Embedded Hidden Markov Models for high-dimensional non-linear state space models

Valerie Bradley, Alan Chau, James Thornton

Abstract

Embedded Hidden Markov Models (EHMM) and Particle Gibbs with Backward Sampling (PGBS) are widely used methods for sampling from complex state space models where standard MCMC techniques fail. We review Shestopaloff et al. [2018] which introduces a new method for sampling from high-dimensional, non-linear, non-Gaussian state space models that builds on both EHMM and PGBS. The paper proposes a new method for generating pool states for EHMM that is as computationally efficient as PGBS, while flexible enough to adapt to the difficulties of particular applications, including multimodal posteriors. We compare the performance of the modified EHMM to that of PGBS and standard EHMM with simulation experiments.

1 Introduction

Consider a standard state space model with latent state $x_t \in \mathcal{X}$ and observation $y_t \in \mathcal{Y}$ at each time $t \in \{1, ..., n\}$. The system has the following form:

$$\begin{aligned} & x_1 \sim \mu(\cdot) \\ \text{Evolution density:} \quad & x_{t+1} \sim f(x_{t+1}|x_t) \\ \text{Observation density:} \quad & y_{t+1} \sim g(y_{t+1}|x_{t+1}) \end{aligned}$$

We assume that the $\{x_t\}$ follow a Markov process and that the observations $\{y_t\}$ are conditionally independent, given the corresponding latent state x_t. Generally, one is interested in making inferences about the posterior, or smoothing, distribution $p(x_{1:n}|y_{1:n})$ and the filtering distribution $p(x_n|y_{1:n})$.

In the case of non-linear and non-Gaussian $f(\cdot|\cdot)$ and $g(\cdot|\cdot)$, is is difficult to sample from the filtering and posterior distributions using standard Markov Chain Monte Carlo (MCMC) methods. The high-dimensional nature of state space models makes constructing proposal distributions difficult, often leading to slow mixing times or a failure to converge altogether (Godsill et al. [2004]). **Sequential importance sampling** (SIS), or particle filters, have been developed as alternative sampling methods. These techniques rely on expressing the smoothing and filtering densities recursively as:

$$p(x_{1:n}|y_{1:n}) = p(x_{1:n-1}|y_{1:n-1}) \frac{p(y_n|x_n)p(x_n|x_{n-1})}{p(y_n|y_{n-1})}$$

$$p(x_n|y_{1:n}) = \frac{p(y_n|x_n)p(x_n|y_{1:n-1})}{\int p(y_n|x_n)p(x_n|y_{1:n-1})dx_n}$$

In a standard particle filter, a proposal, or importance, function of the form

$$\pi(x_{1:n}|y_{1:n}) = \pi(x_1) \prod_{t=1}^{n} \pi(x_t|x_{1:t-1}, y_{1:t})$$

is used to propose new particles x_t without modifying the previous trajectory $x_{1:t-1}$, described in Doucet et al. [2001]. A simple Monte Carlo estimate of a quantity of interest $I(f)$ is calculated by simulating N iid particles from $\pi(x_{1:n}|y_{1:n})$, then calculating:

$$\hat{I}_N(f) = \sum_{j=1}^{N} f\left(x_{1:n}^{(j)}\right) \tilde{w}_n^{(j)}$$

where $\tilde{w}_t^{(j)}$ are the normalized importance weights, expressed recursively as

$$\tilde{w}_t^{(j)} \propto \tilde{w}_{t-1}^{(j)} \frac{p(y_t|x_t^{(j)})p(x_t^{(j)}|x_{t-1}^{(j)})}{\pi(x_t^{(j)}|x_{1:t-1}^{(j)}, y_{1:t})}$$

The main problem with SIS is that the distribution of importance weights becomes highly skewed after only a few time steps, placing almost all the weight on a small number of particles. As described in Doucet et al. [2001], the **bootstrap filter** was proposed as a solution to this degeneracy problem, and adds a selection step to the process of sampling from a high-dimensional state space that eliminates particles with small importance weights. The bootstrap filter is straightforward to implement and easy to parallelize, but provides poor estimates of the joint distribution $p(x_{1:n}|y_{1:n})$. More recent SMC methods have addressed this issue.

Neal [2003] introduces **embedded hidden Markov models** (EHMM) as a method for sampling from the full joint distribution $p(x_{1:n}|y_{1:n})$. EHMMs improve upon previous methods by creating "pool distributions" of L potential states at each time t, discretizing a continuous state space. This reduces the problem to a finite-state HMM and leverages the efficient forward-backward algorithm for HMM inference. This method is discussed in more detail in Section 2.

Andrieu et al. [2010] introduce **particle Markov Chain Monte Carlo** (PMCMC), an important development in methods of inference for high-dimensional probability distributions. PMCMC uses SMC methods to design highly efficient proposal distributions for classic MCMC methods, making Bayesian inference feasible for state space models. In this paper, we will focus specifically on **Particle Gibbs with Backward Sampling** (PGBS), a modification to PMCMC introduced by Lindsten et al. [2012]. Like EHMMs, PGBS simplifies the problem of sampling from $p(x_{1:n}|y_{1:n})$ by discretizing the state space, but while EHMMs construct pools of candidate *states* at each time step, PGBS constructs pools of candidate *sequences*, from which a new sequence is sampled with a backward pass.

