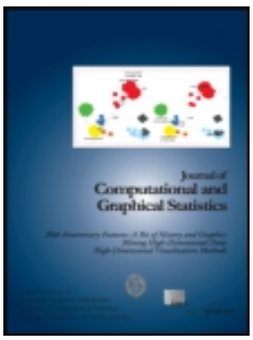

Taylor & Francis
Taylor & Francis Group

Journal of Computational and Graphical Statistics

ISSN: 1061-8600 (Print) 1537-2715 (Online) Journal homepage: https://www.tandfonline.com/loi/ucgs20

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

Sonia Jain & Radford M Neal

To cite this article: Sonia Jain & Radford M Neal (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, DOI: 10.1198/1061860043001

To link to this article: https://doi.org/10.1198/1061860043001

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

Sonia JAIN and Radford M. NEAL

This article proposes 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 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: Gibbs sampler; Latent class analysis; Metropolis–Hastings algorithm.

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, Smith, and Makov (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; Gilks, Richardson, and Spiegelhalter 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. Computational techniques for Dirichlet process mixture models were explored previously by Escobar (1994), Escobar and West (1995), MacEachern (1994), Bush and MacEachern (1996), Neal (1992, 2000), and Green and Richardson (2001).

Sonia Jain is Assistant Professor, Division of Biostatistics, Department of Family and Preventive Medicine, University of California at San Diego, La Jolla, CA 92093-0717 (E-mail: sojain@ucsd.edu). Radford M. Neal is Professor, Department of Statistics and Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G3 (E-mail: radford@utstat.toronto.edu).

Journal of Computational and Graphical Statistics, Volume 13, Number 1, Pages 158–182
DOI: 10.1198/1061860043001

When conjugate priors are used, a Gibbs sampling procedure can easily be constructed for the Dirichlet process mixture model. Here, we fully exploit the conjugacy in the model by analytically integrating away the mixing proportions for the components and the parameters for each component. As a result, the Gibbs sampling procedure updates only 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, Hurn, and Robert (2000) attributed 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, because neighboring observations would support the formation of a new component if appropriate.

Split-merge updates were previously proposed by Green and Richardson (2001). They introduced a complex split-merge scheme in the reversible jump framework. The split-merge proposals are based on conserving specific moment conditions and are evaluated by a Metropolis acceptance probability. Green and Richardson provided general guidelines in constructing split proposals, but it is not clear whether adequate split proposals are simple to construct or compute in high-dimensional multivariate mixture problems.

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.

Section 2 introduces notation and terminology for the Dirichlet process mixture model and a Gibbs sampling algorithm suitable for conjugate priors. Section 3 presents our split-merge Metropolis-Hastings procedure and its variants. Section 4 empirically compares our split-merge method to the Gibbs sampler and demonstrates 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. GIBBS SAMPLING FOR THE DIRICHLET PROCESS MIXTURE MODEL

This section presents the Dirichlet process mixture model (for early references, see Ferguson 1983 and Antoniak 1974) and describes a Gibbs sampling algorithm to sample from the posterior of this model. This procedure completely uses the conjugacy in the model to integrate away model parameters and mixing proportions, eliminating them from the algorithm. Section 4 compares this version of the Gibbs sampler 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 modeled by 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:

$$\begin{aligned} y_i \mid \theta_i &\sim F(\theta_i) \\ \theta_i \mid G &\sim G \\ G &\sim \mathrm{DP}(G_0, \alpha) \end{aligned} \quad , \tag{2.1}$$

where G_0 defines a baseline distribution for the Dirichlet process prior, and α is a total mass parameter that takes values greater than zero. The usual conditional independence assumptions for a hierarchical model apply, so that the only dependencies are those that are explicitly shown. Equation (2.1) 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 (e.g., see MacEachern 1998).

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 (2.1), the θ_i follow a generalized Polya urn scheme (Blackwell and MacQueen 1973). The prior distribution for the θ_i may be represented in this way by the following conditional distributions:

$$\begin{aligned} \theta_1 &\sim G_0 \\ \theta_i \mid \theta_1, \ldots, \theta_{i-1} &\sim \frac{1}{i-1+\alpha} \sum_{j=1}^{i-1} \delta(\theta_j) \; + \; \frac{\alpha}{i-1+\alpha} G_0, \end{aligned} \tag{2.2}$$

where $\delta(\theta_j)$ is the distribution which is a point mass at θ_j. The model of Equation (2.1) has been simplified by integrating away the random distribution, G. We can represent the fact

that (2.2) 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

$$\begin{aligned} P(c_i = c \mid c_1, \ldots, c_{i-1}) &= \frac{n_{i,c}}{i-1+\alpha}, \quad \text{for} \quad c \in \{c_j\}_{j<i} \\ P(c_i \neq c_j \ \text{ for all } \ j < i \mid c_1, \ldots, c_{i-1}) &= \frac{\alpha}{i-1+\alpha}, \end{aligned} \tag{2.3}$$

where $n_{i,c}$ is the number of c_k for $k < i$ that are equal to c. The labeling of the indicator c_i is irrelevant in the above probabilities; all that matters is which c_i are equal to each other.

