Large-Scale Bayesian Logistic Regression for Text Categorization

Alexander Genkin[1], David D. Lewis[3,4], and David Madigan[1,2]

DIMACS[1]; Department of Statistics, Rutgers University[2]; Ornarose Inc.[3]; David D. Lewis Consulting[4]

Abstract

This paper describes an application of Bayesian logistic regression to text categorization. In particular we examine so-called "sparse" Bayesian models that simultaneously select features and provide shrinkage. We present an optimization algorithm for efficient fitting of these models with 10's of thousands of predictors, and provide empirical evidence that these models give good predictive performance while offering significant computational advantages. Publicly available software accompanies this paper.

1 Introduction

Logistic regression is a mainstay for applied statisticians. The model enjoys a substantial body of supporting theory and algorithms, features prominently in essentially all commercial statistical software products, and, in careful hands, often performs as well as newer techniques in terms of predictive accuracy.

However, new applications have emerged that pose computational and statistical challenges to standard logistic regression. A key characteristic of these applications is that the number of predictor variables is large (10^4 and up) and usually exceeds the number of observations. Examples of such "short, fat datasets" include text categorization (our particular interest in this paper), gene expression analysis, adverse event monitoring, longitudinal clinical trials, and a variety of business data mining tasks.

Maximum likelihood estimation often fails in these applications. Numerical ill-conditioning can result in a lack of convergence, large estimated coefficient variances, poor predictive accuracy, and/or reduced power for testing hypotheses concerning model assessment (Pike

et al., 1980). Exact logistic regression suffers from many of the same problems (Greenland *et al.*, 2000). Furthermore, while maximum likelihood has desirable asymptotic properties, it often overfits the data, even when numerical problems are avoided.

Computational efficiency, both during fitting and when the fitted model is used for prediction, is also a problem. Most logistic regression implementations make use of matrix inversion. This imposes modest practical limits on the number of predictor variables. A common workaround uses feature selection to discard the majority of predictor variables before fitting. Feature selection does improve human interpretability, and also reduces memory and computional requirements at prediction time (Kittler, 1986, Mitchell and Beauchamp, 1988). There are, however, several downsides to feature selection as a preprocessing step. First, the statistical foundation of these feature selection methods, and thus of the modeling process they contribute to, is unclear. This makes it difficult, for instance, to come up with principled ways to choose the number of features to use for a given task. Second, these methods typically consider each feature in isolation and may choose redundant or otherwise ineffective combinations of features. Finally, since their foundations are unclear, there is little to guide one in combining these feature selection methods with, for instance, domain knowledge.

We report here on a Bayesian approach which avoids overfitting, gives state-of-the-art effectiveness, and is extremely efficient both during fitting and at prediction time. The key to the approach is the use of a prior probability distribution that favors sparseness in the fitted model, along with an optimization algorithm and implementation tailored to that prior. By "sparseness" we mean that the posterior point estimates for many of the model parameters are zero.

We have publicly released our software at `http://stat.rutgers.edu/~madigan/BBR` and have used it in a wide variety of applications.

We begin in Section 2 by briefly describing how supervised learning is used in text categorization, the motivating application for our work. In Section 3 we present a formal framework for Bayesian learning, focusing on the use of hierarchical priors to achieve sparseness (and thus efficiency) in fitting logistic regression models. Section 4 describes the algorithm we use to fit our model to training data. Section 5 describes the data sets and methods we use in our experiments, and Section 6 the experimental results. We find that the sparse classifiers are competitive with state-of-the-art text categorization algorithms, including widely used

feature selection methods. Finally, Section 7 outlines directions for future research.

2 Text Categorization

Text classification algorithms assign texts to predefined classes. When those classes are of interest to only one user, we often refer to text classification as *filtering* or *routing*. When the classes are of interest to a population of users, we instead refer to *text categorization*.

The study of text categorization algorithms dates back more than forty years (Maron, 1961). In the last decade statistical approaches have dominated the research literature and, increasingly, operational practice. Statistical approaches to text categorization or, more precisely, supervised learning approaches, infer ("learn") a classifier (i.e. a rule that decides whether or not a document should be assigned to a category) from a set of labeled documents (i.e. documents with known category assignments). Depending on the category scheme, a categorization task may be framed as one or more binary and/or polychotomous classification problems. Sebastiani (2002) provides an overview of the approaches that have been taken.

The scale of text categorization applications causes problems for many statistical algorithms. Documents to be classified are typically represented as vectors of numeric feature values derived from words, phrases, or other characteristics of documents. The dimensionality of these vectors ranges from 10^3 to 10^6 or more. Early text categorization researchers therefore focused on learning algorithms that were both computationally efficient (for speed) and restricted in the classifiers they could produce (to avoid overfitting). Examples include Naive Bayes (Maron, 1961, Lewis, 1998) and the Rocchio algorithm (Rocchio, 1971). In addition, heuristic feature selection methods were often used to discard most features from the document vectors.

Recently, increased computing power and new regularization approaches have enabled algorithms to learn less restricted classifiers while simultaneously avoiding overfitting and maintaining sufficient speed during learning. Examples include support vector machines (Joachims, 1998, Zhang and Oles, 2001, Lewis, *et al.*, 2004), boosting (Schapire and Singer, 2000), and ridge logistic regression (Zhang and Oles, 2001). However, these methods still require feature selection if they are to produce compact and thus efficient classifiers.

3 The Model

We are interested in learning text classifiers, $y = f(\boldsymbol{x})$, from a set of training examples $D = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_i}, y_i), \ldots, (\boldsymbol{x_n}, y_n)\}$. For text categorization, the vectors $\boldsymbol{x_i} = [x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,d}]^T$ consist of transformed word frequencies from documents (Section 5.1), though the approach could be applied to numeric vectors representing any form of data. The values $y_i \in \{-1, +1\}$ are class labels encoding membership $(+1)$ or nonmembership (-1) of the vector in the category.[1]

Concretely, we are interested in conditional probability models of the form

$$p(y = +1|\boldsymbol{\beta}, \boldsymbol{x_i}) = \psi(\boldsymbol{\beta}^T \boldsymbol{x_i}) = \psi(\sum_j \beta_j x_{i,j}). \tag{1}$$

In what follows we use the logistic link function:

$$\psi(r) = \frac{\exp(r)}{1 + \exp(r)}, \tag{2}$$

producing a logistic regression model. In preliminary work (Genkin, *et al.*, 2003) we used the very similar probit function (Figueireido, 2003) as our link.

For a text categorization problem, $p(y = +1|\boldsymbol{x_i})$ corresponds to the probability that the ith document belongs to the category. The decision of whether to assign the category can be based on comparing the probability estimate with a threshold or, more generally, by computing which decision gives optimal expected utility (Duda and Hart, 1973, Lewis, 1995).

For a logistic regression model to make accurate predictions for future input vectors (documents we wish to classify), we must avoid overfitting the training data. One Bayesian approach to avoiding overfitting is use a prior distribution for $\boldsymbol{\beta}$ that assigns a high probability that each β_j will take on a value at or near 0. In the remainder of this section we describe several such priors.

[1] We encode class labels as $-1/+1$ rather than $0/1$ to simplify presentation of our fitting algorithm.

3.1 Gaussian Priors & Ridge Logistic Regression

Perhaps the simplest Bayesian approach to the logistic regression model is to impose a univariate Gaussian prior with mean 0 and variance $\tau_j > 0$ on each parameter β_j:

$$p(\beta_j|\tau_j) = N(0, \tau_j) = \frac{1}{\sqrt{2\pi\tau_j}} \exp(\frac{-\beta_j^2}{2\tau_j}). \tag{3}$$

The mean of 0 encodes our prior belief that β_j will be near 0. The variances τ_j are positive constants we must specify. A small value of τ_j represents a prior belief that β_j is close to zero. A large value of τ_j represents a less informative prior belief. In the simplest case we let τ_j equal the same τ for all j. We assume *a priori* that the components of $\boldsymbol{\beta}$ are independent and hence the overall prior for $\boldsymbol{\beta}$ is the product of the priors for each of its component β_j's. Finding the maximum *a posteriori* (MAP) estimate of $\boldsymbol{\beta}$ with this prior is equivalent to ridge regression (Hoerl and Kennard, 1970) for the logistic model (Santner and Duffy, 1989, Le Cessie and Van Houwelingen, 1992).

This Gaussian prior, while favoring values of β_j near 0, does not favor β_j's being exactly equal to 0. Absent unusual patterns in the data, the MAP estimates of all or almost all β_j's will be nonzero. To obtain sparse text classifiers with a Gaussian prior, previous authors have used various feature selection mechanisms - see, for example, Zhang and Oles (2001) and Zhang and Yang (2003).

3.2 Laplace Priors & Lasso Logistic Regression

Here we describe a hierarchical prior distribution for $\boldsymbol{\beta}$ that does achieve sparseness. As above, we give each β_j a Gaussian distribution with mean 0 and variance τ_j:

$$p(\beta_j|\tau_j) = N(0, \tau_j). \tag{4}$$

However, we now allow the variance τ_j to be different for each β_j, and give the τ_j's their own prior distribution. One choice of a prior for τ_j is a one-parameter exponential distribution:

$$p(\tau_j|\gamma) = \frac{\gamma_j}{2} \exp(-\frac{\gamma_j}{2}\tau_j). \tag{5}$$

with $\gamma > 0$. Integrating out τ_j then gives a (nonhierarchical) double exponential or Laplace prior distribution:

$$p(\beta_j|\lambda_j) = \frac{\lambda_j}{2} \exp(-\lambda_j|\beta_j|), \tag{6}$$

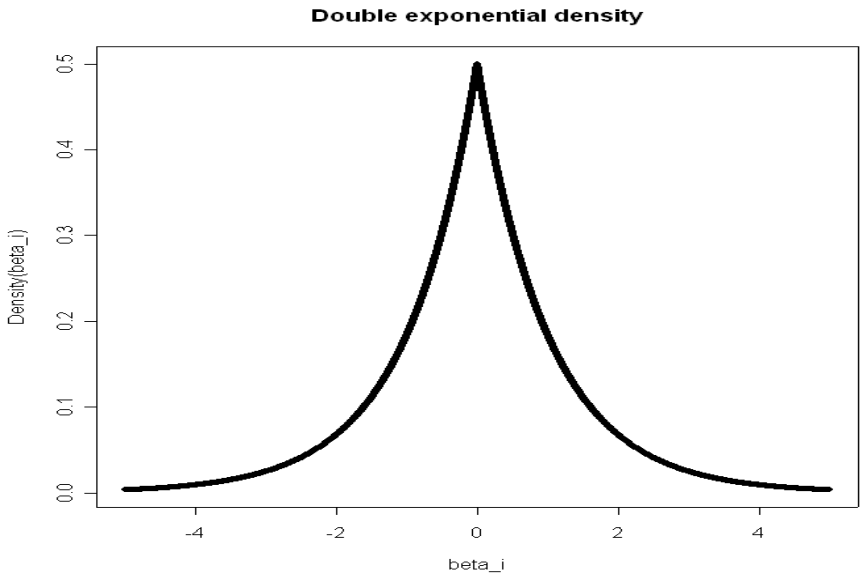

Figure 1: *The Laplace density with* $\lambda_j = 1$.

where $\lambda_j = \sqrt{\gamma}_j > 0$. In what follows we assume that $\lambda_j \equiv \lambda \forall j$. As before, the prior for $\boldsymbol{\beta}$ is the product of the priors for its components.

