Statistics and Computing **14**: 11–21, 2004

Particle filters for mixture models with an unknown number of components

PAUL FEARNHEAD

Department of Mathematics and Statistics, Lancaster University, Lancaster, LA1 4YF

Received April 2002 and accepted July 2003

We consider the analysis of data under mixture models where the number of components in the mixture is unknown. We concentrate on mixture Dirichlet process models, and in particular we consider such models under conjugate priors. This conjugacy enables us to integrate out many of the parameters in the model, and to discretize the posterior distribution. Particle filters are particularly well suited to such discrete problems, and we propose the use of the particle filter of Fearnhead and Clifford for this problem. The performance of this particle filter, when analyzing both simulated and real data from a Gaussian mixture model, is uniformly better than the particle filter algorithm of Chen and Liu. In many situations it outperforms a Gibbs Sampler. We also show how models without the required amount of conjugacy can be efficiently analyzed by the same particle filter algorithm.

Keywords: Dirichlet process, Gaussian mixture models, Gibbs sampling, MCMC, particle filters

1. Introduction

Monte Carlo methods are commonly used tools for Bayesian inference. Markov chain Monte Carlo (MCMC; see Gilks *et al.* 1996, for an introduction) is often the method of choice for batch problems, where interest is in the joint posterior distribution of static parameters conditional on a single batch of observations. However, MCMC is inefficient for real-time, dynamic problems such as target-tracking. In these dynamic problems the interest is in the value of an unobserved, time-varying state, and new observations are obtained through time in order to make inference about the value of this state at different time points. As each new observation is made, there is a new posterior distribution of interest, and it is often computationally infeasible to run an MCMC sampler for sufficiently long to obtain a representative sample from this new posterior. As a result of this problem with MCMC, there has been much recent research into sequential Monte Carlo methods, known as particle filters. A particle filter approximates each posterior distribution by a swarm of weighted particles, which are updated sequentially as each new observation is obtained. The first particle filters were suggested by West (1992) and Gordon, Salmond and Smith (1993), and there is a vast literature on how to implement particle filters efficiently (see for example Liu and Chen 1998, Pitt and Shephard 1999, Carpenter, Clifford and Fearnhead 1999, Gilks and Berzuini 2001, Fearnhead 2003). A good introduction to particle filters, which includes examples of their use in practice, is given by Doucet, de Freitas and Gordon (2001).

Due to the success of particle filters for dynamic problems, there is current interest in applying them to batch problems as a competitive alternative to MCMC (e.g. Chopin 2002). In this paper we use a particle filter to analyse data from mixture models. While Chopin (2002) gives an example of a particle filter being applied to a Gaussian mixture with a known number of components, we consider mixture of Dirichlet process (MDP) models (see Antoniak 1974), where the number of components (or clusters) is unknown. A simple view of these models is that each observation can be assigned to a "cluster", with all observations from a given cluster being independently drawn from the same distribution. Observations from different clusters are drawn independently from different distributions, and the number of clusters in this model is unknown. These models have been used for nonparametric and semiparametric Bayesian analysis (Escobar 1994, Bush and MacEachern 1996), Bayesian density estimation (Escobar and West 1995, Muller, Erkanli and West 1996) and generalized linear models (Mukhopadhyay and Gelfand 1997).

We consider a subset of MDP models where:

(C1) conditional on the assignment of observations to clusters, the joint posterior probabilty of all unknown parameters can be calculated analytically.

(C2) the posterior probability of any assignment of the first n observations to clusters can be calculated analytically (up to a normalising constant).

(These two conditions are closely related, as if the integration required for (C1) is tractable, then so will the integration required for (C2).) For models which satisfy (C1) and (C2), the posterior distribution can be written explicitly as a mixture distribution. Each term in the mixture relates to a different assignment of observations to clusters. (The intractability of the inference problem is due to the number of terms in this mixture increasing exponentially with the number of observations).

The general particle filter approach to this problem is well established (see Quintana 1998, MacEachern, Clyde and Liu 1999, Chen and Liu 2000), with each particle consisting of a specific assignment of observations to clusters. The key to the efficiency of a specific particle filter is how the particles are updated as each new observation is processed. For each possible assignment of the first n observations to clusters, with say k distinct clusters, there are $k+1$ possible assignments of the $(n+1)$st observation to a cluster (it can be assigned to any of the existing clusters, or to a new cluster). Thus given N particles for the first n observations, there will be $M(\geq 2N)$ possible particles for the first $(n+1)$ observations. Particle filters differ in how they "resample" these M possible particles to produce N particles for the first $(n+1)$ observations. (This is necessary to stop the cost of processing each observation from increasing exponentially.) Numerous resampling algorithms exist (for example Akashi and Kumamoto 1977, Tugnait 1982), but currently the standard method is that of Chen and Liu (2000). Instead of using this resampling method, we propose using the resampling algorithm of Fearnhead and Clifford (2003). This algorithm propagates a subset of the M particles which have largest weight, and uses stratified resampling (Carpenter, Clifford and Fearnhead 1999) to resample from the remaining particles. This resampling is unbiased, and optimum in the sense of minimizing some mean square error loss (see Section 4 for more details). In Fearnhead and Clifford (2003) it was shown to be between one and two orders of magnitude more efficient than the method of Chen and Liu (2000) for analyzing a change-point model. We discuss the comparitive efficiencies of the particle filter and Gibbs sampler methods in detail in Section 5.

We illustrate our method using a Gaussian mixture model. In this model each observation is drawn from a Gaussian density with an unknown mean and variance that varies between clusters. We assume conjugate priors for the unknown parameters, so that conditions (C1) and (C2) hold. We use this model in our simulation studies, where we evaluate the performance of our particle filter. We compare our particle filter to both a particle filter using the resampling scheme of Chen and Liu (2000) and a Gibbs sampler. In general, our method is the more efficient particle filter method. In fact for some problems our particle filter is "super-efficient", in the sense that inference based on N weighted particles is more accurate than inference based on an independent sample of size N from the posterior distribution of assignments of observations to clusters. It also out performs the Gibbs sampler on many of the data sets we consider. In particular the particle filter performs well when analyzing the smaller data sets, and data from models where each observation is informative about the corresponding cluster's parameters (relative to the prior for these parameters).

In practice it is possible to weaken the conditions (C1) and (C2) that we imposed on our MDP models. In this case, any intractable integration needs to be computed using either Monte Carlo or numerical integration. We illustrate the practicality of this by considering the Gaussian mixture model under non-conjugate priors the clusters' means and variances. When analyzing the galaxy data of Postman, Huchra and Geller (1986), for a fixed number of particles and using a simple Monte Carlo integration scheme, the particle filter was about half as accurate as the particle filter when analyzing the same data under a model with conjugate priors. See Section 4.3.

2. Mixture Dirichlet process models

Mixture Dirichlet process (MDP) models were introduced by Ferguson (1973). They are commonly used in Bayesian nonparametric and Bayesian semi-parametric methods. One motivation behind such methods is to pool information across a number of similar experiments. Another application of MDP models is for Bayesian density estimation. See the introduction for references which describe the use of MDP models in practice.

