A CASE STUDY: Comparing Gibbs, MH and HMC

Let P be a strongly correlated ($\beta = 0.9$) two-dimensional Gaussian distribution;

$$P = \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & \beta\\ \beta & 1\end{pmatrix}\right), \quad \text{with} \quad P(x,y) \propto \exp\left(\frac{-\left[x^2+y^2-2\beta xy\right]}{2[1-\beta^2]}\right). \tag{1}$$

This simple distribution can be efficiently sampled analytically (e.g. by using the error function and the inverse CDF to sample two univariate Gaussian distributions and then applying a rotation based on the eigenvectors of the covariance matrix) so there is no reason to use any MCMC method at all. Nevertheless, we will use this simple target distribution as a testbed for the three main families of MCMC algorithms we have encountered so far.

Gibbs sampling. This requires sampling the 1-dimensional conditional distributions $x|y$ and $y|x$. Fortunately, this can be done easily in this case as the conditional distributions are also Gaussian.

$$P(x|y) \propto \exp\left(\frac{-(x-\beta y)^2}{2[1-\beta^2]}\right) \quad \Rightarrow \quad x|y \sim \mathcal{N}(\beta y, 1-\beta^2). \tag{2}$$

Because P is symmetric in x and y, the other distribution is the same, $y|x \sim \mathcal{N}(\beta x, 1-\beta^2)$.

A major benefit of the Gibbs method is that it is extremely easy to implement (provided we have the conditional distributions) and it has no free parameters for the user to tune.

A Markov chain initialised at the origin $(x_0, y_0) = (0, 0)$ is evolved for $n = 10^5$ iterations using the basic Gibbs sampling algorithm where one component is randomly chosen to be updated at each iteration. A small part (20 iterations) of the chain is shown in Fig. 1.

Because Gibbs iterations use the conditional distributions the chain only moves horizontally or vertically and therefore can only make small steps in this strongly correlated target

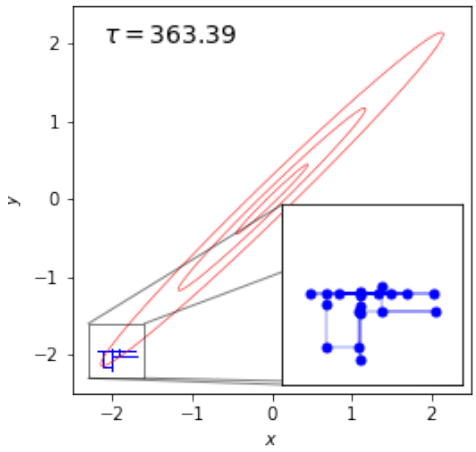

Figure 1: Illustration of the Gibbs algorithm showing 20 iterations of the Markov chain (blue) superposed on contours of the target distribution (red). The chain exhibits random walk behaviour; this is not desirable because it will take a long time for the chain to explore the full distribution. The IAT τ is also shown.

distribution. This leads to the chain performing an undesirable *random walk* behaviour that take many iterations to *diffuse* across the target distribution. This can be quantified by the IAT which is rather long. Gibbs sampling struggles with highly correlated distributions.

If we had known in advance what correlation to expect in the target distribution, then we could have define new rotated variables $(u, v) = \mathcal{R}_{\pi/4}(x, y)$ to use in the Gibbs algorithm. This would have dramatically improved the performance of the Gibbs sampling.

Metropolis-Hastings sampling. The MH algorithm is slightly more involved to implement than Gibbs (although it doesn't require the conditional distributions) and requires us to choose a proposal distribution. Here, a symmetric, Gaussian proposal is chosen with $Q = \mathcal{N}([0, 0], \mathbf{\Sigma})$; three choices for the covariance matrix $\mathbf{\Sigma}$ are considered:

$$\mathbf{\Sigma}_{\text{Small}} = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.05 \end{pmatrix}, \quad \mathbf{\Sigma}_{\text{Large}} = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix}, \quad \text{and} \quad \mathbf{\Sigma}_{\text{Corr}} = \begin{pmatrix} 1 & \beta \\ \beta & 1 \end{pmatrix}. \tag{3}$$

Three Markov chains are initialised at the origin $(x_0, y_0) = (0, 0)$ is evolved for $n = 10^5$ iterations using the MH sampling algorithm using the three proposals in Eq. 3. (Actually, because the proposal is symmetric, this is just the Metropolis algorithm.) A small part (40

iterations) of each chain is shown in Fig. 2.

