A Non-Markovian Method for Full, Parametric Bayesian Inference

Michael Braun
Edwin L. Cox School of Business
Southern Methodist University
Dallas, TX 75275
braunm@smu.edu

Paul Damien
McCombs School of Business
University of Texas at Austin
Austin, TX 78712
paul.damien@mccombs.utexas.edu

September 30, 2018

To appear in:
Marketing and Big Data Technologies: New Trends and Applications.
Ed. by S. Tsafarakis and N. Matsatsinis. Springer-Verlag.

Abstract

Despite recent advances in high-speed computing, Bayesian inference in high dimensional hierarchical models remains a non-trivial undertaking. Markov chain Monte Carlo (MCMC) methods have come a long way in resolving several problems in this regard, but these methods have, in turn, introduced a different set of computational issues like monitoring convergence rates. Such issues, typically, get accentuated in marketing research and practice when non-conjugate Bayesian hierarchical models are used. Here, we use a new method to generate *independent* samples from posterior distributions in these types of Bayesian models, obviating many of the difficulties associated with MCMC algorithms. Challenging illustrative analysis exemplifies the ease with which one can implement this method.

1 Introduction

Since the seminal paper by Gelfand and Smith (1990), the corpus of statistical literature on Bayesian computational methods has exploded. It would be difficult to list the numerous excellent papers that deal with so many aspects of Markov chain Monte Carlo (MCMC), and other simulation-based methods. Hence, we cite three books, and the hundreds of references therein, that have helped statisticians over the years: Gelman et al. (2003), Liu and Sabatti (2000), Chen, Shao, and Ibrahim (2000), and Brooks et al. (2010). The popularity of MCMC is due, in part, to its general applicability and theoretical properties. However, it is well-known that the main drawback of using MCMC is that each chain generates dependent samples. Knowing how long to run the chain, and when to begin collecting samples for posterior inference, are critical considerations. Indeed, there is a vast literature that addresses the difficulties and proposed remedies in diagnosing convergence of MCMC chains.

In Braun and Damien (2016), a new estimation algorithm that generates independent samples from a target posterior distribution *in parallel* is introduced. The ability to sample in parallel is a notable contrast with MCMC for which, in the absence of parallel independent chains, samples are collected sequentially. With the new method (henceforth denoted as BD), there is no need to concern oneself with issues like chain convergence and autocorrelation. Our interest is in conducting full Bayesian inference on complex hierarchical models, with or without conjugacy, but with thousands of heterogeneous units, without MCMC. One key inspiration for a non-MCMC approach to doing Bayesian inference for such hierarchical models is perfectly articulated by Papaspiliopoulos and Roberts (2008): *"However, to date, there has been little theoretical analysis linking the stability of the Gibbs sampler to the structure of hierarchical models."*

The only restrictions on the BD method are that one must be able to compute the unnormalized log posterior of the parameters (or a good approximation of it); that the posterior distribution must be bounded from above over the parameter space; and that one is able to locate any local maxima of the log posterior function using available computing resources. There are no conditions of conjugacy or even conditional conjugacy. We find that these conditions are easily met for most parametric models, especially given the maturity of existing nonlinear optimization algorithms, making the new approach a viable alternative to MCMC. BD show that the method is scalable under the assumption of conditional independence across heterogeneous units.

In this paper we illustrate the relative performance of the BD algorithm for both hierarchical and non-hierarchical models. We restate the BD method, and the theoretical justification for it, in Section 2. Section 3 includes some examples of the method in action. Finally, in Section 4, we discuss practical issues that one should consider when implementing the algorithm, including some "lessons from the field."

2 The Method

The non-MCMC BD approach is a variant on well-known importance sampling methods. The goal is to sample $\theta \in \Omega$ from a posterior density $\pi(\theta|y) \propto f(y|\theta)\pi(\theta) = \mathcal{D}(\theta, y)$, where $\sup \mathcal{D}(\theta^*, y) = f(y|\theta^*)\pi(\theta^*) = c_1 < \infty$, y is some observed data, and θ^* is the posterior mode.

Just like in importance sampling, we select a proposal density $g(\theta)$ chosen such that the mode of $g(\theta)$ is also θ^*. Define $c_2 = g(\theta^*)$, and define the function

$$\Phi(\theta|y) = \frac{\mathcal{D}(\theta, y) \cdot c_2}{g(\theta) \cdot c_1}. \tag{1}$$

Through substitution and rearranging terms,

$$\pi(\theta|y) \propto \Phi(\theta|y) \cdot g(\theta) \cdot \frac{c_1}{c_2}. \tag{2}$$

Note that $g(\theta)$ must be chosen such that $0 < \Phi(\theta|y) \leq 1$ holds at least for any θ with a non-negligible posterior density.

Next, let $u|\theta, y$ be an auxiliary variable that is distributed uniformly on $\left(0, \frac{\Phi(\theta|y)}{\pi(\theta|y)}\right)$, and construct a joint density of $\theta|y$ and $u|\theta, y$:

$$p(\theta, u|y) = \frac{\pi(\theta|y)}{\Phi(\theta|y)} \mathbb{1}\left[u < \Phi(\theta|y)\right] \tag{3}$$

$$\propto g(\theta) \mathbb{1}\left[u < \Phi(\theta|y)\right] \tag{4}$$

By integrating Equation 3 over u, the marginal density of $\theta|y$ is

$$p(\theta|y) = \frac{\pi(\theta|y)}{\Phi(\theta|y)} \int_0^{\Phi(\theta|y)} du = \pi(\theta|y). \tag{5}$$

Thus, sampling from $p(\theta|y)$ is equivalent to simulating from the target posterior $\pi(\theta|y)$.

Using Equations 2 and 3, the marginal density of $u|y$ is

$$p(u|y) \propto \int_\theta \mathbb{1}\left[u < \Phi(\theta|y)\right] g(\theta)\, d\theta = q(u). \tag{6}$$

This $q(u)$ function is the probability that any candidate draw from $g(\theta)$ will satisfy $\Phi(\theta|y) > u$. The proposed sampler comes from recognizing that $p(\theta, u|y)$ can also be written differently from, but equivalently to, Equation 3.

$$p(\theta, u|y) = p(\theta|u, y)\; p(u|y). \tag{7}$$

Using the definitions in Equations 1, 2, and 3, we get

$$p(\theta|u,y) \propto \frac{g(\theta)\mathbb{1}\left[u < \Phi(\theta|y)\right]}{p(u|y)}. \tag{8}$$

To sample *directly* from $p(\theta, u|y)$, one needs only to sample from an approximation to $p(u|y)$ and then sample repeatedly from $g(\theta)$ until $\Phi(\theta|y) > u$. The samples of θ form the marginal distribution $p(\theta|y)$, and since sampling from $p(\theta|y)$ is equivalent to sampling from $\pi(\theta|y)$, they form an empirical estimate of the target posterior density.

