Technical Report No. 2003, Department of Statistics, University of Toronto

A Split-Merge Markov Chain Monte Carlo Procedure for the Dirichlet Process Mixture Model

Sonia Jain *　　　Radford M. Neal †

July 27, 2000

Abstract. We propose a split-merge Markov chain algorithm to address the problem of inefficient sampling for conjugate Dirichlet process mixture models. Traditional Markov chain Monte Carlo methods for Bayesian mixture models, such as Gibbs sampling, can become trapped in isolated modes corresponding to an inappropriate clustering of data points. This article describes a Metropolis-Hastings procedure that can escape such local modes by splitting or merging mixture components. Our Metropolis-Hastings algorithm employs a new technique in which an appropriate proposal for splitting or merging components is obtained by using a restricted Gibbs sampling scan. We demonstrate empirically that our method outperforms the Gibbs sampler in situations where two or more components are similar in structure.

Key words: Dirichlet process mixture model, Markov chain Monte Carlo, Metropolis-Hastings algorithm, Gibbs sampler, split-merge updates

1 Introduction

Mixture models are often applied to density estimation, latent class analysis, and classification problems, as discussed, for example, by Everitt and Hand (1981), McLachlan and Basford (1988), and Titterington *et al* (1985). The Bayesian approach to mixture models has recently generated interest due to advances in statistical computation, in particular Markov chain Monte Carlo (see Tierney 1994 and Gilks *et al* 1996). In this article, we consider Bayesian mixture models in which a Dirichlet process prior on the mixing distribution is used to handle a countably infinite number of mixture components. Some earlier references on computational techniques for Dirichlet process mixture models include Escobar (1994), Escobar and West (1995), MacEachern (1994), Bush and MacEachern (1996), Neal (1992), Richardson and Green (1997), and Neal (2000).

Gibbs sampling is a common Markov chain Monte Carlo scheme for sampling the posterior distribution of the Dirichlet process mixture model. When conjugate priors are used, the Gibbs sampling procedure is easily constructed. We fully exploit the conjugacy in the model by analytically inte-

*Department of Statistics, University of Toronto, Toronto, Ontario, Canada M5S 3G3. Email: `sonia@utstat.toronto.edu` ; Internet: `http://www.utstat.toronto.edu/~sonia/`

†Department of Statistics and Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G3. Email: `radford@utstat.toronto.edu` ; Internet: `http://www.cs.toronto.edu/~radford/`

grating away the mixing proportions for the components and the parameters for each component. As a result, the Gibbs sampling procedure only updates the latent indicator variables associating mixture components with data observations. This particular Gibbs sampling method was first discussed by Neal (1992) for models of categorical data and MacEachern (1994) for normal mixture models.

Although the Gibbs sampling approach is straightforward and easily implemented, it can be slow to converge and mix poorly. When two or more mixture components have similar parameters, the Gibbs sampling method may become trapped in a local mode that corresponds to an incorrect clustering of data points. Celeux *et al* (2000) attribute this mixing problem to the incremental nature of the Gibbs sampler, which is unable to simultaneously move a group of observations to a new mixture component. Incremental updates are unlikely to move a single observation to a new mixture component because such an intermediate state has low probability. A sampling scheme which allows a group of observations to be updated simultaneously may remedy this problem, since neighbouring observations would support the formation of a new component if appropriate.

Split-merge and birth-death updates have previously been proposed by Phillips and Smith (1996) and Green and Richardson (1999). Phillips and Smith (1996) consider the birth and death of mixture components using a jump-diffusion sampling algorithm. In the mixture model context, they wish to estimate the number of components and approach this as a model comparison problem for models of varying dimensions. Their algorithm generates a Markov chain by a local "jump" step in which discrete transitions are made between mixture models differing by one component and a "diffusion" step in which model parameters are updated between the jumps. In their one-dimensional example, the parameters of a possible new component are drawn from the prior. When proposing to create or delete a component, the weights of adjacent components are modified appropriately. This method does not generalize easily to higher dimensional problems.

Green and Richardson (1999) introduce a complex split-merge scheme in the reversible jump framework (Green 1995). Changes to the parameter space are proposed by increasing or decreasing the number of mixture components by one in a single iteration. The split-merge proposals are based on conserving specific moment conditions and are evaluated by a Metropolis acceptance probability. Their method requires the inclusion of tuning parameters so that asymmetrical splits may be proposed and to adjust for unequal sample sizes when merging. Depending on the type of statistical mixture model, there are numerous ways to propose an appropriate split proposal. Green and Richardson provide general guidelines in constructing split proposals that satisfy various moment conditions. It is not clear whether adequate split proposals are simple to construct or compute in high-dimensional multivariate mixture problems using their technique.

This article introduces a new Metropolis-Hastings method that avoids the problems associated with the Gibbs sampling procedure and is suitable for high-dimensional data. Typically, Metropolis-Hastings updates involve simple parametric distributions as the proposal distribution. To split mixture components, our method employs a more complex proposal distribution obtained by using a restricted Gibbs sampling scan for the latent class variables. This method is able to quickly traverse the state space and frequently visit high-probability modes because it splits or merges a group of observations in each update, thereby bypassing the incremental updates of the Gibbs sampler. Furthermore, although the proposal distribution used is complex, it does not need to be specially tailored to each model, since the same scheme can be applied to any model with a conjugate prior.

In Section 2, notation and terminology for the Dirichlet process mixture model and a Gibbs sampling algorithm suitable for conjugate priors are introduced. Section 3 presents our split-merge Metropolis-Hastings procedure and its variants. In Section 4, we empirically compare our split-merge method to the Gibbs sampler and demonstrate its improved performance for a categorical data problem. We conclude, in Section 5, by discussing possible extensions of the split-merge algorithm and the general applicability of our new Metropolis-Hastings technique.

2 A Gibbs sampling procedure for the Dirichlet process mixture model

In this section, we present the Dirichlet process mixture model (for early references, see Ferguson 1973 and Antoniak 1974) and describe a Gibbs sampling algorithm to sample from the posterior of this model. This procedure completely utilizes the conjugacy in the model to integrate away model parameters and mixing proportions, eliminating them from the algorithm. In Section 4, this version of the Gibbs sampler is compared to the new split-merge Metropolis-Hastings algorithm.

2.1 The Dirichlet process mixture model

We consider a hierarchical mixture model in which the observations, $y_1, \ldots, y_n$, are modelled as a mixture of distributions having the form $F(\theta)$. There is no restriction on the dimensionality of the y_i, and the data may be categorical or quantitative. For each observation y_i, the model parameters, θ_i , are considered to be independent draws from some mixing distribution, G. Rather than requiring G to take some parametric form, a Dirichlet process prior, a distribution over the space of distribution functions, is placed on G. This yields a mixture model of the following form:

$$\begin{aligned} y_i \mid \theta_i &\sim F(\theta_i) \\ \theta_i \mid G &\sim G \\ G &\sim DP(G_0, \alpha) \end{aligned} \tag{1}$$

where G_0 and α are the two Dirichlet process parameters. G_0 defines a baseline distribution for the Dirichlet process prior, while α is a total mass parameter that takes values greater than zero. Equation (1) represents the most basic Dirichlet process mixture model. Further stages may be added to this hierarchy by placing priors on α and the parameters of G_0 (for example, see MacEachern 1998). The usual conditional independence assumptions for a hierarchical model apply, so that the only dependencies are those that are explicitly shown.

This model may be regarded as a countably infinite mixture model (Ferguson 1983), a view that is adopted in the remainder of this article. When G is integrated over its prior distribution in equation (1), we see that the θ_i follow a generalized Polya urn scheme (for details, refer to Blackwell and MacQueen 1973 and Ferguson 1973). The prior distribution for the θ_i may be represented by the following conditional distributions:

$$\begin{aligned} \theta_1 &\sim G_0 \\ \theta_i \mid \theta_1, \ldots, \theta_{i-1} &\sim \frac{1}{i-1+\alpha} \sum_{j=1}^{i-1} \delta(\theta_j) \;+\; \frac{\alpha}{i-1+\alpha} G_0 \end{aligned} \tag{2}$$

where $\delta(\theta_j)$ is the distribution which is a point mass at θ_j. The model of equation (1) has been simplified by integrating away the random distribution, G. We can represent the fact that (2) results in some of the θ_i being identical by setting $\theta_i = \phi_{c_i}$, where c_i represents the "latent class" associated with observation i. The Polya urn scheme for sampling the θ_i is equivalent to the following scheme for sampling the latent variables, c_i, and associated ϕ_c:

$$\begin{aligned} P(c_i = c_j \text{ for some } j < i \mid c_1, \ldots, c_{i-1}) &= \frac{n_{i,c_j}}{i-1+\alpha} \\ P(c_i \neq c_j \text{ for all } j < i \mid c_1, \ldots, c_{i-1}) &= \frac{\alpha}{i-1+\alpha} \end{aligned} \tag{3}$$

where $n_{i,c}$ is the number of c_k for $k < i$ that are equal to c. The labelling of the indicator c_i is irrelevant in the above probabilities, since counts of the c_i that are equal to each other are all that matter. The ϕ_c are drawn independently from the initial distribution G_0. The probabilities shown in (3) define the Dirichlet process model and are equivalent to the mixture model in equation (1). This notation will be employed in subsequent sections.

2.2 Gibbs sampling for the conjugate Dirichlet process mixture model

For this model, if G_0 is a conjugate prior for F, it is straightforward to sample from the posterior distribution using Gibbs sampling. There have been several Gibbs sampling approaches proposed in the Dirichlet process mixture model literature, but we consider the procedure in which conjugacy is fully exploited, which was first introduced by Neal (1992) and MacEachern (1994). This procedure integrates away the model parameters, ϕ_{c_i}. Eliminating ϕ_{c_i} simplifies the algorithm considerably, so that the state of the Markov chain for the Gibbs sampler now consists only of the indicators c_i.

The Markov chain is initialized by setting the c_i to some initial state, for example, all the c_i set to the same component or all to different components. The c_i are then updated via Gibbs sampling by repeatedly drawing a new value for each c_i from its conditional distribution, which is proportional to the product of its conditional prior and likelihood. Because the observations are exchangeable, the conditional prior is derived from equation (3) by considering observation i to be the last of the n observations. This yields the following conditional probabilities:

$$\begin{aligned} P(c_i = c_j \text{ for some } j \neq i \mid c_{-i}, y_i) &= b \, \frac{n_{-i,c_j}}{n-1+\alpha} \int F(y_i, \phi) \, dH_{-i,c_j}(\phi) \\ P(c_i \neq c_j \text{ for all } j \neq i \mid c_{-i}, y_i) &= b \, \frac{\alpha}{n-1+\alpha} \int F(y_i, \phi) \, dG_0(\phi) \end{aligned} \tag{4}$$

where c_{-i} represents the c_j for $j \neq i$, $n_{-i,c}$ is the number of c_j for $j \neq i$ that are equal to c, n is the number of observations, $H_{-i,c}$ is the posterior distribution of ϕ based on the prior G_0 and all data observations y_j for which $j \neq i$ and $c_j = c$, $F(y_i, \phi)$ is the likelihood, and b is the appropriate normalizing constant so that the probabilities sum to one. When G_0 is a conjugate prior for F, the integrals $\int F(y_i, \phi)\, dG_0(\phi)$ and $\int F(y_i, \phi)\, dH_{-i,c}(\phi)$ may be analytically computed.

