Large-Scale Bayesian Logistic Regression for Text Categorization

Alexander Genkin, DIMACS, Rutgers University, Piscataway, NJ 08854
(*alexgenkin@iname.com*)

David D. Lewis, David D. Lewis Consulting, Chicago, IL 60614
(*tmpap06@DavidDLewis.com*)

David Madigan, Department of Statistics, Rutgers University, Piscataway, NJ 08854
(*dmadigan@rutgers.edu*)

Abstract

Maximum likelihood logistic regression faces mathematical and implementational problems on high-dimensional data, including natural language text. We present a simple Bayesian logistic regression approach that uses a Laplace prior to avoid overfitting and produce sparse predictive models for text data. On a range of document classification problems, this approach produces compact predictive models at least as effective as those produced by SVM classifiers or ridge logistic regression combined with feature selection. We describe our model fitting algorithm, our open source implementations (BBR and BMR), and experimental results.

KEY WORDS: Lasso; Ridge regression; Variable selection; Support vector classifier; Penalization; Information retrieval

1 INTRODUCTION

Maximum likelihood logistic regression is a mainstay for statisticians. The model enjoys a body of supporting theory and algorithms, features prominently in commercial statistical software, and its predictive accuracy is often competitive with that of newer techniques.

However, new applications have emerged that pose computational and statistical challenges to logistic regression. In these applications 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 focus in this paper), gene expression analysis, adverse event monitoring, longitudinal clinical trials, and some 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, Hill, and Smith 1980). Exact logistic regression suffers from many of the same problems (Greenland, Schwartzbaum, and Finkle 2000). Furthermore, while maximum likelihood has desirable asymptotic properties (Hadjicostas, 2003), 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, which in practice limits the number of predictor variables. While feature selection (Kittler 1986; Mitchell and Beauchamp 1988) reduces memory and computional requirements, it introduces new problems. First, the statistical foundation of most feature selection methods is unclear. This makes it difficult, for instance, to choose the number of features for a given task in a principled way. Second, the most efficient feature selection methods consider each feature in isolation, and may choose redundant or ineffective combinations of features. Finally, it is typically unclear how to combine heruistic feature selection methods methods with, for instance, domain knowledge.

We report here on a Bayesian approach to logistic regression which avoids overfitting, gives state-of-the-art effectiveness, and is 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 begin in Section 2 by describing how supervised learning is used in language processing, the motivating area for our work. In Section 3 we present the basics of our Bayesian approach to logistic regression, while Section 4 presents the fitting algorithm. Section 5 describes the data sets and methods we use in our experiments, and Section 6 our experimental results.

We find that the sparse classifiers we describe are competitive with state-of-the-art text categorization algorithms, including widely used feature selection methods. Finally, Section 7 presents directions for future work.

2 TEXT CATEGORIZATION AND STATISTICAL LANGUAGE PROCESSING

Text classification algorithms choose which of a set of classes to assign to a text. 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 more general interest (e.g. Library of Congress headings), we instead refer to *text categorization*.

The study of automated text categorization dates back more than forty years (Maron 1961). In the last decade statistical approaches have dominated the research literature and, increasingly, operational practice. Statistical supervised learning approaches approaches to text categorization induce ("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 approaches.

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). Feature selection was often used to discard most features from the document vectors.

Greater computing power and new regularization approaches now allow learning of less restricted classifiers both efficiently and with little overfitting. Example approaches include support vector machines (Joachims 1998; Zhang and Oles 2001; Lewis, Yang, Rose, and Li 2004), boosting (Schapire and Singer 2000), and ridge logistic regression (Zhang and Oles 2001). However, these methods still require feature selection and/or stopping fitting short

of convergence, if they are to produce compact (and thus efficient) classifiers.

Many of the characteristics of text categorization (e.g. short, fat, data sets with murky relationships between features and class labels) are shared by other language processing applications. Indeed, as in text categorization, statistical techniques have displaced, or at least supplemented, the once dominant knowledge engineering approach across all of computational linguistics. We briefly discuss other language processing tasks where the approach described in this paper might prove applicable.

As mentioned above, text categorization is just one of a range of text classification tasks. Supervised learning approaches have been applied to more personalized text classification tasks, such as sorting user email, and alerting users to news stories of interest. Filtering of text streams (for junk email, pornography, or proprietary information) straddle topical and personalized classification, and in some cases must deal with an adversary attempting to evade classification (Madigan 2005). Authorship attribution (Mosteller and Wallace 1964; Madigan, et al. 2005a; Madigan, Genkin, Lewis, and Fradkin, D. 2005b) is an unusual classification task where choosing training data, defining features, and even specifying the number of classes are tricky issues.

The above applications treat documents as atomic units converted to feature vectors. Other tasks process language directly as a sequence of linguistic units. The goal may be linguistic analysis, such as finding the syntactic structure of a sentence or the meaning of an amibiguous word. Or it may be application-oriented, such as finding all the names of companies in a set of documents. Still other tasks involve transducing language from one form to another, as in optical character recognition, speech recognition, machine translation, and summarization. A range of statistical approaches have been applied to such tasks, some explicitly sequential (many flavors of Markov models) and others which convert sequential data into vectors.

Good texts on statistical language processing include Manning and Schütze (1999) and Jurafsky and Martin (2000). The use of text in data mining is discussed by Weiss, Indurkhya, Zhang, and Damerau (2005) and Berry (2004).

3 THE MODEL

We are interested in learning 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). The values $y_i \in \{+1, -1\}$ are class labels encoding membership $(+1)$ or non-membership (-1) of the vector in the category.

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:

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

thereby producing a logistic regression model.

For a text categorization problem, $p(y = +1|\boldsymbol{x_i})$ will be an estimate of 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.

For a logistic regression model to make accurate predictions for future inputs, we must avoid overfitting the training data. One Bayesian approach to avoiding overfitting involves a prior distribution on $\boldsymbol{\beta}$ specifying that each β_j is likely to be near 0. We next describe several such priors.

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:

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

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 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, sec. 5.4; 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 β_j's will be nonzero. To obtain sparse text classifiers with a Gaussian prior, previous authors have used feature selection (Zhang and Oles 2001; Zhang and Yang 2003).

3.2 Laplace Priors & Lasso Logistic Regression

Next we consider priors that favor sparseness. Assume β_j arises from a Gaussian distribution with mean 0 and variance τ_j:

$$p(\beta_j|\tau_j) = N(0, \tau_j), j = 1, \ldots, d. \tag{4}$$

Further assume, *a priori*, that the τ_j's arise from an exponential distribution with density:

$$p(\tau_j|\gamma) = \frac{\gamma_j}{2} \exp(-\frac{\gamma_j}{2}\tau_j), \gamma > 0. \tag{5}$$

Integrating out τ_j then gives an equivalent nonhierarchical double exponential (Laplace) distribution with density:

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

where $\lambda_j = \sqrt{\gamma}_j > 0$. In what follows we assume that λ_j equals λ for all j. As before, the prior for $\boldsymbol{\beta}$ is the product of the priors for its components. At zero the first derivative of this density is discontinuous. The distribution has mean 0, mode 0, and variance $2/\lambda^2$.