The focus of this paper is a method introduced by Shestopaloff et al. [2018] for creating pool state distributions for EHMMs that reduces the computation time to be similar to that of PGBS, and combines EHMM and PGBS samplers to sample more efficiently from high-dimensional and multimodal posteriors.

We outline EHMM and PGBS in more detail in Section 2. In Section 3 we introduce the innovative method for selecting pool states for EHMM from Shestopaloff et al. [2018], and compare the new method to EHMM and PGBS on simulated data in Section 4.

2 Embedded HMM & Particle Gibbs with Backward Sampling

Two popular methods for sampling from non-linear, non-Gaussian state space models are the Embedded HMM method and the Particle Gibbs with Backward Sampling (PGBS) method. Both procedures handle the complexity of sampling from infinite state spaces by first discretizing the space, then sampling from the narrower pool of candidate states (in the case of EHMMs) or particle chains (in PGBS).

2.1 Embedded HMM

Consider a state space model with latent states $x_t \in \mathcal{X}$ and observations $y_t \in \mathcal{Y}$ for $t \in \{1, ..., n\}$. The Embedded HMM MCMC procedure introduces L auxiliary variables at each time point t, known as pool states, $x_t^{[l]} \in \mathcal{X}$ where $l \in \{1, \ldots, L\}$. The sets of pool states are denoted $\mathcal{P}_t = \{x_t^{[1]}, \ldots, x_t^{[L]}\}$. At each iteration of the MCMC scheme, the states are updated with the following steps:

Step 1: Update pool states $x_t^{[l]}$ for each l at each time t

Step 2: Update state x_t for each t by sampling from the corresponding pool states, $\mathcal{P}_t$.

Neal [2003] showed that it is possible to construct these updates in such a way that the state posterior coincides with the target posterior $p(x_{1:n}|y_{1:n})$.

Step 1 may be performed using algorithm 1 in the appendix by setting $x_t^{[l_t]}$ to x_t (the value of the current state sequence at time t) where l_t is sampled uniformly from $\{1, \ldots, L\}$. The remaining $L-1$ pool states may be generated by running a Markov chain with the correct invariant distribution $\kappa_t(x)$. More specifically, the Markov chain transitions must satisfy:

$$\kappa_t(x) R_t(x'|x) = \kappa_t(x') \tilde{R}_t(x|x')$$

for all x and x' and transition kernel densities R and $\tilde{R}$. Pool states $x_t^{[l]}$ are generated using $\tilde{R}_t(\cdot|x_l)$ for $l < l_t$ and from $R_t(\cdot|x_l)$ for $l > l_t$.

Step 2 may be performed in two ways, as described by algorithms 2 and 3 in the appendix. Firstly, one may calculate the forward probabilities $\alpha_t(x)$, $\forall x \in \mathcal{P}_t$ and use these to update the states:

$$\alpha_1(x) = \frac{p(x)p(y_1|x)}{\kappa_1(x)} \tag{1}$$

$$\alpha_t(x) = \frac{p(y_t|x)}{\kappa_t(x)} \sum_{l=1}^{L} p(x|x_{t-1}^{[l]}) \alpha_{t-1}(x_{t-1}^{[l]}) \qquad \forall t > 1 \tag{2}$$

To sample the new state sequence, we first select x'_n from $\mathcal{P}_n$ with probabilities proportional to $\alpha_n(x)$. For $t \in \{n-1, \ldots, 1\}$, state x_t is selected from corresponding $\mathcal{P}_t$ in a backward pass with probabilities proportional to $\alpha_t(x) p(x'_{t+1}|x)$.

Neal [2003] shows that **Step 2** can alternatively be performed by calculating backward probabilities $\beta_t(x)$ and then selecting a new state sequence x' with a forward pass. x_1 is set as $x_1^{[l]} \in \mathcal{P}_1$ with probabilities $\beta_1(x_1^{[l]}) p(x_1^{[l]}) p(y_1|x_1^{[l]})$ and x_t for $t > 1$ is set as $x_t^{[l]} \in \mathcal{P}_t$ with probabilities $\beta_t(x_t^{[l]}) p(x_t^{[l]}|x_{t-1}) p(y_t|x_t^{[l]})$.

$$\beta_n(x) = 1 \tag{3}$$

$$\beta_t = \frac{1}{\kappa_t} \sum_{l=1}^{L} p(y_{t+1}|x_{t+1}^{[l]}) p(x_{t+1}^{[l]}|x) \beta_{t+1}(x_{t+1}^{[l} \qquad \forall t < n \tag{4}$$

Note that computing the forward or backward probabilities for each time step requires summing over the L pool states in the previous time step for each of L new pool states. Therefore, the computation time for a single pass of the EHMM is proportional to nL^2.

The focus of this report is the method introduced by Shestopaloff et al. [2018] on how to perform **Step 1** in such a way that it is possible to perform **Step 2** efficiently, and reduce computation time from $\mathcal{O}(nL^2)$ to $\mathcal{O}(nL)$.

