Logistics Review

Cumulative Outstanding Reading

- Much of today's lecture comes from slides by Simon Godsill, and from the book by Cappe, Moulines, and Ryden, “Inference in Hidden Markov Models”
- read the section on dynamic Bayesian networks in 'doc.pdf'
- read the section on inference in 'doc.pdf'
Final Project - pending deadlines

- All deadlines today.
- March 18th, **10:00am**: 4 page project writeup (conference style, with abstract).
- March 18th, **12:30pm**: presentation slides (pdf, pptx, or key)
- Final Presentations, Monday, March 18, 2013, 230-420 pm, PCAR 492

Note, March 18th deadlines are at 10:00am and 12:30pm, while the others are at 11:45pm at night.

Final Project

- Final presentations, 15 minutes
- Final reports, no more than 4 pages and no less than 11 point font, and no more than 1 inch margins on all sides!
Sequential Monte Carlo

- Note: much of this next lecture material is based on Simon Godsill’s tutorial on Sequential Monte Carlo. Some of the material is a literal copy of his slides. Please see his web page at http://www-sigproc.eng.cam.ac.uk/~sjg
- Another method to deal with extremely large state space models.
- Well motivated when the state space is continuous (uncountably infinite). But also works well when state space is discrete but extremely large (and when separator finding is insufficient).
- The general methods are called Sequential Monte Carlo (SMC) or “particle filters”, and these have been very successful in dealing with monitoring and filtering methods in object tracking (e.g., robotics), vision, speech and audio, financial time series, and so on.
- There has not yet been a full comparison of SMC to the pruning and the factorization approaches mentioned earlier, but as we will see SMC approaches are particularly useful for continuous state spaces.

Continuous State spaces

- We’ve mostly so far considered discrete state spaces, where $D_{Q_t}$ and $D_{H_t}$ are both finite and countable.
- SMC is appropriate for when one or both of $Q$ and $H$ is continuous, and where the dependencies between variables can be highly non-linear (although we will spend significant time on the linear Gaussian case).
- Discretization of the state space is an option, but there are reasons against doing so.
- Numerical integration is also an option, but reasons against doing this as well.
Continuous State spaces

- Linear Gaussian state spaces are such that exact inference can still be performed, since passing messages is really just computing and/or updating conditional means and conditional covariance matrices (which we now know how to do).
- Many cases, however, the dependencies are not linear and/or the noise is not Gaussian. Here, Monte Carlo filters (or particle filters) work very well in this case.
- They have the property of adaptive resolution: the resolution of the “sampling” of the state space adjusts based on where the probability is, and this happens adaptively as time progresses. Any fixed grid over the state space could be simultaneously wasteful (over regions where there is little probability) and also coarse grained (over regions where there is much probability).

Monte Carlo Methods

- MC methods estimate continuous densities by a bunch of samples.
- Key aspect goal of Monte Carlo methods is to estimate expectations of some function $h(x)$ w.r.t. distribution $p(x)$ where $x$ might be continuous and very high dimensional.

$$E h(X) = \int h(x) p(x) dx$$  \hspace{1cm} (18.1)

where $h(\cdot)$ is some function that we need for some estimation purpose.
In cases where this cannot be achieved analytically the approximation problem can be tackled indirectly, as it is often possible to generate random samples from \( p(x) \), i.e. by representing the distribution as a collection of random points:

\[
x^{(i)}, \quad i = 1, \ldots, N, \quad \text{for very large } N \tag{18.2}
\]

We can think of Monte Carlo representation informally as:

\[
p(x) \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x) \tag{18.3}
\]

Then the Monte Carlo expectation falls out easily as:

\[
Eh(X) = \int h(x)p(x)dx \approx \int h(x)\frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x)dx = \frac{1}{N} \sum_{i=1}^{N} h(x^{(i)}) \tag{18.4}
\]

Alternatively, suppose we draw the random samples \( x^{(i)} \) from a distribution \( q(x) \) instead of \( p(x) \). Now the expectation can be estimated using importance sampling:

\[
Eh(X) = \int h(x)p(x)dx = \int h(x)\frac{q(x)p(x)}{q(x)}dx \tag{18.5}
\]

\[
\approx \int h(x)\frac{p(x)}{q(x)} \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x)dx = \frac{1}{N} \sum_{i=1}^{N} \frac{p(x^{(i)})}{q(x^{(i)})} h(x^{(i)}) = \sum_{i=1}^{N} w^{(i)} h(x^{(i)}) \tag{18.6}
\]

where \( w^{(i)} \propto \frac{p(x^{(i)})}{q(x^{(i)})} \) is known as the importance weight, and where we can informally think of \( p(x) \) as

\[
p(x) \approx \sum_{i=1}^{N} w^{(i)} \delta_{x^{(i)}}(x), \quad \text{with } \sum_{i} w^{(i)} = 1 \tag{18.7}
\]
There are numerous versions of Monte Carlo samplers, including Markov chain Monte Carlo, simulated annealing, importance sampling, quasi-Monte Carlo, ...

Here we limit attention to Sequential Monte Carlo methods, which are proving very successful for solving challenging state-space modeling problems.

