Muestreo de modelos PyMC con Sequential Monte Carlo en BlackJax

Carlos Iguaran Osvaldo A. Martin1

[1]IMASL-CONICET. Universidad Nacional de San Luis. San Luis, Argentina

Objetivos

El objetivo del presente trabajo es mejorar la integración entre PyMC[6] y Blackjax[5], permitiendo el muestreo de modelos definidos en el primero, utilizando algoritmos de Sequential Monte Carlo implementados en el segundo. Además busca implementar optimizaciones propuestas en la literatura[2] para HMC en BlackJax, realizando luego una comparación experimental. Todo esto debería disponibilizar código SMC performante, extensible e integrado para que la comunidad científica pueda aplicarlo a sus problemas particulares.

Sequential Monte Carlo

1. Inicializar β en 0.
2. Generar S muestras s_β de la posterior temperada $p(\boldsymbol{\theta} \mid \boldsymbol{y})_\beta \propto p(\boldsymbol{y} \mid \boldsymbol{\theta})^\beta \, p(\boldsymbol{\theta})$
3. Incrementar β de manera de mantener el effective sampling size ESS_{IS} en un valor predefinido.
4. Calcular S pesos de importancia W. Los pesos tienen en cuenta la nueva y vieja posterior temperada.
5. Obtener s_w via re-muestrear s_β de acuerdo a W.
6. Correr S cadenas MCMC, empezando cada una en una en una partícula diferente de s_w.
7. Repetir desde paso 3 hasta llegar a $\beta = 1$
8. Ajustar parámetros de las cadenas de MCMC a partir de la población de partículas.

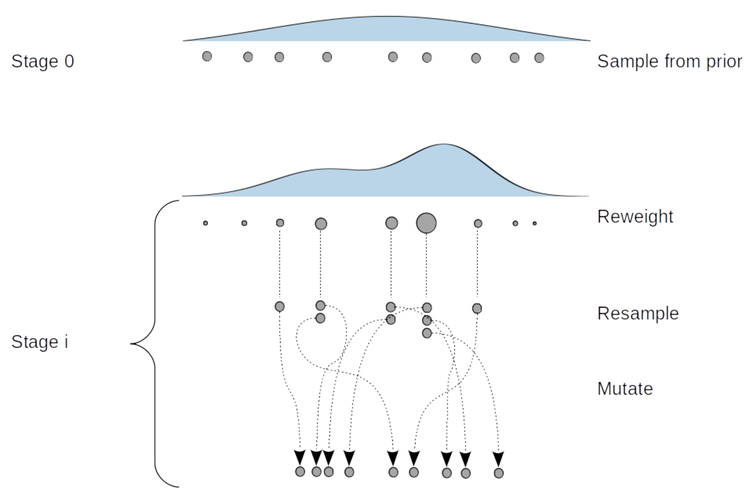

Figura 1. Iteraciones en Sequential Monte Carlo

Ventajas y desventajas

Ventajas

- Mejor muestreo que MCMC en posterior multimodales, al menos en dimensiones no muy altas.
- Fácil paralelización
- Cálculo de marginal likelihood incorporado en el algoritmo.
- Mejor escalabilidad que MH, para modelos para los cuales no podemos calcular gradientes (funciones blackbox, ABC, etc).

Desventajas

- SMC subdesarrollado teóricamente y empíricamente respecto de MCMC
- Falta de métodos de diagnóstico
- Falta de software fácil de usar

BlackJax

1. Escrita sobre Jax[1]
2. Compilada Just-in-time
3. Permite correr en GPU o CPU sólo cambiando un parámetro
4. Diseño basado en programación funcional
5. Diseño pensado para extensibilidad via composición.
6. Contaba con una implementación base de SMC al comienzo de este trabajo.

Sequential Monte Carlo es intensivo en recursos computacionales y altamente paralelizable. BlackJax permite compilación JIT y ejecución paralela, posibilitando la adopción de estos métodos.