Figures 1 and 2 show the effect of hyperparameter settings on the MAP logistic regression parameters on a particular data set with eight predictor variables. Figure 1 shows the effect of a Gaussian prior distribution on each parameter, with all Gaussians having the same

variance. When that variance is small, the resulting MAP estimates are small, approaching zero as the variance approaches zero. When that variance is large, the MAP estimates are similar to the maximum likelihood estimates. The vertical dashed line corresponds to a variance of 0.01. Figure 2 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, intermediate choices for the prior variance lead to MAP estimates where some components of $\boldsymbol{\beta}$ are zero while others are not. The vertical dashed line corresponds to a variance of 0.27, giving a posterior mode where two of the parameters are zero. Hastie, Tibshirani, and Friedman (2001) show similar plots for linear regression.

Tibshirani (1996) introduced the *lasso* for linear regression 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 observed that this was equivalent to a Bayesian MAP estimate using a Laplace prior, as presented above. Since then the use of constraints or penalties based on the absolute values of coefficients has been used to achieve sparseness in logistic regression and many other data fitting tasks (Girosi 1998; Tipping 2001; Figueiredo and Jain 2001; Figueiredo 2003; Efron, Hastie, Johnstone, and Tibshirani 2004; Madigan and Ridgeway 2004). Zou and Hastie (2005) discuss some of the limitations of the lasso and a particular generalization.

4 FINDING THE MAP ESTIMATE

Ideally we would use the posterior distribution of $\boldsymbol{\beta}$ to compute a posterior predictive distribution for a desired y_{new}, given the corresponding $\boldsymbol{x_{new}}$ (Mallick, Ghosh, and Ghosh 2005). In many practical tasks, however, efficiency requires that we base predictions on a point estimate of $\boldsymbol{\beta}$. This simplification does not necessarily lead to less accurate predictions (Smith, 1999).

For the logistic regression model, with the priors we have discussed, no inexpensive computational procedure for finding the posterior seems to exist. Hence we focus on posterior mode estimation.

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

$$L(\boldsymbol{\beta}) = p(\boldsymbol{\beta}|D) \propto (\prod_{i=1}^{n} \frac{1}{1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)}) p(\boldsymbol{\beta}) \tag{7}$$

where $p(\boldsymbol{\beta})$ is the prior on $\boldsymbol{\beta}$ and i indexes the training examples in D. For Gaussian priors with mean 0 and variance τ on the β_j's the log posterior (ignoring the normalizing constant) is given by:

$$l(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \ln(1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)) - \sum_{j=1}^{d} (\ln \sqrt{\tau} + \frac{\ln 2\pi}{2} + \frac{\beta_j^2}{2\tau}), \tag{8}$$

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

$$l(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \ln(1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)) - \sum_{j=1}^{d} (\ln 2 - \ln \lambda_j + \lambda_j |\beta_j|), \tag{9}$$

with $j = 1 \ldots d$ indexing the features in both cases. The MAP estimate is then the $\boldsymbol{\beta}$ that maximizes $l(\boldsymbol{\beta})$ (or minimizes $-l(\boldsymbol{\beta})$).

4.1 The Logistic Model from an Optimization Standpoint

The negated log-posterior for a logistic regression model is convex with either the Gaussian or Laplace prior and a wide variety of convex 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 implemented via iteratively reweighted least squares (Dennis and Schnabel 1989; Hastie and Pregibon 1992). Newton algorithms have the advantage of converging in very few iterations. Further, the matrix of second derivatives they compute has other uses, e.g. in finding asymptotic confidence intervals for parameters.

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 and maximum *a posteriori* logistic regression in the large d case (Kivinen and Warmuth 2001; Zhang and Oles 2001; Malouf 2002; Jin, Yan, Zhang, and Hauptmann 2003; Komarek and Moore 2003).

4.2 The CLG Algorithm for Ridge Logistic Regression

We based our implementation on CLG (Zhang and Oles 2001), a *cyclic coordinate descent* (Luenberger 1984, sec. 7.9) optimization algorithm tuned for fitting a logistic model with a Gaussian prior, due to its efficiency and ease of implementation. We describe the CLG algorithm in detail here.

A cyclic coordinate descent algorithm begins by setting all variables to some initial value. It then sets the first variable to a value that minimizes the objective function, holding all other variables constant. 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, the one-dimensional problems involve finding β_j^{new}, the value for the j'th parameter that gives the minimum value for $-l(\boldsymbol{\beta})$, assuming that the other $\beta_{j'}$'s are held at their current values. Finding this β_j^{new} is equivalent to finding the z that minimizes:

$$g(z) = (\sum_{i=1}^{n} f(r_i + (z - \beta_j)x_{ij}y_i)) + \frac{z^2}{2\tau_j}, \tag{10}$$

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 Gaussian penalty terms not involving z are constant and thus 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 optimization procedure must be used. Zhang and Oles use a method that is related to the one-dimensional Newton's method (Press, Teukolsky, Vetterling, and Flannery 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). This approximation is:

$$\tilde{g}(z) = g(\beta_j) + g'(\beta_j)(z - \beta_j) + \frac{1}{2}g''(\beta_j)(z - \beta_j)^2 \approx g(z), \tag{11}$$

where for a ridge logistic regression model we have:

$$g'(\beta_j) = \left.\frac{dg(z)}{dz}\right|_{z=\beta_j} = \left(\sum_{i=1}^{n} x_{ij} y_i \frac{1}{1+\exp(r_i)}\right) + \frac{\beta_j}{\tau_j} \tag{12}$$

and

$$g''(\beta_j) = \left.\frac{d^2 g(z)}{d^2 z}\right|_{z=\beta_j} = \left(\sum_{i=1}^{n} x_{ij}^2 \frac{\exp(r_i)}{(1+\exp(r_i))^2}\right) + \frac{1}{\tau_j} \tag{13}$$

The approximation in (11) with minimum at:

$$\beta_j^{(new)} = \arg\min_z g(z) = \beta_j - \frac{g'(\beta_j)}{g''(\beta_j)}. \tag{14}$$

The increment we must add to β_j to reach $\beta_j^{(new)}$ is then:

$$\Delta\beta_j = \beta_j^{(new)} - \beta_j = -\frac{g'(\beta_j)}{g''(\beta_j)}. \tag{15}$$

Note that since $\tilde{g}''(z) = g''(z)$ is strictly positive (assuming $x_{ij} \neq 0$ for some i), we know we have a minimum.

