Filtering, Smothing and Parameter Estimation for General State Space Models

Hans R. Künsch

Seminar für Statistik, ETH Zürich

EPFL, June 30, 2006.

Contents

1. State space and hidden Markov models: Introduction

A **general state space model** consists of an unobserved state sequence (X_t) and an observation sequence (Y_t) with the following properties:

State evolution: $X_0, X_1, X_2, \ldots$ is a Markov chain with $X_0 \sim a_0(x)d\mu(x)$ and

$$X_t \mid X_{t-1} = x_{t-1} \sim a_t(x_{t-1}, x)d\mu(x)$$

Generation of observations: Conditionally on (X_t), the Y_t's are independent and Y_t depends on X_t only with

$$Y_t \mid X_t = x_t \sim b_t(x_t, y)d\nu(y).$$

Some Notation

The term **Hidden markov model** is sometimes used as a synonym for state space model, but other people reserve it for the case where the states are discrete.

For any $s < t$, we define $Y_{s:t} = (Y_s, Y_{s+1}, \ldots Y_t)$ and similarly $X_{s:t}$.

In general, we use p as the generic symbol for a (conditional) density of its arguments, but we reserve the symbol a_t for $p(x_t \mid x_{t-1})$ and b_t for $p(y_t \mid x_t)$.

Graphical representation of state space models

The dependence between the variables of a state space model can be represented as follows

$$\begin{array}{ccccccccc} \ldots & \rightarrow & X_{t-1} & \rightarrow & X_t & \rightarrow & X_{t+1} & \rightarrow & \ldots \\ & & \downarrow & & \downarrow & & \downarrow & & \\ \ldots & & Y_{t-1} & & Y_t & & Y_{t+1} & & \ldots \end{array}$$

It implies various conditional independencies, which we will use when we treat the smoothing problem.

Extensions with additional dependencies (e.g. second order Markovian states) can be handled either by enlarging the state space, or directly by modifying the formulae.

History:

Origin in the 1960's in engineering (Kalman-Bucy, Baum-Welch).

Recognized in time series analysis in 1980's (Akaike, Hannan, Harvey).

Large interest in 1990's (Molecular biology, Monte Carlo methods).

General references:

J. Durbin and S. J. Koopman, Time Series Analysis by State Space Methods, Oxford 2001

H.-R. K., Chapter 3 in Complex Stochastic Systems, Barndorff-Nielsen et al., eds., Chapman and Hall (2001).

A. Doucet, N. de Freitas and N. Gordon (eds.), Sequential Monte Carlo Methods in Practice, Springer (2001).

P. Del Moral, Feynman-Kac Formulae, Springer 2004.

O. Cappé, E. Moulines, T. Rydén, Inference in Hidden Markov Models, Springer 2005.

2. Examples of state space models

Depending on the application, states can be interpreted as latent variables or as time varying parameters.

An alternative representation of a general state space model is

$$X_t = F_t(X_{t-1}, U_t), \quad Y_t = H_t(X_t, V_t)$$

where (U_t) and (V_t) are two independent white noises and F_t and H_t arbitrary functions. This indicates the flexibility of the model.

A linear state space model has the form

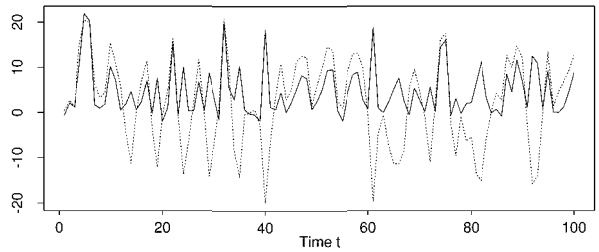
$$X_t = F_t X_{t-1} + U_t, \quad Y_t = H_t X_t + V_t$$

where now F_t and H_t are matrices.

As an example, the figure below shows a simulation of the model

$$X_t = \alpha X_{t-1} + \beta \frac{X_{t-1}}{1 + X_{t-1}^2} + \gamma \cos(1.2t) + V_t, \quad Y_t = \frac{X_t^2}{20} + W_t.$$

which goes back to Andrade Netto et al., IEEE Trans. Autom. Control (1978). Y_t carries no information about the sign of X_t.



ARMA models as state space models

A stationary Gaussian ARMA(p, q) process (Y_t) can be represented as a linear state space model by defining the $k = \max(p, q + 1)$-dimensional state vector

$$X_t = (Y_t, Y_{t+1|t}, \dots Y_{t+k-1|t})^T.$$

Here $Y_{t|s}$ the conditional expectation of Y_t given Y_u for all $u \leq s$. The state equation follows because with suitably defined coefficients g_j

$$\begin{aligned} Y_{t+j|t+1} &= Y_{t+j|t} + g_j(Y_{t+1} - Y_{t+1|t}) \quad (j = 1, \dots, k), \\ Y_{t+k|t} &= \sum_{j=1}^{p} \phi_j Y_{t+k-j|t}. \end{aligned}$$

This can be extended to ARIMA models.

Structural time series

$$Y_t = T_t + S_t + V_t$$

where T_t is a (stochastic) trend, S_t a (stochastic) seasonal component and V_t the irregular part. The trend is modeled as locally constant

$$T_t = T_{t-1} + U_t$$

or locally linear

$$T_t = T_{t-1} + M_{t-1} + U_t^{(1)}, \quad M_t = M_{t-1} + U_t^{(2)}.$$

The seasonal part is modeled in the time domain as

$$S_t = -\sum_{j=1}^{s-1} S_{t-j} + U_t^{(3)},$$

(or alternatively in the frequency domain).

We then have a state space model with

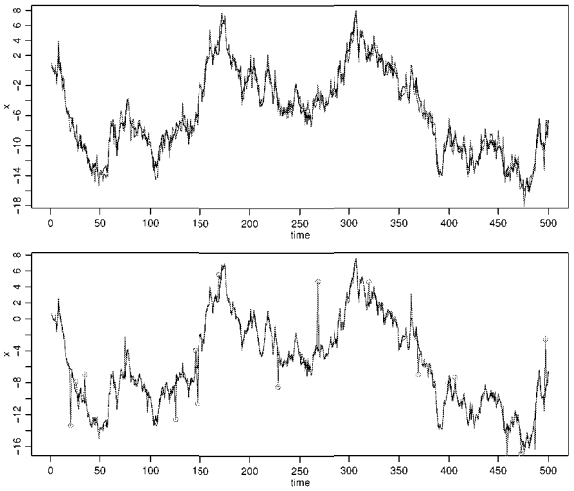
$$X_t = (T_t, M_t, S_t, \dots, S_{t-s+2})^T,$$

and suitable definitions of F and H. It is easy to see that this is a seasonal ARIMA-model with $d = 2, D = 1$ $p = 0$ and $q = s + 1$.

All this can be extended to the multivariate case. Gaussian ARIMA models and linear Gaussian state space models are equivalent.

By taking heavy-tailed noise distributions, we can model level shifts, innovation outliers and observation outliers. This is important for robust time series analysis.

Random walk with Gaussian and non-Gaussian noise with the same variance.



Dynamic generalized linear models are an extension of structural time series models. The definition of the state vector and its evolution remains the same, but the observation density b_t is now an exponential family:

$$b_t(x_t, y_t) = \exp(y_t \cdot Hx_t - c(Hx_t))h(y_t)$$

where $HX_t = T_t + S_t$. In particular, this can be used to model count data as conditionally Poisson or binomial.

Stochastic volatility models

Here the Y_t's are the log returns of some asset, and the X_t's are exogenous stochastic volatilities. The simplest model is

$$X_t = m + \phi X_{t-1} + U_t, \quad Y_t = \exp(X_t/2)\, V_t,$$

where (U_t) and (V_t) are independent Gaussian white noises. By considering $\log|Y_t|$ we obtain a linear, non-Gaussian state space model, but the noise is strongly non-Gaussian and this should not be ignored in the analysis.

