Inferring biological dynamics 101

3. Iterated filtering (IF)

- IF originated in 2006 [6].
- For plug-and-play likelihood-based inference on POMP models, there are not many alternatives. Directly estimating the likelihood surface, or applying generic optimization methods such as simulated annealing, are not applicable to larger models.
- Plug-and-play Bayesian alternatives are computationally problematic. PMCMC is extraordinarily computationally intensive. The Liu-West algorithm does not reliably cope with particle depletion.
- Some case studies are referenced at `wikipedia.org/wiki/Iterated_filtering`
- We're going to discuss pseudo-code [3, 7] for the implementation of iterated filtering in the R package pomp.

Supplementary Methods

Algorithm: maximum likelihood via iterated filtering

Model input:

process model $f(\cdot)$, measurement model $g(\cdot|\cdot)$, data $y_1, \ldots, y_N$, times $t_0, \ldots, t_N$

Algorithmic parameters:

number of particles J, fixed lag L, number of iterations M;
cooling factor $0 < a < 1$, $b > 0$; initial state vector $X_I^{(1)}$, initial parameter vector $\theta^{(1)}$;
variance-covariance matrices Σ_I, Σ_θ.

Procedure:

1. for $m = 1$ to M
2. draw $X_I(t_0, j) \sim \text{normal}(X_I^{(m)}, a^{m-1}\Sigma_I), \quad j = 1, \ldots, J$
3. set $X_F(t_0, j) = X_I(t_0, j)$
4. draw $\theta(t_0, j) \sim \text{normal}(\theta^{(m)}, ba^{m-1}\Sigma_\theta)$
5. set $\bar{\theta}(t_0) = \theta^{(m)}$
6. for $n = 1$ to N
7. set $X_P(t_n, j) = f(X_F(t_{n-1}, j), t_{n-1}, t_n, \theta(t_{n-1}, j), W)$
8. set $w(n, j) = g(y_n | X_P(t_n, j), t_n, \theta(t_{n-1}, j))$
9. draw $k_1, \ldots, k_J$ such that $\text{Prob}[k_j = i] = w(n, i) / \sum_l w(n, l)$
10. set $X_F(t_n, j) = X_P(t_n, k_j)$
11. set $X_I(t_n, j) = X_I(t_{n-1}, k_j)$
12. draw $\theta(t_n, j) \sim \text{normal}(\theta(t_{n-1}, k_j), a^{m-1}(t_n - t_{n-1})\Sigma_\theta)$
13. set $\bar{\theta}_i(t_n)$ to be the sample mean of $\{\theta_i(t_{n-1}, k_j), j = 1, \ldots, J\}$
14. set $V_i(t_n)$ to be the sample variance of $\{\theta_i(t_n, j), j = 1, \ldots, J\}$
15. end for
16. $\theta_i^{(m+1)} = \theta_i^{(m)} + V_i(t_1) \sum_{n=1}^{N} V_i^{-1}(t_n)(\bar{\theta}_i(t_n) - \bar{\theta}_i(t_{n-1}))$
17. set $X_I^{(m+1)}$ to be the sample mean of $\{X_I(t_L, j), j = 1, \ldots, J\}$
18. end for

Return:

maximum likelihood estimate for parameters, $\hat{\theta} = \theta^{(M+1)}$;
maximum likelihood estimate for initial values, $\hat{X}(t_0) = X_I^{(M+1)}$;
maximized log likelihood estimate, $\log \mathcal{L}(\hat{\theta}) = \sum_n \log(\sum_j w(n, j)/J)$

Note:

Here, $\text{normal}(\mu, \Sigma)$ denotes a multivariate normal random variable with mean vector μ and covariance matrix Σ. $X(t_n)$ takes values in $\mathbb{R}^{d_x}$, y_n takes values in $\mathbb{R}^{d_y}$, θ takes values in $\mathbb{R}^{d_\theta}$ and has components $\{\theta_i, i = 1, \ldots, d_\theta\}$. The computationally challenging steps (6–15) correspond to a standard implementation of particle filtering[8]; many refinements are possible within the context of this algorithm. The update equation (16) is the main innovation of Ionides *et al.*[9].

- **The name(s)**: "Iterated filtering" and "maximum likelihood via iterated filtering" and "mif" are synonyms.

- **Process model**. $f(\cdot)$ generates numerical solutions to the Markov transition model, so

$$X(t_n) = f(X(t_{n-1}), t_{n-1}, t_n, \theta, W).$$

  ◊ Here, W is random quantity, drawn independently each time $f(\cdot)$ is evaluated.

  ◊ Any algorithm which inputs the model in this black box form has the plug-and-play property.

- **Measurement model**. $g(\cdot \,|\, \cdot)$ is $f_{Y_n|X_n}(y_n \,|\, x_n, t_n, \theta)$. We require that this conditional density can be evaluated.

- **Initial time** t_0. This is the time at which initial state variables are defined. In principle, initial state variables could be defined at t_1. Setting $t_0 < t_1$ gives the dynamic process some time to disperse though the state space before the first observation.

- **Number of particles,** J. Often, $J = 1000$ is about right. If the effective sample size drops below about 100, that can be problematic.

- **Fixed lag,** L. This is used for estimating initial value parameters (IVPs). It should correspond to the mixing time of the system. After time t_L, it is assumed there is negligible additional information about $X(t_0)$.

- **Number of filtering iterations,** M. Usually, $M = 50$ or $M = 100$ is enough. By contrast, for PMCMC one might use $M = 50000$ [1]. The difference arises since IF investigates the likelihood surface within each filter iteration, whereas PMCMC (and other black box methods such as simulated annealing) use each filter iteration only for its likelihood at a fixed parameter value.

