ECE 6123
Advanced Signal Processing:
Markov Chain Monte Carlo

Peter Willett

Fall 2017

1 Importance Sampling

1.1 Estimation of Small Probabilities

Suppose we want to estimate a small probability

$$\alpha \equiv Pr(x \in \Omega) \tag{1}$$

This may sound trivial, and it would be if we were interested, say, in the $\Omega = \{x : x > \tau\}$. But suppose it is not so simple, and Ω is the set of noise samples[1] that produces an error in an OFDM system with LDPC, zero-forcing equalization and carrier-offset recovery. We have no hope of an analytic probability calculation, all we can do is simulate and count the errors. That is, we estimate

$$\hat{\alpha} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}(x_i \in \Omega) \tag{2}$$

where $\mathcal{I}$ is the indicator, N is the number of Monte Carlo trials, these indexed by i. It is vary simple to see that

$$\mathcal{E}\{\hat{\alpha}\} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{E}\{\mathcal{I}(x_i \in \Omega)\} \tag{3}$$

$$= \int_{\Omega} p(x)dx = \alpha \tag{4}$$

which is good news, but

$$\frac{Var\{\hat{\alpha}\}}{(\mathcal{E}\{\hat{\alpha}\})^2} \approx \frac{\alpha/N}{\alpha^2} = \frac{1}{N\alpha} \tag{5}$$

[1] Don't worry if this OFDM stuff means nothing to you; the point is that it's a complicated event.

which is not good news. Equation (5) means that if you want the standard deviation of $\hat{\alpha}$ to be (say) less than 10% of its value, you need $N > 100/\alpha$ MC trials; and if α is 10^{-8} this can be a chore.

Fortunately we have *importance sampling* to help. Consider a new estimator

$$\hat{\alpha} \;=\; \sum_{i=1}^{N} \mathcal{I}(x_i \in \Omega) \frac{p(x_i)}{q(x_i)} \tag{6}$$

where $p(\cdot)$ is the true probability density governing whatever is random about your problem and $q(\cdot)$ some other "importance" pdf that the samples used to estimate $\hat{\alpha}$ are actually drawn from. For example, we might have $\Omega = \{(x_1, x_2) : (x_1 - 10)^2 + (x_2 - 15)^2 \leq 1\}$ and $p(\cdot)$ bivariate Gaussian with mean zero and unity variance. It is fairly clear that $\hat{\alpha}$ from (2) will include exactly no indicators that "happen" for any reasonable value of N – it is useless. But suppose we use (6) with $q(\cdot)$ to mean a Gaussian pdf with mean $(10, 15)$ and variance of 0.5: then many indicators will fire, and each of them will force the inclusion to the sum in (6) of many relatively small values determined by the *importance weights* $p(x_i)/q(x_i)$.

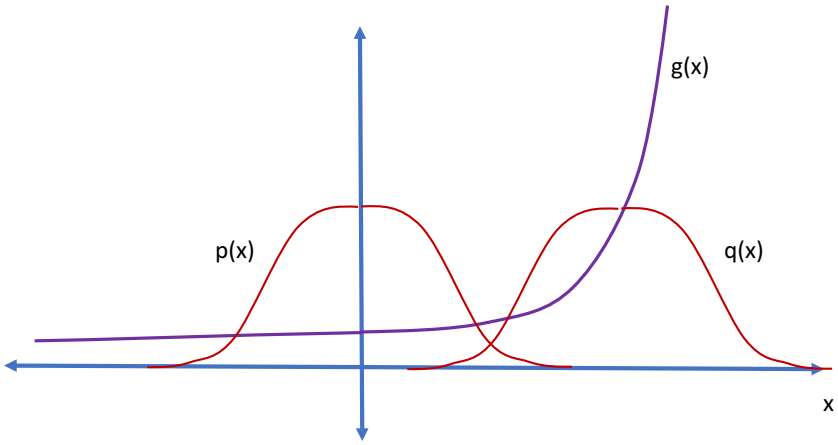
The variance of (6) is easily seen to be

$$Var(\hat{\alpha}) \;=\; \frac{1}{N} \left(\int_{\Omega} \frac{p(x)^2}{q(x)} dx - \alpha^2 \right) \tag{7}$$

which is illuminating for two reasons. The first is that it is minimized (actually cut down to zero) by

$$q(x) \;=\; p(x|x \in \Omega) \;=\; \frac{p(x)\mathcal{I}(x \in \Omega)}{\alpha} \tag{8}$$

which gives us the helpful information that if we already knew the answer we could easily use MC techniques to find the answer. This actually really *is* useful, since it tells us that there is no "magic bullet" for importance-sampling – choosing a good $q(\cdot)$ is an art form. But (7) also suggests intuition: if we want to have a low variance we should try to reduce the variation of $p(\cdot)/q(\cdot)$ of Ω as much as we can. By that logic choosing $q(\cdot)$ to have mean $(9, 14)$ and unity variance in the previous example may be better than the $q(\cdot)$ given; and mean mean $(11, 16)$ worse.



