 0; i < N; i++) {
    sum_right += scores[i];
  }
```

```
i = 0;
do {
  /* update sufficent statistics */
  while ((i < N) && (scores[i] <= theta)) {
    /* move this example from the right of threshold to the left */
    num_left++;
    num_right--;
    sum_left += scores[i];
    sum_right -= scores[i];
    i++;
  }
  diff_left = num_left * theta - sum_left;
  diff_right = sum_right - num_right * theta;

  /* compute beta */
  if (diff_left == 0) {
    /* default value for beta */
    beta = EPSILON_INF;
  }
  else {
    /* compute closed MLE for beta */
    beta = N / (diff_left + sqrt(diff_left) * sqrt(diff_right));
  }

  /* compute gamma */
  if (diff_right == 0) {
    /* default value for gamma */
    gamma = EPSILON_INF;
  }
  else {
    /* compute closed MLE for gamma */
    gamma = N / (diff_right + sqrt(diff_left) * sqrt(diff_right));
  }

  /* compute log posterior */
  ln_posterior = (N * (log(beta * gamma) - log(beta + gamma))
                  - beta * diff_left - gamma * diff_right);

  /* update set of best parameters */
  if (max_init) {
    if (ln_posterior > max_ln_posterior) {
      *final_theta = theta;
      *final_beta = beta;
      *final_gamma = gamma;
      max_ln_posterior = ln_posterior;
    }
  }
  else {
    *final_theta = theta;
    *final_beta = beta;
    *final_gamma = gamma;
    max_ln_posterior = ln_posterior;
    max_init = 1;
```

```
    }

    /* get new choice for theta */
    if (theta == max) {
      /* already tried max so we're done */
      done = 1;
    }
    else {
      double next = max;
      if (i != N)
        next = scores[i];
      slice_num++;
      if ((slice_num % num_interval_slices) == 0) {
        prev_score_theta = next;
        theta = prev_score_theta;
        slice_num = 0;
      }
      else {
        theta = slice_num * ((next - prev_score_theta) /
                             num_interval_slices) + prev_score_theta;
      }
      /* check if that's the bound */
      if (theta > max)
        done = 1;
    }
  } while (!done);
}
```

Appendix C: Derivation of MLEs for Asymmetric Gaussian Distribution

For $\boldsymbol{D} = \{x_1, x_2, \ldots, x_N\}$ where the x_i are *i.i.d.* and $X \sim \Gamma(X \mid \theta, \sigma_l, \sigma_r)$, the likelihood is:

$$\prod_{i=1}^{N} \Gamma(x_i \mid \theta, \sigma_l, \sigma_r) \tag{16}$$

We desire to find the maximum likelihood estimates for σ_l, σ_r and θ. To do so, we fix θ and compute the maximum likelihood for that choice of θ. Then, we can simply consider all choices of θ and choose the one with the maximum likelihood (or equivalently the loglikelihood) over all choices of θ.

The loglikelihood we must compute then is:

$$\begin{aligned}
\log \prod_{i=1}^{N} \Gamma(x_i \mid \theta, \sigma_l, \sigma_r) &= \sum_{i=1}^{N} \log \Gamma(x_i \mid \theta, \sigma_l, \sigma_r) &(17)\\
&= \sum_{x \in \boldsymbol{D} \mid x \leq \theta} \log \Gamma(x_i \mid \theta, \sigma_l, \sigma_r) + \sum_{x \in \boldsymbol{D} \mid x > \theta} \log \Gamma(x_i \mid \theta, \sigma_l, \sigma_r) &(18)\\
&= \sum_{x \in \boldsymbol{D} \mid x \leq \theta} \left[\log \frac{2}{\sqrt{2\pi}(\sigma_l + \sigma_r)} - \frac{(x-\theta)^2}{2\sigma_l^2} \right] + \sum_{x \in \boldsymbol{D} \mid x > \theta} \left[\log \frac{2}{\sqrt{2\pi}(\sigma_l + \sigma_r)} - \frac{(x-\theta)^2}{2\sigma_r^2} \right] &(19)\\
&= N \log \frac{2}{\sqrt{2\pi}(\sigma_l + \sigma_r)} - \frac{1}{2\sigma_l^2} \sum_{x \in \boldsymbol{D} \mid x \leq \theta} (x-\theta)^2 - \frac{1}{2\sigma_r^2} \sum_{x \in \boldsymbol{D} \mid x > \theta} (x-\theta)^2 &(20)
\end{aligned}$$

Let $N_l = |\ \{x \in \boldsymbol{D} \mid x \leq \theta\}\ |\ ,\ N_r = |\ \{x \in \boldsymbol{D} \mid x > \theta\}\ |\ ,\ S_l = \sum_{x \in \boldsymbol{D} \mid x \leq \theta} x,\ S_r = \sum_{x \in \boldsymbol{D} \mid x > \theta} x,$

$S_{l^2} = \sum_{x \in \boldsymbol{D} \mid x \leq \theta} x^2$, and $S_{r^2} = \sum_{x \in \boldsymbol{D} \mid x > \theta} x^2$.

$$= N \log \frac{2}{\sqrt{2\pi}(\sigma_l + \sigma_r)} - \frac{1}{2\sigma_l^2} \left[S_{l^2} - S_l\theta + N_l\theta^2 \right] - \frac{1}{2\sigma_r^2} \left[S_{r^2} - S_r\theta + N_r\theta^2 \right] \tag{21}$$

Let $D_{l^2} = S_{l^2} - S_l\theta + \theta^2 N_l,\ D_{r^2} = S_{r^2} - S_r\theta + \theta^2 N_r$

$$= N \log \frac{2}{\sqrt{2\pi}(\sigma_l + \sigma_r)} - \frac{1}{2\sigma_l^2} D_{l^2} - \frac{1}{2\sigma_r^2} D_{r^2} \tag{22}$$

The partial derivatives are: $\frac{\partial \sigma_l}{\partial l} = \frac{D_{l2}}{\sigma_l^3} - \frac{N}{\sigma_l + \sigma_r}$ and $\frac{\partial \sigma_r}{\partial l} = \frac{D_{l2}}{\sigma_r^3} - \frac{N}{\sigma_l + \sigma_r}$. We can set the derivatives to zero and solve them analytically to find for a fixed θ only one feasible solution:

$$\sigma_{l,MLE} = \sqrt{\frac{D_{l^2} + D_{l^2}^{2/3} D_{r^2}^{1/3}}{N}} \qquad \sigma_{r,MLE} = \sqrt{\frac{D_{r^2} + D_{r^2}^{2/3} D_{l^2}^{1/3}}{N}}. \tag{23}$$

We then can iterate through alternate choices for θ.

For comparison of the symmetry of this solution to the asymmetric Laplace, we can also write the solution as:

$$\sigma_{l,MLE} = \frac{D_{l^2}^{1/3}}{N^{1/3}} \sqrt{\frac{D_{l^2}^{1/3} + D_{r^2}^{1/3}}{N^{1/3}}} \qquad \sigma_{r,MLE} = \frac{D_{r^2}^{1/3}}{N^{1/3}} \sqrt{\frac{D_{l^2}^{1/3} + D_{r^2}^{1/3}}{N^{1/3}}}. \tag{24}$$

The second part of each equation is equal to $(\sigma_{l,MLE} + \sigma_{r,MLE})^{1/3}$.

Appendix D: C Code for MLE of Asymmetric Gaussian

This code is also currently available at http://www.cs.cmu.edu/~pbennett/asymmetric/aGaussian.c .

