Stochastic Simulations

Autumn Semester 2018

Prof. Fabio Nobile *Assistant: Dr. Panagiotis Tsilifis, Juan Pablo Madrigal Cianci*

Mini-project

submission deadline: 9 January 2019

Bayesian inverse problems in large dimensions

Consider the problem of finding a set of parameters $\xi \in \mathbb{R}^P$ from some measured data $y \in \mathbb{R}^J$, such that the following relation holds:

$$y = G(\xi) + \eta. \tag{1}$$

In the previous equation G is a given "observation operator" (i.e a smooth, non-necessarily linear, map from ξ to y), and η is some additive random noise polluting the model. For instance, G may involve some differential equation and return the solution observed at certain locations in space and time, whereas ξ may represent some parameter in the equation we want to infer from observed data. For simplicity we will assume that the noise is Gaussian of the form $\eta \sim N(0, C_\eta)$. Given the random nature of the noise, and the fact that in general $P \neq J$, it is not possible to simply invert G in order to obtain ξ. Thinking probabilistically enables us to overcome this difficulty. Treating y, ξ, and η as random variables, we can try estimating the probability distribution of ξ given y, $\pi(\xi|y)$, which, in light of Bayes theorem, can be written as

$$\pi(\xi|y) = \frac{\pi(y|\xi)\pi_0(\xi)}{\pi(y)}. \tag{2}$$

In the previous equation, $\pi(\xi|y)$ is called the *posterior distribution*, $\pi(y|\xi)$ is called the *likelihood*, $\pi_0(\xi)$ is called the *prior* and $\pi(y)$ is called the evidence. In the particular case of model (1) with η an additive Gaussian noise, we can write the likelihood as:

$$\pi(y|\xi) = N(G(\xi), C_\eta). \tag{3}$$

Moreover, the Bayesian approach allows us to include *a priori* information on ξ into the prior distribution $\pi_0(\xi)$. Thus, in order to obtain samples from the posterior distribution, we can try devising a strategy to sample directly from the right hand side of (2) instead. In most cases however, the evidence term $\pi(y)$ is not known, and thus we need to resort to Markov Chain Monte Carlo methods, as we can only evaluate the un-normalized posterior $\tilde{\pi}(y|\xi) = \pi(y|\xi)\pi_0(\xi)$ point-wise.

Bayesian inference for log permeability

For this project we will try to infer ξ from the following model:

$$y_j = p(0.2j; \xi) + \eta_j, \quad j = 1, \ldots, 4, \quad \eta_j \sim N(0, \sigma^2), \tag{4}$$

where p is the solution to the following elliptic PDE:

$$\frac{d}{dx}\left(e^{u(\xi)}\frac{d}{dx}p(x;\xi)\right) = 0, \quad p(0;\xi) = 0, \quad p(1;\xi) = 2, \quad x \in [0,1], \tag{5}$$

$$u(x,\xi) = \frac{\sqrt{2}}{\pi}\sum_{k=1}^{P}\xi_k \sin(k\pi x), \tag{6}$$

This equation (albeit in 2 and 3 spatial dimensions) is used for modeling subsurface flows. A typical real-world application is to infer the log-permeability $u(x,\xi)$ given some measurements of the pressure p, or to estimate the expected value of some output quantity under the posterior distribution. The model (6) can be thought of as a sine-series expansion of the log-permeability, thus recasting the inference problem on the Fourier coefficients $\xi = (\xi_1, \ldots, \xi_P)$. For computational purposes, the series needs to be truncated at the P-th term. The goal of this project is to understand the influence of the truncation level P in the performance of MCMC algorithms, particularly when $P \to \infty$. We will model the likelihood as

$$\pi(y|\xi) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{|y - G(\xi)|^2}{2\sigma^2}\right), \tag{7}$$

where $|\cdot|$ is the Euclidean norm,$G(\xi) = (p(0.2;\xi), \ldots, p(0.8;\xi))$ and the prior as

$$\pi_0 = N(0,C),\ C := \operatorname{diag}\{k^{-2}\ \ k \in \mathbb{N}\},\ \text{ i.e, } \pi_0(\xi_1,\ldots,\xi_P) = \frac{1}{\prod_{k=1}^{P}\sqrt{2\pi k^{-2}}}\exp\left(-\sum_{k=1}^{P}\frac{\xi_k^2 k^2}{2}\right). \tag{8}$$

In this project we will implement different types of proposals and study their efficiency for the problem at hand. We will use the effective sample size (ESS) as a metric for comparison. Recall that for a scalar function $\xi \to f(\xi) \in \mathbb{R}$ and a stationary process $\{\xi_n\}_{n\in\mathbb{Z}}$, the effective sample size is given by

$$\text{ESS} = \text{ESS}(N, f, \{\xi_n\}_{n=0}^{N}) = N\left[1 + 2\sum_{n=0}\text{Corr}(f(\xi_0), f(\xi_n))\right]^{-1}, \tag{9}$$

where N is the number of samples taken in the Markov chain.