1.2 Importance Sampling for Moments

Consider the situation as above, in which we wish to calculate the expected value of a function $g(x)$ under the pdf $p(x)$. A direct MC implementation will probably not work very well, since the "active" part of $g(x)$ occurs where samples from $p(x)$ are rare. Suppose instead we simulate under $q(x)$ as also indicated in the plot. Then we get

$$\bar{g} \;=\; \frac{1}{N}\sum_{i=1}^{N} g(x_i)\frac{p(x_i)}{q(x_i)} \tag{9}$$

so that

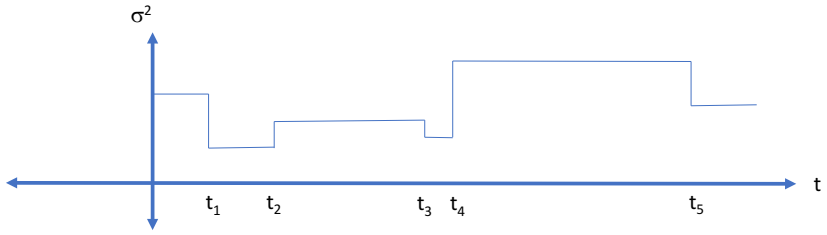
$$\mathcal{E}\{\bar{g}(x)\} \;=\; \int g(x)\frac{p(x)}{q(x)}q(x)dx \;=\; \mathcal{E}\{g(x)\} \tag{10}$$

meaning that the importance-sampling estimator is unbiased for this case, too.

2 Motivation for Markov Chain Monte Carlo

2.1 Segmentation

Consider the problem that we are given a record of N data: $\{u[n]\}$ is zero mean, independent and Gaussian. There are M segments to the data, such that if $t_{i-i} \le n < t_i$ then the variance of $u[n]$ is σ_i^2 – see below. The problem is that we don't know the t_i's (but naturally assume $t_0 = 0$ and $t_M = N-1$) and we don't know the σ's. What do we do?



Suppose we *did* know the t_i's. Then solving for σ_i is fairly simple:

$$\hat{\sigma}_i^2 = \frac{1}{t_i - t_{i-1}} \sum_{n=t_{i-1}}^{t_i-1} u[n]^2 \tag{11}$$

is the maximum-likelihood estimate (MLE). But finding the t_i's is more of a problem.

Define $\mathbf{t}_{\bar{i}} \equiv \{t_0, \ldots, t_{i-1}, t_{i+1}, \ldots, t_M\}$ and assume the prior information is (on $\mathbf{t}$, say) is uniform. We write

$$\begin{aligned}
p(t_i|\mathbf{t}_{\bar{k}}, \mathbf{u}) &= \frac{p(\mathbf{t}|\mathbf{u})}{p(\mathbf{t}_{\bar{i}}|\mathbf{u})} && (12)\\
&\propto p(\mathbf{t}|\mathbf{u}) && (13)\\
&\propto p(\mathbf{u}|\mathbf{t}) && (14)\\
&\propto p(\{u[n]\}_{n=t_{i-1}}^{t_i-1}|\sigma_i^2) \times p(\{u[n]\}_{n=t_i}^{t_{i+1}-1}|\sigma_{i+1}^2) && (15)\\
&= \left(\prod_{n=t_{i-1}}^{t_i-1} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{u[n]^2}{2\sigma_i^2}}\right)\left(\prod_{n=t_i}^{t_{i+1}-1} \frac{1}{\sqrt{2\pi\sigma_{i+1}^2}} e^{-\frac{u[n]^2}{2\sigma_{i+1}^2}}\right) && (16)
\end{aligned}$$

where (14) follows from a assumption of uniformity on $\mathbf{t}$ and (15) from the fact that none of the other segments depends on t_k, only the one preceding and succeeding it. Note that (15) is a set of $t_{i+1} - t_{i-1} - 1$ likelihoods that can be normalized to give a probability mass function. An algorithm follows:

1. Generate some initial $t_i(0)$'s. The initial set does not matter, but a uniform spacing is probably best. Set the iteration counter $k = 1$.
2. Calculate $\{\hat{\sigma}_i^2(k)\}_{i=1}^M$ according to (11).
3. Set $i = 1$.
4. Draw $t_i(k)$ according to

   $$t_i(k) \sim p(\{u[n]\}_{n=t_{i-1}(k)}^{t_i-1}|\sigma_i^2(k)) \times p(\{u[n]\}_{n=t_i}^{t_{i+1}(k-1)-1}|\sigma_{i+1}^2(k)) \tag{17}$$

   This is from (15) and is made explicit in (16).

