

Model-Averaged LASSO

Daniel Percival* and Chris Fraley†

Technical Report, Insightful Corporation

August 22, 2008

Abstract

This paper studies combining ℓ_1 regularization and Bayesian Model Averaging (BMA) techniques.

Introduction Dimension reduction often aids regression problems involving large numbers of predictors by improving both the prediction accuracy and the interpretability of the final model. Regularization is a popular “automatic” dimension reduction technique. Such techniques produce a path of solutions, from which a single solution - corresponding to point on the path - is chosen. For example, in the LASSO method due to Tibshirani (1996), one solves:

$$\hat{\beta} = \operatorname{argmin} \left(\sum_{i=1}^N (y_i - \sum_j \beta_j x_{ij})^2 + \lambda \sum_j |\beta_j| \right)$$

$(\lambda > 0)$

As λ varies, the solution also changes, starting from the OLS solution when $\lambda = 0$ to an intercept-only solution for some high positive value of λ . This produces the path of solutions, which can be indexed by λ or $\|\beta\|_1$.

There are many competing ways to choose the final solution from such a path of solutions, each with its own advantages and drawbacks. However, most current techniques are alike in that they choose a single point on the

*Department of Statistics, Carnegie Mellon University

†Insightful Corporation, 1700 Westlake Avenue North, Suite 500, Seattle, WA 98109

path. This property raises the question of model uncertainty, or the confidence in the selection of that single point. The issue of model uncertainty is generally recognized and understood in other applications. Bayesian model averaging (BMA) is a technique used to resolve model uncertainty by taking a weighted average over many possible models in a model space (Hoeting et al. 1999). By treating the entire regularization path as a model space, a combination technique of regularization and model averaging is developed in the following sections to attempt to resolve the model uncertainty issues arising from path point selection.

Model Averaging - Markov Chain Monte Carlo Model Composition Suppose we are given data D , and a model space $\{M\}$, and wish to compute a quantity Δ . For example, in a regression problem, Δ can be a single coefficient or the probability that a particular coefficient is zero. The posterior distribution of Δ is given by:

$$\mathbf{P}(\Delta|D) = \sum_k \mathbf{P}(\Delta|M_k, D)\mathbf{P}(M_k|D)$$

We can estimate this quantity via MCMC by constructing a Markov chain with equilibrium distribution $\mathbf{P}(M|D)$ – the posterior distribution of the entire model space. This process is called Markov Chain Monte Carlo Model Composition (MCMCMC). Indexing the chain as $\{M(t)\}$, $t = 1, 2, \dots, N$, we can then estimate Δ by taking the average:

$$\hat{\Delta} = \frac{1}{N} \sum_{t=1}^N g(M(t))$$

Where $g(M(t))$ calculates the quantity Δ for the model $M(t)$. This quantity will converge almost surely by the law of large numbers to the true value of Δ .

Such a chain with the desired equilibrium property can be constructed via a Metropolis-Hastings algorithm. Start with a model N in the model space. A neighborhood around N in the overall model space is then defined. For example, in the case of linear regression, the set of models using all possible subsets of a set of candidate covariates can be considered the model space. Given a particular model, all models with one additional or fewer covariate included than it could be considered to be the neighborhood of that model in this model space.

Next, a new model N' within the neighborhood of N is next proposed by some random selection. This new proposal is then “accepted” with probability:

$$\mathbf{P}(\text{Accept } N') = \min \left(1, \frac{\mathbf{P}(N'|D)}{\mathbf{P}(N|D)} \right)$$

If accepted, then N' is now considered as N for the next iteration, and a new N' is proposed based on a neighborhood about the old N' . Otherwise, the original model N is kept and another N' is proposed as before, based on a neighborhood of N . The equilibrium distribution of this chain is $\mathbf{P}(M|D)$.

There are two difficulties in this setup. First, the concept of neighborhood in a model space can be hard to define. Later, we will see that this idea can be made quite natural in the context of a regularization path. The second difficulty is calculating $\frac{\mathbf{P}(N'|D)}{\mathbf{P}(N|D)}$. This forces the computation of integrated likelihood values for each model, a notoriously difficult problem.