A continuous time model has the following form

$$\begin{aligned} d\sigma^2(t) &= a(\sigma^2(t))dt + b(\sigma^2(t))dW(t) \\ dY(t) &= (\mu + \beta\sigma^2(t))dt + \sigma(t)dB(t). \end{aligned}$$

Here, $Y(t)$ is the log price process, $\sigma^2(t)$ is the unobserved volatility, and W and B are two independent Brownian motions. We observe Y at discrete time points $t_1 < t_2 < \ldots < t_n$.

If we condition on the volatility $(\sigma^2(t)$,

$$Y(t_i) - Y(t_{i-1}) \sim \mathcal{N}(\mu(t_i - t_{i-1}) + \beta v_i^2, v_i^2),$$

where

$$v_i^2 = \int_{t_{i-1}}^{t_i} \sigma^2(t)dt.$$

Hence this is a state space model if we define the discrete time state variable to be $X_i = (\sigma^2(t_i), v_i)$.

State space models in biology:

Ion channels are proteins located in the cell membrane that can be open or closed. The simplest model has two different means depending on whether the channel is open or closed plus an additive noise. A more realistic model allows for several internally different states of the channel, colored noise with state dependent variance and the effect of filters (de Gunst et al., 2001):

$$Y_t = \sum_{k=-r}^{r} \gamma_k \mu(X_{t-k}) + C_t + \sigma(X_t)\delta_t$$

Here C_t is a Gaussian AR process, δ_t is white noise and X_t is a continuous time Markov chain with the following structure:

$$\begin{array}{ccccccccccc} & & 4q_1 & & 3q_1 & & 2q_1 & & q_1 & & q_3 & \\ C_1 & \rightleftharpoons & C_2 & \rightleftharpoons & C_3 & \rightleftharpoons & C_4 & \rightleftharpoons & O & \rightleftharpoons & C_5 \\ & & q_2 & & 2q_2 & & 3q_2 & & 4q_2 & & q_4 & \end{array}$$

DNA and protein sequences: One of the simplest examples are CG-islands. A homogeneous Markov chain for the sequence of letters from the DNA alphabet $\{A, C, G, T\}$ is not adequate since the frequencies of letters and transition changes. In CG islands the pairs CG occur more frequently. Two different transition regimes P_0 and P_1 can be obtained by letting the state take values in $\{A, C, G, T\} \times \{0, 1\}$ with transition matrix

$$\begin{pmatrix} (1-\varepsilon)P_0 & \varepsilon Q \\ \eta Q & (1-\eta)P_1 \end{pmatrix}.$$

ε and η are the probabilities for switching between the two regimes.

Engineering:

Many examples with **tracking** and **control** applications can be found in Doucet et al. (2001).

Usually, state = position and velocity of a moving object (satellite, robot etc.). Observations = partial, noisy information about the state, e.g. position only or angle only. State transitions usually depend also on a control variable u_t which is a function of $y_{1:t-1}$.

One of the earliest uses of hidden Markov models is in **speech analysis**. In isolated word recognition, one has a hidden Markov model for each word, the states being the different phonems or different stages of the phonems. Transition probabilities satisfy $a(i, j) = 0$ for $j < i$. Observations = features extracted from the acoustic signal over short overlapping intervals. One then computes the likelihood $p(y_{1:t})$ for each word and applies Bayes rule.

State space models in geophysics:

Rainfall networks (Guttorp et al.): Observations = daily amount of rainfall at a network of stations; states = weather types. Given the state, the rainfalls at different stations are independent and Gamma-distributed with an additional atom at zero.

Data assimilation in numerical weather prediction: State vector = 7 atmospheric variables (wind, density, potential temperature, pressure temperature) on a spatial grid, containing about 10^7 elements. State transitions are determinstic and are obtained by a numerical scheme for a partial differential equation. Filtering is used to combine forecasts with actual observations to obtain initial conditions for next integration period (usually 6h). Number of observations at each time point $> 10^6$.

Partially observed diffusion models

In various areas one considersa stochastic differential equation

$$dX_t = f(X_t)dt + \sigma(X_t)dW_t, \quad Y_k = h(X_{t_k}) + V_k$$

where W is a Brownian motion and (V_k) an independent white noise.

For interest rates, this includes for instance the Cox-Ingersoll-Ross model. In data assimilation, the additional noise term reflects model deficiencies. A popular toy example in this area is the **double well** model which has

$$f(x) = -\frac{d}{dx}(x^2 - 1)^2, \quad \sigma(x) \equiv \sigma.$$

The density of the invariant distribution is proportional to $\exp(-2(x^2 - 1)^2)/\sigma^2)$ and is therefore bimodal.

Another justification of such a diffusion model are time varying parameters. Assume

$$f(x) = f_0(x) + \sum_j \beta_j f_j(x)$$

where β_j are some parameters, e.g. a growth or reaction rate. With time varying parameters

$$\beta_j(t) = \bar{\beta}_j + \text{ "white noise",}$$

X_t becomes a diffusion if we interprete "white noise" as the "derivative" of a Brownian motion.

However, the highly irregular fluctuations in such a model somehow contradict biological intuition. An alternative is to model $\beta_j(t)$ as mean-reverting Ornstein-Uhlenbeck processes.

3. Filtering and smoothing

The main tasks we need to solve are

1. Inference about the states based on a stretch of observed values $y_{s:t}$ for a given model (i.e. a_t and b_t known).

2. Inference about unknown parameters in $a_t,\ b_t$.

Inference about X_s given $y_{1:t}$ is called **prediction** if $s > t$, **filtering** if $s = t$ and **smoothing** if $s < t$. We use the special symbol $f_{s|t}$ for $p(x_s|y_{1:t})$.

We begin with the filtering problem, and later on we will discuss smoothing and parameter estimation. Difficulty increases in this order.

It is clear that the joint density of $(X_{0:t}, Y_{1:t})$ is given by

$$p(x_{0:t}, y_{1:t}) = a_0(x_0) \prod_{s=1}^{t} a_s(x_{s-1}, x_s) b_s(x_s, y_s).$$

In particular (X_t, Y_t) is also a Markov process.

The joint density of the observations $y_{1:t}$ follows by integration, but this is difficult or impossible to compute. We see that the observations alone are not Markovian.

The conditional density $p(x_{0:t} \mid y_{1:t})$ is proportional to the joint density $p(x_{0:t}, y_{1:t})$. This implies that conditionally on $y_{1:t}$ the state variables are still Markovian (because the density factors into a product containing only pairs (x_{s-1}, x_s)). The densities $f_{s|t}$ can be obtained in principle from this by integration, but again this is not a practical way to proceed.

3.1 Filtering

With a recursive procedure, we can break down the multiple integrals that occur in $f_{s|t}$ in a series of lower-dimensional integrals.

From filter to prediction density (**Propagation**):

$$f_{t|t-1}(x_t \mid y_{1:t-1}) = \int f_{t-1|t-1}(x \mid y_{1:t-1}) a_t(x, x_t) d\mu(x).$$

This follows from the law of total probability and the fact that x_t is conditionally independent of $y_{1:t-1}$ given x_{t-1}.

Similarly, we can go from $f_{s|t}$ to $f_{s+1|t}$ for any $s > t$.

From prediction to filter density (**Update**):

$$\begin{aligned} f_{t|t}(x_t \mid y_{1:t}) &= \frac{f_{t|t-1}(x_t \mid y_{1:t-1}) b_t(x_t, y_t)}{p(y_t \mid y_{1:t-1})} \\ &\propto f_{t|t-1}(x_t \mid y_{1:t-1}) b_t(x_t, y_t). \end{aligned}$$

This follows from Bayes rule and the fact that y_t is conditionally independent of $y_{1:t-1}$ given x_t.

