Estimating Parameters and Hidden States in Biological Networks with Particle Filters

Nicolas Brunel, Minh Quach and Florence d'Alché-Buc

IBISC FRE CNRS 2873
University of Evry and Genopole, France

4th September 07 / PMNP workshop
Outline

1. Problem
2. Filtering in State Space Models
3. Particle Filter
4. Results
Outline

1. Problem
2. Filtering in State Space Models
3. Particle Filter
4. Results
Problem: Reverse Engineering of Biological Networks

ODE Model

\[
\begin{align*}
\frac{dx(t)}{dt} &= f(x(t); \theta) \\
y(t) &= H(x(t); \theta) + \epsilon(t)
\end{align*}
\]

- \(x(t)\): state variables at time \(t\) (protein, mRNA, metabolite concentrations)
- \(f\): nonlinear function, derived from biochemical reactions.
- \(\theta\): parameter set (kinetic parameters, rate constants,...)
- \(H\) is a nonlinear observation function
- \(\epsilon(t)\) is a i.i.d measurement noise

Problem

- Given: A sequence of observed data: \(y_{1:K} = \{y_1, \ldots, y_K\}\) at time \(t_1, t_2, \ldots, t_K\)
- Goal: Estimation of parameters \(\theta\) and states \(x(t)\)
Outline

1. Problem
2. Filtering in State Space Models
3. Particle Filter
4. Results
Nonlinear State-Space Model

Continuous time ODE model

\[
\frac{dx(t)}{dt} = f(x(t); \theta) \\
y(t) = H(x(t); \theta) + \epsilon(t)
\]
Nonlinear State-Space Model

Continuous time ODE model

\[
\begin{align*}
\frac{dx(t)}{dt} &= f(x(t); \theta) \\
y(t) &= H(x(t)); \theta + \epsilon(t)
\end{align*}
\]

The corresponding discrete-time augmented state-space model

- Augmented states:
  \[
  \begin{align*}
  \theta_{k+1} &= \theta_k \\
x(t_{k+1}) &= F(x(t_k); \theta_k)
  \end{align*}
  \]
  with
  \[
  F(x(t_k); \theta) = x(t_k) + \int_{t_k}^{t_{k+1}} f(x(\tau); \theta) d\tau
  \]
- Observation model:
  \[
  y(t_k) = H(x(t_k); \theta_k) + \epsilon(t_k)
  \]
Bayesian estimation: Filtering

Given:
- Prior distribution over the initial state and parameters: \( p(x_1, \theta_1) \)
- A state transition model: \( p(x_k|x_{k-1}, \theta_{k-1}) \)
- An observation model: \( p(y_k|x_k, \theta_k) \)
- A sequence of observations: \( y_{1:K} = \{y_1, \ldots, y_K\} \)

Estimating the posterior distributions

\[ p(x_k, \theta_k | y_{1:k}) \quad k = 1, 2, \ldots, K \]
Recursive Filtering Algorithm

Two steps

1. Prediction: \( p(x_{k+1} | y_{1:k}) = \int p(x_{k+1} | x_k) p(x_k | y_{1:k}) \, dx_k \)
Recursive Filtering Algorithm

Two steps

1. Prediction: \( p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k})dx_k \)
2. Update:

\[
p(x_{k+1}|y_{1:k+1}) = \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|y_{1:k})}{p(y_{k+1}|y_{1:k})}
\]

where:

\[
p(y_{k+1}|y_{1:k}) = \int p(y_{k+1}|x_{k+1})p(x_{k+1}|y_{1:k})dx_{k+1}
\]
Recursive Filtering Algorithm

Two steps

1. **Prediction:**
   \[ p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k}) \, dx_k \]

2. **Update:**
   \[ p(x_{k+1}|y_{1:k+1}) = \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|y_{1:k})}{p(y_{k+1}|y_{1:k})} \]
   where:
   \[ p(y_{k+1}|y_{1:k}) = \int p(y_{k+1}|x_{k+1})p(x_{k+1}|y_{1:k}) \, dx_{k+1} \]

- **Analytical solution obtained only when** \( F, H \) **are linear and** \( p(x_1) \) **and** \( \epsilon \) **are Gaussian \( \rightarrow \) Kalman Filter**

- **When** \( F, H \) **are nonlinear, the integrals are usually intractable \( \rightarrow \) Approximate solutions**
  - Gaussian Approximations: Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF).
  - **Sequential Monte Carlo Methods or Particle filters** [Gordon 1993, Doucet 1998]
Outline