Blackwell and MacQueen (1973) introduced a Polya-Urn formulation of MDP models, which we use in the following model formulation. Consider data $y_{1:n} = (y_1, \ldots, y_n)$ from n experiments Under the MDP model each y_i belongs to a cluster. For any given cluster, all observations belonging to that cluster are independent draws from the same distribution. Let an assigment $z_{1:n} = (z_1, \ldots, z_n)$ be a vector of cluster labels, and k (which is a function of $z_{1:n}$) be the number of distinct clusters in the assignment $z_{1:n}$. Then $z_i \in \{1, \ldots, k\}$ for all $i = 1, \ldots, n$, and each cluster contains at least one observation.

We will assume that each observation is drawn from a member of a family of distributions, which is parameterized by θ. Let $f(y;\theta)$ denote the density of y for a given value of θ. (There will be a different value of θ for each cluster.)

The probability distribution for $y_{1:n}$ under a MDP model has a hierarchical form. Firstly we have a Dirichlet-based prior for $z_{1:n}$, $\pi(z_{1:n})$. Then conditional on there being k clusters under our assignment, we have k parameters $\theta_{1:k} = (\theta_1, \ldots, \theta_k)$, which are independently drawn from a known prior $\pi(\theta)$. Finally, conditonal on $z_{1:n}$ and $\theta_{1:k}$, observation y_i is drawn independently from $f(y;\theta_{z_i})$. Thus we have the following joint density

$$p(y_{1:n}, z_{1:n}, \theta_{1:k}) \propto \pi(z_{1:n}) \prod_{j=1}^{k} \pi(\theta_j) \prod_{i=1}^{n} f(y_i;\theta_{z_i}). \quad (1)$$

The prior for $z_{1:n}$, is parameterized by α, and can be defined recursively by

$$p(z_{i+1} = j \mid z_{1:i}) = \begin{cases} n_j/(i+\alpha) & \text{for } j = 1, \ldots, k_i \\ \alpha/(i+\alpha) & j = k_i + 1 \end{cases}, \quad (2)$$

where k_i is the number of clusters in the assignment $z_{1:i}$, and n_j is the number of observations that $z_{1:i}$ assigns to cluster j.

In this paper, we mainly consider a special class of MDP models. We will assume that:

(C1) conditional on knowning the assignment, the joint posterior probabilty of all unknown parameters can be calculated analytically.

(C2) the posterior probability of any assignment of the n observations to clusters can be calculated analytically (up to a normalising constant).

A sufficient condition for both (C1) and (C2) to hold is that the integral of (1) with respect to $\theta_{1:k}$ is tractable. The structure of the MDP model is such that conditional on the assignment, the parameters θ_i and θ_j are independent for $i \neq j$. Thus (C1) and (C2) will hold providing a conjugate prior for θ is used (for example see the multinomial-Beta model of Quintana and Newton 2000, and the example below).

The analysis of MDP models simplifies if conditions (C1) and (C2) hold. In particular we can write

$$p(z_{1:n}, \theta_{1:k} \mid y_{1:n}) = \sum_{z_{1:n}} p(z_{1:n} \mid y_{1:n}) p(\theta_{1:k} \mid y_{1:n}, z_{1:n}), \quad (3)$$

where the $p(\theta_{1:k} \mid y_{1:n}, z_{1:n})$ terms can be calculated analytically, and the $p(z_{1:n} \mid y_{1:n})$ terms can be calculated up to a constant of proportionality. In practice it is not possible to work with (3) as the number of terms in the sum increases exponentially with n, and the analysis of MDP models, even when conditions (C1) and (C2) hold, is still very challenging. As we will see, the usefulness of (3) is that an approximation to $p(z_{1:n}, \theta_{1:k} \mid y_{1:n})$ can be obtained from an approximation to $p(z_{1:n} \mid y_{1:n})$. The latter distribution is over a (discrete) space of smaller dimension than the former, and is substantially easier to approximate using particle filters.

One example of an MDP models satisfying these conditions is the following Gaussian mixture model.

Example 1 (Gaussian mixture model). Each observation is drawn from a univariate Gaussian distribution with unknown mean and variance. For this model $\theta = (\mu, \sigma^2)$, the mean and variance of the Gaussian distribution. We assume an inverse gamma prior for σ^2; so that if $s = 1/\sigma^2$ then

$$\pi(s) = s^{a-1} b^a \exp\{-bs\} / \Gamma(a),$$

where the shape parameter, a, and the scale parameter, b, are known, and $\Gamma(a)$ is the gamma function. Conditional on s, μ has a normal prior distribution with mean η and variance τ/s.

This is the standard conjugate prior (see, for example West and Harrison 1997, Sections 2.5 and 17.3). Conditional on an assignment $z_{1:n}$, the posterior distribution of s_j and μ_j (the parameters for the jth cluster) depend on n_j the number of observations in the jth cluster, $\bar{y}_j$ the mean of these observations, and $\hat{\sigma}_j^2$ the variance of these observations.

We get that $p(s_j \mid n_j, \bar{y}_j, \hat{\sigma}_j^2)$ is a Gamma density with shape parameter $a + n_j/2$ and scale parameter $b + n_j(\hat{\sigma}_j^2 + (\bar{y}_j - \eta)^2/(1+n_j\tau))/2$. While $p(\mu_j \mid n_j, \bar{y}_j, \hat{\sigma}_j^2, s_j)$ is a Gaussian density with mean $(\eta + n_j\tau\bar{y}_j)/(1+n_j\tau)$ and variance $\tau/(s_j + s_j n_j \tau)$.

Furthermore, the posterior probability of assignment $z_{1:n}$ is proportional to

$$\pi(z_{1:n}) \prod_{j=1}^{k} \left\{ \frac{b^a \Gamma(a + n_j/2)}{\Gamma(a)\sqrt{n_j\tau + 1}} \big(b + n_j\big[\hat{\sigma}_j^2 + (\bar{y}_j - \eta)^2 \big/ (1 + n_j\tau)\big]/2\big)^{-(a+n_j/2)} \right\}.$$

3. Particle filter approach

Particle filters were developed for analysing dynamic problems in real-time. For example, consider inference about an unobserved state which varies over time. Inference is based on a set of observations made at discrete points through time. We let y_n denote the observation at the nth time point, and x_n denote the value of the state at that time point. We assume a prior for x_1, and a model which specifies $p(x_{n+1} \mid x_n)$ and $p(y_n \mid x_n)$. Interest is often in the posterior distributions $p(x_n \mid y_{1:n})$ for all time points $n = 1, 2, \ldots$.

