Parameter elimination in particle Gibbs sampling

Anna Wigren (UU), Riccardo Sven Risuleo (UU), Lawrence Murray (Uber AI), Fredrik Lindsten (LiU)

UPPSALA UNIVERSITET

LINKÖPING UNIVERSITY

Uber AI

The paper in 30 seconds

The performance of particle Gibbs samplers is limited by the underlying Gibbs sampler that is approximated. We can improve on this by eliminating the parameters from the state update using marginalization.

Marginalization is possible for models with conjugacy between the parameter prior and the complete data likelihood. It yields a non-Markovian state update, but can still be done in linear time.

Marginalizing by hand is cumbersome, but can be automated using probabilistic programming.

Take with you

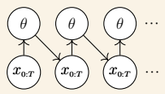
- Particle Gibbs is limited by Gibbs
- We can improve using marginalization
- Automatic marginalization is possible through probabilitic programming

Gibbs sampling

We aim to infer the joint posterior $p(x_{0:T}, \theta | y_{1:T})$ of a state-space model.

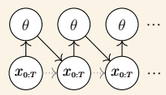
Not analytically tractable in general, instead we use **Gibbs sampling**.

$$x^*_{0:T} \sim p(x_{0:T}|\theta', y_{1:T})$$
$$\theta^* \sim p(\theta|x^*_{0:T}, y_{1:T})$$



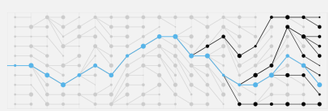
Sampling the high-dimensional state trajectory is challenging. **Particle Gibbs** uses conditional sequential Monte Carlo (cSMC) to do this.

$$x^*_{0:T} \sim \text{cSMC}(x_{0:T}|x'_{0:T}; \theta', y_{1:T})$$
$$\theta^* \sim p(\theta|x^*_{0:T}, y_{1:T})$$



Conditional sequential Monte Carlo: Represents the target $p(x_{0:T}|\theta', y_{1:T})$ as N particles $\{x^i_{0:T}, w^i_T\}_{i=1}^N$ generated from iterating, for $t = 1, \dots, T$:

- Resample a^i_t using w^i_{t-1}, set $a^N_t = N$
- Draw $x^i_t \sim q(x_t|x^{a^i_t}_{0:t-1})$, set $x^N_t = x'_t$
- Update $w^i_t \propto \frac{p(x^i_t, y_t|x^{a^i_t}_{t-1}, \theta)}{q(x^i_t|x^{a^i_t}_{0:t-1})}$

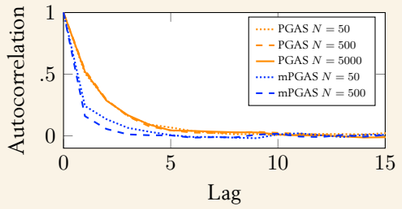


Motivation - why do we want to eliminate the parameters?

In Gibbs sampling there is a dependence between $x_{0:T}$ and θ which leads to correlated samples.

Particle Gibbs is an **exact approximation** of the Gibbs sampler and can never do better than the sampler it approximates.

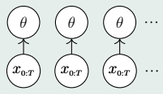
By **marginalizing** out the parameters from the state update we can improve on this!



When can we marginalize?

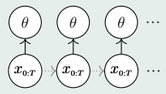
Marginalized Gibbs sampling amounts to

$$x^*_{0:T} \sim p(x_{0:T}|y_{1:T})$$
$$\theta^* \sim p(\theta|x^*_{0:T}, y_{1:T})$$



Marginalized cSMC in the state update yields

$$x^*_{0:T} \sim \text{mcSMC}(x_{0:T}|x'_{0:T}; y_{1:T})$$
$$\theta^* \sim p(\theta|x^*_{0:T}, y_{1:T})$$



Problem: State trajectory update is **non-Markovian**, in particular $w^i_t \propto \frac{p(x^i_t, y_t|x^{a^i_t}_{1:t-1}, y_{1:t-1})}{q(x^i_t|x^{a^i_t}_{0:t-1})}$

Solution: Possible if we have **conjugacy** between parameter prior and complete data likelihood!

Restricted exponential family:

Compete data likelihood:

$$p(x_t, y_t|x_{t-1}, \theta) = h_t \exp\left(\theta^\mathsf{T} s_t - A^\mathsf{T}(\theta) r_t\right)$$

Prior:

$$\pi(\theta|\chi_0, \nu_0) = g(\chi_0, \nu_0) \exp\left(\theta^\mathsf{T} \chi_0 - A^\mathsf{T}(\theta)\nu_0\right)$$

$$\Longrightarrow$$

$$p(x_t, y_t|x_{0:t-1}, y_{1:t-1}) = \frac{g(\chi_{t-1}, \nu_{t-1})}{g(\chi_t, \nu_t)} h_t$$

$$\chi_t = \chi_0 + \sum_{k=1}^{t} s_k = \chi_{t-1} + s_t$$

$$\nu_t = \nu_0 + \sum_{k=1}^{t} r_k = \nu_{t-1} + r_t$$

For the restricted exponential family marginalized cSMC amounts to iterating

- Update hyperparameters χ^i_t, ν^i_t
- Resample a^i_t using w^i_{t-1}, set $a^N_t = N$
- Draw $x^i_t \sim q(x_t|x^{a^i_t}_{0:t-1})$, set $x^N_t = x'_t$
- Update $w^i_t \propto \frac{g(\chi^{a^i_t}_{t-1}, \nu^{a^i_t}_{t-1}) h^i_t}{g(\chi^i_t, \nu^i_t) q(x^i_t|x^{a^i_t}_{0:t-1})}$

We can run this in **linear time**!

