BAYESIAN STATISTICAL MODELS

Chapter 2 - Computational methods

Lecturer: Riccardo Corradin University of Milano-Bicocca

1 SAMPLING FROM POSTERIOR DISTRIBUTIONS

In an ideal world, once we have a posterior distribution everything is analytically tractable. In a more realistic world, even if we do not have analytical tractability, we can easily sample from the posterior distribution of interest, and use that sample for inferential purposes. In the real world, we cannot. We need tailored strategies to produce a sample even when the posterior distribution is hardly tractable. There are many strategies to work with hardly tractable distributions. Among these, in the module we will consider three alternatives:

- Metropolis-Hastings;
- Gibbs sampler;
- Hamiltonian Monte Carlo.

These strategies can be used to produce a Markov chain whose ergodic distribution is the posterior distribution of interest. The idea is to produce a sample $\{\boldsymbol{\theta}^{(r)}\}_{r=1}^{R}$ that mimics the posterior distribution behavior, and instead of looking at the posterior distribution summaries, we consider summaries of such a sample. For example, if we aim to produce a point estimate of our parameter of interest $\boldsymbol{\theta}$,

$$\mathbb{E}[\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}] \qquad \text{can be approximated by} \qquad \frac{1}{R}\sum_{r=1}^{R}\boldsymbol{\theta}^{(r)},$$

with $\{\boldsymbol{\theta}^{(r)}\}_{r=1}^{R}$ being a suitable sequence of values sampled from $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$, or a suitable approximation.

1.1 METROPOLIS-HASTINGS

One of the early approaches to produce a Markov chain with a specific distribution as ergodic is the Metropolis-Hastings. The underlying idea is to generate a chain with Markovian dependence $\{\boldsymbol{\theta}^{(t)}\}_{t\geq 1}$ such that is behaving like the target distribution. The Markovian dependencies helps with tractability, since at a given time, we can sample a value only knowing the previous state. The ergodicity allows to use the sequence for inferential procedure, as far as we sample for enough time.

Let $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$ be the distribution of interest that we want to sample from. The core of Metropolis-Hastings is an auxiliary proposal (instrumental) distribution $q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta})$, such that the support of $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$ is a subset of the proposal support, or equivalently the proposal covers the target distribution, i.e. $\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}) \ll q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta})$. Hence, all suitable values for the posterior distribution can be reached by the proposal.

Different proposals lead to specific versions of the Metropolis-Hastings algorithm. The main families of proposal distributions are the followings.

- $q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}) = q(\boldsymbol{\theta}^N)$, independent proposal. We ignore the previous value and we propose independently the current one.
- $q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}) = q(\boldsymbol{\theta}^N - \boldsymbol{\theta})$, random walk. The previous value affect the location of the proposal.

Suppose we propose a value $\boldsymbol{\theta}^{(N)} \sim q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta})$. A first version of the algorithm accept the proposed value with probability

$$\alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})}\right\}.$$

This is the so-called Metropolis algorithm. Although it was a first attempt to produce a chain sampling from a target distribution, ergodicity cannot be guaranteed. Later, an extension introduced an additional term to guarantee ergodicity, resulting in the celebrated Metropolis-Hastings algorithm, with acceptance rate given by

$$\alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^N)}{\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta})}\right\}.$$

Hence, the latter acceptance rate produce a chain with ergodic distribution $\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n})$. To give an interpretation of the acceptance rate, $\alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta})$ is the probability to move from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^N$ while $[1 - \alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta})]$ is the probability to stay on $\boldsymbol{\theta}$.

The algorithm works not only for posterior distribution, but for general probability distributions that satisfy its assumptions (e.g. we can sample from the prior as well). The algorithm works also up to a normalization constant. For example, let $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}) = \mathrm{C} \times \pi^*(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$. Then is easy to see that

$$\alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^N)}{\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta})}\right\} = \min\left\{1, \frac{\cancel{\mathrm{C}}\pi^*(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^N)}{\cancel{\mathrm{C}}\pi^*(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta})}\right\}.$$

An sketch of implementation is the following.