Figure 1 shows the density of the Laplace distribution. At zero the density is finite but discontinuous. The distribution has mean 0 and variance $2/\lambda^2$.

Figures 2 and 3 show the effect of hyperparameter settings on the MAP logistic regression parameters on a particular data set with eight predictor variables. Figure 2 shows the effect of a Gaussian prior distribution on each parameter, with all Gaussians having the same variance, τ. When τ is small, each β_j has a small prior variance, and the resulting MAP estimates are small, approaching zero as $\tau \to 0$. When τ is large, the MAP estimates are similar

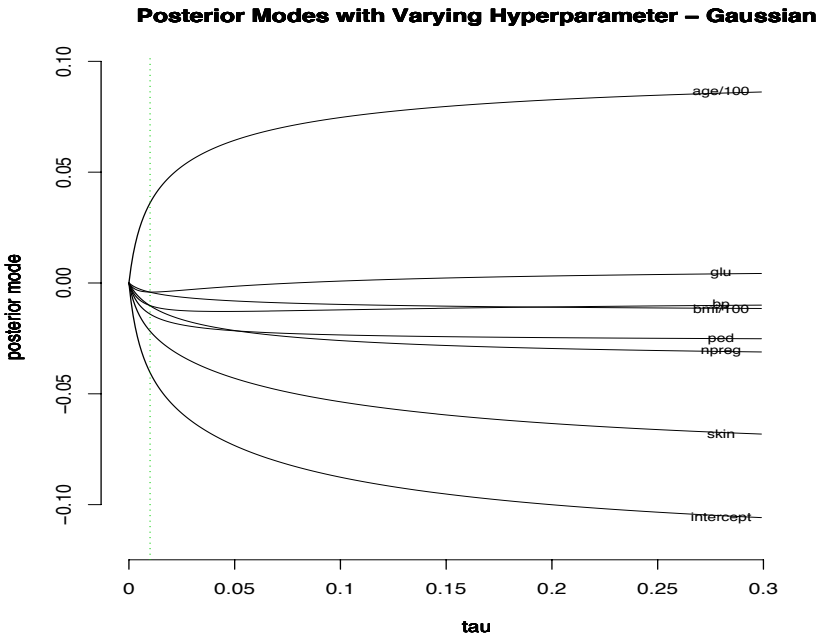

Figure 2: MAP estimates (y-axis) of the eight parameters of a logistic regression as the hyperparameter γ of a Gaussian prior on those parameters varies (x-axis). For example, the vertical dotted line shows the eight estimates corresponding to a particular choice of γ.

to the maximum likelihood estimates. The vertical dashed line corresponds to $tau = 0.01$. Figure 3 shows the equivalent picture for the Laplace prior. As with the Gaussian prior, the MAP estimates range from all zeroes to the maximum likelihood estimates. However, unlike the Gaussian case, particular choices for the hyperparameter lead to MAP estimates where some components of $\boldsymbol{\beta}$ are zero while others are not. The vertical dashed line corresponds to prior distributions with a variance of 0.01. Note that with this particular choice for the hyperparameter, the posterior modes of two of the parameters are zero. Hastie *et al.* (2003) show similar plots for linear regression.

The Laplace distribution has a larger kurtosis than Gaussian distribution. Using this kind of prior represents a prior belief that a small portion of the input variables have a substantial effect on the outcome while most of the others are most likely unimportant.

Tibshirani (1996) introduced *Lasso* for linear regression models as an alternative to feature selection for producing sparse models. The lasso estimate was defined as a least squares estimate subject to a constraint on the sum of absolute values of the coefficients. Tibshirani

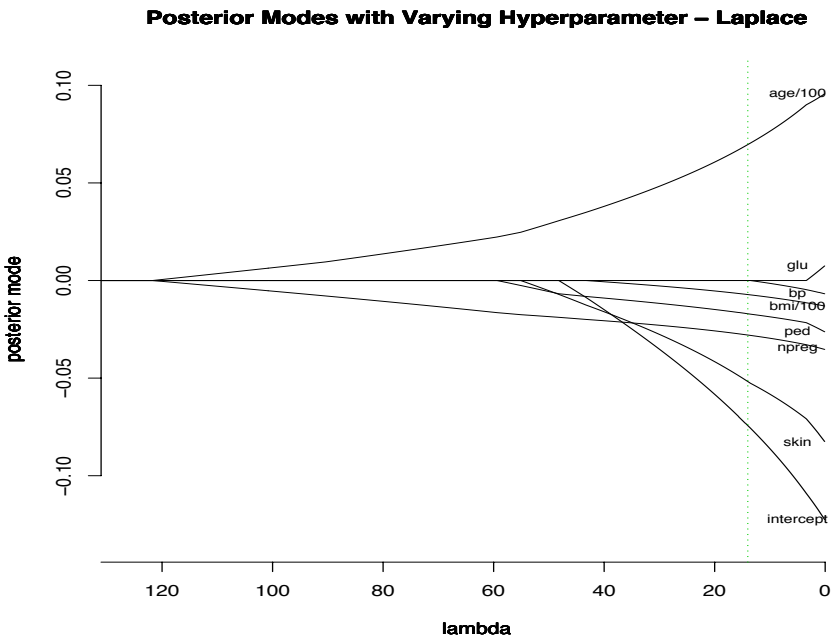

Figure 3: MAP estimates of logistic regression parameters on the same data set as Figure 2, shown for varying values of $\gamma = \lambda^2$ where λ is the hyperparameter of a Laplace prior. For example, the vertical dotted line shows the eight estimates corresponding to a particular choice of γ, a choice that selects six of the eight predictor variables.

observed that this was equivalent to a Bayesian MAP estimate using a Laplace prior, as presented above. The use of constraints or penalties based on the absolute values of coefficients has been used to achieve sparseness in a variety of data fitting tasks (see, for example, Efron *et al.*, 2004, Figueiredo, 2003, Girosi, 1998, and Tipping, 2001).

3.3 A Hierarchical Prior with No Hyperparameters

Another hierarchical prior that leads to sparse MAP estimates is described by Figueiredo and colleagues (Figueiredo and Jain, 2001, Figueiredo, 2003). They produce MAP estimates for linear and probit models using Gaussian priors for the β_j's, with a Jeffreys' prior on the τ_j's:

$$p(\tau_j) \propto 1/\tau_j. \tag{7}$$

The Jeffreys prior is an improper distribution. It has no adjustable hyperparameters, which can either be viewed as a simplifying advantage, or as a lack of opportunity for encoding task knowledge.

Figueiredo presents an EM algorithm for finding a local maximum of the posterior distribution of $\boldsymbol{\beta}$. However, because the posterior distribution for $\boldsymbol{\beta}$ with the Jeffrey's prior is not necessarily convex (Genkin, *et al.*, 2003), the EM algorithm is not guaranteed to find the MAP estimate.

4 Finding the MAP Estimate of $\boldsymbol{\beta}$

In an ideal setting, prediction with a Bayesian logistic regression model would use the posterior distribution of $\boldsymbol{\beta}$ to compute a posterior distribution for a desired y_{new}, given the corresponding $\boldsymbol{x_{new}}$ (Mallick *et al.*, 2003). In many practical prediction tasks, however, computational efficiency requires that we base predictions on a point estimate of $\boldsymbol{\beta}$. Smith (1999) points out that this simplification does not necessarily lead to less accurate predictions.

The most common Bayesian point estimates used are the posterior mean (the expected value of $\boldsymbol{\beta}$ over its posterior distribution) and the posterior mode (any value of $\boldsymbol{\beta}$ at which

the posterior distribution takes on its maximum value). For the logistic regression model, with the priors we have discussed, there is neither an analytic expression nor inexpensive computational procedure for finding the posterior mean and hence we focus on posterior mode estimation.

The posterior distribution for $\boldsymbol{\beta}$ with the logistic link on data set D is:

$$L(\boldsymbol{\beta}) = p(\boldsymbol{\beta}|D) = (\prod_{i=1}^{n} \frac{1}{1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)}) p(\boldsymbol{\beta}) \tag{8}$$

where $p(\boldsymbol{\beta})$ is the prior on $\boldsymbol{\beta}$ and i indexes the training examples. For numerical convenience, we typically work instead with the log posterior:

$$l(\boldsymbol{\beta}) = \ln p(\boldsymbol{\beta}|D) = -(\sum_{i=1}^{n} \ln(1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)) + \ln p(\boldsymbol{\beta}), \tag{9}$$

which has the same maximum as $L()$. For Gaussian priors with mean 0 and variance τ on the β_j's we have:

$$l(\boldsymbol{\beta}) = -(\sum_{i=1}^{n} \ln(1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)) - \sum_{j=1}^{d} (\ln \sqrt{\tau} + \frac{\ln 2\pi}{2} + \frac{\beta_j^2}{\tau}), \tag{10}$$

and for Laplace priors with mean 0 and variance $2/\lambda_j^2$ we have:

$$l(\boldsymbol{\beta}) = -(\sum_{i=1}^{n} \ln(1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)) - \sum_{j=1}^{d} (\ln 2 - \ln \lambda_j + \lambda_j |\beta_j|), \tag{11}$$

with $j = 1 \ldots d$ indexing the features in both cases.

