Lesson 5: Iterated filtering: Theory

Spencer J. Fox Qianying (Ruby) Lin Jesse Wheeler

Introduction

- This tutorial covers likelihood estimation via the method of iterated filtering.
- It presupposes familiarity with building partially observed Markov process (POMP) objects in the `R` package pomp (King, Nguyen, and Ionides 2016).
- This tutorial follows on from the topic of particle filtering (also known as sequential Monte Carlo) via `pfilter` in pomp.

Objectives

1. To review the available options for inference on POMP models, to put iterated filtering in context.
2. To understand how iterated filtering algorithms carry out repeated particle filtering operations, with randomly perturbed parameter values, in order to maximize the likelihood.
3. To gain experience carrying out statistical investigations using iterated filtering in a relatively simple situation: fitting an SIR model to data from a measles outbreak.

Classification of statistical methods for POMP models I

- Many, many statistical methods have been proposed for inference on POMP models King, Nguyen, and Ionides (2016).
- The volume of research indicates both the importance and the difficulty of the problem.
- Let's start by considering three criteria to categorize inference methods:
  - the plug-and-play property
  - full-information or feature-based
  - frequentist or Bayesian

Plug-and-play (also called simulation-based) methods I

- Inference methodology that calls `rprocess` but not `dprocess` is said to be *plug-and-play*. All popular modern Monte Carlo methods for POMP models are in this category.
- "Simulation-based" is equivalent to "plug-and-play".
- Historically, simulation-based meant simulating forward from initial conditions to the end of the time series.
- However, particle filtering methods instead consider each observation interval sequentially. They carry out multiple, carefully selected, simulations over each interval.

Plug-and-play (also called simulation-based) methods II

- Plug-and-play methods can call `dmeasure`. A method that uses only `rprocess` and `rmeasure` is called "doubly plug-and-play".
- Two *non-plug-and-play* methods—expectation-maximization (EM) and Markov chain Monte Carlo (MCMC)—have theoretical convergence problems for nonlinear POMP models. The failures of these two workhorses of statistical computation have prompted development of alternative methodologies.

Full-information and feature-based methods I

- *Full-information* methods are defined to be those based on the likelihood function for the full data (i.e., likelihood-based frequentist inference and Bayesian inference).
- *Feature-based* methods either consider a summary statistic (a function of the data) or work with an an alternative to the likelihood.
- Asymptotically, full-information methods are statistically efficient and feature-based methods are not.
- In some cases, loss of statistical efficiency might be an acceptable tradeoff for advantages in computational efficiency.

Full-information and feature-based methods II

- However:
  - Good low-dimensional summary statistics can be hard to find.
  - When using statistically inefficient methods, it can be hard to know how much information you are losing.
  - Intuition and scientific reasoning can be inadequate tools to derive informative low-dimensional summary statistics (Shrestha, King, and Rohani 2011, Ionides2011a).

Bayesian and frequentist methods I

- Recently, plug-and-play Bayesian methods have been discovered:
  - particle Markov chain Monte Carlo (PMCMC) (Andrieu, Doucet, and Holenstein 2010).
  - approximate Bayesian computation (ABC) (Toni et al. 2009).
- Prior belief specification is both the strength and weakness of Bayesian methodology:
- The likelihood surface for nonlinear POMP models often contains nonlinear ridges and variations in curvature.

Bayesian and frequentist methods II

- These situations bring into question the appropriateness of independent priors derived from expert opinion on marginal distributions of parameters.
- They also are problematic for specification of "flat" or "uninformative" prior beliefs.
- Expert opinion can be treated as data for non-Bayesian analysis. However, our primary task is to identify the information in the data under investigation, so it can be helpful to use methods that do not force us to make our conclusions dependent on quantification of prior beliefs.

Summary

	Frequentist	Bayesian
Plug-and-play		
Full-information	iterated filtering	particle MCMC
Feature-based	simulated moments synthetic likelihood (SL) nonlinear forecasting	ABC SL-based MCMC
Not plug-and-play		
Full-information	EM algorithm Kalman filter	MCMC
Feature-based	Yule-Walker[1] extended Kalman filter[2]	extended Kalman filter[2]

[1]Yule-Walker is a method of moments for ARMA, a linear Gaussian POMP.

[2]The Kalman filter gives the exact likelihood for a linear Gaussian POMP. The extended Kalman filter gives an approximation for nonlinear models that can be used for quasi-likelihood or quasi-Bayesian inference.