Combining the two steps, we have

$$f_{t|t}(x_t \mid y_{1:t}) \propto \int f_{t-1|t-1}(x \mid y_{1:t-1}) a_t(x, x_t) d\mu(x)\, b_t(x_t, y_t).$$

Because the denominator is just a normalization:

$$p(y_t \mid y_{1:t-1}) = \int f_{t|t-1}(x \mid y_{1:t-1}) b_t(x, y_t) d\mu(x).$$

3.2 Smoothing

Similarly, we have recursions for the conditional densities of all states up to time t:

Propagation:

$$p(x_{0:t} \mid y_{1:t-1}) = a_t(x_{t-1}, x_t) p(x_{0:t-1} \mid y_{1:t-1}).$$

Update:

$$p(x_{0:t} \mid y_{1:t}) = \frac{b_t(x_t, y_t) p(x_{0:t} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}.$$

Combining the two steps

$$p(x_{0:t} \mid y_{1:t}) \propto b_t(x_t, y_t) a_t(x_{t-1}, x_t) p(x_{0:t-1} \mid y_{1:t-1}).$$

Instead of the recursion with increasing dimension from the last slide, use that given $y_{1:t}$ the state process is still a Markov chain (a fact we have seen before).

The forward transition densities of this conditional chain are

$$\begin{aligned} p(x_s \mid x_{s-1}, y_{1:t}) &= p(x_s \mid x_{s-1}, y_{s:t}) \\ &= \frac{a_s(x_{s-1}, x_s) b_s(x_s, y_s) p(y_{s+1:t} \mid x_s)}{p(y_{s:t} \mid x_{s-1})}. \end{aligned}$$

The backward transition densities of this conditional chain are

$$\begin{aligned} p(x_s \mid x_{s+1}, y_{1:t}) &= p(x_s \mid x_{s+1}, y_{1:s}) \\ &= \frac{a_{s+1}(x_s, x_{s+1}) f_{s|s}(x_s \mid y_{1:s})}{f_{s+1|s}(x_{s+1} \mid y_{1:s})}. \end{aligned}$$

Forward computation - backward simulation

Assume that we can compute the filter densities $f_{s|s}$ for all $s \leq t$. Then we can simulate from the conditional density $p(x_{0:t} \mid y_{1:t})$ by generating X_t according to $f_{t|t}$ and then simulating recursively X_s given x_{s+1} according to the density proportional to

$$a_{s+1}(x_s, x_{s+1}) f_{s|s}(x_s \mid y_{1:s}).$$

This is an exact algorithm, there is no need to iterate.

Computation of smoothing marginals

We obtain $f_{s|t}$ from $f_{s|s}$ by incorporating the additional information $y_{s+1:t}$ with Bayes formula:

$$\begin{aligned} f_{s|t}(x_s \mid y_{1:t}) &= \frac{p(y_{s+1:t} \mid x_s, y_{1:s}) p(x_s \mid y_{1:s})}{p(y_{s+1:t} \mid y_{1:s})} \\ &= \frac{p(y_{s+1:t} \mid x_s)}{p(y_{s+1:t} \mid y_{1:s})} f_{s|s}(x_s \mid y_{1:s}). \end{aligned}$$

The ratio

$$r_{s|t}(x_s, y_{1:t}) = \frac{p(y_{s+1:t} \mid x_s)}{p(y_{s+1:t} \mid y_{1:s})} = \frac{f_{s|t}(x_s \mid y_{1:t})}{f_{s|s}(x_s \mid y_{1:s})}$$

satisfies the backward recursion

$$r_{s-1|t}(x_{s-1}, y_{1:t}) = \frac{\int a_s(x_{s-1}, x_s) b_s(x_s, y_s) r_{s|t}(x_s, y_{1:t}) d\mu(x_s)}{p(y_s \mid y_{1:s-1})}.$$

Moreover, $r_{s|t}$ is also useful for the forward transitions

$$p(x_s \mid x_{s-1}, y_{s:t}) = a_s(x_{s-1}, x_s) b_s(x_s, y_s) \frac{p(y_{s+1:t} \mid x_s)}{p(y_{s:t} \mid x_{s-1})}$$

because

$$\frac{p(y_{s+1:t} \mid x_s)}{p(y_{s:t} \mid x_{s-1})} = \frac{r_{s|t}(x_s, y_{1:t})}{r_{s-1|t}(x_{s-1}, y_{1:t})} \frac{1}{p(y_s | y_{s-1})}.$$

Hence in order to compute low-dimensional marginals of the smoothing distribution, we compute $f_{s|s}$ and $p(y_s \mid y_{1:s-1})$ by a forward filter recursion, and the ratio $r_{s|t}$ by a backward recursion.

3.3 Likelihood

Assume now that both a_s and b_s depend on a finite dimensional parameter θ. Then the likelihood of θ given the observed series $y_{1:T}$ is

$$p(y_{1:t} \mid \theta) = \prod_{t=1}^{T} p(y_i \mid y_{1:t-1}, \theta).$$

Each factor on the right is obtained as a normalization during the filter recursion.

Maximization can be done by a general purpose optimization algorithm.

The EM algorithm

Since the joint likelihood of the observations and the hidden states can be written explicitely, using the EM algorithm is natural (It was developed by Baum and Welch, several years before the Dempster et al. paper).

In the E-step, we have to compute

$$Q(\theta, \theta') = \mathbf{E}\left[\log p(x_{0:T}, y_{1:T}; \theta) \mid y_{1:T}, \theta'\right]$$

where θ' is the current approximation to the MLE. In the M-step we maximize $Q(\theta, \theta')$ with respect to θ. The maximizer becomes our new current approximation. It can be shown that in each cycle the likelihood increases.

The function $Q(\theta, \theta')$ contains the terms

$$\mathbf{E}\left[\log a_t(x_t - 1, x_t; \theta) \mid y_{1:T}, \theta'\right]$$

and

$$\mathbf{E}\left[\log b_t(x_t, y_t; \theta) \mid y_{1:T}, \theta'\right].$$

Hence for the EM-algorithm we need the smoothing densities for consecutive pairs and singletons of state variables.

4. Exact computations

Discrete states

If X_t is **discrete** with M possible values, integrals are sums. We obtain the forward-backward algorithm due to Baum and Welch.

The filter recursion is

$$f_{t+1|t+1}(j) \propto \sum_{k=1}^{M} f_{t|t}(k) a_{t+1}(k, j) b_{t+1}(j, y_{t+1})$$

(because the observations are fixed, we drop them in the filter densities). This is of the form row vector times a matrix, followed by elementwise multiplication of two vectors and a summation for normalization.

Each step of this recursion needs roughly $O(M^2)$ operations. Thus the complexity of the whole filter is $O(TM^2)$.

Similarly, the recursion for $r_{s|t}$ is

$$r_{s-1|t}(j) = \frac{1}{p(y_s \mid y_{1:s-1})} \sum_k a_s(j, k) b_s(k, y_s) r_{s|t}(k).$$

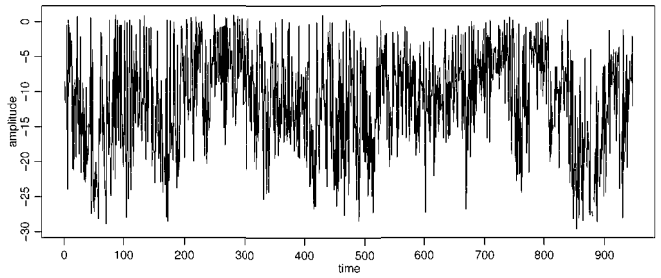
This is the forward-backward algorithm due to Baum and Welch. It has the complexity $O(TM^2)$.