2.2 Particle Gibbs with Backward Sampling

Particle Gibbs with Backward Sampling MCMC is another method used to sample from $p(x_{1:n}|y_{1:n})$. This procedure is reviewed here, but for full details refer to Andrieu et al. [2010].

Let $q_1(x|y_1)$ be the importance density from which we sample particles at time 1, and let $q_t(x|y_t, x_{t-1})$ be the importance density for sampling particles particles at times $i > 1$. To initialise the process, we start with sequence $x = x_1, ..., x_n$. We set our first particle $x_1^{[1]}$ to be x_1, and sample $L-1$ particles $x_1^{[2]}, ..., x_1^{[L]}$ from q_1. For each particle, compute the following weights:

$$w_1^{[l]} = \frac{p(x_1^{[l]}) p(y_1|x_1^{[l]})}{q(x_1^{[l]}|y_1)} \tag{5}$$

$$W_1^{[l]} = \frac{w_1^{[l]}}{\sum_{m=1}^{[l]} w_1^{[m]}} \tag{6}$$

After setting up the scene, for time $t > 1$, we proceed by setting $x_t^{[1]} = x_t$. We then sample a set of $L-1$ ancestor indices from the previous pool of particles with probability proportional to $W_{t-1}^{[l]}$. We denote this set of indices by $A_{t-1}^{[l]}$. The remaining $\{x_t^{[l]}\}_{l=2}^{L}$ particles at time t, are then sampled from $q_t(x|y_t, x_{t-1}^{[A_{t-1}^{[l]}]})$. Again, for each particle we compute the following weights:

$$w_t^{[l]} = \frac{p(x_t^{[l]}|x_{t-1}^{[A_{t-1}^{[l]}]})p(y_t|x_t^{[l]})}{q_t(x_t^{[l]}|y_t, x_{t-1}^{[A_{t-1}^{[l]}]})} \tag{7}$$

$$W_1^{[l]} = \frac{w_1^{[l]}}{\sum_{m=1}^{[l]} w_1^{[m]}} \tag{8}$$

We can now select a new sequence $x^{'}$ from these pools of particles using a backward sampling pass. This is done by first selecting $x_n^{'}$ from the set of particles at time n with probabilities $W_n^{[l]}$ and then selecting the rest of the sequence going backward in time until reaching time 1, setting $x_t^{'}$ to $x_t^{[l]}$ with probability:

$$\frac{w_t^{[l]} p(x_{t+1}^{'}|x_t^{[l]})}{\sum_{m=1}^{L} w_t^{[m]} p(x_{t+1}^{'}|x_t^{[m]})} \tag{9}$$

A common choice for q is the model's transition density among the states.

3 Improving pool state generation

Shestopaloff et al. [2018] introduces a new method for generating pool states in the EHMM setting that reduces computation time such that it is comparable to that of the PGBS approach. The new method generates pool states sequentially, conditional on pool states at the previous time step, instead of independently at each time step. The method can be performed in a forward manner, by generating pool states sequentially from $t = 1, \ldots, n$, or backward by generating pool states from $t = n, \ldots, 1$. The corresponding pool densities are denoted $\kappa_t^f(x)$ and $\kappa_t^b(x)$, respectively.

In the forward scheme, the pool state density $\kappa_t^f(x)$ has the following form:

$$\begin{array}{ll} \kappa_1^f(x) \propto p(x)p(y_1|x) & t = 1 \\ \kappa_t^f(x|\mathcal{P}_{t-1}) \propto p(y_t|x) \sum_{l=1}^{L} p(x|x_{t-1}^{[l]}) & t > 1 \end{array}$$

Due to the definition of $\alpha_t(x)$ in Equations 1 and 2, specifying $\kappa_t^f(x)$ as above leaves $\alpha_t(x) \propto 1$. Therefore, when selecting new state sequence x', we simply perform a backward pass from time n to time 1, selecting a new x_t' uniformly from $\mathcal{P}_t$.

We can also construct pool states for forward selection by setting the pool state distribution to:

$$\begin{array}{ll} \kappa_1^b(x) \propto p_n(x) & t = n \\ \kappa_t^b(x|\mathcal{P}_{t+1}) \propto \sum_{l=1}^{L} p(y_{t+1}|x_{t+1}^{[l]}) p(x_{t+1}^{[l]}|x) & t < n \end{array}$$

This specification also leaves the backward probabilities $\beta_t(x)$ constant for all $t = 1, \ldots, n$, so we select a new state sequences x_t', $t = 1, \ldots, n$ uniformly from $\mathcal{P}_t$. Note that this selection of k_t^f does not improve upon the $\mathcal{O}(nL^2)$ computation time of the standard EHMM since we must still calculate L sums of L quantities to calculate each $\kappa_t^f(x)$. In order to reduce computation time, Shestopaloff et al. [2018] recognizes that the pool state densities $\kappa_t^f(x)$ can be considered the marginals of the joint density

$$\lambda_t(x, l) \propto p(y_t|x)p(x|x_{t-1}^{[l]}) \tag{10}$$

and similarly, the backward pool state densities $\kappa_t^b(x)$ are marginals of the joint density

$$\gamma_t(x, l) \propto p(y_{t+1}|x_{t+1}^{[l]})p(x_{t+1}^{[l]}|x) \tag{11}$$

In this scheme, we sample values for both l and x with probabilities proportional to $\lambda_t(x, l)$ for the forward method or $\gamma_t(x, l)$ for the backward method. Sampling from the joint densities reduces computation time to $\mathcal{O}(nL)$, but in doing so, we produce slightly worse proposals that depend only on a single particle from the previous time step instead of on the entire pool distribution. We will examine this trade-off in more depth in Section 4.