2.2 A GIBBS SAMPLING PROCEDURE

If G_0 is a conjugate prior for F, it is straightforward to sample from the posterior distribution of the above model 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 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 consists only of the class indicators, c_i, on a discrete state space.

The Markov chain is initialized by setting the c_i to some initial state. The c_i are then updated via Gibbs sampling by repeatedly drawing a new value for each c_i from its conditional distribution given the others, which is proportional to the product of its conditional prior and likelihood. Because the observations are exchangeable, the conditional prior can be derived from Equation (2.3) by considering observation i to be the last of the n observations. This yields the following conditional probabilities:

$$\begin{aligned} P(c_i = c \mid c_{-i}, \mathbf{y}) &= b\,\frac{n_{-i,c}}{n-1+\alpha} \int F(y_i, \phi)\, dH_{-i,c_j}(\phi), \\ &\qquad \text{for} \quad c \in \{c_j\}_{j<i} \\ P(c_i \neq c_j \ \text{ for all } \ j \neq i \mid c_{-i}, \mathbf{y}) &= b\,\frac{\alpha}{n-1+\alpha} \int F(y_i, \phi)\, dG_0(\phi), \end{aligned} \tag{2.4}$$

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 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 $\int F(y_i, \phi)\, dH_{-i,c}(\phi)$ can be analytically computed.

3. SPLIT-MERGE METROPOLIS-HASTINGS UPDATES

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 and slow movement between posterior modes when the data are not sufficient to determine whether one component or two is appropriate. Here, we introduce a nonincremental Markov chain sampling method based on the Metropolis-Hastings algorithm that avoids this problem. Our algorithm also splits or merges groups of data points, but avoids 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 is a form of the Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970). This 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:

$$a(x^*, x) \quad = \quad \min\left[1, \frac{q(x|x^*)}{q(x^*|x)} \frac{\pi(x^*)}{\pi(x)}\right]. \tag{3.1}$$

The next state will be set to this candidate state with probability $a(x^*, x)$. Otherwise, the new state is the same 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 among valid Metropolis-Hastings algorithms by randomly selecting a proposal distribution, $q(x^*|x)$. Note that when calculating the Metropolis-Hastings acceptance probability, the ratio $q(x|x^*)/q(x^*|x)$ may be calculated for the particular proposal distribution that was chosen, rather than 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.

Simple Random Split-Merge Procedure

1. Select two distinct observations, i and j, uniformly at random.
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, that is, $c_i = c_j$, then:
   (a) Propose a new assignment of data items to mixture components, denoted as $\mathbf{c}^{\text{split}}$, 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, $\mathbf{c}^{\text{split}}$, as follows:
      - Let c_i^{split} be a new component such that $c_i^{\text{split}} \notin \{c_1, \ldots, c_n\}$
      - Let $c_j^{\text{split}} = c_j$
      - 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 $c_k^{\text{split}} = c_k$

   (b) Evaluate the proposal in (a) by the Metropolis-Hastings acceptance probability $a(\mathbf{c}^{\text{split}}, \mathbf{c})$. If the proposal is accepted, $\mathbf{c}^{\text{split}}$ becomes the next state in the Markov chain. If the proposal is rejected, the original vector, $\mathbf{c}$, remains as the next state.
4. Otherwise, if i and j belong to different mixture components, that is, $c_i \neq c_j$, then:
   (a) Propose a new assignment of data items to mixture components, denoted as $\mathbf{c}^{\text{merge}}$, in which components, c_i and c_j, are combined into a single component. Each element of the proposal vector, $\mathbf{c}^{\text{merge}}$, is assigned as follows:

- Let $c_i^{\text{merge}} = c_j$
- Let $c_j^{\text{merge}} = c_j$
- For every observation $k \in S$, let $c_k^{\text{merge}} = c_j$
- For observations $k \notin S \cup \{i, j\}$, let $c_k^{\text{merge}} = c_k$

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

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 $\mathbf{c}^{\text{split}}$, $\mathbf{c}^{\text{merge}}$, and $\mathbf{c}$ 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.

The Metropolis-Hastings acceptance probability (Equation (3.1)) used in Steps 3 and 4 of this procedure takes the form

$$a(\mathbf{c}^*, \mathbf{c}) \quad = \quad \min\left[1,\ \frac{q(\mathbf{c}|\mathbf{c}^*)}{q(\mathbf{c}^*|\mathbf{c})}\,\frac{P(\mathbf{c}^*)}{P(\mathbf{c})}\,\frac{L(\mathbf{c}^*|\mathbf{y})}{L(\mathbf{c}|\mathbf{y})}\right], \tag{3.2}$$

where $\mathbf{c}^*$ is either the vector $\mathbf{c}^{\text{split}}$ or $\mathbf{c}^{\text{merge}}$ depending on the type of proposal. The posterior distribution, $\pi(\mathbf{c})$, in Equation (3.1) has been expanded into a product of its factors: the prior, $P(\mathbf{c})$, and the likelihood, $L(\mathbf{c}|\mathbf{y})$, where $\mathbf{y} = (y_1, \ldots, y_n)$ is the vector of observations. Note that factors not involving $\mathbf{c}$ may be ignored.