Algorithm 1: Pseudocode for the Metropolis-Hastings algorithm, sample of size T.

1: Set initial values for $\boldsymbol{\theta}^{(0)}$.
2: **for** $t = 1$ to T **do**
3: Sample $\boldsymbol{\theta}^N \sim q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^{(t-1)})$.
4: Set $\alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}^{(t-1)}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^N)}{\pi(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{y}_{1:n}) q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^{(t-1)})}\right\}$.
5: Sample $U \sim Unif(0, 1)$.
6: If $U < \alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}^{(t-1)})$, set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^N$. Else, set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$.
7: **end for**

Recall that a Markov chain satisfies the detailed balance condition if there exists a function $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$ such that

$$\mathcal{K}(\boldsymbol{\theta}^N, \boldsymbol{\theta}^{(t-1)}) \pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}) = \mathcal{K}(\boldsymbol{\theta}^{(t-1)}, \boldsymbol{\theta}^N) \pi(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{y}_{1:n}),$$

where $\mathcal{K}(\boldsymbol{\theta}^N, \boldsymbol{\theta}^{(t-1)})$ is the transition kernel, and that such condition implies (i) $\pi(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{y}_{1:n})$ is the invariant distribution of the chain and (ii) the chain is reversible. Then, it is possible to prove the following result.

Theorem 1.1. *Let $\{\boldsymbol{\theta}^{(t)}\}_{t\geq 1}$ be the chain produced by a Metropolis-Hastings algorithm, such that the support of the target distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$ is covered by the support of the proposal $q(\cdot \mid \boldsymbol{\theta}^{(t-1)})$. Then,*

(a) the kernel of the chain satisfies the detailed balance condition with $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$;

(b) $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$ is the stationary distribution of the chain.

Note that (a) $\Rightarrow$ (b), hence we just need to prove that (a) holds true.

Proof. Suppose we are proposing $\boldsymbol{\theta}^N$, given the previous value $\boldsymbol{\theta}^O$. We have to show that

$$\mathcal{K}(\boldsymbol{\theta}^N, \boldsymbol{\theta}^O)\pi(\boldsymbol{\theta}^O \mid \boldsymbol{y}_{1:n}) = \mathcal{K}(\boldsymbol{\theta}^O, \boldsymbol{\theta}^N)\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n}).$$

The transition kernel can be expanded as

$$\mathcal{K}(\boldsymbol{\theta}^N, \boldsymbol{\theta}^O) = \alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}^O)q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^O) + \big[1 - r(\boldsymbol{\theta}^O)\big]\delta_{\boldsymbol{\theta}^O}(\boldsymbol{\theta}^N),$$

where

$$r(\boldsymbol{\theta}^O) = \int \alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}^O)q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^O)\mathrm{d}\boldsymbol{\theta}^N.$$

Note that the transition kernel is the summation of two distinct terms, hence we can study them separately. For the first part, we have the following identity

$$\begin{aligned}
&\alpha(\boldsymbol{\theta}^N,\boldsymbol{\theta}^O)q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^O)\pi(\boldsymbol{\theta}^O \mid \boldsymbol{y}_{1:n}) \\
&\quad = \frac{\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n})q(\boldsymbol{\theta}^O \mid \boldsymbol{\theta}^N)}{\pi(\boldsymbol{\theta}^O \mid \boldsymbol{y}_{1:n})q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^O)}q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^O)\pi(\boldsymbol{\theta}^O \mid \boldsymbol{y}_{1:n}) \\
&\quad = \pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n})q(\boldsymbol{\theta}^O \mid \boldsymbol{\theta}^N) \\
&\quad = \alpha(\boldsymbol{\theta}^O, \boldsymbol{\theta}^N)\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n})q(\boldsymbol{\theta}^O \mid \boldsymbol{\theta}^N),
\end{aligned}$$

where the latter identity holds since if $\alpha(\boldsymbol{\theta}^N, \boldsymbol{\theta}^O) < 1$ then $\alpha(\boldsymbol{\theta}^O, \boldsymbol{\theta}^N) = 1$. Similarly, for the second part we have

$$\begin{aligned}
&\big[1-r(\boldsymbol{\theta}^O)\big]\delta_{\boldsymbol{\theta}^O}(\boldsymbol{\theta}^N)\pi(\boldsymbol{\theta}^O \mid \boldsymbol{y}_{1:n}) \\
&\qquad = \big[1 - r(\boldsymbol{\theta}^N)\big]\delta_{\boldsymbol{\theta}^N}(\boldsymbol{\theta}^O)\pi(\boldsymbol{\theta}^N \mid \boldsymbol{y}_{1:n})
\end{aligned}$$

The two previous part together lead to the transition kernel detailed balance condition.