Extension: Particle Gibbs with ancestor sampling (PGAS)

Partice Gibbs is known to mix slowly. To improve we can use marginalized **ancestor sampling**, where the ancestor for the reference trajectory is sampled according to $\tilde{w}^i_{t-1|T} = \bar{w}^i_{t-1} \frac{p([x^i_{0:t-1}, x'_{t:T}], y_{1:T})}{p(x^i_{0:t-1}, y_{1:t-1})}$

For the restricted exponential family we get $\tilde{w}^i_{t-1|T} \propto \bar{w}^i_{t-1} \frac{g(\chi^i_{t-1}, \nu^i_{t-1})}{g(\chi^i_T, \nu^i_T)} h^i_t$
with $\chi^i_T = \chi^i_{t-1} + s_t(x'_t, x^i_{t-1}, y_t) + s'_{t+1:T}$ and $\nu^i_T = \nu^i_{t-1} + r_t(x^i_{t-1}) + r'_{t+1:T}$

Additional steps in marginalized cSMC with ancestor sampling are

- Update statistics $s'_{t+1:T}$, $r'_{t+1:T}$
- Update hyperparameters χ^i_T, ν^i_T
- Sample a^N_t from $\tilde{w}^i_{t-1|T}$

We can run this in **linear time**!

Contact:

anna.wigren@it.uu.se
riccardo.risuleo@it.uu.se
lawrence.murray@uber.com
fredrik.lindsten@liu.se

Links:

Webpage Birch: birch-lang.org/

Probabilistic programming and Birch

Marginalization by hand is time-consuming, we do it automatically using probabilistic programming!

What is probabilistic programming?

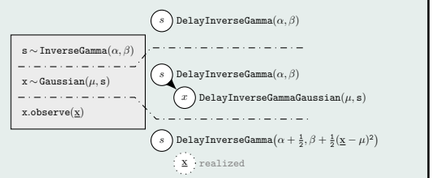
- Tool for encoding probabilistic models and performing inference in them.
- Separates the model and the inference algorithm.
- Written in a probabilistic programming language.

```
β:Real <- 0.9;
σ2:Real <- 1.0;
T:Integer <- 10;
x:Random<Real>[T];
y:Random<Real>[T];

x[1] ~ Gaussian(0.0, σ2);
y[1] ~ Gaussian(x[1], σ2);
for (t:Integer in 2..T) {
  x[t] ~ Gaussian(β*x[t - 1], σ2);
  y[t] ~ Gaussian(x[t], σ2); }
```

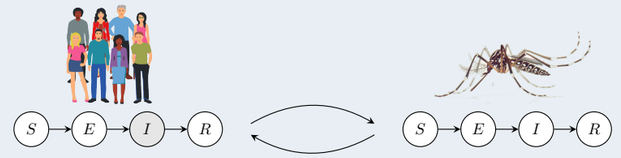
Birch:

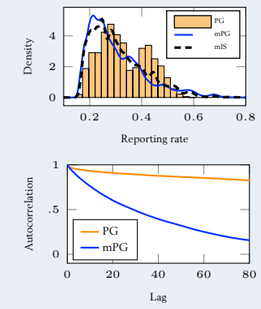
- Object-oriented probabilistic programming language
- Uses inference methods based on SMC
- Employs **delayed sampling** to find and use conjugacy relations, enables **automatic marginalization**!



Example: Dengue fever dataset

The dataset contains the number of newly infected people during an outbreak of dengue fever on the island Yap in Micronesia.





Transitions between compartments are binomial, with beta priors on the parameters → **conjugacy**!

Extension: Partial marginalization

For models where only some parameters are conjugate, split them into θ_m (conjugate) and θ_u (not conjugate).

Marginalize out θ_m and sample θ_u using any MCMC method.

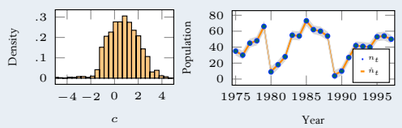
We use marginalized PMMH in Birch.

Models where some parameters can be marginalized out are common in practice.

Example: Song sparrow dataset

The dataset contains the number of observed song sparrows on an island during 20 years.

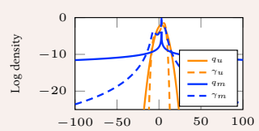
$$\log n_{t+1} = \log n_t + \begin{bmatrix} 1 & (n_t)^{\mathbf{c}} \end{bmatrix} \boldsymbol{b} + \boldsymbol{\sigma}_v v_t$$
$$y_t = n_t + \boldsymbol{\sigma}_w w_t$$



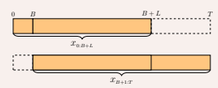
Extension: Diffuse priors and blocking

A diffuse prior on the parameters leads to a diffuse prior on the states when marginalizing.

This can lead to poor mixing and numerical problems for the first time steps.



We propose to separate the state trajectory into two overlapping blocks and do Gibbs updates of each block separately.



- Sample $x_{0:B+L}$ using standard cSMC
- Sample $x_{B+1:T}$ using marginalized cSMC
- Sample $\theta \sim p(\theta|x_{0:T}, y_{1:T})$