In CLG, Zhang and Oles modify the update in Equation 15 in three ways. First, as in most applications of Newton's method, convergence requires avoiding large updates in regions where a quadratic is a poor approximation to the objective. Zhang and Oles specify a value $\Delta_j > 0$ 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:

$$\Delta_j^{new} = \max(2|\Delta\beta_j|, \Delta_j/2) \tag{16}$$

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 11) as their quadratic approximation in Newton's method, CLG uses

$$\hat{g}(z) = g(\beta_j) + g'(\beta_j)(z-\beta_j) + \frac{1}{2}G(\beta_j)(z-\beta_j)^2 \approx g(z) \tag{17}$$

where

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

Zhang and Oles allow $F(r, \delta)$ to be any convenient function that satisfies (for $\delta > 0$):

$$\begin{aligned} F(r,\delta) &\geq \sup_{|\Delta r| \leq \delta} f''(r + \Delta r) \\ &= \sup_{|\Delta r| \leq \delta} \frac{\exp(r + \Delta r)}{(1 + \exp(r + \Delta r))^2}. \end{aligned}$$

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:

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

In our implementation we used:

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

Using $\hat{g}()$, the update in Equation 15 becomes:

$$\Delta v_j = -\frac{\hat{g}'(\beta_j)}{\hat{g}''(\beta_j)} = -\frac{\sum_{i=1}^{n} \frac{x_{ij} y_i}{1 + \exp(r_i)} + \beta_j/\tau_j}{(\sum_{i=1}^{n} x_{ij}^2 F(r_i, \Delta_j |x_{ij}|) + 1/\tau_j} \tag{18}$$

Applying the trust region restriction then gives the actual update used in CLG:

$$\Delta\beta_j = \begin{cases} -\Delta_j & \text{if } \Delta v_j < -\Delta_j \\ \Delta v_j & \text{if } -\Delta_j \leq \Delta v_j \leq \Delta_j \\ \Delta_j & \text{if } \Delta_j < \Delta v_j \end{cases} \tag{19}$$

Third, instead of iterating $\beta_j^{(new)} = \beta_j + \Delta\beta_j$ to convergence, CLG does this only once before going on to the 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 $\beta_{j'}$.

Summarizing, Figure 3 presents pseudocode for our implementation of CLG. Zhang and Oles present several possibilities for the convergence test. Our code declares convergence when $(\sum_{i=1}^{n} |\Delta r_i|)/(1+\sum_{i=1}^{n} |r_i|) <= \epsilon$, where $\sum_{i=1}^{n} |\Delta r_i|$ is sum of the changes in the linear scores of the training examples between the beginning and end of a pass through the coordinates, and ϵ is a user specified tolerance. The optimization is considered to converge if the change is small either in absolute terms or as a fraction of the magnitude of the linear scores. The experiments we report here use $\epsilon = 0.0005$, but larger values can be used with minimal impact on effectiveness.

Some implementation details 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. Only the r_i's for examples where $x_{ij} \neq 0$ are affected by a change in β_j, and since text data is sparse, the vast majority of x_{ij}'s are equal to 0. We can therefore update the r_i's very efficiently by storing the vectors in inverted index form (i.e. 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 to fit lasso logistic regression models as well. Since the log-posterior density corresponding to a Laplace prior lacks finite first and second derivatives when one or more β_j's are 0, some special handling is necessary.

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

$$\Delta v_j = \frac{\sum_{i=1}^{n} x_{ij} y_i \frac{1}{1+\exp(r_i)} - \lambda_j \operatorname{sgn}(\beta_j)}{\sum_{i=1}^{n} x_{ij}^2 F(r_i, \Delta_j x_{ij})} \tag{20}$$

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

This update has two problems: it is undefined at $\beta_j = 0$, and it is not guaranteed to decrease the objective 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 20 yields a $\Delta v_j > 0$, or setting $\text{sgn}(\beta_j)$ 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 3) except that the computation of Δv_j is done as in Figure 4.

4.4 Selecting the Hyperparameter

The Gaussian and Laplace priors both require a prior variance, σ_j^2, for parameter values. (The actual hyperparameters 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:

$$\sigma_j^2 = d/u = dn/\sum_{i=1}^{n} ||\boldsymbol{x_i}||_2 \tag{21}$$

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. (Preliminary experiments showed using two folds gave results very similar to those using all 10 folds, and was of course 5 times faster.) 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 publicly released an open-source C++ implementation of the above algorithms, Bayesian Binary Regression (BBR), at *http://stat.rutgers.edu/ madigan/BBR* and have used it in a wide variety of applications. We have also developed an extension of the algorithms to polytomous logistic regression (Madigan et al. 2005a; Madigan et al. 2005b) and provide an implementation, Bayesian Multinomial Regression (BMR), at *http://stat.rutgers.edu/ madigan/BMR*. Both BBR and BMR include an "autosearch" capability for hyperparameter selection. Similar algorithms for logistic regression have been developed by Shevade and Keerthi (2003), by Krishnapuram, Hartemink, Carin, and Figueiredo (2005), and by others.

5 METHODS

We tested lasso logistic regression on five text categorization data sets. In this section we discuss how texts were represented as numeric vectors, what documents and category labels were used, and how effectiveness was measured and tuned. We also discuss two state-of-the-art text categorization approaches used as benchmarks: 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 breaks the character string into *tokens*, i.e. individual terms such as words or phrases. We used very simple methods described below. Term weighting methods tabulate the number of occurrences of each distinct term in a document and across documents. They then compute a numeric weight for each term with respect to each document. We represent each document as a vector of such weights.

We used a form of $TF \times IDF$ (term frequency times inverse document frequency) term 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. Incrementing the IDF numerator and denominator by 1 is a variant of IDF weighting that gives terms occurring only on the test set a defined IDF value. All experiments separated the data into training and test sets, and all IDF weights were based on the training set.

Then to reduce the impact of document length, we "cosine normalize" the feature vectors to have a Euclidean norm of 1.0. The final weights are:

$$x_{ij} = \frac{x^u_{ij}}{\sqrt{\sum_{j'} x^u_{ij'} \times x^u_{ij'}}}.$$

The summation is over all terms in the corpus, but of course most $x_{ij} = 0$.

In addition to one parameter for each term, our vectors include a constant term, 1.0 (which is omitted from cosine normalization). A constant term is usual in statistical practice, but is often omitted in text categorization studies. The experiments here use a Gaussian or Laplace prior on the model parameter for the constant term, just as for the other model parameters.

5.2 Datasets

Our experiments used five standard text categorization test collections. Three of them, ModApte, RCV1-v2, and OHSUMED each contain on the order of 100 distinct binary classification problems corresponding to predicting expert human indexing decisions. The other two, WebKB Universities and 20 Newsgroups, are based on a smaller number of binary classications of uncertain quality, but are included due to wide use in published research.

5.2.1 ModApte

Our first collection was the ModApte subset of the Reuters−21578 collection of news stories (Lewis 2004) (*http://www.daviddlewis.com/resources/testcollections/reuters21578/*). The ModApte subset contains 9,603 training documents and 3,299 test documents. Documents in the ModApte data set belong to 0 or more of a set of "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 used Lemur (*http://www-2.cs.cmu.edu/~lemur/*), which performed a simple tokenization into words (using its TrecParser module), discarded words from the SMART stopword list of 572 words (available at *ftp://ftp.cs.cornell.edu/pub/smart/english.stop* or as part of the RCV1-v2 data set), and applied the Lemur variant of the Porter stemmer (Porter 1980, 2003) to 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, a test categorization test collection of 804,414 newswire stories based on data released by Reuters, Ltd. (*http://trec.nist.gov/data/reuters/reuters.html*). We used the LYRL2004 training/test split (Lewis, Yang, Rose, and Li 2004) of RCV1-v2, which has 23,149 training documents and 781,265 test documents. However, for efficiency we took 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.