In Equation 6, we see that $p(u|y)$ is proportional to the function $q(u)$. Thus, we approximate $q(u)$ empirically by repeatedly sampling from $g(\theta)$, and computing $\Phi(\theta|y)$ for each of those proposal draws. To avoid numerical precision issues, we actually sample a transformed variable $v = -\log u$ instead of u. Applying a change of variables, $q_v(v) = q(u)\exp(-v)$. With $q_v(v)$ denoting the "true" CDF of v, let $\widehat{q}_v(v)$ be the empirical CDF of v after taking M proposal draws from $g(\theta)$. Order the proposals such that $0 < v_1 < v_2 < \ldots < v_M < \infty$. Because $\widehat{q}_v(v)$ is discrete, we can sample from a density proportional to $q(u)\exp(-v)$ by partitioning the domain into $M+1$ segments with the break point of each partition at each v_i. The probability of sampling a new v that falls between v_i and v_{i+1} is now

$$\varpi_i = \widehat{q}_v(v)\left[\exp(-v_i) - \exp(-v_{i+1})\right], \tag{9}$$

so we can sample an interval bounded by v_i and v_{i+1} from a multinomial density with weights proportional to ϖ_i. Once we have the i that corresponds to that interval, we can sample the continuous v by sampling ϵ from a standard exponential density, truncated on the right at $v_{i+1} - v_i$, and setting $v = v_i + \epsilon$. Putting it all together, we sample v by sampling i with weight ϖ_i, sampling a standard uniform random variable η, and then finally setting

$$v = v_i - \log\left[1 - \eta\left(1 - \exp(v_i - v_{i+1})\right)\right]. \tag{10}$$

To sample R independent draws from the target posterior, we need R "threshold" draws of v. Then, for each v, we repeatedly sample from $g(\theta)$ until $-\log(\Phi(\theta|y)) < v$. Once we have a θ that meets this criterion, we save it as a valid sample from $\pi(\theta|y)$. The complete algorithm is summarized below as Algorithm 1.

A restriction on $g(\theta)$ is that the inequality $0 < \Phi(\theta|y) \leq 1$ must hold, at least for any θ with a non-negligible posterior density. In principle, it is up to the researcher to choose $g(\theta)$, and some choices may be more efficient than others. We have found that a multivariate normal (MVN) proposal distribution, with mean at θ^*, works well because it is an asymptotic approximation to the posterior density itself. The covariance is the inverse Hessian at θ^*, times a scaling constant s. That proposal distribution will be valid as long as s is large enough so that $\Phi(\theta|y)$ is between

Algorithm 1 Algorithm to collect R samples from $\pi(\theta|y)$ (Braun and Damien 2015)

```
1: $R \leftarrow$ number of required samples from $\pi(\theta|y)$
2: $M \leftarrow$ number of proposal draws for estimating $\widehat{q}_v(v)$.
3: $\theta^* \leftarrow$ mode of $\mathcal{D}(\theta, y)$
4: $c_1 \leftarrow \mathcal{D}(\theta^*, y)$
5: FLAG← TRUE
6: **while** FLAG **do**
7:    Choose new proposal distribution $g(\theta)$
8:    FLAG←FALSE
9:    $c_2 \leftarrow g(\theta^*)$.
10:   **for** $m := 1$ **to** $M$ **do**
11:      Sample $\theta_m \sim g(\theta)$.
12:      $\log \Phi(\theta_m|y) \leftarrow \log \mathcal{D}(\theta_m, y) - \log g(\theta_m) - \log c_1 + \log c_2$.
13:      $v_m = -\log \Phi(\theta_m|y)$
14:      **if** $\log \Phi(\theta_m|y) > 0$ **then**
15:         FLAG← TRUE
16:         **break**
17:      **end if**
18:   **end for**
19: **end while**
20: Reorder elements of $v$, so $0 < v_1 < v_2 < \ldots < v_M < \infty$. Define $v_{M+1} := \infty$
21: **for** $i := 1$ **to** $M$ **do**
22:    $\widehat{q}_v(v_i) \leftarrow \sum_{j=1}^{M} \mathbb{1}\left[v_j < v_i\right]$.
23:    $\varpi_i \leftarrow \widehat{q}_v(v_i)\left[\exp(-v_i) - \exp(-v_{i+1})\right]$.
24: **end for**
25: **for** $r = 1$ **to** $R$ **do**
26:    Sample $j \sim$ Multinomial$(\varpi_1 \ldots \varpi_M)$.
27:    Sample $\eta \sim$ Uniform(0,1).
28:    $v^* \leftarrow v_j - \log\left[1 - \eta\left(1 - \exp\left(v_j - v_{j+1}\right)\right)\right]$.
29:    $p \leftarrow 0$
30:    $n_r \leftarrow 0$. {Counter for number of proposals}
31:    **while** $p > v^*$ **do**
32:       Sample $\theta_r \sim g(\theta)$.
33:       $p \leftarrow -\log \Phi\left(\theta_r|y\right)$.
34:       $n_r \leftarrow n_r + 1$.
35:    **end while**
36: **end for**
37: **return** $\theta_1 \ldots \theta_R$ (plus $n_1 \ldots n_R$ and $v_1 \ldots v_M$ if computing a marginal likelihood).
```

0 and 1 for any plausible value of θ, and that the mode of $g(\theta)$ is at θ^*.

2.1 Discussion of the Algorithm and MCMC

The primary advantages of the BD method are that (a) *independent* samples from the target density are obtained, possibly in parallel; and that (b) it is straightforward to implement. Other than the computer code that is needed to implement the steps of the algorithm, all that is required are:

1. a function that computes the log of the unnormalized target posterior density (i.e., $\log \mathcal{D}(\theta, y)$); and
2. functions to sample from, and to compute the density of, $g(\theta)$, which is typically a standard, known density.

Optionally, one might also want to compute derivatives of $\log \mathcal{D}(\theta, y)$ to help find θ^* using numerical nonlinear optimization routines. The Hessian matrix will certainly be useful when $g(\theta)$ is an MVN distribution.

The log posterior density is additive across the components of the hierarchical model; i.e., $\log \mathcal{D}(\theta, y) = \log f(y|\theta) + \log \pi(\theta|\cdot) + \dots$ This makes it easy to make changes to the model specification. To try a different prior, just change the function that computes that one additive term. For a hierarchical model with conditionally independent units, the data likelihood can be further decomposed as $\log f(y|\theta) = \sum_i \log f(y_i|\theta_i)$. In contrast, Gibbs sampling requires combining terms that have common parameters into conditional posterior distributions. Whether or not model terms possess conditional conjugacy has a direct effect on the simplicity of the implementation of the sampler. Thus, researchers are inclined to choose certain priors over others for computational convenience even if they are not entirely appropriate for the problem. One key example is finding more flexible or informative alternatives to the inverse Wishart distribution as a prior on covariance parameters (Gelman 2006). Without conditional conjugacy, the researcher needs another way to draw from each conditional posterior, and there are few options that are both efficient and easy to estimate in high dimensions. As examples, consider the difficulty in adapting the scale parameter of a random walk Metropolis-within-Gibbs update, or computing high-order derivatives for Hamiltonian-like MCMC alternatives as in Girolami and Calderhead (2011). Thus, there is quite a bit of work facing the researcher who wants to build an efficient MCMC sampler.

Admittedly, MCMC dominates on one important dimension: its broadly general applicability. But in many problems, the BD method is an attractive alternative to MCMC because the samples from the posterior distribution are independent, hence fewer draws are needed than an MCMC algorithm to achieve an equivalent *effective sample size*. Additionally, the draws can be collected in parallel by exploiting parallel computing technology. Approaches for MCMC-based Bayesian inference that also take advantage of parallel computation exist; see, for example, Suchard et al. (2010). An example is a parallel implementation of a multivariate slice sampler (MSS) (Tibbits, Haran, and Liechty 2010). But the MSS itself remains a Markovian algorithm, and thus will still generate dependent draws. Using parallel technology to generate a single draw from a distribution is not the same as generating all of the required draws themselves in parallel. On the other hand, the sampling steps of our approach can be run in their *entirety* in parallel.

3 Illustrative Analysis

We now provide some examples of the BD method in action, and highlight its relative advantages over MCMC.