Remarks on implementation

i. Equation (5) has a closed form solution given by

$$p(x) = 2\frac{S_x(e^{-u})}{S_1(e^{-u})}, \quad \text{where } S_x(f) = \int_0^x f(y)dy. \tag{10}$$

To implement it, discretize the interval $[0,1]$ into M sub-intervals of size h. The integration can be then computed using scipy's `scipy.integrate.trapz` or `scipy.integrate.cumtrapz`. For a given chosen spatial resolution it makes sense to truncate the series expansion in (6) to $P = 2M$. This implies that the higher the spatial resolution, the higher will be the size of the vector ξ, i.e, the dimension of the state space.

Data y is obtained and it is given in Table 1. Moreover, this data is assumed to be polluted by an additive Gaussian noise $\eta \sim N(0, \sigma^2 I)$, with $\sigma = 0.04$.

x	0.2	0.4	0.6	0.8
$y(x;\xi)$	0.5041	0.8505	1.2257	1.4113

Table 1: Measured data.

Goals of the project

Random walk Metropolis

Implement the random walk Metropolis (RWM) algorithm using proposals of the form $Q(\xi, \cdot) = \mathcal{N}(\xi, s^2 C)$ for different values of $s < 1$. Run a chain of length $N = 10^4$ samples for different values of M, and $P = 2M$. Compare your results in terms of mixing of the chains and ESS for the following functions $f_1 = \xi_1$, $f_2 = \xi_2$, $f_3 = \xi_{10}$ and $f_4 = q(\xi)$ as in (15). Include a plot of the autocorrelation for each value of P. Explain your results.

Improving on RWM: preconditioned Crank-Nicholson (pCN)

An improvement over the standard random walk metropolis algorithm in large dimensions is the preconditioned Crank-Nicholson (pCN). In this case, the proposal distribution is

$$Q(\xi, \cdot) = N(\sqrt{1 - s^2}\xi, s^2 C), \quad C := \mathrm{diag}\{k^{-2} \quad k \in \mathbb{N}\}, \tag{11}$$

for some $s < 1$. Implement a metropolis-hastings MCMC algorithm using this proposal. Repeat the experiments in the previous point and compare the performances of pCN and RWM.

Laplace's approximation

Another idea is to use Laplace's approximation, i.e, to set the proposal distribution Q to be a normal $N(\xi_{\text{map}}, H)$, where ξ_{map} is the maximum a posteriori point, i.e,

$$\xi_{\text{map}} = \arg\max_{\xi \in \mathbb{R}^P} \left(-\log \pi(\xi|y)\right).$$

and H is the Hessian of $-\log \pi(\xi|y)$ evaluated at ξ_{map}. An approximation of these quantities can be obtained by standard python optimization libraries (see BFGS on the Scipy optimization reference). Notice that the BFGS algorithm provides a low-rank approximation $\tilde{H}$ of the Hessian H. In practice, we set $Q = \mathcal{N}(\xi_{\text{map}}, \tilde{H} + \alpha^2 I)$, for some $\alpha > 0$. Once such a proposal is constructed, an independent sampler Metropolis algorithm can be implemented. Run the same experiments as before and compare your results.

Combining Laplace's approximation and pCN: Generalized pCN

One last improvement is to generate proposals of the form $Q(\xi, \cdot) = N(A_\Gamma \xi, s^2 C_\Gamma)$, with

$$A_\Gamma = C^{1/2}\sqrt{I - s^2 + (I + H_\Gamma)^{-1}}C^{-1/2}, \tag{12}$$

$$C_\Gamma = \left(C^{-1} + \Gamma\right)^{-1} \tag{13}$$

$$H_\Gamma = C^{1/2}\Gamma C^{1/2}, \quad \Gamma = \sigma^{-2}(\nabla G(u(\xi_{\text{map}})))(\nabla G(u(\xi_{\text{map}})))^T. \tag{14}$$

Implement this method and compare with previous results in term of ESS vs dimensionality. Explain your results. Lastly, use the implemented methods to obtain a MCMC estimator of the posterior expectation of the following scalar quantity of interest

$$q(\xi) = \int_0^1 e^{u(x,\xi)} dx. \tag{15}$$

References

1. Rudolf, Daniel, and Björn Sprungk. "On a generalization of the preconditioned Crank-Nicolson Metropolis algorithm." Foundations of Computational Mathematics 18.2 (2018): 309-343.

2. Cotter, Simon L., et al. "MCMC methods for functions: modifying old algorithms to make them faster." Statistical Science (2013): 424-446.