5. Set $i \leftarrow i + 1$ and if $i < M$ go to 4.

6. Set $k \leftarrow k + 1$ and go to 2 if $k \leq K$.

Here K is the number of iterations to perform, and K_b is some number that will be elided as "burn-in" samples. At the end, estimate

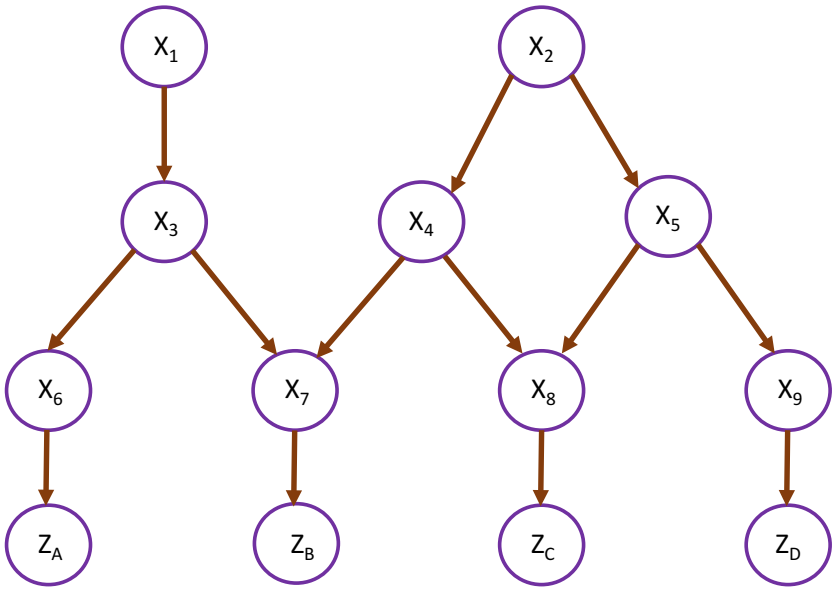
$$\hat{t}_i \;=\; \frac{1}{K - K_b} \sum_{k=K_b+1}^{K} t_i(k) \tag{18}$$

for the average. Actually we could take the variance as well to determine our posterior variance, as we shall see. But what gives us any right to do such an operation and expect any meaning at the end?

2.2 Bayesian Inference Networks

Actually the procedure just discussed[2] is *Gibbs sampling*, which is far more general. An example, perhaps the canonical one, is the Bayesian Inference Network (BIN), pictured below. The arrows indicate known conditional probabilities, which are assumed known. Each "node" x_i is a hidden state variable, and the z's are observations – and of these it is possible that only a subset is known. For example, x_2 might be a stock valuation: underpriced (0), fairly-priced (1) or overpriced (2) – clearly this is a hidden node that you are interested in. Maybe x_4 is institutional interest in the stock (yes / no); and x_1 is the company's growth potential. Observation z_a might be existence of a dividend, and z_c is the company's price-to-earnings ratio – these are both something you can observe. Finally, let's say that z_b is whether there have been buys of the stock by company insiders – this is something you might know, but might not.

[2] Actually it was not quite Gibbs sampling, since the MLE step for the σ's has no place with Gibbs.



The same approach as in the previous segmentation example can be used there. It works best if the nodes can only take on a finite number of values, but that is not necessary. Specifically, do the following:

1. Initialize the instantiated observation nodes – those z's that you know – to their true values. These will never change, of course.

2. Initialize all other nodes to random values: that is, the $x^{(0)}{}_i$'s and also the un-instantiated z's. Set $k = 1$.

3. For all (uninstantiated) nodes calculate

$$p(x_i) \;=\; \kappa \left(\prod_{j \in S_p} p(x_i | x_j^{(k-1)}) \right) \left(\prod_{j \in S_c} p(x_j^{(k-1)} | x_i) \right) \tag{19}$$

   for all possible values of x_i, where κ normalizes the sum over these to unity. The set S_p indicates the *parent* nodes of x_i and S_c the child nodes; see below for an example.

4. Draw new x_i's from the pmf's calculated in the previous step.

5. Set $k \leftarrow k + 1$ and go to 3 if $k \leq K$.

For example, for x_4 we have $S_p = \{x_2\}$ and $S_c = \{x_7, x_8\}$; that is,

$$p(x_4 | \mathbf{x}_{\bar{4}}^{(k-1)}, \mathbf{z}) \;\propto\; p(x_4, \mathbf{x}_{\bar{4}}^{(k-1)} | \mathbf{z}) \tag{20}$$

$$\propto\; p(x_4 | x_2^{(k-1)}) \times p(x_7^{(k-1)} | x_4) \times p(x_8^{(k-1)} | x_4) \tag{21}$$

Note that although in this algorithm it seems like one generates all of the $\mathbf{x}^{(k)}$ based on $\mathbf{x}^{(k-1)}$, it is a perfectly legal algorithm that simply uses whatever the present node values might be, whether updated or not; that is, (19) may use a combination of $x_j^{(k)}$'s and $x_j^{(k-1)}$'s.

3 Metropolis-Hastings Algorithm

3.1 Theory

Consider these steps, which form the MH (meta-) algorithm:

1. Initialize x^0. Set $n = 1$.
2. Generate y according to $q(y|x^n)$.
3. Generate u uniform on $(0, 1)$.
4. Form
$$\alpha(x^n, y) \;=\; \min\left\{1, \frac{\pi(y)q(x^n|y)}{\pi(x^n)q(y|x^n)}\right\} \tag{22}$$
In this step $\pi(\cdot)$ is the probability (mass function or density) of the system you are investigating.
5. If $u \leq \alpha(x^n, y)$ set $x^{n+1} = y$. Otherwise keep $x^{n+1} = x^n$.
6. Increment $n \leftarrow n + 1$. Go to 2 unless finished with iterations.