For forward computation - backward simulation, we use

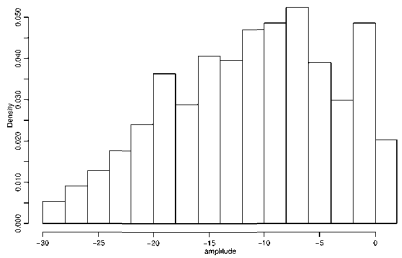
$$P[X_s = k \mid x_{s+1} = j, y_{s+2:T}] \propto a_{s+1}(k, j) f_{s|s}(k).$$

Typically, generating N values of the whole smoothing distribution has complexity $O(T(N + M^2))$.

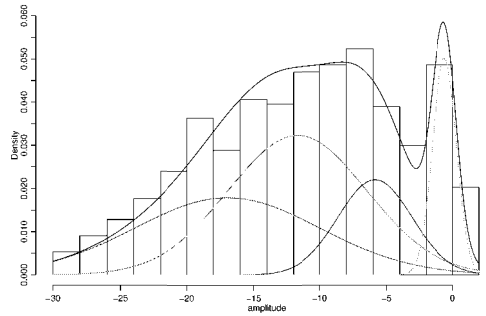
Example: Excited potentials in a nerve celle (data by C. Stricker, Neuroinformatik). Measurements of the current reaching a single nerve cell. This cell is connected via synapses with a small number of other cells which are excited every second.



One hypothesis is that this current is a superposition of a small number of unit currents. This leads to a (normal) mixture model for the measurements. The data are dependent, so we fit a normal mixture model with a Markovian regime.



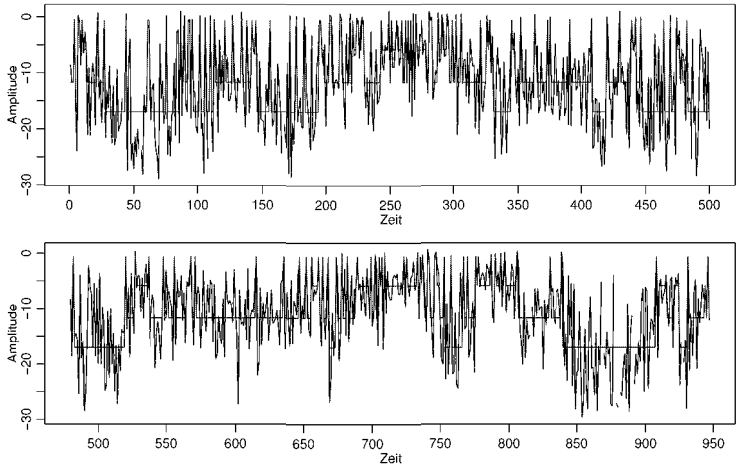
Fitted mixture with 4 components (using the EM-algorithm)



The estimated transition matrix between the regimes is

$$\begin{pmatrix} 0.913 & 0.002 & 0.000 & 0.085 \\ 0.000 & 0.826 & 0.039 & 0.135 \\ 0.027 & 0.137 & 0.699 & 0.137 \\ 0.201 & 0.458 & 0.259 & 0.081 \end{pmatrix}$$

Observations together with the mean of the most likely regime.



Linear Gaussian models

For a linear state space model

$$X_t = F_t X_{t-1} + U_t, \quad Y_t = H_t X_t + V_t$$

with Gaussian noises, all $f_{s|t}$ are Gaussian. Their means $m_{s|t}$ and variances $R_{s|t}$ follow from the general recursions with the help of some linear algebra.

For the filter, we obtain the famous Kalman filter

$$\begin{aligned} m_{t|t-1} &= F_t m_{t-1|t-1} \\ m_{t|t} &= m_{t|t-1} + K_t(y_t - H_t m_{t|t-1}) \\ R_{t|t-1} &= E(U_t U_t^T) + F_t R_{t-1|t-1} F_t^T, \\ R_{t|t} &= R_{t|t-1} - K_t H_t R_{t|t-1}, \end{aligned}$$

where the so-called gain matrix is

$$K_t = R_{t|t-1} H_t^T (E(V_t V_t^T) + H_t R_{t|t-1} H_t^T)^{-1}.$$

For the smoothing means and variances, we obtain

$$\begin{aligned} m_{s|t} &= m_{s|s} + \overline{K}_s(m_{s+1|t} - m_{s+1|s}) \\ R_{s|t} &= R_{s|s} - \overline{K}_{s+1}(R_{s+1|s} - R_{s+1|t})\overline{K}_{s+1}^T \end{aligned}$$

where

$$\overline{K}_{s+1} = R_{s|s} F_{s+1}^T R_{s+1|s}^{-1}.$$

There are many equivalent forms of the smoother in the literature. They differ numerically with respect to speed and accuracy.

For the forward computation - backward simulation algorithm, we use that conditional on x_{s+1} and $y_{1:t}$, X_s is Gaussian with mean $\overline{m}_s$ and covariance matrix $\overline{R}_s$ where

$$\begin{aligned} \overline{m}_s &= m_{s|s} + \overline{K}_{s+1}(x_{s+1} - m_{s+1|s}), \\ \overline{R}_s &= (F_{s+1}^T E[U_{s+1} U_{s+1}^T]^{-1} F_{s+1} + R_{s|s}^{-1})^{-1} \\ &= R_{s|s} - \overline{K}_{s+1} F_{s+1} R_{s|s}. \end{aligned}$$

Extended Kalman filter and smoother

In practically all other cases, the recursions are difficult to compute. In linear state space models with non-Gaussian noises, the Kalman filter mean is still the best *linear* unbiased estimator of the state. However, nonlinear estimators can be much better.

In engineering, the extended Kalman filter is by far the most popular approximation. It linearizes the system and then applies the Kalman filter. For instance, the linearization for the prediction step is

$$\begin{aligned} X_t &= F_t(X_{t-1}, U_t) \approx F_t(\hat{x}_{t-1|t-1}, 0) \\ &+ \left.\frac{\partial F_t(x,u)}{\partial x}\right|_{(\hat{x}_{t-1|t-1},0)} (X_{t-1} - \hat{x}_{t-1|t-1}) \\ &+ \left.\frac{\partial F_t(x,u)}{\partial u}\right|_{(\hat{x}_{t-1|t-1},0)} U_t. \end{aligned}$$

The extended Kalman filter works reasonably in many, but not in all situations. The disadvantages are that error bounds are extremely difficult to produce and that we cannot reduce the error by a better (although more complicated) approximation. In addition, the method gives no information about the conditional distributions which can be very non-Gaussian.

Numerical integration is problematic in high dimensions. We do not know in advance where the filter density has its main mass, adn thus it is difficult to construct a reasonable grid in advance.

Much current interest focuses on Monte Carlo methods. These will be discussed in the remainder.

5. MCMC algorithms

Standard Markov chain Monte Carlo methods can be used to simulate from the smoothing density $p(x_{0:t} \mid y_{1:t})$ which is known up to normalization. The filter then follows by marginalisation. It is clear that a recursive implementation is not possible: With each new observation one has to start in principle from scratch. Still, for off-line problems this is often useful.

We can apply a single site Gibbs or Metropolis-Hastings sampler. The Gibbs sampler starts with an arbitrary series $x_{0:t}^{(0)}$ and then changes one component at a time according to the full conditionals that were given on a previous slide. If we cannot sample directly from the full conditionals, we still can use the Metropolis-Hastings recipe.

However, in most applications, this chain mixes extremely slowly. If we know x_{t-1} y_t and x_{t+1} then X_t is often determined almost completely, and thus the changes at each step are too small.

Static MCMC: blockwise updates

Large gains are possible in the following situation. Assume that each x_t consists of two components (ξ_t, η_t), and that we can sample from $p(\xi_{0:t} \mid \eta_{0:t}, y_{1:t})$ and also from $p(\eta_{0:t} \mid \xi_{0:t}, y_{1:t})$. Then obviously we can iteratively update $\xi_{0:t}$ and $\eta_{0:t}$, always keeping the other component fixed.