The MAP estimate of $\boldsymbol{\beta}$ is then simply

$$\arg\max_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} -l(\boldsymbol{\beta}) \tag{12}$$

where by convention we use the negated log-posterior as the objective function and treat the optimization problem as a minimization.

4.1 The Logistic Model from an Optimization Standpoint

The negated log-posterior for a logistic regression model is strictly convex with either the Gaussian or Laplace prior. Further, $-l(\boldsymbol{\beta}) \to -\infty$ as $||\boldsymbol{\beta}||_2 \to \infty$ (or $||\boldsymbol{\beta}||_1 \to \infty$). Therefore

the only local minimum of the objective is the unique, finite global minimum, and a wide variety of optimization algorithms are applicable. For maximum likelihood logistic regression the most common optimization approach in statistical software is some variant of the multidimensional Newton-Raphson method (Dennis and Schnabel, 1989). Newton algorithms have the advantage of converging in very few iterations. Further, the matrix of second derivatives they compute has other uses, such as in finding asymptotic confidence intervals on parameters. For the logistic likelihood, a Newton method can conveniently be implemented as an iteratively reweighted least squares algorithm (Hastie and Pregibon, 1992).

For high-dimensional problems such as text categorization, however, Newton algorithms have the serious disadvantage of requiring $O(d^2)$ memory, where d is the number of model parameters. A variety of alternate optimization approaches have therefore been explored for maximum likelihood logistic regression, and for MAP logistic (or probit) regression with a Gaussian prior. Some of these algorithms, such as limited memory BFGS (Malouf, 2002), conjugate gradient (Malouf, 2002), and hybrids of conjugate gradient with other methods (Komarek and Moore, 2003), compute the gradient of the objective function at each step. This requires only $O(d)$ memory (in addition to the data itself). Efron *et al.* (2004) describe a new class of "least angle" algorithms for lasso linear regression and related models. Madigan and Ridgeway (2004) discuss possible extensions to the logistic model.

Other methods solve a series of partial optimization problems. Some of these methods use the subproblems to maintain an evolving approximation to the gradient of the full objective, which still requires $O(d)$ memory. Others use each subproblem only to make progress on the overall objective, using only constant memory beyond that for the parameter vector. The smaller problems may be based on processing one example at a time, as in dual Gauss-Seidel (Zhang and Oles, 2001), stochastic gradient descent[2] (Bishop, 1995), exponentiated gradient descent (Kivinen and Warmuth, 2001), and Bayesian online algorithms (Chai, *et al.*, 2002, Ridgeway and Madigan, 2003). Or the one dimensional problems may be based on processing one parameter at a time, as in iterative scaling (Jin, *et al.*, 2003) and cyclic coordinate descent (Zhang and Oles, 2001). Some of these algorithms have already shown promise on text categorization or other language processing tasks. Of these, we base our work on the cyclic coordinate decsent algorithm, as described in the next section, due to its speed and simplicity of implementation.

[2]Also known as *sequential*, *incremental*, *online*, or *pattern-based* gradient descent.

4.2 The CLG Algorithm for Ridge Logistic Regression

CLG is a *cyclic coordinate descent* optimization algorithm[3] (Bazaraa and Shetty, 1979, Luenberger (sec. 7.9), 1984) for fitting a logistic model with Gaussian prior. Cyclic coordinate descent algorithms begin by setting all variables to some initial value. They then find which value of the first variable minimizes the objective function, assuming all other variables are held constant at their initial values. This is a one-dimensional optimization problem.

The algorithm then finds the minimizing value of a second variable, while holding all other values constant (including the new value of the first variable). Then the third variable is optimized and so on. When all variables have been traversed, the algorithm returns to the first variable and starts again. Multiple passes are made over the variables until some convergence criterion is met.

When fitting a ridge logistic regression model, each one-dimensional problem involves finding β_j^{new}, the value for the j'th parameter that gives the lowest value for $-l(\boldsymbol{\beta})$ (the negative of the log-posterior in Equation 10), assuming that the other $\beta_{j'}$'s are held at their current values. Finding this β_j^{new} is equivalent to finding the minimum of the following one-dimensional objective function:

$$g(z) = (\sum_{i=1}^{n} f(r_i + (z - \beta_j)x_{ij}y_i)) + \frac{z^2}{\tau_j}, \tag{13}$$

where the $r_i = \boldsymbol{\beta}^T \boldsymbol{x_i} y_i$ are computed using the current value of $\boldsymbol{\beta}$ and so are treated as constants, $f(r) = \ln(1 + \exp(-r))$, and where Gaussian penalty terms not involving z are treated as constant and omitted.

The β_j^{new} that gives the minimum value of $g()$ does not have a closed form, so even for this one-dimensional problem an iterative optimization procedure must be used. Zhang and Oles use an approach inspired by the one-dimensional Newton's method (Press, *et al.*, 1992).

The classic Newton's method would approximate the objective function $g()$ by the first three terms of its Taylor series at the current value of the parameter being optimized (β_j for us).

[3]Cyclic coordinate algorithms have also been called *alternating directions*, *univariate search parallel to an axis*, *one-at-a-time*, *sectioning*, *axial iteration*, *relaxation*, and *Gauss-Seidel type* algorithms. (The classic Gauss-Seidel algorithm is a cyclic coordinate descent algorithm for solving a system of linear equations.) Zhang and Oles refer to CLG a *Gauss-Seidel relaxation* algorithm.

This approximation is:

$$\tilde{g}(z) = g(\beta_j) + g'(\beta_j)(z - \beta_j) + \frac{1}{2}g''(\beta_j)(z - \beta_j)^2 \approx g(z), \tag{14}$$

where for a ridge logistic regression model we have:

$$\begin{aligned} g'(\beta_j) &= \left.\frac{dg(z)}{dz}\right|_{z=\beta_j} \\ &= \left.\left(\sum_{i=1}^{n} x_{ij} y_i f'(r_i + (z - \beta_j)x_{ij}y_i)\right) + \frac{2z}{\tau_j}\right|_{z=\beta_j} \\ &= \left(\sum_{i=1}^{n} x_{ij} y_i f'(r_i)\right) + \frac{2\beta_j}{\tau_j} \\ &= \left(\sum_{i=1}^{n} x_{ij} y_i \frac{1}{1 + \exp(r_i)}\right) + \frac{2\beta_j}{\tau_j} \end{aligned} \tag{15}$$

and

$$\begin{aligned} g''(\beta_j) &= \left.\frac{d^2 g(z)}{d^2 z}\right|_{z=\beta_j} \\ &= \left.\left(\sum_{i=1}^{n} x_{ij}^2 f''(r_i + (z - \beta_j)x_{ij}y_i)\right) + \frac{2}{\tau_j}\right|_{z=\beta_j} \\ &= \left(\sum_{i=1}^{n} x_{ij}^2 f''(r_i)\right) + \frac{2}{\tau_j} \\ &= \left(\sum_{i=1}^{n} x_{ij}^2 \frac{\exp(r_i)}{(1 + \exp(r_i))^2}\right) + \frac{2}{\tau_j} \end{aligned} \tag{16}$$

The approximation in (14) is a quadratic function of z, and so its minimum[4] can be expressed in closed form:

$$\beta_j^{(new)} = \arg\min_z g(z) = \beta_j - \frac{g'(\beta_j)}{g''(\beta_j)} \tag{17}$$

or, expressing in terms of the increment we make to β_j:

$$\Delta\beta_j = \beta_j^{(new)} - \beta_j = -\frac{g'(\beta_j)}{g''(\beta_j)}. \tag{18}$$

Zhang and Oles modify the the update in Equation 18 in three ways. First, as in most applications of Newton's method, care must be taken to avoid making large and potentially

[4]For our $g()$ we know we have a minimum and not a maximum or saddle point since $\tilde{g}''(z) = g''(z)$ is strictly positive, assuming $x_{ij} \neq 0$ for some i.

misdirected updates in regions where a quadratic is a poor approximation to the objective. Zhang and Oles specify a value Δ_j which $|\Delta\beta_j|$ is not allowed to exceed on a single iteration. This is similar to the trust region approach to Newton's method (Dennis and Schnabel, 1989). Zhang and Oles present several alternative update rules for adapting the width, $2\Delta_j$, of the trust region from iteration to iteration. We used this update:

$$\Delta_j^{new} = \max(2|\Delta\beta_j|, \Delta_j/2) \tag{19}$$

where Δ_j is the trust region half-width used with β_j on the current pass through the coordinates, $\Delta\beta_j$ was the update made to β_j on this pass, and Δ_j^{new} is the trust region half-width to be used on the next pass.

Second, instead of using a truncated Taylor series (Equation 14) as their quadratic approximation in Newton's method, Zhang and Oles use

$$\hat{g}(z) = g(\beta_j) + g'(\beta_j)(z - \beta_j) + \frac{1}{2}G(\beta_j)(z - \beta_j)^2 \approx g(z) \tag{20}$$

where

$$G(\beta_j) = (\sum_{i=1}^{n} x_{ij}^2 F(r_i, \Delta_j|x_{ij}|)) + \frac{2}{\tau_j}. \tag{21}$$

$F(r, \delta)$ is allowed to be any convenient function that satisfies:

$$F(r, \delta) \geq \sup_{|\Delta r| \leq \delta} f''(r + \Delta r) \tag{22}$$

$$= \sup_{|\Delta r| \leq \delta} \frac{\exp(r + \Delta r)}{(1 + \exp(r + \Delta r))^2}. \tag{23}$$

In words, $F(r_i, \Delta_j|x_{ij}|)$ is an upper bound on the second derivative of f for values of r_i reachable by updates in the trust region. For the logistic model Zhang and Oles use:

$$F(r, \delta) = \min\left(0.25, \frac{1}{2\exp(-|\delta|) + \exp(r - |\delta|) + \exp(|\delta| - r)}\right) \tag{24}$$

In our implementation we used the very similar:

$$F(r, \delta) = \begin{cases} 0.25, & \text{if } |r| \leq |\delta| \\ 1/(2 + \exp(|r| - |\delta|) + \exp(|\delta| - |r|)), & \text{otherwise.} \end{cases} \tag{25}$$

Using $\hat{g}()$, the update in Equation 18 becomes:

$$\Delta v_j = -\frac{\hat{g}'(\beta_j)}{\hat{g}''(\beta_j)} = -\frac{(\sum_{i=1}^{n} x_{ij} y_i \frac{1}{1+exp(r_i)}) + 2\beta_j/\tau_j^2}{(\sum_{i=1}^{n} x_{ij}^2 F(r_i, \Delta_j|x_{ij}|) + 2/\tau_j} \tag{26}$$

Algorithm 1 (*CLG*) (Zhang and Oles, 2001)

(1) initialize $\beta_j \leftarrow 0, \Delta_j \leftarrow 1$ for $j = 1, ..., d$; $r_i \leftarrow 0$ for $i = 1, ..., n$;
(2) **for** $k = 1, 2, ...$ **until** convergence
(3) **for** $j = 1, ..., d$
(4) compute tentative step Δv_j (Equation 26)
(5) $\Delta\beta_j \leftarrow \min(\max(\Delta v_j, -\Delta_j), \Delta_j)$ (limit step to trust region)
(6) $\Delta r_i \leftarrow \Delta\beta_j x_{ij} y_i, \quad r_i \leftarrow r_i + \Delta r_i \qquad$ for $i = 1, ..., n$
(7) $\beta_j \leftarrow \beta_j + \Delta\beta_j$
(8) $\Delta_j \leftarrow \max(2|\Delta\beta_j|, \Delta_j/2)$ (update size of trust region)
(9) **end**
(10) **end**

Figure 4: Zhang and Oles' CLG algorithm, with our choice of trust region update.