The choice of $q(\cdot|\cdot)$ is a matter of tuning. The superscript for x refers to the iteration number.

The probability $p(\cdot)$ is assumed to be available. This latter may seem strange, but in many problems the overall probability is explicit and what is sought is a marginal probability of some component. Turning the BIN previously pictured the overall probability is actually fairly simple to write. In fact it is

$$\begin{aligned}\pi(\mathbf{x}, \mathbf{z}) \;=\;& p(z_A|x_6)p(z_B|x_7)p(z_C|x_8)p(z_D|x_9) \\ & \times p(x_6|x_3)p(x_7|x_3, x_4)p(x_8|x_4, x_5)p(x_9|x_5) \\ & \times p(x_3|x_1)p(x_4|x_2)p(x_5|x_2)p(x_1)p(x_2)\end{aligned} \tag{23}$$

which is, as advertised, simple; but $p(x_4|z_A, z_C)$ is not at all simple and would involve a great deal of awkward summation. Another example is nonlinear filtering: it is (relatively) easy to write $\pi(x_1, \ldots, x_t, z_1, \ldots, z_t)$; but it is not easy to find $p(x_t|z_1, \ldots, z_t)$. These are cases in which $\pi(\cdot)$ is

explicit; our goal is to generate samples of $\mathbf{x}$ (like $\{x_1, \ldots, x_t\}$) such that we can trivially investigate subsets of them (like x_t).

Let us first note that we have

$$\frac{\alpha(u,v)}{\alpha(v,u)} \;=\; \frac{\pi(v)q(u|v)}{\pi(u)q(v|u)} \tag{24}$$

or

$$\pi(u)q(v|u)\alpha(u,v) \;=\; \pi(v)q(u|v)\alpha(v,u) \tag{25}$$

since either the numerator or denominator of the LHS must have been "clamped" at unity.

Now, (22) instructs us that we have

$$\begin{aligned} p(x^{n+1}|x^n) &= \alpha(x^n, x^{n+1})q(x^{n+1}|x^n) \\ &\quad + \delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^n, y)q(y|x^n)dy\right) \end{aligned} \tag{26}$$

Multiply (26) by $\pi(x^n)$ and we get

$$\begin{aligned} \pi(x^n)p(x^{n+1}|x^n) &= \pi(x^n)\alpha(x^n, x^{n+1})q(x^{n+1}|x^n) \\ &\quad + \pi(x^n)\delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^n, y)q(y|x^n)dy\right) \end{aligned} \tag{27}$$

and inserting (25) we have

$$\begin{aligned} \pi(x^n)p(x^{n+1}|x^n) &= \pi(x^{n+1})\alpha(x^{n+1}, x^n)q(x^n|x^{n+1}) \\ &\quad + \pi(x^n)\delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^n, y)q(y|x^n)dy\right) \end{aligned} \tag{28}$$

The δ-function allows us to switch the terms, hence we have

$$\begin{aligned} &\pi(x^n)p(x^{n+1}|x^n) \\ &\quad= \pi(x^{n+1})\alpha(x^{n+1}, x^n)q(x^n|x^{n+1}) \\ &\qquad + \pi(x^{n+1})\delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^{n+1}, y)q(y|x^{n+1})dy\right) \end{aligned} \tag{29}$$

$$= \pi(x^{n+1})p(x^n|x^{n+1}) \tag{30}$$

Equation (30) tells us that we can identify $\pi(x^n)$ and $p(x^{n+1}|x^n)$ as respectively the stationary and transition probability mass functions (or densities) of a Markov chain. Thus, discarding *burn-in* (transient) x^n's at the beginning, we know that the x^n's that we accumulate by following the given procedure are distributed according to $\pi(\cdot)$. That is, we can do things like take an average over one of the dimensions to get an expected value.

It is interesting that the *correctness* of the MH algorithm does not depend on the choice of $q(\cdot|\cdot)$. Nonetheless, the *efficiency* is strongly connected to $q(\cdot|\cdot)$: an "aggressive" $q(\cdot|\cdot)$ can cover a lot of ground, but may end up rejecting many putative samples via $\alpha(\cdot,\cdot)$; similarly a timid $q(\cdot|\cdot)$ does not waste samples but may take many iterations to explore its space.

3.2 Special Cases of Metropolis-Hastings

3.2.1 Metropolis Sampler

This is probably the original version, and uses a conditional density $q(\cdot|\cdot)$ such that

$$q(v|u) \;=\; q(u|v) \tag{31}$$

An example of such a density is the (obvious) Gaussian: under $q(v|u)$, v is Gaussian with mean u. If the Metropilis version is used we have

$$\alpha(x^n, y) \;=\; \min\left\{1, \frac{\pi(y)}{\pi(x^n)}\right\} \tag{32}$$

The statements recently made about aggressive or timid $q(\cdot|\cdot)$ are very clear in light of (32).

3.2.2 Independence Sampler

The independence sampler uses

$$q(v|u) \;=\; q(v) \tag{33}$$

and hence

$$\alpha(x^n, y) \;=\; \min\left\{1, \frac{\pi(y)q(x^n)}{\pi(x^n)q(y)}\right\} \tag{34}$$

Equation (33) means that its mode of exploration does not depend on its current knowledge at all – the second pair of MC's in MCMC are suspect. But it works, and can be thought of as a way to understand the celebrated "bootstrap" particle filter.