The assumption is fulfilled if conditional on $\xi_{0:t}$, (η_s, Y_s) is either a linear Gaussian state space model or a hidden Markov model (and the same holds if we exchange ξ and η). Then we can use a forward computation backward simulation algorithm. Such models are called partial non-Gaussian models.

Examples

Gaussian AR with t-distributed observation noise:

Write the t-distribution as scale mixture of Gaussian distributions:

$$Y_t = X_t + \frac{V_t}{\sqrt{Z_t/\nu}}$$

where V_t is standard normal and $Z_t \sim \Gamma(\nu, 1)$. By including Z_t among the states, we obtain a different representation as a state space model. This representation belongs to the class of partial non-Gaussian models: Conditionally on (Z_t), it is a linear Gaussian state space model, and conditionally on (X_t) we have independence.

In this model, the filter density $f_{t|t}$ can be bimodal when there is ambiguity whether an observation is due to an observation outlier or a large innovationl

The same idea has been used for stochastic volatility models. Taking log's, the observation equation is

$$\log(Y_t^2) = X_t + \log(V_t^2),$$

and thus one has to approximate the log of a chisquare(1) by a mixture of normals.

Unknown parameters

With MCMC, it is straightforward to do Bayesian inference about the states and unknown parameters θ at the same time. One simply iterates between sampling from the density of θ given $x_{0:t}$ and $y_{1:t}$ and sampling from the density of $x_{0:t}$ given θ and $y_{1:t}$. The former usually presents no difficulty, and the latter has been discussed before. However, the additional component can make the convergence even slower.

Example: Ion channels (de Gunst et al.).

$$Y_t = \sum_{k=-1}^{1} \gamma_k \mu(X_{t-k}) + C_t + \sigma(X_t)\delta_t$$

We alternate between 3 blocks of updates, of (X_t) given (Y_t, C_t, θ), of (C_t) given (Y_t, X_t, θ) and of θ given (X_t, Y_t, C_t).

6. Particle filters

Importance (re)sampling and particle filtering

The passage from $f_{t|t}$ to $f_{t+1|t}$ is complicated analytically, but usually simple to implement by sampling: If $(x_{t,j})$ is a sample from $f_{t|t}$, then

$$z_{t+1,j} \sim a_{t+1}(x_{t,j}, z)d\mu(z) \quad (j = 1, \ldots, N)$$

is a sample from $f_{t+1|t}$. In order to generate a sample from $f_{t+1|t+1}$ and thus to close the recursion, we need a sampling implementation of Bayes rule.

Dropping the time index for a moment, this is the following problem: Given a sample $(\tilde{x}_j)$ from the prior f_{prior}, generate a sample from the posterior

$$f_{\text{post}}(x) \propto f_{\text{prior}}(x)b(x).$$

Importance sampling uses the same $\tilde{x}_j$'s with weights

$$\lambda_j = \frac{b(\tilde{x}_j)}{\sum_k b(\tilde{x}_k)}.$$

Sampling importance resampling (Rubin, 1988) takes a sample from $\tilde{x}_1, \ldots, \tilde{x}_N$ with probabilities λ_j:

$$x_j = \tilde{x}_{I_j}, \quad P[I_j = k] = \lambda_k.$$

Combining this, we obtain the standard particle filter

1. Generate $(x_{0,1}, \ldots, x_{0,N})$ from $a_0(x)d\mu(x)$ and set $t = 0$.
2. Generate $\tilde{x}_{t+1,j} \sim a_{t+1}(x_{t,j}, x)d\mu(x)$.
3. Compute weights

$$\lambda_{t+1,j} = \frac{b_{t+1}(\tilde{x}_{t+1,j}, y_{t+1})}{\sum_k b_{t+1}(\tilde{x}_{t,k}, y_{t+1})}.$$

4. Generate I_j with $P[I_j = k] = \lambda_{t+1,k}$ and set $x_{t+1,j} = \tilde{x}_{t+1,I_j}$.
5. Increase t by 1 and go back to 2.

This is extremely simple to implement and very intuitive. We consider the $x_{t,j}$'s as particles who live in the state space. They move according to the dynamics of the state process, and then there is a selection step to take into account the next observation: Particles who do not fit to the next observation die (with high probability), those who fit have several descendants which move independently in the next step.

A simple implementation is shown in R !

Disadvantage: If the next observation is highly informative, most of the particles will not fit. The weights $\lambda_{t+1,j}$ become very unbalanced, and there are many ties among the $x_{t+1,j}$.

The most general form of the particle filter

Here, we consider weighted samples $(x_{t,j}, \lambda_{t,j})$ as approximations of $f_{t|t}$:

$$f_{t|t} \approx \sum_{j=1}^{N} \lambda_{t,j}\Delta(x_{t,j})$$

where $\Delta(x)$ is the point mass (Dirac function) at x.

Inserting the approximation into the recursion, we obtain

$$f_{t+1|t+1}(x) \approx f^N_{t+1|t+1}(x) \propto b_{t+1}(x, y_{t+1}) \sum_{j=1}^{N} \lambda_{t,j} a_{t+1}(x_{t,j}, x)$$

($f^N_{t+1|t+1}$ is the exact filter density at time $t+1$, if the approximation at time t is exact). Hence we need to generate a sample from $f^N_{t+1|t+1}$.

One obtains a whole class of algorithms to sample from $f^N_{t+1|t+1}$ by considering $f^N_{t+1|t+1}$ as the marginal of the following joint distribution of an index j and a state variable x:

$$\pi_{t+1}(j,x) \propto \lambda_{t,j} b_{t+1}(x, y_{t+1}) a_{t+1}(x_{t,j}, x).$$

This is the auxiliary variables idea of Pitt and Shephard (1999).

Importance sampling from π_{t+1} selects pairs (j,x) from a proposal distribution $\tau_j \rho(j,x) d\mu(x)$ and attaches weights porportional to

$$\frac{\pi_{t+1}(j,x)}{\tau_j \rho(j,x)}$$

to each pair. Finally, we discard the indices and keep only the state values and the weights.

Thus the general version of the particle filter is

1. Generate $(x_{0,1}, \ldots, x_{0,N})$ from $a_0(x)d\mu(x)$ and set $\lambda_{0,j} = \frac{1}{N}$ and $t = 0$.
2. Choose probabilities (τ_k) for the index and generate I_j with $P[I_j = k] = \tau_k$.
3. Choose a transition density $\rho(x, x')$ and generate $x_{t+1,j} \sim \rho(x_{t,I_j}, x) d\mu(x)$.
4. Compute weights

$$\lambda_{t+1,j} \propto \frac{\lambda_{t,I_j} b_{t+1}(x_{t+1,j}, y_{t+1}) a_{t+1}(x_{t,I_j}, x_{t+1,j})}{\tau_{I_j} \rho(x_{t,I_j}, x_{t+1,j})}.$$

5. Increase t by 1 and go back to 2.

With $\tau_j = \lambda_{t,j}$ and $\rho(x,x') = a_{t+1}(x,x')$, this is the same algorithm as before: The sampling step now occurs at the beginning of an iteration (in step 2) instead of at the end.

The choice of the proposal

An algorithm that is even simpler than the standard particle filter is obtained by choosing $\tau_j \equiv \frac{1}{N}$. Then the sampling step 2 is not needed, we can simply take $I_j = j$, i.e. we select each particle exactly once. Hence this amounts to propagate particles independently and to multiply and renormalize at each time step the importance weights. The algorithm is therefore called **sequential importance sampling**. However, this is in most cases a poor algorithm since the weights become quickly unbalanced. In the end, all the weight is carried by one or a few particles.

The sampling step is needed to avoid this sample depletion. The advantage of the general form above is that the components τ and ρ of the proposal distribution can take y_{t+1} into account. The goal is to choose them in such a way that the new weights $\lambda_{t+1,j}$ are as equal as possible.