Optimizaciones

Podemos ajustar los parámetros de las cadenas MCMC a partir de la población de partículas o métricas del muestreo en el paso anterior (por ejemplo, el ESS).

1. Hamiltonian Monte Carlo (HMC): **inverse mass matrix** a partir de la población de partículas [2].
2. Independent Metropolis Hastings (IRMH) con proposal gaussiana: **media y matriz de covarianza** (completa o diagonal) a partir de la población de partículas.
3. Escala de cualquiera de las matrices anteriores, basada en la tasa de aceptación de cada paso.
4. Nuevas implementaciones que el usuario quiera probar gracias a **código extensible**

```
smc_with_inner_kernel_tuning(
        logprior_fn=model_logprior,
        loglikelihood_fn=model_loglikelihood,
        mcmc_factory=kernels.hmc,
        ...
        mcmc_parameters=dict(step_size=1e-4,
                             num_integration_steps=30),
        mcmc_parameter_factory=lambda state, info: jnp.array(mass_matrix_from_particles(state.particles)),
        initial_parameter_value=jnp.eye(30),
    )
```

Figura 2. Ejemplo de optimización de la inverse mass matrix de Hamiltonian Monte Carlo

Al integrar con PyMC, las funciones logprior y loglikelihood se calcularán automáticamente, simplificando todavía más probar nuevas alternativas sobre distintos modelos. Además, la sintaxis de PyMC es muy amigable incluso para la construcción de modelos complejos.

Preguntas

- ¿Puede la versión optimizada dar una aproximación de la posterior de igual o mejor calidad que la no optimizada?
- ¿Es la versión optimizada mas rápida que la no optimizada?
- ¿Como se comparan HMC e IRMH como kernels de SMC?
- ¿Como se relacionan los parámetros fijos (por ejemplo *step_size* en HMC) con aquellos que se adaptan (*inverse mass matrix* en HMC).
- ¿Se pueden dar recomendaciones sobre cuando y como utilizar estas optimizaciones?

Benchmarks

1. Potencial bimodal n-variado

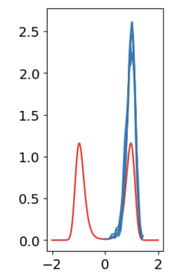

Figura 3. Posterior muestreada con NUTS (azul) y verdadera posterior (rojo). El modelo es 5-variado siendo todas las marginales iguales.

2. Radon centrado [3, 4]
3. Radon descentrado[3, 4]

Agradecimientos

Este trabajo es posible gracias a una Small Development Grant de NUMFOCUS.

Referencias

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang.
JAX: composable transformations of Python+NumPy programs, 2018.

[2] Alexander Buchholz, Nicolas Chopin, and Pierre E. Jacob.
Adaptive tuning of hamiltonian monte carlo within sequential monte carlo, 2020.

[3] Alex Andorra Oriol Abril Chris Fonnesbeck, Colin Carroll and Farhan Reynaldo.
A primer on bayesian methods for multilevel modeling.
In PyMC Team, editor, *PyMC examples*.

[4] Andrew Gelman and Jennifer Hill.
Data Analysis Using Regression and Multilevel/Hierarchical Models.
Analytical Methods for Social Research. Cambridge University Press, 2006.

[5] Junpeng Lao and Rémi Louf.
Blackjax: A sampling library for JAX, 2020.

[6] Colin Carroll Larry Dong Christopher J Fonnesbeck Maxim Kochurov Ravin Kumar4 Junpeng Lao Christian C. Luhmann Osvaldo A Martin Michael Osthege Ricardo Vieira Thomas Wiecki Oriol Abril-Pla, Virgile Andreani and Robert Zinkov.
PyMC: A modern, and comprehensive probabilistic programming framework in Python.
PeerJ Computer Science 2023 (in press). 2023.

Contacto