Applying the trust region restriction then gives the actual update used by Zhang and Oles:

$$\Delta\beta_j = \begin{cases} -\Delta_j & \text{if } \Delta v_j < -\Delta_j \\ \Delta v_j & \text{if } -\Delta_j \leq \Delta v_j \leq \Delta_j \\ \Delta_j & \text{if } \Delta_j < \Delta v_j \end{cases} \tag{27}$$

Third, instead of iterating Equation 27 to convergence (replacing β_j with $\beta_j^{(new)} = \beta_j + \Delta\beta_j$ on each application), Zhang and Oles apply it only once, and then go on to next $\beta_{j'}$. The optimal value of $\beta_j^{(new)}$ during a particular pass through the coordinates depends on the current values of the other $\beta_{j'}$'s. These are themselves changing, so there is little reason to tune $\beta_j^{(new)}$ to high precision. We simply want to decrease $g()$, and thus decrease $-l(\boldsymbol{\beta})$, before going on to the next $\beta_{j'}$. As long as $-l(\boldsymbol{\beta})$ decreases on each step we are guaranteed to converge to the optimum (Luenberger (sec 7.9), 1984).

Summarizing the above, Figure 4 presents pseudocode for our implementation of CLG. One issue we have not yet discussed is the convergence criterion, i.e. when to stop cycling through the parameters. Zhang and Oles describe several possibilities, including stopping when $(\sum_{i=1}^n |\Delta r_i|)/(1 + \sum_{i=1}^n |r_i|) \leq 0.001$. In other words, at the end of each pass through the coordinates, one measures the total change in the linear scores of the training examples between the beginning and end of the pass. The algorithm terminates if that change is small either in absolute terms or as a fraction of the magnitude of the linear scores. We used this criterion, but with a tolerance of 0.0005 instead of 0.001.

Some aspects of the implementation of CLG are worth noting. Like Zhang and Oles, we maintain for each training document the value, r_i, of the dot product between the current $\boldsymbol{\beta}$ and $\boldsymbol{x_i}$. Each time a parameter β_j in $\boldsymbol{\beta}$ is updated, the affected values of r_i are updated. Note that only the r_i's for examples where $x_{ij} \neq 0$ are affected by a change in β_j. Since the data is sparse, i.e. the vast majority of x_{ij} are equal to 0, we make these updates very efficient by storing the documents in inverted form. This representation can be thought of as an array of triples (j, i, x_{ij}), sorted by j, containing entries for all and only the nonzero values of x_{ij}.

4.3 Modifying CLG for the Laplace Prior

We modified the CLG algorithm for fitting a lasso logistic regression model. With the Laplace prior the negative log-posterior (Equation 11) is convex. Unlike the case of the Gaussian prior, the log-posterior function lacks finite first and second derivatives when one or more β_j's are 0, so some special handling is necessary.

With the Laplace prior, the Zhang-Oles tentative update is defined only for $\beta_j \neq 0$ and is:

$$\Delta v_j = \frac{\sum_{i=1}^{n} x_{ij} y_i \frac{1}{1+\exp(r_i)} - \lambda_j \operatorname{sgn}(\beta_j)}{\sum_{i=1}^{n} x_{ij}^2 F(r_i, \Delta_j x_{ij})} \tag{28}$$

where $\operatorname{sgn}(\beta_j)$ is $+1$ for $\beta_j > 0$ and -1 for $\beta_j < 0$.

Besides being undefined at $\beta_j = 0$, this update is invalid if it would change the sign of β_j. (The necessary condition on $F()$ does not hold for intervals that span 0.)

We solve the second problem simply by setting $\beta_j^{(new)}$ to 0 if the update would otherwise change its sign. To deal with the first problem, when the starting value of β_j is 0 we attempt an update in both directions and see if either succeeds. That is, if setting $\operatorname{sgn}(\beta_j)$ to $+1$ in Equation 28 yields a $\Delta v_j > 0$, or setting $\operatorname{sgn}(\beta_j)$ to -1 yields a $\Delta v_j < 0$, we accept the corresponding update. Otherwise, we keep β_j at 0. Note that at most one update direction can be successful, due to the convexity of $g()$.

The resulting **CLG-lasso** algorithm is identical to CLG (Figure 4) except that the computation of Δv_j is done as in Figure 5.

Algorithm 2 (Computation of Δv_j in *CLG-lasso*)

(4.1) **if** $\beta_j = 0$
(4.2) $s \leftarrow 1$ (try positive direction)
(4.3) compute Δv_j by formula (28)
(4.4) **if** $\Delta v_j \leq 0$ (positive direction failed)
(4.5) $s \leftarrow -1$ (try negative direction)
(4.6) compute Δv_j by formula (28)
(4.7) **if** $\Delta v_j \geq 0$ (negative direction failed)
(4.8) $\Delta v_j \leftarrow 0$
(4.9) **endif**
(4.10) **endif**
(4.11) **else**
(4.12) $s \leftarrow \beta_j / |\beta_j|$
(4.13) compute Δv_j by formula (28)
(4.14) **if** $s(\beta_j + \Delta v_j) < 0$ (cross over zero)
(4.15) $\Delta v_j \leftarrow -\beta_j$
(4.16) **endif**
(4.17) **endif**

Figure 5: Computation of Δv_j in the CLG-lasso algorithm.

4.4 Selecting the Hyperparameter

The Gaussian and Laplace priors both require we choose a prior variance, σ_j^2, for parameter values. (The actual hyperparameters for the distributions are $\tau_j = \sigma_j^2$ for the Gaussian, and $\lambda_j = \sqrt{2}/\sigma_j$ for the Laplace.) We tried two approaches to setting the prior variance. The first was simply:

$$\sigma_j^2 = \frac{d}{u} = \frac{d}{\frac{\sum_{i=1}^{n} ||\boldsymbol{x_i}||_2}{n}}. \tag{29}$$

where d is the number of predictor features (plus 1 for the constant term) and u is the mean squared Euclidean norm of the training examples (after feature selection, if any). This heuristic was loosely inspired by a similar heuristic used to choose the regularization parameter in *SVM_Light* (Section 5.3.1). We call the value chosen this way the *norm-based* value of the hyperparameter.

We also tested choosing the hyperparameter by partial 10-fold cross-validation on the training set. For each category, we randomly separated the training set into 10 portions, and did two runs of training on 9 portions and testing on the 10th (validation) portion.[5] In each run we tested values for the Laplace hyperparameter λ_j from the range 0.01 to 316 by multiples of $\sqrt{10}$, or values for the Gaussian hyperparameter τ_j from the range 0.0001 to 10000 by multiples of 10. For each choice of the hyperparameter, we computed the sum of log-likelihoods for the documents in the validation portions, and chose the hyperparameter value that maximized this sum. We call this the *cross-validated* value of the hyperparameter.

4.5 Implementation

We have implemented the above algorithms in C++ as the program "Bayesian Binary Regression (BBR)." The software is available at `http://stat.rutgers.edu/~madigan/BBR`. An extension to polychotomous logistic regression will appear on the website shortly.

[5] Preliminary experiments showed using two folds gave results very similar to those using all 10 folds, and was of course 5 times faster.

5 Experimental Methods

To study whether lasso logistic regression provides an efficient and effective text categorization approach, we tested it on several large data sets. In this section we discuss our experimental methods: how texts were represented as numeric vectors, what collections of documents and category labels were used, and how effectiveness was measured (and optimized). We also discuss two state-of-the-art text categorization approaches with which we compared lasso logistic regression: support vector machines, and ridge logistic regression combined with feature selection.

5.1 Text Representation

Representing a document for statistical classification has two major aspects: text processing and term weighting. Text processing involves breaking the character string up into *tokens*, i.e. individual terms such as words or phrases. We used very simple methods as described below.

Term weighting methods tabulate the number of occurrences of each distinct term in a document and across documents. They compute a numeric term weight for each term with respect to each document. For training and classification purposes, we represent each document as a vector of such term weights.

We computed term weights using a form of $TF \times IDF$ (term frequency times inverse document frequency) weighting with cosine normalization (Salton and Buckley, 1988). This gives term j in document i an initial unnormalized weight of

$$x_{ij}^u = \begin{cases} 0, & \text{if } n(i,j) = 0 \\ (1 + \ln n(i,j)) \ln \frac{|\mathcal{D}|+1}{n(j)+1}, & \text{otherwise,} \end{cases}$$

where $n(j)$ is the number of training documents that contain term j, $n(i,j)$ is the number of occurrences of term j in document i, and $|\mathcal{D}|$ is the total number of training documents.[6]. All experiments used a separation of the data into training and test sets, and all IDF weights were computed on the training set.

[6]Incrementing the numerator and denominator by 1 is a slight variant on the usual IDF weighting. We make this change so that terms occurring only on the test set have a defined IDF value.

Weights computed as above would be strongly affected by document length. So-called "cosine normalization" results in feature vectors with a Euclidean norm of 1.0. The resulting normalized final weights are:

$$x_{ij} = \frac{x_{ij}^u}{\sqrt{\sum_{j'} x_{ij'}^u \times x_{ij'}^u}}$$

The summation in the denominator is over all terms in the corpus, not just those that occur in the training set. Note that for any given document, the vast majority of terms j will have a weight $x_{ij} = 0$. Our software therefore stores documents sparsely.