The prior distribution, $P(\mathbf{c})$, for the entire vector $\mathbf{c}$ will be a product over distinct $c \in \{c_1, \ldots, c_n\}$ of the factors presented in Equation (2.3). This product yields the prior distribution

$$P(\mathbf{c}) \quad = \quad \alpha^D\,\frac{\prod_{c\in\{c_1,\ldots,c_n\}}(n_c-1)!}{\prod_{k=1}^{n}(\alpha+k-1)}, \tag{3.3}$$

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

Notice that the ratio of the prior distributions in Equation (3.2) simplifies considerably because the denominator in Equation (3.3) will cancel, as well as factors in Equation (3.3) associated with components that are not directly involved in the Metropolis-Hastings update. For the split proposal, the prior distribution ratio reduces to

$$\frac{P(\mathbf{c}^{\text{split}})}{P(\mathbf{c})} \quad = \quad \alpha\,\frac{(n_{c_i^{\text{split}}}-1)!\,(n_{c_j^{\text{split}}}-1)!}{(n_{c_i}-1)!}, \tag{3.4}$$

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

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

$$\frac{P(\mathbf{c}^{\text{merge}})}{P(\mathbf{c})} = \frac{1}{\alpha}\frac{(n_{c_i^{\text{merge}}}-1)!}{(n_{c_i}-1)!\,(n_{c_j}-1)!}, \tag{3.5}$$

where $\mathbf{c}$ 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

$$L(\mathbf{c}|\mathbf{y}) = \prod_{k=1}^{n}\int F(y_k,\phi)\,dH_{k,c_k}(\phi), \tag{3.6}$$

where H_{k,c_k} is the posterior distribution of ϕ based on the prior G_0 and all observations y_g for which $g < k$ and $c_g = c_k$. 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, because no data from that mixture component precedes item k. Alternatively, $L(\mathbf{c}|\mathbf{y})$ may be expressed as a double product over components, c, and items, $k \in \{1,\ldots,n\}$, associated with each component

$$L(\mathbf{c}|\mathbf{y}) = \prod_{c=1}^{D}\prod_{k\,:\,c_k=c}\int F(y_k,\phi)\,dH_{k,c}(\phi), \tag{3.7}$$

where D is the number of distinct components.

Because factors involving items associated with components not directly involved in the split proposal will cancel, the ratio of likelihoods in Equation (3.2) reduces to

$$\frac{L(\mathbf{c}^{\text{split}}|\mathbf{y})}{L(\mathbf{c}|\mathbf{y})} = \frac{\displaystyle\prod_{k\,:\,c_k^{\text{split}}=c_i^{\text{split}}}\int F(y_k,\phi)\,dH_{k,c_i^{\text{split}}}(\phi)\prod_{k\,:\,c_k^{\text{split}}=c_j^{\text{split}}}\int F(y_k,\phi)\,dH_{k,c_j^{\text{split}}}(\phi)}{\displaystyle\prod_{k\,:\,c_k=c_i}\int F(y_k,\phi)\,dH_{k,c_i}(\phi).} \tag{3.8}$$

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

$$\frac{L(\mathbf{c}^{\text{merge}}|\mathbf{y})}{L(\mathbf{c}|\mathbf{y})} = \frac{\displaystyle\prod_{k\,:\,c_k^{\text{merge}}=c_i^{\text{merge}}}\int F(y_k,\phi)\,dH_{k,c_i^{\text{merge}}}(\phi)}{\displaystyle\prod_{k\,:\,c_k=c_i}\int F(y_k,\phi)\,dH_{k,c_i}(\phi)\prod_{k\,:\,c_k=c_j}\int F(y_k,\phi)\,dH_{k,c_j}(\phi).} \tag{3.9}$$

In the first step of this procedure, 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

$$q(\mathbf{c}^{\text{split}}|\mathbf{c}) \quad = \quad \left(\frac{1}{2}\right)^{n_{c_i^{\text{split}}}+n_{c_j^{\text{split}}}-2}. \tag{3.10}$$

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

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

$$q(\mathbf{c}^{\text{merge}}|\mathbf{c}) \quad = \quad 1, \tag{3.11}$$

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

It follows from Equations (3.10) and (3.11) that the appropriate ratio of transition probabilities for the split proposal is

$$\frac{q(\mathbf{c}|\mathbf{c}^{\text{split}})}{q(\mathbf{c}^{\text{split}}|\mathbf{c})} \quad = \quad \frac{1}{\left(\frac{1}{2}\right)^{n_{c_i^{\text{split}}}+n_{c_j^{\text{split}}}-2}}. \tag{3.12}$$

