MCMC for Conditionally Linear Gaussian State-Space Models

Kevin Murphy

2/22/00
Updated 5/24/00

1 Introduction

Consider the problem of offline inference (smoothing) in the Dynamic Bayesian Network (DBN) shown in Figure 1, i.e., we want to compute $P(\mathbf{X_{1:T}}, \mathbf{R_{1:T}}|\mathbf{y_{1:T}})$. If the Conditional Probability Distributions (CPDs) for X and Y are linear Gaussian, this is called a conditionally linear-Gaussian state-space model (CLGSSM): conditioned on $R_{1:T}$, the model can be solved using a Kalman smoother. In the special case that R is discrete, it is called a jump Markov linear system (JMLS). This model is very flexible, and can handle sensor failures, outliers and regime changes.

Carter and Kohn [CK96] proposed an efficient MCMC algorithm for inference in JMLSs, as follows:

1. Randomly initialise $r_{1:T}^{(0)}$.
2. For $k = 1, 2, \ldots$ until convergence
   (a) For $t = 1, \ldots, T$, sample $r_t^{(k)} \sim P(R_t|\mathbf{y}_{1:T}, \mathbf{r}_{-t}^{(k)})$
   (b) Compute $\mathbf{x}_{0:T}^{(k)} = E\left[\mathbf{x}_{0:T}|\mathbf{y}_{1:t}^{(k)}, \mathbf{r}_{1:T}^{(k)}\right]$

where $\mathbf{r}_{-t}^{(k)} \stackrel{\text{def}}{=} \left(r_1^{(k)}, \ldots, r_{t-1}^{(k)}, r_{t+1}^{(k-1)}, \ldots, r_T^{(k-1)}\right)$ contains new values of R to the left of R_t, and old values to

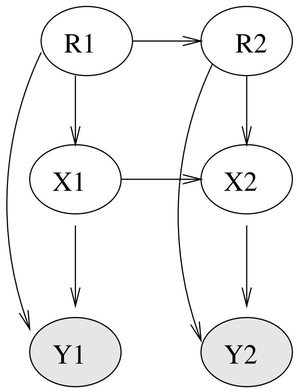

Figure 1: A simple DBN.

the right. The final estimate is then obtained by averaging the samples $\mathbf{r}_{1:T}^{(k)}$ and $\mathbf{x}_{1:T}^{(k)}$, possibly discarding an initial segment corresponding to the burn-in period of the Markov chain.

The main issue is how to efficiently compute the sampling distribution $P(R_t|\mathbf{y}_{1:T}, \mathbf{r}_{-t}^{(k)})$. [CK96] propose an $O(T)$ algorithm which involves first running the backwards Kalman filter[1], and then working forwards, sampling R_t and doing Bayesian updating conditional on the sampled value. (See [DA99] for some slight improvements.) Both papers are difficult to read because of the heavy notation involved in explicitely manipulating Gaussians. Here, we rederive the algorithm for the simpler case in which X is a discrete random variable. (Y can have an arbitrary distribution, since it is observed.) The original algorithm can then be recovered by replacing the discrete multiplication and addition operators on the X_t nodes with their Gaussian equivalents in the obvious way [Mur98]. We then relate this to Dawid's random sampling algorithm [Daw92].

2 Computing the sampling distribution

The sampling distribution is

$$P(r_t|\mathbf{y}_{1:T}, \mathbf{r}_{-t}) \propto P(r_t|\mathbf{r}_{-t}) P(\mathbf{y}_{1:T}|\mathbf{r}_{1:T})$$