3.1 A Hierarchical Non-Gaussian Linear Model

Consider this motivating example of a linear hierarchical model discussed by Papaspiliopoulos and Roberts (2008). They provide excellent insights on why MCMC fails even in this deceptively simple illustration.

$$Y = X + \epsilon_1 \tag{11}$$

$$X = \Theta + \epsilon_2 \tag{12}$$

In this example, each Y is an observation, each X is the latent mean for the prior on Y, and ϵ_1 and ϵ_2 are random error terms, each with mean 0. Papaspiliopoulos and Roberts note that to improve the robustness of inference on X to outliers of Y, it is common to model ϵ_1 as having heavier tails than ϵ_2 . Let $\epsilon_1 \sim \text{Cauchy}(0,1)$, $\epsilon_2 \sim N(0,5)$, and $\Theta \sim N(0, 50{,}000)$, and suppose there is only one observation available, $Y = 0$. We plot the posterior joint distribution of X and Θ in Figure 1; the contours represent the logs of the computed posterior densities. Around the mode, X and Θ appear uncorrelated, but in the tails they are highly dependent. Papaspiliopoulos and Roberts show that the Gibbs sampler performs extremely poorly in this case. Indeed, they note that almost all diagnostic tests will erroneously conclude that the chain has converged. The reason for this failure is that the MCMC chains are attracted to, and get "stuck" in, the modal region where the variables are uncorrelated. Once the chain enters the tails, where the variables are more correlated, the chains moves slowly, or not at all.

To start our method, the posterior mode, and Hessian of the log posterior at the mode, are $\theta^* = (0, 0)$ and

$$H = \begin{pmatrix} -2.2 & 0.2 \\ 0.2 & -0.2 \end{pmatrix}$$

Our proposal distribution $g(\theta)$ is a bivariate normal with mean θ^* and covariance $-sH^{-1}$, with $s = 37$. This scaling factor was the smallest value of s for which $\Phi(\theta|y) \leq 1$ for all $M = 20{,}000$

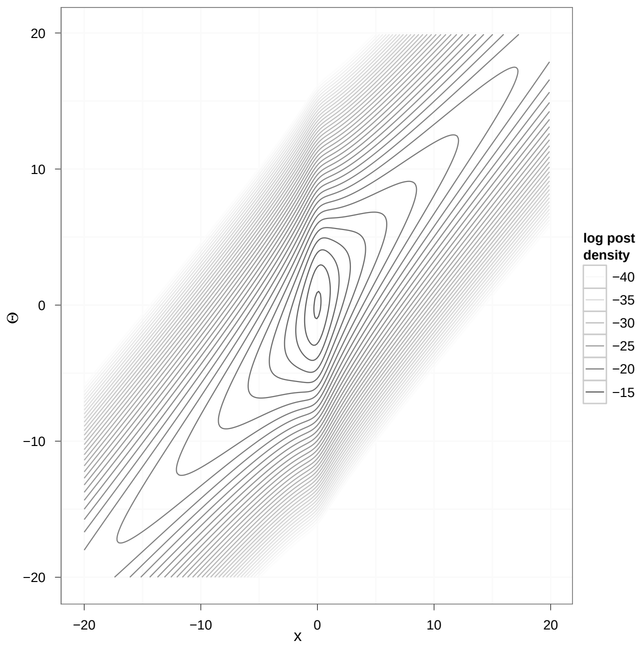

Figure 1: Contours of the ”true” posterior distribution of the non-Gaussian linear model example.

of the proposal draws. After setting $N = 30,000$, we followed the algorithm to collect 50,000 independent posterior draws of X and Θ.

In Figure 2, we plot each of the independent draws, where darker regions represent higher values of log posterior density. Not only does the BD method pick up the correct shape of the regions of high posterior mass near the origin, but also the dependence in the long tails. As a point of contrast, consider Figure 3, which shows, on axes of the same scale, the estimated posterior that we would have got with a Laplace approximation by collecting all samples from a $MVN(0, -H^{-1})$ distribution. This approximation is more concentrated around the mode, and it fails to capture any of the tail dependence in the true distribution.

This contrast also helps us understand why, in this case, we needed to scale the Hessian by such a large factor to get a valid proposal distribution. The tails of the multivariate normal are much thinner than that of the target posterior, and so without a diffuse proposal distribution it is likely that $\Phi(\theta|y) > 1$ for many of the proposal draws. One might be concerned that “over-scaling” would make it too easy for a bad proposal draw to be accepted into the posterior. The method corrects for this possibility by transforming the acceptance threshold in the same way, so the BD approach will whittle away at the proposal distribution. Only samples from the posterior distribution remain. Table 1 presents the number of attempts required for each of the 50,000 posterior draws. We see that the vast majority of proposal samples are accepted on the first attempt, but that there are many draws for which it takes many attempts before getting an acceptance. Of course, these draws correspond to draws with higher values of the acceptance

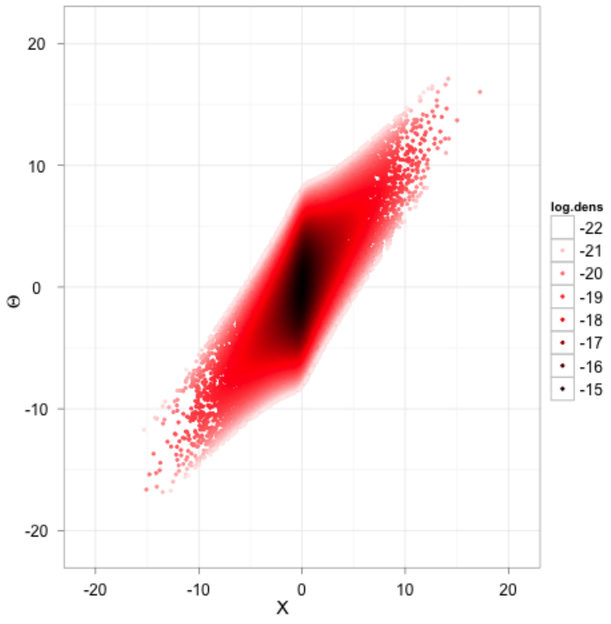

Figure 2: Posterior draws from non-Gaussian linear regression example, using GDS. Darker colors represent regions of higher posterior density

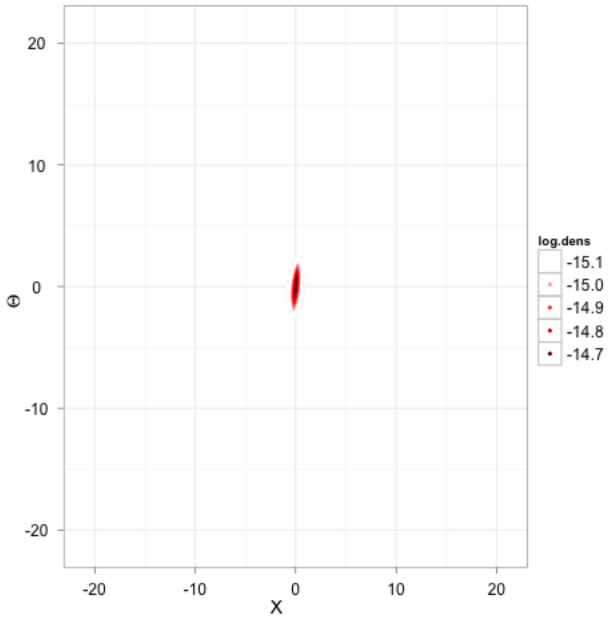

Figure 3: Posterior draws from non-Gaussian linear regression example, using a Laplace approximation. Darker colors represent regions of higher posterior density

Attempts	1	2	3	4	5	6	7	8	9	10+
Samples	48126	1362	289	108	54	17	15	5	4	20

Table 1: Number of attempts for each sample from the posterior distribution in the non-Gaussian linear hierarchical model example. Maximum number of attempts is 624.

threshold u. Our point here is that the method remains selective when accepting proposal draws, even though the acceptance threshold for some, but not all, of the draws may be low.