```
/* For the data given in scores,
   finds the maximum likelihood set of parameters for
   an asymmetric Gaussian family

   p(x | THETA, BETA, GAMMA) =
        if (x <= THETA) then
             (2 / (sqrt(2 * pi) (BETA + GAMMA))) exp [-1/2 ((x -
THETA) / BETA)^2]
        else
             (2 / (sqrt(2 * pi) (BETA + GAMMA))) exp [-1/2 ((x -
THETA) / GAMMA)^2]

   candidate thetas are restricted to the range [min,max]
*/
void find_aGaussian_parameters(double * scores, /* vector of scores  - assumed
                                                      to be sorted least to
                                                      greatest */
                              long int N, /* number of scores */
                              double min, /* min mode to try */
                              double max, /* max mode to try */
                              double defaultm, /* default mode if N == 0 */
                              int num_interval_slices, /* breaks scores[i] and
                                                           scores[i + 1] into
                                                           this many pieces to
                                                           try as candidate
                                                            modes */
                              /* next three are return values of distribution */
                              double * final_theta,
                              double * final_beta,
                              double * final_gamma) {
  double max_ln_posterior, ln_posterior;
  double max_init = 0;
  double prev_score_theta = min;
  double theta = prev_score_theta, beta, gamma;
  long int num_left = 0, num_right = N;
  double sum_left = 0, sum_right = 0;
  double sum_squares_left = 0, sum_squares_right = 0;
  double diff_left, diff_right, diff_left_3root, diff_right_3root;
  int done = 0, slice_num = 0;
  long int i;

  if (N == 0) {
    /* no scores */
    *final_theta = defaultm;
    *final_beta = EPSILON_INF;
    *final_gamma = EPSILON_INF;
    return;
  }

  /* loop through and get the sum and sum of square of all scores */
  for (i = 0; i < N; i++) {
```

```
    sum_right += scores[i];
    sum_squares_right += (scores[i] * scores[i]);
  }

  i = 0;
  do {
    /* update sufficent statistics */
    while ((i < N) && (scores[i] <= theta)) {
      double score_squared = scores[i] * scores[i];
      /* move this example from the right of threshold to the left */
      num_left++;
      num_right--;
      sum_left += scores[i];
      sum_right -= scores[i];
      sum_squares_left += score_squared;
      sum_squares_right -= score_squared;
      i++;
    }
    diff_left = sum_squares_left - 2 * sum_left * theta + theta * theta * num_left;
    diff_right = sum_squares_right - 2 * sum_right * theta + theta * theta * num_right;
    diff_left_3root = cbrt(diff_left);
    diff_right_3root = cbrt(diff_right);