MacEachern (1994) demonstrated the application of this Gibbs sampling algorithm for a conjugate normal mixture model with fixed variance. In this case, F is the normal distribution $N(\theta_i, \sigma^2)$, and G_0 is the normal distribution $N(\mu, \tau^2)$ which is conjugate to F. Similarly, Neal (1992) illustrated that this algorithm could be applied to categorical data. In his example, the data observations are dichotomous, so F is a product of independent Bernoulli distributions, and G_0 is a product of independent Beta prior distributions. In Section 4, we consider a similar categorical data example to establish that the algorithm we present below is an improvement over Gibbs sampling.

3 A split-merge Metropolis-Hastings algorithm for the conjugate Dirichlet process mixture model

When two or more mixture components have similar parameters, Gibbs sampling can be inefficient. The Markov chain can become trapped in a local mode, in which two distinct mixture components are merged and assigned parameters which are a compromise between the two separate components. Because the Gibbs sampler incrementally updates each observation, the Markov chain must pass through a low-probability intermediate state in order to split such a component. This leads to slow convergence to the true posterior distribution. Here, we introduce a nonincremental Markov chain sampling method based on the Metropolis-Hastings algorithm that avoids this problem. Our algorithm is able to split or merge groups of data points, avoiding the need to pass through a low-probability intermediate state in order to make major changes.

We begin by reviewing Metropolis-Hastings updates. We then discuss two possible proposal distributions for Metropolis-Hastings updates for the Dirichlet process mixture model, based on a simple random split and on a more complex restricted Gibbs sampling scan. Even though these algorithms are ergodic, performance may be further improved by combining the split-merge updates with a regular Gibbs sampling scan.

3.1 Metropolis-Hastings updates

Our algorithm, in which we propose split and merge moves that nonincrementally update groups of observations, is a form of the Metropolis-Hastings algorithm (Metropolis *et al* 1953, Hastings 1970). The Metropolis-Hastings algorithm samples from a distribution with density $\pi(x)$ by first drawing a candidate state, x^*, according to a proposal density $q(x^*|x)$. This proposed state, x^*, is evaluated by the Metropolis-Hastings acceptance probability which is calculated as follows:

$$a(x^*, x) \quad = \quad \min\left[1, \ \frac{q(x|x^*)}{q(x^*|x)} \frac{\pi(x^*)}{\pi(x)}\right] \tag{5}$$

The next state will be set to this candidate state with probability $a(x^*, x)$. Otherwise, the new state remains as the current state, x. These Metropolis-Hastings updates leave the posterior distribution, π, invariant and produce a valid Markov chain Monte Carlo sampling scheme provided the chain is ergodic.

As discussed by Tierney (1994), when constructing Markov chains, it is acceptable to select a transition probability at random from a set of appropriate transition probabilities. In particular, we may randomly choose amongst valid Metropolis-Hastings algorithms by randomly selecting a proposal distribution, $q(x^*|x)$. Note that when calculating the Metropolis-Hastings acceptance probability, the ratio, $\frac{q(x|x^*)}{q(x^*|x)}$, may be calculated for the particular proposal distribution that was chosen rather than computing the ratio by summing over all possible proposal distributions. Both lead to valid Metropolis-Hastings updates, but the latter calculation may be computationally infeasible.

3.2 Random split-merge proposals

First, we introduce the split-merge algorithm when the proposal distribution is based on a simple random split of the subset of observations associated with one mixture component into two separate components, without reference to the properties of the observed data. This is the simplest version of the split-merge algorithm, which we do not expect to work well, but which illustrates the basic construction. A more elaborate version of this procedure, based on a restricted Gibbs sampling proposal, produces more sensible splits and is discussed in Section 3.3.

This split-merge approach will be applied to the conjugate Dirichlet process mixture model, in which the random distribution, G, and the model parameters, ϕ_{c_i}, are integrated away. The state of the Markov chain consists only of the mixture component indicators, c_i. The Markov chain is initialized by assigning each observation to a mixture component. Typical initial states we have used are placing all the data in the same component and assigning each observation to a different component. Below, we outline the steps for the simple random split-merge procedure.

3.2.1 Algorithm 1: Simple random split-merge procedure

1. Select two distinct observations, i and j, at random uniformly. The random selection of these two items will decide the particular Metropolis-Hastings proposal distribution considered.

2. Let S denote the set of observations, $k \in \{1, \ldots, n\}$, for which $k \neq i$ and $k \neq j$, and $c_k = c_i$ or $c_k = c_j$.

3. If items i and j belong to the same mixture component, i.e. $c_i = c_j$, then:

(a) Propose a new assignment of data items to mixture components, denoted as $\boldsymbol{c}^{split}$, in which component $c_i = c_j$ is split into two separate components, c_i^{split} and c_j^{split}. We define each element of the proposal vector, $\boldsymbol{c}^{split}$, as follows:
   - Let c_i^{split} be a new component such that $c_i^{split} \notin \{c_1, \ldots, c_n\}$
   - Let $c_j^{split} = c_j$
   - For every observation $k \in S$, let c_k^{split} be randomly set, independently with equal probability, to either component c_i^{split} or c_j^{split}
   - For observations $k \notin S \cup \{i, j\}$, let $c_k^{split} = c_k$

(b) Evaluate the proposal in (a) by the Metropolis-Hastings acceptance probability $a(\boldsymbol{c}^{split}, \boldsymbol{c})$ discussed below. If the proposal is accepted, $\boldsymbol{c}^{split}$ becomes the next state in the Markov chain. If the proposal is rejected, the original vector, $\boldsymbol{c}$, remains as the next state.

4. Otherwise, if i and j belong to different mixture components, i.e. $c_i \neq c_j$, then:

   (a) Propose a new assignment of data items to mixture components, denoted as $\boldsymbol{c}^{merge}$, in which components, c_i and c_j, are combined into a single component. Each element of the proposal vector, $\boldsymbol{c}^{merge}$, is assigned as follows:
   - Let $c_i^{merge} = c_j$
   - Let $c_j^{merge} = c_j$
   - For every observation $k \in S$, let $c_k^{merge} = c_j$
   - For observations $k \notin S \cup \{i, j\}$, let $c_k^{merge} = c_k$

   (b) Evaluate the proposal in (a) by the Metropolis-Hastings acceptance probability $a(\boldsymbol{c}^{merge}, \boldsymbol{c})$. If the proposal is accepted, $\boldsymbol{c}^{merge}$ becomes the next state. If the merge proposal is rejected, the original configuration, $\boldsymbol{c}$, remains as the next state.

5. Repeat steps 1-4 for T iterations.

Note that because the numerical values of the c_k are irrelevant, it does not matter in steps 3(a) and 4(a) which item, i or j, remains fixed at its original mixture component. The labels are significant only in that they distinguish which items are grouped in the same mixture component. Also note that the vectors $\boldsymbol{c}^{split}$, $\boldsymbol{c}^{merge}$, and $\boldsymbol{c}$ designate which mixture component is assigned to each observation in the data — not just to observations that are involved in the split or merge steps. However, items not associated with i, j, or set S remain unchanged and unaffected during the Metropolis-Hastings update.

3.2.2 Metropolis-Hastings acceptance probability for the simple random split algorithm

When updating the vector $\boldsymbol{c}$ associating observations with mixture components, the Metropolis-Hastings acceptance probability of equation (5) takes the following form:

$$a(\boldsymbol{c}^*, \boldsymbol{c}) \quad = \quad \min\left[1, \; \frac{q(\boldsymbol{c}|\boldsymbol{c}^*)}{q(\boldsymbol{c}^*|\boldsymbol{c})} \frac{P(\boldsymbol{c}^*)}{P(\boldsymbol{c})} \frac{L(\boldsymbol{c}^*|\boldsymbol{y})}{L(\boldsymbol{c}|\boldsymbol{y})}\right] \tag{6}$$

where $\boldsymbol{c}^*$ is either the vector $\boldsymbol{c}^{split}$ or $\boldsymbol{c}^{merge}$ depending on the type of proposal. The posterior distribution, $\pi(\boldsymbol{c})$, in equation (5) has been expanded into a product of its factors: the prior, $P(\boldsymbol{c})$, and the likelihood, $L(\boldsymbol{c}|\boldsymbol{y})$, where $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the vector of observations. Note that factors not involving $\boldsymbol{c}$ may be ignored.

The prior distribution, $P(\boldsymbol{c})$, for the entire vector $\boldsymbol{c}$ will be a product over distinct $c \in \{c_1, \ldots, c_n\}$ of factors presented in equation (3). This product yields the following prior distribution:

$$P(\boldsymbol{c}) \quad = \quad \alpha^D \frac{\prod_{c\in\{c_1,\ldots,c_n\}}(n_c-1)!}{\prod_{k=1}^{n}(\alpha+k-1)} \tag{7}$$

where D is the number of distinct mixture components contained in vector $\boldsymbol{c}$ and n_c is the count of items belonging to mixture component c in $\boldsymbol{c}$.

Notice that the ratio of the prior distributions in equation (6) simplifies considerably because the denominator in equation (7) will cancel, as well as factors in equation (7) associated with components that are not directly involved in the Metropolis-Hastings update. For the split proposal, the prior distribution ratio reduces to the following:

$$\frac{P(\boldsymbol{c}^{split})}{P(\boldsymbol{c})} \quad = \quad \alpha \frac{(n_{c_i^{split}}-1)!\,(n_{c_j^{split}}-1)!}{(n_{c_i}-1)!} \tag{8}$$

where $\boldsymbol{c}$ represents the original state in which i and j belong to the same mixture component. Here, $n_{c_i^{split}}$ and $n_{c_j^{split}}$ represent the number of observations that belong to the two split mixture components. The factor of α in the numerator arises from D being one greater in $\boldsymbol{c}^{split}$ than in $\boldsymbol{c}$.

Similarly, for the merge proposal, the prior ratio simplifies to:

$$\frac{P(\boldsymbol{c}^{merge})}{P(\boldsymbol{c})} \quad = \quad \frac{1}{\alpha} \frac{(n_{c_i^{merge}}-1)!}{(n_{c_i}-1)!\,(n_{c_j}-1)!} \tag{9}$$

where $\boldsymbol{c}$ represents the original state in which items i and j belong to separate components.

The likelihood for the vector of component indicators will be a product over the n observations:

$$L(\boldsymbol{c}|\boldsymbol{y}) \quad = \quad \prod_{k=1}^{n} \int F(y_k, \phi)\, dH_{k,c_k}(\phi) \tag{10}$$

where H_{k,c_k} is the posterior distribution of ϕ based on the prior G_0 and all observations y_l for which $l < k$ and $c_l = c_k$. We assume that the integral $\int F(y_k, \phi)\, dH_{k,c_k}(\phi)$ is analytically tractable, which is the case if G_0 is a conjugate prior. Note that when k is the first item observed from a particular component, then $H_{k,c}$ will be the prior distribution, G_0, since no data from that mixture component precedes item k.

Alternatively, $L(\boldsymbol{c}|\boldsymbol{y})$ may be expressed as a double product over components, c, and items, $k \in \{1, \ldots, n\}$, associated with each component:

$$L(\boldsymbol{c}|\boldsymbol{y}) \quad = \quad \prod_{c=1}^{D} \prod_{k\,:\,c_k=c} \int F(y_k, \phi)\, dH_{k,c}(\phi) \tag{11}$$

where D is the number of distinct components.