In addition to one parameter for each term, our vectors include a constant term, 1.0. (The constant term is not taken into account during length normalization, and is not changed by it.) The presence of a constant term is usual in statistical practice, but often omitted in text categorization studies. We use a Gaussian or Laplace prior on the model parameter for the constant term, just as with the other model parameters.

5.2 Datasets

Our experiments used three standard text categorization test collections.

5.2.1 ModApte

Our first collection was the ModApte subset of the Reuters−21578 collection of news stories (Lewis, 2004).[7] The ModApte subset contains 9,603 Reuters news articles in the training set, and 3,299 in the test set. Documents in the ModApte data set belong to 0 or more of a set of 90 "Topic" categories corresponding to news areas of economic interest. (We used the 90 Topic categories that have at least one positive training example and one positive test example on the ModApte subset.)

Text processing was done using Lemur toolkit[8], which performed a simple tokenization into words (using its TrecParser module), discarded words from the SMART stopword list of 572 words[9], and applied the Lemur variant of the Porter stemmer (Porter, 1980, 2003) to

[7] *http://www.daviddlewis.com/resources/testcollections/reuters21578/*

[8] *http://www-2.cs.cmu.edu/~lemur/*

[9] Available at *ftp://ftp.cs.cornell.edu/pub/smart/english.stop* or as part of the RCV1-v2 data set.

remove word endings. All stems from text in the <TITLE> and <BODY> SGML elements were combined to produce raw TF weights. There were 21,989 unique terms in the ModApte data set, 18,978 of which occur in the training set and thus potentially have nonzero parameters in a classifier.

5.2.2 RCV1-v2

The second data set was RCV1-v2[10], a test categorization test collection of 804,414 newswire stories based on data released by Reuters, Ltd.[11] We used the LYRL2004 training/test split (Lewis, *et al.*, 2004) of RCV1-v2, which has 23,149 training documents and 781,265 test documents. However, for efficiency reasons we used a fixed, random, roughly 10% subset (77,993 documents) of the test documents as our test set in all experiments.

We used all 103 RCV1-v2 "Topic" categories in our experiments. (The Topic categories in RCV1-v2 are different from those in Reuters-21578, and cover a broader range of news types.) Two of the categories have no positive training examples, and so a default classifier that predicts "No" for all test documents was assumed. Classifiers were trained for the other 101 categories. Each RCV1-v2 document is known to belong to at least one of the remaining 101 categories, but we did not use that fact in classification.

For our text representation we used stemmed token files distributed with RCV1-v2. A total of 47,152 unique terms were present in the training set, and 288,062 unique terms in the union of the training set and the 77,993 document test set. We computed term weights as described above, rather than using the vector files distributed with RCV1-v2.

5.2.3 OHSUMED

The third data set consists of Medline records from the years 1987 to 1991, formatted for the SMART retrieval system, and distributed as part of the OHSUMED text retrieval test collection (Hersh, *et al.*, 1994).[12] Of the 348,566 OHSUMED records, we used the 233,445 records where the title, abstract, and MeSH (Medical Subject Headings) category fields were

[10] *http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm*
[11] *http://about.reuters.com/researchandstandards/corpus/*
[12] *ftp://medir.ohsu.edu/pub/ohsumed/*

all nonempty. We used the 83,944 such documents from the years 1987 and 1988 as our training set, and the 149,501 such documents from the years 1989 to 1991 as our test set.

We did not use the queries and relevance judgments distributed with the collection. Instead, we based binary classification tasks on predicting the presence or absence of MeSH controlled vocabulary terms in the records (Lowe and Barnett, 1994). We used the same 77 categories as Lewis, *et al.* (1996). These were drawn from a set of 119 MeSH Heart Disease categories extracted by Yiming Yang from the April 1994 (5th Ed.) UMLS CD-ROM, distributed by the National Library of Medicine in the United States.

Text processing was done by Lemur and was identical to that used with the ModApte collection. All text from the $.T$ (title) and $.W$ (abstract) fields was used in computing the TF weights. There were 73,269 distinct terms in the training set, and 122,076 in the union of the training and test sets.

5.3 Benchmark Algorithms

We compared the effectiveness of lasso logistic regression to two state-of-the-art approaches to text categorization.

5.3.1 Support Vector Machines

The support vector machine algorithm for learning linear classifiers has consistently been one of the most effective approaches in text categorization studies (Lewis, *et al.*, 2004). It also produces models with a form of instance sparseness that may or may not translate into sparseness of linear model coefficients.

We trained a single SVM classifier for each category using Version 5.00 of *SVM_Light* (Joachims, 1998, Joachims, 2002).[13] All software parameters were left at default values. This meant, in particular, that we used a linear kernel (by leaving *-t* unspecified), equal weighting of all examples whether positive or negative (by leaving *-j* unspecified), and set the regularization parameter C to the reciprocal of the average Euclidean norm of training examples (by leaving *-c* unspecified). Since we were using cosine-normalized training exam-

[13] *http://svmlight.joachims.org/*

ples, leaving -c unspecified meant C was set to 1.0 when no feature selection was used, and some smaller value in feature selection experiments.

5.3.2 Logistic Regression with Feature Selection

To produce sparse text classifiers when using ridge logistic regression, it is common to attempt to discard low quality features before model fitting. We tested three traditional feature selection methods in combination with ridge logistic regression, as well as trying no feature selection whatsoever.

Each feature selection approach is based on computing some quality measure for each feature, ranking features by that measure, and then using only the top-ranked features when learning a classifier. A different set of features was chosen for each category. We tested each method at choosing feature sets with 5, 50, or 500 features, with the same number of features used for all categories. The constant term was always used, so the total number of model parameters was 6, 51, and 501, respectively. We did not test more computationally expensive approaches to choosing the number of features, such as cross-validation.

The first feature quality measure was the chi-square test for independence between two variables. As a feature selection measure, it chooses the features that are least independent from the class label, and is widely used in text categorization (Yang and Pedersen, 1997, Sebastiani, 2002).

The chi-square measure is based on a 2×2 contingency table between a predictor term j and a predicted category label. We let:

- a = number of training documents in category & containing term j (true positives)
- b = number in category, NOT containing j (false negatives),
- c = number NOT in category, but containing j (false positives),
- d = number NOT in category, and NOT containing j (true negatives),
- n = total number of training documents.

The chi square measure is then:

$$\chi^2 = \frac{n(ad - bc)^2}{(a+b)(c+d)(a+c)(b+d)}. \tag{30}$$

Our second measure, bi-normal separation (BNS), was the best measure on several criteria in a recent comparison of feature selection methods for text categorization (Forman, 2003). Forman defines it as:

$$B(j) = |\Phi^{-1}(\frac{a}{a+b}) - \Phi^{-1}(\frac{c}{c+d})| \tag{31}$$

where Φ is the standard normal cumulative distribution function. We used Ackham's algorithm to compute Φ^{-1} (Ackham, 2004) and, as suggested by Forman, replaced values of $\Phi^{-1}(0)$ by 0.0005. Forman provides a justification of BNS in term of ROC (receiver operating characteristic) analysis.

Most feature selection measures studied in text classification research (including the two discussed above) take into account only the binary presence or absence of terms in documents. In contrast, the most effective text representations are non-binary ones such as $TF \times IDF$ weighting. Our third measure, the Pearson product-moment correlation, makes use of the values of within document weights in choosing features. The measure is:

$$r_j = \frac{\sum_{i=1}^{n}(x_{ij} - \overline{x^{(j)}})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_{ij} - \overline{x^{(j)}})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}, \tag{32}$$

where $\overline{x^{(j)}}$ is the mean of x_{ij} across the training documents, and $\overline{y}$ is the mean of y_i ($+1$ or -1 class labels) for the category of interest across the training documents. We use $|r_j|$ as our feature quality measure.

The Pearson product-moment correlation measures the degree to which there is a linear relationship between the x_{ij}'s for some term j and the y_i's for the category of interest. It has been used for feature selection in a variety of machine learning tasks.

5.4 Effectiveness Measure

After training a classifier for a category on a particular training set, we evaluate it on the corresponding test set. We can describe a classifier's behavior on a test set in terms of a contingency table, much as for feature selection:

- *true positives* = number of documents in category on which classifier says "yes" (puts the document in category)
- *false negatives* = number in category, but classifier says "no"
- *false positives* = number NOT in category, but classifier says "yes"
- *true negatives* = number NOT in category, and classifier says "no"

One effectiveness measure is the number of test set errors (*false negatives* + *false positives*) or, if normalized, error rate:

$$E = \frac{\textit{false positives} + \textit{false negatives}}{|\mathcal{T}|} \tag{33}$$

where $|\mathcal{T}|$ is the size of the test set.

We also measured effectiveness of our classifiers using Van Rijsbergen's F measure (Van Rijsbergen, 1972, 1979, Lewis, 1995). We set the F measure's parameter β (not to be confused with the β's of our logistic models) to 1.0, giving a measure $F1$:

$$F1 = \frac{2 \times \textit{true positives}}{2 \times \textit{true positives} + \textit{false positives} + \textit{false negatives}} \tag{34}$$

We define $F1$ to have the value 1.0 if the denominator is 0, which can happen if no test set documents belong to the class of interest.

F1 is also equal to the harmonic mean of recall (R) and precision (P), a pair of measures that are also widely used in information retrieval:

$$R = \frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}}, \tag{35}$$

$$P = \frac{\textit{true positives}}{\textit{true positives} + \textit{false positives}}, \tag{36}$$

$$F1 = \frac{2RP}{R+P}. \tag{37}$$

We will sometimes report the unweighted arithmetic mean of $F1$ across the categories for a particular test collection, i.e. the *macroaveraged* $F1$. (Macroaveraged number of errors was not computed, since it is dominated by the few most common categories.)

5.5 Threshold Selection

Logistic regression models output not a class label, but an estimate of $p(y = +1|\boldsymbol{\beta}, \boldsymbol{x_i})$, the probability that vector $\boldsymbol{x_i}$ belongs to the category of interest. To use a logistic model as a classifier, we must convert these estimates to binary class label predictions. The simplest approach is predicting $y = +1$ (assign the category) when $p(y = +1|\boldsymbol{\beta}, \boldsymbol{x_i}) \geq 0.5$. This minimizes expected test set error rate under the assumption that the predicted probabilities are the actual probabilities used in a probabilistic labeling. We refer to 0.5 as the *default* threshold for logistic regression classifiers.