Combination Algorithm MCMCMC has particular appeal in the case of BMA when applied to a model space generated by LASSO since the metric on this model space is quite natural. A LASSO path can be represented by the interval $[0, \lambda_{max}]$, which is simply the range of possible λ penalization values i.e., a point $\alpha \in (0, \lambda_{max})$ means that:

$$\hat{\beta}_\alpha = \operatorname{argmin} \left(\sum_{i=1}^N (y_i - \sum_j \beta_j x_{ij})^2 + \alpha \sum_j |\beta_j| \right)$$

We can thus write M_α to represent a model from this space at the point $\lambda = \alpha$ along the LASSO path. Under this representation, λ_{max} represents the intercept only model and 0 represents the full regression. A similar setup can be used for the “elastic net” regularization technique (Zou and Hastie 2005). Since neighborhoods on the real line are intuitive concepts, this makes the model space framework quite natural.

The MCMCMC method described was implemented for elastic net paths for both normal linear regression and for binomial family generalized linear models. The elastic net models were fit using the R package `glmnet` (Friedman et al. 2008). Note that the LASSO is a special case of the elastic net. The algorithm proceeds as follows:

1. Begin by choosing a random starting point $\alpha \in [0, \lambda_{max}]$ using a single $U(0, \lambda_{max})$ draw.
2. Propose a new α' (discussed below)
3. Calculate $\mathbf{P}(\text{Accept } M_{\alpha'}) = \min\left(1, \frac{\mathbf{P}(M_{\alpha'}|D)}{\mathbf{P}(M_{\alpha}|D)}\right)$
4. Draw a single $U(0, 1)$ to determine acceptance. Set $\alpha = \alpha'$ if accepted, keep $\alpha = \alpha$ otherwise.
5. Repeat steps 2 - 4 a large number of times (10000 times is a sufficient number)
6. Discard the first portion (2500 is a sufficient amount) of this sequence of α values, and use the remaining tail as an approximation of the model space distribution (this is the distribution of the single parameter α).

There are several issues that arise with this implementation idea. In the second step, the proposal distribution must be adjusted to ensure good properties. In this area, there are three main concerns - (1) the acceptance rate, how often a proposed model (α) is accepted, (2) the autocorrelation of the samples (high autocorrelation is not desired) and (3) the potential for the chain to become stuck at a certain point in the middle, or somehow be forced to the ends.

Using either one of a $U(\alpha - a, \alpha + a)$ or a $N(\alpha, a)$ proposal distribution can produce good results with respect to concerns (1) and (2). However, concern (3) is not fully addressed, since this scheme has the potential to become trapped at the ends of the interval representing the regularization path. When the chain is near the end, a significant part of the density of either proposal distribution lies outside of the space. If one of these points is chosen, it is simply mapped to the border. This creates an attracting effect.

There are two main ways in which this attracting effect could be resolved. First, a new proposal distribution could be devised. The distributions could be truncated to exclude any support outside of the model space. This solution produces the opposite effect: the endpoints now are repelling. Another possibility is to alter this truncation to keep the distributions symmetric around the current model. This is not a viable solution either, it only serves to enhance the attracting effect of the endpoints. A second solution is to re-parameterize the model space to be constraint free. A transformation such

as a logit transform is appropriate. However, this too creates problems with the “endpoints”, which are distorted to occupy large areas of the real line by the transform. This creates a similar attracting effect.

For now, the best solution to the problem is to choose the tuning parameter a wisely. This choice must take into account both the attracting concern and the number of modes in the posterior distribution of the model space. If the posterior is suspected to be unimodal, then a small tuning parameter is acceptable, and can be chosen to be small enough to avoid ever reaching one of the endpoints. If the posterior has many modes, then a large tuning parameter can allow the chain to jump between modes easily. Note that the notion of small and large is relative to the value of λ_{max} .