Since factors involving items associated with components not directly involved in the split proposal will cancel, the ratio of likelihoods in equation (6) reduces to the following:

$$\frac{L(\boldsymbol{c}^{split}|\boldsymbol{y})}{L(\boldsymbol{c}|\boldsymbol{y})} = \frac{\prod_{k\,:\,c_k^{split}=c_i^{split}} \int F(y_k,\phi)\, dH_{k,c_i^{split}}(\phi) \prod_{k\,:\,c_k^{split}=c_j^{split}} \int F(y_k,\phi)\, dH_{k,c_j^{split}}(\phi)}{\prod_{k\,:\,c_k=c_i} \int F(y_k,\phi)\, dH_{k,c_i}(\phi)} \tag{12}$$

Similarly, for the merge proposal, the ratio of likelihoods is:

$$\frac{L(\boldsymbol{c}^{merge}|\boldsymbol{y})}{L(\boldsymbol{c}|\boldsymbol{y})} = \frac{\prod_{k\,:\,c_k^{merge}=c_i^{merge}} \int F(y_k,\phi)\, dH_{k,c_i^{merge}}(\phi)}{\prod_{k\,:\,c_k=c_i} \int F(y_k,\phi)\, dH_{k,c_i}(\phi) \prod_{k\,:\,c_k=c_j} \int F(y_k,\phi)\, dH_{k,c_j}(\phi)} \tag{13}$$

The random split proposal distribution is the simplest version of this algorithm. The selection of observations, i and j, decides which Metropolis-Hastings proposal will be used. As a result, when calculating the acceptance probability, i and j are fixed. The probability of proposing a particular split of the items in set S from the merged state is:

$$q(\boldsymbol{c}^{split}|\boldsymbol{c}) = \left(\frac{1}{2}\right)^{n_{c_i^{split}}+n_{c_j^{split}}-2} \tag{14}$$

Notice that $q(\boldsymbol{c}^{split}|\boldsymbol{c})$ is equivalent to $q(\boldsymbol{c}|\boldsymbol{c}^{merge})$.

The probability of proposing a merge move for the items in S from a split state is:

$$q(\boldsymbol{c}^{merge}|\boldsymbol{c}) = 1 \tag{15}$$

since there is only one way to assign all items in S to the same component. Note that $q(\boldsymbol{c}^{merge}|\boldsymbol{c})$ is equivalent to $q(\boldsymbol{c}|\boldsymbol{c}^{split})$.

It follows from equations (14) and (15) that the appropriate ratio of transition probabilities for the split proposal is:

$$\frac{q(\boldsymbol{c}|\boldsymbol{c}^{split})}{q(\boldsymbol{c}^{split}|\boldsymbol{c})} = \frac{1}{\left(\frac{1}{2}\right)^{n_{c_i^{split}}+n_{c_j^{split}}-2}} \tag{16}$$

Similarly, the appropriate ratio of transition probabilities for the merge proposal is:

$$\frac{q(\boldsymbol{c}|\boldsymbol{c}^{merge})}{q(\boldsymbol{c}^{merge}|\boldsymbol{c})} = \left(\frac{1}{2}\right)^{n_{c_i}+n_{c_j}-2} \tag{17}$$

Therefore, the resulting acceptance probability (6) for a split proposal is based on the product of equations (8), (12), and (16). Likewise, the acceptance probability for a merge proposal is based on the product of equations (9), (13), and (17). By employing the Hastings (1970) version of the Metropolis (1953) algorithm when calculating the acceptance probability, we correct for the fact that the probability of proposing a particular split is smaller than the probability of proposing to merge the two resulting components.

This basic form of our procedure illustrates how we may nonincrementally update groups of observations. If a proposed split is appropriate for the data, it will likely be accepted, since neighbouring observations will lend support for the creation of a new component, bypassing the problem of being trapped in a local mode. Unfortunately, as stated earlier, we do not expect the simple random split version of this algorithm to perform well. Since components are split without reference to the observed data, the split proposals are unlikely to be appropriate, and hence are unlikely to be accepted.

3.3 Restricted Gibbs sampling split-merge proposals

Next, we describe a proposal distribution in which properties of the observed data are used to decide how to split mixture components via a restricted Gibbs sampling scan. This yields a method in which reasonable splits of components are more frequently proposed. First, as a way to validate the main algorithm, we introduce the Gibbs sampling split-merge proposal when the state prior to the Gibbs sampling scan is fixed. This pre-Gibbs state will be referred to as a *launch* state. We then present a generalized version in which the launch state is itself randomly selected; in particular, we can select the launch state by conducting several "intermediate" restricted Gibbs sampling scans.

3.3.1 Restricted Gibbs sampling proposals from a fixed launch state

Here, we replace the simple random split step in Algorithm 1 (Section 3.2) by a restricted Gibbs sampling scan on the component indicators, c_k, starting from a predetermined fixed launch state. The fixed state version of this algorithm is not expected to be of any particular use, except as a method to prove the validity of the subsequent random launch state algorithm. The restricted Gibbs sampling proposal distribution is more elaborate than typical Metropolis-Hastings proposals. However, by fixing the state prior to the scan, the proposal probabilities may be explicitly computed, and this yields a valid Metropolis-Hastings update. Each fixed launch state for the Gibbs sampling scan defines one particular Metropolis-Hastings algorithm. All fixed launch states will produce valid proposal distributions since they satisfy the usual requirements, such as independence of proposals from past states.

For the split proposal, once observations i and j have been assigned to different components, other observations (i.e. those in S) that belong to the merged component will first be assigned to one of these two split components in some predetermined manner. Once this launch state, $\boldsymbol{c}^{launch}$, is determined, one restricted Gibbs sampling scan is conducted to decide how the items in S will be allocated between the two split components. The Gibbs sampling scan is restricted in that it is only performed on a subset of the data (set S) and can assign these items to only two of the mixture components. To update a c_k in S via restricted Gibbs sampling, a new value of c_k is drawn from its (restricted) conditional distribution as follows:

$$P(c_k \mid c_{-k}, y_k) \;=\; \frac{n_{-k,c_k} \displaystyle\int F(y_k,\phi)\, dH_{-k,c_k}(\phi)}{n_{-k,c_i} \displaystyle\int F(y_k,\phi)\, dH_{-k,c_i}(\phi) \;+\; n_{-k,c_j} \displaystyle\int F(y_k,\phi)\, dH_{-k,c_j}(\phi)} \tag{18}$$

To simplify notation, we refer to component indicators for the launch state as c_k in equation (18). However, throughout the Gibbs sampling scan, these values (as well as the values for the other terms) are continually modified as the c_k are incrementally updated and used for the next computation leading to c^{split}. Here, c_{-k} represents the c_l for $l \neq k$ in $S \cup \{i,j\}$, $n_{-k,c}$ is the number of c_l for $l \neq k$ in $S \cup \{i,j\}$ that are equal to c, $F(y_k,\phi)$ is the likelihood, and $H_{-k,c}$ is the posterior distribution of ϕ based on the prior G_0 and data observations y_l such that $c_l = c$ where $l \in S \cup \{i,j\}$, for which $l \neq k$. Again, when G_0 is a conjugate prior for F, the above integrals may be analytically computed.

In general, the transition probability for a full sequential Gibbs sampling scan is a product of the conditional probabilities of each individual update. The restricted Gibbs sampling transition probability from state $\boldsymbol{c}^{launch}$ to $\boldsymbol{c}^{split}$ is the product of the probabilities of assigning each observation $k \in S$ to a particular split mixture component via Gibbs sampling from the fixed launch state as given by equation (18). In our algorithm, this product is the Metropolis-Hastings proposal probability, $q(\boldsymbol{c}^{split}|\boldsymbol{c})$.

For the merge proposal, as in the simple random split-merge procedure, there is still only one way to merge items in two components to one component, so $q(\boldsymbol{c}^{merge}|\boldsymbol{c}) = 1$. However, to obtain the corresponding probability, $q(\boldsymbol{c}|\boldsymbol{c}^{merge})$, we need to calculate the probability of generating the original split state from the fixed launch state in one Gibbs sampling scan. This is done in the same way as for the split proposal, except that no actual sampling is performed since the "split" state is already known.

As in the simple random split case, to obtain the Metropolis-Hastings acceptance probability, the appropriate split or merge proposal distribution ratio (now based on restricted Gibbs sampling) is substituted into equation (6). The prior and likelihood ratios in equation (6) remain as shown in Section 3.2.

Since only one scan of Gibbs sampling is conducted, we do not expect that the allocation of items between the two components has reached equilibrium. Because the Metropolis-Hastings proposal distribution can take any form and still produce a valid algorithm, lack of convergence will not invalidate this algorithm. However, it is quite likely that the proposed splits using a single iteration of Gibbs sampling may not be that sensible. That is, we would still like to improve the proposals so that the splits proposed are appropriate for the data.

3.3.2 Restricted Gibbs sampling proposals from a random launch state

Every fixed launch state for the algorithm of the previous section produces a valid Metropolis-Hastings update. As was discussed in Section 3.1, we may select a Markov chain transition at random from the set of valid transitions (Tierney 1994). Therefore, a launch state for the restricted Gibbs sampling scan may be chosen at random from the set of all fixed states. However, if only a single scan of Gibbs sampling is done from a launch state chosen uniformly at random, it may still lead to an unreasonable assignment of observations to the two mixture components.

To achieve more reasonable splits, several intermediate restricted Gibbs sampling scans are performed before the final scan. When calculating the split proposal probability, the result of the last intermediate Gibbs sampling scan is considered the random launch state, from which the restricted Gibbs sampling transition probability is explicitly calculated. We would prefer to incorporate all of the intermediate Gibbs sampling scans in the proposal distribution, but summing probabilities over all of these intermediate states is computationally infeasible. Although equilibrium will probably not be reached after only a few restricted Gibbs sampling scans, the clustering of observations between the two mixture components will probably be a better reflection of the actual attributes of the data (compared to just a single scan of Gibbs sampling). This version of the algorithm shows much improvement over the traditional Gibbs sampling method and previous versions of this algorithm (simple random split and Gibbs sampling proposals from a single fixed state). Increasing the number of intermediate Gibbs sampling scans leads to further improvement in convergence measured in iterations, but at the cost of computational time per iteration. In Section 4, the effect of varying this tuning parameter is examined.

Split proposal probabilities are calculated in the same way as for the fixed launch state Gibbs sampling proposals (equation 18). For the merge proposal, to obtain $q(\boldsymbol{c}|\boldsymbol{c}^{merge})$, the same intermediate Gibbs sampling operations that are performed when proposing a split must be conducted here to arrive at a launch state, even though no actual split is performed. The Gibbs sampling transition probability is calculated from the launch state (which is the last intermediate Gibbs sampling state) to the original split state. These operations are necessary in order to produce the correct proposal ratios. Below, the procedure for the restricted Gibbs sampling split-merge procedure with a random launch state is summarized.

3.3.3 Algorithm 2: Restricted Gibbs sampling split-merge procedure

1. Select two distinct observations, i and j, at random uniformly.

2. Let S denote the set of observations, $k \in \{1, \ldots, n\}$, for which $k \neq i$ and $k \neq j$, and $c_k = c_i$ or $c_k = c_j$.

3. Define the launch state, $\boldsymbol{c}^{launch}$, that will be used to compute Gibbs sampling probabilities. If $c_i = c_j$, then let c_i^{launch} be set to a new component such that $c_i^{launch} \notin \{c_1, \ldots, c_n\}$ and let $c_j^{launch} = c_j$. Otherwise, if $c_i \neq c_j$, then let $c_i^{launch} = c_i$ and $c_j^{launch} = c_j$. For every $k \in S$, set c_k^{launch} to either of the distinct components, c_i^{launch} or c_j^{launch}, as follows:

- Select an initial state by randomly setting, independently with equal probability, c_k^{launch} to either c_i^{launch} or c_j^{launch}.
- Modify $\boldsymbol{c}^{launch}$ by performing t intermediate restricted Gibbs sampling scans.