3.1 Connection to PGBS

Recall that in the standard EHMM scheme, we select an l_t uniformly from $1, \dots L$, set $x_t^{l_t}$ to the current state x_t, and iteratively generate $x_t^{[l-1]}|x_t^l$ for $l < l_t$ and $x_t^{[l+1]}|x_t^l$ for $l > l_t$. In the new scheme, however, we sample l in addition to x. There are a number of valid methods for sampling (x, l), and one such method results in a sampler with properties similar to that of PGBS.
At time $t = 1$, generate pool states $\mathcal{P}_1 = \{x_1^{[1]}, \dots, x_1^{[L]}\}$ from the prior distribution of x, $p(x)$. At times $t \geq 2$, first propose l' uniformly at random from $\{1, \dots, L\}$, then propose x' from $p(x|x_{t-1}^{[l]})$. The proposal is then accepted with probability:

$$1 \wedge \frac{p(y_t|x_t')}{p(y_t|x_t)}$$

To highlight the connection to PGBS, consider a PGBS sampler with proposal distribution $p(x_t|x_{t-1}^{[l]})$. With this proposal distribution, the importance weights for each particle at time t are proportional to $p(y_t|x_t^{[l]})$. Recall that ancestors $A_{t-1}^{[l]}$ are selected with probabilities proportional to the importance weights. In this new EHMM scheme, sampling l' is analogous to sampling an ancestor particle, then we propose x' using the same proposal distribution as the PGBS sampler just described, and jointly accept or reject the proposal with probability proportional to $p(y_t|x_t^{[l]})$. The two methods generate pool states according to the same probability densities, the key difference is that PGBS uses importance weights while EHMM uses an accept-reject M-H step. Both methods have $\mathcal{O}(nL)$ computation time and steps that depend only on a single particle from the previous time step.

3.2 Other sampling methods

The EHMM method is highly flexible and updates to x and l can be made in a number of ways - either independently, conditionally or jointly. Shestopaloff et al. [2018] describes three primary methods for sampling from $\lambda_t(x, l)$ or $\gamma_t(x, l)$: autoregressive updates, flip updates and shift updates. It is important to note that the performance and utility of an update method are highly dependent on the particulars of the application.

3.2.1 Pool Ancestor Index (l) Update

Given some update for $x|l$, one may update l in a number of ways. A simple way of selecting l', as discussed in Shestopaloff et al. [2018], is some proposal on support $\{1, \dots, L\}$ and a Metropolis-Hastings accept/ reject step e.g.sampling k from some proposal distribution with support $\{-K, \dots, -1, 1, \dots, K\}$, then setting $l' = l + k$ or sampling l' uniformly at random from $\{1, \dots, L\}$.

3.2.2 Autoregressive state updates

Autoregressive updates, Neal [1998], are designed to sample from a distribution of the form $p(y|x)p(x)$ where x is a d-dimensional $\text{MN}(\mu, \Sigma)$ and $p(y|x)$ is the distribution of observed data y, dependent on x. Proposals x' are generated according to:

$$x' = \mu + \sqrt{1 - \epsilon^2}(x - \mu) + \epsilon L n$$
$$n \sim \text{N}(0, I^d)$$

where x is the current state, $LL^T = \Sigma$, and $\epsilon \in [-1, 1]$ is a parameter used to tune the magnitude of the proposal. Proposals are accepted with probability:

$$1 \wedge \frac{p(y|x')}{p(y|x)}$$

Autoregressive updates are useful for situations in which the underlying model is a Gaussian process, or when time steps are highly correlated with one another so local proposals are likely to perform well.

3.2.3 Shift updates

Shift updates are a new type of update proposed by Shestopaloff et al. [2018] that jointly updates x and l. First, l' is proposed using any valid method (for example selected uniformly at random), then x' is proposed such that the relationship between x'_t and $x^{[l']}_{t-1}$ mirrors that of x_t and $x^{[l]}_{t-1}$. For example, in the Gaussian setting where $x_t|x_{t-1} \sim \text{MV}(x_{t-1}, \Sigma)$, x' is proposed according to:

$$x'_t = x_t - x^{[l]}_{t-1} + x^{[l']}_{t-1}$$

and accepted with probability $(1 \wedge p(y|x')/p(y|x))$. Shift updates are designed to ensure that the pool distribution is well-dispersed, however they have the potential to be over-dispersed, depressing the acceptance rate and leading to poor mixing, so should be carefully tuned.