**Important trick:** Automatic marginalization using Monte Carlo: If

\[(x, \theta) \sim p(x, \theta|y) \quad (18.8)\]

then

\[x \sim (x|y) \quad (18.9)\]

and

\[\theta \sim p(\theta|y) \quad (18.10)\]

Since

\[p(x, \theta|y) = p(x|y)p(\theta|x, y) = p(\theta|y)p(x|\theta, y) \quad (18.11)\]
State space models, filtering and smoothing

We will focus here on a broad and general class of models. Examples include:

- Hidden Markov models
- Most standard time series models: AR, MA, ARMA,...
- Special models from tracking, computer vision, finance, communications, bioinformatics, ...

We consider a very general class of time series models, the state space model. Almost all models of practical utility can be represented within this category, using a state vector of finite dimension. The sequential inference methods presented can readily be extended beyond the Markovian state space models given here, and also to factored state spaces (such as DBNs), but for simplicity we retain the standard Markovian setup.

\[
p(q_{1:t}, x_{1:t}) = p(q_1)p(x_1|q_1) \prod_t p(q_t|q_{t-1})p(x_t|q_t) \tag{18.12}
\]

- Standard time series \( q_t, t \in \{1, \ldots, T\} \)
- Observations \( x_t \), like an standard 1st order HMM
- Standard joint density

\[
q_{t+1} \sim p(q_{t+1}|q_t) \tag{18.13}
\]

\[
x_{t+1} \sim p(x_{t+1}|q_{t+1}) \tag{18.14}
\]
Ex: linear AR model with Gaussian noise

\[
z_t = \sum_{i=1}^{P} a_i z_{t-i} + e_t \tag{18.15}
\]

\[
y_t = z_t + w_t \tag{18.16}
\]

with \(e_t\) and \(w_t\) independently distributed as zero mean Gaussians with variance \(\sigma^2_e\) and \(\sigma^2_w\) respectively (fixed and known).

\(a_i\) are the AR coefficients, of order \(P\), also assumed here to be fixed and known.

We observe the noisy signal \(y_t\). The only unknown here is the signal \(z_t\).

This can be represented as a standard state-evolution and observation, like in an HMM, as we see next:

State vector

\[
x_t = (z_t, z_{t-1}, \ldots, z_{t-P+1})^T \tag{18.17}
\]

State space model becomes

\[
x_t = Ax_{t-1} + \epsilon_t \tag{18.18}
\]

\[
y_t = Bx_t + w_t \tag{18.19}
\]

with
State vector

\[ x_t = (z_t, z_{t-1}, \ldots, z_{t-P+1})^T \] \hfill (18.20)

State space model becomes

\[ x_t = A x_{t-1} + \epsilon_t \] \hfill (18.21)
\[ y_t = B x_t + w_t \] \hfill (18.22)

with

\[
A = \begin{bmatrix}
a_1 & a_2 & \ldots & a_P \\
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1 \\
\end{bmatrix}
\]  
\[
B = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
\end{bmatrix}
\]

\[
\Sigma_\epsilon = \begin{bmatrix}
\sigma^2_\epsilon & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 0 \\
\end{bmatrix}
\]

Alternatively, in terms of state evolution and observation densities:

\[
p(x_{t+1} | x_t) = \mathcal{N}(x_{t+1} | Ax_t, \Sigma_\epsilon) \] \hfill (18.23)
\[
g(y_t | x_t) = \mathcal{N}(y_t | Bx_t, \sigma^2_w) \] \hfill (18.24)

Note the above are all conditional Gaussians (we’ve seen these earlier). This therefore is an example of the linear Gaussian state space model, an important special case that is used extensively to construct algorithms in the nonlinear non-Gaussian case (extended Kalman filters, Rao-Blackwellised particle filters, and so on).
Example: nonlinear model

\[ x_t = A(x_{t-1}) + v_t \]  \hspace{1cm} (18.25)

\[ = \frac{x_{t-1}}{2} + 25 \frac{x_{t-1}}{1 + x_{t-1}^2} + 8 \cos(1.2t) + v_t \]  \hspace{1cm} (18.26)

\[ y_t = B(x_t) + w_t \]  \hspace{1cm} (18.27)

\[ = \frac{(x_t)^2}{20} + w_t \]  \hspace{1cm} (18.28)

where \( v_t \sim \mathcal{N}(0, \sigma_v^2) \) and \( w_t \sim \mathcal{N}(0, \sigma_w^2) \).

So noise is still normal, but mean relationship is non-linear, and thus we may express this as a normal with nonlinear mean functions.

\[ f(x_{t+1}|x_t) = \mathcal{N}(x_{t+1}|A(x_t), \sigma_v^2) \]  \hspace{1cm} (18.29)

\[ g(y_t|x_t) = \mathcal{N}(y_t|B(x_t), \sigma_w^2) \]  \hspace{1cm} (18.30)

The quantities we typically wish to compute for an HMM include:

- Compute \( p(q_t|x_{1:t}) \), or the *filtering* problem.
- Compute \( p(q_t|x_{1:s}) \), with \( t > s \), or the *prediction* problem.
- Compute \( p(q_t|x_{1:u}) \), with \( t < u \), or the *smoothing* problem.
- Note: above includes \( p(q_t|x_{1:T}) \).
- Also needed query is \( p(q_t, q_{t+1}|x_{r:s}) \) (often \( r = 1 \) and \( s = T \)).
- In all above cases, we need to sum out hidden variables from joint distributions. E.g., \( p(q_t|x_{1:T}) = p(q_t, x_{1:T})/p(x_{1:T}) \), so also need \( p(x_{1:T}) \). I.e., we compute both the numerator and denominator in each of the above queries.
Estimation tasks in current setting

- Filtering: we wish to estimate $p(q_t|x_{1:t})$ or expectations based on it

$$Eh(q_t) = \int h(q_t)p(q_t|x_{1:t})dq_t$$  \hspace{1cm} (18.31)

recall, now, $q_t$ is a continuous (and possibly vector) variable.

