Modern Computational Statistics: Alternatives to MCMC

Paul Fearnhead

Outline of Course

- Discrete-State HMMs: Forward-Backward Algorithm
- General HMMs: Sequential Monte Carlo (SMC)
- Estimating Parameters within SMC
- New MCMC: Pseudo-Marginal and Particle MCMC
- Approximate Bayesian Computation

Discrete-state Hidden Markov Models

Discrete-state HMMs

We define a Hidden Markov Model (HMM):

(i) a hidden (latent) state, X_t, which is a Markov chain:

$$p(x_t|\mathbf{x}_{1:t-1}) = p(x_t|x_{t-1}),$$

where $\mathbf{x}_{1:t-1} = \{x_1, \ldots, x_{t-1}\}$.

(ii) a series of observations, Y_t, where Y_t is conditionally independent of X_s and Y_s for $s \neq t$ given X_t.

We can write down the following joint probability:

$$p(x_{1:t}, y_{1:t}) = p(x_1) \left(\prod_{i=2}^{t} p(x_i|x_{i-1}) \right) \left(\prod_{i=1}^{t} p(y_i|x_i) \right).$$

Discrete-state HMMs

Initially we will consider models where X_t has a finite number of states.

We will consider how to answer the following

(1) Can we calculate the likelihood: $p(\mathbf{y}_{1:t})$?

(2) Can we simulate from the conditioned process: $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$?

(3) Can we calculate the most likely values of $\mathbf{X}_{1:t}$ given $\mathbf{y}_{1:t}$?

A Simple Example

Assume you have n biased coins. Coin i has probability p_i of landing heads up. Each coin toss is independent of all others.

How would you:

(i) Calculate the probability of observing y heads if you tossed all n coins?

(ii) Simulate from the conditional distribution of which coins landed heads up if you observed y heads?

[Here it is trivial to calulate the most likely realisation of heads/tails.]

Recursions

We can derive general recursions for more general HMMs.

These come by many names, including Forward-Backward recursions/algorithm. The latter comes from the fact these are described by:

- a set of forward recursions that calculate the likelihood and filtering distributions; and
- a backward simulation step to sample from the distribution of the conditioned state-process.

Summary of Recursions

$$p(y_1) = \sum_i \Pr(X_1 = i) p(y_1 | X_1 = i) \qquad \Pr(X_1 = i | y_1) = \frac{p(y_1 | X_1 = i)}{p(y_1)} \Pr(X_1 = i)$$

Forward Recursions (for $t \geq 2$).

- Likelihood:

$$p(y_t | \mathbf{y}_{1:t-1}) = \sum_i \left(p(y_t | X_t = i) \sum_j \Pr(X_{t-1} = j | \mathbf{y}_{1:t-1}) \Pr(X_t = i | X_{t-1} = j) \right)$$

- Filtering distribution:

$$\Pr(X_t = i | \mathbf{y}_{1:t}) = \frac{p(y_t | X_t = i)}{p(y_t | \mathbf{y}_{1:t-1})}) \sum_j \Pr(X_{t-1} = j | \mathbf{y}_{1:t-1}) \Pr(X_t = i | X_{t-1} = j)$$

Summary of Recursions

Backward Simulation

- Simulate X_n from $\Pr(X_n|y_{1:n})$;
- For $t = n-1, \ldots, 1$:
  - Simulate X_t given x_{t+1} from

$$\begin{aligned}\Pr(X_t = i|\mathbf{y}_{1:n}, x_{t+1}) &= \Pr(X_t = i|\mathbf{y}_{1:t}, x_{t+1}) \\ &\propto \Pr(X_t = i|\mathbf{y}_{1:t}) \Pr(X_{t+1} = x_{t+1}|X_t = i)\end{aligned}$$

Most Likely State Path: Viterbi Algorithm

We can also use the Markov property to calculate $\hat{x}_{1:t}$, that maximises $p(x_{1:t}, y_{1:t})$.

Define

$$Q_i(x) = \max_{x_{1:i-1}} \log p(x_{1:i}, y_{1:i}).$$

Then

$$Q_1(x) = \log \Pr(X_1 = x) + \log p(y_1 | X_1 = x),$$

and for $i = 2, \ldots, t$

$$Q_i(x) = \max_{x'} \left\{ Q_{i-1}(x') + \log \Pr(X_i = x | X_{i-1} = x') + \log p(y_i | X_i = x) \right\}.$$

Most Likely State Path: Viterbi Algorithm

A backward recursion now gives the MAP estimate $\hat{x}_{1:t}$.

First

$$\hat{x}_t = \arg\max_x Q_t(x).$$

Then, for $i = t, \ldots, 2$

$$\hat{x}_{i-1} = \arg\max_x{}' \left\{ Q_{i-1}(x') + \log \Pr(X_i = \hat{x}_i | X_{i-1} = x') + \log p(y_i | X_i = \hat{x}_i) \right\}.$$

Discussion

Both the Forward-Backward algorithm and the Viterbi algorithm have a CPU cost that is $O(tK^2)$ where K is the number of states.

Often models will have unknown parameters which need to be estimated. In a fully-Bayesian analysis, priors would be introduced for these, and we could use MCMC to sample the parameters. The Forward-Backward algorithm can be used with an MCMC algorithm to update the path of hidden state.

Often a computationally quicker approach is to maximise the marginal likelihood to estimate the parameters, and then condition on this estimated parameter value. This can underestimate uncertainty, but can be accurate if there is a lot of information about the parameters.

Example: Changepoint Models

We will describe a simple, but non-standard application of the Forward-Backward algorithm: to multiple changepoint models.

The key is seeing how we can re-formulate the problem as a HMM.

We first describe a motivating example.

Example: Typhoid Divergence

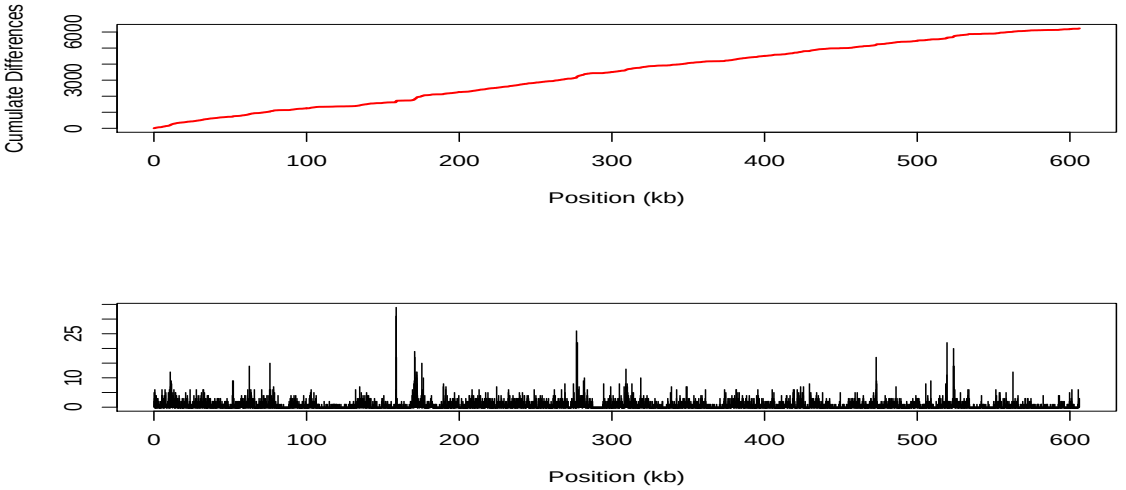
Consider data on the divergence of two strains of (say) *Salmonella*.

The data is obtained by (i) aligning regions of the genome of the two strains; (ii) finding the nucleotide differences between the two aligned sequences.

This divergence data can be informative about how closely related the two strains are; and how *Salmonella* has evolved.

We consider analysing data from two strains Typhi and Paratyphi A. [Data taken from Didelot et al. 2007.]

Data: Typhi v Paratyphi A



A simple model

A simple hierarchical model would be:

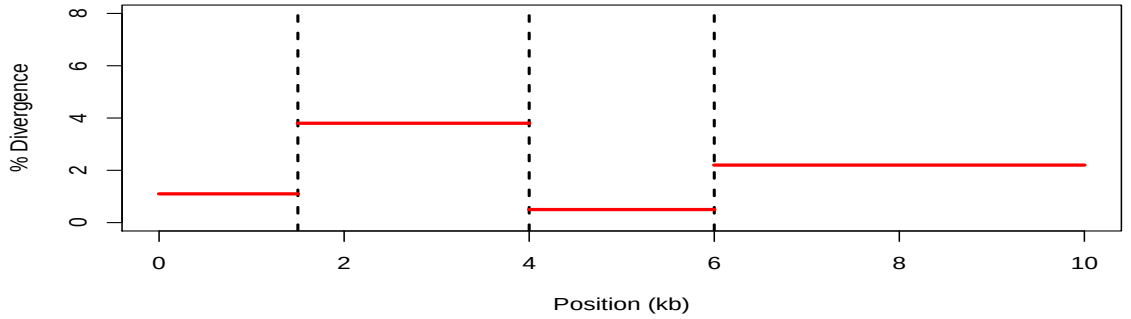
(i) A model for the number and position of changepoints; these will split the data into segments.

