Bayesian Analysis (2007) **2**, Number 3, pp. 473–478

Comment on Article by Jain and Neal

David B. Dahl*

1 Introduction

Sonia Jain and Radford Neal (JN) make a significant contribution to the literature on Markov chain Monte Carlo (MCMC) sampling techniques for Dirichlet process mixture (DPM) models. The paper presents some very nice ideas and will be on my required reading list for students working with me. DPM models are widely used for Bayesian nonparametric analyses and efficient sampling techniques are essential for their routine application. Incremental samplers for nonconjugate DPM models, such as the Auxiliary Gibbs sampler in Neal (2000), are easily implemented and potentially very efficient. Unfortunately, these samplers can also have difficulty mixing over the entire sample space and standard MCMC diagnostics may fail to indicate the problem. JN's paper represents a significant advance by providing a non-incremental sampler for conditionally conjugate DPM models.

The authors have a history of influential papers in this area, including Neal (2000) and Jain and Neal (2004). Their 2004 paper provided a split-merge sampler for conjugate DPM models, where the base distribution G_0 in the Dirichlet process prior is conjugate to the likelihood F. By exploiting this conjugacy, the model parameters of a cluster can be integrated away. The state of the Markov chain is merely the clustering of observations. Thus, sampling algorithms for conjugate DPM model attempt to sample from the posterior clustering distribution.

In nonconjugate DPM models, the model parameters of a cluster cannot be integrated away. Sampling algorithms must simultaneously address the clustering and the model parameters associated with each cluster. Green and Richardson (2001) were the first to propose a split-merge sampler for nonconjugate DPM models. Their procedure is based on reversible jump MCMC (Green 1995; Richardson and Green 1997) where the Metropolis-Hastings proposals are model-specific. In this paper, JN provide an MCMC sampler that can be generically applied to any conditionally conjugate DPM model.

2 Conditional Conjugate vs. Nonconjugate

It is important to note that conditional conjugacy is a necessary prerequisite for the application of JN's sampler. Suppose the model parameters for the cluster containing observation i are $\phi_1, \ldots, \phi_H$ with likelihood $F(y_i|\phi_1, \ldots, \phi_H)$ and prior $G_0(\phi_1, \ldots, \phi_H)$. A DPM model is conditionally conjugate if, for each $\phi_h \in \{\phi_1, \ldots, \phi_H\}$, $G_0(\phi_1, \ldots, \phi_H)$ is conjugate to $F(y_i|\phi_1, \ldots, \phi_H)$ in ϕ_h. JN's procedure relies on conditional conjugacy

*Department of Statistics, Texas A&M University, College Station, TX, http://www.stat.tamu.edu/~dahl

and hence their procedure is not applicable to all nonconjugate DPM model. Whether the conditional conjugacy constraint imposes a practical limitation is perhaps problem-specific.

3 Cluster Labels, Set Partition, and Implementation

JN describe their algorithm using notation involving cluster labels $c_1, \ldots, c_n$. An alternative way of describing sampling algorithms for DPM models uses set partition notation. In my experience, the set partition notation provides a straightforward presentation with simple notation. A set partition $\pi = \{S_1, \ldots, S_q\}$ of $S_0 = \{1, \ldots, n\}$ divides the n integers into mutually-exclusive, non-empty, and exhaustive clusters $S_1, \ldots, S_q$. I especially find the set partition notation helpful when translating sampling algorithms for DPM models to computer code. I use an array of length n whose elements point to C++ classes representing clusters containing the model parameters and a set of integers (for the cluster membership).

Regardless of notational preference, readers should be assured that the actual implementation of JN's sampler need not be complex. The core of my implementation of JN's split-merge procedure is 158 lines of C++ code, whereas the core of my implementation of the Auxiliary Gibbs sampler is 53 lines of C++ code. The extra time and mental effort needed to implement their split-merge sampler can pay large dividends for models and datasets where the Auxiliary Gibbs sampler is likely to have problems.

4 Initial States & Benefits of Split/Merge Samplers

I applied JN's split-merge algorithm to their Normal-Gamma mixture model and the beetle data used in their example. In the top two plots of JN's Figure 1, they use their vague priors with hyperparameters $w_j = (100, 100, 50, 100, 25, 100)$, $B_j^{-1} = (500, 100, 25, 100, 25, 150)$, and $r = R = 1$ across all six dimensions. The top left plot corresponds to Auxiliary Gibbs sampling and shows that this sampler never moves away from the configuration with all observations in one cluster. JN contrast that with their Split-Merge (5,1,1,5) sampler (shown in the top right plot of JN's Figure 1) which is able to readily find the true three-component structure in the data.