- Smoothing (fixed lag), compute $p(q_{t-L}|x_{1:t})$

- Smoothing (fixed interval), compute overall poster

$$p(q_{1:T}|x_{1:T})$$  \hspace{1cm} (18.32)

Filtering

At time $t$, Suppose we have $p(q_t|x_{1:t})$ and wish to compute $p(q_{t+1}|x_{1:t+1})$. We use the two step filtering recursions that we’ve already seen several times, but now it is for continuous variables.

Prediction step:

$$p(q_{t+1}|x_{1:t}) = \int p(q_t, q_{t+1}|x_{1:t})dq_t$$  \hspace{1cm} (18.33)

$$= \int p(q_t|x_{1:t})p(q_{t+1}|q_t, x_{1:t})dq_t$$  \hspace{1cm} (18.34)

$$= \int p(q_t|x_{1:t})p(q_{t+1}|q_t)dq_t$$  \hspace{1cm} (18.35)
The sequential scheme is as follows:

\[
\begin{align*}
T & t - 1 \quad t \quad t + 1 \\
D & y_t - 1 \quad y_t \quad y_{t + 1} \\
F & p(x_{t - 1} | y_0 : t - 1) \quad p(x_t | y_0 : t) \quad p(x_{t + 1} | y_0 : t + 1) \\
P & p(x_{t + 1} | y_1 : t) \quad p(x_{t + 1} | y_{t + 1}) \\
\end{align*}
\]

However, in the general case the integral is intractable and approximations must be used. (\(x_t\) high-dimensional, \(f()\), \(g()\) non-Gaussian, ...)

We saw this with the BK algorithm, and where after each prediction and correction step, we project down to a factored representation, when \(q_t\) is highly structured. Here, \(q_t\) is a vector continuous process, and there could be nonlinearities, so that the integrals above are not evaluable analytically.

Correction step:

\[
\begin{align*}
p(q_{t + 1} | x_{1 : t + 1}) & = \frac{p(x_{t + 1} | q_{t + 1}) p(q_{t + 1} | x_{1 : t})}{\int p(x_{t + 1} | q_{t + 1}) p(q_{t + 1} | x_{1 : t}) \, dq_{t + 1}} \\
& = \frac{p(x_{t + 1} | q_{t + 1}) p(q_{t + 1} | x_{1 : t})}{p(x_{t + 1} | x_{1 : t})} \\
\end{align*}
\]

(18.36) (18.37)

In cases where the state space model is linear and Gaussian, the classic Kalman filter arises. We’re going to use standard notation, so \(x_t\) is hidden continuous state and \(y_t\) is the observation.

The next set of slides come from the Godsill lectures.
- We can write this equivalently as:

\[
\begin{align*}
x_{t+1} &= Ax_t + v_t \\
y_t &= Bx_t + w_t
\end{align*}
\]

(11) (12)

where \(v_t\) and \(w_t\) are zero mean Gaussian vectors with covariance matrices \(C\) and \(D\), respectively. \(v_t\) and \(w_t\) are independent over time and also independent of one another (not strictly necessary).
• We can write this equivalently as:

\[ x_{t+1} = Ax_t + v_t \]  \hspace{1cm} (11)
\[ y_t = Bx_t + w_t \]  \hspace{1cm} (12)

where \( v_t \) and \( w_t \) are zero mean Gaussian vectors with covariance matrices \( C \) and \( D \), respectively. \( v_t \) and \( w_t \) are independent over time and also independent of one another (not strictly necessary).

• We also require that the initial state be Gaussian distributed:

\[ p(x_0) = \mathcal{N}(x_0|\mu_0, P_0) \]
• We first require \( p(x_{t+1}|y_{0:t}) \), the prediction step from the above filtering recursion:
\[
p(x_{t+1}|y_{0:t}) = \int p(x_{t}|y_{0:t}) f(x_{t+1}|x_{t}) dx_t
\]

• Suppose that we have already that at time \( t \):
\[
p(x_t|y_{0:t}) = \mathcal{N}(x_t|\mu_t, P_t)
\]
Now, from (11) we have

\[ x_{t+1} = Ax_t + v_t \]

Thus from standard transformation of variables theory (linear Gaussian case) we have:

\[ x_t \sim \mathcal{N}(x_t | \mu_t, P_t) \]
\[ x_{t+1} = Ax_t + v_t \]
Now, from (11) we have:
\[ x_{t+1} = Ax_t + v_t \]

Thus from standard transformation of variables theory (linear Gaussian case) we have:
\[ x_t \sim N(x_t | \mu_t, P_t) \]

Therefore:
\[ p(x_{t+1}|y_0:t) = N(x_{t+1} | \mu_{t+1}|t, P_{t+1}|t) \]

where:
\[ \mu_{t+1}|t = A\mu_t, \quad P_{t+1}|t = C + AP_tA^T \tag{13} \]