We can make all $\lambda_{t+1,j}$ equal to $\frac{1}{N}$ by choosing

$$\tau_k \propto \lambda_{t,k} p(y_{t+1} \mid x_{t,k}) = \lambda_{t,k} \int a_{t+1}(x_{t,k}, x_{t+1}) b_{t+1}(x_{t+1}, y_{t+1}) d\mu(x_{t+1})$$

and

$$\rho(x_t, x_{t+1}) = p(x_{t+1} \mid x_t, y_{t+1}) = \frac{a_{t+1}(x_t, x_{t+1}) b_{t+1}(x_{t+1}, y_{t+1})}{p(y_{t+1} \mid x_t)}.$$

An example where we can compute this "ideal" proposal is

$$X_{t+1} = f(X_t) + U_t, \quad Y_{t+1} = HX_{t+1} + V_t$$

with Gaussian noises U_t and V_t and arbitrary nonlinear conditional mean $f(x_t)$. Then both $\log a_{t+1}(x_t, x_{t+1}$ and $\log b_{t+1}(x_{t+1}, y_{t+1})$ are quadratic functions in x_{t+1} and thus $a_{t+1}(x_t, x_{t+1}) b_{t+1}(x_{t+1}, y_{t+1})$ is proportional to a Gaussian distribution.

If $a_{t+1}(x_t, .)$ is close to a point mass, then we can construct a proposal as follows:

$$p(y_{t+1} \mid x_t) \approx b_{t+1}(\mu(x_t), y_{t+1})$$

where $\mu(x_t)$ is the mean or median of $a_{t+1}(x_t, .)$ implies

$$p(x_{t+1} \mid x_t, y_{t+1}) \approx a_{t+1}(x_{t+1}, x_t).$$

Hence we take $\rho = a_{t+1}$ and

$$\tau_j \propto \lambda_{t,j} b_{t+1}(\mu(x_{t,j}), y_{t+1}).$$

The new weights are

$$\lambda_{t+1,j} \propto \frac{b_{t+1}(x_{t+1,j}, y_{t+1})}{b_{t+1}(\mu(x_{t,I_j}), y_{t+1})}.$$

Another general strategy for constructing good proposals is to approximate

$$\log(a_{t+1}(x_t, x_{t+1}) b_{t+1}(x_{t+1}, y_{t+1}))$$

by a piecewise quadratic function of x_{t+1}

$$\sum_i \left(c_i(x_t, y_{t+1}) + d_i(x_t, y_{t+1})^T x_{t+1} - x_{t+1}^T F_i(x_t, y_{t+1}) x_{t+1}\right) 1_{B_i}(x_{t+1}).$$

Exponentiating the right hand side gives a mixture of truncated normal distributions times a normalizing constant $Z(x_t, y_{t+1})$. Hence we can take as ρ this mixture and $\tau_k \propto \lambda_{t,k} Z(x_{t,k}, y_{t+1})$.

This is particularly simple if $a_{t+1}(x_t, .)$ is a normal distribution and $b_{t+1}(., y_{t+1})$ is log concave.

Example: Stochastic volatility

Here (X_t) is a Gaussian AR(1), and $Y_t = \exp(X_t/2)V_t$ implies

$$\log b_t(x_t, y_t) = -\frac{1}{2}(\log 2\pi + x_t + y_t^2 \exp(-x_t)).$$

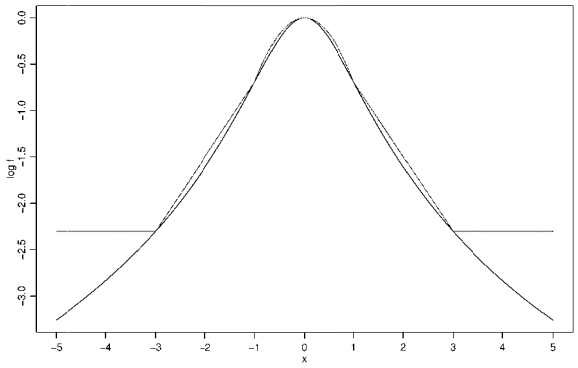
For $|y_t|$ small, this varies strongly as a function of x_t (and for $y_t = 0$ it is even unbounded). In this case, the standard particle filter can give stronly unbalanced weights. The recipes discussed in the previous slides can be applied and give much better results. In particular, we can use the accept/reject method easily.

Gaussian AR with t-distributed observation noise

In this case the filter density $f_{t|t}$ can be bimodal. This occurs if it is not clear whether an observation is an outlier or not. With the standard particle filter, one proposes mostly values around one mode and thus can miss the second mode. This will not show up in unbalanced weights.

The observation density $b_t(., y_t)$ is not log concave. Still, we can bound it from above by a function which is quadratic in the center, linear in the intermediate range and constant in the extremes. Using this, one obtains proposal that are mixtures of truncated Gaussian distributions.

log of a t-density and a piecewise quadratic approximation.



Partially observed diffusions

Here, the main difficulty is that the transition density from X_{t_k} to $X_{t_{k+1}}$ is not available. Fearnhead et al. (2006) have suggested to replace it with an unbiased estimator which is available for some diffusions.

The alternative is to include the values of X_t at intermediate time points $t_k + jh$, $h = (t_{k+1} - t_k)/m$ in the state vector. If m is large enough, we can use the Euler approximation

$$\begin{aligned} & X_{t_k+jh} \mid X_{t_k+(j-1)h} \\ & \quad \sim \quad \mathcal{N}(X_{t_k+(j-1)h} + hf(X_{t_k+(j-1)h}), h\sigma^2(X_{t_k+(j-1)h})). \end{aligned}$$

If the observation equation is linear with Gaussian noise, we can obtain a Gaussian approximation for

$$p(x_{s+h} \mid x_s, y_{k+1}) \propto p(x_{s+h} \mid x_s) p(y_{k+1} \mid x_{s+h}).$$

The log of the first factor is quadratic in x_{s+h}. The following approximation leads to a quadratic expression in x_{s+h} also in the log of the second factor.

$$\begin{aligned} Y_{k+1} &= HX_{t_{k+1}} + V_{k+1} \\ &\approx H(X_{s+h} + (t_{k+1} - s - h)f(X_{s+h}) \\ &\quad + \sigma(X_{s+h})(W_{t_{k+1}} - W_{s+h})) + V_k \\ &\approx H(X_{s+h} + (t_{k+1} - s - h)f(X_s) \\ &\quad + \sigma(X_s)(W_{t_{k+1}} - W_{s+h})) + V_k. \end{aligned}$$

(Based on Durham and Gallant, 2002).

Balanced sampling

In the resampling step, we choose indices $I_1, \ldots, I_N$ with probabilities $P[I_j = k] = \tau_k$. This is necessary to avoid sample depletion in the long run, but the additional randomness increases the variance in the short run.

If $\tau_k = \frac{1}{N}$, we have seen before that we can take each index exactly once, i.e. $I_j = j$ (the order is irrelevant). In general, we cannot eliminate randomness completely: The index k should be chosen $N\tau_k$ times, but $N\tau_k$ is not an integer. The best we can achieve is

$$N_k = \text{Number of times } k \text{ is chosen} \in \{[N\tau_k], [N\tau_k] + 1\}.$$

We call this balanced sampling.

There are at least two algorithms for balanced sampling.