4. If items i and j are in the same mixture component, i.e. $c_i = c_j$, then:

   (a) Propose a new assignment of data items to mixture components, denoted as $\boldsymbol{c}^{split}$, in which component $c_i = c_j$ is split into two separate components, c_i^{split} and c_j^{split}. Define each element of the proposal vector, $\boldsymbol{c}^{split}$, as follows:
   - Let $c_i^{split} = c_i^{launch}$ (note that $c_i^{launch} \notin \{c_1, \ldots, c_n\}$)
   - Let $c_j^{split} = c_j^{launch}$ (which is the same as c_j)
   - For every observation $k \in S$, let c_k^{split} be set to either component c_i^{split} or c_j^{split} by conducting **one** final Gibbs sampling scan from the launch state, $\boldsymbol{c}^{launch}$
   - For observations $k \notin S \cup \{i, j\}$, let $c_k^{split} = c_k$

   (b) Calculate the proposal probability, $q(\boldsymbol{c}^{split}|\boldsymbol{c})$, by computing the Gibbs sampling transition probability from the launch state, $\boldsymbol{c}^{launch}$, to the final proposed state, $\boldsymbol{c}^{split}$. The Gibbs sampling transition probability is the product, over $k \in S$, of the probabilities of setting each c_k^{split} to its final value in the final Gibbs sampling scan.

   (c) Evaluate the proposal by the Metropolis-Hastings acceptance probability $a(\boldsymbol{c}^{split}, \boldsymbol{c})$. If the proposal is accepted, $\boldsymbol{c}^{split}$ becomes the next state in the Markov chain. If the proposal is rejected, the original vector, $\boldsymbol{c}$, remains as the next state.

5. Otherwise, if i and j are in different mixture components, i.e. $c_i \neq c_j$, then:

   (a) Propose a new assignment of data items to mixture components, denoted as $\boldsymbol{c}^{merge}$, in which components, c_i and c_j, are combined into a single component. Assign each element of the proposal vector, $\boldsymbol{c}^{merge}$, as follows:
   - Let $c_i^{merge} = c_j$
   - Let $c_j^{merge} = c_j$
   - For every observation $k \in S$, let $c_k^{merge} = c_j$
   - For observations $k \notin S \cup \{i, j\}$, let $c_k^{merge} = c_k$

   (b) Calculate the proposal probability, $q(\boldsymbol{c}|\boldsymbol{c}^{merge})$, by computing the Gibbs sampling transition probability from the launch state, $\boldsymbol{c}^{launch}$, to the original split configuration, $\boldsymbol{c}$. The Gibbs sampling transition probability is the product, over $k \in S$, of the probabilities of setting each c_k in the original split state to its original value in a (hypothetical) Gibbs sampling scan from the launch state.

   (c) Evaluate the proposal by the Metropolis-Hastings acceptance probability $a(\boldsymbol{c}^{merge}, \boldsymbol{c})$. If the proposal is accepted, $\boldsymbol{c}^{merge}$ becomes the next state. If the merge proposal is rejected, the original configuration, $\boldsymbol{c}$, remains as the next state.

6. Repeat steps 1-5 for T iterations.

3.3.4 Computational issues

When calculating the prior ratios in equations (8) and (9), counts of observations associated with each mixture component are required. Similar counts are needed when performing Gibbs sampling, as seen in equations (4) and (18). To improve efficiency, it is useful to maintain these counts incrementally in an array, by decrementing and incrementing appropriate counts when observations are moved between components. Depending on the statistical model, an incremental update of the relevant sufficient statistics may also be advantageous for use in likelihood calculations.

Another way of reducing the computational cost is to minimize the time spent searching the arrays for mixture components that are in use. Since the numerical value of a component indicator is only relevant in that it distinguishes components from one another, when creating new components, previous labels that are currently not being utilized may be recycled. This will reduce the time required to search through the count arrays.

As a final consideration, when calculating the ratios for the Metropolis-Hastings acceptance probability of equation (6), to avoid numerical overflow problems, the priors, likelihoods, and proposal probabilities should be calculated in terms of logarithms.

3.4 Cycling Metropolis-Hastings and Gibbs sampling updates

The split-merge Metropolis-Hastings algorithm produces a Markov chain that leaves the posterior distribution invariant. The Markov chain is also irreducible, since for any statistical model in which the data have non-zero prior probability, there is a non-zero probability that the chain started from any initial state will assign every observation to a separate mixture component as a result of a series of split moves. Further, except for some extremely degenerate models, the Markov chain is aperiodic, since there is a non-zero probability that the chain will remain in its current state (i.e. that at least some split or merge proposals have a non-zero probability of being rejected). The split-merge algorithm is therefore ergodic.

Even though Algorithm 2 is ergodic and produces nonincremental splits or merges of components, further improvements in convergence may be obtained by combining this algorithm with traditional Gibbs sampling. Algorithm 2 addresses the problem of making major changes in the allocation of items by moving observations as a cluster during a single iteration. However, it may take longer to move a single observation between components. In this situation, a "fine tuning" approach is required, which the regular Gibbs sampling scan can provide. Consequently, we propose combining these two algorithms by alternately performing a Metropolis-Hastings update and a full scan of Gibbs sampling. By doing this, we exploit the nonincremental (major) changes from the Metropolis-Hastings step, and the incremental (minor) refinement from the Gibbs sampling step.

Tierney (Section 2.4, 1994) notes that if Markov chain transition kernels are applied in cycles, and one of the kernels is ergodic, then the cycle kernel is not guaranteed to be ergodic. However, in our case, since both the Metropolis-Hastings and Gibbs sampling steps have a non-zero probability of leaving the state unchanged, applying each transition in turn will produce an ergodic Markov chain.

We can tune this algorithm by modifying the number of Metropolis-Hastings updates and the number of final Gibbs sampling scans in each full iteration. As expected, by increasing the values for both of these tuning parameters, convergence (measured in full iterations) is improved, but at the cost of computation time per iteration. In Section 4, we illustrate the effects of these modifications and provide guidelines for setting these tuning parameters.

4 Example: Bernoulli data with a conjugate Beta prior

In this section, we empirically compare the split-merge procedure (and its variants) to Gibbs sampling. We consider a categorical mixture model, in which the data, $\boldsymbol{y} = (y_1, \ldots, y_n)$, are independent and identically distributed, such that each observation, y_i, has m Bernoulli attributes, $(y_{i1}, \ldots, y_{im})$. Given the class, c_i, that observation i belongs to, the item's attributes are independent of each other. This type of model is common in latent class analysis (see, for example, Everitt 1984), in which the mixture components are considered latent classes that represent heterogeneous mechanisms which underly or produce the observed data. Neal (1992) considered a similar model when examining the performance of the Gibbs sampling procedure discussed in Section 2. For simplicity of exposition, we consider only dichotomous attributes, but the model and algorithms easily generalize to categorical attributes with more than two values.

4.1 The statistical model

In the Bayesian framework, the Bernoulli data can be modelled as a Dirichlet process mixture model. The observations, $y_i = (y_{i1}, \ldots, y_{im})$, are multivariate Bernoulli, such that each $y_{ih}|\theta_i \sim \text{Bernoulli}(\theta_{ih})$, and the likelihood is as follows:

$$F(y_i, \theta_i) \quad = \quad \prod_{h=1}^{m} \theta_{ih}^{y_{ih}} \, (1 - \theta_{ih})^{1-y_{ih}} \tag{19}$$

The parameters of a component, θ, give the probabilities of each attribute having the value one. Each such probability is given a Beta distribution prior with parameters (β_1, β_0). Under G_0, which is the prior over the vector $\theta = (\theta_1, \ldots, \theta_m)$, the θ_h are independent. (Note that here we use subscripts to denote different attributes rather than different observations.) The density for G_0 is:

$$P(\theta) \quad = \quad \prod_{h=1}^{m} \left(\frac{\Gamma(\beta_{1,h} + \beta_{0,h})}{\Gamma(\beta_{1,h}) \, \Gamma(\beta_{0,h})} \, \theta_h^{\beta_{1,h}-1} \, (1 - \theta_h)^{\beta_{0,h}-1} \right) \tag{20}$$

where $\beta_{0,h}$ and $\beta_{1,h}$ are greater than zero.

Because this is a conjugate prior for $F(y_i, \theta_i)$, the model parameters may be integrated away. To update c_i via Gibbs sampling, a new value of c_i is drawn from its conditional distribution (equation 4), which for this model is the following, when the Beta prior and Bernoulli likelihood are substituted:

$$\begin{aligned} P(c_i = c_j \text{ for some } j \neq i \mid c_{-i},\, y_i) &= b\,\frac{n_{-i,c_j}}{n-1+\alpha} \prod_{h=1}^{m} \frac{\sum_{k\neq i} \delta(c_k, c_j)\delta(y_{kh}, y_{ih}) + \beta_{y_{ih},h}}{n_{-i,c_j} + \beta_{0,h} + \beta_{1,h}} \\ P(c_i \neq c_j \text{ for all } j \neq i \mid c_{-i},\, y_i) &= b\,\frac{\alpha}{n-1+\alpha} \prod_{h=1}^{m} \frac{\beta_{y_{ih},h}}{\beta_{0,h} + \beta_{1,h}} \end{aligned} \tag{21}$$

The delta function $\delta(x, y)$ is equal to one if $x = y$ and zero otherwise. The term $\sum_{k\neq i} \delta(c_k, c_j)\delta(y_{kh}, y_{ih})$ counts the number of observations associated with component c_j that match y_i with respect to attribute h. The second formula gives the probability for setting c_i to a new mixture component that is currently not assigned to any other observation. In both equations, b is the factor that normalizes the distribution to sum to one.

For the Metropolis-Hastings acceptance probability in equation (6), the prior is calculated as shown in equation (7). The appropriate ratio of the transition probabilities based on restricted Gibbs sampling is obtained by using the first formula in equation (21). The likelihood (equation 11) based on the Bernoulli-Beta model is as follows:

$$L(\boldsymbol{c}|\boldsymbol{y}) = \prod_{c=1}^{D} \prod_{k\,:\,c_k=c} \prod_{h=1}^{m} \frac{\sum_{i<k} \delta(c_i, c)\delta(y_{ih}, y_{kh}) + \beta_{y_{kh},h}}{n_{k,c} + \beta_{0,h} + \beta_{1,h}} \tag{22}$$

If we interchange the products over k and h, equation (22) simplifies and can be computed as follows:

$$L(\boldsymbol{c}|\boldsymbol{y}) = \prod_{c=1}^{D} \prod_{h=1}^{m} \frac{\left[\Gamma\left(\sum_k \delta(c_k, c)\delta(y_{kh}, 0) + \beta_{0,h}\right)/\Gamma(\beta_{0,h})\right] \left[\Gamma(\sum_k \delta(c_k, c)\delta(y_{kh}, 1) + \beta_{1,h})/\Gamma(\beta_{1,h})\right]}{\Gamma(n_c + \beta_{0,h} + \beta_{1,h})/\Gamma(\beta_{0,h} + \beta_{1,h})} \tag{23}$$

As mentioned in Section 3.3.4, it is useful to incrementally update the sufficients statistics for the model. Here, efficiency is improved by maintaining a count of items associated with each mixture component having particular values for each attribute. This array of counts can be used for Gibbs sampling (equation 21) and in the likelihood calculation above (equation 23).

4.2 The simulated data sets

Although Dirichlet process mixture models consider the number of mixture components to be countably infinite, the model can nevertheless be applied to finite mixtures. The prior chosen ensures that some of the infinite number of components are given significant probability, so overfitting does not occur. The model will assign a small probability to observations being from one of the infinite number of additional components, but this does not cause serious problems. For simplicity, we will therefore test the algorithms on simulated data from a finite mixture.