- **cooling factor,** $0 < \alpha < 1$. This gives fraction by which temperature (i.e., the random walk intensity of the parameters) is reduced at each iteration. $\alpha = 0.95$ and $\alpha = 0.98$ are standard. Theoretically, this geometric cooling is too quick to guarantee asymptotic convergence [5]. In practice, multiple re-starts with geometric is often preferred [4].

- **initial variance multiplier,** $b > 0$. At the start of each iteration, the parameter space must be re-populated using the current parameter point estimate. The initial diversity is equivalent to b steps of the random walk. In pomp, the `mif()` argument `var.factor` happens to multiply the standard deviation rather than the variance (i.e., it matches $\sqrt{b}$).

  ◊ Values `var.factor` $= 2$ and `var.factor` $= 4$ are usual.

  ◊ A larger value of b leads to more ambitious jumps in the parameter update of step 16.

- **initial state vector,** $X_I^{(1)}$. Here, we should say "starting initial state vector." For better or worse, we are not describing the initial state vector as part of the parameter space. We are also assuming an identity map from the IVPs to the initial states.

- **initial parameter vector,** $\theta^{(1)}$. Here, we should say "starting parameter vector."

- **variance-covariance matrix,** Σ_θ. The random walk variance for each time step when $m = 1$. In `mif()`, Σ_θ is assumed diagonal, with entries specified by the `rw.sd` argument.

  ♢ It is helpful to transform all parameters so their uncertainty is on a unit scale (e.g., log and logistic transformations where appropriate). Then, using a common value such as `rw.sd` $= 0.02$ works surprisingly often.

- **variance-covariance matrix,** Σ_I. Similar to Σ_θ, but the noise only gets added once per iteration.

- **step 1 (outer loop)**. For each value of m, an entire filtering operation and parameter update is carried out.
- **steps 2 and 4 (parameter initialization)**. The choice of the normal distribution is convenient but plays no theoretical role. Any distribution with the given mean and variance would suffice.
- **step 3 (state initialization)**. The filter distribution at time t_0 is defined to be the unconditional distribution of $X(t_0)$.
- **step 5**. The filter mean of $\theta(t_0)$ is defined to be the unconditional mean.
- **step 6 (inner loop)**. This is the vanilla SMC recursion, with the state process augmented by a parameter vector undergoing a random walk.
- **step 7 (particle updating)**. Constructing the prediction distribution at time t_n. It is implicit here that this is carried out for each particle $j = 1, \ldots, J$.

- **step 8 (filtering weight computation)**.
- **steps 9, 10, 11 (resample states)**. Conceptually, one can think of multinomial resampling with weights $w(n,j)$. There are alternatives which have the same marginal probabilities, and hence the correct theoretical properties, but reduced Monte Carlo variance. `mif()` uses a systematic resampling method [2].
- **step 12 (resample and move parameters)**.
- **steps 13, 14**. Compute the filtering mean and prediction variance for use in step 16. Here, θ_i is the ith component of θ, and the operation is implicitly carried out for each value of i.
- **step 16 (parameter update)**. The quantity

$$\sum_{n=1}^{N} V_i^{-1}(t_n)(\bar{\theta}_i(t_n) - \bar{\theta}_i(t_{n-1}))$$

is an approximation to the derivative of the log likelihood in a neighborhood of $\theta^{(m)}$. The premultiplier $V_i(t_1)$ is chosen to have an appropriate scale, but is somewhat arbitrary.

- **MLE,** $\hat{\theta} = \theta^{(M+1)}$. This is an approximation to the MLE: more correctly, $\hat{\theta} \approx \theta^{(M+1)}$. Simulation studies, multiple starting values, and Monte Carlo replications are needed to validate the estimator on non-trivial problems.
- **MLE,** $\hat{X}(t_0) = X_I^{(M+1)}$. Estimating initial values is not a strength of IF. Additional attention to initial values, such as addition iterations filtering only to time t_L, may be warranted.

- **Estimate** $\ell(\hat{\theta}) \approx \sum_{n=1}^{N} \log(\sum_j w(n,j)/J)$.
This estimate of the maximized log likelihood is a free byproduct of IF. Weaknesses are:
(i) since the added noise is not quite zero, even on the last iteration, this is the likelihood of a perturbed model not the desired model;
(ii) fewer particles are needed to run IF successfully compared to accurate likelihood evaluation (this is because IF borrows strength between iterations).

  ◊ Additional SMC runs (`pfilter`) should be carried out at interesting sets of parameters.

  ◊ Monte Carlo error on likelihood evaluation can be relatively easily measured and reduced. It helps scientific progress to separate likelihood evaluation error from maximization error.

  ◊ Monte Carlo error in likelihood estimation of $\ll 1$ log unit is negligible. Replication is needed to assess error, but increasing J is often more efficient than averaging over replications.

References

[1] Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 72:269–342.

[2] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear, non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174 – 188.

[3] Bretó, C., He, D., Ionides, E. L., and King, A. A. (2009). Time series analysis via mechanistic models. *Annals of Applied Statistics*, 3:319–348.

[4] Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18:29–57.

[5] Ionides, E. L., Bhadra, A., Atchadé, Y., and King, A. A. (2011). Iterated filtering. *Annals of Statistics*, 39:1776–1802.

[6] Ionides, E. L., Bretó, C., and King, A. A. (2006). Inference for nonlinear dynamical systems. *Pro-*

ceedings of the National Academy of Sciences of the USA, 103:18438–18443.

[7] King, A. A., Ionides, E. L., Pascual, M., and Bouma, M. J. (2008). Inapparent infections and cholera dynamics. *Nature*, 454:877–880.