For a given time point n, a particle filter will approximate the posterior distribution $p(x_n \mid y_{1:n})$ by a set of N weighted particles. The particles, $\{x_n^{(i)}\}_{i=1}^N$, are just potential realisations of the state. Each particle is assigned a weight, and we denote the set of weights by $\{w_n^{(i)}\}_{i=1}^N$, and assume the set of weights has been normalised to sum to 1. The posterior distribution, $p(x_n \mid y_{1:n})$, is approximated by a discrete distribution with support $\{x_n^{(i)}\}_{i=1}^N$, and which, for $i = 1, \ldots, N$, assigns weight $w_n^{(i)}$ to the value $x_n^{(i)}$. Thus for any function of the state, $g(x_n)$ we get the approximation

$$\mathrm{E}(g(X_n) \mid y_{1:n}) \approx \sum_{i=1}^{N} w_n^{(i)} g\big(x_n^{(i)}\big). \quad (4)$$

Thus given a set of particles at time n, $\{x_n^{(i)}\}_{i=1}^N$, and their normalised weights $\{w_n^{(i)}\}_{i=1}^N$, we can approximate both the prior at time $n+1$:

$$\hat{p}(x_{n+1} \mid y_{1:n}) = \sum_{i=1}^{N} w_n^{(i)} p\big(x_{n+1} \mid x_n^{(i)}\big); \quad (5)$$

and the posterior at time $n+1$:

$$\hat{p}(x_{n+1} \mid y_{1:n+1}) \propto \sum_{i=1}^{N} w_n^{(i)} p\big(x_{n+1} \mid x_n^{(i)}\big) p(y_{n+1} \mid x_{n+1}). \quad (6)$$

In particle filters, on receipt of the latest observation y_{n+1}, the particles $\{x_n^{(i)}\}_{i=1}^N$, and weights $\{w_n^{(i)}\}_{i=1}^N$, are updated to approximate (6). The simplest method to do this is to use importance

sampling with (5) as the proposal density (Gordon, Salmond and Smith 1993). This involves simulating the new particles, $\{x_{n+1}^{(i)}\}_{i=1}^N$, from (5) and assigning particle $x_{n+1}^{(i)}$ a weight proportional to $p(y_{n+1} \mid x_{n+1}^{(i)})$. More efficient algorithms exist; for example using a better proposal density (Pitt and Shephard 1999), using stratified sampling to simulate from (5) (Carpenter, Clifford and Fearnhead 1999), or MCMC (Gilks and Berzuini 2001, Fearnhead 2003). A review of particle filters is given by Liu and Chen (1998) and Doucet, de Freitas and Gordon (2001). All (sensible) particle filters satisfy convergence results which give convergence in probability of estimates of the posterior mean of (well-behaved) functions of the state (see equation (4)) as the number of particles tends to infinity (see Crisan and Doucet 2000). Under certain regularity conditions a central limit theorem also holds (Del Moral and Guionnet 1999).

For our problem we introduce artificial time, so that observation y_n is thought to occur at time n. For general MDP models our choice of state would be $(z_{1:n}, \theta_{1:k})$. However for MDP models satisfying our conditions (C1) and (C2) we can choose our state to be just $z_{1:n}$. Given a set of particles, $\{z_{1:n}^{(i)}\}_{i=1}^N$, and associated weights, $\{w_n^{(i)}\}_{i=1}^N$, we can estimate the full posterior by

$$p(z_{1:n}, \theta_{1:k} \mid y_{1:n}) \approx \sum_{i=1}^{N} w_n^{(i)} p\big(\theta_{1:k}, z_{1:n} \,\big|\, z_{1:n}^{(i)}, y_{1:n}\big).$$

The densities in the sum on the right-hand side of this equation can be calculated explicitly due to assumption (C1). (Remember that, although it is not explicit in our notation, k is a function of $z_{1:n}$). Furthermore, posterior expectations of functions of $z_{1:n}$ and $\theta_{1:k}$ can be approximated by

$$\mathrm{E}(g(z_{1:n}, \theta_{1:k}) \mid y_{1:n}) \approx \sum_{i=1}^{N} w_n^{(i)} \mathrm{E}\big(g\big(z_{1:n}^{(i)}, \theta_{1:k}\big) \,\big|\, z_{1:n}^{(i)}, y_{1:n}\big).$$

The expecations on the right-hand side of this equation can either be evaluated analytically or approximated by simulation. See Quintana and Newton (2000) for examples of this in practice.

The idea of reducing the dimension of the state via integration (in our case by integrating out the parameters $\theta_{1:k}$) has been termed collapsing (Liu and Chen 1998, see also Chen and Liu 2000) or marginalising (Liu, Chen and Logvinenko 2001), and can result in a substantial increase in efficiency of the particle filter. For a Binomial mixture model, Liu (1996b) implements a particle filter with state $(z_{1:n}, \theta_{1:k})$ while MacEachern, Clyde and Liu (1999) implement a marginalised particle filter with state $z_{1:n}$. The latter is up to an order of magnitude more efficient that the former (see MacEachern, Clyde and Liu 1999). Hence we consider only implementing the marginalised particle filter in the following.

Given a set of N particles which approximate $p(z_{1:n} \mid y_{1:n})$ we can obtain a discrete approximation of $p(z_{1:n+1} \mid y_{1:n+1})$ in a way analagous to (6). For any given particle $z_{1:n}^{(i)}$, with k_i distinct clusters, there are $k_i + 1$ possible allocations for the $(n+1)$st observation. Thus our discrete approximation to $p(z_{1:n+1} \mid y_{1:n+1})$ has support $(z_{1:n}^{(i)}, j)$ for $j = 1, \ldots, k_i + 1$ and $i = 1, \ldots, N$. The probability mass assigned to point $(z_{1:n}^{(i)}, j)$ is proportional to

$$w_n^{(i)} \frac{p\big(\big(z_{1:n}^{(i)}, j\big) \,\big|\, y_{1:n+1}\big)}{p\big(z_{1:n}^{(i)} \,\big|\, y_{1:n}\big)},$$

which can be evaluated (up to a common normalizing constant) under out assumption of condition (C2).

We can view this discrete approximation as an approximation based on $M = \sum_{i=1}^{N}(k_i + 1)$ particles each with some associated weight. We denote these "putative" particles (upon renumbering) by $\{z_{1:n+1}^{*(i)}\}_{i=1,\ldots,M}$, and their weights by $\{w_{n+1}^{*(i)}\}_{i=1,\ldots,M}$. In order to limit the computational cost of the particle filter algorithm we need to reduce this set of M weight particles to a set of N weighted particles, and we do this via resampling. Various possible resampling strategies exist, and we discuss the possibilities next.

3.1. *Resampling algorithms*

As stated above, the basic idea of the particle filter is to sequentially update the particles, commonly by using importance sampling to generate a weighted sample from (6), using (5) as a proposal density. The resampling stage of these algorithms is used when generating a sample from the proposal density. Generating independent samples from (5) is equivalent to multinomial resampling of the particles $\{x_n^{(i)}\}_{i=1}^N$ (see for example Gordon, Salmond and Smith 1993); so that if K_i is the number of times particles $x_n^{(i)}$ is resampled then

$$(K_1, \ldots, K_N) \sim \mathrm{Multinomial}\big(N, w_n^{(1)}, \ldots, w_n^{(N)}\big).$$

This produces a set of resampled particles $\{x_n^{*(j)}\}_{j=1}^N$ such that, for $i = 1, \ldots, N$, precisely K_i of these resampled particles are replicates of $x_n^{(i)}$. The particles at time $n+1$ are then obtained by drawing, for $j = 1, \ldots, N$, $x_{n+1}^{(j)}$ independently from $p(x_{n+1} \mid x_n^{*(j)})$. (It is straightforward to show this is equivalent to drawing an independent, identically distributed sample from equation (5)). Improvements on this strategy exist, such as residual sampling (Liu and Chen 1998) and stratified sampling (Carpenter, Clifford and Fearnhead 1999), which aim to reduce the variance of each K_i.