Our primary goal is to partition observations into appropriate latent classes using the Bernoulli-Beta Dirichlet process mixture model. Computationally, this classification problem becomes more difficult as the dimensionality increases and as the sets of attributes that distinguish the various components become more similar in structure. We illustrate this difficulty by considering three simulated data sets, in which the number of attributes is increased so that the different components appear more alike as the dimensionality increases.

The data are composed of five equally-probable mixture components, in which each component produces a distribution over m dichotomous attributes. To maintain uniformity amongst the examples, $n = 100$ observations were produced for each example, and 20 observations were generated from each of the five mixture components.

Data for the three simulated examples were randomly generated from the mixture distributions shown in Tables 1-3. The mixture components are distinguished by the first four attributes, which for consistency, have been kept constant throughout all three examples. The examples differ in the number of additional attributes. Dimensionality is increased by simply replicating the distribution for the last attribute, which makes the components more similar, and thereby, more difficult to distinguish. Note the intentional asymmetry in the construction of the mixture components, in which the first three components are more similar than the last two components. This is intended to test whether the split-merge algorithms can handle "three-way" splits.

For the following demonstrations of the algorithms, the Dirichlet process parameter, α, is set to one. A small value of α implies that the number of mixture components present in the data set is likely to be small. The β_{0h} and β_{1h} parameters for the Beta prior distribution have also been set to one. These priors may not be realistic, but for consistency, these values are fixed at one during the simulations. In actual problems, α and the β's would be set by prior knowledge or given higher-level priors.

Table 1: True mixture distribution for Example 1.

c	$P(c_i = c)$	$P(y_{ih} = 1 \mid c_i = c), h = 1, \ldots, 6$
1	0.2	.95 .95 .95 .95 .95 .95
2	0.2	.05 .05 .05 .05 .95 .95
3	0.2	.95 .05 .05 .95 .95 .95
4	0.2	.05 .05 .05 .05 .05 .05
5	0.2	.95 .95 .95 .95 .05 .05

Table 2: True mixture distribution for Example 2.

c	$P(c_i = c)$	$P(y_{ih} = 1 \mid c_i = c), h = 1, \ldots, 15$
1	0.2	.95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95
2	0.2	.05 .05 .05 .05 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95
3	0.2	.95 .05 .05 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95 .95
4	0.2	.05 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05
5	0.2	.95 .95 .95 .95 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05 .05

Table 3: True mixture distribution for Example 3.

c	$P(c_i = c)$	$P(y_{ih} = 1 \mid c_i = c), h = 1, \ldots, 18$																	
1	0.2	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95
2	0.2	.05	.05	.05	.05	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95
3	0.2	.95	.05	.05	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95	.95
4	0.2	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
5	0.2	.95	.95	.95	.95	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05

4.3 Performance of the algorithms

For each example, the Gibbs sampling algorithm was compared to five versions of the split-merge algorithm: Simple Random Split, Split-Merge (0,1,0), Split-Merge (0,1,1), Split-Merge (5,1,0), and Split-Merge (5,1,1). The first number in parentheses is the number of intermediate Gibbs sampling scans to reach the launch state, the second is the number of Metropolis-Hastings updates in a single iteration, and the third is the number of complete Gibbs sampling scans after the final Metropolis-Hastings update. For each algorithm, all observations were assigned to the same mixture component for the initial state, and each algorithm was run for 2000 iterations. All simulations were performed in Matlab, Version 5.3, on a SGI system with a 200 MHz MIPS processor.

The performance of each algorithm was evaluated by examining trace plots (Figures 1-3) and the computation time per iteration (Table 4). In each trace plot, the five values plotted are the fraction of observations associated with the most common, two most common, three most common, four most common, and five most common mixture components. Since each of the five components appear equally in the samples, if the true situation were captured exactly, the five traces would occur at values of 0.2, 0.4, 0.6, 0.8, and 1.0.

Table 4: Time per iteration in seconds for algorithms tested.

Algorithm	*Example 1*	*Example 2*	*Example 3*
Gibbs Sampling	1.1	1.5	1.6
Simple Random Split	0.2	0.4	0.5
Split-Merge (0,1,0)	0.3	0.7	0.9
Split-Merge (0,1,1)	1.1	1.9	2.2
Split-Merge (5,1,0)	0.9	1.9	2.5
Split-Merge (5,1,1)	1.6	3.2	4.0

4.3.1 Example 1

The first example is the simplest. It is relatively low-dimensional (six attributes) and has five well-separated mixture components. From the trace plots (Figure 1), it appears that all of the algorithms except for the Simple Random Split have appropriately separated the data into the five

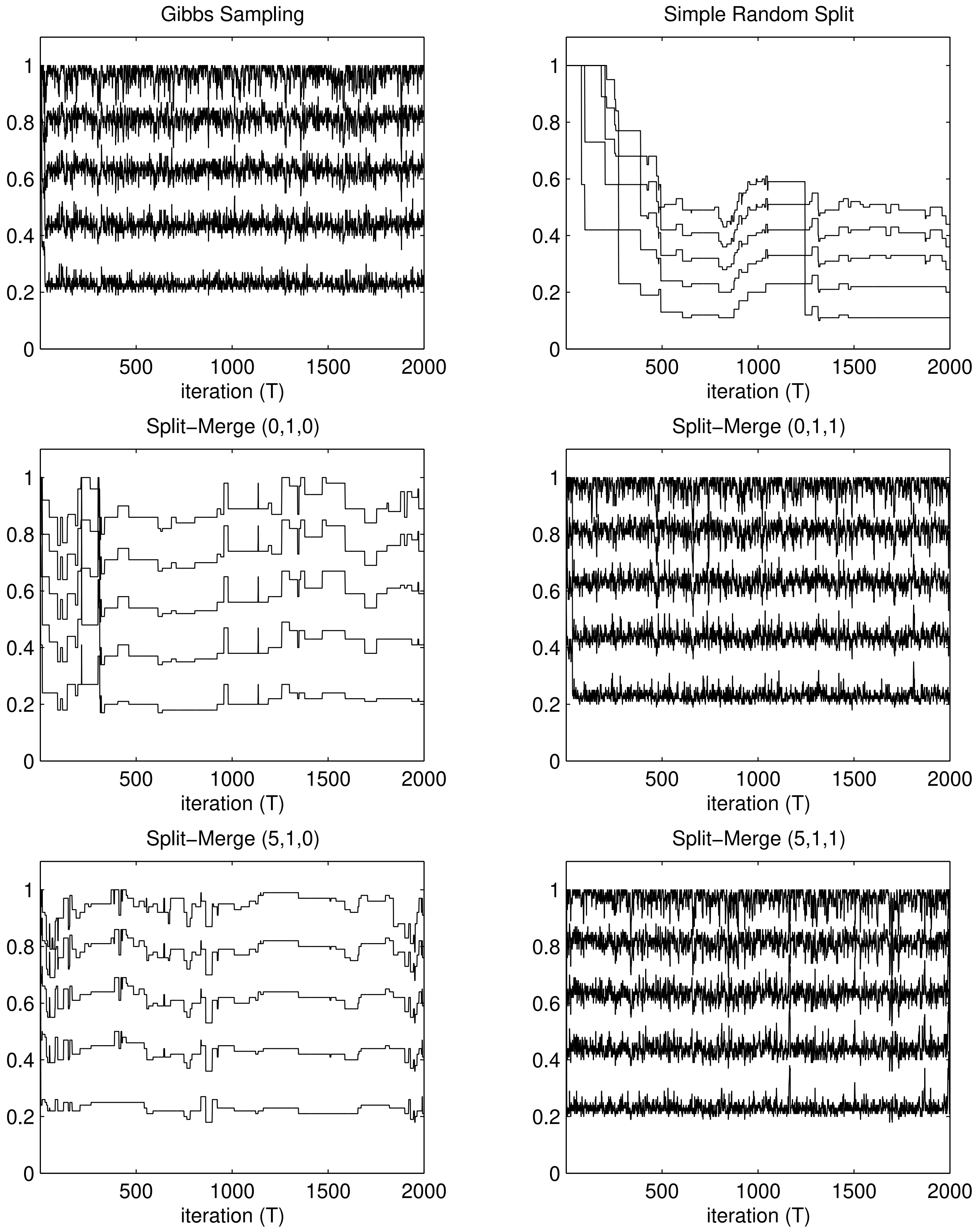

Figure 1: Trace plots of the six algorithms in Example 1.

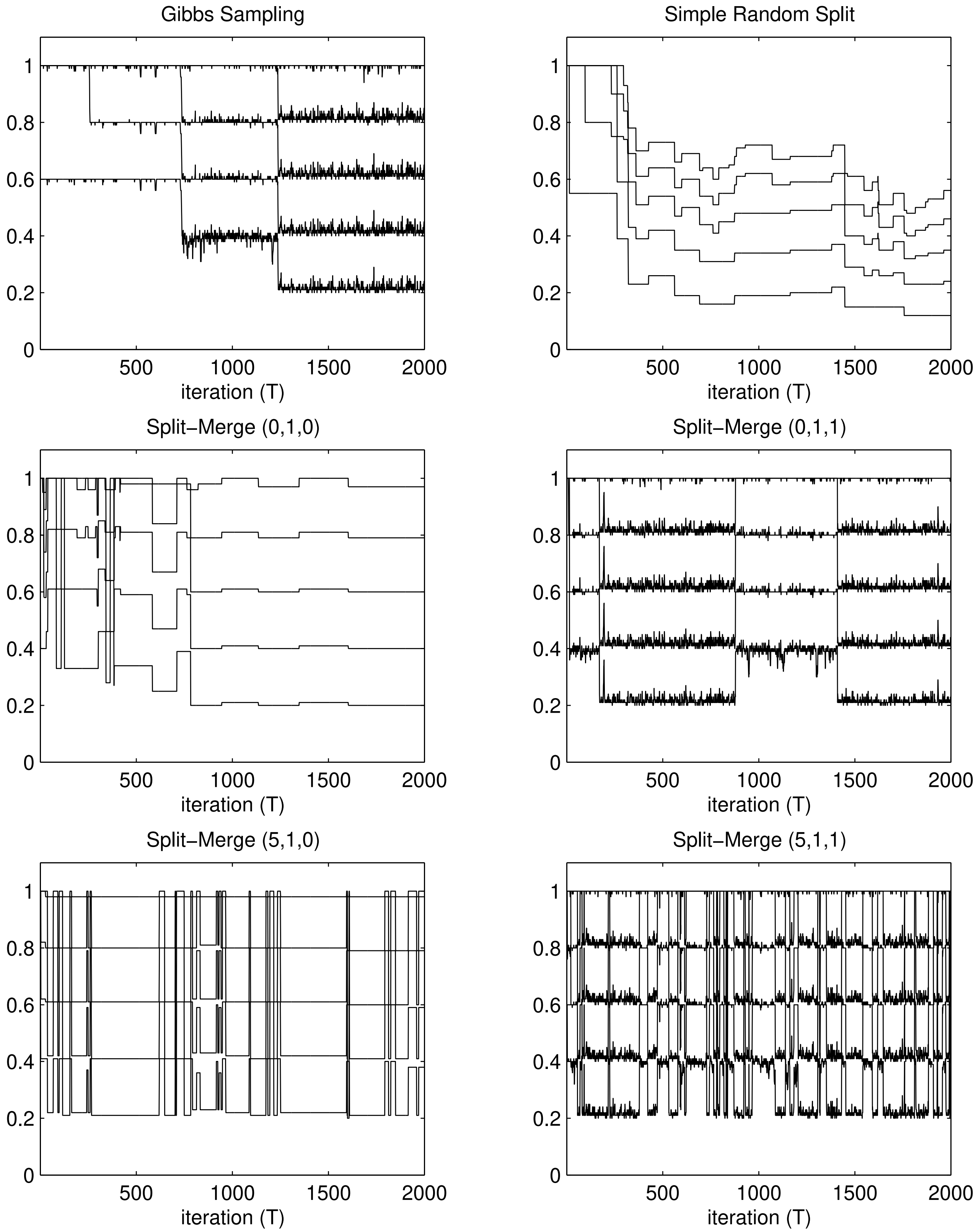

Figure 2: Trace plots of the six algorithms in Example 2.