□

Exercise 1.2. Let us consider a sample from a Cauchy distribution,

$$Y_1, \ldots, Y_n \mid \mu, \lambda \sim Cauchy(\mu, \lambda),$$

where

$$f(y) = \frac{e^{-\lambda}}{\pi(1 + (y-\mu)^2 e^{-2\lambda})}, \qquad \mu \in \mathbb{R}, \lambda \in \mathbb{R}^+.$$

Note that for the Cauchy distribution $\mathbb{E}[Y]$ and $var(Y)$ do not exist, and μ corresponds to mode and median of the distribution.

- Write explicitly the target distribution of interest.
- Define a suitable Metropolis-Hastings strategy using a random walk with Gaussian proposal.

1.2 GIBBS SAMPLER

Quite commonly, even if we are unable to sample directly a distribution for the whole parameter $\boldsymbol{\theta}$, we can express one (or more) of its dimension conditioning on the others. Intuitively, this is the idea beyond Gibbs samplers: if we have tractability of one (or more) dimension, we should use it. Suppose, for simplicity, that we can express our parameter of interest as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$. If the conditional distributions

$$\pi(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_2, \boldsymbol{y}_{1:n}),$$
$$\pi(\boldsymbol{\theta}_2 \mid \boldsymbol{\theta}_1, \boldsymbol{y}_{1:n}),$$

can be sampled easily, then we can iteratively update both subsets of parameters. The previous concept can be extended to $d > 2$ subsets. An idea of implementation is the following.

Algorithm 2: Pseudocode for the Metropolis-Hastings algorithm, sample of size T.

```
Set initial values for θ_1^(0), ..., θ_d^(0).
for t = 1 to T do
    for j = 1 to d do
        Sample θ_j^(r) ~ π(θ_j^(r) | θ_1^(r), ..., θ_{j-1}^(r), θ_{j+1}^(r-1), ..., θ_d^(r-1) y_{1:n})
    end for
end for
```

Note that this is a quite common scenarios, since working with specific distributional choices lead to models that a posteriori factorize in tractable terms. In this case, the sampling can be done efficiently and we do not have to worry about ergodicity and convergence, up to running the algorithm enough to mitigate the effect of initial condition choices.

The Gibbs sampler is a special case of Metropolis-Hastings, where we accept all the proposed values. Note that if we propose a move from $\boldsymbol{\theta}_1^{(r-1)}$ to $\boldsymbol{\theta}_1^N$, then we have that the proposal distribution is $\pi(\boldsymbol{\theta}_1^N \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})$. The corresponding acceptance rate is then

$$\begin{aligned}\alpha(\boldsymbol{\theta}_1^N, \boldsymbol{\theta}_1^{(r-1)}) &= \frac{\pi(\boldsymbol{\theta}_1^N, \boldsymbol{\theta}_2^{(r-1)} \mid \boldsymbol{y}_{1:n})\pi(\boldsymbol{\theta}_1^{(r-1)} \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})}{\pi(\boldsymbol{\theta}_1^{(r-1)}, \boldsymbol{\theta}_2^{(r-1)} \mid \boldsymbol{y}_{1:n})\pi(\boldsymbol{\theta}_1^N \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})} \\ &= \frac{\pi(\boldsymbol{\theta}_1^N \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})\pi(\boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})\pi(\boldsymbol{\theta}_1^{(r-1)} \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})}{\pi(\boldsymbol{\theta}_1^{(r-1)} \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})\pi(\boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})\pi(\boldsymbol{\theta}_1^N \mid \boldsymbol{\theta}_2^{(r-1)}, \boldsymbol{y}_{1:n})} = 1.\end{aligned}$$

All the properties of Metropolis-Hastings hold, for example that the invariant distribution of the chain is the posterior distribution of interest.