Becuase of possible model inadequacies, there are no guarantees, theoretical or otherwise, that the estimated predicted probabilities are well calibrated. Hence we also considered a per-category tuned threshold that minimizes the number of training set errors. We call this the *tuned* threshold.

Interestingly, the optimal expected test set $F1$ value cannot be achieved by any single pre-set threshold for all test sets (Lewis, 1995). It can be achieved if the set of predicted probabilities on the test set is processed as a whole, but this complexity would take us away from our main concerns. Therefore, we simply used the same thresholds when evaluating by $F1$ that we did when evaluating by number of test set errors.

For linear models produced by *SVM_Light* we tested both the default threshold (0.0), and a tuned threshold chosen by minimizing number of training set errors.

6 Results

Our first question was whether lasso logistic regression yields state-of-the art text categorization effectiveness. After demonstrating that, we conducted experiments focused on hyperparameter selection. Finally, we tested lasso against feature selection for producing not only effective, but sparse classifiers.

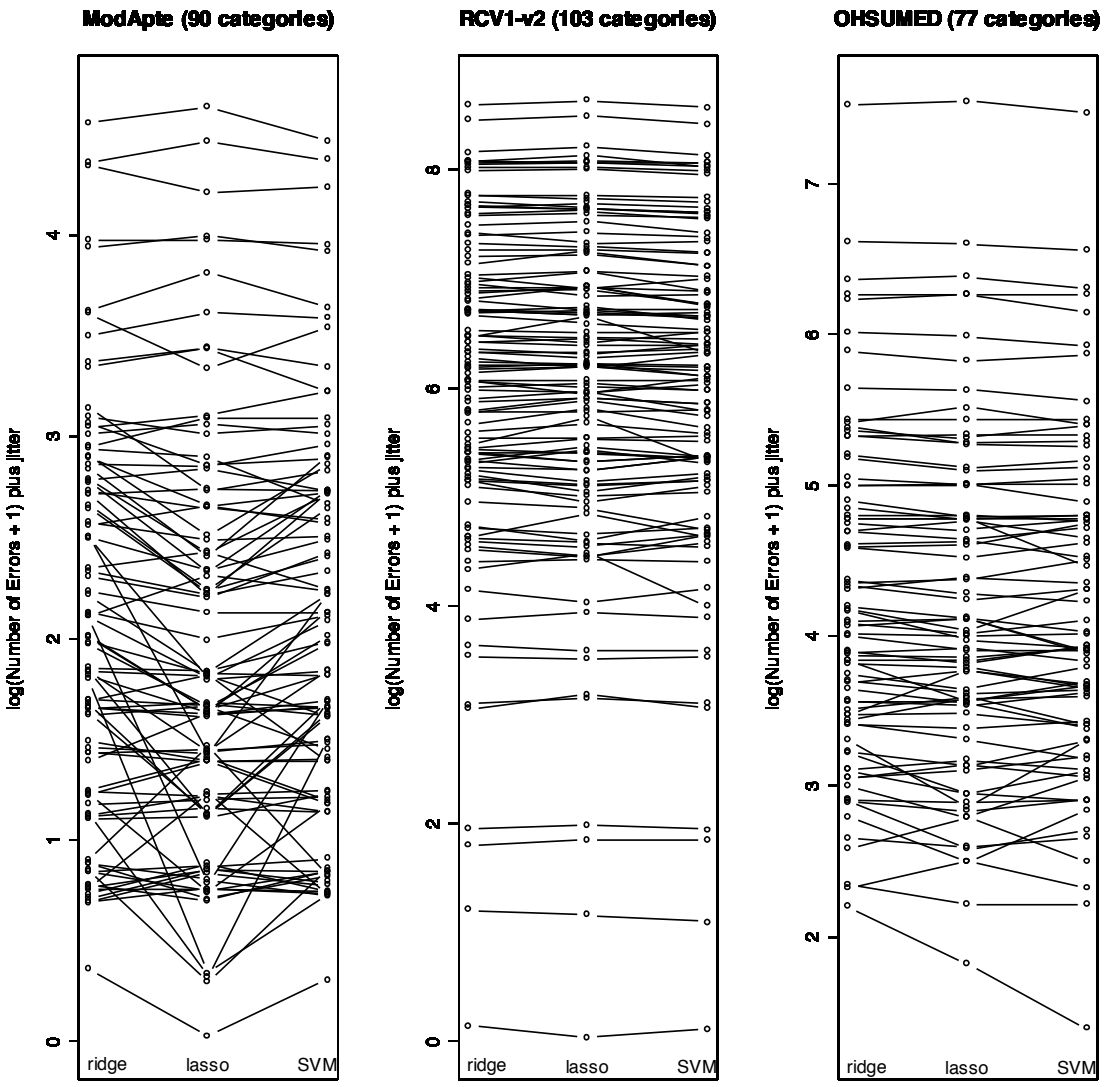

Figure 6: *Parallel coordinate plots of test errors for each category within each of the three datasets (ModApte, RCV1-v2, and OHSUMED) as provided by the three algorithms (ridge, lasso, and SVM). The vertical axis in each panel shows the log of 1 plus the number of errors plus some small random unform jitter to visually separate the categories.*

left vs right	left has fewer errors than right	left has more errors than right	equal	p-value (Wilcoxon)
lasso vs ridge	39	20	31	.005
lasso vs SVM	36	22	32	.044
ridge vs **SVM**	12	31	47	.005

Table 1: Comparison of lasso logistic regression, ridge logistic regression, and support vector machines on ModApte corpus. We show number of categories on which the first method has fewer, greater, or the same number of test set errors as the second method. All methods were used without feature selection, and with a tuned threshold. Lasso and ridge logistic regression used a hyperparameter chosen by cross-validation on the training set. SVMs used the *SVM_Light* default regularization parameter. Significance values are based on the two-sided Wilcoxon matched-pairs signed-ranks test. Algorithms significantly better at the $p = 0.05$ level are indicated in bold.

left vs right	left has fewer errors than right	left has more errors than right	equal	p-value (Wilcoxon)
lasso vs. ridge	48	48	7	.321
lasso vs. **SVM**	32	66	5	.000
ridge vs. **SVM**	22	69	12	.000

Table 2: Comparing number of errors on RCV1-v2 corpus. Details as in Table 1.

left vs right	left has fewer errors than right	left has more errors than right	equal	p-value (Wilcoxon)
lasso vs. ridge	45	26	6	.004
lasso vs. SVM	37	34	6	.705
ridge vs. **SVM**	25	40	12	.002

Table 3: Comparing number of errors on OHSUMED corpus. Details as in Table 1.

	threshold	ModApte	RCV1-v2	OHSUMED
lasso	default	48.64	53.69	47.23
ridge	default	37.63	46.79	36.09
SVM	default	38.52	46.43	34.32
lasso	tuned	52.03	56.54	51.30
ridge	tuned	39.71	51.40	42.99
SVM	tuned	53.75	57.23	50.58

Table 4: Macroaveraged $F1$ measure for lasso logistic regression, ridge logistic regression, and support vector machines on three text categorization test collections. All methods were used without feature selection. Lasso and ridge logistic regression used a hyperparameter tuned by cross-validation; SVMs used their default regularization parameter. Values are shown for both default and tuned thresholds.

6.1 Effectiveness with Full Feature Set

Our first set of experiments compared the effectiveness of lasso logistic regression with ridge logistic regression and SVMs. Each algorithm used all features as inputs, a regularization parameter tuned by cross-validation on the training set, and default thresholds. For each category we computed the number of test set errors for each algorithm. Figure 6 uses a a parallel coordinates plot to show the results. Each panel corresponds to a particular dataset. Within each panel, each connected triple of points concerns the number of errors made by ridge, lasso, and SVM respectively for a particular category. Specifically, we plot the logarithm of one plus the number of errors, adding some small random jitter to separate the categories. Perhaps the most striking feature of this plot is the similarity of the three algorithms for almost all categories on all three datasets.

Tables 1, 2, and 3 provide a different perspective and show the number of categories on which one algorithm had fewer, more, or the same number of errors as an alternative algorithm.

To provide an approximate significance test, we looked at the category-wise difference in number of errors between pairs of algorithms for each data set and applied the 2-tailed Wilcoxon matched-pairs signed-ranks test. By this measure, lasso logistic regression was significantly better than ridge logistic regression on the ModApte and OHSUMED datasets ($p = 0.005$ and 0.004), and tied on the RCV1-v2 data set. Lasso logistic regression was sig-

nificantly better than SVMs on ModApte ($p = 0.044$), insignificantly better on OHSUMED, and significantly worse ($p < 0.001$) on RCV1-v2.

To get a sense of overall effectiveness, we computed macroaveraged $F1$ for each algorithm/collection pair. Table 4 shows the results. This measure shows lasso logistic regression essentially tied with SVMs when tuned thresholds are used, and superior when untuned thresholds are used. Ridge logistic regression is considerably inferior in all cases.

6.2 Hyperparameter Selection

We tested a norm-based hyperparameter inspired by SVMs, as discussed in Section 4.4. Tables 5, 6, and 7 compare macroaveraged $F1$ values for lasso logistic regression with norm-based and cross-validated hyperparameter, and the same for ridge logistic regression. Effectiveness does not vary much, nor in a consistent direction, between the two methods for either full or reduced (Section 6.3) feature sets. The norm-based hyperparameter is therefore a plausible alternative both to cross-validation and to the Jeffreys prior (Section 3.3).

6.3 Comparison with Feature Selection

		hyperparameter	
	# features	CV	norm
lasso	all (18,978)	48.64	50.77
ridge	all (18,978)	37.63	38.94
ridge	500	39.97	38.82
ridge	50	45.73	45.80
ridge	5	46.67	46.20

Table 5: Comparing hyperparameters chosen by training set cross-validation with ones chosen using the norm-based heuristic. Macroaveraged $F1$ values for the ModApte collection are given for lasso logistic regression with the full feature set and ridge logistic regression with either the full feature set, or category specific feature sets of 5, 50, or 500 features chosen by the BNS heuristic. The default threshold is used for all classifiers.

