Large-Scale Bayesian Logistic Regression

Eric Lai *ellai@uci.edu*
Alex Peterson *alexanlp@uci.edu*
Wendy Rummerfield *wrummerf@uci.edu*

STATS 230: Statistical Computing Methods Final Report
Instructor: Professor Babak Shahbaba
Department of Statistics
University of California, Irvine
October 18, 2017

Abstract

Classification problems of high-dimensional data such as natural language text pose computational and statistical challenges. Since maximum likelihood estimation often fails in high dimensional situations, Alexander Genkin, David D. Lewis, and David Madigan propose two Bayesian approaches to implement logistic regression models which avoid overfitting. The two algorithms are based on cyclic coordinate descent where a Gaussian prior (CLG) and Laplace prior (CLG-lasso) are utilized. We reproduced the algorithms in R using our own simulated data and we were able to draw similar conclusions as that seen in [2]. On large data sets where the number of parameters exceeds the number of observations, both algorithms outperformed maximum likelihood estimation in run-time and accuracy. CLG performed better than CLG-lasso in run-time, but overall the CLG-lasso algorithm produced smaller mean squared error values than CLG.

1 Introduction

The study of natural language processing (NLP) began to pick up momentum in the 1950's, shortly after World War II, when message decryption became critial to wartime strategy. From Colossus and the Enigma machine to Siri, knowledge of computational linguistics has made leaps and bounds in the last 60 years. Today, NLP involves speech recognition, machine learning, and machine translation all combining to enhance the field of artificial intelligence. In the real world, algorithms such as these can be employed for categorizing news stories, academic articles, patient reports, etc. and are most commonly used for spam filtering.

In this paper, we focus on binary classification using logistic regression. The most popular technique among statisticians to estimate the parameters of a model is maximum likelihood estimation (MLE). While the accuracy of MLE is hard to compete with, when dealing with high dimensional data like in the classification problem, the precision of - and even the

capability to obtain - estimates begins to break down. With the number of predictor variables in these types of models exceeding 10^4 and even surpassing the number of observations, matrix inversion becomes immensely time consuming and in some cases, impossible. This illustrates the need for novel methods to accurately and cost-effectively find model estimates.

Genkin et. al suggested a Bayesian approach to this problem. To deal with issues like overfitting and computational inefficiencies when using MLE, they proposed using priors so that posterior point estimates for many of the model parameters will be 0. The rest of this paper will be a replication and examination of the work presented in *Large-Scale Bayesian Logistic Regression for Text Categorization* by Genkin et. al.

In section 2 we discuss our methods for simulating the data and examining two Bayesian approaches for logistic regression. Both techniques are based on a cyclic coordinate descent method and employ either a Gaussian or Laplace prior on the coefficients to favor sparsity of the estimates. In section 3 we describe our approach to evaluating the effectiveness and efficiency of the two methods proposed in the paper using mean squared errors and run-times. Finally, section 4 we consider shortcomings in the algorithms, suggest possible areas of improvement, and suggest directions of potential future work.

2 Methods

2.1 Source of the Data

The classification data used by Genkin et. al consists of various feature vectors based on attributes of the data such as the number of words/phrases from multiple texts. More information about the data sets that were analyzed can be found in section 4. Instead of using these data sets to test the proposed algorithms, we simulated high dimensional design matrices with the number of observations and the number of covariates varying between 200 and 1000. Each row of our design matrix was randomly sampled from a multivariate normal distribution with mean $\mathbf{0}$, and variance-covariance matrix as the identity matrix, ensuring independence between covariates. We then set initial values for our covariate coefficients. For our purposes, we set 40% of our coefficients to 0 while the remaining 60% came from a Uniform[-1,1] distribution. Next, we simulated our response vector by drawing random samples from a Bernoulli distribution. Since we are working with logistic regression, we used $p = \frac{e^{X\beta}}{1+e^{X\beta}}$ as the probability of success for our Bernoulli distribution. We ran the simulated data sets through both of the proposed algorithms and compared the coefficient values to the true values. We also compared the run time of both algorithms.