3.2 Probit Regression

In a probit model, for individuals $i = 1 \dots H$, we observe a binary outcome $z_i \in (0, 1)$. Whether or not an individual's outcome is 0 or 1 depends on an unobserved latent variable $y_i = x_i\theta + \varepsilon_i$, where x_i is vector of observed covariates for person i, θ is a vector of coefficients, and $\varepsilon_i \sim N(0, 1)$ is a source of random variation. The posterior density of θ is

$$\pi(\theta|z_{1:H}, x_{1:H}) \propto \pi(\theta) \prod_{i=1}^{H} \Phi(x_i\theta)^{z_i} \left(1 - \Phi(x_i\theta)\right)^{1-z_i} \tag{13}$$

where Φ is the standard normal cdf function (not the function defined in Section 2). Details on the MCMC sampling algorithm for the probit model are found in Albert and Chib (1993); we use the implementation in the *MCMCpack* package for R (Martin, Quinn, and Park 2011) for this example. We started the MCMC sampler at the posterior mode, and generated 1,000,000 draws; this took 67 minutes on a recent vintage Apple Mac Pro with two Intel Xeon X5670 processors and 32GB of RAM. Trace plots for these draws are in Figure 4. In Table 2 we show the effective sample sizes for these 1,000,000 draws for each of the 10 parameters, as well as results of a Raftery-Lewis diagnostic test (Raftery and Lewis 1996) for determining the number of iterations required to estimate the posterior median within 0.01 with probability 0.95. These statistics were computed using the R *coda* package (Plummer et al. 2006). Also shown are the lag-250 autocorrelations for the 10 parameters.

The results in Table 2 may be striking to a reader expecting that one million draws should be sufficient to estimate a 10-parameter binary probit model. However, the effective sample size for many of the parameters, relative to the total number of draws, is quite low. If we wanted to estimate the posterior median at the precision and confidence suggested by the Raftery-Lewis test for all parameters, we would need to run the chain for nearly 122 million iterations. Moreover, barring two parameters, the lag-250 autocorrelations are very large. In contrast, using our method, just under 10,000 *independent* draws from the marginal posteriors for all parameters would be sufficient.

To estimate the same model using the BD method, we set the proposal distribution to be a multivariate normal, with a mean at the posterior mode, and a covariance matrix of the inverse negative Hessian at the mode, scaled by 1.04. This scale is the smallest factor for which $\Phi(\theta|y) \leq$

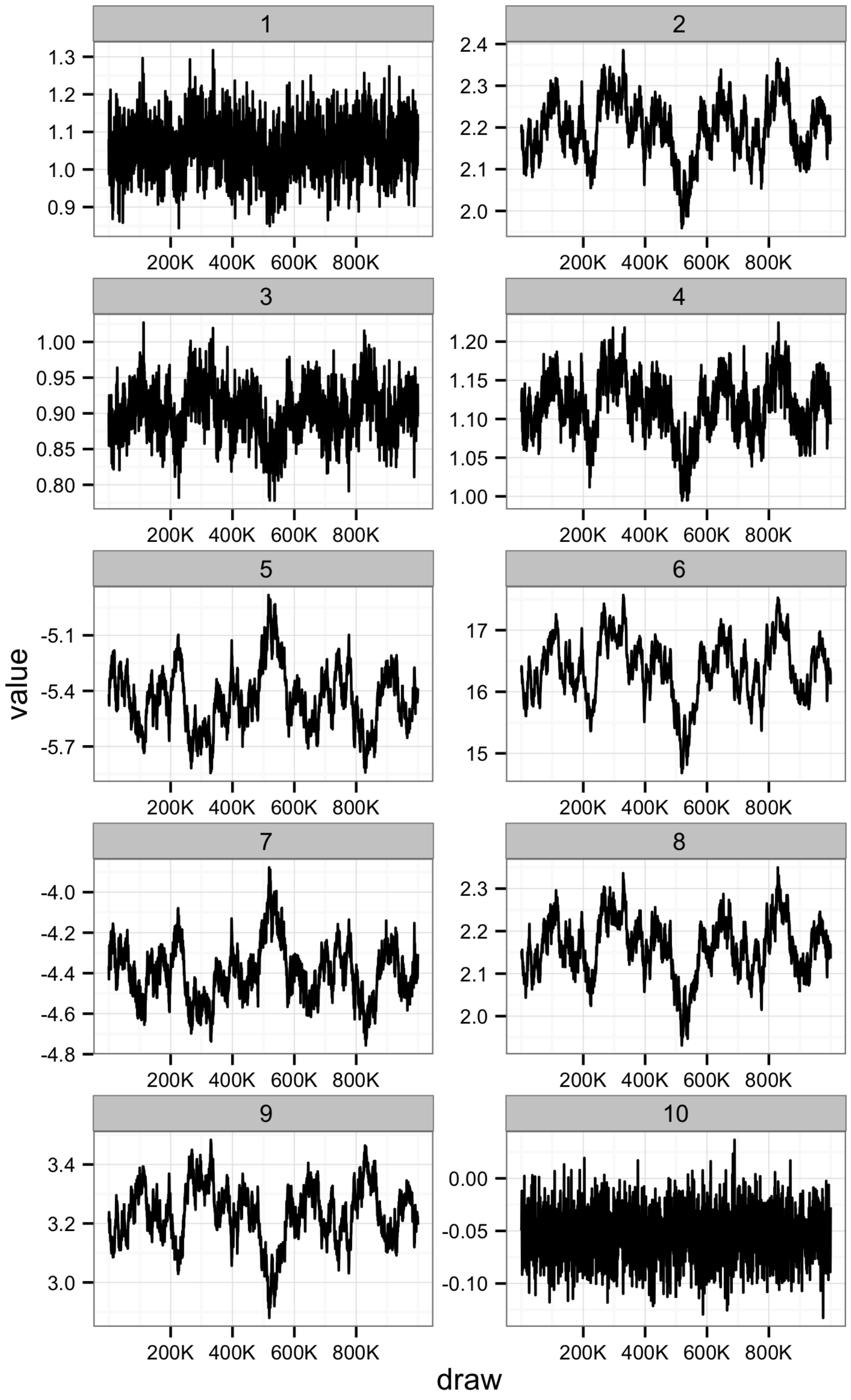

Figure 4: Traceplots for Gibbs sampling chains (thinned by 500 only to reduce the size of graphics file) for the binary probit regression.

	Effective	Raftery-Lewis Diagnostics			
	Sample Size	Burn-In (M)	Required Draws (N)	Scale Factor (I)	ACF-250
1	11585	296	971916	101	.173
2	256	27324	82969920	8640	.938
3	3296	4998	15474522	1610	.542
4	916	13923	41022072	4270	.814
5	105	33000	100016400	10400	.971
6	17	30728	94116552	9800	.991
7	241	22300	68337012	7120	.940
8	165	22743	69260738	7210	.957
9	133	39783	121862928	12700	.964
10	19731	115	382053	39	.014

Table 2: Effective sample sizes, Raftery-Lewis diagnostics, and lag-250 autocorrelations, for the binary probit example. Effective sizes are based on 1 million draws. Raftery-Lewis statistics were generated from 250,000 pilot draws, and represent the number of burn-in and collected iterations needed to estimate the posterior median within 0.01 with probability 0.95 for the 10 parameters. The equivalent number of GDS independent draws is 9,607.