Some particular cases of the Gibbs sampler can be achieved by specific choices of the parameter subsets.

- **Full Gibbs sampler**, where we have d dimension and we express each dimension conditionally on the others, e.g.
$$\theta_j \mid \theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_d.$$
Hence, we are looking at the full conditional distribution of each parameter separately.

- **Blocked Gibbs sampler**, we express some of the dimension conditioned on the others, e.g.

$$\boldsymbol{\theta}_j \mid \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{j-1}, \boldsymbol{\theta}_{j+1}, \dots, \boldsymbol{\theta}_d.$$

  We are looking at the conditional distribution jointly for a subset of parameters.

- **Collapsed Gibbs sampler**, we marginalize some of the dimensions, e.g.

$$\theta_1 \mid \theta_3, \dots, \theta_d.$$

  We marginalize the distribution of a subset of parameters (here θ_2), which may results in having a more tractable form.

1.3 HAMILTONIAN MONTE CARLO

Hamiltonian Monte Carlo is a particular class of samplers which defines a kind of Metropolis-Hastings where the proposal is informed by the target distribution shape. Suppose we want to sample $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n})$. We can think of $\boldsymbol{\theta}$ as the position of a dynamical system. We augment the problem by an auxiliary variable, the momentum of our system, which describes the force applied to the system during its dynamical evolution.

The Hamiltonian Monte Carlo intuitively works by iterating the following steps

(a) update position and momentum, according to a trajectory based on Hamiltonian dynamics;

(b) perform a Metropolis-Hastings step to accept the proposed value.

The Hamiltonian Monte Carlo is exploring nicely the target distribution support. Works for continuous target distributions.

In short, we have a position $\boldsymbol{\theta} \in \mathbb{R}^d$ and a momentum $\boldsymbol{p} \in \mathbb{R}^d$. The system is described by a function of $(\boldsymbol{\theta}, \boldsymbol{p})$, say $\mathrm{H}(\boldsymbol{\theta}, \boldsymbol{p})$, known as Hamiltonian function. Such a function describe the evolution of the system over time, specifically by looking at its partial derivatives

$$\begin{cases} \dfrac{\mathrm{d}\theta_j}{\mathrm{d}t} = \dfrac{\partial H}{\partial p_j}, & j = 1, \dots, d, \\ \dfrac{\mathrm{d}p_j}{\mathrm{d}t} = -\dfrac{\partial H}{\partial \theta_j}, & j = 1, \dots, d. \end{cases}$$

We consider function $\mathrm{H}(\boldsymbol{\theta}, \boldsymbol{p})$ of the type

$$\underbrace{\mathrm{H}(\boldsymbol{\theta}, \boldsymbol{p})}_{\text{ENERGY}} = \underbrace{\mathrm{E}(\boldsymbol{\theta})}_{\text{POTENTIAL ENERGY}} + \underbrace{\mathrm{K}(\boldsymbol{p})}_{\text{KINETIC ENERGY}},$$

with $\mathrm{K}(\boldsymbol{p}) = \frac{1}{2}(\boldsymbol{p} M^{-1} \boldsymbol{p})$ and $\mathrm{E}(\boldsymbol{\theta}) = -\log \pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}) + \log C_{\boldsymbol{\theta}}$. $\mathrm{K}(\boldsymbol{p})$ leads to an independent $\mathbf{0}$-mean Gaussian distribution for the momentum. Assuming the previous function $\mathrm{K}(\boldsymbol{p})$, we have

$$\begin{cases} \dfrac{\mathrm{d}\theta_j}{\mathrm{d}t} = [M^{-1}\boldsymbol{p}]_j, & j = 1, \dots, d, \\ \dfrac{\mathrm{d}p_j}{\mathrm{d}t} = -\dfrac{\partial \mathrm{E}}{\partial \boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}(t) \mid \boldsymbol{y}_{1:n}). \end{cases}$$