While such resampling algorithms are standard practice, there are two features of our problem that means that these algorithms are inefficient. These features are that the state-space is discrete, and that there are sufficiently few distinct descendants of each current particle that it is feasible to calculate the weights of all possible putative particles at the next time-step. As a result, there is no advantage of having multiple copies of particles.

To illustrate this consider a simple example. The set of five particles $\{1, 1, 1, 2, 2\}$ with weights $\{0.3, 0.1, 0.2, 0.3, 0.1\}$ is equivalent to the following set of two particles, $\{1, 2\}$ with weights $\{0.6, 0.4\}$. (In a similar way all the information in any set of N weighted particles which contains only D distinct values can be conveyed in a set of just D weighted particles.) Furthermore there is no advantage in having multiple copies of particles

(e.g. 3 copies of particle 1) in terms of being able to explore more fully the future values of particles which currently have large weights (this is the main reason for resampling and having multiple copies of particles in more standard applications of particle filters). Even if we store just one copy of each distinct particle, all their possible descendants at the next time-step will be considered. Thus the set of putative particles of the two current particles, {1, 2}, will be as rich as the set of putative particles of the five current particles, {1, 1, 1, 2, 2}. Although only a single copy of each particle is kept, multiple descendants of any particle may appear in the resampled particles at the next time-step. Thus duplicated particle values in a set of particles is wasteful.

Here we consider two resampling algorithms specifically designed for the type of problem that we are considering, and which avoid wasteful duplication of particle values. We outline each briefly below. In order to help the discussion of these resampling algorithms it is useful to think about the resampling algorithms in the following way. Using the notation of the previous section, prior to resampling the particle filter has a set of $M(>N)$ putative particles, $\{z_{1:n+1}^{*(i)}\}_{i=1}^{M}$, with associated weights $\{w_{n+1}^{*(i)}\}_{i=1}^{M}$. A resampling algorithm then produces a new set of weights $\{w_{n+1}^{(i)}\}_{i=1}^{M}$, (which is a realisation of some vector-valued random variable $\{W_{n+1}^{(i)}\}_{i=1}^{M}$) where precisely N of these new weights are non-zero. (To reduce the number of particles to N as required, all those putative particles whose new weight is zero are removed.)

We define such a resampling algorithm to be unbiased if $\mathrm{E}(W_{n+1}^{(i)}) = w_{n+1}^{*(i)}$ for all $i = 1, \ldots, M$. If the resampling algorithm is unbiased then for any suitable function g,

$$\mathrm{E}\left(\sum_{i=1}^{M} W_{n+1}^{(i)} g\big(z_{1:n+1}^{*(i)}\big)\right) = \sum_{i=1}^{M} w_{n+1}^{*(i)} g\big(z_{1:n+1}^{*(i)}\big).$$

So the expected value of any function of the particles after resampling is equal to its value prior to resampling. Both the following resampling algorithms are unbiased.

The first resampling algorithm was devised by Chen and Liu (2000). Given N particles at time n, $\{z_{1:n}^{(i)}\}_{i=1,\ldots,N}$, with weights $\{w_n^{(i)}\}_{i=1,\ldots,N}$, the new particles at time $n+1$ are obtained by

$$z_{1:n+1}^{(i)} = \big(z_{1:n}^{(i)}, j\big), \text{ for } i = 1, \ldots, N,$$

where j is drawn from a discrete distribution with support $\{1, \ldots, k_i + 1\}$, and a probability mass proportional to $p((z_{1:n}^{(i)}, j) \mid y_{1:n+1})$ assigned to value j. Regardless of the choice of j, the weight assigned to particle $z_{1:n+1}^{(i)}$ is proportional to

$$w_n^{(i)} \sum_{j=1}^{k_i+1} p\big(\big(z_{1:n}^{(i)}, j\big) \,\big|\, y_{1:n+1}\big) \big/ p\big(z_{1:n}^{(i)} \,\big|\, y_{1:n}\big).$$

After calculating the set of new particles and their weights, a further resampling (called rejuvenation) occurs if the sum of the square of the (normalised) weights exceeds some abitrarily chosen constant. This rejuvenation consists of independent draws from the discrete distribution that places probability mass $w_{n+1}^{(i)}$ on point $z_{1:n+1}^{(i)}$.

Without the rejuvenation stage, this method is equivalent to the sequential imputation scheme method of Kong, Liu and Wong (1994) (see also Akashi and Kumamoto 1977). The drawback with sequential imputation is that the weights can become very skewed, with a large number of particles having negligible weight. Rejuvenation allows such particles to be removed and replaced by extra copies of particles with larger weights. (While this temporarily allows duplicated values in our set of particles, this duplication vanishes as each particle is then propogated forward.) See Quintana and Newton (2000) for an example of how rejuvenation can increase the efficiency of the particle filter over that of sequential imputation.

The second resampling algorithm was devised by Fearnhead and Clifford (2003), and is the resampling procedure that we propose using for analyzing data from the models we are considering. The basic idea is to keep all of the putative particles whose weight is greater than some threshold, $1/c$, and resample from the remaining putative particles (the stratified sampling algorithm of Carpenter, Clifford and Fearnhead 1999, is used, which ensures no putative particle is resampled more than once). This threshold is chosen so that the resulting resampling algorithm is optimum, over all unbiased resampling algorithms (which includes the resampling algorithm of Chen and Liu 2000), in terms of minimising

$$\sum_{i=1}^{M} \mathrm{E}\big(\big(W_{n+1}^{(i)} - w_{n+1}^{*(i)}\big)^2\big), \tag{7}$$

a measure of the variability introduced through the resampling (see Fearnhead and Clifford 2003, for the proof). The reciprocal of the threshold, c, is the unique solution of

$$\sum_{i=1}^{M} \min\big\{c w_{n+1}^{*(i)}, 1\big\} = N.$$

The weight of resampled particles is set to $1/c$, and the weight of the propagated particles is kept unchanged. (There are similarities between this resampling appraoch, and the rejection control idea of Liu, Chen and Wong 1996, both methods employ a cutoff, with particles whose weight is larger than this cutoff being automatically kept.) For the changepoint problem analyzed in Fearnhead and Clifford (2003), a particle filter using this resampling algorithm was nearly two orders of magnitude more efficient than one using the resampling algorithm of Chen and Liu (2000). The following Section contains a comparison of particle filters based on these two resampling methods for the MDP model.

4. Results

The aim of our simulation study is to see whether particle filters offer a competitive alternative to MCMC for mixture models, and what characterizes the problems where this occurs. Similar

comparisons have previoulsly been done; for example, a comparison of a particle filter (using the resamplig algorithm of Chen and Liu 2000) and a Gibbs sampler is given in Quintana and Newton (2000). In that comparison, each observation consists of a binary time series, which was modelled by a first order Markov model. In the simulation study the Gibbs sampler was consistently, and often substantially, more efficient than the particle filter. Quintana and Newton (2000) concluded that it is doubtful whether a particle filter could be designed to efficiently analyze a large data set from such mixture models.