Tree based resampling was proposed by Crisan and Lyons (2002). Consider the empirical and the true distributions

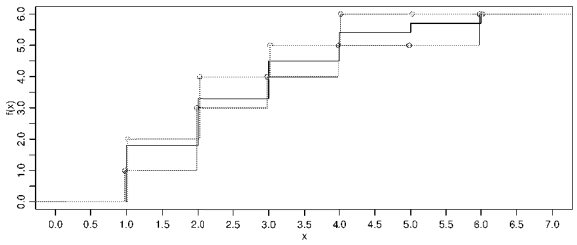
$$F_N(j) = \frac{1}{N}\sum_{k=1}^{j} N_k, \quad F(j) = \sum_{j=1}^{k} \tau(k).$$

Then the algorithm generates $F_N(j)$ as a Markov chain such that

$$|N(F_N(j) - F(j))| \leq 1, \quad |N_j - N\tau(j)| \leq 1.$$

(A little thought shows that this is possible). The same procedure can be used with more general recursive splittings of $\{1, 2, \ldots, N\}$.

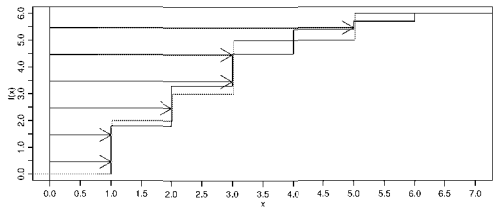
Illustration of tree sampling. Black: F. Green and Red: Upper and lower envelopes for F_N. One out of the 4 possible jumps is excluded by the condition $|N_j - N\tau_j| < 1$.



Circular resampling is due to Whitley (1994) (rediscovered by Carpenter et al., 1999). It takes U uniform and starts with the balanced sample

$$\left\{\frac{U}{N}, \frac{U+1}{N}, \ldots, \frac{U+N-1}{N}\right\}$$

on $(0, 1)$. This is then transformed by F^{-1}.



Other approaches to reduce the variability due to resampling are:

1. Use sequential importance sampling, i.e. $\tau_k \equiv \frac{1}{N}$, as long as the variance of the weights (or some other criterion of unbalance) is less than a threshold. Otherwise use resampling.

2. Choose $\tau_k = \lambda_{t,k}^{1/2}$ (or some other fractional power) as a compromise between sequential importance sampling and resampling.

Particle filters with accept-reject

The "ideal" proposal produces an unweighted sample from the target $f^N_{t+1|t+1}$. We can obtain an unweighted sample directly with the accept-reject method.

If at time t we have an unweighted sample, then we have to sample from

$$f^N_{t+1|t+1}(x_{t+1}) \propto \sum_j b_{t+1}(x_{t+1}, y_{t+1}) a_{t+1}(x_{t,j}, x_{t+1}).$$

The straightforward approach is to propose from

$$\frac{1}{N} \sum_j a_{t+1}(x_{t,j}, x_{t+1})$$

and accept with probability

$$b_{t+1}(x_{t+1}, y_{t+1}) / \sup_x b_{t+1}(x, y_{t+1}).$$

More efficient proposals can be constructed with the auxiliary variable idea of Pitt and Shephard.

Ensemble Kalman filter

This is due to Evensen and has become popular in atmospheric physics. Like the particle filter, it approximates the filter density at time t by a sample $(x_{t,j})$ which is propagated forward according to the state dynamics to give a prediction sample $(\tilde{x}_{t+1,j})$.

The difference is in the update step. It assumes a linear Gaussian observation model

$$Y_{t+1} = H_{t+1} X_{t+1} + V_{t+1}.$$

If $f_{t+1|t}$ were Gaussian, then $f_{t+1|t+1}$ would also be Gaussian with mean and variance given on the next slide. The ensemble Kalman filter draws from this Gaussian distribution, even when the prediction sample is not Gaussian.

Mean and variance are

$$\begin{aligned} m_{t+1|t+1} &= m_{t+1|t} + K_{t+1}(y_{t+1} - H_{t+1} m_{t+1|t}) \\ R_{t+1|t+1} &= R_{t+1|t} - K_{t+1} H_{t+1} R_{t+1|t}, \end{aligned}$$

where K_{t+1} is the gain matrix

$$K_{t+1} = R_{t+1|t} H^T_{t+1} (E(V_{t+1} V^T_{t+1}) + H_{t+1} R_{t+1|t} H^T_{t+1})^{-1}.$$

One can easily check that the following sample has the correct first two moments

$$x_{t+1,j} = \tilde{x}_{t+1,j} + K_{t+1}(y_{t+1} - H_{t+1} \tilde{x}_{t+1,j} + V_{t+1,j}),$$

(The gain matrix is computed with the covariance matrix $R_{t+1|t}$ estimated from the sample $\tilde{x}_{t+1,j}$.)

Even though there is no theoretical foundation for this update step, it is in some applications preferable to an update by weighting.

7. Particle smoothing

Particle algorithms for forward smoothing

Consider particle appoximations for the joint smoother:

$$p(x_{0:t} \mid y_{1:t}) \approx \sum_{j=1}^N \lambda_{t,j} \Delta(x_{0:t,j}).$$

(The notation would need another index to indicate how many observations we use, but this becomes clumsy, and so I assume that it is clear from the context).

Inserting this into the recursion, we obtain

$$\begin{aligned} p(x_{0:t+1} \mid y_{1:t+1}) &\approx p^N(x_{0:t+1} \mid y_{1:t+1}) \\ &\propto \sum_{j=1}^N \lambda_{t,j} \Delta(x_{0:t,j}) a_{t+1}(x_{t,j}, x_{t+1}) b_{t+1}(x_{t+1}, y_{t+1}). \end{aligned}$$

Using the same ideas as for the particle filter, we obtain the algorithm:

1. Generate $(x_{0,1}, \ldots, x_{0,N})$ from $a_0(x)d\mu(x)$ and set $\lambda_{0,j} = \frac{1}{N}$ and $t = 0$.
2. Choose probabilities (τ_k) for the index and generate I_j with $P[I_j = k] = \tau_k$.
3. Choose a transition density $\rho(x, x')$ and let

$$z_{0:t,j} = x_{0:t,I_j}, \quad z_{t+1,j} \sim \rho(z_{t,j}, x)d\mu(x).$$

4. Compute weights

$$\lambda_{t+1,j} \propto \frac{\lambda_{t,I_j} b_{t+1}(z_{t+1,j}, y_{t+1}) a_{t+1}(z_{t,j}, z_{t+1,j})}{\tau_{I_j} \rho(z_{t,j}, z_{t+1,j})}.$$

5. Increase t by 1, set $x_{0:t,j} = z_{0:t,j}$ and go back to 2.

We sample from a distribution that is discrete in the first t variables. The resampling step leads to sample depletion in $(x_{s,j})$ for any fixed s as t increases.

This has drastic consequences: The particle approximation to quantities like

$$\mathbf{E}\left[h(X_s) \mid Y_{1:t}\right]$$

degenerates for fixed s and N when $t \to \infty$. The same is true for

$$\frac{1}{t}\sum_{s=1}^{t} \mathbf{E}\left[h(X_s) \mid Y_{1:t}\right].$$

A possible remedy proposed by Berzuini et al. is to do additional Gibbs-sampler or Metropolis-Hastings transitions with the samples $(x_{0:t,j})$ before generating $x_{t+1,j}$.

Let $q(x_{0:t}, x'_{0:t})$ be any transition density which leaves $p(x_{0:t} \mid y_{1:t})$ invariant. Then another approximation of $p(x_{0:t+1} \mid y_{1:t+1}$ is the density proportional to

$$\sum_{j=1}^{N} \lambda_{t,j} q(x_{0:t,j}, x_{0:t}) a_{t+1}(x_t, x_{t+1}) b_{t+1}(x_{t+1}, y_{t+1}).$$

Sampling from this approximation can be done similarly as before. Because we now sample also $x_{0:t}$ from a continuous distribution, all ties that occur in sampling the indices I_j are broken. Sample depletion does not occur any more.

However, the complexity of the procedure is now quadratic in the number of time points.

Particle filtering followed by particle smoothing

This is an attempt to use the results of the particle filter $(x_{s,j})$. In order to avoid confusion, we will call the smoother sample $(z_{0:t,j})$.