1. Problem
2. Filtering in State Space Models
3. Particle Filter
4. Results
Sequential Monte Carlo Methods or Particle filters

- Map intractable integrals of optimal Bayesian solution to tractable discrete sums of samples drawn from the posterior distribution.

\[ p(x_k | y_{1:k}) \]

\[ x_k^{(i)} \overset{i.i.d.}{\sim} p(x_k | y_{1:k}) \]

\[ \hat{p}(x_k | y_{1:k}) = \frac{1}{N} \sum_{i=1}^{N} \delta(x_k - x_k^{(i)}) \]

- So, any expectation of the form

\[ E[g(x_k)] = \int g(x_k)p(x_k | y_{1:k})dx_k \]

may be approximated by:

\[ E[g(x_k)] \approx \frac{1}{N} \sum_{i=1}^{N} g(x_k^{(i)}) \]
Importance sampling

- Often impossible to sample directly from true posterior density \( p(x_k|y_{1:k}) \).
- Use importance sampling:

\[
\int g(x_k)p(x_k|y_{1:k})dx_k = \int g(x_k) \frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})} q(x_k|y_{1:k})dx_k \approx \sum_i w^{(i)}_k g(x^{(i)}_k)
\]

where \( x^{(i)}_k \) is sampled from a proposal distribution \( q(x_k|y_{1:k}) \):

\[
x^{(i)}_k \sim q(x_k|y_{1:k})
\]

and the importance weights given by:

\[
w^{(i)}_k = \frac{p(x^{(i)}_k|y_{1:k})}{q(x^{(i)}_k|y_{1:k})}
\]
Sequential Importance Sampling with Resampling

- Recursive implementation of sequential importance sampling.

\[ w_k^{(i)} = w_{k-1}^{(i)} \frac{p(y_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)})}{q(x_k^{(i)} | x_{k-1}^{(i)}, y_1:k)} \]

- Most popular choice of proposal distribution is the transition prior:

\[ q(x_k | x_{k-1}, y_{1:k}) = p(x_k | x_{k-1}) \]

- So,

\[ w_k^{(i)} = w_{k-1}^{(i)} p(y_k | x_k^{(i)}) \]

- Particles degenerate over time (only few particles have significant weights)

- Sampling-importance Resampling (SIR): keep / multiply particles with high importance weights and map N unequally weighted particles into a new set of N equally weighted samples.

\[ \{x_k^{(i)}, w_k^{(i)}\} \longrightarrow \{x_k^{(i)}, \frac{1}{N}\} \]
Sampling-importance Resampling: Bootstrap Filter [Gordon 1993]

1. **Discrete Monte Carlo representation of** $p(x_{k-1}|y_{1:k-1})$.
2. **Set of N particles**: $x_{k-1}^{(i)}$.
3. $x_k^{(i)} = F(x_{k-1}^{(i)})$.
4. **Draw new particles from proposal distribution** $p(x_k^{(i)}|x_{k-1}^{(i)})$.
5. **Given new observation** $y_k$.
6. **Evaluate importance weights using likelihood function** $w_k^{(i)} \propto p(y_k|x_k^{(i)})$.
7. **Resample particles**.
8. **Discrete Monte Carlo representation (approximation) of** $p(x_k|y_{1:k})$. 

[van der Merwe 01]
Theoretical Convergence

**Theorem**

If the importance weight

$$w_k \propto \frac{p(y_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{k-1}, y_{1:k})}$$

is upper bounded for any \((x_{k-1}, y_{1:k})\) then, for all \(k \geq 0\), there exists \(c_k\) independent of \(N\) such that for any \(f_k\)

$$E \left[ \left( \frac{1}{N} \sum_{i=1}^{N} f_k(x_{1:k}^{(i)}) - \int f_k(x_{1:k})p(x_{1:k}|y_{1:k}) \right)^2 \right] \leq c_k \frac{\|f_k\|^2}{N}$$
Problem Filtering in State Space Models

Particle Filter Results

Problem with Bootstrap Filter

- Deterministic parameter and state evolution:
  \[ \theta_{k+1} = \theta_k \quad x(t_{k+1}) = F(x(t_k), u; \theta_k) \rightarrow p(x_{k+1}, \theta_{k+1}|x_k, \theta_k) \text{ is a Dirac-Delta function.} \]