1 for all proposal draws. Figure 5 compares the estimates of the marginal posterior distributions using MCMC and BD, and Figure 6 compares the posterior means and sampling errors. In both figures, the box is the interquartile range, the crossbar is the posterior mean, and the vertical lines span from the 5th to 95th percentiles. The dark horizontal lines at the top and bottom of each panel show the 5th and 95th percentiles for the posterior interval that one would expect from a Laplace approximation. We see that both methods give comparable estimates, but the point estimates for the posterior means are more precise using our method than using MCMC. It is true that our estimates are computed using a larger effective sample size, but it is unlikely that, in practice, modelers would (or should) need to collect 122 million iterations from any method to do inference on such a low-dimensional problem.

3.3 Hierarchical repeated binary choice

In the probit regression example in Section 3.2, the coefficients are homogeneous across individual units. In this next example, we consider a model that incorporates unobserved heterogeneity in the choice probabilities.

Suppose we have a dataset of N households, each with T opportunities to purchase a particular product. Let y_i be the number of times household i purchases the product, out of the T purchase opportunities. Furthermore, let p_i be the probability of purchase; p_i is the same for all T opportunities, so we can treat y_i as a binomial random variable. The purchase probability p_i is heterogeneous, and depends on both k continuous covariates x_i, and a heterogeneous coefficient vector β_i, such that

$$p_i = \frac{\exp(x_i'\beta_i)}{1+\exp(x_i'\beta_i)},\ i = 1...N. \tag{14}$$

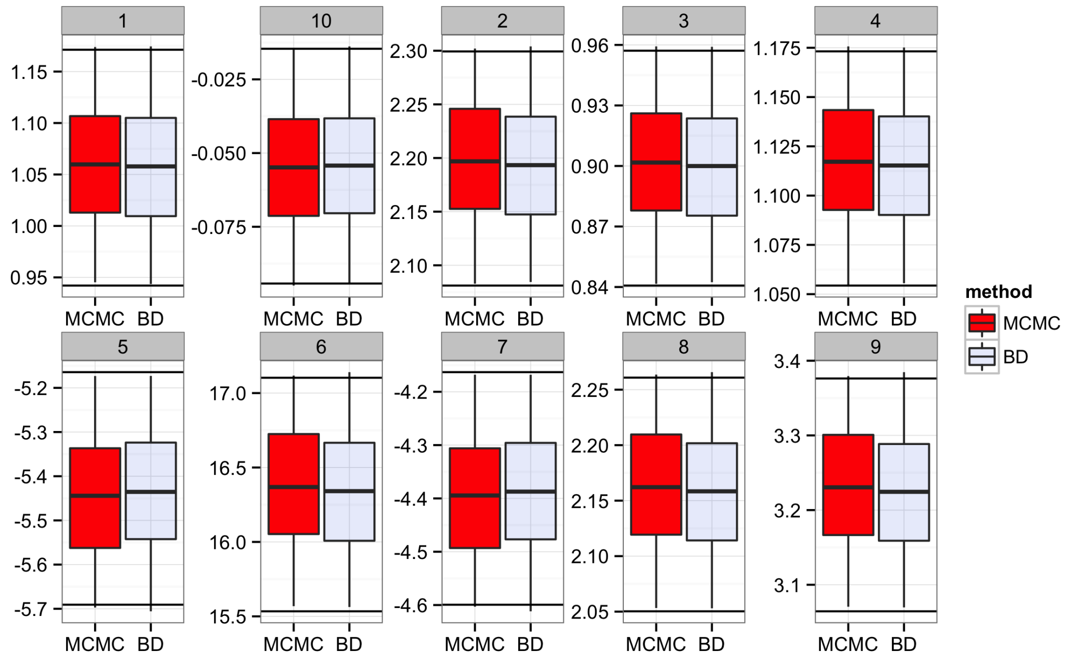

Figure 5: Posterior distributions of parameters in the binary probit regression example.

The coefficients can be thought of as sensitivities to the covariates, and they are distributed across the population of households following a multivariate normal distribution with mean μ and covariance Σ. We assume that we know Σ, but we do not know μ. Instead, we place a multivariate normal prior on μ, with mean 0 and covariance Ω_0. Thus, each β_i, and μ are $k-$dimensional vectors, and the total number of unknown variables in the model is $(N+1)k$.

In this model, we will make an assumption of *conditional independence* across households. A household's purchase count y_i depends on that household's β_i, but not the parameters of any other household, β_j, conditional on other population level parameters. Since μ and Σ depend on β_i for *all* households, we cannot say that y_i and y_j are truly independent. A change in β_i affects μ and Σ, which in turn affect β_j for some other household j. However, if we condition on μ and Σ, then y_i and y_j are independent, so we describe the data likelihoods as conditionally independent.

This conditional independence assumption is what allows us to write the joint likelihood of the data as a product of individual-level probability models. Therefore, the log posterior density, ignoring any normalization constants, is

$$\log \pi(\beta_{1:N}, \mu | Y, X, \Sigma_0, \Omega_0) = \sum_{i=1}^{N} p_i^{y_i} (1-p_i)^{T-y_i} - \frac{1}{2}(\beta_i - \mu)' \Sigma^{-1} (\beta_i - \mu) - \frac{1}{2}\mu' \Omega_0^{-1} \mu \tag{15}$$

An implication of the conditional independence assumption is that the cross-partial derivatives of the unnormalized log posterior density are zero for all pairs of parameters across different

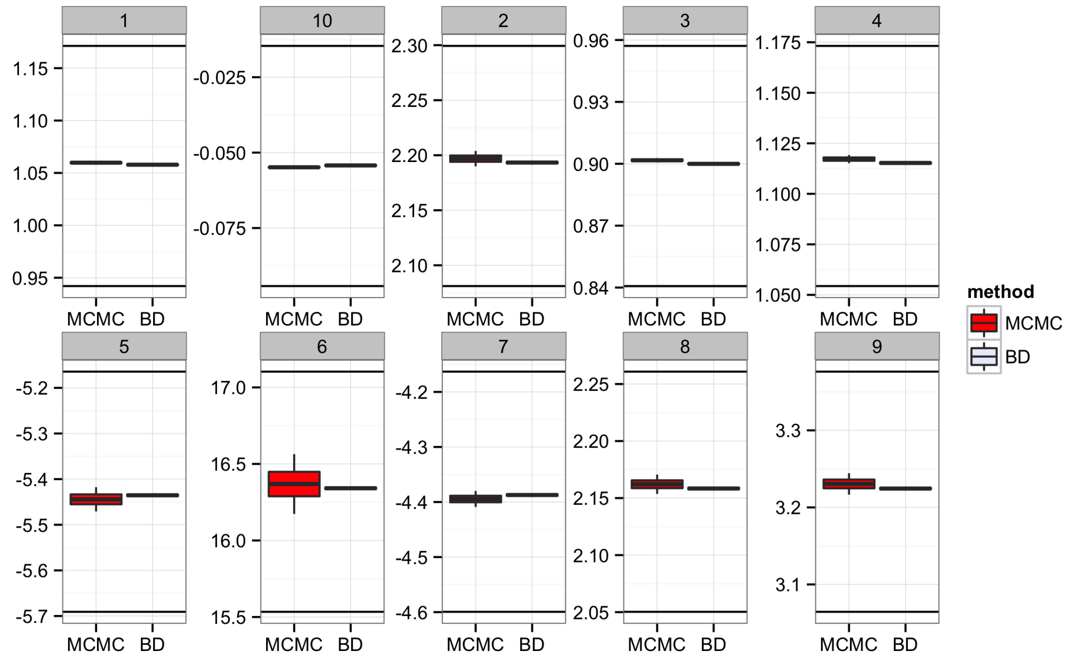

Figure 6: Posterior means and Monte Carlo error for the binary probit regression example.

heterogeneous units. As the number of households in the dataset increases, the number of elements in the Hessian matrix increases quadratically, but the number of *non-zero* elements increases only linearly. The Hessian becomes sparser as the data set gets larger.