2.2 Statistical Methods

These algorithms use a technique called supervised learning. In this approach, categories are predefined and documents are manually classified as (-1) if the element is not in the category or $(+1)$ if the element is in the category. The goal of our classifier, $y = f(\mathbf{x})$, where $\mathbf{x}_i = [x_{i,1}, \cdots, x_{i,d}]^T$ represents the numeric feature vectors described earlier, is to correctly label the elements in the vector. This vector comes from the training data $D = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$. We have simulated each of these components. In terms of our

logistic regression model, we are interested in the conditional probability model:

$$p(y = +1|\boldsymbol{\beta}, \boldsymbol{x}_i) = \frac{\exp(\sum_j \beta_j x_{i,j})}{1 + \exp(\sum_j \beta_j x_{i,j})}$$

where this probability represents correctly labeling the i^{th} observation. Next, we describe two Bayesian approaches to accurately obtain logistic regression estimates by specifying a prior on $\boldsymbol{\beta}$.

2.2.1 Gaussian Prior - Ridge Regression

Consider a Gaussian prior centered at 0 with variance, $\sigma_j^2 = \tau_j$:

$$p(\beta_j|\tau_j) = \frac{1}{\sqrt{2\pi\tau_j}} \exp\left(-\frac{\beta_j^2}{2\tau_j}\right) \sim N(0, \tau_j),\ j = 1, \cdots, d$$

where d is the length of the feature vector.

The purpose of a 0 mean is to induce sparseness into the model through our prior belief that many of the β_j's are close to 0. Also, depending on the value of τ_j, we can make our prior more or less information. Here, smaller values of τ_j would indicate more certainty that our estimates are near 0. For simplicity, we will assume $\tau_j = \tau$ for all j throughout the rest of the report. However, this method does not lead to many β_j's equaling 0, rather just being close to 0.

Since we want to inject our prior beliefs about $\boldsymbol{\beta}$- i.e. many β_j's are near/equal to zero- into the estimate, we turn to maximum a posteriori (MAP) estimation. Assuming that $\beta_i \perp \beta_j$ for all $i \neq j$, then we have that the prior for $\boldsymbol{\beta}$ is simply the product of the individual priors. For the logistic regression model, this method is actually equivalent to ridge regression.

Ridge regression is an approach to solve $A\mathbf{x} = b$ when there are either many solutions or no solutions since ordinary least squares will lead to overfitting. The resolution to this issue is to minimize the residual sums of squares including a $\mathcal{L}_2$-regularization term. This method will shrink the estimates of the regression coefficients to 0; however, as mentioned previously the L_2 (quadratic) penalty will result in small non-zero estimates [3].

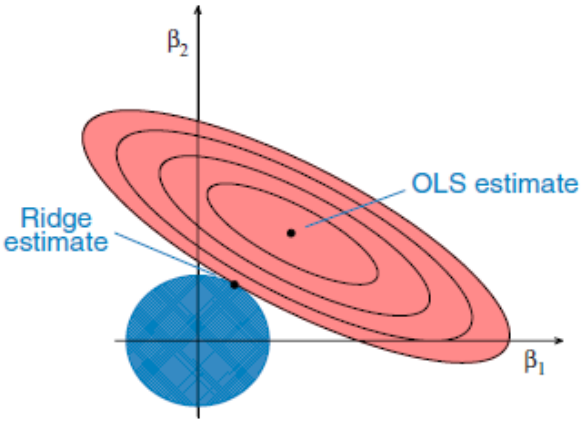

Figure 1: A geometric interpretation of Ridge Regression given in [1].

Geometrically, we can think of an ellipse with contours corresponding to the residual sum of squares (RSS). Centered at 0, is the constraint in ridge regression, corresponding to a circle $\sum_{j=1}^{[} \beta_j^2 < c$, where p is the number of predictors and $c > 0$ is some constraint. The ridge regression estimate is the single point at which the ellipse and the circle

meet. We aim to minimize the ellipse size and the circle simultaneously as shown in Figure 1. This creates a trade-off between the penalty and the RSS [1].

2.2.2 Laplace Prior - Lasso Regression

In the interest of creating a prior that reflects the desire for sparsity, we again consider β_j's from a Gaussian distribution. This time, we also inflict a prior on the variability:

$$p(\beta_j|\tau_j) \sim N(0, \tau_j) \qquad \text{where } \; p(\tau_j|\gamma) = \frac{\gamma_j}{2} \exp\{-(\gamma_j \tau_j)/2)\}, \; \gamma > 0.$$

Integrating out τ_j, we obtain a double-exponential, or Laplace, distribution

$$p(\beta_j|\lambda_j) = \frac{\lambda_j}{2} \exp\{-\lambda_j|\beta_j|\}, \quad \lambda_j = \sqrt{\gamma_j} > 0.$$

where we assume $\lambda_j = \lambda$ for all j and independent β_j's.