(ii) For segment k we associate a parameter β_k that governs the mean divergence in that segment.

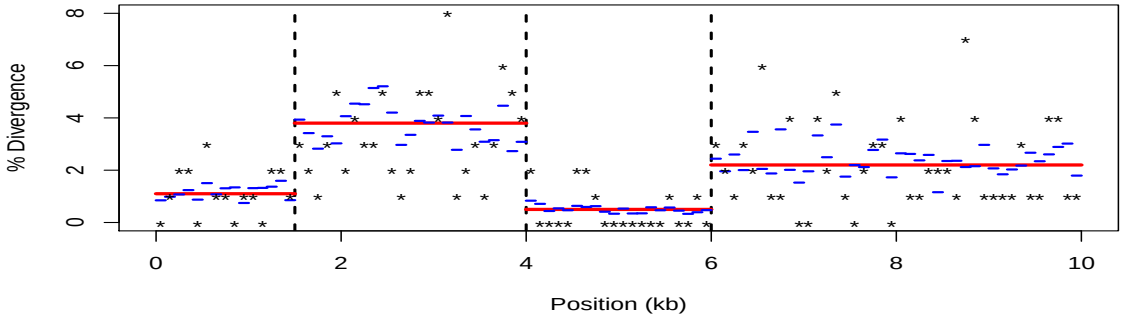
(iii) For each window in the segment we simulate an expected number of differences: $\theta_i \sim \text{Gamma}(\alpha, \beta_k)$.

(iv) The data within window i is $\text{Poisson}(\theta_i)$.

A simple model



A simple model



Changepoint Model

We are interested in estimating the number of changepoints, m, and their position, $\tau_1, \tau_2, \ldots, \tau_m$. We let $\tau_0 = 0$ and $\tau_{m+1} = t$. We have $m+1$ segments, with segment k consisting of observations $\mathbf{y}_{\tau_k+1:\tau_{k+1}}$.

For a segment consisting of observations $\mathbf{y}_{j+1:i}$ we will have a set of unknown parameters, β with a prior distribution, $\pi(\beta)$, independent of parameters in other segments.

The marginal likelihood for the segment

$$P(j,i) = \int p(\mathbf{y}_{j+1:i}|\beta, \mathbf{y}_{1:i-1})\pi(\beta)\mathrm{d}\beta,$$

and assume that these probabilities can be calculated for all $j < i$.

Changepoint Model (2)

Consider changepoint models which have a conditional independence property: given the position of a changepoint, data after the changepoint contain no information about segment parameters and changepoints prior to the changepoint.

This implies that the changepoints satisfy a Markov property.

Changepoint model in State Space Form

Introduce the state X_t to be the time of the most recent changepoint prior to time t.

We have that X_t is a Markov process, and that $X_i = i - 1$ or $X_i = X_{i-1}$, corresponding to the presence or absence of a changepoint at time $i - 1$.

Using the conditional independence property, it can be shown that

$$p(y_i | X_i = j, \mathbf{y}_{1:i-1}) = \frac{P(j, i)}{P(j, i-1)},$$

where $P(\cdot, \cdot)$ is the marginal likelihood.

Forward-Backward for changepoint model

The filtering recursions become

$$p(X_i = j|\mathbf{y}_{1:i}) \propto \frac{P(j,i)}{P(j,i-1)} p(X_i = j|X_{i-1} = j) p(X_{i-1} = j|\mathbf{y}_{1:i-1})$$

for $j < i-1$, and

$$p(X_i = i-1|\mathbf{y}_{1:i}) \propto P(i-1,i) \sum_{j=0}^{i-2} p(X_i = i-1|X_{i-1} = j) p(X_{i-1} = j|\mathbf{y}_{1:i-1})$$

These can be calculated recursively as in the Forward-Backward algorithm.

The normalising constant of these equations is just $p(y_i|\mathbf{y}_{1:i-1})$.

Forward-Backward for changepoint model (2)

Furthermore, simulating from the joint distribution of changepoints is straightforward via backward simulation, with

$$p(X_i|\mathbf{y}_{1:t}, X_{i+1} = i) = p(x_i|\mathbf{y}_{1:i})p(x_{i+1} = i|x_i)$$

and $X_i = j$ if $X_{i+1} = j$ for $j < i$.

Analysis of *Salmonella* data: Model specifics

We assume an Exponential prior on the distribution of segment lengths.

Our prior for β is related to an $F(2a, 2b)$-distribution:

$$\pi_0(\beta; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \frac{\beta^{a-1}}{(1+\beta)^{a+b}}$$

In practice we use a mixture distribution:

$$\pi(\beta) = p\pi_0(\beta; a_1, b_1) + (1-p)\pi_0(\beta; a_2, b_2).$$

Calculating $P(s,t)$

We can first integrate out θ_i:

$$p(y_i|\beta) = \frac{\Gamma(\alpha+y_i)\beta^\alpha}{\Gamma(\alpha)y_i!(\beta+1)^{\alpha+y_i}}.$$

Then we can integrate out β. Assuming prior $\pi_0(\beta;a,b)$, we get

$$P_0(s,t;a,b) = \frac{\Gamma(a+b)\Gamma(a+(t-s)\alpha)\Gamma(b+S)}{\Gamma(a)\Gamma(b)\Gamma(a+b+(t-s)\alpha+S)},$$

where S is the sum of observations in the segment.

Then we have $P(s,t) = pP_0(s,t;a_1,b_1) + (1-p)P_0(s,t;a_2,b_2)$.

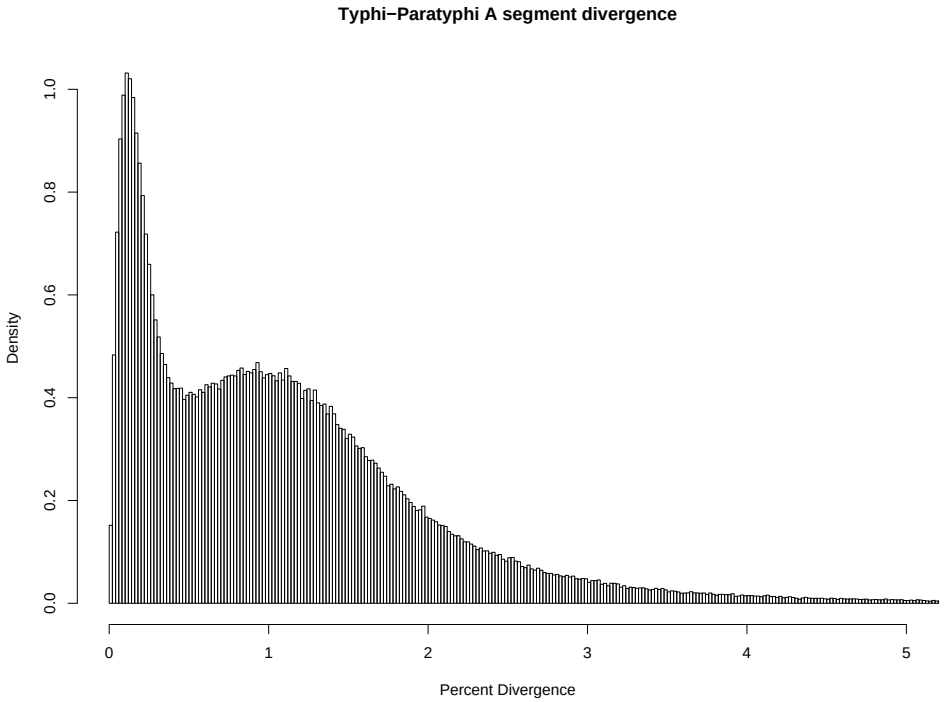
Results for *Salmonella* data

Solving the filtering recursions takes about 1 minute on a desktop PC.

We estimated the hyperparameters (for distribution of segment lengths; and prior on βs) via EM. Convergence took about 20 iterations.

Of interest is the distribution of mean divergences across segments. The Bi-modality of this distribution suggests substantial recent genetic exchange between the two strains.

Results for *Salmonella* data



Discussion

Similarly, we can use the Viterbi algorithm to obtain the MAP estimates of change-point location.

Both Forward-Backward and Viterbi algorithms are $O(t^2)$ in computation. This can bue reduced through truncation/pruning. For the Forward-Backward algorithm this introduces error (Fearnhead and Liu 2007, though this can be avoided using Pseudo-Marginal MCMC ideas); for the Viterbi this does not (Killick et al. 2012). In some cases computation can become $O(t)$.

The segment prior, $\pi(\beta)$, needs to be carefully chosen. Use of uninformative priors is not possible here.

One approach is to introduce hyper-parameters $\pi(\beta|\gamma)$ and uninformative priors on γ.