3.2.4 Flip updates

The last type of update Shestopaloff et al. [2018] considers is the "flip" update. This update is designed to be used in situations where the posterior may be multimodal, so local proposals in high-dimensional settings may not fully explore the space. A flip update sets (x, l) to $(-x, l')$, so for any x_t in the pool, $-x_t$ is also included in the pool. Flip updates are intended to be combined with any other update method, and updates from $l = (1 \dots L)$ alternate between a normal update and a flip update.

4 Simulation Study

Consider the following model, as detailed in Shestopaloff et al. [2018], for $n = 250$ and $P = 10$, $\forall t \in \{1, \dots, n\}$, $\forall j \in \{1, \dots, P\}$:

$$X_1 \sim \mathcal{N}(0, \Sigma_{init}) \tag{12}$$
$$X_t \sim \mathcal{N}(0.9x_{i-1}, \Sigma) \qquad X_t = (X_{t,1}, \dots, X_{t,P})' \tag{13}$$
$$Y_{t,j} \sim Poisson(exp(-0.4 + 0.6x_{t,j})) \tag{14}$$

$$\{\Sigma_{init}\}_{i,j} = \begin{cases} \frac{0.7}{1-0.9^2} & \text{if } i \neq j \\ \frac{1}{1-0.9^2} & \text{if } i = j \end{cases} \qquad \{\Sigma\}_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

See appendix section A.2 for visualisations of how the observations may appear in one dimension.

Now consider the following simulation procedures for states $\{X_t\}_{t=1}^n$:

- **Metropolis Hastings with Autoregressive Proposal (MH)**
  One may exploit the underlying Gaussian state propagation by using autoregressive Metropolis Hastings updates detailed in Neal [1998] and section 3.2.2 to sample states.
- **Particle Gibbs Backward Sampling (PGBS)**
  This procedure is detailed in section 2.2 and is implemented using the data generating process' initial density $p(x)$ and transition density $p(x_t|x_{t-1})$ as importance densities to generate particles. The number of particles is set to 250.
- **Particle Gibbs Backward Sampling with Metropolis Hastings (PGBS-MH)**
  This procedure uses a cycle of kernels per MCMC iteration, first the PGBS as detailed above and in section 2.2, then an autoregressive Metropolis Hastings step as in 3.2.2.
- **Embedded HMM MCMC (EHMM)**
  This procedure is detailed in section 3, with the number of pool states as 50. At time $t = 1$, an autoregressive update is used to sample pool states, then for $t > 1$ pool states are sampled using $\lambda(x, l)$ in equation (10) using a cycle of the autoregressive update and then shift update with uniform proposal on l with support $\{1, \dots, L\}$. These sampling procedures are detailed in sections 3.2.2 and 3.2.3

States $\{x_t\}_t$ are initialised to 0 in each procedure. The scaling parameter, ϵ, is set at alternate iterations as 0.2 and 0.8 in the MH and PGBS-MH procedures above. ϵ is chosen from the $\mathcal{U}nif(0.1, 0.4)$ at each iteration for the EHMM procedure. At each iteration of the PGBS, PGBS-MH and EHMM procedures, a forward state sample is taken then a reversed state sample is taken using the time reversibility of the Gaussian state propagation, see Shestopaloff et al. [2018] for full details. The reason for the reverse-sampling is that under alternate direction sampling the tails x_1, x_n and associated pools will be updated using the whole chain of data, and hence likely to be better explored than otherwise.

It is clear from figure 1 below that the PGBS method suffers from some quite prominent spikes in autocorrelation. Upon closer inspection, figure 2 shows that the MH procedure also exhibits relatively large autocorrelation compared to both the PGBS-MH and EHMM procedures. The EHMM appears to be less volatile to PGBS-MH, however, from figure 3 it is difficult to visually compare the two procedures to a conclusive assessment. This observation is intuitive given the interpretation of EHMM being a special case of PGBS differing by proposal distribution and by the accept/reject Metropolis Hastings step in EHMM compared to the importance sampling step in the PGBS procedure. Hence, one expects PGBS with a MH step to be similar to the EHMM MCMC procedure.

It has been noted by Shestopaloff et al. [2018] that PGBS performs well at times t where y_t is informative and does not perform well at less informative y_t. The MH procedure performs well in the opposite case, under less informative y_t. This explains the spikes in figure 1 and can also be seen in figure 2 with the alternative peaks between PGBS and MH. The cycle of both, PGBS-MH, appears to capture positive aspects of the two.

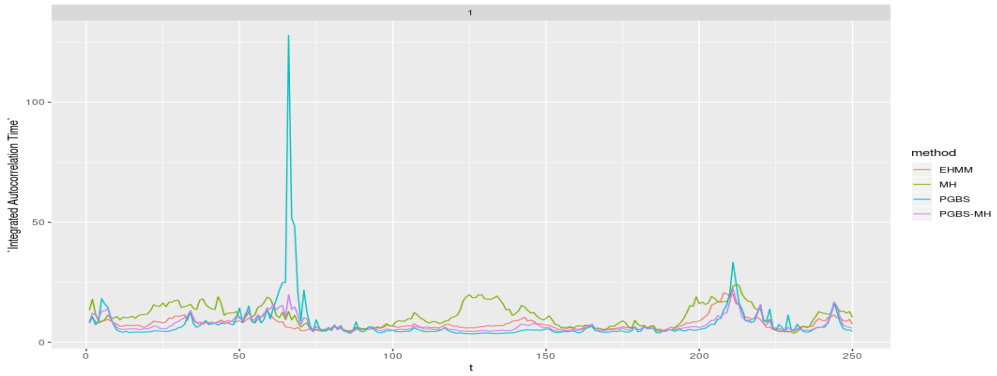

Figure 1: Integrated Autocorrelation Time for $x_{t,1}$ through time for the various sampling procedures

