Summary

- We propose a new forward-backward particle smoother with linear computational complexity.
- The new smoother is based on running a Sequential Monte Carlo sampler backwards in time after an initial forward filtering pass.
- We show through simulation studies that the new smoother can outperform existing smoothers when targeting the marginal smoothing densities.

Forward-Backward Particle Smoothers

Consider a general nonlinear state space model (SSM)

\[ x_{t+1} | x_t \sim f(x_{t+1} | x_t) \]
\[ y_t | x_t \sim g(y_t | x_t) \]

We are interested in inferring the marginal smoothing distribution, \( p(x_t | y_T) \). Forward-backward smoothers are based on sampling \( M \) backward paths after an initial forward particle filter, providing approximations of the filtering distributions \( p(x_t | y_t) \).

Main idea

Improve previous forward-backward smoothers that samples independent backward paths by using a Sequential Monte Carlo approach to exploit information gathered in other backward paths.

SMC smoother

We can express marginal smoothing distribution at time \( t \) as

\[
p(x_t | y_T) = \int p(x_t | y_{t+1}) p(x_{t+1} | y_T) dx_{t+1} \tag{1}
\]

By using the empirical approximation of the forward filtering density at time \( t \) and the marginal smoothing densities at time \( t + 1 \) and assuming that the backward particle system at time \( t + 1 \) is in the support of the forward filter system we can approximate (1) by

\[
\tilde{p}(x_t | y_T) \propto \sum_{i=1}^{N} \sum_{j=1}^{M} w_i^{t} \delta_j^{t} g(y_{t+1} | x_{t+1}^j) f(x_{t+1}^j | x_t) \tag{2}
\]

\[
\tilde{p}(x_{t+1} | y_{T+1}) = \begin{cases} w_{t+1}^j & \text{if } x_{t+1}^j = x_{t+1} \setminus x_{t+1}^j \neq x_{t+1} \end{cases} \tag{3}
\]

This allows us to target (2) using importance sampling similar in spirit to the auxiliary particle filter.

Backward SMC smoother

1. Sample \( M \) backward particles, \( \bar{x}_t^j \) from the forward particles at time \( T \), \( \{x_T, w_T^i\}_{i=1}^{N} \)
2. For \( t = T - 1, \ldots, 1 \)
3. For \( j = 1, \ldots, M \)
4. Sample \( a_t^j \sim \text{Cat}(\{w_t^i\}_{i=1}^{N}) \)
5. Sample \( b_t^j \sim \text{Cat}(\{w_{t+1}^i \delta_j^{t} g(y_{t+1} | x_{t+1}^i) f(x_{t+1}^i | x_t)\}_{i=1}^{N}) \)
6. Set \( \bar{x}_t^j = x_t^a \)
7. Set \( \tilde{w}_t^j = f(x_{t+1}^j | \bar{x}_t^j) \)
8. End For
9. Set \( \tilde{w}_t^j = \sum_{i=1}^{N} \tilde{w}_i^j \) for \( j = 1, \ldots, M \)
10. End For

Numerical Illustration

Consider a generic ten dimensional linear Gaussian SSM

\[ x_{t+1} | x_t \sim N(Ax_t, Q) \tag{4a} \]
\[ y_t | x_t \sim N(Cx_t, R) \tag{4b} \]

For this model 50 independent realizations of \( A \) and \( C \) were generated using the MATLAB command \( 	ext{drss}(10,10,0) \), and \( Q \) and \( R \) were set to the identity matrix.

<table>
<thead>
<tr>
<th>FFBSi</th>
<th>RS-FFBSi</th>
<th>MH-FFBSi</th>
<th>Proposed smoother</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>0.06</td>
<td>0.66</td>
<td>0.66</td>
</tr>
<tr>
<td>Runtime</td>
<td>2.02</td>
<td>1.75</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Average mean squared error compared to the Rauch-Tung-Striebel smoother and runtime for linear Gaussian model (4), using \( N = 200 \) forward filtering particles and \( M = 100 \) backward particles.

Although the new smoother performs well in practice with reasonable particles counts, we have found that it exhibits a small non-vanishing bias. In our current efforts we are investigating methods to quantify and eliminate this bias.

More information and source code

http://vcl.itn.liu.se/publications/2014/KBD14/