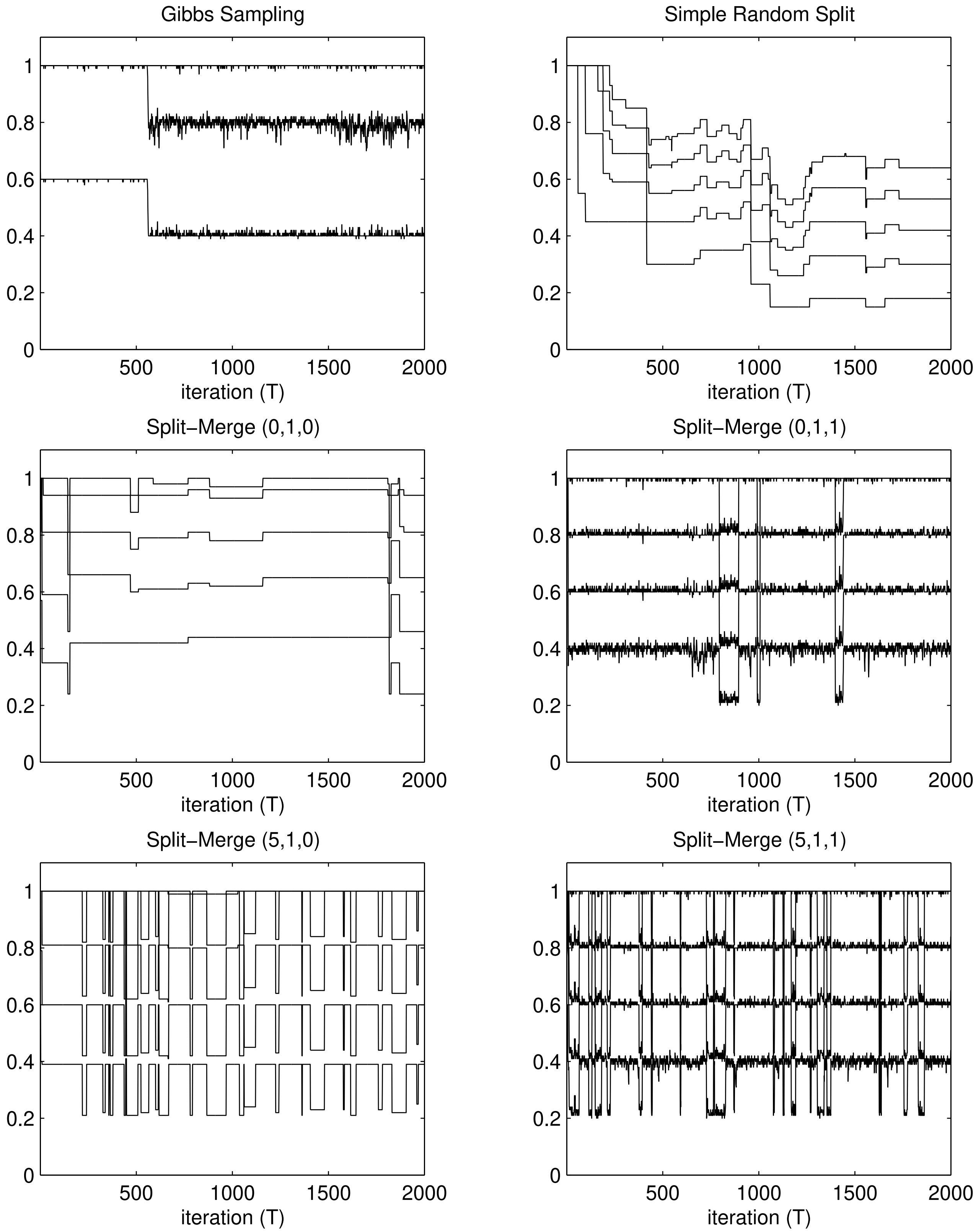

Figure 3: Trace plots of the six algorithms in Example 3.

mixture components. As discussed earlier, the Simple Random Split is not expected to converge rapidly, even in simple situations, because the split proposals are usually nonsensical.

By inspection, Gibbs sampling, Split-Merge (0,1,1) and (5,1,1) seem to have short burn-in times and mix about equally well. To further examine the performances of these three algorithms, the autocorrelation times were computed for the first trace on the plots (corresponding to the fraction of items associated with the most common mixture component) and for the indicator variable, called $I_{13,42}$, which codes if observations 13 and 42 are assigned to the same mixture component. Items 13 and 42 were generated from components 3 and 2, respectively. However, due to random noise, item 42 differs in one of the four distinguishing attributes from its true distribution (given in Table 2), which makes it as likely to have come from component 3 as from its actual component, 2. Consequently, the mean of this indicator function is approximately 0.5 (i.e. these two items should be grouped together half of the time).

The autocorrelation time is defined as one plus twice the sum of the autocorrelations for a quantity at lags one up to infinity. This is the factor by which the sample size is effectively reduced when estimating the expectation of that quantity, when compared to an estimate based on independent draws from the posterior distribution (Ripley 1987, Section 6.3). Autocorrelation time was estimated by one plus the sum of the estimated autocorrelations up to the lag where the autocorrelations are approximately zero. The results are presented in Table 5.

Table 5: Autocorrelation times for algorithms that converged in Example 1.

Algorithm	*Trace 1*	*Indicator* $I_{13,42}$
Gibbs Sampling	2.0	0.9
Split-Merge (0,1,1)	2.1	0.7
Split-Merge (5,1,1)	4.0	0.7

Gibbs sampling, Split-Merge (0,1,1), and Split-Merge (5,1,1) have approximately equal autocorrelation times. Note that the slightly higher trace 1 autocorrelation time for Split-Merge (5,1,1) is not statistically significant, but is due to the chance distribution of the occasional sharp peaks visible in the trace plot (Figure 1). In this simple problem, Gibbs sampling is successful in correctly splitting the items amongst the five components, so the split-merge algorithms are not necessary. Comparing Split-Merge (0,1,1) and (5,1,1), we see that the addition of several intermediate Gibbs sampling scans does not improve autocorrelation times and are not worth the extra computation time. Split-Merge (0,1,0) and (5,1,0), which do not include the final complete Gibbs sampling scan, also separate the data into five components. However, from the trace plots, it is clear that mixing is much slower than when intermediate Gibbs sampling scans are included.

4.3.2 Example 2

Example 2 is a higher dimensional problem (fifteen attributes), in which the posterior distribution, given the priors assigned, gives substantial probability to configurations with four or five main components. From the trace plots (Figure 2), we see that Gibbs sampling is extremely slow to separate the data into four or five components, and remains stuck in a particular configuration for a long time. When Gibbs sampling is initialized to all observations in a different mixture component (plot not shown), the observations are quickly split into five components, but then stays in that configuration for a very long time, thereby failing to explore the true posterior distribution. Split-Merge (5,1,1), on the other hand, mixes rapidly between the four and five component configurations. There is a drawback in that Split-Merge (5,1,1) takes double the time per iteration compared to Gibbs sampling. However, since Gibbs sampling fails to mix adequately over the posterior distribution and Split-Merge (5,1,1) reaches equilibrium almost immediately, the extra time per iteration is clearly well spent.

Split-Merge (5,1,0) seems to do reasonably well in moving between four and five components, but minor changes (moving one or two observations between components) are still a problem. The split-merge algorithms without any intermediate Gibbs sampling scans are slow in mixing between four and five components; however, Split-Merge (0,1,1) seems to mix better than Gibbs sampling. Again, the Simple Random Split is unable to separate the data adequately and performs the worst.

4.3.3 Example 3

Example 3 is the highest dimensional example (eighteen attributes) considered. The posterior distribution, again given the prior distribution we have used, is a mixture of (mainly) four and five component configurations. The trace plots (Figure 3) show that Gibbs sampling remains in an incorrect split that is not typical of the true posterior distribution for the entire 2000 iteration run. The mixture components are now quite similar, so an incremental creation of a new component via Gibbs sampling is quite rare. If each item is initially assigned to a different mixture component (plot not shown), Gibbs sampling splits the data into five components immediately, but takes roughly 1000 iterations to move to the four-component configuration, showing that it mixes poorly between the four and five component configurations.

Split-Merge (5,1,1) separates the observations into the proper configuration immediately and mixes well between the four and five components. Split-Merge (5,1,0) also mixes between four and five components, but the minor adjustments are slow. The two split-merge algorithms without any intermediate Gibbs sampling scans find the four and five component configurations, but are stuck in the four-component split for a long time. This is a result of non-optimal Metropolis-Hastings split proposals. Again, the Simple Random Split performs extremely poorly.

4.3.4 Summary of results

Both the Gibbs sampling and split-merge methods seem to work reasonably well in low-dimensional cases. However, as the classification task becomes increasingly difficult, Gibbs sampling mixes exceedingly poorly. The cycled split-merge version that includes both intermediate Gibbs sampling scans and a full overall Gibbs sampling scan is the most successful split-merge variation. Its split proposals are more appropriate, yielding better mixing between different major configurations, while the final Gibbs sampling scan handles the necessary minor adjustments. The split-merge algorithms that include intermediate Gibbs sampling scans are successful in handling three-way splits, even though this must be done by two two-way splits. Computation time per iteration is greater than for Gibbs sampling, but in situations where Gibbs sampling is unable to arrive at the correct stationary distribution in any reasonable length of time, this burden is clearly acceptable.

4.4 Tuning parameters

In this section, we examine the role that the tuning parameters play in our split-merge algorithm. There are three adjustable parameters: the number of intermediate Gibbs sampling scans, the number of Metropolis-Hastings updates conducted in a single iteration, and the number of complete Gibbs sampling scans conducted after the Metropolis-Hastings updates. We reconsider the data from Example 2, and examine the effect of varying each tuning parameter holding the other two parameters constant. Table 6 displays the computational time per iteration and autocorrelation times for trace 1 and indicator $I_{13,42}$ for various settings of this algorithm. Trace plots are shown in Figures 4-6.

Table 6: Effects of the tuning parameters.