We can solve the previous numerically, resorting to a discretization, for example using a leapfrog integrator. We start from $t = 0$, with an initial value for $\boldsymbol{\theta}$ and $\boldsymbol{p}$. We iterate the following to get a trajectory for position and momentum.

$$
\begin{aligned}
p_j(t + \epsilon/2) &= p_j(t) - \frac{\epsilon}{2}\left[\frac{\partial E}{\partial \theta_j}(\boldsymbol{\theta}(t))\right], \\
\theta_j(t + \epsilon) &= \theta_j(t) + \epsilon\left\{[M^{-1}\boldsymbol{p}(t + \epsilon/2)]_j\right\}, \\
p_i(t + \epsilon/2) &= p_i(t) - \frac{\epsilon}{2}\left[\frac{\partial E}{\partial \theta_i}(\boldsymbol{\theta}(t))\right].
\end{aligned}
$$

Ideally, with the previous for $t = 1, \ldots, L$ we can produce L distinct values of position and momentum, approximating locally the system trajectory. In our environment, we have

$$
\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:n}) = C_{\boldsymbol{\theta}} \mathrm{e}^{-\mathrm{E}(\boldsymbol{\theta})}, \qquad \mathcal{L}(\boldsymbol{p}) = C_{\boldsymbol{p}} \mathrm{e}^{-\mathrm{K}(\boldsymbol{p})},
$$

and we can simulate the joint distribution $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{p} \mid \boldsymbol{y}_{1:n}) \propto \mathrm{e}^{-\{\mathrm{E}(\boldsymbol{\theta}) + \mathrm{K}(\boldsymbol{p})\}}$. An idea of implementation is the following.

Algorithm 3: Pseudocode for the Hamiltonian Monte Carlo algorithm, sample of size T.

1: Set initial values for $\boldsymbol{\theta}^{(0)},\ \epsilon,\ L$.
2: **for** $t = 1$ to T **do**
3: Sample the momentum $\boldsymbol{p}_0 \sim C_{\boldsymbol{p}} \mathrm{e}^{-\mathrm{K}(\boldsymbol{p})}$ and set $\boldsymbol{\theta}_0^N = \boldsymbol{\theta}^{(r-1)}$
4: **for** $t = 1$ to L **do**
5: $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} - \frac{\epsilon}{2}\nabla \mathrm{E}(\boldsymbol{\theta}_{t-1}^N)$
6: $\boldsymbol{\theta}_t^N = \boldsymbol{\theta}_{t-1}^N + \epsilon M^{-1}\boldsymbol{p}_t$
7: $\boldsymbol{p}_t = \boldsymbol{p}_t - \frac{\epsilon}{2}\nabla \mathrm{E}(\boldsymbol{\theta}_t^N)$
8: **end for**
9: Negate the momentum $\boldsymbol{p}_L = -\boldsymbol{p}_L$.
10: Perform a Metropolis-Hastings step with acceptance rate

$$
\alpha = \min\left\{1, \frac{\exp\left\{\mathrm{E}(\boldsymbol{\theta}_L^N) + \mathrm{K}(\boldsymbol{p}_L)\right\}}{\exp\left\{\mathrm{E}(\boldsymbol{\theta}_0^N) + \mathrm{K}(\boldsymbol{p}_0)\right\}}\right\}.
$$

11: **end for**

Note that step 9, negating the momentum, is fundamental for having reversibility and hence to satisfy the detailed balanced condition.

Exercise 1.3. Let us consider the case with

$$
\boldsymbol{\theta} \sim N(\boldsymbol{\mu}, \Sigma), \qquad \boldsymbol{\mu} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \qquad \Sigma = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}.
$$

Write explicitly the quantities required to implement an Hamiltonian Monte Carlo to sample from the distribution of $\boldsymbol{\theta}$. Try to implement the algorithm in R.

2 STAN

Open source software, can be freely downloaded at http://mc-stan.org/.

STAN is a probabilistic programming language implementing full Bayesian statistical inference with MCMC sampling (NUTS, HMC) and penalized maximum likelihood estimation with Optimization (BFGS). STAN is coded in C++ and runs on all major platforms (Linux, Mac, Windows). STAN is an open-source software developed mainly at Columbia University.