General HMMs: Sequential Monte Carlo (SMC)

General-state HMMs

We define a Hidden Markov Model (HMM):

(i) a hidden (latent) state, X_t, which is a Markov chain:

$$p(x_t|\mathbf{x}_{1:t-1}) = p(x_t|x_{t-1}),$$

where $\mathbf{x}_{1:t-1} = \{x_1, \ldots, x_{t-1}\}$.

(ii) an series of observations, Y_t, where Y_t is conditionally independent of X_s and Y_s for $s \neq t$ given X_t.

We can write down the following joint probability:

$$p(x_{1:T}, y_{1:T}) = p(x_1) \left(\prod_{t=2}^{T} p(x_t|x_{t-1}) \right) \left(\prod_{t=1}^{T} p(y_t|x_t) \right).$$

Filtering Problem

We can derive recursions for general HMMs in the same way as for discrete-state HMMs. The difference is that in general they cannot be solved.

One important exception is for Linear-Gaussian models: where the solution of the recursions gives the Kalman Filter.

We will focus on non-linear and non-Gaussian models. We will also focus on filtering: that is calculating $p(x_t|y_{1:t})$.

General Filtering Recursions

Initially:

$$p(x_1|y_1) = \frac{p(x_1)p(y_1|x_1)}{p(y_1)} \propto p(x_1)p(y_1|x_1).$$

Then for t=2,3,...

$$p(x_t|y_{1:t}) = \frac{p(x_t|y_{1:t-1})p(y_t|x_t)}{p(y_t|y_{1:t-1})} \propto p(y_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1},$$

and

$$p(y_t|y_{1:t-1}) = \int \left\{ p(y_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1} \right\} \mathrm{d}x_t.$$

Path Recursions

If we are interested in the path of the hidden process, we can derive equivalent recursions for a new state

$$\mathcal{P}_t = (X_1, \ldots, X_t) = \mathbf{X}_{1:t}.$$

Importance Sampling

We will look at Monte Carlo methods for solving the filtering equations.

These methods have a variety of names: Sequential Importance Sampling, Sequential Importance Resampling Filters, Particle Filters and Sequential Monte Carlo amongst others.

The are based on the use Importance Sampling; and ideas of approximating distributions by weighted samples.

Importance Sampling

Importance Sampling (**IS**) is a Monte Carlo integration technique. Consider the integral

$$I = \int h(x)p(x)\mathrm{d}x = \int h(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x,$$

where $p(x)$ and $q(x)$ are densities, $h(x)$ is arbitrary and $p(x) > 0 \Rightarrow q(x) > 0$.

We can view this as an expectation with respect to $q(x)$. Thus

1. Sample x_i, $i = 1, \ldots, N$, iid from $q(x)$;
2. Estimate the integral by the unbiased, consistent estimator:

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N} h(x_i)\frac{p(x_i)}{q(x_i)}.$$

Importance Sampling

In Bayesian Statistics, the distribution of interest, $p(x)$, is often known only up to a normalising constant.

In this case we can write expectations as

$$I = \frac{\int h(x)p(x)\mathrm{d}x}{\int p(x)\mathrm{d}x} = \frac{\int h(x)\{p(x)/q(x)\}q(x)\mathrm{d}x}{\int \{p(x)q(x)\}q(x)\mathrm{d}x}.$$

Importance Sampling

We can view this as the ratio of expectations with respect to $q(x)$. Thus

1. Sample x_i, $i = 1, \ldots, N$, iid from $q(x)$;
2. Estimate I by the consistent estimator:

$$\hat{I} = \frac{\frac{1}{N}\sum_{i=1}^{N} h(x_i)\frac{p(x_i)}{q(x_i)}}{\frac{1}{N}\sum_{i=1}^{N} \frac{p(x_i)}{q(x_i)}}.$$

Importance Sampling

Now if we define weights

$$w_i = \frac{p(x_i)/q(x_i)}{\sum_{j=1}^{N} p(x_j)/q(x_j)}$$

we get

$$\hat{I} = \sum_{i=1}^{N} w_i h(x_i).$$

Now the choice of $h(\cdot)$ was arbitrary, so we can use the sample and weights $\{x_i, w_i\}_{i=1}^{N}$ to approximate any (suitable) expectation with respect to p.

Thus we can view $\{x_i, w_i\}_{i=1}^{N}$ as giving an approximation to $p(x)$.

Accuracy of Importance Sampling

The accuracy of $\hat{I}$ depends on both the weight function and the value of $h(\cdot)$.

However, as a rough guide Liu (1996) show that for most $h(\cdot)$ it depends mostly on the variance of the weights.

One way of quantifying this is in terms of an effective sample size. This can be estimated as

$$\text{ESS} = \frac{N}{1 + \text{Var}\left(\frac{p(X)}{q(X)}\right)} \approx \frac{\left(\sum_{i=1}^{N} w_i\right)^2}{\sum_{i=1}^{N} w_i^2}.$$

Importance Sampling is roughly as accurate as using an iid sample of size ESS from $p(x)$.

Sequential Importance Sampling

How can we use Importance Sampling to solve the filtering recursions?

The idea is to approximate $p(x_t|y_{1:t})$ by a weighted sample (of particles) $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$.

We then only need to think how to generate our weighted-particle approximation to $p(x_t|y_{1:t})$ given a weighted-particle approximation to $p(x_{t-1}|y_{1:t-1})$.

Sequential Importance Sampling

Assume we have a weighted-partivcle approximation $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$ to $p(x_{t-1}|y_{1:t-1})$.

If substitute this into the filtering recursions we get approximations:

$$\hat{p}(x_t|y_{1:t}) \propto p(y_t|x_t) \sum_{i=1}^{N} w_{t-1}^{(i)} p(x_t|x_{t-1}^{(i)}),$$

and

$$\hat{p}(y_t|y_{1:t-1}) = \int \left(p(y_t|x_t) \sum_{i=1}^{N} w_{t-1}^{(i)} p(x_t|x_{t-1}^{(i)}) \right) \mathrm{d}x_t.$$

Both can be approximated using importance sampling.

SIR Filter

A natural choice of proposal distribution for the importance sampling is

$$q(x_t) = \sum_{i=1}^{N} w_{t-1}^{(i)} p(x_t | x_{t-1}^{(i)}).$$

This is the choice of the SIR filter of Gordon, Salmond and Smith (1993).

Sampling from this can be done by

(i) **Resample** an existing particle, $x_{t-1}^{(i)}$, with probability $w_{t-1}^{(i)}$.

(ii) **Propagate** the particle to time t by sampling an x_t-value from $p(x_t | x_{t-1}^{(i)})$.

SIR Filter

Initialise

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

(1) Sample $x_1^{(i)}$ from $p(x_1)$.

(2) Assign weights

$$w_1^{(i)} \propto p(y_1|x_1^{(i)}),$$

(weights normalised to sum to 1).

(3) Estimate Likelihood:

$$\hat{p}(y_1) = \frac{1}{N} \sum_{i=1}^{N} p(y_1|x_1^{(i)}).$$

SIR Filter

Recursion:for $t = 2, 3, \ldots$; And for $i = 1, \ldots, N$:

(1) Resample: simulate $\tilde{x}(i)_{t-1}$ by resampling from existing particles, $\{x_{t-1}^{(i)}\}_{i=1}^{N}$, with probabilities $\{w_{t-1}^{(i)}\}_{i=1}^{N}$.

(2) Propagate: simulate $x_t^{(i)}$ from $p(x_t|\tilde{x}_{t-1}^{(i)})$.

(3) Assign weights (normalised to sum to 1):

$$w_t^{(i)} \propto p(y_t|x_t^{(i)}).$$

(4) Estimate Likelihood:

$$\hat{p}(y_t|y_{1:t-1}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t^{(i)}).$$

SIR Path Filter

We can implement the same filter, but with interest in the path of the hidden state.

The algorithm is identical, except our particles are values of the path $\mathcal{P}_t = \mathbf{X}_{1:t}$.

In the resample step we have to resample $\tilde{\mathcal{P}}_{t-1}^{(i)}$.

In the propagate step we simulate $x_t^{(i)}$ from $p\left(x_t|\tilde{\mathcal{P}}_{t-1}^{(i)}\right)$, and set

$$\mathcal{P}_t^{(i)} = \left(\tilde{\mathcal{P}}_{t-1}^{(i)}, x_t^{(i)}\right).$$

SIS Filter

Independent to the development of the SIR filter, Kong, Liu and Wong (1994) developed a similar filter, called sequential importance sampling.

It did not have the resampling step of the SIR filter. So $\tilde{x}_{t-1}^{(i)} = x_{t-1}^{(i)}$, and the weighting step is altered to:

$$w_t^{(i)} \propto w_{t-1}^{(i-1)} p(y_t | x_t^{(i)}).$$

However this causes the SIS filter to degenerate over medium-to-long time periods.