More encouraging are the results from Fearnhead and Clifford (2002) where a particle filter using their resampling algorithm was shown to be between one and two orders of magnitude more efficient than the particle filter of Chen and Liu (2000). If a similar increase in efficiency occurs for the MDP models that we are considering, then such a particle filter will be competitive with, and at times more efficient than MCMC methods such as the Gibbs sampler.

In order to guage the efficiency of particle filters relative to MCMC, we compared both particle filters with a Gibbs sampler. We chose the Gibbs sampler which is most similar to the particle filter. For a data set containing n observations, the state of the Gibbs sampler that we used is $z_{1:n}$ (i.e. the same state as is used in the particle filter). In one iteration of the Gibbs sampler we sequentially update z_i, for $i = 1, \ldots, n$, by drawing a new value from the full conditonal of the assignment of the ith observation conditional on the data and the assignments of the other $n - 1$ observations. The computational cost of running such a Gibbs sampler for N iterations is similar to running a particle filter with N particles. This Gibbs sampler takes advantage of the conjugate priors by integrating out the parameters, $\theta_{1:k}$. This collapsing scheme generally improves the efficiency of the Gibbs sampler (see Liu 2001, pp. 146–151).

Our comparisons of these three algorithms are based primarily on analyses of simulated data. We tested each algorithm on data sets simulated from a Gaussian mixture model. We analyzed the data under the model described in Example 1. We also compared each algorithm on the galaxy data (Roeder 1990), which is a standard test data-set for mixture models. Throughout our comparison our aim was purely to compare the computational efficiency of the three algorithms.

We summarize the performance of each method on a given data set using the effective sample size (ESS) of Carpenter, Clifford and Fearnhead (1999). (This is different to the effective sample size of Liu 1996a, which cannot be used to measure the performance of particle filters which use resampling). This involves running each particle filter (or Gibbs sampler) independently 100 times on each data set. For a specific function of the state of interest, the ESS is just the ratio of the estimate of the posterior variance of this function to the variance in estimates of the posterior mean of this function. For example, if the ith independent run of a particle filter produces an estimate, M_i, of the posterior mean of the number of components in the model, and an estimate, $(M^2)_i$, of the posterior mean of the square of the number of components in the model, the the ESS (for estimating the number of components in the model) is

$$\frac{\sum_{i=1}^{100}(M^2)_i/100 - \bar{M}^2}{\sum_{i=1}^{100}(M_i - \bar{M})^2/100}, \quad \text{where } \bar{M} = \sum_{i=1}^{100} M_i/100.$$

This measure of efficieny can be used for both particle filters and MCMC algorithms. The interpretation of the ESS is that if a particle filter, say, has an ESS of L, then inference based on that particle filter is roughly as accurate as inference based on an independent sample of size L from the full posterior distribution. A comparison based on this ESS is equivalent to a comparison based on the variability of estimates of the function of interest, across independent runs of the particle filter or Gibbs sampler. Such a comparison is sensible as any bias of each filter appears negligible.

For simplicity, in the following results we choose our function of interest to be the number of clusters. (This avoids the problems of summarizing the output that is caused by the posterior distribution being invariant under relabelling of the mixture components, see Stephens 2000b). Similar results would be obtained for different choices of this function. For ease of exposition we will denote the particle filter algorithm which uses the Fearnhead and Clifford (2003) resampling algorithm as FC, the particle filter which uses the Chen and Liu (2000) resampling algotihm as CL, and the Gibbs sampler as GS in the following sections.

4.1. *Gaussian mixture models*

We simulated data from a 3-component Gaussian mixture model:

$$X_t \sim \begin{cases} N(0, 0.5^2) & \text{with probability } 1/2, \\ N(\mu, 0.5^2) & \text{with probability } 1/6, \\ N(2\mu, \sigma^2) & \text{with probability } 1/3. \end{cases}$$

We simulated 8 data sets, each consisting of 200 observations. Four data sets were simulated with $\sigma = 0.5$ and $\mu = 0.5, 1, 2, 5$ respectively, and four with $\sigma = 2.5$ and $\mu = 0.5, 1, 2, 5$ respectively. We analyzed each data under the Gaussian mixture model of Example 1, with prior parameters $\alpha = 1/2$, $a = 1$, $b = 1$, $\eta = 0$ and $\tau = 5^2$. (We did not simulate data directly from this model of Example 1, partly because it is unrealistic in practice to believe that the model used for analysis is the true model; but also because the Gaussian mixture model we did simulate from enables us to directly alter the features of the density of X_t, and to see how that effects the computational performance of each of the three algorithms.)

We analyzed each data set independently with each of the three algorithms. Both particle filter algorithms used 5,000 particles, where as the Gibbs sampler was run for 5,500 iterations, with the results from the first 500 iterations being discarded. (Note increasing the number of particles or iterations will affect the ESS of the corresponding algorithm proportionately: for example, doubling the number of particles will roughly double the accuracy of the particle filter.) The performace of CL depends on the what rule for performing rejuvenation (the extra resampling

Table 1. *Comparison of the efficiency of two particle filters, FC using the resampling algorithm of Fearnhead and Clifford (2003) and CL using the resampling algorithm of Chen and Liu (2000), and a Gibbs sampler, GS, at analysing 200 observations from different Gaussian mixture models*

	$\sigma = 0.5$				$\sigma = 2.5$			
	$\mu = 0.5$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 0.5$	$\mu = 1$	$\mu = 2$	$\mu = 5$
FC	2.8×10^4	730	110	2.4×10^5	380	29	97	600
CL	5,300	340	11	870	270	9	43	54
GS	370	330	470	270	300	220	380	370

The values in the table are the ESS for estimating the number of components in the mixture; the larger the ESS the more efficient the algorithm is. The computational cost of the GS is 10% greater than that of either particle filter.

Table 2. *Comparison of the efficiency of two particle filters, FC using the resampling algorithm of Fearnhead and Clifford (2003) and CL using the resampling algorithm of Chen and Liu (2000), and a Gibbs sampler, GS, at analysing 100 observations from different Gaussian mixture models*

	$\sigma = 0.5$				$\sigma = 2.5$			
	$\mu = 0.5$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 0.5$	$\mu = 1$	$\mu = 2$	$\mu = 5$
FC	1.8×10^5	1,200	160	3.0×10^5	850	41	3,900	1,900
CL	5,500	320	18	750	350	13	1,700	140
GS	590	580	350	720	780	410	720	790

The values in the table are the ESS for estimating the number of components in the mixture; the larger the ESS the more efficient the algorithm is. The computational cost of the GS is 10% greater than that of either particle filter.

stage used when the weights become skewed) is chosen. Rejuvenation was used when the coefficient of variation of the particles' weights exceeded some threshold. We experimented with values of this threshold between 2 and 50, and found the results we got to be qualitatively similar. The results we present are for a coefficient of variation of 50.

As mentioned above, we summarized the performance of the three algorithms by their ESS at estimating the number of components in the mixture. The results are given in Table 1. In order to see the effect that the size of the data set had on the performance of each algorithm, we repeated our study, but just analyzed the first 100 observations from each data set. See Table 2.

The FC particle filter performs uniformly better than the CL particle filter. On 11 of the 16 data sets it also out performs the Gibbs Sampler (GS), though the GS gives a more robust performance, with its ESS varying least across the different data sets. For four of the data sets, the ESS of the FC particle filter is greater than the number of particles used. In these cases, the FC algorithm is "super-efficient" in the sense that inference based on 5,000 particles generated by the FC algorithm is more accurate than inference based on an independent sample of 5,000 from the posterior distribution of the assignments. (For smaller data sets, the FC algorithm is super-efficient more frequently; results not shown.)