3.2.3 Gibbs Sampler

This one is slightly trickier to understand. The key is to acknowledge that x^n and y are actually multi-dimensional. Let us use y_i to refer to the i^{th} dimension of y; and $y_{\bar{i}}$ to refer to all dimensions except the i^{th}. Then according to the Gibbs idea we use

$$q(y_i|x^n_{\bar{i}}) \;=\; \pi(y_i|x^n_{\bar{i}}) \tag{35}$$

meaning that we draw a new i^{th} dimension using the *true* pdf based on all other dimensions (which are unchanged). In the BIN formulation we can see how this is accomplished: for our example, we know how to draw x_4 based on $x_{\bar{4}}$; and this amounts to (21). Now what is especially interesting about the Gibbs sampler is that we have

$$\alpha(x_{\bar{i}}^n, y_i) \;=\; \min\left\{1, \frac{\pi(x_i|x_{\bar{i}}^n)\pi(y_i|x_{\bar{i}}^n)}{\pi(y_i|x_{\bar{i}}^n)\pi(x_i|x_{\bar{i}}^n)}\right\} \;=\; 1 \tag{36}$$

which means that the Gibbs Sampler never "rejects".

4 Particle Filters

The *particle filter*, an approach to the solution of the *Chapman-Kolmogorov* equation (CKE) via a Monte Carlo (MC) method, has evolved considerably over the last years, and there are many versions. The basic idea is most easily explained using the first version that was *feasible*, known as the bootstrap filter or the *sequential importance sampling/resampling (SIR or SIS)*.

Consider the general Markov chain "target" pdf model, which may be nonlinear and/or non-Gaussian,

$$p(x_{[1:t]}) = p(x_1)\prod_{k=2}^{t} p(x_k|x_{k-1}) \tag{37}$$

in which x_k is the n_x-dimensional target state at time k,

$$x_{[1:t]} \equiv \{x_1, x_2, \ldots, x_t\} \tag{38}$$

and $p(x_k|x_{k-1})$ is the *state transition pdf*. The assumption of white process noise is what allows one to write (37) in the product form. As usual, the measurement sequence must be – conditioned on the state – independent (i.e., white measurement noise), in which case

$$p(z_{[1:t]}|x_{[1:t]}) = \prod_{k=1}^{t} p(z_k|x_k) \tag{39}$$

which again is a fairly arbitrary collection of conditionally-independent pdfs.

To develop the idea of the bootstrap filter, consider the following. Using the Monte Carlo method, N samples $\{x_{[1:t]}^i\}_{i=1}^N$ — *multiscan particles*, which are $n_x t$ vectors — are drawn from the prior density function $p(x_{[1:t]})$ shown

in (37) Then the likelihood of each such sample is evaluated according to $p(z_{[1:t]}|x_{[1:t]})$ in (39). We can normalize these likelihoods to "weights"

$$\omega^i_{[1:t]} = \frac{1}{c} p(z_{[1:t]}|x^i_{[1:t]}) \qquad\qquad i = 1, \ldots, N \tag{40}$$

where c is chosen such that $\sum_{i=1}^N \omega^i_{[1:t]} = 1$, i.e.,

$$c \equiv \sum_{i=1}^N p(z_{[1:t]}|x^i_{[1:t]}) \tag{41}$$

The weights in (40) are probabilities, assuming equal priors for the samples $\{x^i_{[1:t]}\}_{i=1}^N$. Then one can claim that the desired posterior $p(x_{[1:t]}|z_{[1:t]})$ is well represented by the point-mass pdf (or pmf)

$$\hat{p}[x_{[1:t]}|z_{[1:t]}] = \sum_{i=1}^N \omega^i_{[1:t]} \delta(x_{[1:t]} - x^i_{[1:t]}) \tag{42}$$

assuming N, the number of particles, is sufficiently large.

That the representation is reasonable for the posterior mean

$$\begin{aligned} \hat{x}_{[1:t]} &\equiv \int x_{[1:t]} \hat{p}(x_{[1:t]}|z_{[1:t]}) dx_{[1:t]} \\ &= \int \sum_{i=1}^N \omega^i_{[1:t]} x_{[1:t]} \delta(x_{[1:t]} - x^i_{[1:t]}) dx_{[1:t]} \\ &= \sum_{i=1}^N \omega^i_{[1:t]} x^i_{[1:t]} \end{aligned} \tag{43}$$

is straightforward to see as follows. Consider the expected value of (43) over the samples $\{x^i_{[1:t]}\}_{i=1}^N$

$$\begin{aligned} E\left\{\hat{x}_{[1:t]}|z_{[1:t]}\right\} &= E\left\{\sum_{i=1}^N \omega^i_{[1:t]} x^i_{[1:t]}\right\} \\ &= \int \sum_{i=1}^N \omega^i_{[1:t]} x^i_{[1:t]} p(x^i_{[1:t]}) dx^i_{[1:t]} \\ &= \int \sum_{i=1}^N \frac{1}{c} p(z_{[1:t]}|x^i_{[1:t]}) x^i_{[1:t]} p(x^i_{[1:t]}) dx^i_{[1:t]} \\ &\approx \int \sum_{i=1}^N \frac{1}{N p(z_{[1:t]})} p(z_{[1:t]}|x^i_{[1:t]}) x^i_{[1:t]} p(x^i_{[1:t]}) dx^i_{[1:t]} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{N}\sum_{i=1}^{N}\int x^i_{[1:t]}p(x^i_{[1:t]}|z_{[1:t]})dx^i_{[1:t]} \\
&= E\left\{x_{[1:t]}|z_{[1:t]}\right\} \qquad (44)
\end{aligned}
$$