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

$$\frac{q(\mathbf{c}|\mathbf{c}^{\text{merge}})}{q(\mathbf{c}^{\text{merge}}|\mathbf{c})} \quad = \quad \left(\frac{1}{2}\right)^{n_{c_i}+n_{c_j}-2}. \tag{3.13}$$

Therefore, the resulting acceptance probability (3.2) for a split proposal is based on the product of Equations (3.4), (3.8), and (3.12). Likewise, the acceptance probability for a merge proposal is based on the product of Equations (3.5), (3.9), and (3.13). By employing the Hastings (1970) version of the Metropolis et al. (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 neighboring 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. Because 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 immediately prior to the Gibbs sampling scan is fixed. This pre-Gibbs state will be referred to as the *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 our earlier procedure 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, but the proposal probabilities can still be explicitly computed. Each fixed launch state for the Gibbs sampling scan defines a particular Metropolis-Hastings algorithm, all of which are valid, 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 *deterministic* manner. This configuration assigning observations to mixture components is the launch state, $\mathbf{c}^{\text{launch}}$. From this launch state, one restricted Gibbs sampling scan is conducted to reallocate the observations in S *randomly* between the two split components. The Gibbs sampling scan is restricted in that it is performed only 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

$$P(c_k \mid c_{-k},\, y_k) \;=\; \frac{n_{-k,c_k} \int F(y_k,\phi)\, dH_{-k,c_k}(\phi)}{n_{-k,c_i} \int F(y_k,\phi)\, dH_{-k,c_i}(\phi) \;+\; n_{-k,c_j} \int F(y_k,\phi)\, dH_{-k,c_j}(\phi)}. \tag{3.14}$$

To simplify notation, we refer to component indicators for the launch state as c_k in Equation (3.14). 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_g for $g \neq k$ in $S \cup \{i,j\}$, $n_{-k,c}$ is the number of c_g for $g \neq k$ 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_g such that $c_g = c$ where $g \in S \cup \{i,j\}$, for which $g \neq k$. 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 probability that $\mathbf{c}^{\text{split}}$

will be produced by a restricted Gibbs sampling scan starting from $\mathbf{c}^{\text{launch}}$ 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 (3.14). In our algorithm, this product is the Metropolis-Hastings proposal probability, $q(\mathbf{c}^{\text{split}}|\mathbf{c})$.

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 $q(\mathbf{c}^{\text{merge}}|\mathbf{c}) = 1$. However, to obtain the corresponding probability, $q(\mathbf{c}|\mathbf{c}^{\text{merge}})$, 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 (3.2). The prior and likelihood ratios in Equation (3.2) remain as shown in Section 3.2.

Because 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. We would like to improve the proposals further 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. We could, for example, choose the launch state uniformly at random. However, if only a single scan of Gibbs sampling is performed from a random launch state, 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 be a better reflection of the actual attributes of the data than is produced by a single scan of Gibbs sampling.

Split proposal probabilities are calculated in the same way as for the fixed launch state Gibbs sampling proposals (Equation (3.14)). For the merge proposal, to obtain $q(\mathbf{c}|\mathbf{c}^{\text{merge}})$, 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.

We could modify the algorithm 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. Below, the procedure for the restricted Gibbs sampling split-merge update with a random launch state is summarized.

Restricted Gibbs Sampling Split-Merge Procedure

1. Select two distinct observations, i and j, uniformly at random.
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, $\mathbf{c}^{\text{launch}}$, 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 $c_i^{\text{launch}} \notin \{c_1, \ldots, c_n\}$ and let $c_j^{\text{launch}} = c_j$. Otherwise, if $c_i \neq c_j$, then let $c_i^{\text{launch}} = c_i$ and $c_j^{\text{launch}} = c_j$. 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 $\mathbf{c}^{\text{launch}}$ by performing t intermediate restricted Gibbs sampling scans or some other type of Markov chain update.
4. If items i and j are in the same mixture component, that is, $c_i = c_j$, then:
   (a) Propose a new assignment of data items to mixture components, denoted as $\mathbf{c}^{\text{split}}$, 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, $\mathbf{c}^{\text{split}}$, as follows:
      - Let $c_i^{\text{split}} = c_i^{\text{launch}}$ (note that $c_i^{\text{launch}} \notin \{c_1, \ldots, c_n\}$)
      - Let $c_j^{\text{split}} = c_j^{\text{launch}}$ (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, $\mathbf{c}^{\text{launch}}$
      - For observations $k \notin S \cup \{i, j\}$, let $c_k^{\text{split}} = c_k$

(b) Calculate the proposal probability, $q(\mathbf{c}^{\text{split}}|\mathbf{c})$, by computing the Gibbs sampling transition probability from the launch state, $\mathbf{c}^{\text{launch}}$, to the final proposed state, $\mathbf{c}^{\text{split}}$. 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 $a(\mathbf{c}^{\text{split}}, \mathbf{c})$. If the proposal is accepted, $\mathbf{c}^{\text{split}}$ becomes the next state in the Markov chain. If the proposal is rejected, the original vector, $\mathbf{c}$, remains as the next state.