Term weights were computed from 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.

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, Buckley, Leone, and Hickman 1994) (*ftp://medir.ohsu.edu/pub/ohsumed/*). Of

the 348,566 OHSUMED records, we used the 233,445 records where the title, abstract, and Medical Subject Headings (MeSH) category fields were 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. Our binary classification tasks were to predict the presence or absence of MeSH (Lowe and Barnett 1994) controlled vocabulary categories in the records. We used the same 77 categories as Lewis, Schapire, Callan, and Papka (1996).

Text processing was the same as for ModApte. All text from the *.T* (title) and *.W* (abstract) fields was used in computing TF weights. There were 73,269 distinct terms in the training set, and 122,076 in the union of the training and test sets.

5.2.4 WebKB Universities

This data set (*http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/*) contains HTML pages downloaded from university computer science departments in January 1997. The data set consists of 8,282 pages, each manually classified into exactly one of 7 categories: *student, faculty, staff, department, course, project*, and *other*. However, we treated these as 7 binary classification problems. Four universities are represented by a large number of pages: Cornell (867 pages), Texas (827), Washington (1205), and Wisconsin (1263). Another 4,120 miscellaneous pages were collected from a range of other universities.

We followed the methodological recommendations from the above web page. The Rainbow system (*http://www.cs.cmu.edu/ mccallum/bow*) was used to tokenize the documents. Training was performed on documents from three of the main universities plus those from miscellaneous universities, with testing on the pages from the fourth, held-out university. This was repeated, using each of the four main universities as a test set once, and the results averaged. The number of terms in the training set therefore varied, ranging up to 85,600.

5.2.5 20 Newsgroups

The final test collection was *20 Newsgroups*, which consists of roughly equal-sized samples of postings to 20 Usenet newsgroups. Some postings belong to more than one newsgroup, and we treated the data set as specifying 20 binary classification problems. We used the "by

date" training/test split (*http://people.csail.mit.edu/people/jrennie/20Newsgroups/*), giving 11,314 training and 7,532 test documents.

After discarding all header fields except *Subject:*, text processing was the same as for ModApte. There were 102,760 distinct terms in the training set, and 138,700 in the union of the training and test sets.

5.3 Benchmark Algorithms

We compared lasso logistic regression to two alternatives.

5.3.1 Support Vector Machines

The support vector machine (SVM) algorithm for learning linear classifiers is consistently one of the most effective approaches to text categorization (Joachims 1998; Zhang and Oles 2001; Lewis et al. 2004). It produces models with a form of dual space 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, 2002) (*http://svmlight.joachims.org/*). All software parameters were left at default values except the regularization parameter C (option *-c*). The value of C was selected from the range 10^{-4} to 10^{-4} (by multiples of 10) using 10-fold cross-validation. The C that gave the highest cross-validated estimate for the sum of hinge losses ($\sum_{i=1}^{n} \max(0, -\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)$ was chosen. (We found this more effective than the usual approach of maximizing cross-validated classification accuracy.)

5.3.2 Ridge Logistic Regression with Feature Selection

Ridge logistic regression is widely used in text classification, but usually only after low quality features are discarded. We tested three feature selection methods in combination with ridge logistic regression, as well as trying no feature selection.

The feature selection approaches each computed some quality measure for each feature, ranked features by that measure, and then used 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. 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. Let a be the number of training documents in category and containing term j ("true positives"), b the number in the category but not containing j ("false negatives"), c the number not in the category but containing j ("false positives"), and d the number neither in the category nor containing j ("true negatives"). Let $n = a+b+c+d$ be the total number of training documents. Then the chi square measure is:

$$\chi^2 = \frac{n(ad - bc)^2}{(a+b)(c+d)(a+c)(b+d)}. \tag{22}$$

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

$$B(j) = |\Phi^{-1}(\frac{a}{a+b}) - \Phi^{-1}(\frac{c}{c+d})| \tag{23}$$

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 justifies BNS in term of ROC (receiver operating characteristic) analysis.

Most feature selection measures used in text classification (including the two discussed above) take into account only the presence or absence of terms in documents. Our third measure, the Pearson product-moment correlation, makes use of term weights in choosing features. The measure is:

$$r_j = \frac{\sum_{i=1}^{n}(x_{ij} - \overline{x^{(j)}})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_{ij} - \overline{x^{(j)}})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}, \tag{24}$$

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

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

$$E = \frac{\textit{false positives} + \textit{false negatives}}{|\mathcal{T}|} \tag{25}$$

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 the measure $F1$:

$$F1 = \frac{2 \times \textit{true positives}}{2 \times \textit{true positives} + \textit{false positives} + \textit{false negatives}} \tag{26}$$

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.

We sometimes report the unweighted arithmetic mean of $F1$ across the categories for a particular test collection, i.e. the *macroaveraged* $F1$.

5.5 Threshold Selection

Logistic regression models produce an estimate of $p(y = +1|\boldsymbol{\beta}, \boldsymbol{x_i})$, the probability that vector $\boldsymbol{x_i}$ belongs to the category of interest. For classification, we must convert these estimates to binary class label predictions. The simplest approach is to predict $y = +1$ (assign the category) when $p(y = +1|\boldsymbol{\beta}, \boldsymbol{x_i}) \geq 0.5$. This minimizes expected test set error

rate when the predicted probabilities equal the actual probabilities used in a probabilistic labeling. We refer to 0.5 as the *default* threshold.

Model inadequacies and finite training data mean that the probabilities produced by the logistic regression models may not be well calibrated. Hence we also tested setting the threshold for each category to the highest value that gave the minimum possible number of training set classification errors. We call these the *tuned* thresholds.

Maximizing $F1$ is trickier than minimizing error rate, since the optimal expected value of $F1$ cannot be achieved by any single pre-set threshold for all test sets (Lewis 1995). For simplicity, we used the same thresholds when evaluating by $F1$ that we did when evaluating by error rate.

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

6 RESULTS

We looked both at the raw effectiveness of lasso logistic regression for text categorization, and at its ability to provide more compact classifiers than competing approaches.

6.1 Effectiveness of Lasso Logistic Regression

Our first set of experiments compared the effectiveness of lasso logistic regression with ridge logistic regression and SVMs. Each algorithm used all training set features and a regularization parameter tuned by cross-validation on the training set. Table 1 compares macroaveraged F1 values for the three algorithms (with both default and tuned thresholds) on the five test collections. Lasso logistic regression outperforms ridge logistic regression under all conditions. With default thresholds it outperforms SVMs on four of five data sets, and with tuned thresholds on three of five data sets.