Least absolute shrinkage and selection operator (Lasso) is equivalent to the Bayesian MAP estimate using a Laplace prior. Lasso is similar to ridge regression, but with a different penalty term, the $\mathcal{L}_1$ norm, on the minimizing equation. This penalty, proposed by Robert Tibshirani, forces some of the coefficients to zero and leads to a more interpretable model, a drawback in ridge regression [4]. However, using this method will have a much less pronounced effect on the non-zero coefficients, with comparatively little shrinkage [3].

2.3 Algorithms for Bayesian Logistic Regression

Both algorithms are based off of the combined local and global (CLG) algorithm. The CLG algorithm is a type of cyclic coordinate descent algorithm with a Guassian prior used for optimization. The authors in [2] have adapted it to perform well for high-dimensional logistic regression problems. The following gives an outline of the algorithms implemented (psuedocode for both algorithms can be found in section 4):

1. Initialize variables to some value.
2. Minimize the objective function (described below) with respect to the first variable, holding all others constant.
3. Minimize the second variable holding all other variables constant and keeping the new value of the first variable.
4. Continue this process until all variables have been traversed.
5. Return to the first variable and repeat steps (1) - (4) until convergence criterion is met.

To find the MAP estimates, we simply obtain the posterior density with the logistic link based on our respective priors, $p(\boldsymbol{\beta})$, Gaussian or Laplace over our data:

$$L(\boldsymbol{\beta}) = p(\boldsymbol{\beta}|D) \propto \left(\frac{1}{1 + \exp(-\boldsymbol{\beta}^T \boldsymbol{x}_i y_i)} \right) p(\boldsymbol{\beta}).$$

From this point, we find the log likelihood and take the partial derivative with respect to each element in $\boldsymbol{\beta}$. This becomes a one-dimensional problem where we only find the $\beta_j^{(new)}$ that minimizes $\log(L(\boldsymbol{\beta})) = -\ell(\boldsymbol{\beta})$ one at a time. There are two different approaches at this point, one for the ridge regression version of the CLG and one for the lasso logistic regression version of the CLG.

For *ridge regression*, the minimum value of $-\ell(\boldsymbol{\beta})$ does not have a closed form, so we use a modified second order Taylor series approximation. More details on this process can be found in section 4). Furthermore, to avoid large updates of $\beta_j^{(new)}$, i.e. $\Delta\beta_j = \beta_j^{(new)} - \beta_j$ we specify a value that $|\Delta\beta_j|$ cannot exceed. For each update, $\beta_j^{(new)}$ depends only on the current values of the other β_j's a slight modification from the original CLG which iterates $\beta_j^{(new)} = \beta_j + \Delta\beta_j$ until convergence.

On the other hand, for *lasso regression*, we cannot use the second order Taylor series approximation because the first and second moments are not finite. Genkin et. al. offered new methods of updating to the β_j's to fix this problem: first, if the update would change the sign of $\beta_j^{(new)}$, they set $\beta_j^{(new)}$ to 0; second, if the starting value of $\beta_j^{(new)}$ is 0, they tried updates in both directions since our objective function is convex, only one direction will be successful. Essentially, the CLG-lasso algorithm is the same as the one described above, but with a different updating step.

Finally, all that remains is to choose the hyperparmaeters for the priors: $\tau_j = \sigma_j^2$ (for the Gaussian prior) and $\lambda_j = \sqrt{2}/\sigma_j$ (for the Laplace prior). Genkin et al. discussed two ways to determine the prior value of σ_j^2:

(1) $\tau_j = \sigma_j^2 = \frac{d}{u} = dn/\sum_{i=1}^{n} ||\boldsymbol{x}_i||_2^2$ where d is the number of coefficients including the intercept and u is the mean squared Euclidean norm of the training examples.

(2) Perform partial 10-fold cross-validation on the training data.

These methods resulted in $\sigma_j^2 \in (.0001 - 10,000)$ by multiples of 10 and $\lambda_j \in (.01 - 316)$ by multiples of $\sqrt{10}$. With these values they computed the sum of the log likelihoods from the documents and chose the hyperparameter values that maximized this sum, calling these values the *cross-validated* hyperparameters. Due to computational limitations, we will instead be utilizing particular values of τ_j and β_j.