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

(a) Propose a new assignment of data items to mixture components, denoted as $\mathbf{c}^{\text{merge}}$, in which components, c_i and c_j, are combined into a single component. Assign each element of the proposal vector, $\mathbf{c}^{\text{merge}}$, as follows:

- Let $c_i^{\text{merge}} = c_j$
- Let $c_j^{\text{merge}} = c_j$
- For every observation $k \in S$, let $c_k^{\text{merge}} = c_j$
- For observations $k \notin S \cup \{i, j\}$, let $c_k^{\text{merge}} = c_k$

(b) Calculate the proposal probability, $q(\mathbf{c}|\mathbf{c}^{\text{merge}})$, by computing the Gibbs sampling transition probability from the launch state, $\mathbf{c}^{\text{launch}}$, to the original split configuration, $\mathbf{c}$. 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 $a(\mathbf{c}^{\text{merge}}, \mathbf{c})$. If the proposal is accepted, $\mathbf{c}^{\text{merge}}$ becomes the next state. If the merge proposal is rejected, the original configuration, $\mathbf{c}$, remains as the next state.

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 nonzero prior probability, there is a nonzero 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 degenerate models, the Markov chain is aperiodic, because there is a nonzero probability that the chain will remain in its current state (i.e., at least some split or merge proposals have a nonzero probability of being rejected). The split-merge algorithm is therefore ergodic.

Even though the above procedure is ergodic and produces nonincremental splits or merges of components, further improvements in convergence may be obtained by combin-

ing this algorithm with traditional Gibbs sampling. This procedure 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 (sec. 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 due to possible periodicity. However, in our case, since both the Metropolis-Hastings and Gibbs sampling steps have a nonzero 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. Section 4 examines the effects of these modifications and provides guidelines for setting these tuning parameters.

4. EXAMPLE: BERNOULLI DATA WITH A CONJUGATE BETA PRIOR

This section empirically compares the split-merge procedure (and its variants) to Gibbs sampling. We consider a categorical mixture model, in which the data, $\mathbf{y} = (y_1, \ldots, y_n)$, 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, e.g., 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 modeled as a Dirichlet process mixture model. The observations, $y_i = (y_{i1}, \ldots, y_{im})$, are multivariate Bernoulli, such that

each $y_{ih}|\theta_i \sim \text{Bernoulli}(\theta_{ih})$, giving the following likelihood:

$$F(y_i, \theta_i) = \prod_{h=1}^{m} \theta_{ih}^{y_{ih}} (1-\theta_{ih})^{1-y_{ih}}. \tag{4.1}$$

The parameter θ_{ih} of component i gives the probability that attribute h has the value one. Each such probability is given a Beta distribution prior with parameters $(\beta_{1,h}, \beta_{0,h})$. Under G_0, which is the prior over the vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$, 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

$$P(\theta) = \prod_{h=1}^{m} \left(\frac{\Gamma(\beta_{1,h}+\beta_{0,h})}{\Gamma(\beta_{1,h})\,\Gamma(\beta_{0,h})} \theta_h^{\beta_{1,h}-1} (1-\theta_h)^{\beta_{0,h}-1} \right), \tag{4.2}$$

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 (2.4)), which for this model is the following, when the Beta prior and Bernoulli likelihood are substituted

$$P(c_i = c \mid c_{-i},\, y_i) = b\,\frac{n_{-i,c}}{n-1+\alpha} \prod_{h=1}^{m} \frac{\sum_{k\neq i} \delta(c_k, c)\delta(y_{kh}, y_{ih}) + \beta_{y_{ih},h}}{n_{-i,c} + \beta_{0,h} + \beta_{1,h}}, \quad \text{for} \quad c \in \{c_j\}_{j\neq i} \tag{4.3}$$

$$P(c_i \neq c_j \text{ for all } j \neq i \mid c_{-i},\, y_i) = b\,\frac{\alpha}{n-1+\alpha} \prod_{h=1}^{m} \frac{\beta_{y_{ih},h}}{\beta_{0,h}+\beta_{1,h}}.$$