Algorithm	*Time per iteration in seconds*	*Autocorrelation time for Trace 1*	*Autocorrelation time for Indicator* $I_{13,42}$
Split-Merge (1,1,1)	2.2	57.4	1.5
Split-Merge (3,1,1)	2.7	40.5	1.5
Split-Merge (5,1,1)	3.2	31.9	1.6
Split-Merge (10,1,1)	4.6	26.7	1.6
Split-Merge (20,1,1)	7.0	24.2	1.6
Split-Merge (100,1,1)	28.8	18.4	1.4
Split-Merge (1,1,1)	2.2	57.4	1.5
Split-Merge (1,2,1)	3.1	48.0	2.0
Split-Merge (1,3,1)	4.1	19.5	1.8
Split-Merge (1,4,1)	5.1	19.2	1.8
Split-Merge (1,5,1)	6.2	17.6	1.9
Split-Merge (1,1,0)	1.0	165.8	41.7
Split-Merge (1,1,1)	2.2	57.4	1.5
Split-Merge (1,1,2)	3.3	63.5	1.7
Split-Merge (1,1,3)	4.5	35.9	1.0
Split-Merge (1,1,5)	6.7	35.3	1.0

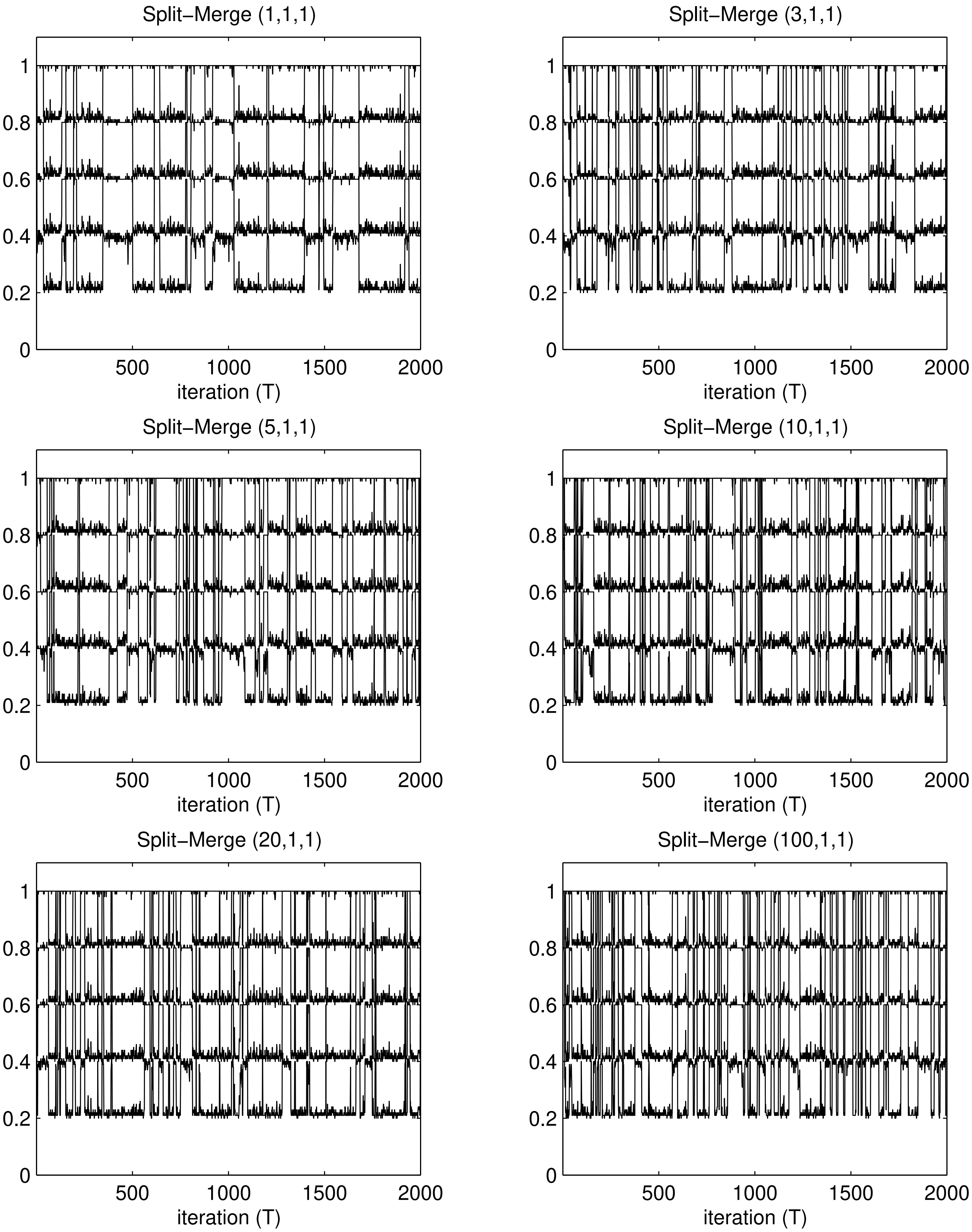

Figure 4: Trace plots examining the effect of varying the number of intermediate Gibbs sampling scans.

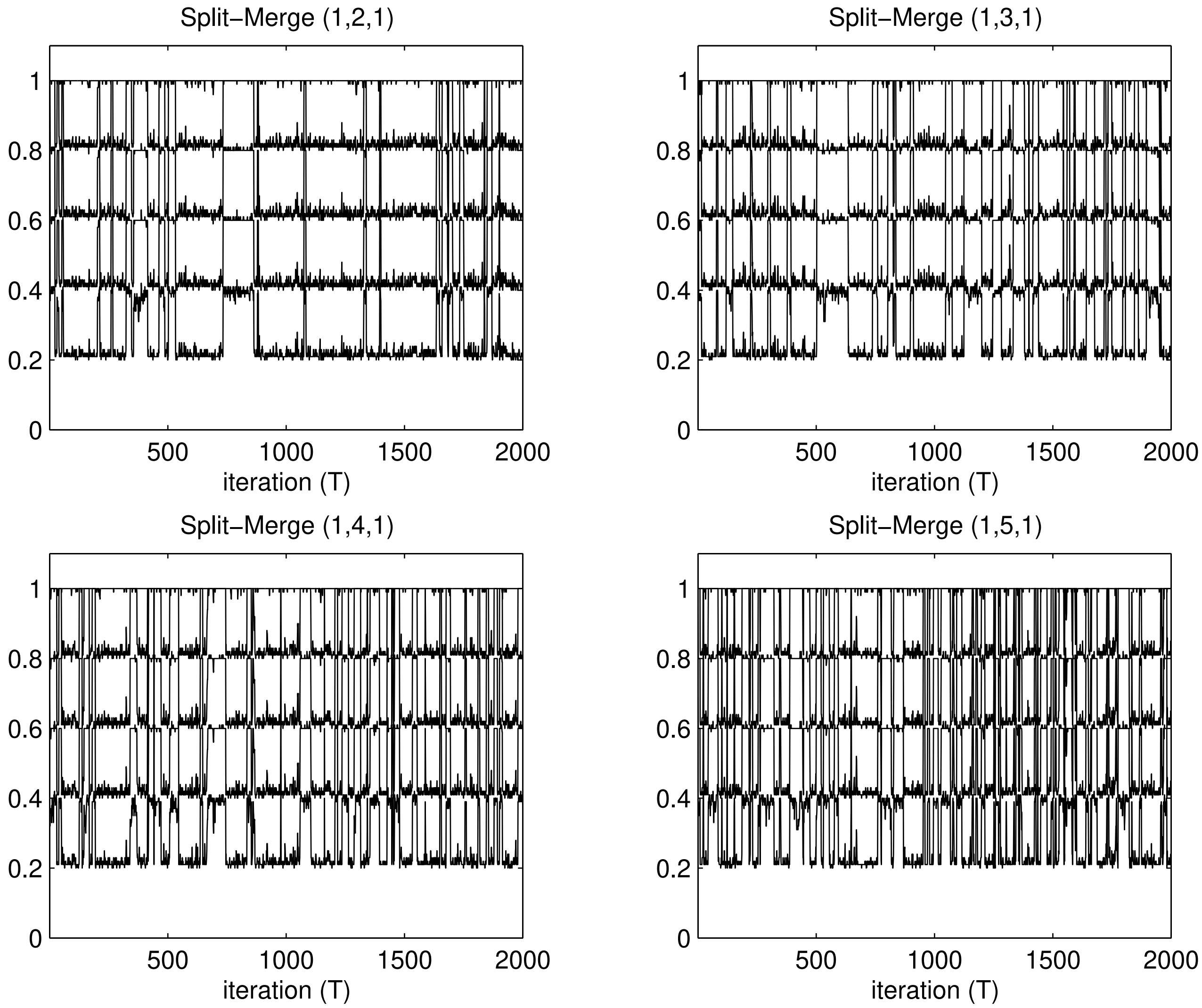

Figure 5: Trace plots examining the effect of varying the number of Metropolis-Hastings updates in a single iteration.

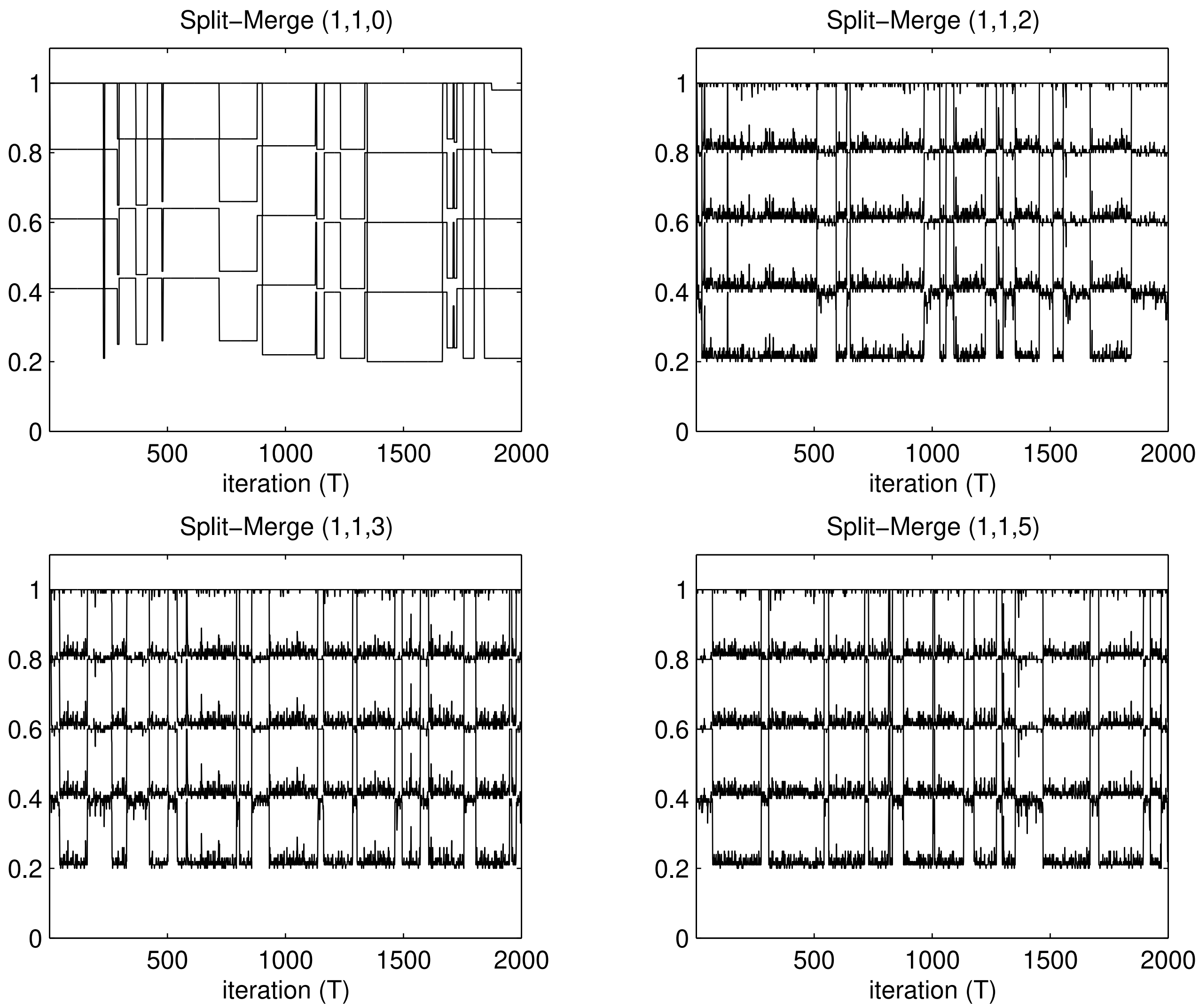

Figure 6: Trace plots examining the effect of varying the number of final complete Gibbs sampling scans.