    /* compute beta */
    if (diff_left == 0) {
      /* default value for beta */
      beta = EPSILON_ZERO;
    }
    else {
      /* compute closed MLE for beta */
      beta = (sqrt(diff_left
                   + diff_left_3root * diff_left_3root * diff_right_3root)
              / sqrt(N));
    }

    /* compute gamma */
    if (diff_right == 0) {
      /* default value for gamma */
      gamma = EPSILON_ZERO;
    }
    else {
      /* compute closed MLE for gamma */
      gamma = (sqrt(diff_right
                    + diff_right_3root * diff_right_3root * diff_left_3root)
               / sqrt(N));
    }

    /* compute log posterior */
    ln_posterior = ( /* N * (log(2) - log(sqrt(2 * pi))) -- constant for all
                          choices */
                    N * (- log (beta + gamma))
                    - 0.5 * diff_left / (beta * beta)
                    - 0.5 * diff_right / (gamma * gamma));
```

```
    /* update set of best parameters */
    if (max_init) {
      if (ln_posterior > max_ln_posterior) {
        *final_theta = theta;
        *final_beta = beta;
        *final_gamma = gamma;
        max_ln_posterior = ln_posterior;
      }
    }
    else {
      *final_theta = theta;
      *final_beta = beta;
      *final_gamma = gamma;
      max_ln_posterior = ln_posterior;
      max_init = 1;
    }

    /* get new choice for theta */
    if (theta == max) {
      /* already tried max so we're done */
      done = 1;
    }
    else {
      double next = max;
      if (i != N)
        next = scores[i];
      slice_num++;
      if ((slice_num % num_interval_slices) == 0) {
        prev_score_theta = next;
        theta = prev_score_theta;
        slice_num = 0;
      }
      else {
        theta = slice_num * ((next - prev_score_theta) /
                             num_interval_slices) + prev_score_theta;
      }
      /* check if that's the bound */
      if (theta > max)
        done = 1;
    }
  } while (!done);
}
```

References

Bennett, P. N. (2000). *Assessing the calibration of naive bayes' posterior estimates* (Technical Report CMU-CS-00-155). Carnegie Mellon, School of Computer Science.

Bourlard, H., & Morgan, N. (1990). A continuous speech recognition system embedding mlp into hmm. In D. S. Touretzky (Ed.), *Advances in neural information processing systems*, vol. 2, 186–193. Morgan Kaufmann.

Brier, G. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, *78*, 1–3.

DeGroot, M. H. (1989). *Probability and statistics*. Addison-Wesley. 2nd edition.

DeGroot, M. H., & Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *Statistician*, *32*, 12–22.

DeGroot, M. H., & Fienberg, S. E. (1986). Comparing probability forecasters: Basic binary concepts and multivariate extensions. In P. Goel and A. Zellner (Eds.), *Bayesian inference and decision techniques*. Elsevier Science Publishers B.V.

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier. *ICML '96*.

Duda, R., Hart, P., & Stork, D. (2001). *Pattern classification*. John Wiley & Sons, Inc.

Dumais, S. T., & Chen, H. (2000). Hierarchical classification of web content. *SIGIR '00*.

Dumais, S. T., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *CIKM '98*.

Freund, Y., & Schapire, R. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, *37*, 277–296.

Good, I. (1952). Rational decisions. *Journal of the Royal Statistical Society, Series B*.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *ECML '98*.

Kotz, S., Kozubowski, T. J., & Podgorski, K. (2001). *The laplace distribution and generalizations: A revisit with applications to communications, economics, engineering, and finance*. Birkhäuser.

Lewis, D. D. (1995). A sequential algorithm for training text classifiers: Corrigendum and additional data. *SIGIR Forum*, *29*, 13–19.

Lewis, D. D. (1997). Reuters-21578, distribution 1.0. http://www.research.att.com/~lewis.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. *SIGIR '94*.

Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training algorithms for linear text classifiers. *SIGIR '96*.

Lindley, D., Tversky, A., & Brown, R. (1979). On the reconciliation of probability assessments. *Journal of the Royal Statistical Society*.

Manmatha, R., Rath, T., & Feng, F. (2001). Modeling score distributions for combining the outputs of search engines. *Sigir '01*.

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. *AAAI '98, Workshop on Learning for Text Categorization*.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Scholkopf and D. Schuurmans (Eds.), *Advances in large margin classifiers*. MIT Press.

Saar-Tsechansky, M., & Provost, F. (2001). Active learning for class probability estimation and ranking. *IJCAI '01*.

Winkler, R. L. (1969). Scoring rules and the evaluation of probability assessors. *Journal of the American Statistical Association*.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. *SIGIR '99*.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *ICML '01*.