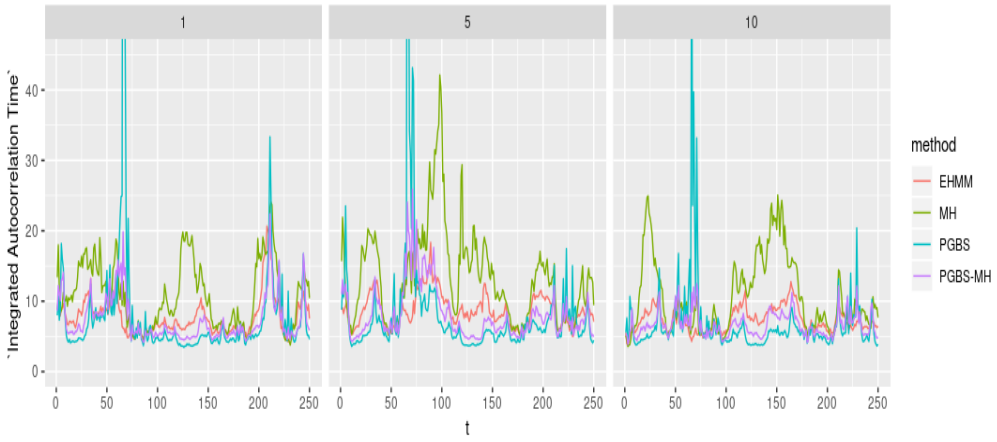

Figure 2: Integrated Autocorrelation Time for $x_{t,1}, x_{t,5}, x_{t,10}$ through time for the various sampling procedures, zoomed-in

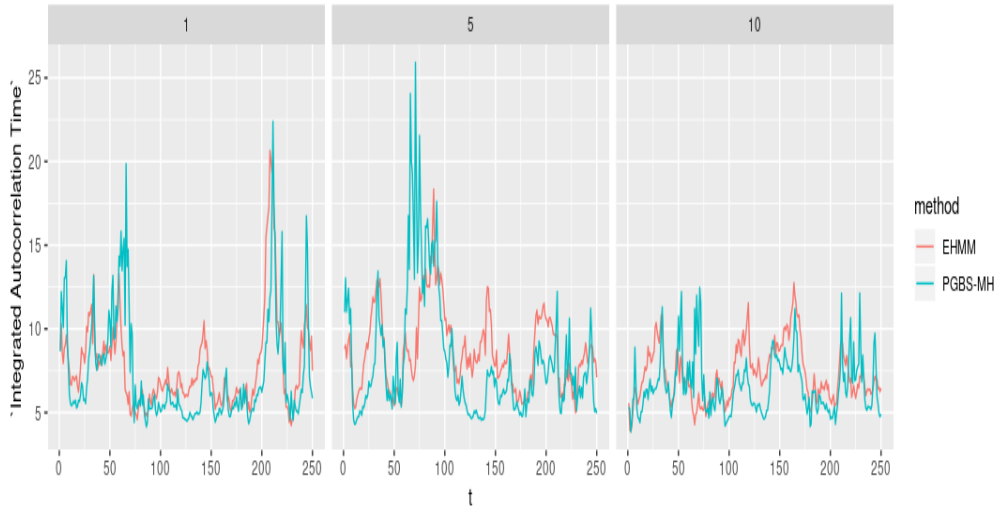

Figure 3: Integrated Autocorrelation Time for $x_{t,1}, x_{t,5}, x_{t,10}$ through time for the EHMM and PGBS-MH sampling procedures, zoomed-in

5 Discussion

In this report we have summarised the main methodological aspects of Particle Gibbs Backward Sampling, the Embedded Hidden Markov Model MCMC process and the extension proposed by Shestopaloff et al. [2018]. We have then detailed specific proposals suggested by Shestopaloff et al. [2018] including the autoregressive MH step and the shift proposal. Our brief simulation study shows some of the attractive properties exhibited when combining multiple schemes in the PGBS-MH procedure and how the EHMM method achieves similar properties.

It was additionally noted how the proposed EHMM method has similar computational complexity to PGBS, $\mathcal{O}(nL)$, which outperforms previous sampling schemes for EHMM procedures with complexity $\mathcal{O}(nL^2)$. This is significant as high dimensional or multi-modal data requires a large number of pool states L to explore the state-space quickly, hence the reduced complexity in L is highly beneficial. At a high-level, the technique of sampling both pool index l and pool state x, which results in the reduced complexity, is essentially substituting overhead in calculating forward or backward probabilities (in equation (1), (2), (3) and (4)) with the additional computation / iterations associated with the added randomness from sampling l. One would expect that the overall computation per update would be lower in this new method, and the mixing slower given there is now a larger space to explore. It would however be interesting to compare the integrated autocorrelation times of the previous EHMM method to assess the trade-off in overhead and randomness appropriately.