We consider a matrix to be sparse if it has a relatively small number of non-zero elements. A sparse matrix can be represented by only the non-zero values, and the row and column indices of those values. All of the elements are known to be zero, so they do not need to be stored explicitly. Thus, the amount of memory required to store a sparse matrix grows with the number of non-zero elements, as opposed to the product of the number of rows and columns. Also, linear algebra operations are more efficient on compressed sparse matrices, because operation on the non-zero elements can be ignored. These computational advantages come into play in nearly all of the steps of the BD algorithm. Although the sparsity of the Hessian of the log posterior density is not a requirement for the BD algorithm, it is that sparsity that makes the algorithm scalable.

The *sparsity pattern* of the Hessian depends on how the variables are ordered within the vector. One such ordering is to group all of the coefficients for each unit together.

$$\beta_{11}, ..., \beta_{1k}, \beta_{21}, ..., \beta_{2k}, \dots , \ \dots , \ \beta_{N1} \ , \ \dots , \ \beta_{Nk}, \mu_1, ..., \mu_p \tag{16}$$

In this case, the Hessian has a ”block-arrow” structure. For example, if $N = 6$ and $k = 2$, then there are 14 total variables, and the Hessian will have the sparsity pattern in Figure 7a.

```
 [1,] | | . . . . . . . . . . | |
 [2,] | | . . . . . . . . . . | |
 [3,] . . | | . . . . . . . . | |
 [4,] . . | | . . . . . . . . | |
 [5,] . . . . | | . . . . . . | |
 [6,] . . . . | | . . . . . . | |
 [7,] . . . . . . | | . . . . | |
 [8,] . . . . . . | | . . . . | |
 [9,] . . . . . . . . | | . . | |
[10,] . . . . . . . . | | . . | |
[11,] . . . . . . . . . . | | | |
[12,] . . . . . . . . . . | | | |
[13,] | | | | | | | | | | | | | |
[14,] | | | | | | | | | | | | | |
```

(a) A "block-arrow" sparsity pattern

```
 [1,] | . . . . . | . . . . . | |
 [2,] . | . . . . . | . . . . | |
 [3,] . . | . . . . . | . . . | |
 [4,] . . . | . . . . . | . . | |
 [5,] . . . . | . . . . . | . | |
 [6,] . . . . . | . . . . . | | |
 [7,] | . . . . . | . . . . . | |
 [8,] . | . . . . . | . . . . | |
 [9,] . . | . . . . . | . . . | |
[10,] . . . | . . . . . | . . | |
[11,] . . . . | . . . . . | . | |
[12,] . . . . . | . . . . . | | |
[13,] | | | | | | | | | | | | | |
[14,] | | | | | | | | | | | | | |
```

(b) An "off-diagonal" sparsity pattern.

Figure 7: Examples of sparsity patterns for a hierarchical model. The pattern depends on the ordering of the coefficients.

Another option would be to group the coefficients by covariate.

$$\beta_{11}, ..., \beta_{1N}, \beta_{21}, ..., \beta_{2N}, ..., ..., \beta_{k1}, ..., \beta_{kN}, \mu_1, ..., \mu_p \tag{17}$$

Now the Hessian has an "off-diagonal" sparsity pattern, as in Figure 7b.

In both cases, the count and values of the non-zeros in the Hessian are the same. Only the order of the values is different. There are 196 elements in both symmetric matrices, but only 76 values are non-zero, and only 45 of those are unique. Although in this example the reduction in RAM from using a sparse matrix structure for the Hessian may be modest, consider what would happen if $N = 1000$ instead. In that case, there are $2,002$ variables in the problem, and more than 4 million elements in the Hessian, yet only $12,004$ of those elements are non-zero. If we work with only the lower triangle of the Hessian we need to work with only $7,003$ values.

As noted in BD, the sparsity of the Hessian is what makes the method scalable, in terms of the number of heterogeneous units in the data set. Each additional unit adds k rows and columns to the Hessian, so the number of formal elements increases quadratically with N. However, in terms of *non-zero* elements, each unit adds a $k \times k$ block on the diagonal (correlation of variables within a unit), and a block in each margin (correlation between unit-level and population-level variables). Thus, the number of non-zero elements grows linearly, not quadratically, with N. Consequently, the complexity of many of the steps of the algorithm, such as multiplying matrices, generating Cholesky factors, and solving sparse linear systems, grow linearly as well.

The *bayesGDS* package (Braun 2015a) for R includes sample code for estimating this model. Also, the *Matrix* package (Bates and Maechler 2015) defines many different classes and methods for storing and operating on sparse matrices.

3.4 A high-dimensional model of online advertising effectiveness

Braun and Moe (2013) estimate the effectiveness of different versions of display ads in the context of an online advertising campaign. For 5,803 anonymous users, the data set records, by week, which ads (if any) from an advertiser were served to each user during the course of that user's web browsing activity; when these users visited the advertisers own website (if ever); and if these website visits resulted in a "successful" visit. The managerial objective is to identify which versions of ads are most likely to generate site visits and sales, taking into account the fact that the return on investment of the ad may not occur until several weeks in the future. The model allows each version of an ad to have a contemporaneous effect in that week, with each repeat view of the same ad having an incrementally smaller effect. The effect of the ad campaign for an individual builds up with each subsequent ad impression, but this accumulated "ad stock" decays from week to week. However, some of the "wear-out" from repeated exposures can be restored during weeks in which the user is not exposed to a particular ad. Understanding these relationships is useful because if some ad versions are initially ineffective, or become less effective over time, the firms can efficiently allocate marketing resources for other activities, and reduce the "clutter" of ineffective advertising that many users endure.

More formally, define E_{it} as the contemporaneous effect of all of the impressions of ads that were presented to individual i in week t, and define A_{it} as the accumulated Ad Stock, such that $A_{it} = \alpha Ai, t-1 + E_{it}$. Define C_j as the effect of an exposure to ad j on E_{it}, y_{ijt} as the cumulative exposures of ad j, τ_{ijt} as the number of weeks since the last exposure to ad j, δ as a wear-out parameter, and ρ as a restoration parameter. The contemporaneous effect of the ad campaign on user i in week t is the sum of the effects from each version.

$$E_{it} = \sum_j C_j \left[1 - \left(1 - \delta^{y_{ijt}}\right) + \frac{\rho\tau_{ijt}}{1 + \rho\tau_{ijt}}\left(1 - \delta^{y_{ijt}}\right)\right]. \tag{18}$$

The count of i's total exposures to the campaign, and visits to the advertiser's website, in week t, are zero-inflated Poisson random variables, with rates λ_{it} and μ_{it}, respectively. Conditional on v_{it} visits, the number of conversions is a zero-inflated binomial random variable with probability p_{it}. The variation in the latent parameters from week to week depends on ad stock A_{it} and other covariates X_t.

$$\log \lambda_{it} = \log \lambda_{0i} + \gamma_\lambda X_t \tag{19}$$

$$\log \mu_{it} = \log \mu_{0i} + \beta_{\mu i} A_{it} + \gamma_\mu X_t \tag{20}$$

$$\operatorname{logit} p_{it} = \operatorname{logit} p_{0i} + \beta_{pi} A_{it} + \gamma_p X_t \tag{21}$$

The parameters λ_{0i}, μ_{0i}, p_{0i}, $\beta_{\mu i}$ and β_{pi} are all heterogeneous, with a MVN mixing distribution.