The approximation in (44) is that

$$
\begin{aligned}
p(z_{[1:t]}) &= \int p(z_{[1:t]}|x_{[1:t]})p(x_{[1:t]})dx_{[1:t]} \\
&\approx \int p(z_{[1:t]}|x_{[1:t]})\sum_{i=1}^{N}\frac{1}{N}\delta(x_{[1:t]}-x^i_{[1:t]})dx_{[1:t]} \\
&= \frac{1}{N}\sum_{i=1}^{N}p(z_{[1:t]}|x^i_{[1:t]}) = \frac{c}{N} \qquad (45)
\end{aligned}
$$

The above forms the basis for particle filter-based track-likelihood evaluation and track-testing

$$
\begin{aligned}
p(z_{[1:t]}) &= p(z(t)|z_{[1:t-1]})p(z_{[1:t-1]}) \\
&= p(z_{[1:t-1]})\int p(z_t|x_t)p(x_t|z_{[1:t-1]})dx_t \\
&\approx p(z_{[1:t-1]})\int p(z_t|x_t)\frac{1}{N}\sum_{i=1}^{N}\delta[x_t-x^i_t]dx_t \\
&= p(z_{[1:t-1]})\frac{1}{N}\sum_{i=1}^{N}p(z_t|x^i_t) \\
&= \prod_{k=1}^{t}\left(\frac{1}{N}\sum_{i=1}^{N}p(z_k|x^i_k)\right) \qquad (46)
\end{aligned}
$$

similar to the χ^2 statistic that one might use in a Kalman filter context, but for more general models. The track likelihood (46) is recognizable as the product of *unnormalized* particle weights (which have to be calculated anyway), and certainly the second formula in (46) shows that it can be evaluated iteratively.

Consequently an efficient means to generate $\{x^i_{[1:t]}\}_{i=1}^N$ — recall that for an n_x-dimensional state, each $x^i_{[1:t]}$ is $n_x t$ dimensional — is key. The steps are as follows:

Prediction: Given $x^i_{[1:t-1]}$ we draw x^i_t from it according to

$$
x^i_t \sim p(x^i_t|x^i_{t-1}) \qquad (47)
$$

and thence augment $x^i_{[1:t-1]}$ to $x^i_{[1:t]}$.

Update: Calculate

$$\begin{aligned}\omega^i_{[1:t]} &= \frac{1}{c}p(z_{[1:t]}|x^i_{[1:t]}) \\ &= \frac{1}{c'}p(z_t|x^i_t)\omega^i_{[1:t-1]} \end{aligned} \tag{48}$$

The last line above will, in general, require a new normalization.

Note that (47) and (48) imply that it is not necessary to work with $n_x t$-dimensional particles $\{x^i_{[1:t]}\}_{i=1}^N$ and weights $\{\omega^i_{[1:t]}\}_{i=1}^N$. Instead, as a practical matter all that one need retain is their value at time t: $\{x^i_t\}_{i=1}^N$ and weights $\{\omega^i(t)\}_{i=1}^N$, which may be taken as a statement that $\{x^i_t\}_{i=1}^N$ and $\{\omega^i(t)\}_{i=1}^N$ are sufficient for $x(t)$ given observations $\{z_k\}_{k=1}^t$. With reference to (43), any moment of $x(t)$ can be approximated from these alone.

To be specific about these steps, if one were to use a particle filter to estimate in a linear/Gaussian situation (which one never would — one would use a Kalman filter) one would *predict* by drawing the i^{th} particle at time t as $x^i_t = Fx^i_{t-1} + v^i_t$, where v^i_t is simply the realization of a Gaussian random vector with covariance Q. The weight for the i^{th} particle is the likelihood

$$\omega^i(t) = \frac{1}{c}\frac{1}{\sqrt{|2\pi R|}}e^{-\frac{1}{2}[z_t - Hx^i_t]'R^{-1}[z_t - Hx^i_t]} \tag{49}$$

with appropriate normalization.

Unfortunately there is a problem — *particle degeneracy* — namely, the tendency for all the weights save one to go to zero. This tendency — really a compulsion in any nontrivial case — arises from the product in (48), and it severely limited the acceptance of Monte Carlo approaches in their early days. *Fortunately* there is a solution: *resampling*. The especially easy resampling in the bootstrap filter is to sample *with replacement* from $\{x^i_t\}_{i=1}^N$ according to probabilities $\{\omega^i(t)\}_{i=1}^N$. We note that in a mathematical sense this sampling is optional, and in some implementations it is performed only when degeneracy seems to be occurring, not necessarily at every update step. It is important to realize that such resampling can (and usually does) result in many repeated (i.e., copied) particles, those corresponding to the largest likelihoods (the largest $\omega^i(t)$'s). This is an evanescent concern, since the next prediction step in (47) adds different "noise" to each of these.