The second term is given by

$$\begin{aligned}
P(\mathbf{y}_{1:T}|\mathbf{r}_{1:T}) &= \sum_i P(\mathbf{y}_{t+1:T}|\mathbf{y}_{1:t}, \mathbf{r}_{1:T}, X_t = i) P(\mathbf{y}_{1:t}, X_t = i|\mathbf{r}_{1:T}) \\
&= \sum_i P(\mathbf{y}_{t+1:T}|\mathbf{r}_{t+1:T}, X_t = i) P(\mathbf{y}_{1:t}, X_t = i|\mathbf{r}_{1:t}) \\
&= P(\mathbf{y}_{1:t}|\mathbf{r}_{1:t}) \sum_i P(\mathbf{y}_{t+1:T}|\mathbf{r}_{t+1:T}, X_t = i) P(X_t = i|\mathbf{y}_{1:t}, \mathbf{r}_{1:t}) \\
&= P(\mathbf{y}_{1:t-1}|\mathbf{r}_{1:t-1}) P(y_t|\mathbf{y}_{1:t-1}, \mathbf{r}_{1:t}) \sum_i P(\mathbf{y}_{t+1:T}|\mathbf{r}_{t+1:T}, X_t = i) P(X_t = i|\mathbf{y}_{1:t}, \mathbf{r}_{1:t})
\end{aligned}$$

Hence, dropping terms that are independent of r_t,

$$P(r_t|\mathbf{y}_{1:T}, \mathbf{r}_{-t}) \propto P(r_t|r_{t-1}) P(r_{t+1}|r_t) P(y_t|\mathbf{y}_{1:t-1}, \mathbf{r}_{1:t}) \sum_i P(\mathbf{y}_{t+1:T}|\mathbf{r}_{t+1:T}, X_t = i) P(X_t = i|\mathbf{y}_{1:t}, \mathbf{r}_{1:t})$$

The $P(X_t = i|\mathbf{y}_{1:t}, \mathbf{r}_{1:t})$ and $P(y_t|\mathbf{y}_{1:t-1}, \mathbf{r}_{1:t})$ terms can be computed by running the forwards algorithm on the X/Y HMM, sampling R_t as we go. The $P(\mathbf{y}_{t+1:T}|\mathbf{r}_{t+1:T}, X_t = i)$ term can be computed by using the backwards algorithm on the X/Y HMM. The key observation is that this term is independent of $\mathbf{r}_{1:t}^{(k)}$, and hence can be computed ahead of time, before sampling $\mathbf{R_{1:t}}$ on the forwards pass. The terms $P(r_t|r_{t-1}^{(k)})$ and $P(r_{t+1}^{(k-1)}|r_t)$ can be computed by looking them up in the transition matrix for R. We give the algorithmic details in the appendix.

3 Relationship to Dawid's algorithm

In [SHJ97] they show that the forwards-backwards algorithm for HMMs is a special case of the junction tree algorithm [CDLS99]. In particular, they show that, when the root is on the left (i.e., the root is the clique containing $X_1 X_2$),

[1] For technical reasons, the backwards form of the KF works with inverse covariance matrices, whereas the forwards form can work with either inverse or regular covariance matrices. Working with inverse matrices is called the information form of the KF. See [Mur98] for details.

performing a collect-to-root operation is equivalent to the backwards pass, which computes $\beta_t(i) = P(y_{t+1:T}|X_t = i)$, and the distribute-from-root operation corresponds to a smoothing forwards pass, which computes $\gamma_t(i) = P(X_t = i|y_{1:T})$. (If the root is on the right, the collect operation computes $\alpha_t(i) = P(X_t = i|y_{1:t})$ and the distribute operation computes γ_t.) In [Mur98] we extend these results to linear Gaussian state space models (LGSSM).

In Dawid's random sampling algorithm [Daw92], the regular distribute-from-root operation is changed so that instead of summing out nodes, we sample them. When applied to a LGSSM, this is equivalent to the Gibbs sampling algorithm of [CK94]. A direct application of Dawid's algorithm to a *conditional* LGSSM would would require the moment-matching approximation of [Lau92] on the backwards pass, because the hidden nodes are both discrete and Gaussian. But consider a modified version of Dawid's algorithm in which we first instantiate the discrete nodes, then do a right-to-left pass (which can now be done exactly, using the backwards Kalman filter), and then run the regular sampling algorithm in the left-to-right pass. This is equivalent to [CK96] described above.