Now, the correction step of the above filtering recursion is
\[
p(x_{t+1}|y_{0:t+1}) = \frac{g(y_{t+1}|x_{t+1})p(x_{t+1}|y_t)}{p(y_{t+1}|y_{0:t})} \tag{14}
\]
where

\[ \mu_{t+1} = P_{t+1}(B^T D^{-1} y_{t+1} + P_{t+1|t} \mu_{t+1|t}) , \quad P_{t+1} = (B^T D^{-1} B + P_{t+1|t})^{-1} \]
Hence the whole Kalman filtering recursion can be summarised as:

\[
\begin{align*}
\mu_{t+1|t} &= A\mu_t \quad (15) \\
P_{t+1|t} &= C + AP_tA^T \quad (16) \\
\mu_{t+1} &= \mu_{t+1|t} + K_t(y_{t+1} - B\mu_{t+1|t}) \quad (17) \\
P_{t+1} &= (I - K_tB)P_{t+1|t} \quad (18) \\
K_t &= P_{t+1|t}B^T(BP_{t+1|t}B^T + D)^{-1} \quad (19)
\end{align*}
\]
Likelihood evaluation.
A key result is that the Kalman filter can sequentially evaluate the likelihood function, \( p(y_{1:t}) \). This is used for maximum likelihood or maximum a posteriori parameter estimation, and also for Bayesian model choice problems and the Rao-Blackwellised particle filter.
Likelihood evaluation.

A key result is that the Kalman filter can sequentially evaluate the likelihood function, \( p(y_{t+1}) \). This is used for maximum likelihood or maximum a posteriori parameter estimation, and also for Bayesian model choice problems and the Rao-Blackwellised particle filter.

To see how this works, start from the Kalman prediction step:

\[
p(x_{t+1}|y_0:t) = \mathcal{N}(x_{t+1}|A\mu_t, C + AP_tA^T)
\]

Now, equation 12 expresses \( y_{t+1} \) in terms of \( Bx_{t+1} \) plus a random Gaussian disturbance \( w_{t+1} \) with covariance matrix \( D \):

\[
y_{t+1} = Bx_{t+1} + w_{t+1}
\]
Hence, again using transformation of Gaussian variables, we can obtain the conditional likelihood:

\[ p(y_{t+1}|y_0:t) = \mathcal{N}(y_{t+1}|B\mu_{t+1}|t, D + BP_tB^T) \]

Finally, using the probability chain rule, we obtain the likelihood function:

\[ p(y_0:T) = p(y_0)\prod_{t=0}^{T-1} p(y_{t+1}|y_0:t) \]

### Numerical methods - or things you can’t do with the Kalman filter

- The Kalman filter is optimal only for the linear Gaussian model. In other cases the Kalman filter will give the best linear estimator in a mean-square error sense, but this may not be good enough for highly non-linear or non-Gaussian models.

- There are numerous methods for dealing with more general models, all based on numerical approximations to the filtering recursions of equations 8 and 9, e.g. the Gaussian sum filter, unscented Kalman filter, etc...

- Here we will consider two important examples: the extended Kalman filter (EKF) and the Monte Carlo particle filter.
The extended Kalman filter (EKF) is the classical method for estimating non-linear state-space systems.

Consider the following non-linear state-space model, which is the non-linear equivalent to equations:

\[ x_{t+1} = A(x_t) + v_t \]  
\[ y_t = B(x_t) + w_t \]

where \( A() \) and \( B() \) are now non-linear functions.

Perform a 1st order Taylor expansion of \( A() \) and \( B() \) around the points \( \mu_t \) and \( \mu_{t|t-1} \), respectively:

\[ A(x_t) \approx A(\mu_t) + \frac{\partial A(x_t)}{\partial x_t} \bigg|_{x_t=\mu_t} (x_t - \mu_t) \]
\[ B(x_t) \approx B(\mu_{t|t-1}) + \frac{\partial B(x_t)}{\partial x_t} \bigg|_{x_t=\mu_{t|t-1}} (x_t - \mu_{t|t-1}) \]

Substituting these approximations into the state-space model leads to a linearized set of equations which can be solved using the standard Kalman filter.

Limitations - the approximation is still unimodal, hence for multimodal distributions the filter will fail.

Also, the tracking performance and error covariance estimates will be sub-optimal.
One (apparently) simple extension to the Kalman filter is to allow
the state to be a combination of a discrete and a continuous
variable, like a mixture of Gaussians.

Both the mixture and the Gaussian variables might evolve overtime
in a Markov chain.

Let $Q_t$ be our discrete Markov chain, as normal, and let $X_t$ be the
hidden continuous state variable. So the joint state at time $t$ is
$(Q_t, X_t)$. The observation sequence, in this case, is $Y_t$. 
Again, our goal is to produce state estimation in this model. I.e.,

\[ p(q_t, x_t | y_{1:t}) \]

We also want a recursion, to produce \( p(q_t, x_t | y_{1:t}) \) based on \( p(q_{t-1}, x_{t-1} | y_{1:t-1}) \) and a few (hopefully simple) update steps.

It seems like this should be done, as this is only a conditional Gaussian model. I.e., for any \( t \), we have that \( p(y_{1:t}, x_{1:t} | q_{1:t}) \) is just a (structured/factorizable) Gaussian, and the entire model

\[ p(y_{1:t}, x_{1:t}) = \sum_{q_{1:t}} p(y_{1:t}, x_{1:t} | q_{1:t}) p(q_{1:t}) \quad (18.38) \]

is just a mixture of Gaussians.

So might seem to be relatively easy.

We consider an easier case, where the discrete states \( Q_{1:t} \) are independent.