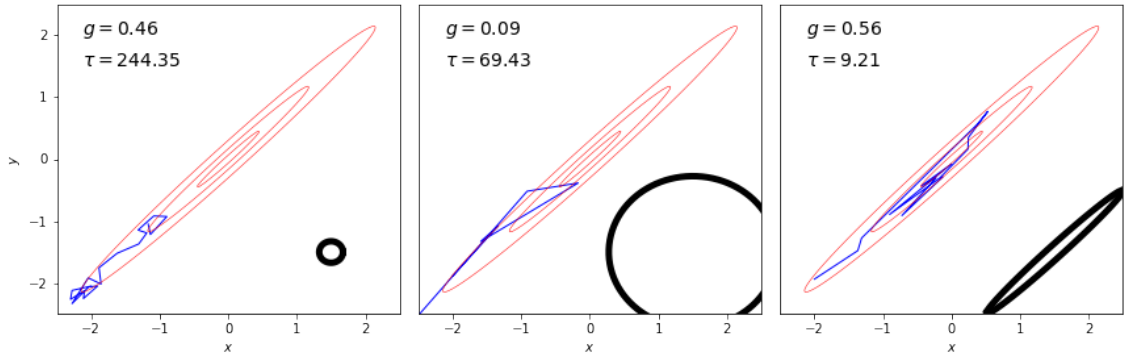

Figure 2: Illustration of the MH algorithm with Q_{small} (left), Q_{large} (centre), and Q_{corr} (right). Each plot shows 40 iterations (including duplicate points) of the Markov chain (blue) superposed on contours of the target distribution (red). Black lines show the shapes of the proposal distributions. Values of the acceptance fraction g and IAT τ are also shown.

Unlike Gibbs, MH algorithm can propose moves in any direction. Therefore, with a good choice of Q, it is possible for the chain to move across the target distribution in a small number of iterations. The price that we pay for this is that not all of the proposed moves are accepted and the chain sometimes doesn't move at all. If the proposal is too small, then we get a high acceptance fraction at the cost of small steps; the chain exhibits random walk behaviour (similar to the Gibbs chain) and we get a long IAT. Alternatively, if the proposal is too big it will frequently propose large steps in the wrong direction (e.g. $\searrow$ or $\nwarrow$) which will likely be rejected; the chain can take bigger steps at the cost of a lower acceptance fraction and because the chain often doesn't move at all, we still get a long IAT. We can try and get the best of both worlds (big steps and a high acceptance fraction with a smaller IAT) by carefully designing a proposal that preferentially proposes steps in the right directions ($\mathbf{\Sigma}_{\text{Corr}}$) but this requires that we know something extra about the shape of the target distribution, which isn't usually the case.

Note, the $\mathbf{\Sigma}_{\text{Corr}}$ proposal distribution is actually the same as the target distribution; i.e. we are stupidly using samples from P to produce samples from P! But this is just a toy example and is included here just show how well the MH algorithm can perform if we have a good proposal distribution.

Hamiltonian sampling. The HMC algorithm is more involved to implement. It also

requires the partial derivatives of the energy function, which for our target distribution are

$$E(x,y) = \frac{x^2+y^2}{2[1-\beta^2]} \quad \Rightarrow \frac{\partial E}{\partial x} = \frac{x-\beta y}{1-\beta^2}. \tag{4}$$

Because P is symmetric, the other derivative is identical; $\partial E/\partial y = \frac{y-\beta x}{1-\beta^2}$.

Three Markov chains are initialised at the origin $(x_0, y_0) = (0, 0)$ and are evolved for $n = 10^5$ iterations using the HMC sampling algorithm. We choose $M = \mathbb{I}_2$ and $\delta t = 0.05$. For the first chain, a small number of leapfrog iterations was used, $L_{\text{Short}} = 5$. For the second, $L_{\text{Mediaum}} = 50$. And for the third, $L_{\text{Long}} = 500$. A small part (10 iterations) of each chain is shown in Fig. 3.