Another major issue in this algorithm is computing the quantity $\frac{\mathbf{P}(M_{\alpha'}|D)}{\mathbf{P}(M_{\alpha}|D)}$ for use in step 3. Noting that $\mathbf{P}(M|D) \propto \mathbf{P}(M, D)\mathbf{P}(M)$ reveals that there are two problems in computing this quantity. First, a prior must be placed on the model space. At this time, we use a flat prior, so that all models are considered a priori equally likely. This effectively eliminates this terms from all calculations. Second, the integrated likelihood ($\mathbf{P}(M, D)$) is a notoriously difficult quantity to compute. At this time, we use the common BIC approximation.

The main way to improve upon the BIC approximation is to apply some sort of Monte Carlo integration technique. As noted in Tibshirani (1996), a LASSO model corresponds to a Bayesian regression model with independent double exponential priors on each coordinate of β , using the posterior mode as the coefficient estimates. This gives us a basic setup for a Monte Carlo integration. However, in many cases this is a very high dimensional integral. This makes such techniques very difficult and arduous to apply. We therefore proceed with the BIC approximation.

Processing MCMCMC Results Post-processing of an MCMCMC analysis is fairly straightforward. In the algorithm described above, at each step the coefficients of the accepted point (either α' if it was accepted or α if α' was rejected) can be stored. These stored values can be used for all calculations by simple averaging or taking quantiles. Some items of interest that are easily obtained:

- *The posterior model space distribution.* An estimate of the posterior distribution of the model space can be obtained using the α values stored in the tail. This can be done via a density estimator. In the

examples in the next section, we will use a histogram of the tail α values.

- *The probability of a coefficient being equal to zero.* This can be found by simply computing the fraction of models in the tail with that coefficient equal to zero.
- *The value of a coefficient.* This can be found by averaging over all of the stored values of that coefficient in the tail.
- *Fitted values.* Fitted values can be produced by using the average coefficient values obtained above. Binomial regression problems can be further investigated by applying a decision boundary to produce classes.
- *Confidence intervals for a coefficient.* These can be found by taking quantiles of the coefficient values stored in the tail.

Examples

Example 1: Diabetes Data The Diabetes dataset used in Efron et al. (2004) is used to demonstrate the algorithm applied a linear regression problem. A LASSO model was fit to the data using `glmnet`. The usual coefficient plot and cross validation results are displayed in Figure 1. According to the cross validation results, the entire last half of the regularization path could contain a good choice for a single point. This displays a degree of model uncertainty that would encourage application of our combination technique. A MCMCMC was ran for 10000 iterations, with the first 2500 discarded to allow time for convergence. For analysis, only every 6th draw from this tail was used to reduce autocorrelation. 59% of all moves in the model space were accepted, and of these 86% were moves made with probability one. The MSE for the “averaged” model was 2884.7. This was obtained by taking the coordinate-wise mean of the β vector over the tail of the Markov chain. The MSE for an OLS fit of the data is 2933. This process thus gives a marginal improvement.