The model is a 1-tree, so this (seemingly) should be particularly easy, especially since the query we wish \( p(q_t, x_t | y_{1:t}) \) doesn’t further couple together variables in the query (there is already an edge between \( Q_t \) and \( X_t \)).
The $y_t$ variables are all observed, so we consider simplified version above.

- Marginal on $x_1$

$$ p(x_1) = \sum_{q_1} p(x_1|q_1)p(q_1)$$ (18.39)

so $p(x_1)$ is a continuous distribution, a mixture of Gaussians.

- Marginal on $x_2$

$$ p(x_2) = \sum_{q_2} \int p(x_2|x_1, q_2)p(x_1)dx_1p(q_2)$$ (18.40)

$$ = \sum p(x_1) \left( \sum_{q_2} p(x_2|x_1, q_2)p(q_2) \right) dx_1$$ (18.41)

$$ = \int p(x_1)p(x_2|x_1)dx_1$$ (18.42)

where $p(x_1)$ is a mixture and $p(x_2|x_1)$ is a mixture of linear conditional Gaussians.
Switching Kalman Filter Tree

\[ p(x_2) = \int p(x_1)p(x_2|x_1)dx_1 \quad (18.43) \]

- If \( p(x_1) \) was a single Gaussian, then \( p(x_2) \) would also be a simple mixture of Gaussians. I.e., \( p(x_2) \) only has as many Gaussian components as the mixture in \( p(x_2|x_1) = \sum_{q_2} p(q_2)p(x_2|x_1,q_2) \) has.

Switching Kalman Filter Tree

\[ p(x_2) = \int p(x_1)p(x_2|x_1)dx_1 \quad (18.44) \]

- With \( p(x_1) \) a mixture, than the marginal \( p(x_2) \) has a squared number of components, i.e., \(|D_{Q_1}| \times |D_{Q_2}|\).
Consider numerical estimation of the following expectation:

$$h = E[h(x_t)] = \int h(x_t) p(x_t|y_0:t) \, dx_t$$

If the integral is intractable then we can resort to a Monte Carlo integration:

$$\hat{h} = \frac{1}{N} \sum_{i=1}^{N} h(x_t^i),$$

where $$x_t^i \overset{iid}{\sim} p(x_t|y_0:t)$$

More generally, when we cannot sample directly from $$p(x_t|y_0:t)$$, we can sample from another distribution $$q(x_t)$$ (‘importance function’) having the same support as $$p(x_t|y_0:t)$$. So we make $$N$$ random draws from $$q()$$ instead of $$p()$$:

$$x_t^i \overset{iid}{\sim} q(x_t), \quad i = 1, \ldots, N$$

Switching Kalman Filter

$$p(x_t) = \int p(x_{t-1}) p(x_t|x_{t-1}) \, dx_{t-1}$$  \hspace{1cm} (18.45)

With $$p(x_{t-1})$$ a mixture of $$\prod_{\tau=1}^{t-1} |D_{Q_1}|$$ components, $$p(x_t)$$ will have $$\prod_{\tau=1}^{t} |D_{Q_1}|$$ components.

Clearly managing this belief state becomes intractable, as the number of mixture components increases exponentially with $$t$$.

In fact, inference in this model is an NP-complete optimization problem, even though it is a 1-tree (Lerner&Parr).

This is where sequential Monte Carlo comes to the rescue.
Now we have to make a correction to ensure that the expectation estimate is good. It turns out that the required correction is proportional to the ratio $p() / q()$, which is termed the importance weight:

$$w_i^{(i)} \propto \frac{p(x_i^{(i)} | y_{0:t})}{q(x_i^t)}$$

If we normalise the importance weights such that $\sum_{i=1}^{N} w_i^{(i)} = 1$ we can form an empirical approximation to the filtering density:

$$p(x_t | y_{0:t}) \approx \sum_{i=1}^{N} w_i^{(i)} \delta_{x_i^{(i)}}(x_t) \quad (23)$$
from which expectation estimates can be obtained as:

\[
\hat{h} = \sum_{i=1}^{N} w^{(i)} h(x^{(i)}),
\]  

(24)

where \( x^{(i)} \) iid \( \sim q(x_t) \), \( w^{(i)} \propto p(x^{(i)} | y_{0:t})/q(x^{(i)}) \), \( \sum_{i=1}^{N} w^{(i)} = 1 \)  

(25)

\[i.e.
\]

\[
\bar{h} = \mathbb{E}h(x_t) = \int h(x_t)p(x_t|y_{0:t})dx_t
\]

\[= \int h(x_t) \sum_{i=1}^{N} w^{(i)} \delta_{x^{(i)}}(x_t)dx_t = \sum_{i=1}^{N} w^{(i)} h(x^{(i)})
\]

42
• **Resampling** (this will prove important in the sequential setting). We now have the option of resampling the particles so they have uniform weights:

\[ x^{(i)}_{t} = x^{(i)}_{t} \text{ with probability } w^{(i)}_{t} \]

and set \( w^{(i)}_{t+1} = 1/N \).

While this is unnecessary in the static case, and would always increase the Monte Carlo variation of our estimators, it is a vital component of the sequential schemes which follow, limiting degeneracy of the importance weights over time. Note that resampling schemes can incorporate variance reduction strategies such as stratification in order to improve performance.
Sequential Monte Carlo (SMC) - the Particle filter

A generic solution involves repeated importance sampling/resampling sequentially through time (particle filter) (see e.g. Gordon et al. 1993 (IEE), Kitagawa (1993, J. Comp. Graph. Stats.), Doucet Godsill Andrieu 2000 (Stats. and computing), Liu and Chen 1997 (JASA)).