In general, the size of the data set affects the performance of the FC algorithm most, and the increase in efficiency of the FC algorithm, as compared to the GS, is more marked when analyzing smaller data sets. Conversely, we would expect that the GS would be relatively more efficient, as compared to the FC algorithm, if larger data sets were analyzed.

The performance of both particle filters, and in particular the FC algorithm, varies between data set. The FC algorithm performed best for data sets simulated under $\mu = 0.5$ and $\mu = 5.0$. When $\mu = 0.5$ both particle filters performed well, but it is the $\mu = 5.0$ case that shows the largest difference between the FC and the CL algorithms.

The difference between the two algorithms is due to the resampling algorithm used. The difference between FC resampling and other resampling algorithms (such as the CL algorithm) is that putative particles with large weights are automatically kept. If the weights of the putative particles are heavily skewed then many particles will be kept, and the loss of information due to resampling, for example as measured via (7), will be small. Conversely, if the weights of all putative particles were identical then the FC resampling algorithm is just a stratified version of multinomial sampling, and is very similar to the CL resampling algorithm. When $\mu = 5.0$, the weights of the putative particles are skewed (since the densities in the mixture model are well seperated, and hence any new observation can be placed in the correct cluster with high probability) and hence we see a large improvement in the performance of the particle filter when the FC resampling algorithm is used.

4.2. *Galaxy data*

We also compared the efficiency of the three algorithms at analyzing the galaxy data of Postman, Huchra and Geller (1986). We actually analyzed the version of this data given in Roeder (1990), which omits one of the original observations (this is the form of the data that has previously been analyzed by Richardson

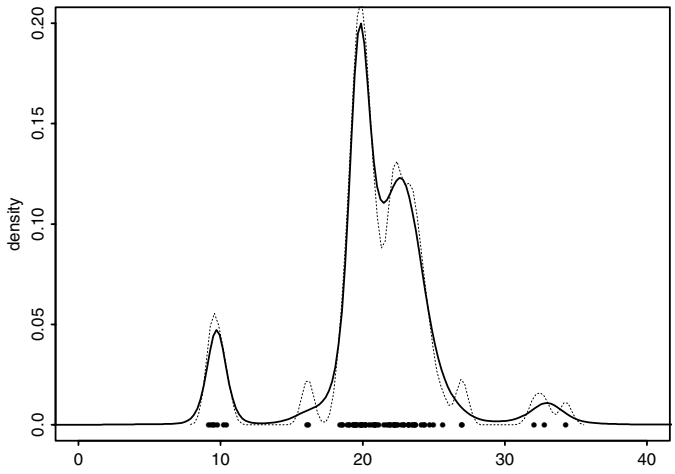

Fig. 1. *The galaxy data. The data is shown by the points. Two density estimates are shown. The full-line is the posterior estimate of the density under the MDP model. (The plot is based on the output of the FC particle filter with 50,000 particles). The dotted line is a kernel density estimate, obtained using the Splus function* `density`

and Green 1997, Stephens 2000a, amongst others). The data is shown in Fig. 1.

To analyze this data, we again assumed the Gaussian mixture model of Example 1, with prior parameters $\alpha = 1$, $a = 1$, $b = 1$, $\eta = 20$ and $\tau = 15^2$. We ran each particle filter with 50,000 particles, and we ran the Gibbs sampler for 55,000 iterations, discarding the output from the first 5,000 iterations. The performance of each method, as again measured via the ESS for estimating the number of mixture components, is shown in Table 3. The GS and the FC particle filter perform similarly (the GS takes about 10% more time to run than the FC algorithm), with the CL algorithm substantially less efficient.

The posterior expectation of the number of components is 5.75 under this model. Figure 1 shows an estimate of the mixture density, obtained from the output of the FC algorithm.

4.3. *Dealing with non-conjugate priors*

So far we have solely considered models with sufficient conjugacy that all necessary integrations, required to marginalise the posterior density to solely the posterior density for the cluster assignments, can be done analytically. For many problems this will not be the case, and here we give an example of how marginalisation can still be performed, with integrals being evaluated via Monte Carlo methods were necessary.

Table 3. *Comparison of the efficiency of two particle filters, FC using the resampling algorithm of Fearnhead and Clifford (2003) and CL using the resampling algorithm of Chen and Liu (2000), and a Gibbs sampler, GS, at analysing the galaxy data*

Algorithm	ESS
FC	1,640
CL	436
GS	1,800

The values in the table are the ESS for estimating the number of components in the mixture; the larger the ESS the more efficient the algorithm is. The computational cost of the GS is 10% greater than that of either particle filter.

Consider the Gaussian mixture model of Example 1, but assume that the prior for the mean associated with cluster j is now Gaussian, with mean η and variance τ. The resulting joint prior for (μ_j, s_j) (remember that s_j is the inverse of the variance associated with cluster j) is no longer conjugate.

Conditional on s_j, the posterior for μ_j is still Gaussian, with mean

$$\frac{\eta + \tau s_j n_j \bar{y}_j}{1 + \tau s_j n_j},$$

and variance $\tau/(1 + \tau s_j n_j)$, where n_j is the number of observations assigned to cluster j, and $\bar{y}_j$ is the mean of those observations. However the posterior for s_j can be only written up to a normalising constant, and is proportional to the following function (the dependence of this function on $z_{1:n}$ and $y_{1:n}$ is not made explicit in this notation), for $s_j > 0$:

$$f(s_j) = \pi(s_j) s_j^{n_j/2} \times \exp\left\{ -\frac{s_j n_j}{2 + 2 s_j n_j \tau}(\eta - \bar{y}_j)^2 - \frac{s_j n_j \hat{\sigma}_j^2}{2} \right\} (1 + s_j n_j \tau)^{-1/2},$$

where $\hat{\sigma}_j^2$ is the variance of the observations assigned to cluster j, and $\pi(s_j)$ is the Gamma prior for s_j (which has parameters a and b). Furthermore the posterior distribution for the assignments is

$$p(z_{1:n} \mid y_{1:n}) \propto \pi(z_{1:n}) \prod_{i=1}^{k} \left(\int_0^\infty f(s_j) \mathrm{d}s_j \right),$$

where k is the number of components in $z_{1:n}$.

To simulate from the posterior for s_j we found that rejection sampling with a Gamma proposal density

$$q(s) \propto s^{a + n_j/2 - 1} \exp\{-(b + n_j \hat{\sigma}_j^2/2)s\}, \text{ for } s > 0,$$

was efficient. To calculate $p(z_n \mid z_{1:n-1}, y_{1:n})$, requires evaluating the integral $\int_0^\infty f(s_j)\, \mathrm{d}s_j$, which we evaluated using importance sampling with proposal density $q(s)$.