STAN can be accessed through several interfaces: RStan (R), PyStan (Python), MatlabStan (Matlab) and also CmdStan (shell), http://mc-stan.org/users/interfaces/cmdstan. Multiple Markov chain Monte Carlo (MCMC) algorithms and optimization algorithms are implemented for the inference:

(1) MCMC algorithms:

- Hamiltonian Monte Carlo (HMC) (default).
- No-U-Turn sampler.

(2) Optimization algorithms:

- BFGS algorithm (default), Nesterov's accelerated gradient descent algorithm, Newton's method.

Typical workflow using stan for posterior sampling.

1. Represent a statistical model by writing its log-posterior density (up to a normalization constant independent from the parameters)
   - this can be done using STAN modeling language.
2. Translate the model coded in STAN to C++ code using the function stanc.
3. Compile the C++ code for the model using a C++ compiler (such as g++) to create a Dynamic Shared Object that can be loaded by R-
4. Run the DSO to sample from the posterior.
5. Diagnose convergence of the MCMC chains of sample.
6. Conduct model inference.

Don't worry! A single rstan call performs implicitly steps 2, 3 and 4.

All the built-in C++ functions and operators are available. For the matrices functions of basic arithmetics, solvers, decompositions,.. are available (check the manual). Each distribution of the library has

- pseudo random number generator;
- log density or mass function;
- cumulative distribution.

2.1 WRITING A MODEL

STAN has specific blocks, which describes all the quantity of interest, possible transformations, and distributional relations. The blocks must be kept in the following order while writing model files.

- **DATA**:
  - Given input data
  - Executed first and load
- **TRANSFORMED DATA**:
  - Transform variables for convenience
- **PARAMETERS**:
  - Result output parameters
  - Updated at each iteration
- **TRANSFORMED PARAMETERS**:
  - Transform parameters for convenience
- **MODEL**:
  - Describe the model
- **GENERATED QUANTITIES**:
  - Generate quantities for monitoring convergence

Blocks are optional (except model block). The data block parse to STAN data (and possibly hyperparameter values). Transformed data apply in advance suitable transformation of your data, instead of applying the transformation within the sampler. The parameters are nothing but the quantity of interest of our inference, what we want to sample, which can be possibly pre-transformed in the block transformed parameters (e.g., you can compute the linear predictor in a regression model specific for each observation). The model describe the distributional relations that we assume, such as likelihood function and prior distributions. The generated quantities block provide a way to parse as output other quantities rather than the parameter values.

2.2 A BIT OF STAN NOTATION

Different type of variable and expression types.

- **Primitive**: int and real
- **Matrix**: vector[n], row_vector[n], matrix[m, n]
- **Bounded**: primitive or matrix type, with $< lower = L >$, $< upper = U >$, $< lower = L, upper = U >$

- **Constrained**: unit_vector for unit-length vectors, simplex for unit simplexes, ordered for ordered vectors, corr_matrix and cov_matrix for symmetric and positive definite matrices.
- **Arrays**: collection of object of any type
- **Sampling**: $y \sim normal(mu, sigma)$
- **Increments log-probability**: $increment\_log\_prob(lp)$ add the value lp to the log-density
- **For/while loop**: $for(n\ in\ 1:N)$, $while(cond)$
- **Block**:{.....} (allows local variables)
- **Print**: $print("TH =",\ theta)$
- ....

Different methods of the class S4 stanfit, for example the followings.

- **show** Print the default summary for the model.
- **print** Print a customizable summary for the model. See print.stanfit.
- **plot** Create various plots summarizing the fitted model. See plot,stanfit-method.
- **summary**Summarize the distributions of estimated parameters and derived quantities using the posterior draws. See summary,stanfit-method.
- **get_posterior_mean** Get the posterior mean for parameters of interest (using pars to specify a subset of parameters). Returned is a matrix with one column per chain and an additional column for all chains combined.
- ....

see https://mc-stan.org/rstan/reference/stanfit-class.html for a summary.