To test the significance of differences among the algorithms, we looked at the difference in per-category $F1$ values between pairs of algorithms and applied the 2-tailed Wilcoxon matched-pairs signed-ranks test (see Table 2). For all categories, the differences between $F1$

values were calculated and ranked from smallest to largest by absolute value. Then the sum of positive ranks (first method has greater $F1$) and sum of negative ranks (second method has greater $F1$) were compared. Algorithms significantly better at the $p = 0.05$ level are indicated in bold. By this measure, lasso logistic regression was significantly better than ridge logistic regression on ModApte, RCV1-v2, and OHSUMED, and tied on WebKB and 20 Newsgroups. Lasso logistic regression was significantly better than SVMs on WebKB, significantly worse on 20 Newsgroups, and tied on the other data sets. Note that these significance tests require a debatable assumption of independence among categories.

Figure 5 is a parallel coordinates plot showing the number of test set errors for each algorithm on the three data sets with largest number of categories. Each connected triple of points shows the number of test set errors made by ridge, lasso, and SVM respectively for a particular category. This plot is a reminder that for any given category, most test examples are clear negatives and are classified similarly by all three algorithms.

6.2 Comparing Lasso with Feature Selection

In text categorization, ridge logistic regression is often used in combination with feature selection, producing sparser and more effective classifiers. To test this approach, we chose feature sets of size 5, 50, 500 features for each category using each of our three feature selection methods. We fit ridge logistic regression models to these feature sets and, for completeness, also fit lasso logistic regression and SVM models as well. Default thresholds were used, and both norm-based and cross-validated hyperparameter settings. We summarize the results as follows:

- Feature selection often (though not always) improved the effectiveness of ridge logistic regression. BNS beat the other feature selection methods on macroaveraged $F1$, so Table 3 shows results only for BNS. In no case, however, did feature selection increase the macroaveraged $F1$ for ridge logistic regression on a data set to exceed that of lasso logistic regression with no feature selection.
- In no case did additional feature selection improve the macroaveraged $F1$ of lasso logistic regression.
- While not a focus of our experiments, we noted that feature selection sometimes improved the results of SVM models with default thresholds, but never the results of

SVM models with tuned thresholds. SVM's built-in regularization is quite effective.

- Our use of cross-validation to choose the lasso hyperparameter on a per-category basis, while using fixed feature set sizes for feature selection (the usual approach in text categorization research), perhaps gives an unfair unadvantage to lasso. Note, however, that lasso retains its dominance even with the (non-cross-validated) norm-based feature selection approach.

The number of features selected by lasso varied from category to category, but was always a small proportion of the full feature set. Figure 6 shows number of features chosen for each category on three collections–the other two are similar. Results in Figure 6 are with cross-validated hyperparameters. With the norm-based hyperparameter we found feature set sizes were 2 to 3 times larger: 2 to 892 (mean 93.8, standard deviation 167.9) for ModApte, from 0 to 3750 (mean 625.9, sd 726.9) for RCV1-v2, and from 0 to 2636 (mean 241.3, sd 393.3) for OHSUMED. So cross-validation gives a substantial benefit in sparsity. Due to a limitation in the version of the software used for these experiments, the intercept term was penalized in the same fashion as other parameters. Unsurprisingly, in in no case did lasso zero out the parameter for the intercept term.

There was a strong correlation between the number of positive training examples, and the number of features chosen. Figure 7 shows the relationship for the 90 categories and the 9,603 training examples of the ModApte collection. The other collections show similar patterns.

In our examples, SVM models included many more features than lasso models. For example, on the RCV1-v2 data set, lasso with cross-validated hyperparameters produced models with 3 to 1737 features (mean 294.1, sd 328.6), while SVM models had from 299 to 28481 features (mean 8414.5, sd 5859.6).

7 SUMMARY

Lasso logistic regression provides state-of-the-art text categorization effectiveness while producing sparse, and thus efficient-to-use, models. The approach is similarly useful in other high dimensional data analysis problems, such as predicting adverse drug events (Hauben, Madigan, Gerrits, and Meyboom, 2005). Recent extensions broaden the range of applicability even further. We have already mentioned the extension to polytomous logistic regression,

and other researchers have applied L1 regularization to more complex models as well (Li and Yang, 2004).

Richer prior distributions may also prove useful. Those used in our experiments, while informative in the statistical sense, are not based on any knowledge about language or the meaning of categories. In recent work (Dayanik, Lewis, Madigan, Menkov, and Genkin 2006) we have used textual descriptions of categories, user suggestions, and published reference materials to build priors that incorporate knowledge that some features are likely to be more useful than others, providing large effectiveness improvements.

Meinshausen (2005), Zhao and Yu (2006), and others have discussed the limitations of lasso as a feature selection algorithm. Future work will explore alternatives such as Meinshausen's "relaxed lasso." Several algorithms that estimate "regularization paths" have emerged in recent years (see, for example, Park and Hastie, 2006). These algorithms have the attractive property of estimating coefficients for all possible hyperparameter values. However, scaling such algorithms to huge applications remains challenging.

Acknowledgments

The US National Science Foundation supported this work through grants EIA-0087022 (KDD program), DMS-0113236 (ITR program), and DMS-0505599. The authors thank Andrei Anghelescu, Steve Benson, Aynur Dayanik, Susana Eyheramendy, Dmitriy Fradkin, Paul Kantor, Sathiya Keerthi, Paul Komarek, Vladimir Menkov, Fred Roberts, and Cun-Hui Zhang for suggestions, as well as thanking Bill Hersh, Ken Lang, Andrew McCallum, the US National Library of Medicine, Jason Rennie, and Reuters Ltd. for making available data sets used in this research.

References

Ackham, P. J. (2004), "An Algorithm for Computing the Inverse Normal Cumulative Distribution Function," *http://home.online.no/~pjacklam/notes/invnorm/*.

Berry, M. W. (ed.) (2004), *Survey of Text Mining: Clustering, Classification, and Retrieval*, New York: Springer-Verlag.

Dayanik, A., Lewis, D., Madigan, D., Menkov, V., Genkin, A. (2006), "Constructing Informative Prior Distributions from Domain Knowledge in Text Classification," in *Proceedings of SIGIR 2006: The Twenty-Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 493–500.

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

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004), "Least Angle Regression," *Annals of Statistics*, 32, 407–499.

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

Figueiredo, M. A. T. and Jain, A. K. (2001), "Bayesian Learning of Sparse Classifiers," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. I-35 to I-41.

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

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.

Hadjicostas, P. (2003), "Consistency of logistic regression coefficient estimates calculated from a training sample," *Statistics and Probability Letters*, 62, 392–303.

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

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

Hauben, M., Madigan, D., Gerrits, C., and Meyboom, R. (2005), "The role of data mining

in pharmacovigilence," *Expert Opinion in Drug Safety*, 4, 929–948.

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: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 192–201.

Hoerl, A. E. and Kennard, R. W. (1970), "Ridge regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, 12, 55–67.

Jin, R., Yan, R., Zhang, J., and Hauptmann, A. (2003), "A Faster Iterative Scaling Algorithm for Conditional Exponential Model," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pp. 282–289.