The delta function $\delta(x, y)$ is equal to one if $x = y$ and zero otherwise. The term $\sum_{k\neq i} \delta(c_k, c)\delta(y_{kh}, y_{ih})$ counts the number of observations associated with component c 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 (3.2), the prior is calculated as shown in Equation (3.3). The appropriate ratio of the transition probabilities based on restricted Gibbs sampling is obtained by using the first formula in Equation (4.3). The likelihood (Equation (3.7)) based on the Bernoulli-Beta model is as follows

$$L(\mathbf{c}|\mathbf{y}) = \prod_{c=1}^{D} \prod_{k:\, c_k = c} \prod_{h=1}^{m} \frac{\sum_{i<k} \delta(c_i, c)\delta(y_{ih}, y_{kh}) + \beta_{y_{kh},h}}{n_{k,c} + \beta_{0,h} + \beta_{1,h}}. \tag{4.4}$$

If we interchange the products over k and h, Equation (4.4) simplifies and can be computed as

$$L(\mathbf{c}|\mathbf{y}) = \prod_{c=1}^{D} \prod_{h=1}^{m} \frac{\left[\Gamma\left(\sum_k \delta(c_k, c)\delta(y_{kh}, 0) + \beta_{0,h}\right)/\Gamma(\beta_{0,h})\right] \left[\Gamma\left(\sum_k \delta(c_k, c)\delta(y_{kh}, 1) + \beta_{1,h}\right)/\Gamma(\beta_{1,h})\right]}{\Gamma(n_c + \beta_{0,h} + \beta_{1,h})/\Gamma(\beta_{0,h} + \beta_{1,h})} \tag{4.5}$$

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	0.95 0.95 0.95 0.95 0.95 0.95
2	0.2	0.05 0.05 0.05 0.05 0.95 0.95
3	0.2	0.95 0.05 0.05 0.95 0.95 0.95
4	0.2	0.05 0.05 0.05 0.05 0.05 0.05
5	0.2	0.95 0.95 0.95 0.95 0.05 0.05

When calculating the prior ratios and performing Gibbs sampling, counts of observations associated with each mixture component are required. 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. Similarly, when computing likelihoods (Equation (4.5)), efficiency is further improved by maintaining a count of items associated with each mixture component having particular values for each attribute.

4.2 THE SYNTHETIC DATASETS

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 synthetic 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 two simulated datasets, 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 between 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 two simulated examples were randomly generated from the mixture distributions shown in Tables 1 and 2. The mixture components are distinguished by the first four attributes, which for consistency, have been kept constant in both 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

Table 2. True Mixture Distribution for Example 2

c	$P(c_i = c)$	$P(y_{ih} = 1 \mid c_i = c)$, $h = 1,\ldots,18$																	
1	0.2	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
2	0.2	0.05	0.05	0.05	0.05	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
3	0.2	0.95	0.05	0.05	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
4	0.2	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
5	0.2	0.95	0.95	0.95	0.95	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

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 is likely to be small. The $\beta_{0,h}$ and $\beta_{1,h}$ 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.

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 2,000 iterations. All simulations were performed in Matlab, Version 5.3, on a SGI system with a 200 MHz MIPS processor.

The performance of these algorithms was evaluated by examining trace plots (Figures 1 and 2) and the computation time per iteration (Table 3). 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. Because 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.

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. All of the algorithms except for the Simple

Table 3. Time Per Iteration in Seconds for Algorithms Tested

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

Random Split appropriately separated the data into the five mixture components. We found that, as expected, Simple Random Split did not work well, failing to converge within the 2,000 iterations done. By inspection, Gibbs sampling, Split-Merge (0,1,1) and (5,1,1) have short burn-in times and mix equally well. Similarity in performance is also confirmed by approximately equal autocorrelation times.

In this simple problem, Gibbs sampling is successful in correctly splitting the items among 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