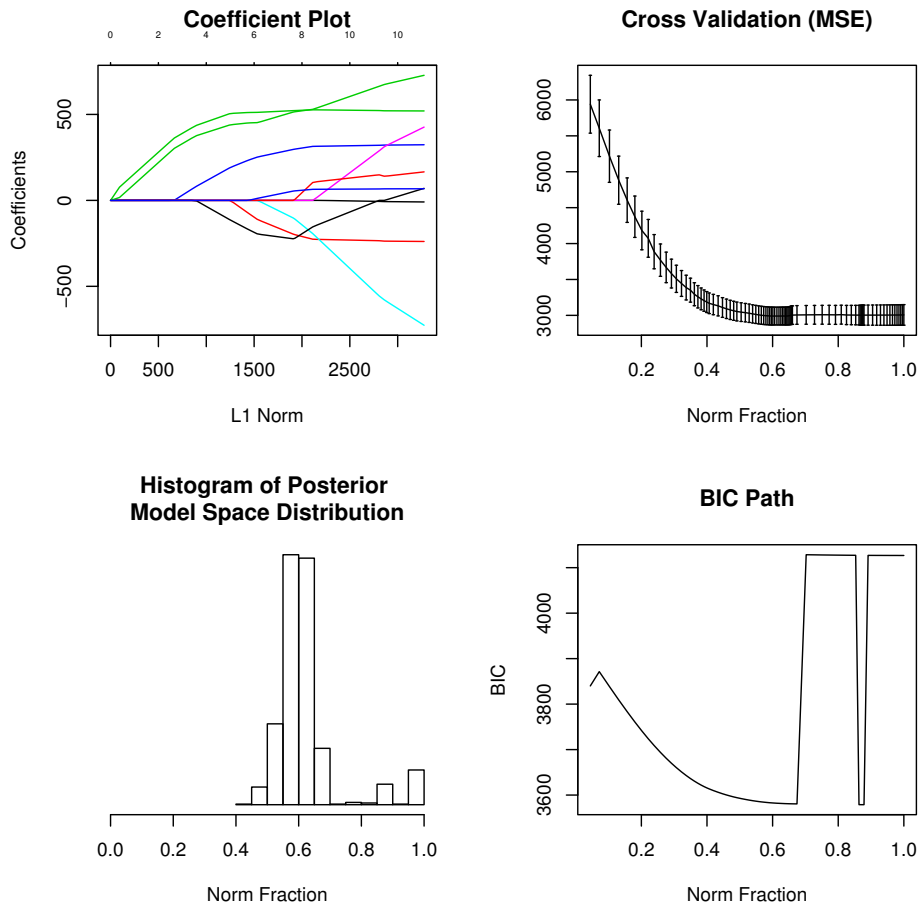


Figure 1: Results of the MCMCMC algorithm. Top left: coefficient path plot, produced by `glmnet`. Top right: cross validation of MSE over the LASSO path. Bottom left: histogram of the posterior distribution over the model space ($\mathbf{P}(M|D)$). Bottom right: plot of BIC over the LASSO path.

The remaining examples are all binomial family generalized linear models.

Example 2: South African Heart Disease Data The South African Heart Disease data was used by Park and Hastie (2007) to demonstrate the `glm` algorithm. This data set consists of nine covariates, with the response being the presence of heart disease - a binary covariate. Cross validation on a LASSO fit of the data suggests model uncertainty issues, see Figure 2. The MCMCMC algorithm was ran for 10000 iterations, with the first 2500 discarded to allow time for convergence. 54% of all proposed moves were accepted, with 99% of these made with probability one. 332 of the 462 cases fit were predicted correctly using the mean coefficient vector and a decision boundary at .5. This is comparable to standard techniques.