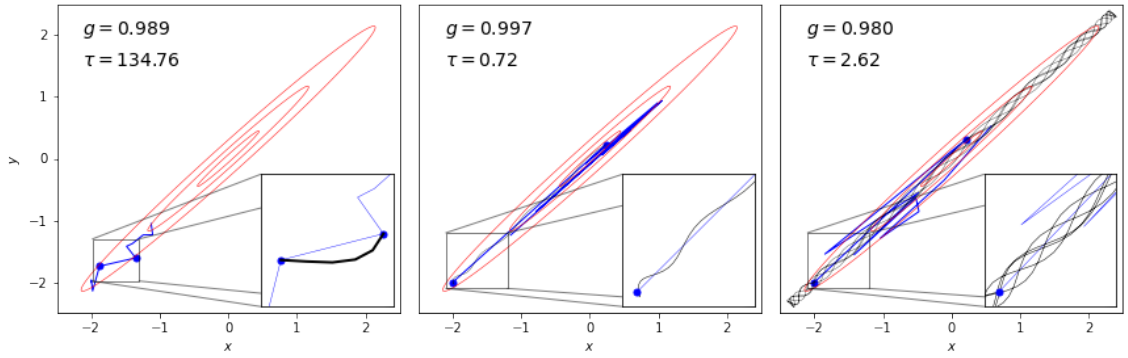

Figure 3: Illustration of the HMC algorithm with L_{Short} (left), L_{Medium} (centre), and L_{Long} (right). Each plot shows 10 iterations (including duplicate points) of the Markov chain (blue) superposed on contours of the target distribution (red). For one iteration, the trajectory of the numerical integration that made the proposal is shown in black. Values of the acceptance fraction g and IAT τ are also shown.

The clever feature of HMC is that it can take large steps in the target distribution (provided $L \gg 1$) while keeping the acceptance fraction high. If the numerical integration is accurate (meaning the Hamiltonian is approximately conserved) then $a \approx 1$. The ability to make large steps with a high acceptance fraction (see, for example, the centre or right panels of Fig. 3) leads to a small IAT. And unlike with the $\mathbf{\Sigma}_{\text{Corr}}$ MH method, HMC achieves without any detailed prior knowledge of the shape of the target distribution.

There are some drawbacks of HMC. The algorithm is more complicated than Gibbs or MH; each iteration requires the numerical integration of Hamilton's equations of motion. The

user must also provide the derivatives of target PDF (and the use of derivatives restricts the method to smooth target distributions). Another drawback is the number of inputs to the algorithm that the user must provide and tune by hand, such the mass matrix M and integration parameters L and ΔT. In particular, the choice of L is very important. Small L will mean the algorithm takes small steps and we are back to the undesirable situation where the Markov chain executes random walk behaviour (left panel of Fig. 3). Large L will lead to wasted time at each iteration numerically integrating the equations as the particle oscillates around the potential well (right panel of Fig. 3, black curve).

A summary of the performances of the three MCMC algorithms is provided in the first table. On this simple toy problem, the best performing algorithm (in the sense of producing the greatest number of independent samples for the least computational time) is HMC with well-chosen values for L and Δt), closely followed by MH with a well-chosen proposal distribution Q. Crude computational time estimates were obtained from the wall time of $n = 10^5$ iterations of my own Python implementation of each algorithm running on a single laptop core. Care should be taken when drawing general conclusions about the three algorithms from this simple case study; their relative performance can vary considerably with different implementation and/or input parameter choices and for different target distributions, especially in high dimensions.

A summary of the algorithm requirements is provided in the second table.

Algorithm	IAT τ	acceptance frac g	num. ind. samples produced $N_{\text{eff}} = \frac{n}{10\lceil\tau\rceil}$	time per iter [ms]	time per sample [ms]
Gibbs	363.4	=1	27	0.6	2222
MH Q_{Small}	244.4	0.46	41	0.3	732
MH Q_{Large}	69.4	0.09	143	0.3	210
MH Q_{Corr}	9.2	0.56	1000	0.3	30
HMC L_{Short}	134.4	0.99	74	0.2	270
HMC L_{medium}	0.72	0.997	10000	0.6	6
HMC L_{Long}	2.6	0.98	3333	5.0	150

Algorithm	needs $x\|y$?	needs $\nabla \log P$?	user inputs
Gibbs	yes	no	-
MH	no	no	proposal dist, Q
HMC	no	yes	many: L, δt, M