- Parameter and state particles degenerate quickly after a few time steps
Problem with Bootstrap Filter

- Deterministic parameter and state evolution:
  \[ \theta_{k+1} = \theta_k \quad x(t_{k+1}) = F(x(t_k), u; \theta_k) \rightarrow p(x_{k+1}, \theta_{k+1}|x_k, \theta_k) \text{ is a Dirac-Delta function.} \]

- Parameter and state particles degenerate quickly after a few time steps → At each time step, a small amount of Gaussian noise is added to each resampled particle, which is equivalent to using a Gaussian kernel to smooth the filtering distribution [Liu and West 2001]

\[
p(\theta_k|y_{1:k}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(\theta_k - \theta_k^{(i)}) \text{ with mean } \bar{\theta}_k \text{ and variance } V_k \text{ is replaced by:}
\]

\[
p(\theta_k|y_{1:k}) \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(\theta_k|m_k^{(i)}, h^2 V_k)
\]

- Shrinkage of kernel locations to retain the mean \( \bar{\theta}_k \) and variance \( V_k \) (to prevent the lost of information caused by adding artificial Gaussian noise):

\[
m_k^{(i)} = a\theta_k^{(i)} + (1 - a)\bar{\theta}_k
\]

where \( a = \sqrt{1 - h^2} \)
Repressilator

[Elowitz, Nature 2000]

mRNAs are observed, proteins are hidden
mRNA and protein degradation rate constants are supposed to be known
Estimate 9 parameters

\[
\begin{align*}
\frac{dr_1}{dt} &= v_1^{\text{max}} \frac{k_{12}^n}{k_{12}^n + p_2^n} - k_1^{\text{mRNA}} r_1 \\
\frac{dr_2}{dt} &= v_2^{\text{max}} \frac{k_{23}^n}{k_{23}^n + p_3^n} - k_2^{\text{mRNA}} r_2 \\
\frac{dr_3}{dt} &= v_3^{\text{max}} \frac{k_{31}^n}{k_{31}^n + p_1^n} - k_3^{\text{mRNA}} r_3 \\
\frac{dp_1}{dt} &= k_1 r_1 - k_1^{\text{protein}} p_1 \\
\frac{dp_2}{dt} &= k_2 r_2 - k_2^{\text{protein}} p_2 \\
\frac{dp_3}{dt} &= k_3 r_3 - k_3^{\text{protein}} p_3
\end{align*}
\]
Result for the Repressilator

20% Gaussian noise, Gaussian priors on parameters and initial states, 5000 particles.
Figure 2. Estimation of protein concentrations with PF. The evolution of the true (dashed) and estimated (solid) protein concentrations.
Table 1. Root mean squared errors of UKF and PF

<table>
<thead>
<tr>
<th></th>
<th>RMSE: mean and variance</th>
<th>Execution time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UKF</td>
<td>3.37 ± 0.29</td>
<td>4.25</td>
</tr>
<tr>
<td>PF</td>
<td>2.99 ± 0.29</td>
<td>2273.94</td>
</tr>
</tbody>
</table>
TABLE 2. Parameter estimation results

<table>
<thead>
<tr>
<th>Parameter</th>
<th>True value</th>
<th>UKF</th>
<th>PF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1^{max}$</td>
<td>150</td>
<td>150.06 ± 2.16</td>
<td>149.69 ± 2.65</td>
</tr>
<tr>
<td>$v_2^{max}$</td>
<td>80</td>
<td>79.73 ± 2.35</td>
<td>79.45 ± 2.98</td>
</tr>
<tr>
<td>$v_3^{max}$</td>
<td>100</td>
<td>98.96 ± 3.85</td>
<td>99.95 ± 4.60</td>
</tr>
<tr>
<td>$k_{12}$</td>
<td>50</td>
<td>49.75 ± 1.49</td>
<td>49.90 ± 1.71</td>
</tr>
<tr>
<td>$k_{23}$</td>
<td>30</td>
<td>30.50 ± 1.25</td>
<td>30.42 ± 1.79</td>
</tr>
<tr>
<td>$k_{31}$</td>
<td>40</td>
<td>40.39 ± 1.37</td>
<td>40.05 ± 1.71</td>
</tr>
</tbody>
</table>
Conclusion and Future work

- Choices of better proposal distributions (e.g. EKF, UKF proposals)
- Adaptive resampling schemes
- Larger networks