3 Results

To quantify the effectiveness of each algorithm on our simulated data, we compared mean-squared error (MSE) values and run-times of both algorithms while varying the dimensions of our data. We first simulated a low dimensional data set with only 5 covariates. We then ran this data set through both algorithms and compared our coefficient estimates to the maximum likelihood estimates. The results are shown in Table 1.

Since this data set consists of only 5 covariates, it is not surprising that the maximum likelihood estimates (MLEs) are more accurate than both algorithms in 3 out of the 5 estimates, though the difference is negligible. When we look at higher dimensional data, specifically

Table 1: Parameter estimates of different estimation methods using a low dimensional data set.

	β_1	β_2	β_3	β_4	β_5
True β Values	1.0	0.3	0.4	0	0
MLEs	0.9993	0.2857	0.4134	-0.0010	-0.0052
CLG β Estimates	0.9640751	0.2813006	0.3988829	0.0060046	-0.0038828
CLG-lasso β Estimates	0.977402	0.2852631	0.4044002	0.0060124	-0.0039610

when the number of covariates is larger than the number of observations, maximum likelihood estimation cannot be performed. We found that both of our proposed algorithms were accurate and efficient in estimating coefficients.

Figure 2

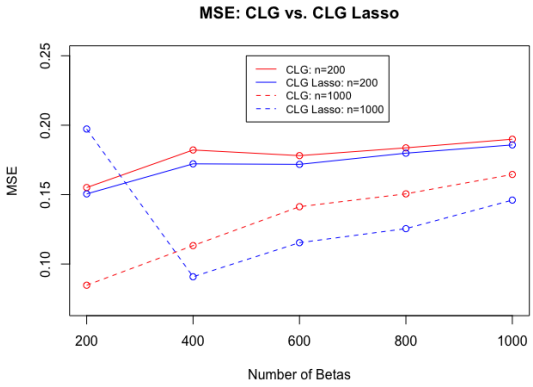

Figure 3

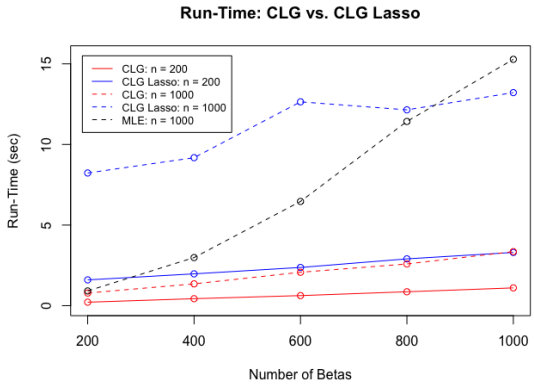

Figure 2 compares the MSEs of the CLG algorithm and the CLG-lasso algorithm with our true coefficient values. We first hold the number of observations constant at 200 and then 1000 while varying our number of covariates. CLG-lasso performs better when the number of observations is 200 as well 1000 across all numbers of covariates except when we have 1000 observations and 200 covariates (right endpoint of our plots). We can also see that as the number of observations increases from 200 to 1000, the difference in MSE between CLG and CLG-lasso increases as well. Therefore, as we continue to increase the number of observations, the CLG-lasso algorithm will outperform the CLG algorithm by a larger margin. Maximum likelihood estimation cannot be performed when the number of covariates is larger than the number of observations, so it is not included in Figure 2.

Table 2 summarizes the results including MLE in the cases where our sample size is greater than or equal to our number of coefficients. Notice that when the number of covariates is 1000 and the sample size is 1000, the MSE value under MLE increases by over 200. This is due to the fact that we are estimating 1001 coefficients (including the intercept term) when we have 1000 covariates. This demonstrates that MLE becomes unreliable immediately once the number of coefficients exceeds the number of observations once. Overall, both algorithms largely outperform MLE. CLG-lasso generally minimizes MSE for both sample sizes and

across all number of covariates.

Table 2: MSE values of different methods for parameter estimation using a high dimensional data sets.