Full-information, plug-and-play, frequentist methods

- Iterated filtering methods (Ionides et al. 2015) are the only currently available, full-information, plug-and-play, frequentist methods for POMP models.
- Iterated filtering methods have been shown to solve likelihood-based inference problems for epidemiological situations which are computationally intractable for available Bayesian methodology (Ionides et al. 2015).

An iterated filtering algorithm (IF2) I

Input:

- simulators for $f_{X_0}(x_0;\theta)$ and $f_{X_n|X_{n-1}}(x_n|x_{n-1};\theta)$
- evaluator for $f_{Y_n|X_n}(y_n|x_n;\theta)$
- data, $y^*_{1:N}$

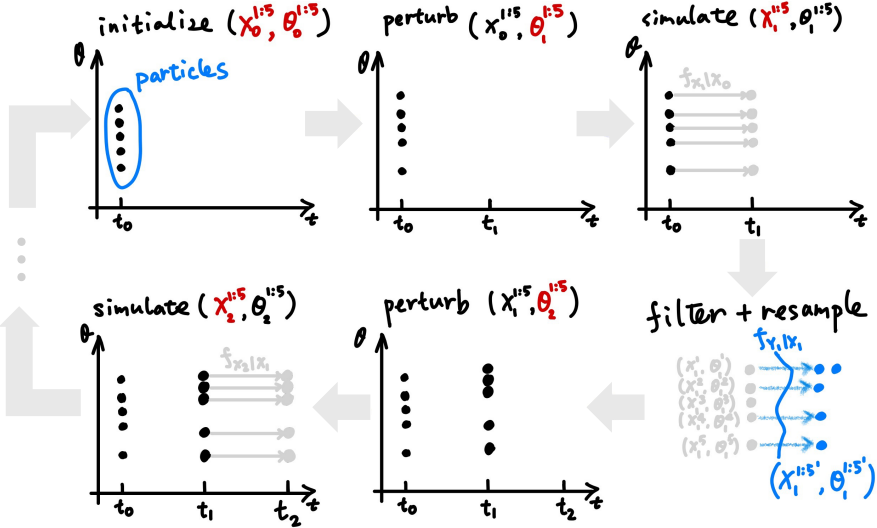
Algorithmic parameters and corresponding `mif2` arguments:

- number of iterations, `Nmif` $= M$
- number of particles, `Np` $= J$
- initial parameter swarm
- perturbation, random walk standard deviation for each parameter, `rw.sd`, squared to construct a diagonal variance matrix, V_n
- decay of perturbation, cooling fraction in 50 iterations, `cooling.fraction.50` $= a$

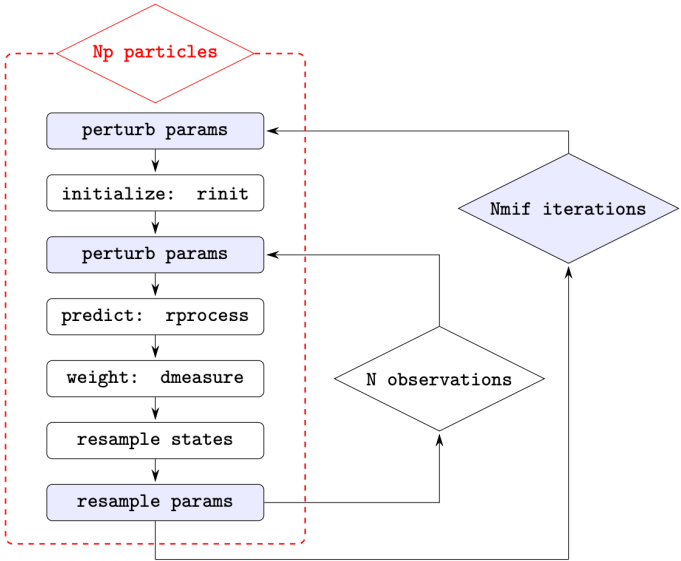
Output:

- final parameter swarm

An iterated filtering algorithm (IF2) II



The IF2 diagram in pomp



Exercise 1: IF2 in pomp, mif2 basics I

The following codes (also see /scripts/exercise_mif_*.R for your OS) run ONE iterated filtering with 50 iterations and show how parameters change over iteration. Change some numbers to see how the results are impacted by those changes.

```
measSIR |>
  mif2(
    Np=2000, Nmif=50, cooling.fraction.50=0.5,
    rw.sd=rw_sd(Beta=0.02, Rho=0.02, Eta=ivp(0.02)),
    partrans=parameter_trans(log="Beta",logit=c("Rho","Eta")),
    paramnames=c("Beta","Rho","Eta")
  ) -> mf

plot(mf)
```