Resampling

Resampling initially increases Monte Carlo variation. This can be reduced by using stratified resampling algorithms.

However, by producing multiple copies of existing particles it enables us to independently explore the future of these particles.

The benefit gained depends on the variability of the current weights. One common approach is to monitor the Effective Sample Size of the weights: when this drops below a threshold you resample.

Particle Filter Theory

There is extensive theory underpinning particle filters. The main message is that, if the model *mixes* quickly, then it is possible to develop particle filters whose Monte Carlo variability is bounded in time. We call such filters stable.

Roughly, the idea of mixing quickly is related to the amount the distribution $X_T|y_{1:T}, x_t$ depends on x_t decaying exponentially as $T - t$ increases.

Resampling is crucial to obtaining long-term stability of particle filter algorithms (when it is possible).

Filters for the path of the hidden state will not be stable in time.

Proposal Distribution: Auxilliary Particle Filter

Can we improve on the proposal distribution of the SIR filter?

Often we can. A general approach (Pitt and Shephard 1999) is to use a proposal distribution of the form

$$q_t(x_t) = \sum_{i=1}^{N} \beta_{t-1}^{(i)} q(x_t | x_{t-1}^{(i)}).$$

The $\beta_{t-1}^{(i)}$s can depend on the current set of weighted particles.

Proposal Distribution: Auxilliary Particle Filter

To simulate from this we

(i) **Resample** an existing particle, $x_{t-1}^{(i)}$, with probability $\beta_{t-1}^{(i)}$.

(ii) **Propagate** the particle to time t by sampling an x_t-value from $q(x_t|x_{t-1}^{(i)})$.

If we store the particle label, i, we resample, we can calculate the weight as proportional to

$$p(y_t|x_t)\frac{w_{t-1}^{(i)}p(x_t|x_{t-1}^{(i)})}{\beta_{t-1}^{(i)}q(x_t|x_{t-1}^{(i)})}.$$

Auxilliary Particle Filter Recursion for $t = 2, 3, \ldots$

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

(1) Resample: simulate $\tilde{x}_{t-1}^{(i)}$ by resampling from existing particles, $\{x_{t-1}^{(i)}\}_{i=1}^{N}$, with probabilities $\{\beta_{t-1}^{(i)}\}_{i=1}^{N}$. Define k_i such that $\tilde{x}_{t-1}^{i} = x_{t-1}^{(k_i)}$.

(2) Propagate: simulate $x_t^{(i)}$ from $p(x_t|\tilde{x}_{t-1}^{(i)})$.

(3) Assign weights (normalised to sum to 1):

$$w_t^{(i)} \propto p(y_t|x_t^{(i)}) \frac{w_{t-1}^{(k_i)} p(x_t|\tilde{x}_{t-1}^{(i)})}{\beta_{t-1}^{(k_i)} q(x_t|\tilde{x}_{t-1}^{(i)})}.$$

(4) Estimate Likelihood:

$$\hat{p}(y_t|y_{1:t-1}) = \frac{1}{N} \sum_{i=1}^{N} p(y_t|x_t^{(i)}) \frac{w_{t-1}^{(k_i)} p(x_t^{(i)}|\tilde{x}_{t-1}^{(i)})}{\beta_{t-1}^{(k_i)} q(x_t^{(i)}|\tilde{x}_{t-1}^{(i)})}.$$

Proposal Distribution

There is an optimal proposal distribution:

$$\beta_{t-1}^{(i)} \propto w_{t-1}^{(i)} p(y_t|x_{t-1}^{(i)}) = w_{t-1}^{(i)} \int p(y_t|x_t) p(x_t|x_{t-1}^{(i)}) \mathrm{d}x_t,$$

and

$$q(x_t|x_{t-1}^{(i)}) = p(x_t|x_{t-1}^{(i)}, y_t) \propto p(y_t|x_t) p(x_t|x_{t-1}^{(i)}).$$

These can be calculated for some models, or approximated (e.g. using Laplace approximations).

Estimating Parameters within SMC

Parameter Estimation

Until now we have assumed any parameters in our model are known.

Now we will introduce an unknown parameter vector θ.

Our model is still a HMM but with transition density $p(x_t|x_{t-1}, \theta)$ and conditional distribution of observations $p(y_t|x_t, \theta)$.

Interest is in $p(\theta|y_{1:t})$ or $p(\theta, x_t|y_{1:t})$.

Parameter Estimation: why not MCMC?

In theory we could use MCMC to sample from $p(\theta, x_{1:t}|y_{1:t})$, which would give us a sample from the distribution of interest.

However, this is not feasible for online applications.

Also, even for offline applications, MCMC may mix poorly. MCMC algorithms often struggle to update the path of the state process, $x_{1:t}$, if the model has strong dependencies.

Parameter Estimation: Easy by SMC?

It is simple to adapt standard SMC/Particle filter algorithms to parameter estimation.

All we do is introduce a new state, $Z_t = (X_t, \theta_t)$, which stores the current state of the hidden process, and the value of the parameter.

Z_t is a (hidden) Markov process: dynamics of the X_t component given by the model for X_t; and with $\theta_t = \theta_{t-1}$ as the parameter is constant over time.

We then apply a particle filter to sample from $p(z_t|y_{1:t}) = p(x_t, \theta|y_{1:t})$.

Parameter Estimation by SMC

When we add the parameter to the state, $Z_t = (X_t, \theta_t)$, our new state model has long-term dependence.

Hence an SMC/particle filter will degenerate over time.

How best to deal with parameter estimation within particle filters is still an open problem.

Parameter Estimation by SMC

Three classes of approach:

Kernel Density Estimation. Use of KDE ideas can allow us to generate new parameter values (Liu and West 1999).

MCMC within SMC. We can use MCMC within an SMC algorithm to update parameter values (Gilks and Berzuini 2001, Fearnhead 2002 and Störvik 2002). Recently this has been reinvented as particle learning (Carvalho et al. 2010).

Maximum Likelihod Methods. We can estimate the log-likelihood and score function within SMC. These can be used to iteratively search for good parameter values (, Ionides et al. 2011, Poyiadis et al. 2011).

Kernel Density Estimation (1-d)

Introduce a Kernel, $K(\theta)$. This is just a density function (often symmetric, with mode at $\theta = 0$).

Given a $\theta^{(1)}, \ldots, \theta^{(N)}$ from an unknown distribution, we can estimate the unknown density function as

$$\hat{f}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma} K\left([\theta - \theta^{(i)}]/\sigma \right).$$

The accuracy of this estimate does not depend too much on the choice of Kernel, $K(\cdot)$. The choice of bandwidth, σ is important.

Kernel Density Estimation

Let the Kernel be the density function for a standard Gaussian random variable.

Sampling from $\hat{f}(\theta)$ can be achieved by

(i) Sample a value $\theta^{(i)}$ at random from $\{\theta^{(j)}\}_{j=1}^{N}$.

(ii) Sample a value ϵ from a Gaussian distribution with variance σ^2; and set

$$\theta = \theta^{(i)} + \epsilon.$$

By adding the noise ϵ we can generate values of θ that we have not sampled.

Kernel Density Estimation: Bandwidth Choice

The choice of bandwidth is important in KDE.

Theory shows that the bandwidth should be proportional to the standard deviation of the underlying distrbution, and should decrease as N increases.

However, (asymptotically) optimal bandwith choices depend on an constant that depends on the underlying distribution, and is often poor for small values of N.

Theory is for iid samples: and not weighted samples as in IS.

SMC with KDE Algorithm

(0) Estimate Variance: calculate variance of posterior for paramaters from particle approximation. Denote, by $\hat{\Sigma}_{t-1}$.

(1) Resample: simulate $(\tilde{x}_{t-1}^{(i)}, \tilde{\theta}_{t-1}^{(i)})$ by resampling from existing particles, $\{(x_{t-1}^{(i)}, \theta_{t-1}^{(i)})\}_{i=1}^{N}$, with probabilities $\{w_{t-1}^{(i)}\}_{i=1}^{N}$.

(2a) Jitter Parameter. Sample $\epsilon^{(i)} \sim \mathrm{N}\left(0, h\hat{\Sigma}_{t-1}\right)$, and set $\theta_t^{(i)} = \tilde{\theta}_{t-1}^{(i)} + \epsilon^{(i)}$.

(2b) Propagate: simulate $x_t^{(i)}$ from $p(x_t|\tilde{x}_{t-1}^{(i)}, \theta_t^{(i)})$.

(3) Assign weights (normalised to sum to 1):

$$w_t^{(i)} \propto p(y_t|x_t^{(i)}, \theta_t^{(i)}).$$

SMC with KDE: Comments

The problem with the KDE approach is that by adding noise, we increase the variance of our approximation to the posterior, $p(\theta|y_{1:t})$.

At some point this increase counteracts any decrease we obtain through the new observation: the algorithm tends to stop learning.

Use of Shrinkage: Liu and West Algorithm

Liu and West (1999) suggested using shrinkage to counteract the increase of variance in KDE methods.