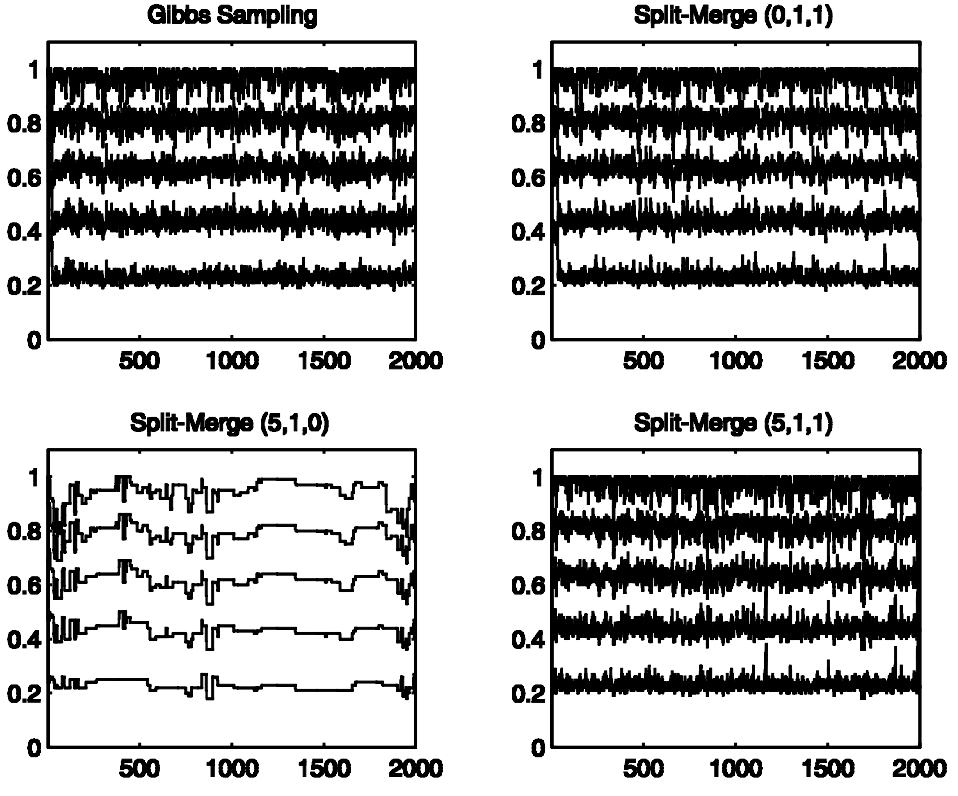

Figure 1. Trace plots of four of the six algorithms used in Example 1 with six attributes.

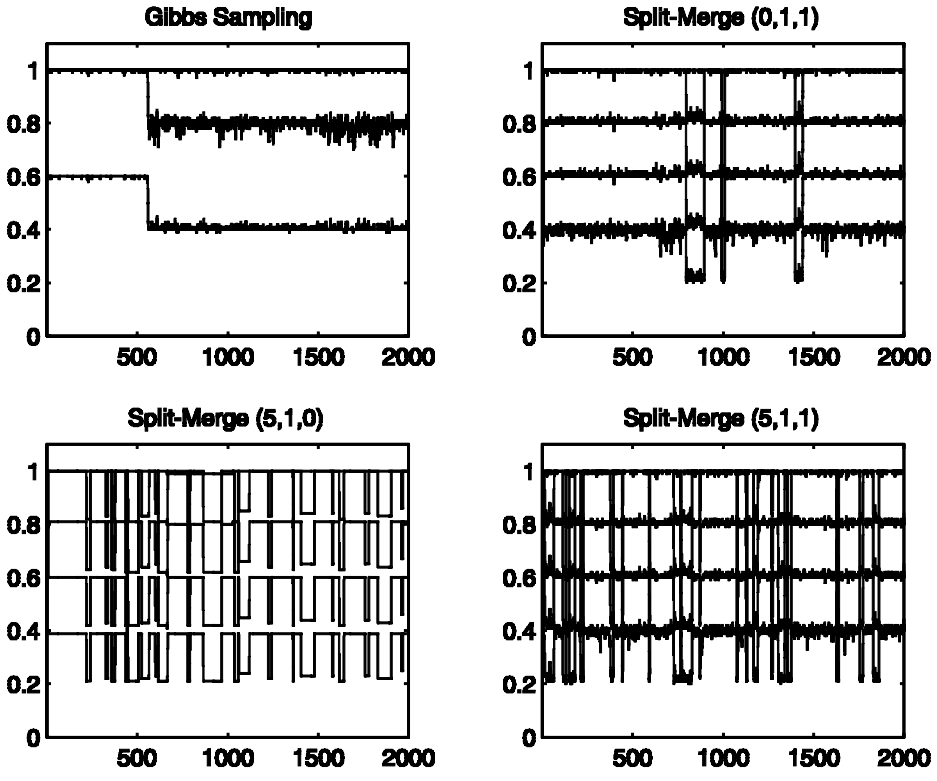

Figure 2. Trace plots of four of the six algorithms used in Example 2 with 18 attributes.

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, but mixing is poor.

4.3.2 Example 2

Example 2 is a high-dimensional problem (18 attributes), in which the posterior distribution, given the priors assigned, gives substantial probability to configurations with (mainly) four or five components. The trace plots (Figure 2) show that Gibbs sampling remains in an incorrect split that is not typical of the true posterior distribution for the entire 2,000 iteration run. This happens because the mixture components in this example are quite similar, so 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 1,000 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 immedi-

ately 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 nonoptimal Metropolis-Hastings split proposals. Again, the Simple Random Split was unable to separate the data adequately and performs the worst.

4.3.3 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

This section examines 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 examine the effect of varying each tuning parameter holding the other two parameters constant by considering data similar to Example 2, but with only 15 (instead of 18) attributes. The last three attributes in Table 2 have been discarded to speed up computations.

To examine the effect of adjusting the tuning parameters, 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, 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

Table 4. 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

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, sec. 6.3). Autocorrelation time was estimated by one plus twice the sum of the estimated autocorrelations up to the lag where the autocorrelations are approximately zero. Table 4 displays the computation time per iteration and autocorrelation times for trace 1 and indicator $I_{13,42}$ for various settings of this algorithm.