We analyzed the Galaxy data under this model, using the same parameter values as in the previous section. We just tried the FC particle filter algorithm (the non-conjugacy means that the Gibbs Sampler algorithm we had been using could not be applied to this problem). The results (again measured in terms of the ESS for estimating the number of components) are shown in Table 4, for different numbers of particles, N, and for different numbers of draws, $N_{\mathbf{IS}}$, from the importance sampling density which were used to evaluate the integrals required to calculate $p(z_n \mid z_{1:n-1}, y_{1:n})$. Despite the intractability of some integrals for this problem, the performance of the particle filters is still good. Even for the particle filter with 50,000 particles, and which used 40 draws in the importance sampling, it takes only a few minutes to analyze the galaxy data on a fast PC.

Table 4. *The efficiency of a particle filter, using the resampling algorithm of Fearnhead and Clifford (2001), when analyzing the Galaxy data under a non-conjugate prior. Results are given for algorithms using different numbers of particles N, and different numbers of draws from the proposal density, $N_{\mathbf{IS}}$, in the importance sampling step*

	$N_{\mathbf{IS}}$			
N	10	20	30	40
10,000	77	150	167	171
50,000	145	270	337	405

The values in the table are the ESS for estimating the number of components in the mixture; the larger the ESS the more efficient the algorithm is.

There are two forms of inaccuracy in the particle filter algorithm: the inaccuracy through approximating the posterior distribution for assignments by a finite number of particles; and inaccuracies in calculating $p(z_n \mid z_{1:n-1}, y_{1:n})$. The former is controlled by the number of particles, N, used, and the latter by the number of draws from the proposal density, $N_{\mathbf{IS}}$, that are used in the importance sampling. To increase the performance of the particle filter, both have to be increased together. For example, if we fix $N = 10{,}000$, and consider increasing $N_{\mathbf{IS}}$, we see a noticeable improvement in accuracy as we we increase $N_{\mathbf{IS}}$ from 10 to 20, but little improvement in accuracy as we increase $N_{\mathbf{IS}}$ further. This is because, for $N = 10{,}000$ and $N_{\mathbf{IS}} = 20$, it is the approximations due to the number of particles that is most affecting the accuracy of the particle filter. Any further increase in $N_{\mathbf{IS}}$ only reduces errors in the less important approximations (those in calculating $p(z_n \mid z_{1:n-1}, y_{1:n})$), and hence has little effect on the overall performance of the particle filter. By comparison, with $N = 50,000$, the performance of the particle filter is substantially improved by each increase in $N_{\mathbf{IS}}$.

5. Discussion

In this paper we have considered using particle filters to analyze data from mixture models. Whilst this has been previously considered, we have proposed the use of the resampling algorithm of Fearnhead and Clifford (2003), in the particle filter algorithm. Simulation results show the resulting particle filter to be more efficient than the particle filter of Chen and Liu (2000). It has been pointed out by a referee that the CL algorithm can be improved by performing resampling of the particles at time t based on their weights given the observations up to time $t + 1$. The result, particularly if the resampling is performed via residual or stratified resampling, is for the CL algorithm to more closely mimic the FC algorithm. The resulting version of the CL algorithm will still suffer from occasions where there are multiple copies of identical particles, which will lead to inefficiencies when compared to the FC algorithm.

The particle filter with the Fearnhead and Clifford (2003) resampling algorithm is also a competitive alternative to an MCMC method. It is particularly well suited to smaller data sets. Sequential methods tend to struggle for large data sets, as Monte Carlo error acculumates over time. However, if observations are very informative about which cluster they belong to, then the extra Monte Carlo error introduced by the FC algorithm processing that observation will be small, and the FC algorithm can maintain its efficiency as a large number of observations are processed. (This can be related to a large skewness in the weights of the putative particles, which means that the error introduced by the FC resampling algorithm will be very small.) In some cases, this particle filter was super-efficient, that is inference based on N particles from the filter was more accurate than inference based on an independent sample of size N from the posterior distribution.

To demonstrate the difference between the Gibbs Sampler and the FC particle filter, consider analyzing data from the following 2-dimensional model (see Fig. 2):

$$X_i \sim \begin{cases} N((0,0), \sigma^2 I) & \text{with probability } 1/3 \\ N((4, 4\sqrt{3}), \sigma^2 I) & \text{with probability } 1/3 \\ N((8,0), \sigma^2 I) & \text{with probability } 1/3 \end{cases},$$

where I is the 2×2 identity matrix and $\sigma = 0.1$. We denote the three components of this model by the letters A, B and C respectively.

Lets assume that we analyze the data under the following two component model (while this is unrealistic in practice, it will help to demonstrate the difference between the two methods):

$$X_i \sim \begin{cases} N(\mu_1, \sigma_1^2 I) & \text{with probability } p \\ N(\mu_2, \sigma_2^2 I) & \text{otherwise.} \end{cases},$$

where the prior for p is uniformly distributed on $[0, 1]$, and we assume the usual conjugate prior for μ_i, σ_i. We have constructed our example so that the true posterior distribution for the assignments will place nearly all its mass on only three different

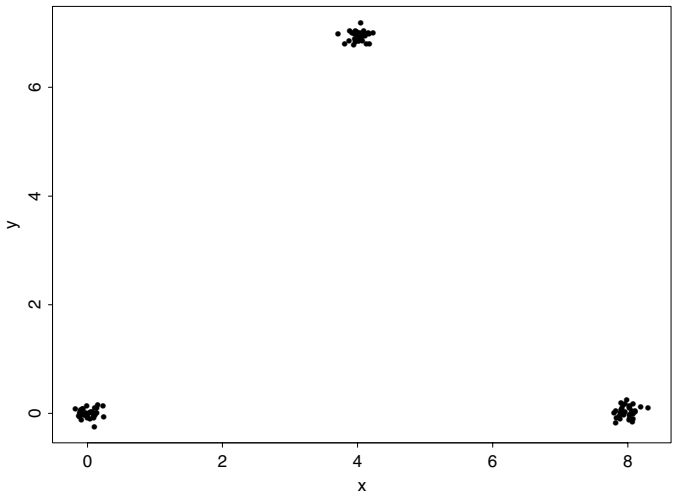

Fig. 2. *Simulated data from a three component Gaussian mixture model*

assignments: where all observations from either components A and B, clusters A and C, or clusters B and C are assigned to one cluster, and all the remaining observations are assigned to another cluster.

When the FC particle filter analyzed the data shown in Fig. 2 under this model, (which can be done as the prior probability of assigning observation $i+1$ to the first cluster, given n_1 of the first i observations have already been assigned to that cluster, is $(n_1+1)/(i+2)$) it estimated the posterior distribution of assignments almost exactly, for any number of particles $N > 3$. By comparison, the Gibbs sampler will almost never move between the three assignments which have non-negligible posterior probability, and will estimate the posterior distribution of assignments by a point-mass on just one of these three assignments.

Whilst the particle filter methods are best suited to models where there is sufficient conjugacy that the cluster parameters can be analytically integrated out of the posterior density, we have shown how they can also be applied to problems without this property. In this case some numerical or Monte Carlo integration is necessary, and we found that even an unsophisticated approach, such as using Importance Sampling to evaluate the intractable integrals, could give decent results. (More sophisticated numerical integration schemes may lead to even more efficient algorithms.) Note that implementing MCMC for such problem is possible, via the reversible jump methodology of Green (1995). However designing sensible proposal moves for such a reversible jump MCMC algorithm is nontrivial.