	Num. Covariates	CLG	CLG-lasso	MLE
$n = 200$				
	200	0.155	0.151	260.7212
	400	0.182	0.172	NA
	600	0.178	0.172	NA
	800	0.184	0.180	NA
	1000	0.190	0.186	NA
$n = 1000$				
	200	0.085	0.197	269.313
	400	0.113	0.091	14.788
	600	0.141	0.115	4.470
	800	0.151	0.125	3.569
	1000	0.165	0.146	213.925

Figure 3 compares the run-time of both algorithms. Similar to Figure 2, we are holding the number of observations constant at 200 while varying the number of coefficients. We are then holding the number of observations constant at 1000 while varying the number of coefficients. This plot tells us that CLG is more efficient than CLG-lasso when we have 200 observations as well as when we have 1000 observations. This holds true for each number of coefficients. We can also see that as we increase the number of observations, CLG outperforms CLG-lasso by a larger margin. Here, we run into the issue of accuracy versus efficiency: both algorithms perform better than maximum likelihood estimation for high dimensional data. So, CLG-lasso should be used when the main goal is to optimize the accuracy of the coefficient estimates and CLG should be used when the main goal is to minimize the run-time of the algorithm.

4 Discussion

Both Ridge logistic regression and Lasso logistic regression produce accurate and fast models compared to MLE for high dimensional classification problems. To improve these algorithms, we can use more informative/richer priors than the uninformative Gaussian and Laplace priors that are based off of information about the given text. This should yield more accurate coefficient estimates resulting in even smaller MSE values and potentially shorter algorithm run-time. Since $\mathcal{L}_2$-regularization does not promote sparsity, Lasso regression is normally preferred over Ridge regression when the true parameter values are sparse. This was demonstrated as well under our simulated data analysis as our true parameter values were very sparse and CLG-lasso outperformed CLG in terms of accuracy. In addition, Lasso regression does have its limitations as a feature selection method. For instance, Peng Zhao and Bin Yu show that an irrepresentable condition is almost necessary and sufficient for Lasso to select the true model [5].

For future work, we could run both algorithms on the data sets used by Genkin, Lewis, and Madigan to verify their results. They used 5 different data sets and found that CLG-lasso performed better than CLG on all 5 data sets. Similar to their findings, the CLG-lasso algorithm performed better than the CLG algorithm on our simulated data set. However, Genkin, Lewis, and Madigan evaluated the performance of each algorithm based on the proportion of true positives produced by each data set. We evaluated the performance of each algorithm based on the MSE when compared to the true coefficient values. For future work, we could evaluate the performance of both algorithms on our simulated data set based on the proportion of true positives. We could also compare our algorithms to the Support Vector Machine (SVM) algorithm. This algorithm is known as one of the most efficient approaches to text categorization, so it would be interesting to see if our algorithms outperformed the SVM algorithm. Finding, extending these two methods to be used in polytomous (multiple responses) logistic regression would be of interest as well.

References

[1] *Ridge regression*, 2017. from https://onlinecourses.science.psu.edu/stat857/node/155.

[2] A. Genkin, D. D. Lewis, and D. Madigan, *Large-scale bayesian logistic regression for text categorization*, Technometrics, 49 (2007), pp. 291–304.

[3] J. Goeman, R. Meijer, and N. Chaturvedi, *L1 and l2 penalized regression models*, 2016.

[4] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodological), (1996), pp. 267–288.

[5] P. Zhao and B. Yu, *On model selection consistency of lasso*, Journal of Machine learning research, 7 (2006), pp. 2541–2563.

Appendix

Source of Data in [2]

The classification data the authors worked with are from documents are preprocessed into a form that is readable to a computer [1] Certain characteristics- words, document type, subject, etc.- are coerced into numeric values and placed into a feature vectors. There are two components to this process: text processing and term weighting. Text processing breaks a string of characters into blocks of text. These blocks can represent either a single word or a short phrase. Term weighting then counts the number of occurences of each distinct block in a given document and across documents. A corresponding numeric weight is then calculated for each block of text. A document is then represented as a vector of these weights. The weight for block j in document i can be computing as the following:

$$x_{ij}'' = = \begin{cases} 0 & \text{if } n(i,j) = 0 \\ [1 + \ln(n(i,j))] \ \ln\left(\frac{|D|+1}{n(j)+1}\right) & \text{otherwise} \end{cases}.$$

Here, $n(j)$ is the number of training documents that contain block j, $n(i,j)$ is the number of occurences of block j in document i, and D is the total number of training documents. A vector used to represent a single document can range in dimensionality from 10^3 to 10^6. Here, we are only interested in binary classification. Genkin et al. ran CLG and CLG-lasso on 5 different data sets all of which are standard test categorization test collections. 3 of the data sets (ModApte, RCV1-v2, and OHSUMED) contain about 100 distinct binary classification problems corresponding to predicting expert human indexing decisions. The remaining data sets (WebKB Universities, and 20 Newsgroup) contain fewer binary classifications commonly used in published research. Instead of using these data sets again, we will perform both algorithms on our own simulated data sets described below.