	# features	hyperparameter CV	norm
lasso	all (47152)	53.69	52.94
ridge	all (47152)	46.79	48.12
ridge	500	48.54	46.27
ridge	50	43.43	41.69
ridge	5	30.33	28.54

Table 6: Comparing cross-validated and norm-based hyperparameter selection on the RCV1-v2 collection. See Table 5 for details.

	# features	hyperparameter CV	norm
lasso	all (122076)	47.23	47.16
ridge	all (122076)	36.09	40.47
ridge	500	39.02	36.93
ridge	50	42.88	42.59
ridge	5	43.57	41.33

Table 7: Comparing cross-validated and norm-based hyperparameter selection on the OHSUMED collection. See Table 5 for details.

Ridge logistic regression did poorly on our full feature sets. In practice, however, analysts often use ridge regression in combination with feature selection, improving both the effectiveness and sparsity of the model. We therefore chose feature sets of size 5, 50, 500 terms for each category using each of our three feature selection methods. (These feature set sizes are ones typically used in text categorization studies.) We then fit ridge logistic regression models, using these feature sets. For completeness, we fit lasso logistic regression and SVM models on these feature sets as well. Default thresholds were used, and both norm-based and cross-validated hyperparameter settings. To summarize the results:

- As expected, feature selection often (though not always) improved the effectiveness of ridge logistic regression. The BNS technique consistently had a small advantage over the other methods, as measured by macroaveraged $F1$ (Tables 5, 6, and 7). With BNS cross-validated hyperparameter selection, the best feature set sizes were 5 features (ModApte and OHSUMED) or 500 features (RCV1-v2). With BNS and norm-based hyperparameters, the best were 5 features (ModApte), 50 features (OHSUMED), or the full feature set (RCV1-v2). In no case, however, did the use of feature selection increase the effectiveness of ridge regression to beat that of lasso regression .

- In no case did feature selection improve the effectiveness (measured by macroaveraged $F1$) of lasso logistic regression with either cross-validated or norm-based hyperparameters. The lasso algorithm seems remarkably good at choosing a feature subset from a large collection of potential features.

- While not a focus of our experiments, we note that variable selection sometimes improved the results of SVMs with default thresholds, but never the results of SVM with tuned thresholds. This suggests that default SVM regularization is itself quite effective as long as appropriate thresholding is used.

A possible criticism of our results is that feature selection might be more effective if a different number of features were chosen for each category. This may be true, but most feature selection methods studied for text categorization do not give a robust way to choose this number, short of computationally expensive methods such as stepwise selection or cross validation. From this standpoint, our results on norm-based hyperparameter selection for lasso logistic regression are particularly interesting. No cross validation is used when fitting these models, but a different (and clearly effective) number of features is implicitly chosen for each category.

The number of features selected by the lasso model (starting with the full feature set) varied from category to category, but was always a small proportion of the full feature set. The number of selected features with the norm-based hyperparameter setting ranged from 2 to 892 (mean 93.8, standard deviation 167.9) for ModApte, from 0 to 3750 (mean 625.9, std. dev. 726.9) for RCV1-v2, and from 0 to 2636 (mean 241.3, std. dev. 393.3) for OHSUMED. Figure 7 provides more details.

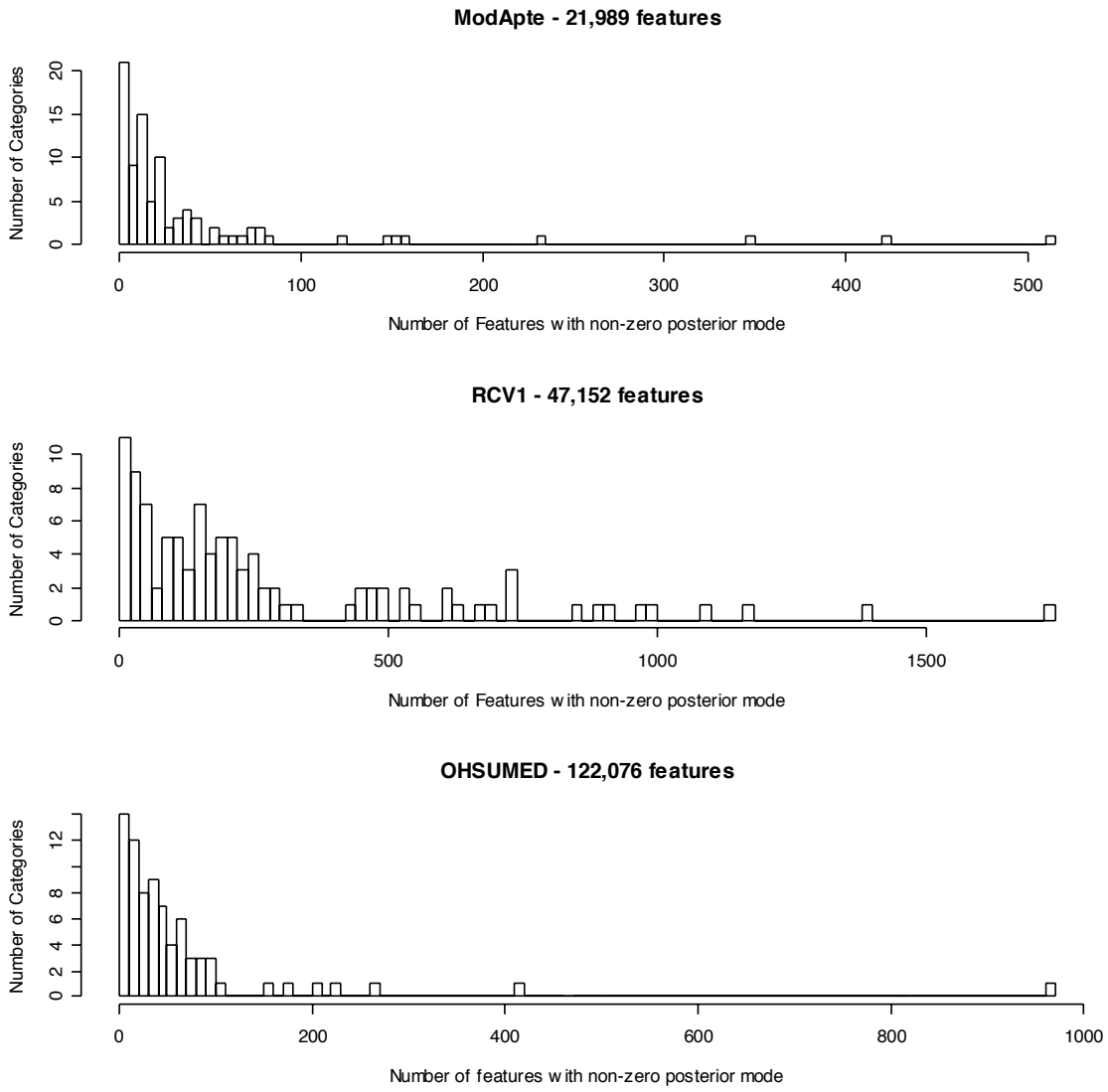

Figure 7: *Number of features that the lasso selected for each of the three datasets (ModApte, RCV1-v2, and OHSUMED).*

The number of selected features was a factor of 2 to 3 lower with cross-validated hyperpara-

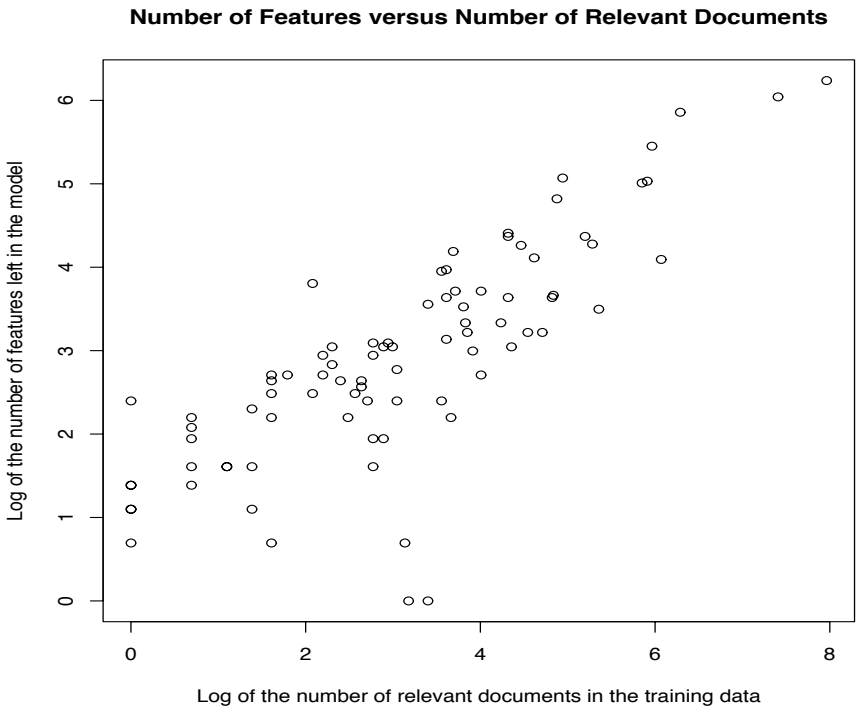

Figure 8: Relationship between number of relevant training documents for a category (out of 9,603 total ModApte training documents) and number of nonzero parameters in the fitted lasso logistic regression model. Hyperparameter was chosen by cross-validation. Both axes use logarithmic (base e) scales.

meter settings than with the norm-based hyperparameter, so cross-validation is still giving an advantage from the standpoint of sparseness. Another observation was that while the parameter for the constant term was penalized in the same fashion as other parameters, in no case did lasso zero it out.

There was a strong correlation between the number of positive examples of a category in the training set, and the number of features chosen. Figure 8 shows the relationship for the 90 categories on the 9,603 training examples of the ModApte collection. The other collections show similar patterns.

7 Future Work

We have shown that lasso logistic regression provides state-of-the-art text categorization effectiveness while producing sparse, and thus more efficient to use, models. While our attention here has been on text categorization, we expect this technique will prove useful in a wide range of high dimensional data analysis problems. There are also a number of interesting directions for improvement.

Feature Set Size

The number of features left in the model by lasso, while small, varies considerably. In some applications it must be guaranteed that exactly (or, more commonly, no more than) a particular number of features be used. For these cases, we have developed an algorithm, *Squeezer*, which uses cross-validation in choosing a hyperparameter value that obeys such restrictions. Recent work by Friedman and Popescu (2004) suggests a more elegant alternative may be possible, with regularization tightened smoothly during a single optimization run until the desired number of features is reached.