One aspect of inference for mixture models that we have not touched on is how to summarize the output. This can be difficult as the posterior distribution is invariant under relabelling of the mixture components, due to this symmetry the distribution of the parameters of each mixture component will be identical. A simple way around this problem is to impose constraints, so that the mixtures labels are defined based on the parameters of each component. One simple example is to label components in order of increasing means (see Richardson and Green 1997). (Such a simple approach can lead to problems, and better approaches exist, see Stephens 2000b). Our only comment on this point is that the particle filter naturally imposes a constraint that defines the mixture labels. This constraint is based on the order in which observations are processed, so that the first observation processed is assigned to the first component, the next observation that belongs to a new component is assigned to the second component, and so on.

Our example, and simulation study, has been based on a one-dimensional Gaussian mixture model. The particle filter methodology is easily extended to higher dimensions. For models with conjugate priors, the number of dimensions should not affect the performance of the particle filter, as the extra parameters necessary for modelling higher dimensional data are intergrated out. For models with non-conjugate priors, the number of dimensions may be crucial to the performance of the filter if it increases the dimension of any integrals that must be calculated numerically.

We have also only considered one specific type of mixture model, the mixture Dirichlet process. In practice, the methods introduced here could be applied to other mixture models, providing the probability distribution $p(z_{n+1} \mid z_{1:n})$ can be calculated analytically.

Recent research has suggested that incorporating MCMC into particle filters can substantially improve its performance (see Gilks and Berzuini 2001, Fearnhead 2003). We did not consider such an approach here, but it may be that using MCMC could improve the performance of the particle filter, and in particular its performance at analyzing larger data sets.

Acknowledgments

I would like to thank Gareth Roberts and Matthew Stephens for helpful discussions.

References

Akashi H. and Kumamoto H. 1977. Random sampling approach to state estimation in switching environments. Automatica 13: 429–434.

Antoniak C.E. 1974. Mixture of Dirichlet processes with applications to Bayesian nonparametric problems. Annals of Statistics 2: 1152–1174.

Blackwell D. and MacQueen J.B. 1973. Ferguson distributions via Polya urn schemes. Annals of Statistics 1: 353–355.

Bush C.A. and MacEachern S.N. 1996. A semiparametric Bayesian model for randomised block design. Biometrika 83: 275–285.

Carpenter J., Clifford P. and Fearnhead P. 1999. An improved particle filter for non-linear problems. IEE Proceedings-Radar, Sonar and Navigation 146: 2–7.

Chen R. and Liu J. 2000. Mixture Kalman filters. Journal of the Royal Statistical Society, Series B 62: 493–508.

Chopin N. 2002. A sequential particle filter for static models. Biometrika 89: 539–551.

Crisan D. and Doucet A. 2000. Convergence of sequential Monte Carlo methods. Technical report: Signal Processing Group, Cambridge University, available from `http://www-sigproc.eng.cam.ac.uk/~ad2/arnaud_doucet.html`.

Del Moral P. and Guionnet A. 1999. Central limit theorem for nonlinear filtering and interacting particle systems. The Annals of Applied Probability 9: 275–297.

Doucet A., de Freitas J.F.G., and Gordon N.J. (Eds.) 2001. Sequential Monte Carlo Methods in Practice. Springer-Verlag, New York.

Escobar M.D. 1994. 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.

Fearnhead P. 2003. MCMC, sufficient statistics and particle filters. Journal of Computational and Graphical Statistics. Biometrika 89: 848–862.

Fearnhead P. and Clifford P. 2003. Online inference for well-log data. Journal of the Royal Statistical Society, Series B. 65: 887–899.

Ferguson T.S. 1973. A Bayesian analysis of some nonparametric problems. Annals of Statistics 1: 209–230.

Gilks W.R. and Berzuini C. 2001. Following a moving target—Monte Carlo inference for dynamic Bayesian models. Journal of the Royal Statistical Society, Series B 63: 127–146.

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

Gordon N., Salmond D., and Smith A.F.M. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings-F 140: 107–113.

Green P. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82: 711–732.

Kong A., Liu J.S., and Wong W.H. 1994. Sequential imputations and Bayesian missing data problems. Journal of the American Statistical Association 93: 278–288.

Liu J.S. 1996a. Metropolised independent sampling with comparisons to rejection sampling and importance sampling. Statistics and Computing 6: 113–119.

Liu J.S. 1996b. Nonparametric hierarchical Bayes via sequential imputations. Annals of Statistics 24: 911–930.

Liu J.S. 2001. Monte Carlo Strategies in Scientific Computing. Springer, New York.

Liu J.S. and Chen R. 1998. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association. 93: 1032–1044.

Liu J.S., Chen R., and Wong W.H. 1996. Rejection control and importance sampling. Technical report: Stanford University, Department of Statistics.

Liu J.S., Chen R., and Logvinenko T. 2001. A theoretical framework for sequential importance sampling with resampling. In: Doucet A., de Freitas N., and Gordon N. (Eds.), Sequential Monte Carlo Methods in Practice. Springer–Verlag, New York, pp. 225–246.

MacEachern S.N., Clyde M.A., and Liu J.S. 1999. Sequential importance sampling for nonparametric Bayes models: The next generation. Canadian Journal of Statistics 27: 251–267.

Mukhopadhyay S. and Gelfand A.E. 1997. Dirichlet process mixed generalized linear models. Journal of the American Statistical Association 92: 633–639.

Muller P., Erkanli A., and West M. 1996. Bayesian curve fitting using multivariate normal mixtures. Biometrika 83: 67–79.

Pitt M.K. and Shephard N. 1999. Filtering via simulation: Auxiliary particle filters. Journal of the American Statistical Association 94: 590–599.

Postman M., Huchra J.P., and Geller M.J. 1986. Probes of large-scale structure in the Corona Borealis region. The Astronomical Journal 92: 1238–1247.

Quintana F.A. 1998. Nonparametric Bayesian analysis for assessing homogeniety in $k \times l$ continguency table with fixed right margin totals. Journal of the American Statistical Association 93: 1140–1149.

Quintana F.A. and Newton M.A. 2000. Computational aspects of nonparametric Bayesian analysis with applications to the modelling of multiple binary sequences. Journal of Computational and Graphical Statistics 9: 711–737.

Richardson S. and Green P.J. 1997. On Bayesian analysis of mixtures with an unknown number of components. Journal of the Royal Statistical Society, Series B 59: 731–792.

Roeder K. 1990. Density estimation with confidence sets exemplified by superclusters and voids in galaxies. Journal of the American Statistical Association 85: 617–624.

Stephens M. 2000a. Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. Annals of Statistics 28: 40–74.

Stephens M. 2000b. Dealing with label-switching in mixture models. Journal of the Royal Statistical Society, Series B 62: 795–809.

Tugnait J.K. 1982. Detection and estimation for abruptly changing systems. Automatica 18: 607–615.

West M. 1992. Modelling with mixtures. In: Bernardo J.M., Berger J.O., Dawid A.P., and Smith A.F.M. (Eds.), Bayesian Statistics 4. Clarendon Press, London.

West M. and Harrison J. 1997. Bayesian Forecasting and Dynamic Models; 2nd ed. Springer-Verlag, New York.