Metropolis-Hastings Implementations of the Forward-Filtering-Backward-Sampling Algorithm for Particle Smoothing

UNIVERSITY OF CAMBRIDGE

P. Bunch and S. Godsill

Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK

{pb404,sjg}@eng.cam.ac.uk

Abstract

The conventional forward-filtering-backward-sampling algorithm for particle smoothing generates samples from the joint smoothing distribution by sequentially sampling backwards from the forward filter approximations. This algorithm has a computational complexity quadratic in the number of particles, which is often restrictive. In addition, the support of the smoothing distribution is restricted to the values which appear in the filtering approximation. Here a Metropolis-Hastings sampling procedure is used to improve the efficiency of the forward-filtering-backward-sampling algorithm. In addition, an algorithm for approximating the joint smoothing distribution without limited support is presented, which achieves simultaneous improvements in both execution time and error. These algorithms also provide a greater degree of flexibility over existing methods, allowing a trade-off between execution time and estimation error, controlled by the length of the Markov chains.

To be published as:

P. Bunch and S. Godsill. "Improved particle approximations to the joint smoothing distribution using Markov chain Monte Carlo." *IEEE Transactions on Signal Processing*.

Introduction

Consider a hidden Markov model for Bayesian inference,

$$x_k \sim p(x_k|x_{k-1})$$
$$y_k \sim p(y_k|x_k).$$

We address the task of joint smoothing — inference of $p(x_{1:K}|y_{1:K})$ (where K is the number of time steps).

For non-linear, non-Gaussian models, particle methods may be used to represent posterior distributions with a set of samples [1].

- Filter: $\hat{p}(x_k|y_{1:k}) = \sum_i w_k^{(i)} \delta_{x_k^{(i)}}(x_k)$.
- Smoother: $\hat{p}(x_{1:K}|y_{1:K}) = \sum_i w_K^{(i)} \delta_{x_{1:K}^{(i)}}(x_{1:K})$.

Forward-Filtering-Backward-Sampling

The forward-filtering-backward-sampling (FFBS) algorithm [2] draws particles from the joint smoothing distribution by sequentially sampling a set of backwards conditional distributions,

$$p(x_{1:K}|y_{1:K}) = p(x_K|y_{1:K}) \prod_{k=1}^{K-1} p(x_k|x_{k+1}, y_{1:K}).$$

The conditional factors may be expressed as,

$$p(x_k|\tilde{x}_{k+1}, y_{1:K}) = p(x_k|\tilde{x}_{k+1}, y_{1:k}) \propto p(\tilde{x}_{k+1}|x_k) p(x_k|y_{1:k}).$$

where $\tilde{x}_{k+1}$ is the future state already sampled. Using the filter approximations, the backwards conditional distributions are approximated as,

$$\hat{p}(x_k|\tilde{x}_{k+1}, y_{1:K}) = \sum_i \tilde{w}_k^{(i)} \delta_{x_k^{(i)}}(x_k)$$
$$\tilde{w}_k^{(i)} \propto w_k^{(i)} p(\tilde{x}_{k+1}|x_k^{(i)}).$$

This may be sampled easily by calculating $\tilde{w}_k^{(i)}$ for each of the N_F filter particles and sampling the resulting categorical distribution. The whole procedure is repeated N_S times, once for each smoothing particle.

1: Run a particle filter to approximate $p(x_k|y_{1:k})$ for each k and store the resulting particles $\{x_k^{(i)}, w_k^{(i)}\}$.
2: **for** $i = 1$ **to** N_S **do**
3: Sample $\tilde{x}_K^{(i)} \sim \hat{p}(x_K|y_{1:K})$.
4: **for** $k = K-1$ **to** 1 **do**
5: Sample $\tilde{x}_k^{(i)} \sim p(x_k|\tilde{x}_{k+1}, y_{1:K})$.
This is the critical step. See text for various methods.
6: **end for**
7: **end for**

Algorithm 1: *Forward-filtering-backward-sampling algorithm*

Deficiencies

The basic FFBS algorithm suffers from two drawbacks:

- The algorithmic complexity is $\mathcal{O}(N_F \times N_S)$ which is restrictive.
- The support for the smoothing approximation is limited to the values appearing in the filtering particles.