In my implementation with 100 different random number seeds, the Auxiliary Gibbs sampler was able to find the three-component structure in 98 instances and a two-component structure (hinted at in the bottom left plot of JN's Figure 1) in the remaining two instances. Why could my implementation of the algorithm find the true structure, but their implementation of the same algorithm could not? The issue was the initial values of the model parameters. I sampled the initial value of the model parameters from the prior G_0. If, instead, I set the initial values of the model parameters to the sample means and precisions, I am able to replicate the results of JN in 100 of 100 instances. Also, if each observation is initially placed in its own cluster, the Auxiliary Gibbs sampler performed well (regardless of the method used to set the model parameters). My Figure 1 summarizes the results, showing that the problem with the Auxiliary Gibbs sampler

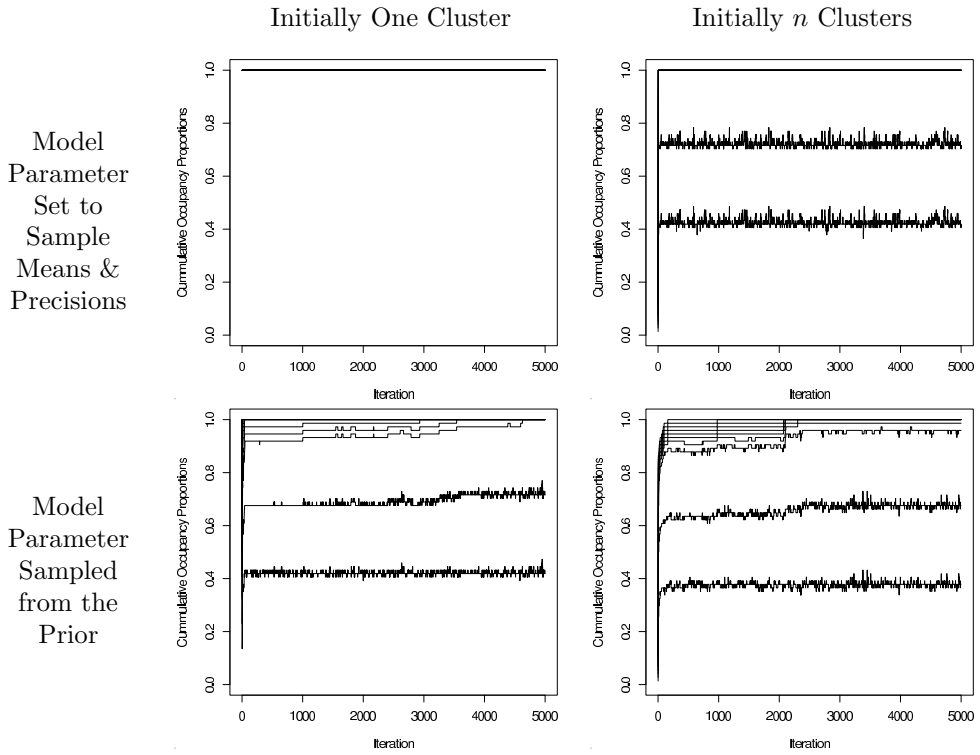

Figure 1: Trace plots from the Auxiliary Gibbs Sampler for the beetle data using JN's vague priors and four typical initial states for the Markov chain. The poor performance of the Auxiliary Gibbs sampler (shown in top left plot of JN's Figure 1) is only present when every observation is initially clustered together and the model parameters are initially set to the sample means and precisions.

is only present when every observation is initially clustered together and the model parameters are initially set to the sample means and precisions.

In my experience replicating the JN's Figure 1, their split-merge sampler is not sensitive to the initial values. For the beetle data and their Normal-Gamma mixture model, their sampler immediately finds the true three-component structure as shown in the plots on the right in JN's Figure 1.

It is interesting to observe that the posterior distribution apparently has virtually no support for anything other than three components. Notice that the Split-Merge (5,1,1,5) sampler never moves away from three clusters (to a configuration with two or four clusters, for example). The jitter present in the right hand size of JN's Figure 1 is due purely to the fact that their split-merge sampler embeds the Auxiliary Gibbs sampler (whose number of scans per split-merge attempt is given as the the third argument in

	Jain & Neal		*Dahl*	
Algorithm	*Example 1*	*Example 2*	*Example 1*	*Example 2*
Gibbs Sampling $v = 3$	1.00	1.00	1.00 (0.80, 1.13)	1.00 (0.84, 1.12)
Split-Merge (0,1,0,0)	0.11	0.17	0.17 (0.17, 0.18)	0.18 (0.17, 0.19)
Split-Merge (0,1,1,0)	0.60	0.58	0.61 (0.60, 0.62)	0.61 (0.60, 0.63)
Split-Merge (5,1,0,5)	0.36	0.40	0.84 (0.79, 0.88)	0.86 (0.82, 0.89)
Split-Merge (5,1,1,5)	0.89	0.88	1.32 (1.30, 1.35)	1.34 (1.32, 1.37)
Seconds per Iteration	0.45	0.60	5.50×10^{-4}	6.75×10^{-4}