We therefore restate the operation of the bootstrap filter as

Prediction: For $i = 1, \ldots, N$ draw $\tilde{x}^i_t$ from x^i_{t-1} according to

$$\tilde{x}^i_t \sim p(x^i_t|x^i_{t-1}) \tag{50}$$

Update: For $i = 1, \ldots, N$ calculate and normalize to unity-sum the weights

$$\omega^i(t) = \frac{1}{c} p(z_t | \tilde{x}_t^i) \tag{51}$$

Resampling: Draw $\{x_t^i\}_{i=1}^N$ from $\{\tilde{x}_t^i\}_{i=1}^N$ according to the pmf $\{\omega^i(t)\}_{i=1}^N$.

This shows that multiscan particles, while helpful to intuition, are not part of a practical particle filter system. Note that the fact that the resampling operation at $t-1$ used $\{\omega^i(t-1)\}_{i=1}^N$, makes it inappropriate to use these weights again in the update step for $\{\omega^i(t)\}_{i=1}^N$: only the present likelihoods are used.

One interesting variation on the particle filter is having $x^i(t)$ drawn according to some alternative *proposal density* $q(x_t^i | x_{t-1}^i)$ for the prediction instead of the prior (or transition) density $p(x_t^i | x_{t-1}^i)$. Then this "incorrect" prediction step can be exactly canceled in the typical importance sampling manner by using the appropriate importance weight. That is, the three particle filtering steps become

Importance-Weighted Prediction: Draw $\tilde{x}_t^i$ from x_{t-1}^i, according to

$$\tilde{x}^i(t) \sim q(\tilde{x}_t^i | x_{t-1}^i) \tag{52}$$

Importance-Weighted Update: Calculate

$$\omega_{[1:t]}^i = \frac{1}{c} p(z_t | \tilde{x}_t^i) \frac{p(\tilde{x}_t^i | x_{t-1}^i)}{q(\tilde{x}_t^i | x_{t-1}^i)} \tag{53}$$

Resampling: Draw $\{x_t^i\}_{i=1}^N$ from $\{\tilde{x}_t^i\}_{i=1}^N$ via the pmf $\{\omega^i(t)\}_{i=1}^N$.

The resampling step is unaffected. That (53) is appropriate, is easily checked by a development similar to (44)

A properly chosen $q(\cdot|\cdot)$ can "steer" particles to be predicted to places where they are likely to be corroborated by measurements, namely, (52) becomes

$$\tilde{x}_t^i \sim q(\tilde{x}_t^i | x_{t-1}^i; z_t) \tag{54}$$

This is as opposed to the standard procedure (47) and (48) where many predicted particles can be, in effect, wasted with essentially-zero weights, leaving relatively few (or no!) "working" particles near where the measurement makes them likely. Common choices for the proposal density include the transition density with inflated noise, the EKF and UKF.

The key step – which must be credited to Gordon, Salmond & Smith – in making particle filters practical was resampling and that estimation is

performed recursively in time, as opposed to the batch approach with the notional "multiscan particles" discussed at the beginning of this section. In addition to proposal density improvements there have been many interesting variations. One of the most widely accepted is the auxiliary particle filter that directly uses the measurement at time t to guide the proposal density selection. It is worth noting that success in dealing with data association problems has proven elusive to particle filters — this is perhaps despite the interpretation of measurement origin uncertainty as a form of non-Gaussian noise. Nonetheless, multiple model systems (of the sort for which the IMM would be appropriate) can be treated using the Rao-Blackwellization policy, which splits the inference task into parts that are "easy" to solve (like filtering with a known mode sequence) relegated to quick algorithms, with more difficult ones (like mode estimation) that are assigned to particles.

5 The Ensemble Kalman Filter

The EnKF is only really of interest in problems that are extremely large-scale in their state space, as happens in geophysics and meteorology, for example. If the state space has dimension of several thousand (or more!) we might decide to implement a Kalman filter in parallel Monte Carlo form. That is, many processors each are given responsibility for a few particles – and the total number of particles across all processors may be less than the state dimension. It is typical to specify

$$\begin{aligned} \text{dimension of observations } (m) &\ll \text{number of particles } (N) \\ &\ll \text{dimension of state } (n) \end{aligned} \tag{55}$$

for an EnKF to be useful.