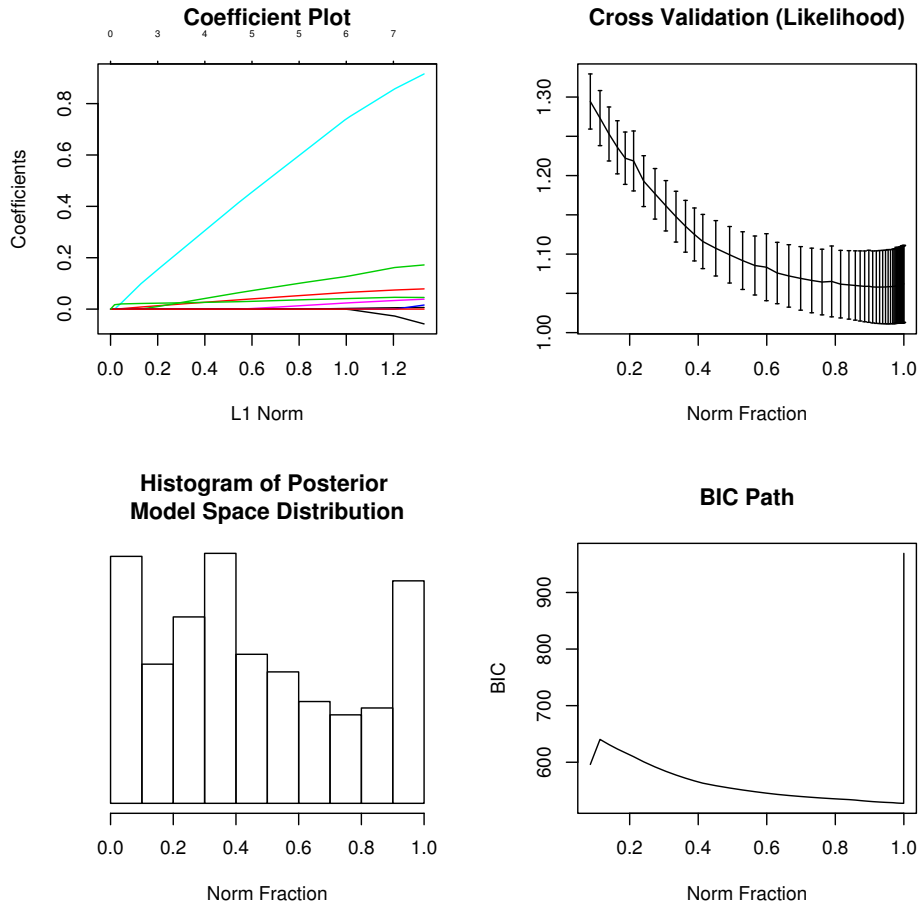


Figure 2: Results of the MCMCMC algorithm. Top left: coefficient path plot, produced by `glmnet`. Top right: cross validation of log-likelihood over the LASSO path. Bottom left: histogram of the posterior distribution over the model space ($\mathbf{P}(M|D)$). Bottom right: plot of BIC over the LASSO path.

Example 3: Leukemia Data The Leukemia data was used by Yeung et al. (2005) to demonstrate the iterative Bayesian model averaging algorithm (iBMA). We only consider the two class version of the data set. The data set consists of 38 training samples and 34 test samples. There are 3051 covariates, each representing a gene expression. The response is the presence of leukemia. Cross validation over a LASSO path again shows potential model uncertainty issues. The MCMCMC algorithm was ran for 10000 iterations, with the first 2500 discarded to allow time for convergence. 53% of proposed moves were accepted, with 92% of these moves made with probability one. Every 5th observation of the tail was used to reduce autocorrelation. See Figure 3 for plots of the results.

The “averaged” model predicted 33 of 34 of the cases in the test set. As in Yeung et al. (2005), we also evaluate performance using Brier score. The Brier score for this model was 1.385. Yeung et al. achieved a score of 1.5, with 33 of 34 test cases predicted correctly. Alternately, we can drop all covariates with $\mathbf{P}(\beta_i = 0) < .5$ so that our solution is sparse. This strategy leaves 18 genes (compared to 20 in Yeung et al.) plus an intercept. This model predicts 31 of 34 test cases correctly, and has a Brier score of 1.876.