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

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

Jurafsky, D. and Martin, J. H. (2000), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Englewood Cliffs, NJ: Prentice-Hall.

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

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

Komarek, P. and Moore, A. (2003), "Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pp. 197–204.

Krishnapuram, B., Hartemink, A. J., Carin, L., and Figueiredo, M. A. T. (2005), "Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 957–968.

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

Li, F. and Yang, Y. (2004), "Recovering Genetic Regulatory Networks from Micro-Array Data and Location Analysis Data," *Genome Informatics*, 15(2), 131–140.

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

Lewis, D. D. (1998), "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," in *Machine Learning: ECML'98, 10th European Conference on Machine Learning*, pp. 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," in *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pp. 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*, 5, 361–397. Data set at *http://jmlr.csail.mit.edu/papers/volume5/lewis04a/ lyrl2004_rcv1v2_README.htm.*

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, 1103–1108.

Luenberger, D. G. (1984), *Linear and Nonlinear Programming* (2nd ed.), Reading, MA: Addison-Wesley.

Madigan, D. (2005), "Statistics and the War on Spam," *Statistics: A Guide to the Unknown*, eds. R. Peck, G. Casella, G. Cobb, R. Hoerl, D. Nolan, R. Starbuck and H. Stern, Duxbury Press, pp. 135-147.

Madigan, D., Genkin, A., Lewis, D. D., Argamon, S., Fradkin, D., Ye, L. (2005a), "Author

Identification on the Large Scale," in *Proceedings of the CSNA & INTERFACE Annual Meetings*.

Madigan, D., Genkin, A., Lewis, D. D., Fradkin, D. (2005b), "Bayesian Multinomial Logistic Regression for Author Identification," in *Bayesian Analysis and Maximum Entropy Methods in Science and Engineering: 25th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, AIP Conference Proceedings (vol. 803), pp. 509–516.

Madigan, D. and Ridgeway, G. (2004), Discussion of "Least Angle Regression" by B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Annals of Statistics*, 32, 465–469.

Mallick, B. K., Ghosh, D., and Ghosh, M. (2005), "Bayesian Classification of Tumors Using Gene Expression Data," *Journal of the Royal Statistical Society Series B*, 67, 219–234.

Malouf, R. (2002), "A Comparison of Algorithms for Maximum Entropy Parameter Estimation," in *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pp. 49–55.

Manning, C. and Sütze, H. (1999), *Foundations of Statistical Natural Language Processing*, Cambridge, MA: MIT Press.

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

Meinshausen, N. (2005), "Lasso with relaxation," Technical Report, U.C. Berkeley Department of Statistics.

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

Mosteller, F. and Wallace, D.L. (1964), *Inference and Disputed Authorship: The Federalist*, Addison-Wesley.

Park, M.-Y. and Hastie, T. (2006), "An L1 Regularization-path Algorithm for Generalized Linear Models," http://www-stat.stanford.edu/∼hastie/Papers/glmpath.pdf.

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. C 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* (2nd ed.), Cambridge, UK: Cambridge University Press.

Rocchio, Jr., J. J. (1971), "Relevance Feedback Information Retrieval," in *The SMART Retrieval System: Experiments in Automatic Document Processing*, ed. G. Salton, Englewood Cliffs, NJ: Prentice-Hall.

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

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

Schapire, R. E. and Singer, Y. (2000), "BoosTexter: A Boosting-Based System for Text Categorization," *Machine Learning*, 39, 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*, eds. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, Oxford, UK: Oxford University Press.

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. (1979), *Information Retrieval* (2nd. ed.), London: Butterworths.

Shevade, S. K. and Keerthi, S. S. (2003), "A Simple and Efficient Algorithm for Gene Selection Using Sparse Logistic Regression," *Bioinformatics*, 19, 2246–2253.

Weiss, S.M., Indurkhya, N., Zhang, T., and Damerau, F.J. (2005), *Text Mining: Predictive Methods for Analyzing Unstructured Information.* New York: Springer.

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

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

Zhang, T. and Oles, F. (2001), "Text Categorization Based on Regularized Linear Classifiers," *Information Retrieval*, 4, 5–31.

Zhao, P. and Yu, B (2006), "On Model Selection Consistency of Lasso," *Journal of Machine Learning Research*, to appear.

Zou, H. and Hastie, T. (2005), "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society (Series B)*, 67, 301–320.

Table 1: Text Categorization Effectiveness (Macroaveraged F1) on Five Collections

		Test Collection				
Algorithm	Threshold	ModApte	RCV1-v2	OHSUMED	WebKB	20 NG
lasso	default	48.64	54.75	47.23	46.96	73.24
ridge	default	37.63	47.72	36.09	45.73	71.47
SVM	default	37.69	44.87	25.28	44.14	75.15
lasso	tuned	52.03	57.66	51.30	47.28	75.13
ridge	tuned	39.71	52.42	42.99	46.03	73.67
SVM	tuned	52.09	57.26	49.35	39.50	81.19

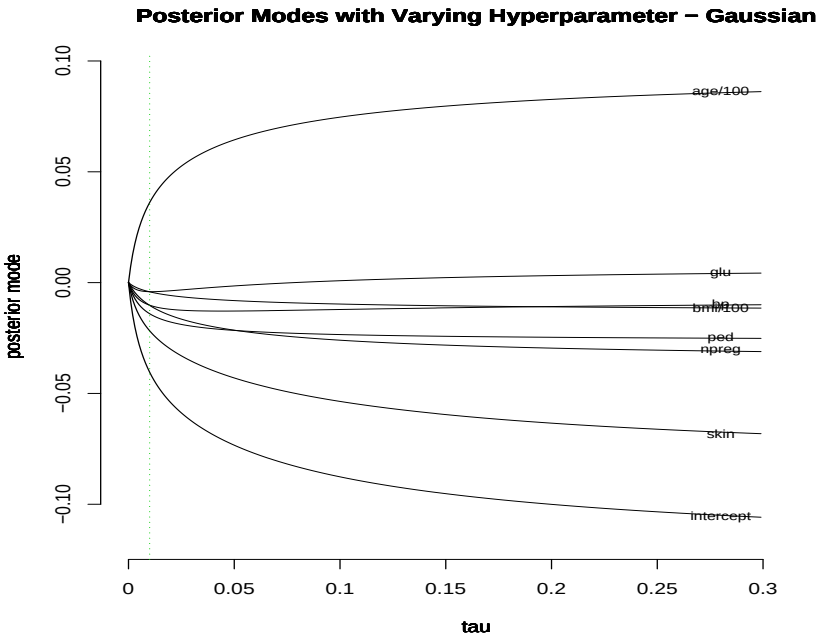

Figure 1: MAP Estimates of Logistic Regression Parameters for Gaussian Priors with Specified Variances. Vertical dotted line shows estimates with a mean 0, variance 0.01, Gaussian prior on each parameter.