A Details of the discrete Carter-Kohn algorithm

We define the following CPDs.

$$\begin{aligned}
T_R(r'; r) &\stackrel{\text{def}}{=} P(R_t = r|R_{t-1} = r') \\
T_X(r, x'; x) &\stackrel{\text{def}}{=} P(X_t = x|R_t = r, X_{t-1} = x') \\
T_Y(r, x; y) &\stackrel{\text{def}}{=} P(Y_t = y|R_t = r, X_t = x) \\
\pi(i, r) &\stackrel{\text{def}}{=} \Pr(X_1 = i|R_1 = r)
\end{aligned}$$

In the backwards pass, for a fixed instantiation $\mathbf{r}_{1:T}^{(k-1)}$, we compute

$$\beta_t(i) \stackrel{\text{def}}{=} P(\mathbf{y}_{t+1:T}|X_t = i, \mathbf{r}_{t+1:T}^{(k-1)})$$

using the regular backwards algorithm for HMMs:

1. $\beta_T(i) = 1$ for all i
2. for $t = T-1, \ldots, 1$,

$$\beta_t(i) = \sum_j T_X(r_{t+1}, i; j) T_Y(r_{t+1}, j; y_{t+1}) \beta_{t+1}(j)$$

In the forwards pass, for each $t = 1, \ldots, T$, we compute the following quantities for $r = 1, \ldots, |R|$:

$$\begin{aligned}
\alpha_t(i, r) &\stackrel{\text{def}}{=} P(X_t = i|\mathbf{y}_{1:t}, \mathbf{r}_{1:t-1}, R_t = r) \\
L_t(r) &\stackrel{\text{def}}{=} P(y_t|\mathbf{y}_{1:t}, \mathbf{r}_{1:t-1}, R_t = r) \\
I_t(r) &\stackrel{\text{def}}{=} P(\mathbf{y}_{t+1:T}|\mathbf{y}_{1:t}, \mathbf{r}_{-t}, R_t = r) \\
q_t(r) &\stackrel{\text{def}}{=} P(R_t = r|\mathbf{y}_{1:T}, \mathbf{r}_{-t})
\end{aligned}$$

We then sample $r_t \sim q_t(r)$ and set

$$\alpha_t(i) \stackrel{\text{def}}{=} P(X_t = i|\mathbf{y}_{1:t}, \mathbf{r}_{1:t}) = \alpha_t(i, r_t)$$

The forwards algorithm is as follows. For $t = 1, \ldots, T$

1. Compute

$$\alpha_t(i,r) \propto \begin{cases} \pi(i,r)T_Y(r,i;y_t) & \text{if } t=1 \\ \left[\sum_j T_X(r,j;i)\alpha_{t-1}(j)\right] T_Y(r,i;y_t) & \text{if } 1 < t \leq T \end{cases}$$

   where $1/L_t(r)$ is the normalizing constant.

2. Compute $I_t(r) = \sum_i \beta_t(i)\alpha_t(i,r)$.

3. Compute

$$q_t(r) = \begin{cases} T_R(r;r_{t+1})L_t(r)I_t(r) & \text{if } t=1 \\ T_R(r_{t-1};r)T_R(r;r_{t+1})L_t(r)I_t(r) & \text{if } 2 \leq t < T \\ T_R(r_{t-1};r)L_t(r)I_t(r) & \text{if } t=T \end{cases}$$

4. Sample $r_t \sim q_t(r)$

5. Set $\alpha_t(i) = \alpha_t(i, r_t)$.

References

[CDLS99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[CK94] C. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

[CK96] C. Carter and R. Kohn. Markov chain Monte Carlo in conditionally Gaussian state space models. *Biometrika*, 83:589–601, 1996.

[DA99] A. Doucet and C. Andrieu. Iterative algorithms for state estimation of jump markov linear systems. Technical report, Cambridge Univ. Engineering Dept., 1999.

[Daw92] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[Lau92] S. L. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *JASA*, 87(420):1098–1108, December 1992.

[Mur98] K. P. Murphy. Filtering and smoothing in linear dynamical systems using the junction tree algorithm. Technical report, U.C. Berkeley, Dept. Comp. Sci, 1998.

[SHJ97] P. Smyth, D. Heckerman, and M. I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9(2):227–269, 1997.