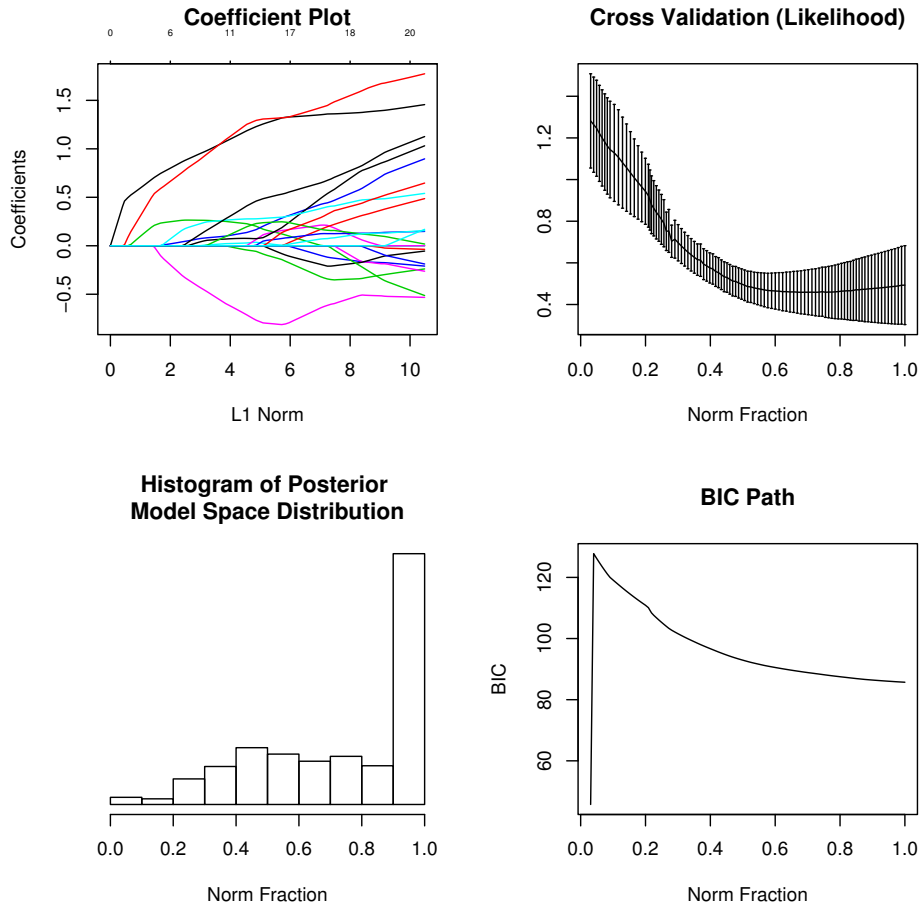


Figure 3: Results of the MCMCMC algorithm. Top left: coefficient path plot, produced by `glmnet`. Top right: cross validation of log-likelihood over the LASSO path. Bottom left: histogram of the posterior distribution over the model space ($\mathbf{P}(M|D)$). Bottom right: plot of BIC over the LASSO path.

Example 4: Breast Cancer Data The breast cancer data was also used by Yeung et al. (2005) to demonstrate the `iBMA` algorithm. This data set contain about 10000 covariates, 78 training cases and 19 test cases. A LASSO fit of the data, with optimal point chosen by AIC or BIC, gives very poor performance on the test data. The MCMCMC algorithm was ran for 10000 iterations, with the first 2500 discarded to allow time for convergence. 53% of proposed moves were accepted, with 92% of these moves made with probability one. Every 5th observation of the tail was used to reduce autocorrelation. See Figure 4 for plots of the results.

The “averaged” model predicts 11 of 19 of the test cases correctly, with a Brier score of 4.79. This suggests a weakness in this technique. The poor performance of this technique on this dataset may be due to poor performance of the LASSO on the data.