The most critical tuning parameter is the number of intermediate Gibbs sampling scans, since this controls the quality of the Metropolis-Hastings split proposals. 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 trace plots (not shown), 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 computation time per iteration (3.2 vs. 28.8 seconds). However, after five intermediate scans, the improvement is fairly minimal. With respect to the Metropolis-Hastings acceptance rate, 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 computation time.

Autocorrelation times decrease and mixing between four and five components improves when the number of Metropolis-Hastings updates is increased. However, these improve-

ments 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 computation time. Instead, it is better to perform a final Gibbs sampling scan after multiple Metropolis-Hastings updates, which are relatively faster.

Metropolis-Hastings updates need to be supplemented by some complete Gibbs sampling scans in order to make minor clustering changes (see Figures 1 and 2). Improvement 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.

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, allowing the posterior distribution to be fully explored. The quality of the proposals can be controlled by varying the number of intermediate Gibbs sampling scans. Implementing this method is relatively simple and does not become more difficult in higher dimensions. It is 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. We have implemented the split-merge algorithm for conjugate normal mixture models when the variance is known and observed similar improvements as described here.

The method by which the random selection of i and j determines a split or merge operation introduces an inefficiency into the algorithm. 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 should 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 labeling 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 for observations other than i and j may not overcome this initial bias, leaving i and j in the "wrong" mixture components.

A quick way to fix this nuisance labeling problem would be to perform a Metropolis update before reaching the launch state that evaluates a proposal to swap the labels of the items associated with each split component (equivalent to switching the component indicators of i and j). As mentioned in Section 3.3.2, any valid Markov chain update can be used to arrive at a launch state. We recommend use of the swap proposal immediately after the intermediate Gibbs sampling scans, so that this is the last step to reaching a particular launch state. Empirical trials show that the addition of the swap proposal will, as expected, improve the overall Metropolis-Hastings acceptance rate by up to a factor of two.

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.

Recently, we extended the split-merge algorithm to handle nonconjugate mixture models, in which the model parameters cannot be analytically integrated away (Jain 2002). This version of our split-merge algorithm also performs better than Gibbs sampling in situations where mixture components are similar. Note that when the model is conjugate, the algorithm that we have described in this article is both simpler and more efficient than the nonconjugate version of our method. We plan to discuss the nonconjugate technique in a future article.

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 multidimensional 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 prob-

ability to make this a valid Markov chain update, as in the algorithms we have presented in this article.

ACKNOWLEDGMENTS

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.

[Received August 2001. Revised October 2002.]

REFERENCES

Antoniak, C. E. (1974), "Mixtures of Dirichlet Processes With Applications to Bayesian Nonparametric Problems," *The Annals of Statistics*, 2, 1152–1174.

Blackwell, D., and MacQueen, J. B. (1973), "Ferguson Distributions via Pólya Urn Schemes," *The Annals of Statistics*, 1, 353–355.

Bush, C. A., and MacEachern, S. N. (1996), "A Semiparametric Bayesian Model for Randomised Block Designs," *Biometrika*, 83, 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*, 95, 957–970.

Escobar, M. D. (1994), "Estimating Normal Means With a Dirichlet Process Prior," *Journal of the American Statistical Association*, 89, 268–277.

Escobar, M. D., and West, M. (1995), "Bayesian Density Estimation and Inference Using Mixtures," *Journal of the American Statistical Association*, 90, 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. (1983), "Bayesian Density Estimation by Mixtures of Normal Distributions," in *Recent Advances in Statistics*, eds. H. Rizvi and J. Rustagi, New York: Academic Press, pp. 287–303.

Gelman, A., and Meng, X.-L. (1998), "Simulating Normalizing Constants: From Importance Sampling to Bridge Sampling to Path Sampling," *Statistical Science*, 13, 163–185.

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

Green, P. J., and Richardson, S. (2001), "Modelling Heterogeneity With and Without the Dirichlet Process," *Scandinavian Journal of Statistics*, 28, 355–375.

Hastings, W. K. (1970), "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, 57, 97–109.

Jain, S. (2002), "Split-Merge Techniques for Bayesian Mixture Models," unpublished Ph.D. dissertation, University of Toronto, Department of Statistics.

MacEachern, S. N. (1994), "Estimating Normal Means With a Conjugate Style Dirichlet Process Prior," *Communications in Statistics: Simulation and Computation*, 23, 727–741.

——— (1998), "Computational Methods for Mixture of Dirichlet Process Models," in *Practical Nonparametric and Semiparametric Bayesian Statistics*, eds. D. Dey et al., New York: Springer-Verlag, pp. 23–43.

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*, 21, 1087–1092.

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

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

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

Tierney, L. (1994), "Markov Chains for Exploring Posterior Distributions" (with discussion), *The Annals of Statistics*, 22, 1701–1762.

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