If we have a sample $\theta^{(1)}, \ldots, \theta^{(N)}$ from an unknown distribution with mean $\mathbf{m}$ and variance Σ. We then sample ϵ from a Gaussian distribution with variance $h\Sigma$.

The update in step (ii) above becomes:

$$\theta = \lambda\theta^{(i)} + (1 - \lambda)\mathbf{m} + \epsilon.$$

By choosing $\lambda^2 + h = 1$ we ensure θ has the correct mean and variance.

Use of Shrinkage: Liu and West Algorithm

Changes to KDE algorithm:

(0) Estimate Mean and Variance: calculate mean and variance of posterior for paramaters from particle approximation. Denote, by $\mathbf{m_{t-1}}$ and $\hat{\Sigma}_{t-1}$ respectively.

(2a) Jitter Parameter. Sample $\epsilon^{(i)} \sim \mathrm{N}\left(0, h\hat{\Sigma}_{t-1}\right)$, and set

$$\theta_t^{(i)} = \lambda\tilde{\theta}_{t-1}^{(i)} + (1-\lambda)\mathbf{m_{t-1}} + \epsilon^{(\mathbf{i})}.$$

Particle Learning

An alternative approach to parameter inference by Particle Filters is to use MCMC updates within the Particle Filter algorithm to generate new parameter values.

This is easiest to see with a particle filter that stores the path of the latent process, and the parameter; so has a state $Z_t = (\mathbf{X}_{1:t}, \theta) = (\mathcal{P}_t, \theta)$.

We run the basic Particle Filter algorithm for state Z_t, but add an extra MCMC-update for each particle after resampling. This step will draw a new θ value from an iteration of an MCMC algorithm with $p(\theta, \mathcal{P}_t | y_{1:t})$ as its stationary distribution.

This is simplest to do when we can simulate from $p(\theta | \mathcal{P}_t, y_{1:t})$.

SIR Filter with MCMC

Recursion:for $t = 2, 3, \ldots$; And for $i = 1, \ldots, N$:

(1a) Resample: simulate $\tilde{z}_{t-1}^{(i)} = (\tilde{\theta}_{t-1}^{(i)}, \tilde{\mathcal{P}}_{t-1}^{(i)})$ by resampling from existing particles, $\{z_{t-1}^{(i)}\}_{i=1}^{N}$, with probabilities $\{w_{t-1}^{(i)}\}_{i=1}^{N}$.

(1b) Simulate $\theta_t^{(i)}$ from $p(\theta|\tilde{\mathcal{P}}_{t-1}^{(i)}, y_{1:t-1})$.

(2) Propagate: simulate $z_t^{(i)} = (\theta_t^{(i)}, \mathcal{P}_t^{(i)})$ from $p(\mathcal{P}_t|\tilde{\mathcal{P}}_{t-1}^{(i)}, \theta_t^{(i)})$.

(3) Assign weights (normalised to sum to 1):

$$w_t^{(i)} \propto p(y_t|\mathcal{P}_t^{(i)}, \theta_t^{(i)}).$$

Particle Learning

Often the CPU and storage costs of the above algorithm can be reduced by storing sufficient statistics of the path.

This approach can work well, but it still suffers from degeneracy for long time-series.

New MCMC: Pseudo-Marginal and Particle MCMC

or

Exact Approximate MCMC

Missing Data Problems

Consider the so-called missing data models.

We can write down the joint probability of parameters, θ, a latent process, $\mathbf{x}$, and observations, $\mathbf{y}$:

$$p(\theta, \mathbf{x}, \mathbf{y}) = p(\theta)p(\mathbf{x}|\theta)p(\mathbf{y}|\mathbf{x}, \theta).$$

The latent process is unobserved, and we are interested in the posterior

$$p(\theta|\mathbf{y}) \propto p(\theta)p(\mathbf{y}|\theta) \propto p(\theta) \int p(\mathbf{x}|\theta)p(\mathbf{y}|\mathbf{x}, \theta)\mathrm{d}\mathbf{x}.$$

Problems with MCMC

Examples of this sort of model include the HMMs we have considered.

In general we are unable to integrate out the latent process. Thus MCMC algorithms need to explore the joint parameter-latent process space, $\theta, \mathbf{x}$.

Often MCMC mixes slowly, either because of difficulty in updating the latent-process, $\mathbf{x}$, or due to strong correlation between θ and $\mathbf{x}$.

Marginal MCMC

Ideally we would integrate out the latent-process, calculating the marginal likelihood

$$p(\mathbf{y}|\theta) = \int p(\mathbf{x}|\theta)p(\mathbf{y}|\mathbf{x},\theta)\mathrm{d}\mathbf{x}.$$

Then we could run an MCMC on the parameter space only. Such an MCMC algorithm proposes new parameter values, θ' from a proposal $q(\theta'|\theta)$, and accepts them with probability

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')p(\mathbf{y}|\theta')}{q(\theta'|\theta)p(\theta)p(\mathbf{y}|\theta)}\right\}.$$

Approximate Marginal MCMC: idea

Whilst we cannot implement this marginal MCMC without calculating $p(\mathbf{y}|\theta)$: what if we can approximate this? For example by Importance Sampling.

We could substitute such an approximation into the acceptance probability. Perhaps, if the approximation is good, we will sample from a good approximation to the true posterior.

There are two natural ways of implementing such an approximate MCMC algorithm: does the choice matter?

Approximate Marginal MCMC: Algorithm 1

Assume a current state θ of the MCMC algorithm

(1) Propose a move to θ'.

(2) Calculate Monte Carlo approximations to the marginal likelihood: Z to $p(\mathbf{y}|\theta)$; and Z' to $p(\mathbf{y}|\theta')$.

(3) Accept θ' with probability

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')Z'}{q(\theta'|\theta)p(\theta)Z}\right\}.$$

Approximate Marginal MCMC: Algorithm 2

Assume a current state θ and estimate Z of the marginal likelihood $p(\mathbf{y}|\theta)$.

(1) Propose a move to θ'.

(2) Calculate Monte Carlo approximation to the new marginal likelihood: Z' to $p(\mathbf{y}|\theta')$.

(3) Accept θ' (and Z') with probability

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')Z'}{q(\theta'|\theta)p(\theta)Z}\right\}.$$

Does it Matter which Algorithm you use?

Methods like Algorithm 1 was originally suggested by O'Neill et al. (2000).

Algorithm 2 was proposed by Beaumont 2003 as a more efficient method: it requires calculating one approximation per iteration.

It turns out this comes at the cost of slower mixing.

However, if the Monte Carlo estimator of the marginal likelihood is unbiased then Algorithm 2 samples from the true posterior.

This was shown by Andrieu and Roberts (2009), who called it the pseudo-marginal method.

Validity of Pseudo-Marginal MCMC

Introduce a distribution on $\theta, Z, p(\theta)p(z|\theta)$. Assume that Z is always non-negative, and is an unbiased estimator of the marginal likelihood:

$$\mathrm{E}(Z) = \int zp(z|\theta)\mathrm{d}z = p(\mathbf{y}|\theta).$$

Then you can show that the pseudo-marginal method is an MCMC algorithm with target distribution proportion to $zp(z|\theta)p(\theta)$. The marginal of this target distribution is proportional to

$$\int zp(z|\theta)p(\theta)\mathrm{d}z = p(\theta)\int zp(z|\theta)\mathrm{d}z = p(\theta)p(\mathbf{y}|\theta).$$

[Note often you may not know $p(z|\theta)$, but you can simulate from it: as with importance sampling.]

Pseudo-Marginal MCMC: Mixing

The mixing of the pseudo-marginal algorithm depends on

(i) The mixing of the true-marginal MCMC algorithm.

(ii) The variance in the estimators of the marginal likelihood.

As the variance of the estimators decreases, the mixing will converge on that of the true marginal MCMC algorithm.

The problem with estimators with large variance is that the chain can stick if we ever get a large Z.

Pseudo-Marginal MCMC: State Space Models

If we have a HMM, a natural approach to approximating the likelihood is using a particle filter.

The particle filter gives us estimates $\hat{p}(y_t|\mathbf{y}_{1:t-1})$ of $p(y_t|\mathbf{y}_{1:t-1})$ for $t = 1, \ldots, T$.

Could we use

$$\hat{p}(y_1) \prod_{t=2}^{T} \hat{p}(y_t|\mathbf{y}_{1:t-1}),$$

to estimate $p(\mathbf{y}_{1:T})$?

Particle Metropolis

Yes! This is unbiased estimator (Del Moral (2004), proposition 7.4.1).

Using particle filters to unbiasedly estimate the marginal likelihood within the pseudo-marginal method, gives a simple way of combining them within MCMC.

However Andrieu et al. (2010) noticed that there was a lot more flexibility to use particle filters within MCMC: and developed a number of **particle MCMC** algorithms.

This simple one is called Particle independent Metropolis-Hastings.