4.4.1 Varying the number of intermediate Gibbs sampling scans

Better splits are expected when more intermediate Gibbs sampling scans are performed, since the proposed splits will be closer to the restricted equilibrium distribution. From Figure 4, we observe improved mixing when the number of intermediate Gibbs sampling scans used to arrive at the launch state is increased. The autocorrelation times for trace 1 are lower (compare five vs. one-hundred intermediate scans), but there is an increased cost of computational time per iteration (3.2 vs. 28.8 seconds). However, after five intermediate scans, the improvement is fairly minimal. In Table 7, the acceptance rates for the Metropolis-Hastings updates are given. There is only minor improvement in the acceptance rate after five scans. When these simulations were repeated with different pseudo-random seeds, we found that, on occasion, five intermediate Gibbs sampling scans would appear as good as 20 or 100 intermediate scans (in terms of rejection rate and autocorrelation times). Therefore, it seems that improvements level off after only a few intermediate Gibbs sampling scans, and additional scans are not worth the increased computational time.

Table 7: Acceptance rate for different numbers of intermediate Gibbs sampling scans.

Algorithm	*Acceptance rate in percent*
Split-Merge (1,1,1)	1.5
Split-Merge (3,1,1)	3.1
Split-Merge (5,1,1)	3.3
Split-Merge (10,1,1)	4.1
Split-Merge (20,1,1)	3.7
Split-Merge (100,1,1)	4.3

4.4.2 Varying the number of Metropolis-Hastings updates per iteration

Autocorrelation times decrease and mixing between four and five components improves when the number of Metropolis-Hastings updates is increased. However, these improvements seem to taper off after three Metropolis-Hastings updates, while the computational cost continues to increase. Three updates per iteration seem to be the best number in this particular example. This suggests that a full Gibbs sampling scan is not imperative after each Metropolis-Hastings update and may be a waste of computational time.

4.4.3 Varying the number of final complete Gibbs sampling scans

From Figure 6, we observe that the Metropolis-Hastings updates need to be supplemented by some complete Gibbs sampling scans in order to make minor clustering changes. This is also evident from the autocorrelation times for $I_{13,42}$, which drop from 41.7 to 1.5 when a full scan of Gibbs sampling is included. Splitting or merging these two particular observations is most easily done by a small-scale incremental update.

The autocorrelation time for trace 1 also decreases as the number of final scans is increased. However, from the trace plots, it appears that differences in mixing are minimal. The time per iteration grows when the number of final Gibbs sampling scans is increased, and beyond one Gibbs sampling scan per iteration, it does not appear that the improvements in autocorrelation times offset this.

4.4.4 Guidelines for selecting tuning parameters

The most critical tuning parameter is the number of intermediate Gibbs sampling scans, since this controls the quality of the Metropolis-Hastings split proposals. For this problem (and other similar problems that we have examined), a small number of scans (say, 4 to 6) is the best compromise between computation time and autocorrelation time. For data sets in which the number of observations per mixture component is large, more intermediate Gibbs sampling scans may be required.

Two or three Metropolis-Hastings updates per iteration appear to provide a good balance between computation time and autocorrelation time. Because final Gibbs sampling scans take a long time to compute, it is better to perform them after multiple Metropolis-Hastings updates, which are relatively faster.

Conducting a final Gibbs sampling scan after Metropolis-Hastings updates has been shown to be a necessity. However, since these final Gibbs sampling scans are computationally expensive, performing more than one scan seems undesirable.

5 Discussion

The split-merge Metropolis-Hastings procedure has been shown to be an improvement over traditional Gibbs sampling in high-dimensional problems in which mixture components are similar. The nonincremental clustering changes of our method avoid the problem of being trapped in local modes, and the posterior distribution is fully explored. Implementing this method is relatively simple and does not become more difficult in higher dimensions. It should be straightforward to apply this method to any conjugate model, including normal mixture models for real-valued data with the conjugate normal-inverse gamma priors for the mean and variance. The quality of the proposals can be controlled by varying the number of intermediate Gibbs sampling scans.

One inefficiency of this procedure that we are currently investigating is the random selection and treatment of the observations, i and j, that define the split or merge operation. If i and j are initially in the same component (hence, a split is proposed), the probability that the "correct" split configuration will be proposed can be as low as 25%, even when the split should be into components of equal size. After i and j are set to different mixture components, their component labels cannot change. Two types of problems may arise from this restriction. First, if i and j actually belong to the same mixture component, then these two items have unnecessarily been separated. If this problem does not occur (so i and j should be separated), a labelling problem is still possible. The initial random split of the other items in the merged component could assign labels biased towards a split that is opposite to the fixed labels of i and j. The intermediate Gibbs sampling scans may not overcome this initial bias, and are not expected to, since the component labels of i and j are not involved in these intermediate scans. Therefore, i and j end up in the "wrong" mixture components. This could be fixed by allowing the labels of i and j to adapt during the intermediate Gibbs sampling, but this would then need to be accounted for in the Metropolis-Hastings acceptance probability.

Our algorithm could also be easily modified by replacing the intermediate restricted Gibbs sampling scans by Markov chain updates of some other type. However, replacing the final Gibbs sampling scan (from the launch state) with some other update would be possible only if the transition probability for this update could be calculated.

When the data is very high dimensional, or there are very many observations, it is possible that our algorithm may only rarely accept the splits and merges that are proposed, even if they are appropriate. This potential problem is most easily seen for merge proposals, which will have a high probability of being accepted only if the current split configuration has a high probability of being produced from the launch state in a single Gibbs sampling scan. For difficult problems, however, the distance that can be traversed in one Gibbs sampling scan may be small compared to the extent of posterior variation. Determining whether a split or merge proposal should be accepted is analogous to the problem of Bayesian model choice, for which the introduction of intermediate models has been found to be useful (see Gelman and Meng 1998). Some analogous technique may be useful if a low acceptance rate for split and merge proposals proves to be a problem in practice.

Our main priority for future research is extending the algorithm to handle non-conjugate models, in which the model parameters cannot be analytically integrated away. We believe that this should be possible, but how well the resulting algorithm will work remains to be seen.

Finally, the technique we use of producing Metropolis-Hastings proposals using restricted Gibbs sampling scans may be applicable in other contexts as well. Simple Gibbs sampling often fails to work well when dependencies between variables prevent one of them from changing much (or at all) when the others are fixed. This can be overcome by performing Gibbs sampling on blocks of several variables, provided that the conditional distribution for all variables in a block can be sampled from. When sampling for all variables in a block is infeasible, one might propose to change all the variables in a block simultaneously using a Metropolis-Hastings update, but finding a suitable multi-dimensional proposal distribution can be difficult. An alternative that seems worth exploring is to initially propose a change to only one (or a few) of the variables in the block, and to find appropriate proposed values for the other variables in the block using restricted Gibbs sampling updates, from some randomly chosen initial state. It should be possible to compute a suitable acceptance probability to make this a valid Markov chain update, as in the algorithms we have presented in this article.

Acknowledgements

The first author acknowledges support from a Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship. The second author's research was supported by the Natural Sciences and Engineering Research Council of Canada and by the Institute for Robotics and Intelligent Systems.

References

Antoniak, C. E. (1974) "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems", *Annals of Statistics*, vol. 2, pp. 1152-1174.

Blackwell, D. and MacQueen, J. B. (1973) "Ferguson distributions via Pólya urn schemes", *Annals of Statistics*, vol. 1, pp. 353-355.

Bush, C. A. and MacEachern, S. N. (1996) "A semiparametric Bayesian model for randomised block designs", *Biometrika*, vol. 83, pp. 275-285.

Celeux, G., Hurn, M., and Robert, C. P. (2000) "Computational and inferential difficulties with mixture posterior distributions", *Journal of the American Statistical Association*, to appear.

Escobar, M. D. (1994) "Estimating normal means with a Dirichlet process prior", *Journal of the American Statistical Association*, vol. 89, pp. 268-277.

Escobar, M. D. and West, M. (1995) "Bayesian density estimation and inference using mixtures", *Journal of the American Statistical Association*, vol. 90, pp.577-588.

Everitt, B. S. (1984) *An Introduction to Latent Variable Models*, London: Chapman and Hall.

Everitt, B. S. and Hand, D. J. (1981) *Finite Mixture Distributions*, London: Chapman and Hall.

Ferguson, T. S. (1973) "A Bayesian analysis of some nonparametric problems", *Annals of Statistics*, vol. 1, pp. 209-230.

Ferguson, T. S. (1983) "Bayesian density estimation by mixtures of normal distributions", in H. Rizvi and J. Rustagi (editors) *Recent Advances in Statistics*, pp. 287-303, New York: Academic Press.

Gelman, A. and Meng, X.-L. (1998) "Simulating normalizing constants: From importance sampling to bridge sampling to path sampling", *Statistical Science*, vol. 13, pp. 163-185.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (editors) (1996) *Markov Chain Monte Carlo in Practice*, London: Chapman and Hall.

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

Green, P. J. and Richardson, S. (1999) "Modelling heterogeneity with and without the Dirichlet process", draft manuscript.

Hastings, W. K. (1970) "Monte Carlo sampling methods using Markov chains and their applications", *Biometrika*, vol. 57, pp. 97-109.

MacEachern, S. N. (1994) "Estimating normal means with a conjugate style Dirichlet process prior", *Communications in Statistics: Simulation and Computation*, vol. 23, pp. 727-741.

MacEachern, S. N. (1998) "Computational methods for mixture of Dirichlet process models", in D. Dey, *et al* (editors) *Practical Nonparametric and Semiparametric Bayesian Statistics*, pp. 23-43, New York: Springer-Verlag.

McLachlan, G. J. and Basford, K. E. (1988) *Mixture Models: Inference and Applications to Clustering*, New York: Marcel Dekker.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953) "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, vol. 21, pp. 1087-1092.

Neal, R. M. (1992) "Bayesian mixture modeling", in C. R. Smith, G. J. Erickson, and P. O. Neudorfer (editors) *Maximum Entropy and Bayesian Methods: Proceedings of the 11th International Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis*, Seattle, 1991, pp. 197-211, Dordrecht: Kluwer Academic Publishers.

Neal, R. M. (2000) "Markov chain sampling methods for Dirichlet process mixture models", *Journal of Computational and Graphical Statistics*, vol. 9, pp. 249-265.

Phillips, D. B. and Smith, A. F. M. (1996) "Bayesian model comparisons via jump diffusions", in Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (editors) *Markov Chain Monte Carlo in Practice*, pp. 215-239, London: Chapman and Hall.

Richardson, S. and Green, P. J. (1997) "On Bayesian analysis of mixtures with an unknown number of components" (with discussion), *Journal of the Royal Statistical Society, Series B*, vol. 59, pp. 731-792.

Ripley, B. D. (1987) *Stochastic Simulation*, New York: Wiley.

Tierney, L. (1994) "Markov chains for exploring posterior distributions" (with discussion), *Annals of Statistics*, vol. 22, pp. 1701-1762.

Titterington, D. M., Smith, A. F. M., and Makov, U. E. (1985) *Statistical Analysis of Finite Mixture Distributions*, Chichester, New York: Wiley.