An aspect of the EHMM procedure that was not covered in this report, due to brevity, was the benefit over PGBS in a multi-modal setting, as detailed by Shestopaloff et al. [2018].

Further directions would be to derive further theoretical properties of the EHMM. A natural extension not covered in this report is the case where the parameters in the data-generating process are unknown and modelled as random with priors. It may be interesting to compare how the EHMM method performs in this Bayesian setting.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. pages 3–14, 2001.

Simon J Godsill, Arnaud Doucet, and Mike West. Monte carlo smoothing for nonlinear time series. *Journal of the american statistical association*, 99(465):156–168, 2004.

Fredrik Lindsten, Thomas Schön, and Michael I Jordan. Ancestor sampling for particle gibbs. pages 2591–2599, 2012.

Radford M Neal. Regression and classification using gaussian process priors. *Bayesian statistics*, 6:475—-501, 1998.

Radford M Neal. Markov chain sampling for non-linear state space models using embedded hidden markov models. *arXiv preprint math/0305039*, 2003.

Alexander Y Shestopaloff, Radford M Neal, et al. Sampling latent states for high-dimensional non-linear state space models with the embedded hmm method. *Bayesian Analysis*, 13(3):797–822, 2018.

A Appendix

A.1 Algorithms

Algorithm 1: Pool States Update

Input: $\{x_t\}_t$;
Output: $\mathcal{P}_t = \{x_t^{[1]}, \ldots, x_t^{[L]}\}$, $\forall t \in \{1, \ldots, n\}$;
foreach $t \in \{1, \ldots, n\}$ **do**
  Set $x_t^{[1]} \leftarrow \mathrm{x}_t$;
  Sample $l_t \sim \mathcal{U}nif(\{1, \ldots, L\})$;
  foreach $l \in \{l_t + 1, \ldots, L\}$ **do**
    Sample $x_t^{[l]} \sim \mathcal{M}arkovChain(x_t^{[l-1]}, R)$;
  end
  foreach $l \in \{l_t - 1, \ldots, 1\}$ **do**
    Sample $x_t^{[l]} \sim \mathcal{M}arkovChain(x_t^{[l+1]}, \tilde{R})$;
  end
end

Algorithm 2: Forward probability computation and backward state sampler

Input: $\mathcal{P}_t = \{x_t^{[1]}, \ldots, x_t^{[L]}\}$, $\forall t \in \{1, \ldots, n\}$;
Output: $\{x_t\}_t$;
foreach $t \in \{1, \ldots, n\}$ **do**
  Compute α_t according to equations: (1) and (2);
end
Sample l from $\{1, \ldots, L\}$ with probabilities proportional to $\alpha_n(x_n^{[l]})$;
Set $x_n \leftarrow x_n^{[l]}$;
foreach $t \in \{n-1, \ldots, 1\}$ **do**
  Sample l with probability proportional to $\alpha_t(x_t^{[l]})p(x_{t+1}|x_t^{[l]})$;
  Set $x_t \leftarrow x_t^{[l]}$
end

Algorithm 3: Backward probability computation and forward state sampler

Input: $\mathcal{P}_t = \{x_t^{[1]}, \ldots, x_t^{[L]}\}$, $\forall t \in \{1, \ldots, n\}$;
Output: $\{x_t\}_t$;
foreach $t \in \{1, \ldots, n\}$ **do**
  Compute β_t according to equations: (3) and (4);
end
Sample l from $\{1, \ldots, L\}$ with probabilities proportional to $\beta_1(x_1^{[l]})p(x_1^{[l]})p(y_1|x_1^{[l]})$;
Set $x_1 \leftarrow x_1^{[l]}$;
foreach $t \in \{2, \ldots, n\}$ **do**
  Sample l with probability proportional to $\beta_t(x_t^{[l]})p(x_t^{[l]}|x_{t-1})p(y_t|x_t^{[l]})$;
  Set $x_t \leftarrow x_t^{[l]}$
end

A.2 Explanatory Simulation

A.2.1 Experiment 1: Gaussian Transition & Gaussian Observation

We will start with a simple example for the original embedded HMM with independent pool sampling scheme. Consider a model in which the state space $\mathcal{X}$ and the observation space, $\mathcal{Y}$, are both in the reals $\mathcal{R}$. Let each observation be simply the state plus Gaussian noise of standard deviation σ_m, ie. $P(y_t|x_t) = \mathcal{N}(y_t|x_t, \sigma_m^2)$. We also let the transitions be defined by $P(x_t|x_{t-1}) = \mathcal{N}(x_t|\tanh(\eta x_{t-1}), \sigma^2)$, for some constant expansion factor η and transition noise standard deviation τ. For the pool states, we will choose the Markov transition to be the following:

$$R_t(x'|x) = \mathcal{N}(x'|\rho x, \tau^2) \tag{15}$$

with this transition, we get the following stationary distribution, given we sample the first particle from a standard normal:

$$\kappa(x) = \mathcal{N}(x|0, \frac{\tau^2}{1-\rho^2}) \tag{16}$$

In this experiment, we have set $\sigma_m = 2.5, \sigma = 0.4, \eta = 2.5, \tau = 1, \rho = 0$.