As derived before, the backward transitions are:

$$p(x_s \mid x_{s+1}, y_{1:t}) \propto a_{s+1}(x_s, x_{s+1}) f_{s|s}(x_s \mid y_{1:s}).$$

If we directly insert the particle filter approximation for $f_{s|s}$, we have

$$p(x_s \mid x_{s+1}, y_{1:t}) \propto a_{s+1}(x_s, x_{s+1}) \sum_{j=1}^{N} \lambda_{s,j} \Delta(x_{s,j})$$

This means that the smoother sample is concentrated on the same values as the filter. More precisely, $z_{t,j} = x_{t,j}$ and given $z_{s+1,j}$ we set $z_{s,j} = x_{s,k}$ with probability proportional to

$$a_{s+1}(x_{s,k}, z_{s+1,j})\lambda_{s,k}.$$

This can suffer from the problem of sample depletion (if the smoother is much more concentrated than the filter). More seriously, this algorithm has complexity $O(N^2)$.

A better idea is to use the approximation $f^N_{s|s}$ which is based on the filter sample at time $s-1$:

$$p(x_s \mid x_{s+1}, y_{1:t}) \propto a_{s+1}(x_s, x_{s+1})b_s(x_s, x_{s+1}) \sum_{i=1}^{N} \lambda_{s-1,i} a_s(x_{s-1,i}, x_s).$$

Hence given $z_{s+1,j}$ we have to generate $z_{s,j}$ from the density proportional to

$$a_{s+1}(x_s, z_{s+1,j})b_s(x_s, y_s) \sum_{i=1}^{N} \lambda_{s-1,i} a_s(x_{s-1,i}, x_s).$$

Since for every j we have a different density to simulate from, importance sampling (or resampling) is not useful. Accept/reject can be used, again with the index as auxiliary variable.

However, we should not use a different proposal for every j because this will lead to a $O(N^2)$ complexity. We can construct efficient algorithms if the state is low-dimensional or if the dependence in the state process is not too strong.

8. Particle methods for parameter estimation

Approximate likelihood

The particle filter gives an approximate likelihood as a by-product:

$$\begin{aligned} p(y_{t+1} \mid y_{1:t}, \theta) &\approx \sum_j \lambda_{t,j} \int a_{t+1}(x_{t,j}, x; \theta) b_{t+1}(x, y_{t+1}; \theta) d\mu(x) \\ &= \mathbf{E}\left[\frac{\lambda_{t,I} a_{t+1}(x_{t,I}, X; \theta) b_{t+1}(X, y_{t+1}; \theta)}{\tau_I \rho(x_{t,I}, X)} \right] \end{aligned}$$

where (I, X) has the distribution

$$I \sim \tau, \quad X \mid I = i \sim \rho(x_{t,i}, x) d\mu(x).$$

The particle filter has generated a sample from this distribution.

For an approximate MLE, we need the likelihood at many values θ. Running independent particle filters for different θ's is computationally demanding and leads to a non-smooth likelihood.

A better idea is to use the same τ and ρ for all θ's in a small neighborhood. We then have to store the indices in addition to the particles (LeGland et al., 2005).

Pitt has suggested running particle filters for different θ's with different proposals, but the same random numbers. Still, this will not give a continuous likelihood approximation, because sampling from

$$(I, X) \sim \tau_i \rho(x_{t,i}, x) d\mu(x)$$

is not continuous. We need to modify the discrete approximation $\sum \lambda_{t,j} \Delta(x_{t,j})$. In one dimension, we can smooth the empirical distribution and use the quantile transform. In higher dimensions, things are more difficult.

There is a method to compute a smooth approximation of log likelihood from a smoother sample from one fixed parameter, see HRK (2001).

Alternatively, a stochastic version of the EM-algorithm has also been proposed. Again, I refer to HRK (2001).

Recursive estimation

In a sequential setting, maximizing the whole likelihood function each time a new observation becomes available is too complicated and often not feasible. One would like a simple explicit update formula to compute $\widehat{\theta}_{t+1}$ from y_{t+1} $\widehat{\theta}_t$ and some other statistics that can be updated simply.

This leads to the area of recursive estimation. Almost all proposals discussed so far rely on approximations not only of $f_{t|t}$, but also of the derivative of $f_{t|t}$ with respect to θ, the so-called tangent filter.

Bayesian estimation

The simplest idea is to include θ among the states, with the trivial evolution $\theta \equiv$ const., i.e. $\theta_{t+1} = \theta_t$.

At the beginning, we generate $(\theta_{0,j})$ from the prior and $x_{0,j}$ from $a_0(x_0; \theta_{0,j})d\mu(x_0)$, and we take uniform weights. One iteration of the particle filter then goes as follows:

1. Choose probabilities (τ_k) for the index and generate I_j with $P[I_j = k] = \tau_k$.

2. Choose a transition density $\rho(x \mid x', \theta)$ and let

$$\theta_{t+1,j} = \theta_{t,I_j}, \quad x_{t+1,j} \sim \rho(x \mid x_{t,I_j}, \theta_{t,I_j})d\mu(x).$$

3. Compute new weights

$$\lambda_{t+1,j} \propto \frac{\lambda_{t,I_j} b_{t+1}(x_{t+1,j}, y_{t+1}; \theta_{t+1,j}) a_{t+1}(x_{t,I_j}, x_{t+1,j}; \theta_{t+1,j})}{\tau_{I_j} \rho(x_{t,I_j}, x_{t+1,j}; \theta_{t+1,j})}.$$

In doing so, we encounter again the problem of sample depletion: $(\theta_{t+1,j})$ is a subsample of $(\theta_{t,j})$. If the posterior converges to a point mass, only one value from the original sample will survive.

There are several approaches to solve this problem: One can introduce jittering by adding some noise with small variance to the $\theta_{t+1,j}$'s. In order to compensate for the increased variance we should also shrink the $\theta_{t+1,j}$'s towards their mean.

The choice of the spread of the jitter is difficult. For consistent estimation, it has to decrease.

Another possible remedy is to use an additional Gibbs or Metropolis-Hastings step for θ. However, because $p(\theta \mid x_t, y_{1:t})$ is not available, we cannot find a transition density which leaves this density invariant. We have to approximate the whole smoothing distribution.

This means that at time $t+1$ we sample from the density proportional to

$$\sum_{j=1}^{N} \lambda_{t,j} \Delta(x_{0:t,j}) p(\theta \mid x_{0:t,j}, y_{1:t}) a_{t+1}(x_{t,j}, x_{t+1}; \theta) b_{t+1}(x_{t+1}, y_{t+1}; \theta).$$

This can be done similarly as before. The problem of sample depletion in the smoother sample remains. If we are not interested in the smoother sample, this seems not to matter.

In some situations, we don't even have to store the smoother samples. Assume there is a finite dimensional sufficient statistics $T_t(x_{0:t}, y_{1:t})$ such that

$$p(\theta \mid x_{0:t,j}, y_{1:t}) = p(\theta \mid T_t)$$

and there is a simple update

$$T_{t+1} = \psi(T_t, x_{t+1}, y_{t+1}).$$

A simple example is a dynamic generalized linear model, i.e. a Gaussian AR-model for (x_t) and an exponential family for $b_t(x_t, y_t)$.

Then we can implement a particle filter recursion for the variables $(x_{t,j}, \theta_j, T_{t,j})$ with weights $\lambda_{t,j}$ where ties or sample depletion apparently does not occur. This is due to Storvik (2002).

Still, there is a hidden depletion in the state variables that enter into $T_{t,j}$, and in general it is not true that

$$\sum_{j=1}^{N} \lambda_{t,j} h(T_{t,j}) \to \mathbf{E}\left[h(T_t) \mid y_{1:t}\right]$$

as $N \to \infty$, uniformly in t.

The simulations in Storvik (2002) look good, but presumably the procedure breaks down eventually.