Particle independent Metropolis-Hastings: Algorithm

Assume a current state θ and estimate Z of the marginal likelihood $p(\mathbf{y}|\theta)$.

(1) Propose a move to θ'.

(2) Run a Particle Filter to obtain an unbiased approximation to the new marginal likelihood: Z' to $p(\mathbf{y}|\theta')$.

(3) Accept θ' (and Z') with probability

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')Z'}{q(\theta'|\theta)p(\theta)Z}\right\}.$$

Particle marginal Metropolis-Hastings

What if we are interested in the joint posterior, $p(\theta, x_{1:n}|y_{1:n})$?

Andrieu et al. (2010) show that you can implement the previous algorithm, with a Path-Particle Filter, and then sample a path from the particle filter.

If you accept, you accept θ', Z' and $x'_{1:t}$.

Particle marginal Metropolis-Hastings: Algorithm

Assume a current state $(\theta, x_{1:n})$ and estimate Z of the marginal likelihood $p(\mathbf{y}|\theta)$.

(1) Propose a move to θ'.

(2) Run a Path-Particle Filter. Obtain an unbiased approximation to the new marginal likelihood: Z' to $p(\mathbf{y}|\theta')$; and sample a path for the latent process $x'_{1:n}$.

(3) Accept $(\theta', x'_{1:n})$ (and Z') with probability

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')Z'}{q(\theta'|\theta)p(\theta)Z}\right\}.$$

Particle Gibbs

A common approach to designing MCMC algorithms that target $p(\theta, x_{1:n}|y_{1:n})$ is to use a Gibbs sampler, where you iterate between:

(1) Sample a new θ from $p(\theta|x_{1:n}, y_{1:n})$; and

(2) Sample a new $x_{1:n}$ from $p(x_{1:n}|\theta, y_{1:n})$.

This avoids tuning of the proposal.

However, in many cases we cannot sample from $p(x_{1:n}|\theta, y_{1:n})$. Again Andrieu et al. (2010) show how using a particle filters can allow you to do this approximately: but still ensure your MCMC algorithm is valid.

Particle Gibbs: Conditional Simulation

Given a current path $x^*_{1:n}$ and parameter value θ.

(1a) Set $x_1^{(1)} = x_1^*$.

(1b) Simulate $x_1^{(i)} \sim p(x_1|\theta)$ for $i = 2, \ldots, N$.

(1c) Calculate weights $w_1^{(i)} \propto p(y_1|x_1^{(i)})$.

(1d) Store paths $\mathcal{P}_1^{(i)} = x_1^{(i)}$ for $i = 2, \ldots, N$.

Particle Gibbs: Conditional Simulation

(2) For $t = 2, \ldots, n$:

(2a) Set $x_t^{(1)} = x_t^*$.

(2b) For $i = 2, \ldots, N$, sample $A_t^{(i)}$ from $\{1, \ldots, N\}$ with probabilities $\{w_{t-1}^{(1)}, \ldots, w_{t-1}^{(N)}\}$; and simulate

$$x_t^{(i)} \sim p\left(x_t | x_{t-1}^{(A_t^{(i)})}\right).$$

(2c) Calculate weights $w_t^{(i)} \propto p(y_t | x_t^{(i)})$.

(2d) For $i = 2, \ldots, N$, store paths

$$\mathcal{P}_t^{(i)} = \left(\mathcal{P}_{t-1}^{(A_t^{(i)})}, x_t^{(i)}\right).$$

(3) Sample a path from $\{\mathcal{P}_n^{(1)}, \ldots, \mathcal{P}_n^{(N)}\}$ with probabilities $\{w_n^{(1)}, \ldots, w_n^{(N)}\}$.

Particle Gibbs: Why it Works

The particle Gibbs algorithm can be shown to satisfy detailed-balance. The key is to calculate the proposal density as that of proposing the full particle-filter output, and then choosing the new path $x_{1:t}$.

Tedious algebra shows that the ratio of proposals probabilities simplifies to the ratio of conditional densities for the paths given the current parameter value.

Care is needed if you use better resampling strategies: that you implement the conditional SMC simulation correctly.

Recent work has shown how using Particle-Smoothing algorithms (similar to the Forward-Backward algorithm) can improve the Particle Gibbs approach.

Approximate Bayesian Computation

Introduction: Likelihood-free Inference

Recently there has been interest in statistical methods for models that can be simulated from, but for which it is impossible to calculate transition densities or likelihoods.

The general set-up is we have a complex stochastic process, X_t, with unknown parameters θ. For any θ, we can simulate from this process.

We have observations $Y = f(X_{[0,T]})$.

We want to estimate θ; but we cannot calculate $p(Y|\theta)$ (as it involves integrating over the realisations of $X_{[0,T]}$).

Idea of Likelihood-free Inference

Likelihood-free Inference Motivating Idea

- Easy to simulate from model conditional on parameters.
- So run simulations for many parameters.
- See for which parameter value the simulated data sets match observed data best.

Different Likelihood-Free Methods

Likelihood-free methods date back to at least Diggle and Gratton (1984). More recent examples:

- Indirect Inference (Gouriéroux and Ronchetti 1993);
- Approximate Bayesian Computation (ABC) (Marin et al. 2011);
- SIR Filter of Gordon, Salmond and Smith (1993);
- Synthetic Likelihood method of Wood (2010).

Likelihood Free Particle Filters

The SIR filter is a likelihood-free method in the sense that you do not need to be able to calculate $p(x_t|x_{t-1})$, just simulate $X_t|x_{t-1}$.

You still need to be able to calculate $p(y_t|x_t)$. [So likelihood-free is not the best name for this!]

This can be efficient, provided that $p(y_t|x_t)$ is not too peaked, relative to the variability in $X_t|x_{t-1}$.

We need to deal with parameter estimation within the SIR filter: either using Liu and West ideas (say); or by using particle MCMC.

SIR Filter

Recursion:for $t = 2, 3, \ldots$; And for $i = 1, \ldots, N$:

(1) Resample: simulate $\tilde{x}_{t-1}^{(i)}$ by resampling from existing particles, $\{x_{t-1}^{(i)}\}_{i=1}^{N}$, with probabilities $\{w_{t-1}^{(i)}\}_{i=1}^{N}$.

(2) Propagate: simulate $x_t^{(i)}$ from $p(x_t|\tilde{x}_{t-1}^{(i)})$.

(3) Assign weights (normalised to sum to 1):

$$w_t^{(i)} \propto p(y_t|x_t^{(i)}).$$

(4) Estimate Likelihood:

$$\hat{p}(y_t|y_{1:t-1}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t^{(i)}).$$

Likelihood Free Particle Filters: Partially-Observed

Suppose we have a partially-observed system: so $y_t = g(x_t)$. Now $p(y_t|x_t)$ is a point mass. It may be unlikely that we will simulate an X_t value for which $g(X_t) = y_t$.

We can reduce the Monte Carlo variability in the SIR filter, at the expense of approximation, by replacing $p(y_t|x_t)$ by a Kernel (Wilkinson 2011):

$$K\left(\frac{[y_t - g(x_t)]}{h}\right).$$

Likelihood Free Particle Filters: Bias

Unfortunately this can introduce a bias. We can remove the bias by simulating new data in a way consistent with the kernel:

- Simulate $\epsilon_t \sim K(\cdot)$; and define $\tilde{y}_t = y_t + h\epsilon$.
- Weight particle x_t with

$$K\left(\frac{[\tilde{y}_t - g(x_t)]}{h}\right).$$

The bias of the original approximation, and the increased variance of this, is studied in Dean et al. (2011).

Indirect Inference: Overview

Indirect Inference involves:

- Choosing an ancilliary model, which is tractable.
- Finding the MLEs for the parameters in this ancilliary model for your data.
- Simulating lots of new data sets for different values of the parameter in the true model.
- Use the simulated data sets to get an estimate of the parameters of the true model from the MLEs of the ancilliary model's parameters.

Indirect Inference: Approach (1)

Ancilliary Model: has likelihood function $p(\mathbf{x}|\beta)$.

For observed data $\mathbf{x}_{\text{obs}}$ we have MLE for β:

$$\hat{\beta} = \arg\max \left\{ \log p(\mathbf{x}_{\text{obs}}|\beta) \right\}.$$

Indirect Inference: Approach (2)

For a given parameter θ, simulate S data sets from the true model: $\mathbf{y}_{\text{sim}}^{(1)}(\theta), \ldots, \mathbf{y}_{\text{sim}}^{(S)}(\theta)$.

Calculate

$$\hat{\beta}(\theta) = \arg\max \left\{ \frac{1}{S} \sum_{s=1}^{S} \log p(\mathbf{y}_{\text{sim}}^{(s)}(\theta)|\beta) \right\}.$$

Estimate θ as the value that minimises the distance

$$\hat{\theta} = \arg\min (\hat{\beta} - \hat{\beta}(\theta))^T \Sigma (\hat{\beta} - \hat{\beta}(\theta)).$$

Indirect Inference: Discussion

There are two (separate) ideas within Indirect Inference:

(i) Summary of the Data: the ancilliary model constructs a summary of a (high-dimensional) data-set, $\mathbf{x}$, in term of the parameter estimates of the ancilliary model, $\hat{\beta}(\mathbf{x})$.