The SMC scheme mimics the filtering recursions as follows:
• Substitute this into the prediction equation:

\[
p(x_{t+1}|y_{0:t}) = \int p(x_t|y_{0:t}) f(x_{t+1}|x_t)dx_t
\]

\[
\approx \int \frac{1}{N} \sum_{i=1}^{N} \delta(x_t - x_t^{(i)}) f(x_{t+1}|x_t)dx_t
\]

\[
= \frac{1}{N} \sum_{i=1}^{N} f(x_{t+1}|x_t^{(i)})
\]
• SMC is a collection of methods for drawing random samples from the
above Monte Carlo approximation to \( p(x_{t+1}|y_{0:t+1}) \), i.e. producing a
new set of random draws:

\[
x_{t+1}^{(i)} \sim p(x_{t+1}|y_{0:t+1}), \quad i = 1, \ldots, N \quad (N \text{ large})
\]
SMC is a collection of methods for drawing random samples from the above Monte Carlo approximation to $p(x_{t+1}|y_0:t+1)$, i.e. producing a new set of random draws:

$$x^{(i)}_{t+1} \sim p(x_{t+1}|y_0:t+1), \quad i = 1, \ldots, N$$

These random samples can be obtained by many means:

- Rejection sampling (slow but exact - requires an envelope function)
- Importance sampling - most common procedure
- MCMC - effective but slow [note that MCMC can be applied in conjunction with IS - then very effective]
- Special schemes such as annealed importance sampling.

There are many variants on schemes to achieve this (Bootstrap filter (Gordon et al. 1993, Sequential Importance sampling, (Doucet Godsill Andrieu (2000), Liu and Chen (1997)), Auxiliary Particle filters (Pitt and Shephard (1998)), etc.)
• We would like to mimic the three steps here by Monte Carlo operations.
• We would like to mimic the three steps here by Monte Carlo operations.

• Suppose we start off with many 'particles' drawn from the filtering distribution \( p(x_t | y_{0:t}) \). We label these particles as

\[
x^{(i)}_t, \quad i = 1, 2, ..., N \quad \text{with} \quad N \gg 1
\]

• These can be used to plot histogram estimates of \( p(x_t | y_{0:t}) \), form Monte Carlo estimates of expectations, ..., in fact perform almost any inference procedure we care to choose, provided \( N \) is ‘sufficiently’ large.
• We can simulate **Step 0** above by taking each particle $x_i^{(i)}$ in turn and generating a new state from the state transition density according to:

$$x_{i+1}^{(i)} \sim f(x_{t+1}|x_i^{(i)})$$

• Each pair $(x_t^{(i)}, x_{t+1}^{(i)})$ is now a joint random sample from $p(x_t, x_{t+1}|y_{0:t})$. 
• We can simulate **Step 0** above by taking each particle $x_t^{(i)}$ in turn and generating a new state from the state transition density according to:

$$x_{t+1}^{(i)} \sim f(x_{t+1}|x_t^{(i)})$$

• Each pair $(x_t^{(i)}, x_{t+1}^{(i)})$ is now a joint random sample from $p(x_t, x_{t+1}|y_0:t)$.

• By construction, $x_{t+1}^{(i)}$ taken on its own is a random sample from the required marginal distribution $p(x_{t+1}|y_0:t)$, **(Step 1)**
• We now have the option of
  1. retaining weighted particles, in which case the weights are
     accumulated over time as
     
     \[ w_{t+1} \propto w_t g(y_{t+1}|x_{t+1}) \]
• A basic algorithm with (optional) resampling at every time step, the 'Bootstrap Filter', is thus (Gordon, Salmond and Smith 1993, Kitagawa 1996:

For $t = 1, 2, ..., T$

For $i = 1, 2, ..., N$

\[ x_{t+1}^{(i)} \sim f(x_{t+1}^{(i)}|x_t^{(i)}) \]

\[ w_{t+1}^{(i)} \propto u_t^{(i)} g(y_{t+1}|x_{t+1}^{(i)}) \]

End

For $i = 1, 2, ..., N$

(Optional) Resample $x_{t+1}^{(i)}$ with probability $w_t^{(i)}$. Set $w_{t+1}^{(i)} = 1/N$

End

End
Example:
standard nonlinear model
\[
x_t = A(x_{t-1}) + v_t \\
= x_{t-1} + 25 x_{t-1}^2 + x_{t-1}^2 + 8 \cos(1.2t) + v_t
\]
\[
y_t = B(x_t) + w_t = (x_t)^2 + w_t
\]
where \(v_t \sim N(0, \sigma^2_v)\) and \(w_t \sim N(0, \sigma^2_w)\).