Table 2: Two-Tailed Wilcoxon Paired Signed-Ranks Test on Per-Category F-measure Values

		Sum of ranks		
Collection	Algorithms	positive	negative	p-value
ModApte	**lasso** - ridge	1667.0	163.0	.000
(90 cats)	lasso - SVM	969.5	921.5	.863
	ridge - **SVM**	204.5	1811.5	.000
RCV1-v2	**lasso** - ridge	3925.0	926.0	.000
(103 cats)	lasso - SVM	2112.0	2739.0	.267
	ridge - **SVM**	1031.0	3625.0	.000
OHSUMED	**lasso** - ridge	2359.0	416.0	.000
(77 cats)	lasso - SVM	1462.0	1239.0	.540
	ridge - **SVM**	599.0	2102.0	.000
WebKB	lasso - ridge	197.0	128.0	.353
(7 cats)	**lasso** - SVM	299.0	79.0	.008
	ridge - SVM	228.0	123.0	.182
20 NG	lasso - ridge	156.0	54.0	.057
(20 cats)	lasso - **SVM**	1.0	209.0	.000
	ridge - **SVM**	0.0	210.0	.000

NOTE: Number of categories in each test collection is shown in first column.

Table 3: Text Categorization Effectiveness (Marcoaveraged F1) on Subsets of Features

			hyperparameter	
Collection	Alg.	# features	CV	norm
ModApte	lasso	all (18,978)	52.03	52.58
	ridge	all (18,978)	39.71	42.17
	ridge	500	50.14	41.40
	ridge	50	51.72	49.14
	ridge	5	47.04	49.48
RCV1-v2	lasso	all (47,152)	57.66	56.56
	ridge	all (47,152)	52.42	52.85
	ridge	500	51.90	50.82
	ridge	50	44.84	46.31
	ridge	5	31.21	32.54
OHSUMED	lasso	all (122,076)	51.30	48.92
	ridge	all (122,076)	42.99	42.74
	ridge	500	43.46	40.24
	ridge	50	44.27	45.64
	ridge	5	42.40	41.98
WebKB	lasso	all (varies)	47.16	44.94
	ridge	all (varies)	46.07	42.50
	ridge	500	45.11	42.30
	ridge	50	36.62	36.60
	ridge	5	24.58	24.69
20 NG	lasso	all (102,760)	75.13	76.31
	ridge	all (102,760)	73.67	76.44
	ridge	500	74.63	73.44
	ridge	50	66.65	66.94
	ridge	5	53.87	53.97

NOTE: Results are for lasso and ridge logistic regression with prior variances chosen using training set cross-validation (CV) or the norm-based heuristic (norm). Default thresholds were used. Feature subsets of the specified size were chosen for each category using the BNS criterion.

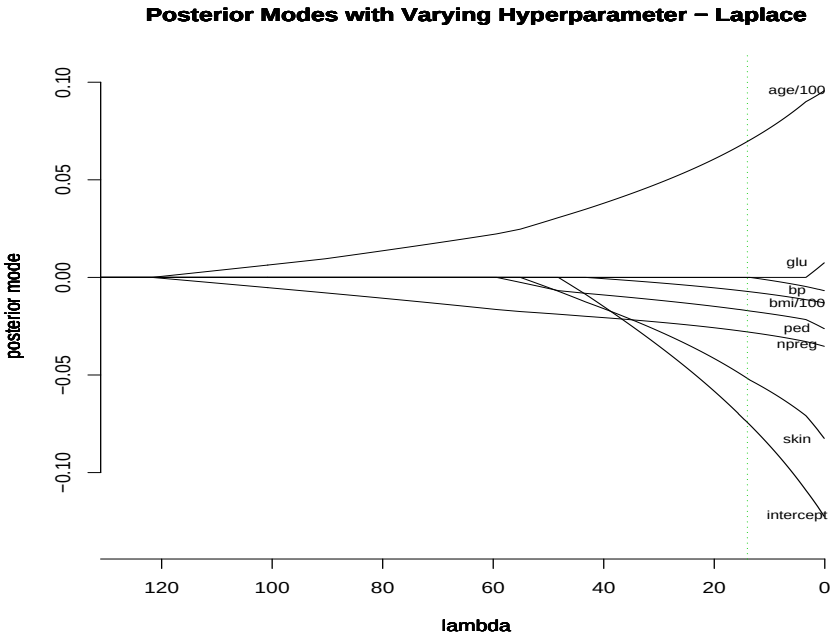

Figure 2: MAP Estimates of Logistic Regression Parameters for Laplace Priors with Specified Variances. Same data set as Figure 1. Vertical dotted line shows estimates with a mean 0, variance 0.27, Laplace prior on each parameter.

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 18)
(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 3: Zhang and Oles' CLG Algorithm with our Choice of Trust Region Update.

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 (20)
(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 (20)
(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 (20)
(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 4: Computation of Δv_j in the CLG-lasso Algorithm.