(ii) Criteria for Estimating Parameters: the choice of minimising a cost-function to get an estimate for θ.

Whilst the above description gives the estimate for θ, asymptotic theory gives results for the variance of this estimator.

Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) is a Bayesian approach to likelihood-free inference.

The set-up is as before, but we now have a prior, $\pi(\theta)$.

Rejection Sampling

If we have discrete data, we can sample from the posterior:

Input: observed data $\mathbf{x}_{\text{obs}}$

For $i = 1, 2, \ldots, n$:

1. Sample parameter vector θ_i from prior $\pi(\theta)$;
2. Simulate data $\mathbf{x}_{\text{sim}}$ from model conditional on θ_i;
3. If $\mathbf{x}_{\text{sim}} = \mathbf{x}_{\text{obs}}$ accept θ_i.

Output is a sample of θ values from the posterior.

ABC Rejection Sampling

Input: observed data $\mathbf{x}_{\text{obs}}$, threshold $h > 0$

For $i = 1, 2, \ldots, n$:

1. Sample parameter vector θ_i from prior $\pi(\theta)$;
2. Simulate data $\mathbf{x}_{\text{sim}}$ from model conditional on θ_i;
3. If $||\mathbf{x}_{\text{sim}} - \mathbf{x}_{\text{obs}}|| < h$ accept θ_i.

Output is a sample of θ values from an approximation to the posterior.

Approximations in ABC

Firstly consider low d-dimensional data and assume $||\mathbf{x}_{\text{sim}} - \mathbf{x}_{\text{obs}}||$ is Euclidean distance.

Call the distribution that the rejection sampler samples from the ABC posterior.

Then as $h \to 0$ the ABC posterior converges to the true posterior. However the Monte Carlo error is inversely proportional to the expected acceptance probability, which is $O(h^d)$.

ABC Algorithms

Rejection sampling ABC is the most simplistic ABC algorithm.

More efficient algorithms exist, for example based on MCMC or SMC. These help sample θ values in areas of high ABC-posterior probability.

They cannot overcome issues of low probability of a match between simulated and observed data for parameter values near the posterior mode.

ABC-MCMC (Marjoram et al. 2003)

This can be viewed as version of the pseudo-marginal method. Assume a current state θ.

(1) Propose a move to $\theta' \sim q(\theta'|\theta)$.

(2) Simulate data $\mathbf{x}_{\text{sim}} \sim p(\mathbf{x}|\theta)$.

(3) Compare simulated data with observed. If $||\mathbf{x}_{\text{sim}} - \mathbf{x}_{\text{obs}}|| > h$ reject θ'.

(4) Accept θ' with probability

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')}{q(\theta'|\theta)p(\theta)}\right\}.$$

ABC Posterior

Now we focus on choice of the condition $||\mathbf{x}_{\text{sim}} - \mathbf{x}_{\text{obs}}|| < h$.

For simplicity consider ABC rejection sampling: though ideas are the same for other implementations of ABC.

High-dimensional data

For high-dimensional data, ABC uses low-dimensional summary statistics, $S(\mathbf{x})$.

Distance is then defined in terms of (say) Euclidean distance between $S(\mathbf{x}_{\text{sim}})$ and $S(\mathbf{x}_{\text{obs}})$.

Ideally we want $S(\cdot)$ to be as low-dimensional as possible, but to contain as much information about the parameters. How do we choose such an $S(\cdot)$ in practice?

Choosing Summary Statistics

Most discussion on choice of summary statistics has been linked to the idea of low-dimensional sufficient statistics. If these exist then they would be the optimal choice. This would mean the ABC posterior can be viewed as an approximation to the true posterior.

However, do such sufficient statistics exist? If so how do we find them?

In practice, users of ABC tend to choose what seems sensible!

Recent work on ABC model choice has suggested that viewing ABC as providing an approximation to true Bayesian analysis is unrealistic in practice. (Robert et al. 2011).

Semi-automatic ABC

Rather than aim for the ABC posterior to approximate the true posterior, we could aim for inferences based on the ABC posterior that will be accurate.

[We would also like to know how to interpret probability statements from the ABC posterior. It turns out that if we use low-dimensional summary statistics, and hence h is small, that these probability statements will be approximately correct.]

Accuracy of ABC

Consider $\phi_i = f_i(\theta)$, for $i = 1, \ldots, d$. We will quantify accuracy in terms of estimating $\phi = (\phi_1, \ldots, \phi_d)$.

Often we would have $\phi = \theta$.

We define this in terms of a Loss Function. We consider square error-loss:

$$L(\phi, \hat{\phi}; A) = (\phi - \hat{\phi})A(\phi - \hat{\phi})^T,$$

where A is a $d \times d$ positive-definite matrix.

Choice of Summary Statistics

Consider $h \to 0$. An optimal choice of summary statistics, in terms of minimising square error-loss is given by for $i = 1, 2, \ldots, d$:

$$S_i(\mathbf{y}) = \mathrm{E}(\phi_i|\mathbf{y}).$$

This follows from

- Given the data, $\mathbf{y}$, the optimal estimate of ϕ_i is $\mathrm{E}(\phi_i|\mathbf{y})$.
- If $S_i(\mathbf{y}) = \mathrm{E}(\phi_i|\mathbf{y})$, then $\mathrm{E}_{\text{ABC}}(\phi_i|S(\mathbf{y})) \to \mathrm{E}(\phi_i|\mathbf{y})$ as $h \to 0$.

[This is optimal for any A]

Implementation

Our result on Summary Statistics is not of direct use: as we do not know the posterior means.

However we can use simulation to estimate the functions $\mathrm{E}(\phi_i|\mathbf{y})$.

We will describe how to construct $S_1(\mathbf{y})$ and approximation to $\mathrm{E}(\phi_1|\mathbf{y})$

Implementation

(i) Simulate $\theta_1, \ldots, \theta_N$; for θ_i calculate $\phi_1^{(i)} = f_1(\theta_i)$.

(ii) Simulate data sets $\mathbf{y}_1, \ldots, \mathbf{y}_N$; where $\mathbf{y}_i$ is simulated from $f(\mathbf{y}|\theta_i)$.

(iii) Fit a model of the form

$$\phi_1^{(i)} = \mathrm{E}(\phi_1|\mathbf{y}_i) + \epsilon_i.$$

We use Linear Regression, including appropriate functions of $\mathbf{y}$ as possible covariates; so

$$\mathrm{E}(\phi_1|\mathbf{y}) \approx \sum_{j=1}^{M} \beta_j g_j(\mathbf{y}).$$

(iv) Set $S_1(\mathbf{y}) = \sum_{j=1}^{M} \hat{\beta}_j g_j(\mathbf{y})$.

Implementation: Details

One implementation (Fearnhead and Prangle 2012) is

(1) **Preliminary Run**. Gain rough estimate of region of high posterior mass. (Only needed if vague priors.)

(2) **Estimate Summary Statistics**. Simulate parameters in this region, and data sets for each parameter value. Fit Linear Regression model to estimate Summary Statistics.

(3) **ABC**. Run ABC with chosen summary statistics.

Roughly spend 10%, 40% and 50% of time on (1)–(3) respectively.

Semi-Automatic ABC

Semi-automatic implementation – as (2) may require user-input (e.g. in choosing appropriate covariates for the linear model).

Whilst our approach still involves user input – we can choose a large number of functions of the data as possible covariates for the linear model. We can use standard methods (e.g. BIC) to choose appropriate covariates for the linear model.

Thus the performance of ABC is more robust to this choice then if we choose a small number of summary statistics to base ABC on.