Now, consider we have a collection of particles $\{\mathbf{x}^i_{t-1|t-1}\}_{i=1}^N$ that represent the state at time $t-1$, and we sample them forward[3] to $\{\mathbf{x}^i_{t|t-1}\}_{i=1}^N$ according to

$$\mathbf{x}^i_{t|t-1} \;=\; \mathbf{F}\mathbf{x}^i_{t-1|t-1} \;+\; \mathbf{v}^i_t \tag{56}$$

where $\mathbf{v}^i_t$ are $iid\ \mathcal{N}(0, \mathbf{Q})$. The natural particle-filter next step is to weight each particle by the observation likelihood

$$p(\mathbf{z}_t|\mathbf{x}^i_{t|t-1}) \;=\; \mathcal{N}(\mathbf{H}\mathbf{x}^i_{t|t-1}, \mathbf{R}) \tag{57}$$

As we have discussed earlier, this will require resampling in order to work well, and resampling is not well suited to parallel implementation.

[3]This is obvious: coast and add noise.

Another approach is to take the ensemble covariance

$$\hat{\mathbf{Q}} \;=\; \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}^i_{t|t-1})(\mathbf{x}^i_{t|t-1})^T \;-\; \left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^i_{t|t-1}\right)^2 \tag{58}$$

and thence compute the "optimal" posterior ensemble of particles

$$\mathbf{x}^i_{t|t} \;=\; \mathbf{x}^i_{t|t-1} \;+\; \hat{\mathbf{Q}}\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{Q}}\mathbf{H}^T+\mathbf{R}\right)^{-1}\left(\mathbf{z}_t-\mathbf{H}\mathbf{x}^i_{t|t-1}\right) \tag{59}$$

Unfortunately all the resulting particles are correlated, since all are the result of the same measurement noise that gave rise to $\mathbf{z}_t$. That is,

$$\mathcal{E}\left\{\left(\mathbf{z}_t-\mathbf{H}\mathbf{x}^i_{t|t-1}\right)\left(\mathbf{z}_t-\mathbf{H}\mathbf{x}^j_{t|t-1}\right)^T\right\} \tag{60}$$

$$= \;\mathcal{E}\left\{\left((\mathbf{z}_t-\mathbf{H}\mathbf{x}_t)-(\mathbf{H}\mathbf{x}^i_{t|t-1}-\mathbf{H}\mathbf{x}_t)\right)\right.$$
$$\left.\times\left((\mathbf{z}_t-\mathbf{H}\mathbf{x}_t)-(\mathbf{H}\mathbf{x}^j_{t|t-1}-\mathbf{H}\mathbf{x}_t)\right)^T\right\} \tag{61}$$

$$= \;\mathbf{R} \tag{62}$$

and this is just wrong.

The EnKF idea is to replace (59) by

$$\mathbf{x}^i_{t|t} \;=\; \mathbf{x}^i_{t|t-1} \;+\; \hat{\mathbf{Q}}\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{Q}}\mathbf{H}^T+\mathbf{R}\right)^{-1}\left(\mathbf{z}^i_t-\mathbf{H}\mathbf{x}^i_{t|t-1}\right) \tag{63}$$

where

$$\mathbf{z}^i_t \;=\; \mathbf{z}_t \;+\; \mathbf{w}^i_t \tag{64}$$

and $\mathbf{w}^i_t$ are *iid* $\mathcal{N}(0,\mathbf{R})$, meaning that we actually add noise[4] to the measurement before doing the Kalman update: each particle $\mathbf{x}^i_{t|t-1}$ is updated via its own noisy measurement $\mathbf{z}^i_t$. The *ensemble* posterior covariance is obviously correct, too.

The implementation of the EnKF uses, instead of (58),

$$\hat{\mathbf{Q}} \;=\; \frac{1}{N-1}\mathbf{A}_t\mathbf{A}_t^T \tag{65}$$

where

$$\mathbf{A}_t \;\equiv\; \begin{pmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ (\mathbf{x}^1_{t|t-1}-\bar{\mathbf{x}}_{t|t-1}) & (\mathbf{x}^2_{t|t-1}-\bar{\mathbf{x}}_{t|t-1}) & \cdots & (\mathbf{x}^N_{t|t-1}-\bar{\mathbf{x}}_{t|t-1}) \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix} \tag{66}$$

[4]This is the right thing to do. But I am still emotionally offended by the idea that the right thing to do is to add noise. It almost implies that the Kalman filter is not optimal.

and we also define

$$\mathbf{D} \equiv \begin{pmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \mathbf{z}_t^1 - \mathbf{z}_t & \mathbf{z}_t^2 - \mathbf{z}_t & \dots & \mathbf{z}_t^N - \mathbf{z}_t \\ \downarrow & \downarrow & \dots & \downarrow \end{pmatrix} \tag{67}$$

for the observation that has this artificial noise added. Note that the expected value of the particles has to be calculated, but this is simple since it is just the mean of the local means at each of the parallel processing units. Now (63) becomes

$$\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{A}_{t-1}(\mathbf{H}\mathbf{A}_{t-1})^T \left((\mathbf{H}\mathbf{A}_{t-1})(\mathbf{H}\mathbf{A}_{t-1})^T + \mathbf{R}\right)^{-1} (\mathbf{D} - (\mathbf{H}\mathbf{A}_{t-1})) \tag{68}$$

which implies that each particle can be updated in relative isolation, with the exception that the $m \times m$ covariance matrix must be calculated and inverted. Note that $(\mathbf{H}\mathbf{A}_{t-1})$ has dimension $m \times N$ – no need for a large matrix computation.