Attempts	1	2	3	4	5	6	7	8	9	10+
Samples	1719	146	46	23	15	14	4	4	3	26

Table 3: Number of attempts for each sample from posterior distribution for the advertising effectiveness model. Maximum number of attempts is 296.

These parameters can be correlated, allowing for potential endogeneity between a user's rate of exposure to a advertiser's online campaign, and the baseline propensity to visit the advertiser's website.

Given all of the different latent, nonstationary and heterogeneous effects, the model does not allow for conditionally conjugate posterior distributions. Thus, a Gibbs sampler is not a viable estimation method. However, in spite of the apparent complexity of the model, it is straightforward to break it down into the additive components of the logs of data likelihood, priors and hyperpriors. This decomposition lets us construct functions that compute $\log \mathcal{D}(\theta, y)$ for each component separately, and add them together. One can then draw from the vast existing library of nonlinear optimization algorithms to find the posterior mode, and which is discussed in the next section. This lets us run the BD method, with a multivariate normal proposal distribution, centered at the posterior mode. The covariance matrix is the inverse Hessian at the mode, scaled by 1.02.

Table 3 shows the "draws to acceptance" counts for 2,000 samples from the posterior distribution. In this case, the acceptance rates are quite high, even for a model with nearly 30,000 parameters.

4 Practical considerations and limitations of the BD method

This section details some of our experiences while implementing the method discussed in this paper, and opportunities for future research.

4.1 Implementation notes

4.1.1 Finding the posterior mode

Finding the posterior mode θ^* requires an algorithm for unconstrained nonlinear optimization. Many programming languages that are used for data analysis (e.g, R, Matlab, Python) contain such algorithms, but they can be difficult to use with a large number of decision variables. Search methods like Nelder-Mead (the default algorithm for the `optim` function in R) are inefficient with a massive number of parameters because the search space is large, and they do not exploit information about slope and curvature to speed up the time to convergence. Conjugate gradient, quasi-Newton, and trust region methods use the gradient, and perhaps the Hessian, to accelerate convergence. However, many implementations of these algorithms store the entire dense Hessian, or its inverse, which is overly resource-intensive for large-scale problems.

There are several possible ways to get around this problem. One is to use an algorithm that estimates the curvature of the objective function by storing a smaller, but less exact, approximation. Conjugate gradient and "limited memory" quasi-Newton methods (e.g., L-BFGS) fall into this category, and are also readily available for most statistical software packages. Since these methods do not store the full Hessian, so they can be more suited for large-scale problems. We expect these methods to perform well for log posterior densities with dense Hessians. However, they are not certain to approximate the curvature of the objective function accurately at any particular iteration, especially if the function is not convex.

Another approach, suitable for hierarchical models, is to use an optimization algorithm for which the user supplies the exact Hessian in a compressed sparse format. An example is the *trustOptim* package for R (Braun 2014). Since the number of non-zero elements in the Hessian of a hierarchical model grows only linearly with the number of heterogeneous units, *trustOptim* can scale for large data sets.

4.1.2 Computing derivatives

To use an algorithm that requires derivatives of the objective function, one needs to be able to compute the gradient and the Hessian. For the purposes of the BD algorithm, there are two "good" ways to compute a derivative. The first is to derive it analytically, and write a function to compute it. This approach is straightforward, but it can be tedious and error-prone for complicated models.

The second is to use automatic, or algorithmic, differentiation (AD). In short, AD generates code for the derivative by applying the chain rule on the same sequence of operations that computes the objective function. There are a number of different approaches to implementing AD, and AD libraries are available for many programming languages. However, as of now, there are none for R that are well-suited for a general class of Bayesian hierarchical models. For R users, we believe that coding the objective function in C++ using the *CppAD* library (Bell 2014), and interfacing with R using *Rcpp* (Eddelbuettel and François 2011), is the best option at the moment. What matters is that functions that return the gradient and Hessian of the log posterior density are available, and that they are sufficiently accurate. The advantage of both analytic and AD derivatives is that they are "exact."

We do not recommend estimating the gradient by numerical approximation via finite differencing (FD). FD involves computing $\partial f/\partial x_j \approx \left[f\left(x_j+h\right)-f\left(x_j\right)\right]/h$, or some variation thereof, for each of the $j = 1...J$ variables, using an arbitrarily small h as a "perturbation factor." As $h \to 0$, this estimated difference approaches the gradient. Not only are FD methods are highly vulnerable to numerical precision problems, but the complexity of the method grows with the number of variables Thus, FD is not a reasonable option for estimating the gradient when the number of variables is large. The time to compute the gradient using AD, on the other hand, is only a small fixed multiple of the time to compute the objective function, regardless of the number of variables (Griewank and Walther 2008). Note that some optimization algorithms may not require the user to provide the gradient explicitly, but will default to a finite differenced gradient instead.

The computational cost of computing a dense Hessian using FD is quadratic in the number of variables, and the numerical precision problems are even more pronounced than for a gradient. Nevertheless, one can use FD to estimate the Hessian if the Hessian is sparse, *and* the sparsity pattern is known in advance, *and* the gradient is exact (either derived analytically or computed using AD). The *sparseHessianFD* R package defines methods for doing this (Braun 2017). To use *sparseHessianFD*, the user must provide the row and column indices of the non-zero elements of the lower triangle of the Hessian. For hierarchical models, the sparsity pattern is predictable. The trade-off from using the *sparseHessianFD* package is that the Hessian is still a numerical approximation. We cannot guarantee that this approximation is "good enough" for all cases, and it will almost certainly fail if the gradient itself is estimated using FD. In that case, the estimate of the Hessian would be a finite difference of finite differences, with too much numerical imprecision to be of much value.

4.1.3 High-dimensional MVN distribution

Sampling proposals from an MVN distribution, and computing the MVN density of those draws, requires matrix operations on the Hessian. Again, exploiting the sparsity of the Hessian in hierarchical models helps the BD algorithm scale for a large number of heterogeneous units. For R users, the *sparseMVN* package will be useful (Braun 2015b). The package contains MVN functions that accept *either* the covariance or precision matrix; providing the latter avoids an explicit inverse of the Hessian at the posterior mode. But more importantly, the internal algorithms for multiplying sparse matrices, and solving sparse linear systems, are more scalable than their dense matrix counterparts.

4.2 Cases requiring further research

4.2.1 Multimodal densities

The BD algorithm requires finding the global posterior mode, and so far we have considered only models with unimodal posterior distributions. When the posterior is multimodal, one many need a multimodal proposal. The idea is to not only find the global mode, but any local ones as well, and center each component of, say, a finite mixture of MVN distributions at each local mode. The algorithm itself is unchanged, as long as the global posterior mode matches the global proposal mode. We concede that this strategy requires more research for we can claim it to be viable in practice. For instance, one possible criticism is that finding all of the local modes could be a hard problem. We agree; there is no guarantee that any mode-finding algorithm will find all modes of a posterior distribution. Fortunately, the nonlinear optimization literature is rife with methods that help facilitate efficient location of multiple modes, even if there is no guarantee of finding all of them. Also, even though MCMC sampling chains may, in *theory*, be guaranteed to explore the entire space of any posterior distribution (including multiple regions of high posterior mass), there is no guarantee that this will happen after a large finite number of iterations for general nonconjugate hierarchical models.