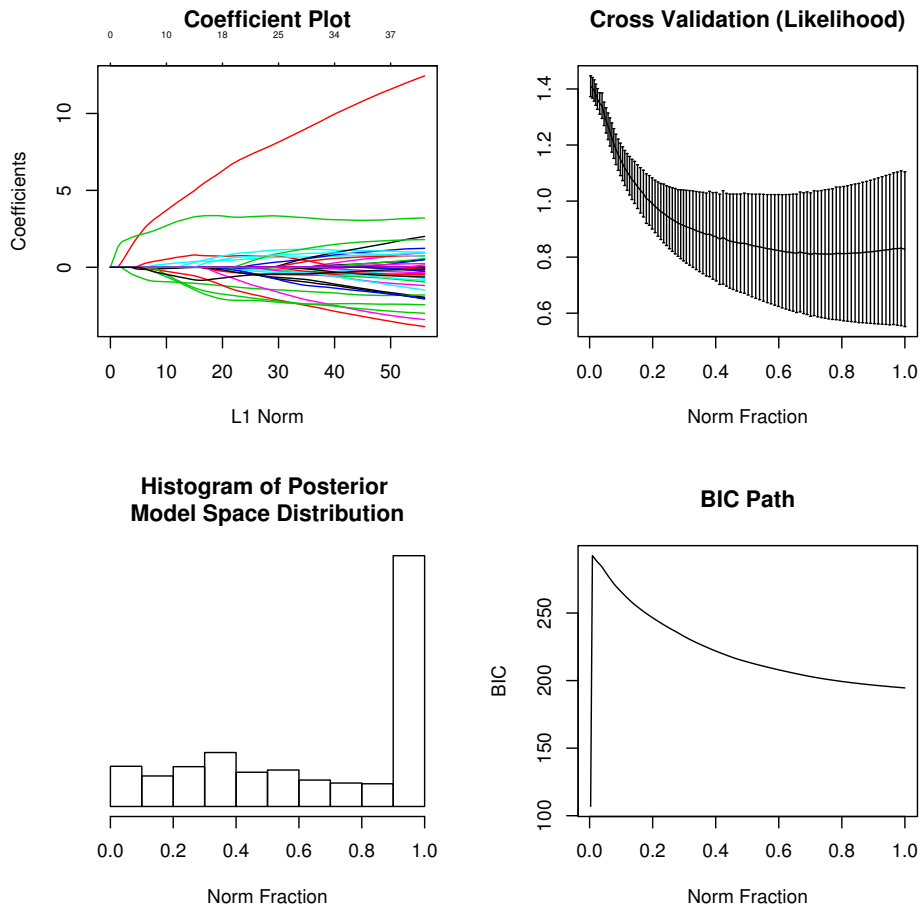


Figure 4: Results of the MCMCMC algorithm. Top left: coefficient path plot, produced by `glmnet`. Top right: cross validation of log-likelihood over the LASSO path. Bottom left: histogram of the posterior distribution over the model space ($\mathbf{P}(M|D)$). Bottom right: plot of BIC over the LASSO path.

Future Work There are several areas of potential improvement in this combination algorithm. First, there are issues with the proposal distribution in the Metropolis-Hastings step. The most significant of these is a tendency to become trapped near the edges of the regularization path. A new proposal distribution or an automatic way to smartly choose the tuning parameter for the distribution could solve this issue. Re-parameterization of the model space could also help.

Second, the BIC approximation is fast, but it is in many cases poor. BIC approximations are worst when the number of observations is very small compared to the number of covariates. This is a situation where regularization techniques are very desirable. Therefore, we would prefer that the algorithm did not rely so heavily on BIC. Computing integrated likelihood is a tough problem, so this improvement will probably be the most challenging to make.

Third, the algorithm did not work well compared to other methods on the fourth example data set. We suspect that this was a result of poor LASSO performance on the data set. A closer investigation will hopefully reveal the true cause, and will certainly aid in more appropriate application of the algorithm.

Fourth, the question of the model space prior has not been fully addressed. Though the flat prior is easy to use, there are others that may be more appropriate. For example, choosing a prior that prefers sparse models could be of some benefit.

Fifth, the current coding of the algorithm in R is somewhat slow. However, it is far from optimal, and speed can be improved greatly through use of a more efficient programming language. The benefits of improved speed are numerous and obvious.

Finally, it is worth noting that a simple direct use of the BIC curve over the regularization path may be as effective as the MCMCMC algorithm. This would be done by simply using BIC to take a weighted average. As in the MCMCMC, BIC could be replaced by a better approximation of the integrated likelihood.

References

- [1] B. Efron, T. Hastie, and R. Tibshirani, *Least angle regression*, The Annals of Statistics **32** (2004), 407–451.
- [2] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, Tech. report, Department of Statistics, Stanford University, 2008.
- [3] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, *Bayesian model averaging: a tutorial*, Statistical Science **14** (1999), no. 4, 382–417.
- [4] M.-Y. Park and T. Hastie, *An L_1 regularization path algorithm for generalized linear models.*, Journal of the Royal Statistics Society Series B **69** (2007), 659–677.
- [5] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistics Society Series B **58** (1996), 385–395.
- [6] Ka Yee Yeung, Roger E. Bumgarner, and Adrian E. Raftery, *Bayesian model averaging: Development of an improved multi-class, gene selection and classification tool for microarray data*, Bioinformatics **21** (2005), 2394–2402.
- [7] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the Royal Statistical Society, Series B **67** (2005), 301–320.