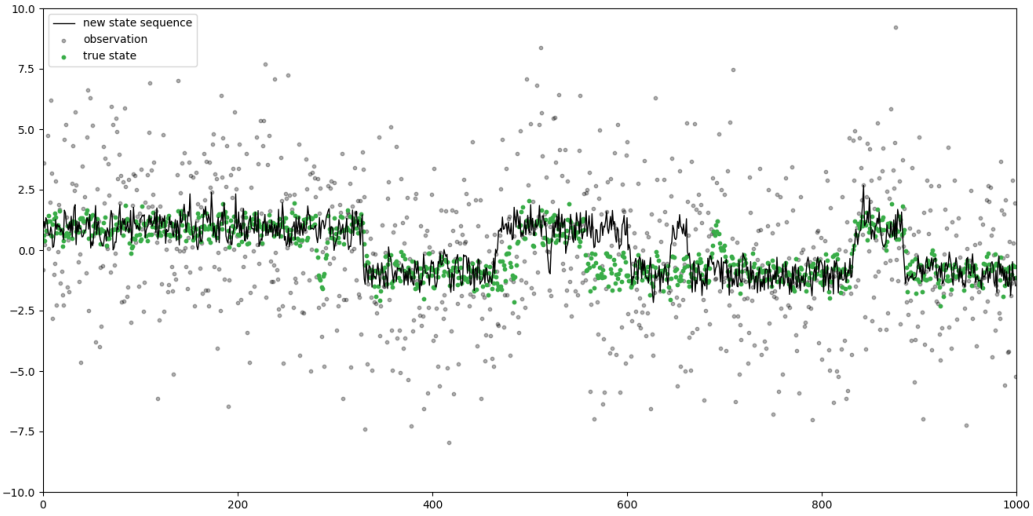

Figure 4: A simple illustration of embedded HMM with Gaussian transition and Gaussian observations. Grey dots being the observation and green dots are the true states. The black line is the predicted latent states from the original embedded HMM model.

A.2.2 Experiment 2: Gaussian Transition & Poisson Observation

With the same Markov transitions and state space transitions as experiment 1, we modified the output of the model to Poisson instead of Gaussian:

$$p(y|x) = Poisson(y|\exp(a + bx)) \tag{17}$$

where a and b are parameters that we set.

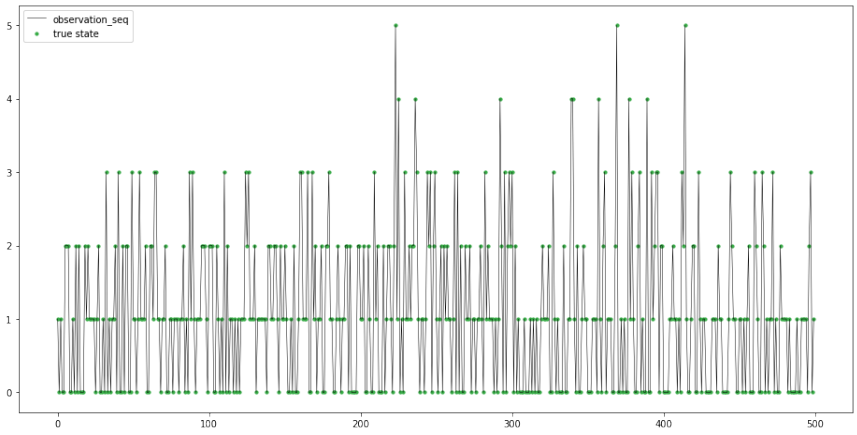

Figure 5: Demonstrating the Poisson count from the underlying latent states in experiment 2.

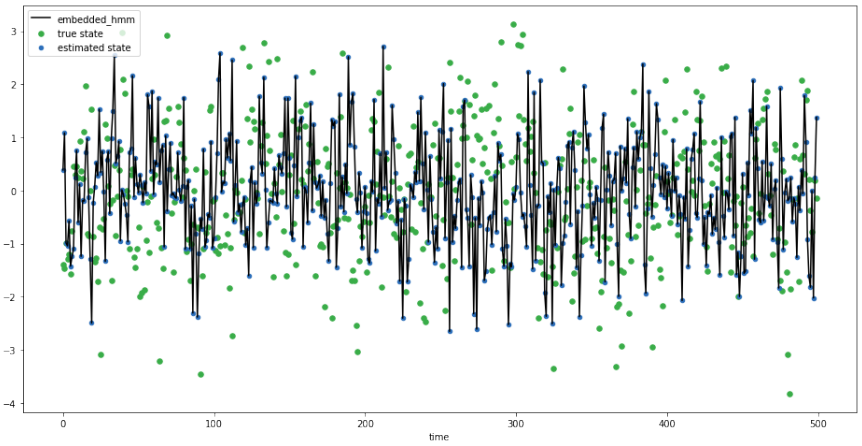

Figure 6: A simple illustration of embedded HMM with Gaussian transition and Poisson observations. Green dots are the true state and the black line is the predicted states.