The non-MCMC method used in this paper is a viable alternative to MCMC for a large class of Bayesian hierarchical models, but we do *not* claim that it is appropriate for all models.

4.2.2 Models with combinatorial optimization elements

In models that include both discrete and continuous elements, finding the posterior mode becomes a mixed-integer nonlinear program (MINLP). An example is the Bayesian variable selection problem George and McCulloch 1997. The problem is that MINLPs are known to be NP-complete, and thus may not scale well for large problems. Hidden Markov models with multiple discrete states might be similarly difficult to estimate using our method.

4.2.3 Intractable likelihoods or posteriors

There are many popular models, namely ordered and multinomial probit models, for which the likelihood of the observed data is not available in closed form. When direct numerical approximations to these likelihoods (e.g., Monte Carlo integration) is not tractable, data augmentation is a popular tool for estimating these models via MCMC. The Albert and Chib (1993) approach to the binary probit model in Section 3.2 is one example. That said, recent advances in parallelization using graphical processing units (GPUs) might make numerical estimation of integrals over regions that define probabilities more practical than it was even 10 years ago; see Suchard et al. (2010). If this is the case, and the log posterior remains sufficiently smooth, then our approach could be a viable, efficient alternative to data augmentation in these kinds of models. This is especially true if the corresponding MCMC draws are autocorrelated, since our method would require many fewer estimations of these probabilities. Certain types of dynamic structural models (e.g., Iyengar, Ansari, and Gupta (2007)) might also fall into this class of problems.

4.2.4 Missing data problems

MCMC-based approaches to multiple imputation of missing data could suffer from the same kinds of problems as with multinomial probit: the latent parameter, introduced for the data augmentation step, is only weakly identified on its own. Normally, we are not interested in the missing values themselves. If the number of missing data points is small, perhaps we could treat the representation of the missing data points as if they were parameters. But the implications of this require additional research.

4.2.5 Spatial models, and other models with dense Hessians

So far in this paper, the hierarchical models that we have considered assume that the outcomes from heterogeneous units are conditionally independent. Although the BD method is scalable under this assumption, the method does not depend on it. Therefore, BD might still be useful for estimating spatial or contagion models (e.g., Yang and Allenby (2003)). The only difficulty we foresee with these models is when a fast approximation of the Hessian is required.

References

Albert, James H and Siddhartha Chib (1993). "Bayesian Analysis of Binary and Polychotomous Response Data". *Journal of the American Statistical Association* 88.422, pp. 669–679.

Bates, Douglas and Martin Maechler (2015). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package. Version 1.1-5. URL: `http://CRAN.R-project.org/package=Matrix`.

Bell, Bradley M. (2014). *CppAD: A Package for C++ Algorithmic Differentiation*. C++ Library. Computational Infrastructure for Operations Research. URL: `www.coin-or.org/CppAD`.

Braun, Michael (September 2014). "trustOptim: An R Package for Trust Region Optimization with Sparse Hessians". *Journal of Statistical Software* 60.4, pp. 1–16. URL: `http://www.jstatsoft.org/v60/i04/`.

Braun, Michael (2015a). *bayesGDS: an R package for generalized direct sampling*. R package. Version 0.6.1. URL: `http://cran.r-package.org/web/packages/bayesGDS`.

Braun, Michael (2015b). *sparseMVN: an R package for MVN sampling with sparse covariance and precision matrices.* R package. Version 0.2.0. URL: `http://cran.r-project.org/package=sparseMVN`.

Braun, Michael (December 2017). "sparseHessianFD: Estimating Sparse Hessian Matrices in R". *Journal of Statistical Software* 82.10, pp. 1–22.

Braun, Michael and Paul Damien (May 2016–June 2016). "Scalable Rejection Sampling for Bayesian Hierarchical Models". *Marketing Science* 35.3, pp. 427–444.

Braun, Michael and Wendy W. Moe (September 2013). "Online Display Advertising: Modeling the Effects of Multiple Creatives and Individual Impression Histories". *Marketing Science* 32.5, pp. 753–767.

Brooks, Steve, Andrew Gelman, Galin Jones, and Xiao-Li Meng, eds. (2010). *Handbook of Markov Chain Monte Carlo*. Boca Raton, Fla.: Chapman and Hall/CRC.

Chen, Ming-Hui, Qi-Man Shao, and Joseph G Ibrahim (2000). *Monte Carlo Methods in Bayesian Computation*. New York: Springer.

Eddelbuettel, Dirk and Romain François (2011). "Rcpp: Seamless R and C++ Integration". *Journal of Statistical Software* 40.8, pp. 1–18. URL: `http://www.jstatsoft.org/v40/i08`.

Gelfand, Alan E and Adrian F M Smith (June 1990). "Sampling-Based Approaches to Calculating Marginal Densities". *Journal of the American Statistical Association* 85.410, pp. 398–409.

Gelman, Andrew (2006). “Prior Distributions for Variance Parameters in Hierarchical Models”. *Bayesian Analysis* 1.3, pp. 515–533.

Gelman, Andrew, John B Carlin, Hal S Stern, and Donald B Rubin (2003). *Bayesian Data Analysis*. Boca Raton, Fla.: Chapman and Hall.

George, Edward I. and Robert E. McCulloch (1997). “Approaches for Bayesian Variable Selection”. *Statistica Sinica* 7.2, pp. 339–373.

Girolami, Mark and Ben Calderhead (March 2011). “Riemann Manifold Langevin and Hamiltonian Monte Carlo”. *Journal of the Royal Statistical Society, Series B* 73.2, pp. 123–214.

Griewank, Andreas and Andrea Walther (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics.

Iyengar, Raghu, Asim Ansari, and Sunil Gupta (November 2007). “A Model of Consumer Learning for Service Quality and Usage.” *Journal of Marketing Research* 44.4, pp. 529–544.

Liu, Jun S and Chiara Sabatti (2000). “Generalised Gibbs Sampler and Multigrid Monte Carlo for Bayesian Computation”. *Biometrika* 87.2, pp. 353–369.

Martin, Andrew D., Kevin M. Quinn, and Jong Hee Park (2011). “MCMCpack: Markov Chain Monte Carlo in R”. *Journal of Statistical Software* 42.9, pp. 1–21. URL: `http://www.jstatsoft.org/v42/i09/`.

Papaspiliopoulos, Omiros and Gareth Roberts (February 2008). “Stability of the Gibbs Sampler for Bayesian Hierarchical Models”. *The Annals of Statistics* 36.1, pp. 95–117.

Plummer, Martyn, Nicky Best, Kate Cowles, and Karen Vines (2006). “CODA: Convergence Diagnosis and Output Analysis for MCMC”. *R News* 6.1, pp. 7–11. URL: `http://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf`.

Raftery, Adrian E and Steven M Lewis (1996). “Implementing MCMC”. In: *Markov Chain Monte Carlo in Practice*. Ed. by W. R. Gilks, S. Richardson, and David Spiegelhalter. Boca Raton, Fla.: Chapman and Hall/CRC, pp. 115–130.

Suchard, Marc A, Quanli Wang, Cliburn Chan, Jacob Frelinger, Andrew Cron, and Mike West (January 2010). “Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures”. *Journal of Computational and Graphical Statistics* 19.2, pp. 419–438.

Tibbits, Matthew M, Murali Haran, and John C Liechty (April 2010). “Parallel Multivariate Slice Sampling”. *Statistics and Computing* 21.3, pp. 415–430.

Yang, Sha and Greg M Allenby (2003). “Modeling Interdependent Consumer Preferences”. *Journal of Marketing Research* 40.3, pp. 282–294.