This may be expressed in terms of density functions as:
\[
f(x_{t+1}|x_t) = N(x_{t+1}|A(x_t), \sigma^2_v)
\]
\[
g(y_t|x_t) = N(y_t|B(x_t), \sigma^2_w)
\]

Figure 1: Bootstrap filter operation - nonlinear model
We now consider \( q(x_t, x_{t+1}|y_{0:t+1}) \) to be an importance function for \( p(x_t, x_{t+1}|y_{0:t+1}) \). The importance weight for Step 2' is hence modified to:

\[
\tilde{w}_{t+1}^{(i)} \propto w_t^{(i)} \frac{g(y_{t+1}|x_{t+1}^{(i)})f(x_{t+1}^{(i)}|x_t^{(i)})}{q(x_{t+1}^{(i)}|x_t^{(i)})}
\]

This is the general sequential importance (SIS) sampling method (Liu and Chen, 1998, Doucet, Godsill and Andrieu, 2000).
Repeated application over time (without resampling) leads to degeneracy of the weights - all the mass becomes concentrated on a few $i$ - hence estimates are poor.
A Basic Importance Sampling/Resampling Particle Filter

The first step initialises the initial states of the filter at $t = 0$:

$$x_0^{(i)} \sim p(x_0|y_0), \quad i = 1, 2, ..., N$$

where it is assumed that this draw can be made easily (use MCMC or static IS if not).

Then, for $t=0,1,2,...$
- At time $t$, have

$$p(x_t|y_{0:t}) \simeq \tilde{p}(x_t|y_{0:t}) = \sum_{i=1}^{N} w_i^{(i)} \delta(x_t - x_i^{(i)}), \quad \sum_{i=1}^{N} w_i^{(i)} = 1$$

- For $i = 1, ..., N$:

$$x_{t+1}^{(i)} \sim q(x_{t+1}|x_i^{(i)})$$
At time $t$, have

$$p(x_t|y_{0:t}) \approx \hat{p}(x_t|y_{0:t}) = \sum_{i=1}^{N} w_i^{(i)} \delta(x_t - x_i^{(i)}), \quad \sum_{i=1}^{N} w_i^{(i)} = 1$$

For $i = 1,\ldots,N$:

$$x_{t+1}^{(i)} \sim q(x_{t+1}|x_t^{(i)})$$

Update the importance weight:

$$w_{t+1}^{(i)} \propto w_t^{(i)} \frac{g(y_{t+1}|x_{t+1}^{(i)}) f(x_{t+1}^{(i)}|x_t^{(i)})}{q(x_{t+1}^{(i)}|x_t^{(i)})}$$

Optionally, resample $\{x^{(i)}_{t+1}\} N$ times with replacement using weights $w_{t+1}^{(i)}$, and then resetting $w_{t+1}^{(i)} = 1/N$.  

The algorithm is now modified to:

For $t = 0, 2, ..., T$

For $i = 1, 2, ..., N$

$x(i)_{t+1} \sim q(x(i)_{t+1} | x(i)_t, y_0)$

$w(i)_{t+1} \propto w(i)_t g(y_{t+1} | x(i)_{t+1}) f(x(i)_t | x(i)_t)$

$q(x(i)_{t+1} | x(i)_t, y_0)$

End

End

(Optional) Resample $x(i)_{t+1}$ with probability $w(i)_{t+1}$. Set $w(i)_{t+1} = 1/N$

End

End
General Sequential Importance Sampling (Part II!)

Question: can we rationalise the resampling step? Why use it, and how should it ideally be used?

Answer: yes! Make it part of the importance sampling.

Consider the problem from the point of view of joint simulation from the smoothing density.

- ‘Particle’ approximation to smoothing density:

\[
p(x_{0:t}|y_{0:t}) \approx \hat{p}(x_{0:t}|y_{0:t}) = \sum_{i=1}^{N} w_{t}^{(i)} \delta(x_{0:t} - x_{0:t}^{(i)}), \quad \sum_{i=1}^{N} w_{t}^{(i)} = 1
\]

where \( w_{t}^{(i)} \) is the ‘weight’ of particle \( x_{0:t}^{(i)} \). Note that \( w_{t}^{(i)} \) will be uniform at time steps where resampling has been carried out.
• Factorise smoothing density at $t + 1$:

$$p(x_{0:t+1}|y_{0:t+1}) = p(x_{0:t}|y_{0:t}) \times \frac{g(y_{t+1}|x_{t+1})f(x_{t+1}|x_t)}{p(y_{t+1}|y_{0:t})}$$

• Apply importance function $q(x_{0:t+1})$:

$$w_{t+1} \propto p(x_{0:t}|y_{0:t}) \times \frac{g(y_{t+1}|x_{t+1})f(x_{t+1}|x_t)}{q(x_{0:t+1})}$$
In a sequential setting we assume that $\hat{p}(x_{0:t}|y_{0:t}) = p(x_{0:t}|y_{0:t})$ is the ‘truth’ (the accumulation of error can be analysed) and apply an importance function which fixes the past history of the new particles to be one of the current particles $x_{0:t}^{(j)}$. This corresponds to choosing an importance function of the form:

$$q(x_{0:t+1}) = q(x_{t+1}|x_{0:t}) \sum_{i=1}^{N} v_i(t) \delta_{x_{0:t}^{(i)}}(x_{0:t})$$

Importance weights can then be computed as (see Godsill and Clapp (2001)):

$$w_{t+1}^{(j)} \propto \frac{w_t^{(j)}}{v_t^{(j)}} \times \frac{g(y_{t+1}|x_{t+1}^{(j)})f(x_{t+1}^{(j)}|x_t^{(j)})}{q(x_{t+1}^{(j)}|x_{0:t}^{(j)})}$$

where $x_{0:t+1}^{(j)} \Delta (x_{0:t}^{(j)}, x_{t+1}^{(j)})$ is the $j$th particle drawn from the joint distribution $q(x_{0:t+1})$. 
• Note the non-standard term here, $v_t^{(j)}$, which serves to boost or diminish the particle histories in the importance function. This weight $v_t$ should intuitively reflect a particle's importance at $t + 1$, whereas its original weight $w_t$ reflects its importance at $t$.