Derivations for Lasso and Ridge Logistic Regression

Log likelihood with Gaussian prior:

$$\ell(\boldsymbol{\beta}) = -\sum_{i=1}^{\infty} \ln(1 + \exp(-\boldsymbol{\beta}^T \boldsymbol{x}_i y_i)) - \sum_{j=1}^{d} \left(\ln \sqrt{\tau_j} + \frac{\ln 2\pi}{2} + \frac{\beta_j^2}{2\tau_j} \right)$$

Log likelihood with Laplace prior:

$$\ell(\boldsymbol{\beta}) = -\sum_{i=1}^{\infty} \ln(1 + \exp(-\boldsymbol{\beta}^T \boldsymbol{x}_i y_i)) - \sum_{j=1}^{d} (\ln 2 + \ln \lambda_j + \lambda_j |\beta_j|$$

Ridge Logistic Regression Model:

Find $\beta_j^{(new)}$: equivalent to finding the value z that minimizes

[1] It should be noted that, depending on the goal of analysis, text may be analyzed in its original form instead of first being converted into numbers, i.e. for linguistic analysis.

$$g(z) = \left(\sum_{i=1}^{n} \ln(+\exp\{r_i + (z - \beta_j)x_{ij}y_i\} \right) + \frac{z^2}{2\tau_j}$$

where $r_i = (\boldsymbol{\beta}^T \boldsymbol{x}_i y_i$

Update: $\Delta_j^{new} = \max(2|\Delta\beta_j|, \Delta_j/2)$.

Convergence attained when $(\sum_{i=1}^{n} |\Delta r_i|)/(1 + \sum_{i=1}^{n} |r_i|) \leq \epsilon$

Relevant formulas for Algorithm 1 below:

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

where

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

Psuedocode for Lasso and Ridge Logistic Regression

Psuedocode for Algorithm 1 (Ridge):

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 $\Delta\nu_j$
(5) $\Delta\beta_j \leftarrow min(max(\Delta\nu_j, -\Delta_j), \Delta_j)$ (limit step to trust region)
(6) $\Delta r_i \leftarrow \Delta\beta_j x_{ij} y_i, r_i \leftarrow r_i + \Delta r_i$ 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**

Psuedocode for Algorithm 2 (Lasso):
Algorithm 2: Computation of $\Delta\nu_j$ in CLG-lasso
(1) **if** $\beta_j = 0$
(2) $s \leftarrow 1$ (try positive direction)
(3) compute $\Delta\nu_j$
(4) **if** $\Delta\nu_j \leq 0$ (positive direction failed)
(5) $s \leftarrow -1$ (try negative direction)
(6) compute $\Delta\nu_j$
(7) **if** $\Delta\nu_j \geq 0$ (negative direction failed)
(8) $\Delta\nu_j \leftarrow 0$
(9) **endif**
(10) **endif**
(11) **else**

(12) $s \leftarrow \beta_j / |\beta_j|$
(13) compute $\Delta\nu_j$
(14) **if** $s(\beta_j + \Delta\nu_j) < 0$ (cross over 0)
(15) $\Delta\nu_j \leftarrow -\beta_j$
(16) **endif**
(17) **endif**

Algorithm Comparison Method used in [2]

To quantify the effectiveness of each algorithm Genkin, Lewis, and Madigan compared $F1$ values of the two algorithms to that of the Support Vector Machine algorithm. The SVM algorithm is known as one of the most effective approaches to text categorization. Therefore, the SVM algorithm will serve as a suitable benchmark for comparison with the two algorithms. They defined $F1$ as

$$F1 = \frac{2 \times \text{true positives}}{2 \times \text{true positives} + \text{false positives} + \text{false negatives}}$$

. Both algorithms as well as the SVM algorithm were performed on 5 different data sets. They found that CLG Lasso outperformed CLG on all 5 data sets. They also found that CLG Lasso outperformed SVM on four of five data sets.