A Metropolis-Hastings implementation

The FFBS bottleneck is the calculation of the N_F backwards weights for every sampling step, in order to sample from $\hat{p}(x_k|\tilde{x}_{k+1}, y_{1:K})$. Instead, we can sample this distribution using Metropolis-Hastings (MH).

- The filter weights are used to propose particles from $\hat{p}(x_k|y_{1:k})$.
- The new state, x_k^*, replaces the old state, x_k with acceptance probability,

$$\alpha = \min\left[1, \frac{p(\tilde{x}_{k+1}|x_k^*)}{p(\tilde{x}_{k+1}|x_k)}\right].$$

Initialisation

- An initial value is required to start the chain.
- Instead of sampling the state, x_k, at each step, sample the entire history, $x_{1:k}$.
- No additional states need be stored, only index of the $(k-1)$-time particle from which each k-time particle originates.
- No burn-in required — initial state is drawn from target distribution.

Effects

- Computational complexity reduced to $\mathcal{O}(M \times N_S)$, where M is the length of Markov chains.
- $M = 0$ reproduces particles from the filter smoother. $M \to \infty$ draws an independent sample from the distribution.
- Performance expected to be bounded by that of the filter smoother and the standard FFBS.
- Much faster than FFBS when M can be made small, i.e. when acceptance rate is high.

Alternatives

Rejection sampling [3] can achieve a similar complexity reduction.

- Rejection sampling can be very slow if rejection rates are high, e.g. when state dimension high.
- MH method more flexible — chain length can be set to trade-off complexity against running time.

Improving the Support

Modify the MH proposal to use,

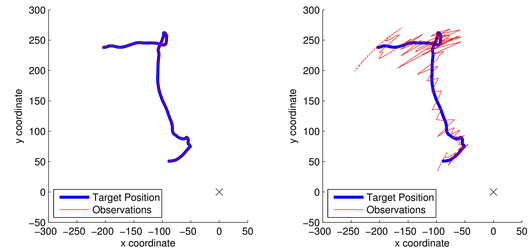
$$\{x_{1:k-1}^{(i^*)}, x_k^*\} \sim \hat{p}(x_{1:k-1}|y_{1:k-1}) q(x_k|x_{k-1}, \tilde{x}_{k+1}, y_k).$$

Support of x_k is no longer restricted to those values appearing in the filter approximation (although support of $x_{1:k-1}$ is). This is inspired by the method used in [4] for two-filter marginal smoothing. Acceptance probability is now,

$$\alpha = \min\left[1, \frac{p(\tilde{x}_{k+1}|x_k^*) p(x_k^*|x_{k-1}^{(i^*)}) p(y_k|x_k^*) q(x_k|x_{k-1}, \tilde{x}_{k+1}, y_k)}{p(\tilde{x}_{k+1}|x_k) p(x_k|x_{k-1}) p(y_k|x_k) q(x_k^*|x_{k-1}^{(i^*)}, \tilde{x}_{k+1}, y_k)}\right].$$

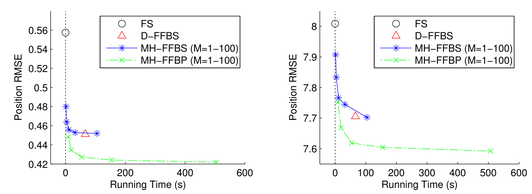
Demonstrations

New algorithms tested on a simulated tracking problem with near constant velocity dynamic model and range-bearing observation model. Two observation covariance cases are considered.



(a) Case 1 (b) Case 2

Figure 1: *Example trajectory with observations from the two noise covariance cases. Constant likelihood contours are shown for the first and last position (dashed)*



(a) Case 1 (b) Case 2

Figure 2: *Position RMSEs and running times of various smoothing algorithms with two noise covariance cases.*

Conclusions

- New smoothing algorithms achieve lower error for a given running time. Similar results apply to consistency of estimates.
- Chain length provides flexibility to trade-off running time against quality of estimate.

References

[1] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovsky, Eds. Oxford University Press, 2009.

[2] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, pp. 156–168, 2004.

[3] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "Sequential Monte Carlo smoothing for general state space hidden markov models," *The Annals of Applied Probability*, vol. 21, no. 6, pp. 2109–2145, 2011.

[4] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, pp. 447–464, 2010.