• Drawing from $q(x_{0:t})$ with weights $v_t$ can be interpreted as the usual ‘resampling’ or ‘selection’ step found in particle filters. The resampling is performed first, but this is only a conceptual difference. It can be performed in a stratified or part-deterministic form to reduce estimator variance.
• This formulation is of interest because it expresses several common variants on particle filtering in one framework. In particular:
This formulation is of interest because it expresses several common variants on particle filtering in one framework. In particular:

- \( v_t^{(j)} = w_t^{(j)} \) - this corresponds to the standard SIR filtering method, e.g. the bootstrap filter of Gordon et al. (93). Particles are resampled according to their weight at time \( t \).

- \( v_t^{(j)} = 1/N \) - filter with ‘no resampling’ (‘sequential imputations’). In practice would implement in fully stratified form by simply selecting particles deterministically 1...N.
- \( v_t^{(j)} \propto g(y_{t+1}|\hat{x}_{t+1}^{(j)}) \), where \( \hat{x}_{t+1}^{(j)} \) is some (any!) ‘estimate’ of the new state. This is the auxiliary particle filter of Pitt and Shephard (1998). This makes use of the particle’s importance at time \( t + 1 \) in an efficient way.

- ‘Fully adapted’ version of auxiliary filter

\[
v_t^{(j)} \propto p(y_{t+1}|x_t^{(j)}, y_{0:t}) = \int p(y_{t+1}, x_{t+1}|x_t^{(j)}, y_{0:t}) dx_{t+1}
\]

- not generally computable.
Rao-blackwellised filtering

• So far we have considered generic models without any significant constraints on their structure.

• What, though, if there is some analytic structure in the model, e.g. some component of the model is linear/Gaussian, or a Hidden Markov model (HMM)?
Rao-blackwellised filtering

- So far we have considered generic models without any significant constraints on their structure.

- What, though, if there is some analytic structure in the model, e.g. some component of the model is linear/Gaussian, or a Hidden Markov model (HMM)?

- One answer is to mix the analytic filtering results of the Kalman filter or HMM filter with the particle filter: ‘Rao-Blackwellised’, ‘Mixture Kalman’ or ‘Marginalised’ particle filter - see Doucet, Godsill and Andrieu (2000), Chen and Liu (2000), Schon, Gustafsson, Nordlund (2005).

- Split the state vector into two parts $x^L$ (‘good cop’) and $x^N$ (‘bad cop’).
• The ‘good’ part of the state, $x_L^t$, can be written as a linear state space model:

$$x_L^t = A(x_N^t)x_L^{t-1} + u_L^t, \quad (27)$$

$$y_t = B(x_N^t)x_L^t + v_L^t. \quad (28)$$

Here $u_L^t$ and $v_L^t$ are independent, zero-mean, Gaussian disturbances with covariances $C_u$ and $C_v$, respectively. Nonlinear state $x_N^t$.

• Now the ‘bad’ nonlinear part of the state obeys a general dynamical model:

$$x_N^t \sim f(x_N^t|x_N^{0:t-1}), \quad x_N^0 \sim f(x_N^0). \quad (29)$$
The idea is now to marginalise the linear state sequence: the calculations can be done exactly and sequentially by the Kalman filter likelihood evaluation!

\[ p(x_{0:t}^N|y_{0:t}) = \int p(x_{0:t}^L, x_{0:t}^N|y_{0:t}) dx_{0:t}^L . \]

Particle filtering is then run on the nonlinear state sequence only.

- **Prediction**
  \[ p(x_{0:t}^N|y_{0:t-1}) = p(x_{0:t-1}^N|y_{0:t-1}) f(x_{t}^N|x_{0:t-1}^N) . \]

- **Correction**
  \[ p(x_{0:t}^N|y_{0:t}) = \frac{p(y_{t}|y_{0:t-1}, x_{0:t}^N)p(x_{0:t}^N|y_{0:t-1})}{p(y_{t}|y_{0:t-1})} , \]
\{ (x_{0:t}^{N,(i)}), \omega_t^{(i)} \}_{i=1,...,N} \text{ denote the nonlinear state particles. Then,}

\textbf{Rao-Blackwellised} estimation scheme for } x^L \text{ is obtained as a random Gaussian mixture approximation given by}

\[ p(x_t^L | y_{0:t}) \approx \sum_{i=1}^{N} \omega_t^{(i)} p(x_t^L | x_{0:t}^{N,(i)}, y_{0:t}) , \]  

(30)

where the conditional densities \( p(x_t^L | x_{0:t}^{N,(i)}, y_{0:t}) \) are Gaussian and computed using Kalman filtering recursions (likelihood evaluation).

Since \( p(x_t^L | x_{0:t}^{N,(i)}, y_{0:t}) \) depends on the particle \( x_{0:t}^{N,(i)} \), the Kalman filter must be run for every particle - very expensive - performance/computation trade-off.
Sources for Today’s Lecture

- “doc.pdf”
- http://www-sigproc.eng.cam.ac.uk/~sjg
- Cappe, Moulines, and Ryden, “Inference in Hidden Markov Models”