Exercise 1: IF2 in pomp, mif2 basics II

We can also recycle the results in the last iteration within `mf` and run another 50 iterations with different setting. Update other arguments to see how it change the results.

```
mf |>
  mif2(
    cooling.fraction.50 = 0.3
  ) -> mf

plot(mf)
```

Exercise 2: `mif2` in parallel I

The following codes run `mif2` for 20 replicates within a parallel computing scheme.

```
library(foreach)
library(doParallel)
library(doRNG)
registerDoParallel(cores=8)
foreach(
  i=1:20,.combine=c, .packages="pomp", .options.RNG=482947940
) %dorng% {
  measSIR |>
    mif2(Np=2000, Nmif=50, cooling.fraction.50=0.5,
      rw.sd=rw_sd(Beta=0.02, Rho=0.02, Eta=ivp(0.02)),
      partrans=parameter_trans(log="Beta",logit=c("Rho","Eta")),
      paramnames=c("Beta","Rho","Eta"))
} -> mifs_local
```

Exercise 2: `mif2` in parallel II

We can track the how parameters change over iteration of each replicate in one plot. Discuss how those 20 replicates change and what the takeaway is.

```
mifs_local |>
  traces() |>
  melt() |>
  ggplot(aes(x=iteration,y=value,group=.L1,color=factor(.L1))) +
    geom_line()+
    guides(color="none")+
    facet_wrap(~name,scales="free_y")
```

Exercise 3: likelihood estimation for `mif2` objects

Why we need to estimate the likelihood in a different manner?

```
foreach(
  mf=mifs_local,.combine=rbind, .packages = "pomp",
  .options.RNG=900242057
) %dorng% {
  evals <- replicate(10, logLik(pfilter(mf,Np=5000)))
  ll <- logmeanexp(evals,se=TRUE)
  mf |> coef() |> bind_rows() |>
  bind_cols(loglik=ll[1],loglik.se=ll[2])
} -> results

results |> filter(loglik==max(loglik))
```

References I

Andrieu, Christophe, Arnaud Doucet, and Roman Holenstein. 2010. "Particle Markov Chain Monte Carlo Methods." *J R Stat Soc B* 72 (3): 269–342. https://doi.org/10.1111/j.1467-9868.2009.00736.x.

He, Daihai, Edward L. Ionides, and Aaron A. King. 2010. "Plug-and-Play Inference for Disease Dynamics: Measles in Large and Small Populations as a Case Study." *J R Soc Interface* 7 (June): 271–83. https://doi.org/10.1098/rsif.2009.0151.

Ionides, Edward L., Dao Nguyen, Yves Atchadé, Stilian Stoev, and Aaron A. King. 2015. "Inference for Dynamic and Latent Variable Models via Iterated, Perturbed Bayes Maps." *Proc Natl Acad Sci* 112 (3): 719–24. https://doi.org/10.1073/pnas.1410597112.

King, Aaron A., Dao Nguyen, and Edward L. Ionides. 2016. "Statistical Inference for Partially Observed Markov Processes via the R Package Pomp." *J Stat Softw* 69 (12): 1–43. https://doi.org/10.18637/jss.v069.i12.

References II

Shrestha, Sourya, Aaron A. King, and Pejman Rohani. 2011. "Statistical Inference for Multi-Pathogen Systems." *PLoS Comput Biol* 7 (8): e1002135. https://doi.org/10.1371/journal.pcbi.1002135.

Toni, Tina, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P. H. Stumpf. 2009. "Approximate Bayesian Computation Scheme for Parameter Inference and Model Selection in Dynamical Systems." *J R Soc Interface* 6 (31): 187–202. https://doi.org/10.1098/rsif.2008.0172.

License, acknowledgments, and links

- This lesson is prepared for the Simulation-based Inference for Epidemiological Dynamics module at the Summer Institute in Statistics and Modeling in Infectious Diseases, SISMID.
- The materials build on previous versions of this course and related courses.
- Licensed under the Creative Commons Attribution-NonCommercial license. Please share and remix non-commercially, mentioning its origin. 
- Produced with R version 4.4.2 and pomp version 6.3.
- Compiled on 2025-07-21.

Back to Lesson
R code for this lesson