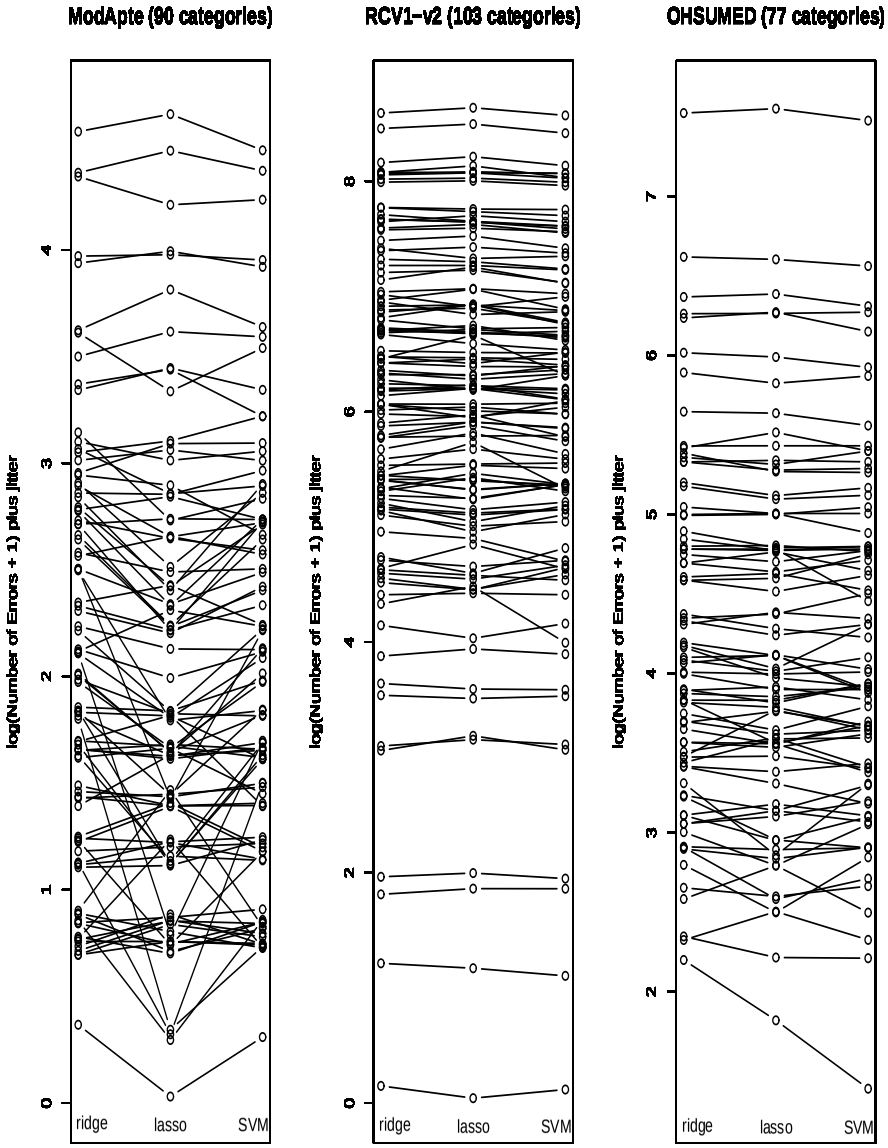

Figure 5: *Number of Test Set Errors For Each Category. Results are shown for ridge logistic regression, lasso logistic regression, and SVMs on the three test collections with the largest number of categories. Vertical axis shows the logarithm of 1 plus the number of test set errors, plus a small uniform random jitter to separate the categories. All algorithms use default thresholds, a regularization parameter chosen by training set cross-validation, and no feature selection.*

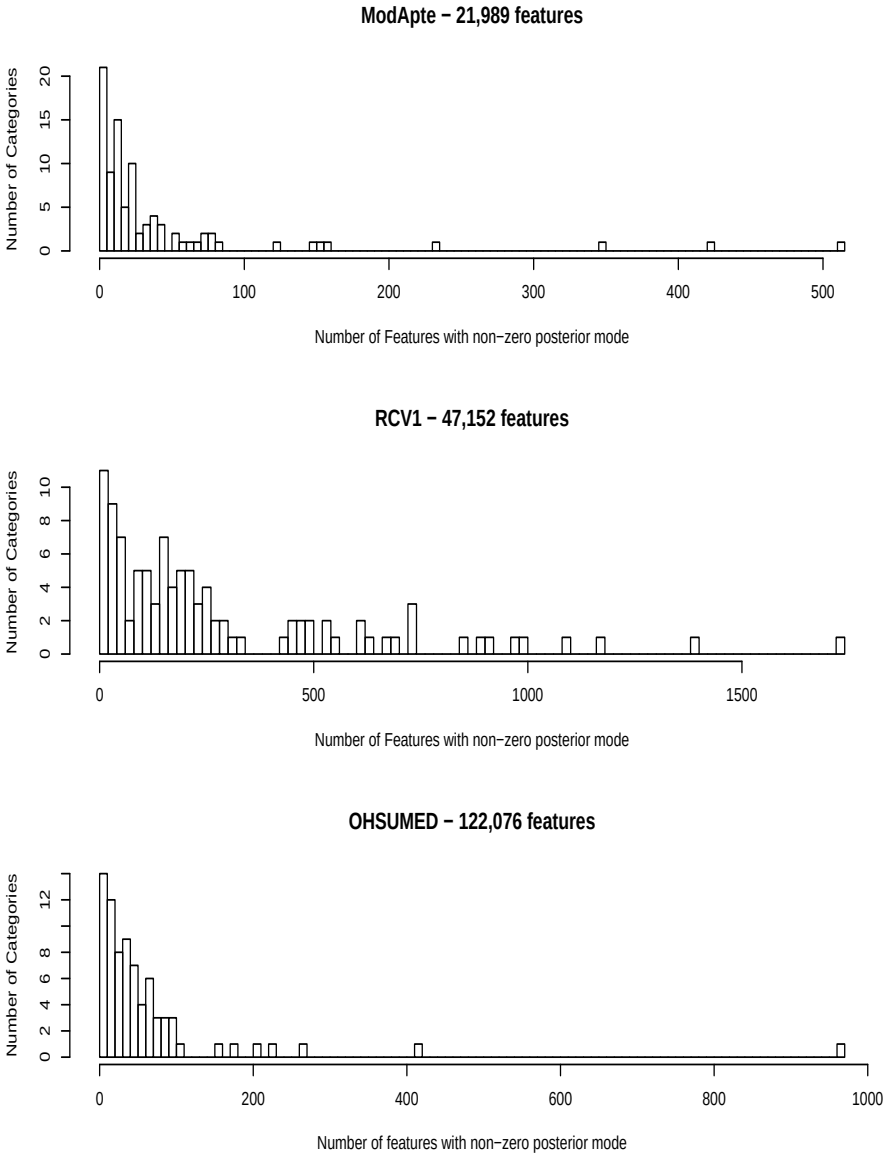

Figure 6: Number of Features Selected for Each Category in Three Test Collections by Lasso Logistic Regression. Hyperparameter values were chosen by training set cross-validation.

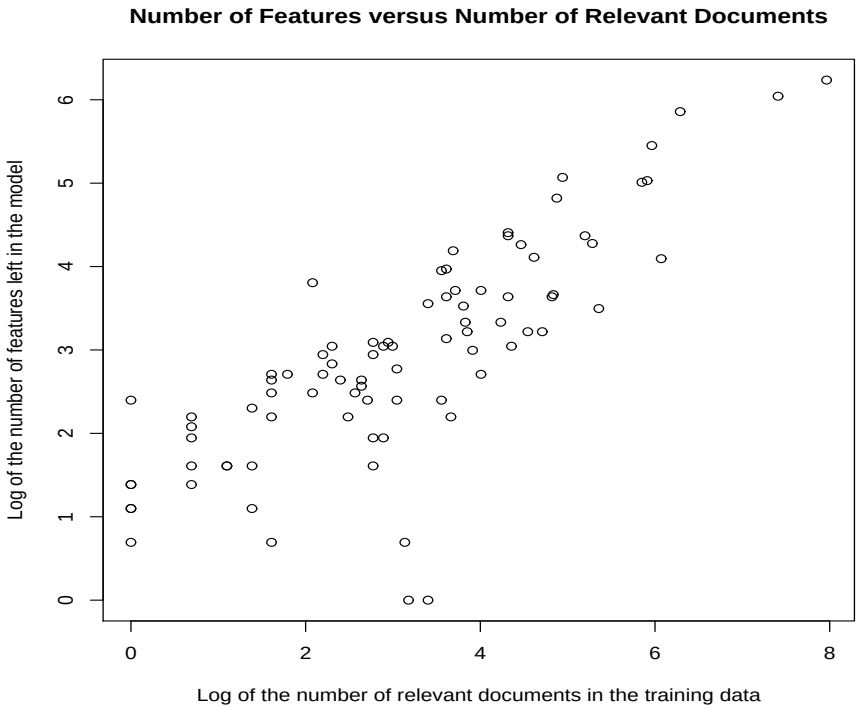

Figure 7: Number of Relevant (Positive) Examples for a Category vs. Number of Nonzero Parameters in Fitted Lasso Model. Fitting was on all 9,603 ModApte training documents. Hyperparameters were chosen by cross-validation. Axes use logarithmic (base e) scales.