Table 1: Comparison of relative CPU time of the various samplers depending on the dataset and the implementation. Jain & Neal columns are derived from Table 3. The Dahl columns show averages from 100 replications and the 2.5^{th} and 97.5^{th} quantiles. The data have been standardized by the "Seconds per Iteration" row to make them comparable across computers and programming languages.

the quad specifying the details of their sampler). Thus, the CPU time spent on trying to merge and split is wasted and time would be better spent on just the Auxiliary Gibbs sampler. The same can be said concerning the first simulated dataset in JN's Figure 4. In contrast, Example 2 (shown in JN's Figure 5) does provide a compelling case for the split-merge sampler. It freely moves between four and five components, whereas the Auxiliary Gibbs sampler is unlikely to easily switch between four and five components.

5 Timing

My final point concerns inherent variability in the implementation of algorithms due to the chosen programming language and data structures. JN have two simulated datasets (labeled Example 1 and Example 2) which they use to compare the various samplers. My Table 1 compares the CPU time of my C++ implementation of the various algorithms with that of JN's Matlab implementation. The first two columns are taken from JN's Table 3. The Dahl columns show averages from 100 replications and the 2.5^{th} and 97.5^{th} quantiles. The important point is the relative performance of the various sampling algorithms (not the speeds of different computers or programming languages), so the data has been scaled by the "Seconds per Iteration" row. Specifically, the Auxiliary Gibbs sampler with three auxiliary parameters (labeled as "Gibbs sampling $v = 3$") is set at 1.0 within each column.

Notice that relative CPU time taken by each of the samplers, within an implementation, is relatively constant across the two example datasets. There are, however, very different relative CPU times across implementations within a dataset. Recall that the Split-Merge(5,1,1,5) sampler embeds one Auxiliary Gibbs update with one auxiliary parameter per split-merge attempt. The Split-Merge(5,1,0,5) does not have any embedded Auxiliary Gibbs updates, leading to a $1 - 0.36/0.89 = 60\%$ reduction in the CPU time per iteration for JN's implementation of Example 1. In contrast, my implementation of Split-Merge(5,1,0,5) provides only $1 - 0.84/1.32 = 36\%$ reduction from my Split-Merge(5,1,1,5).

JN (2007) compare an Auxiliary Gibbs sampler with three auxiliary parameters with their Split-Merge(5,1,1,5) sampler which embeds an Auxiliary Gibbs sampler with one auxiliary parameter. They chose three and one auxiliary parameters respectively to make the CPU times comparable per iteration and then run each sampler for a fixed number of iterations. In my experience, additional auxiliary parameters are often not worth the extra CPU effort. For the sake of comparison, it might be more useful to have the number of auxiliary parameters be the same for both samplers. Comparisons would then be based on a fixed CPU time rather than a fixed number of iterations.

6 Conclusion

JN have made a significant contribution to the literature on sampling algorithms for DPM models. In implementing their algorithm and model and in using their example datasets, I found their method can have substantial benefits over the Auxiliary Gibbs sampler when used to sample from the posterior distribution of conditionally conjugate DPM models. Their algorithm is certainly more complicated than the Auxiliary Gibbs sampler, but perhaps not as difficult as one might initially expect. My experience with JN's Figure 1 reinforced the importance of using a variety of starting states, particularly when using the Auxiliary Gibbs sampler. It was nice to see that initial starting values were not an issue for JN's split-merge sampler. Although the relative CPU timings of JN's implementation and mine can be quite different, the salient point is that split-merge samplers can finding high-probability regions in posterior distributions that may be missed by incremental samplers.

References

Green, P. J. (1995). "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination." *Biometrika*, 82: 711–732. 473

Green, P. J. and Richardson, S. (2001). "Modelling heterogeneity with and without the Dirichlet process." *Scandinavian Journal of Statistics*, 28: 355–375. 473

Jain, S. and Neal, R. M. (2004). "A Split-Merge Markov Chain Monte Carlo Procedure for the Dirichlet Process Mixture Model." *Journal of Computational and Graphical Statistics*, 13(1): 158–182. 473

Neal, R. M. (2000). "Markov Chain Sampling Methods for Dirichlet Process Mixture Models." *Journal of Computational and Graphical Statistics*, 9: 249–265. 473

Richardson, S. and Green, P. J. (1997). "On Bayesian Analysis of Mixtures With An Unknown Number of Components (Disc: P758-792) (Corr: 1998V60 P661)." *Journal of the Royal Statistical Society, Series B, Methodological*, 59: 731–758. 473