Prior Knowledge

By achieving the equivalent of feature selection within the context of a well-founded Bayesian model, the lasso approach provides great scope for combining domain knowledge with learning from training data. The simplest case would be to give rare words priors with a higher variance, reflecting that they have higher content than more common words. This would be a more principled alternative to the usual term weighting heuristics (Section 5.1).

If particular words are known to be positively correlated with a particular category, this could be encoded in the mean of a Laplace prior. If this was done for many words, however, the resulting classifier would no longer be sparse, even if most of the words in practice were not useful. An alternative would be to use a simple exponential prior, forcing the β_j for the word to be nonnegative, but allowing it to be 0 if it does not demonstrate its usefulness. Such an approach would be particularly desirable when models must be understand by nonexperts, who may be disturbed by incorrect signs on parameters, regardless of predictive accuracy.

Polychotomous Classification

The most compelling application of lasso logistic regression may be in polychotomous classification problems, where documents (or other objects) must be assigned to exactly one of three or more classes. Most existing approaches to these problems involve building multiple binary classification models and combining their results. Besides being computationally expensive, such an approach makes it difficult to incorporate prior knowledge or to use the model to gain insight into the data. Polychotomous logistic regression captures all relationships between predictors and classes in a single model, and priors may be used to incorporate domain knowledge.

The benefits of sparsifying priors are particularly desirable in the polychotomous case. Consider a 5-way classification problem, and a feature that has a strong positive correlation with one class and weak negative correlations with the others. Ridge polychotomous logistic regression would fit one parameter with positive sign for the feature, and four with negative sign, while lasso would almost always give one positive parameter and zero out the other four. As the number of classes increases, the benefits of lasso for interpretability and efficiency will as well. We will describe extensions to polychotmous logistic regression in a separate paper.

Acknowledgements

We thank Reuters, Ltd., Bill Hersh, and the National Library of Medicine for making available data sets used in this research. The KD-D group supported this work through National Science Foundation grant "Monitoring Message Streams" EIA-0087022 to Rutgers University. The NSF also partially supported Madigan and Lewis's work through ITR grant DMS-0113236. We thank Andrei Anghelescu, Susana Eyeramendy, Paul Kantor, Vladimir Menkov, and Fred Roberts for suggestions, comments, or help with data sets.

References

Ackham, P. J. (2004). An algorithm for computing the inverse normal cumulative distribution function. Updated 2004-05-04. *http://home.online.no/~pjacklam/notes/invnorm/.*

Bazaraa, M. S. and Shetty, C. M. (1979). Nonlinear Programming: Theory and Algorithms. Wiley, New York.

Bishop, C. M. (1995). Neural Networks for Pattern Recognition. Clarendon Press, Oxford.

Chai, K. M. A., Ng, H. T., and Chieu, H. L. (2002). Bayesian Online Classifiers for Text Classification and Filtering. *Proceedings of SIGIR 2002: The Twenty-Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 97–104.

Dennis, Jr., J. E., and Schnabel, R. B. (1989). A View of Unconstrained Optimization. In: G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, eds. *Optimization*. Elsevier, Amsterdam.

Duda, R. O. and Hart, P. E. (1973). Pattern Classification and Scene Analysis. Wiley-Interscience, New York.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2003). Least angle regression. *Annals of Statistics*, to appear.

Figueiredo, M.A.T. (2001). Adaptive sparseness using Jeffreys prior. *Neural Information Processing Systems*, Vancouver, December 2001.

Figueiredo, M.A.T. and Jain, A. K. (2001). Bayesian learning of sparse classifiers. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR 2001*, Hawaii, December 2001.

Figueiredo, M. A. T. (2003). Adaptive Sparseness for Supervised Learning. *IEEE Transactions and Pattern Analysis and Machine Intelligence*, 25(9), 1150–1159.

Forman, G. (2003) An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, **3**, 1289–1305.

Friedman, J. H. and Popescu, B. E. (2004). Gradient directed regularization for linear regression and classification. Manuscript. *http://www-stat.stanford.edu/~jhf/ftp/path.pdf*.

Genkin, A., Lewis, D. D., Eyheramendy, S. A., Ju, W-H., and Madigan, D. (2003). Sparse Bayesian classifiers for text categorization. Submitted for publication.
http://www.stat.rutgers.edu/∼madigan/PAPERS/jicrd-v13.pdf

Girosi, F. (1998). An equivlance between sparse approximation and support vector machines. *Neural Computation*, **10**, 1445–1480.

Greenland, S., Schwartzbaum, J.A., and Finkle, W.D. (2000). Problems from small samples and sparse data in conditional logistic regression analysis. *American Journal of Epidemiology*, **151**, 531-539.

Hastie, T. J. and Pregibon, D. (1992). Generalized Linear Models. In J. M. Chambers and T. J. Hastie, *Statistical Models in S*, Wadsworth & Brooks, Pacific Grove, CA.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). The Elements of Statistical Learning: Data mining, Inference and Prediction. Springer New York.

Hersh, W., Buckley, C., Leone, T. J., and Hickman, D. (1994). OHSUMED: an interactive retrieval evaluation and new large test collection for research. In SIGIR '94, pp. 192–201.

Hoerl, A.E. and Kennard, R.W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, **12**, 55–67.

Ishwaran, H. and Rao, J.S. (2003). Detecting differentially expressed genes in microrarrays using Bayesian model selection. *Journal of the American Statistical Association*, **98**, 438–455.

Jin, R., Yan, R., Zhang, J., and Hauptmann, A. A Faster Iterative Scaling Algorithm for Conditional Exponential Model. *International Conference on Machine Learning (ICML 2003)*.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Machine Learning: ECML'98, 10th European Conference on Machine Learning*, 137–142.

Joachims, T. (2002). Learning to Classify Text Using Support Vector Machines. Kluwer, Boston.

Kittler, J. (1986) Feature Selection and Extraction. In: T. Y. Young and K.-S. Fu, eds., *Handbook of Pattern Recognition and Image Processing*, Academic Press, Orlando, 59–83.

Kivinen, J. and Warmuth, M. K. (2001). Relative Loss Bounds for Multidimensional Regression Problems. *Machine Learning*, 45(3), 301–329.

Komarek, P. and Moore, A. (2003). Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs. *Proceedings of the Ninth International Workshop on Ar-*

tificial Intelligence and Statistics.

le Cessie, S. and van Houwelingen, J.C. Ridge Estimators in Logistic Regression. *Applied Statistics*, **41**, 191–201, 1997.

Lewis, D. D. (1995). Evaluating and Optimizing Autonomous Text Classification Systems. *SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 246–254.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *ECML'98, The Tenth European Conference on Machine Learning*, 4–15.

Lewis, D. D. (2004). Reuters-21578 text categorization test collection: Distribution 1.0 README file (v 1.3). *http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt*

Lewis, D. D., Schapire, R.E., Callan, J.P., and Papka, R. (1996). Training algorithms for linear text classifiers. *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, ACM Press, 298–306.

Lewis, D. D., Yang, Y., Rose, T., and Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 361–397.

Lowe, H. J., and Barnett, G. O. (1994). Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. Journal of the American Medical Association, 271(14),1103–1108.

Luenberger, D. G. (1984). Linear and Nonlinear Programming (Second Edition). Addison-Wesley, Reading, MA.

Madigan, D. and Ridgeway, G. (2004). Comment on Efron *et al.*, *Annals of Statistics*, to appear. *http://www.stat.rutgers.edu/∼madigan/PAPERS/lars3.pdf*

Mallick, B. K., Ghosh, D., and Ghosh, M. (2003). Bayesian classification of tumors using gene expression data. Technical Report.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, 49–55.

Maron, M. E. (1961). Automatic Indexing: An Experimental Inquiry. *Journal of the ACM*, 8, 404–417.

Mitchell, T.J. and Beauchamp, J.J. (1988). Bayesian Variable Selection in Linear Regression. *Journal of the American Statistical Association*, **83**, 1023–1032.

Pike, M.C., Hill, A.P. and Smith, P.G. (1980). Bias and efficiency in logistic analysis of stratified case-control studies. *American Journal of Epidemiology*, **9**, 89–95.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, **14(3)**, 130–137.

Porter, M. F. (2003). The Porter Stemming Algorithm.
http://www.tartarus.org/~martin/PorterStemmer/index.html.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). Numerical Recipes in C: The Art of Scientific Computing (Second Edition). Cambridge University Press, Cambridge, UK.

Ridgeway, G. and Madigan, D. (2003). A sequential Monte Carlo Method for Bayesian analysis of massive datasets. *Journal of Knowledge Discovery and Data Mining.* To appear.
http://www.stat.rutgers.edu/∼madigan/PAPERS/ridgeway2.pdf

Rocchio, Jr., J. J. (1971). Relevance feedback information retrieval. In: G. Salton, ed., *The SMART Retrieval System-Experiments in Automatic Document Processing.* Prentice-Hall, 313–323.

Salton, G. and Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5), 513–523.

Santner, T. and Duffy, D. (1989). *The Statistical Analysis of Discrete Data*, Springer-Verlag.

Schapire, R. E. and Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135-168.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, **34**, 1–47.

Smith, R.L. (1999). Bayesian and frequentist approaches to parametric predictive inference (with discussion). In: *Bayesian Statistics 6*, edited by J.M. Bernardo, J.O. Berger, A.P.

Dawid and A.F.M. Smith. Oxford University Press, pp. 589-612.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society (Series B)*, **58**, 267-288.

Tipping, M.E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, **1**, 211-244.

van Rijsbergen, C. J. (1972). Automatic Information Structuring and Retrieval. PhD thesis, King's College, Cambridge.

van Rijsbergen, C. J. (1979). Information Retrieval. Second edition. Butterworths, London.

Yang, Y. and Pedersen, J.O. (1997). A Comparative Study on Feature Selection in Text Categorization. *Proceedings of ICML-97, 14th International Conference on Machine Learning ICML97*, 412—420.

Zhang, J. and Yang, Y. (2003). Robustness of Regularized Linear Classification Methods in Text Categorization. *Proceedings of SIGIR 2003: The Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 190–197.

Zhang, T. and Oles, F. (2001). Text categorization based on regularized linear classifiers. *Information Retrieval*, **4**, 5–31.