Example: partially observed queue

- Single server queue (initially empty)
- Service times uniform on $[\theta_1, \theta_2]$
- Inter-arrival times exponential with rate θ_3
- Only inter-departure times $X_1, X_2, \ldots, X_{50}$ observed
- Priors:
  - $\theta_1, \theta_2 - \theta_1$ both uniform on $[0, 10]$,
  - θ_3 uniform on $[0, 1/3]$.
- Previous ABC analysis (Blum and Francois 2010) used 10 quantiles of data as summary statistics

Semi-automatic ABC

1(a) Choose an appropriate training distribution on parameter space

1(b) Sample N parameter vectors from training distribution

Iteration	θ_1	θ_2	θ_3
1	9.25	13.34	4.1
2	9.91	15.21	12.5
3	7.99	14.26	14.1
4	8.67	11.16	6.6
5	6.58	10.46	11.9
⋮	⋮	⋮	⋮

Semi-automatic ABC

1(c) Transform parameters to ϕ (quantities of interest): e.g $\phi_1 = \theta_1$, $\phi_2 = \theta_2$ and $\phi_3 = 1/\theta_3$:

Iteration	ϕ_1	ϕ_2	ϕ_3
1	9.25	13.34	0.24
2	9.91	15.21	0.08
3	7.99	14.26	0.07
4	8.67	11.16	0.15
5	6.58	10.46	0.08
⋮	⋮	⋮	⋮

Semi-automatic ABC

2 For each θ simulate data X from the model

And record some explanatory variables $f(X)$

We'll use ordered inter-departure times

Iteration	ϕ_1	ϕ_2	ϕ_3	$X^{(1)}$	$X^{(2)}$	...
1	9.25	13.34	0.24	10.16	10.30	...
2	9.91	15.21	0.08	9.39	9.43	...
3	7.99	14.26	0.07	6.96	7.08	...
4	8.67	11.16	0.15	7.97	8.20	...
5	6.58	10.46	0.08	7.16	7.18	...
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Semi-automatic ABC

3 Perform linear regression to get crude estimators given a dataset

$$\begin{aligned}\hat{\phi}_1(X) &= 0.016 + 1.039X^{(1)} + 0.003X^{(2)} + \ldots \\ \hat{\phi}_2(X) &= 1.138 - 0.945X^{(1)} + 0.022X^{(2)} + \ldots \\ \hat{\phi}_3(X) &= 0.324 + 0.003X^{(1)} - 0.001X^{(2)} + \ldots\end{aligned}$$

(If fit very poor add other functions of data as explanatory variables, judging via e.g. BIC)

4 Use $S(X) = (\hat{\phi}_1(X), \hat{\phi}_2(X), \hat{\phi}_3(X))$ in ABC

Beaumont et al. Regression Correction

Beaumont et al. (2002) propose a post-processing step for ABC.

For a single parameter, they model

$$\theta^{(i)} = \alpha + \beta^T [S(\mathbf{x}_{\text{obs}}) - S(\mathbf{x}_{\text{sim}}^{(i)})] + \epsilon_i \quad i = 1, \ldots, S.$$

And estimate α and β through minimising

$$\sum_{i=1}^{S} \left[\theta^{(i)} - \alpha - \beta^T [S(\mathbf{x}_{\text{obs}}) - S(\mathbf{x}_{\text{sim}}^{(i)})]\right]^2 K\left([S(\mathbf{x}_{\text{obs}}) - S(\mathbf{x}_{\text{sim}}^{(i)})]\right)$$

Beaumont et al. Regression Correction

They replace each $\theta^{(i)}$ with

$$\theta^{(i)} - \hat{\beta}^T[S(\mathbf{x}_{\text{obs}}) - S(\mathbf{x}_{\text{sim}}^{(i)})]$$

This can be viewed as trying to remove the bias due to accepting when $S(\mathbf{x}_{\text{obs}}) \neq S(\mathbf{x}_{\text{sim}}^{(i)})$.

Why use ABC?

If we are having to estimate $S_i(\mathbf{y}) = \mathrm{E}(\phi_i|\mathbf{y})$ anyway, why use ABC?

ABC has nice **Optimality properties**

- In the limit as $h \to 0$ the expected loss in estimating ϕ will be lower using ABC posterior means than using $S(\mathbf{y}_{\mathrm{obs}})$ directly.
- ABC will also be better than using any function of $S(\mathbf{y}_{\mathrm{obs}})$.
- Result extends to any function of the parameters.

These give the method some robustness to poor estimates of $S_i(\mathbf{y})$.

A comparison of Indirect Inference and ABC (1)

Indirect Inference and ABC differ in two aspects:

(1) Choice of Summary for the Data: Indirect Inference has a specific approach, the use of ancilliary models, to construct summaries. For ABC no consensus approach to do this.

A comparison of Indirect Inference and ABC (2)

(2) How to then Estimate Parameters

- Indirect Inference gives point estimate and a variance; ABC gives a "posterior" distribution.
- For the same summaries (ignoring Monte Carlo error) ABC is more accurate than Indirect Inference.
- And asymptotically for large data-sets (if there is increasing information about the parameters), ABC and Indirect Inference can be equivalent.
- Computational Cost of Indirect Inference is generally smaller.

References

Andrieu, Doucet and Holenstein (2010). Particle Markov chain Monte Carlo methods. *JRSS series B*, **72**, 269–342.

Andrieu and Roberts (2009) The pseudo-marginal approach for efficient Monte Carlo computations. *Annals of Statistics*, **37**, 697–725.

Beaumont (2003). Estimation of population growth or decline in genetically monitored populations. *Genetics* **164**, 1139-1160.

Beaumong, Zhang and Balding (2002) Approximate Bayesian Computation in Population Genetics *Genetics*, **162**, 2025–2035

Boys, Henderson and Wilkinson (2000) Detecting homogeneous segments in DNA sequences by using hidden Markov models. *Applied Statistics*, **49**, p. 269–285.

Blum and Francois (2010) Non-linear regression models for Approximate Bayesian Computation, *Statistics and Computing*, **20**, 63–73.

Carpenter, Clifford and Fearnhead (1999) An Improved particle filter for nonlinear problems. *IEE-F* **146** , 2–7.

Carvalho, Johannes, Lopes and Polson (2010) Particle Learning and Smoothing. *Statistical Science* **25**, 88–106.

Dean, Singh, Jasra and Peter (2011) Parameter Estimation for Hidden Markov Models with Intractable Likelihoods. *ArXiv:1103.5399*

Del Moral (2004). Feyman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications. *New York: Springer.*

Didelot et al. (2007) A bimodal pattern of relatedness between the *Salmonella* Paratyphi A and Typhi genomes: Convergence or divergence by homologous recombination? *Genome Research*, **17**, p.61–68

Diggle and Gratton (1984) Monte Carlo methods of inference for implicit statistical models. *JRSS B* **46**, 193–227.

Fearnhead (2002) Markov chain Monte Carlo, sufficient statistics, and particle filters. *JCGS* **11**, 848–862.

Fearnhead (2008) Computational Methods for Complex Stochastic Systems: A Review of Some Alternatives to MCMC.*Statistics and Computing*, **18**, p. 151–171.

Fearnhead and Liu (2007) Online Inference for Multiple Changepoint Problems. *JRSS B*, **69**, 589–605.

Fearnhead and Prangle (2012) Constructing summary statistics for approximate Bayesian computation: Semi-automatic ABC *JRSS B*, **74**, 419–474.

Gilks and Berzuini (2001) Following a moving targetMonte Carlo inference for dynamic Bayesian models. *JRSS series B* **63**, 127–146.

Gordon, Salmond and Smith (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE-F* **140**, 107–113.

Gouriéroux and Ronchetti (1993) Indirect Inference. *J. Appl. Econometrics*, **8**, s85–s118.

Ionides, Bhatra, Atchadé and King (2011). Iterated Filtering. *Annals of Statistics*, **39**, 1776–1802.

Killick, Fearnhead and Eckley (2012) Optimal detection of changepoints with a linear computational cost. *ArXiv:1101.1438*

Kong, Liu and Wong (1994) Sequential Imputations and Bayesian Missing Data Problems. *JASA* **89**, 278–288.
Liu (1996) Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing* **6**, 113–119.
Liu and West (1999) Combined parameter and state estimation in simulation-based filtering. *ISDS discussion paper 1999-14*.
Marin, Pudlo, Robert and Ryder (2011) Approximate Bayesian computational methods. *Statistics and Computing*, Online First.
Marjoram, Molitor, Plagnol and Tavaré (2003) Markov chain Monte Carlo without likelihoods. *PNAS* **100** 15324–15328.
Wood (2010) Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, **466**, 1102–1104.
O'Neill, Balding, Becker, Eerola and Mollison (2000). Analyses of infectious disease data from household outbreaks by Markov chain Monte Carlo methods. *Applied Statistics*, **49**, 517–542.
Pitt and Shephard (1999) Filtering via simulation: Auxiliary particle filters. *JASA* **94**, 590–599.
Poyiadis, Doucet and Singh (2011) Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, **98**, 65–80.
Robert, Cornuet, Marin and Pillai (2011) Lack of confidence in approximate Bayesian computation model choice. *PNAS* **108**, 15112–15117.
Scott (2002) Bayesian Methods for Hidden Markov Models, Recursive Computing in the 21st Century. *JASA*, **97**, p. 337–351.
Störvik (2002) Particle filters for state-space models with the presence of unknown static parameters. *IEEE T-SP* **50**, 281–289.
Wilkinson (2011) Parameter inference for stochastic kinetic models of bacterial gene regulation: a Bayesian approach to systems biology. *Bayesian Statistics 9* OUP.

Review Articles on Particle Filters:

Doucet, Godsill and Andrieu (2000) On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* **10** 197–208.
Kantas , Doucet, Singh and Maciejowski (2009) An overview of sequential Monte Carlo methods for parameter estimation in general state-space models. *Proceedings of the IFAC*
Liu and Chen (1998) Sequential Monte Carlo Methods for Dynamic Systems. *JASA* **443** 1032–1044.