An Efficient Implementation of Riemannian Manifold Hamiltonian Monte Carlo for Gaussian Process Models

Ulrich Paquet*, Marco Fraccaro[a]

[a]*Technical University of Denmark, Lyngby, Denmark*

Abstract

This note presents pseudo-code for a Riemannian manifold Hamiltonian Monte Carlo (RMHMC) method to efficiently simulate samples from N-dimensional posterior distributions $p(\mathbf{x}|\mathbf{y})$, where $\mathbf{x} \in \mathbb{R}^N$ is drawn from a Gaussian Process (GP) prior, and observations y_n are independent given x_n. Sufficient technical and algorithmic details are provided for the implementation of RMHMC for distributions arising from GP priors.

1. Introduction

When data is modelled with Gaussian process (GP) priors, the resulting posterior distributions are usually highly correlated. There are various avenues to simulating samples from such posterior distributions with Markov chain Monte Carlo (MCMC) methods, ranging from simple Metropolis-Hastings (MH) methods with symmetric proposal distributions to component-wise Gibbs samplers, to more advanced Hamiltonian Monte Carlo (HMC) methods that use the gradient of the log-posterior to guide the sampler towards high-density regions. The strong posterior correlations can adversely affect the mixing rates of these methods.

The mixing rates of an MCMC method can be increased significantly when, instead of using only first-order gradient information, one additionally relies on local second-order statistics of the log-posterior to guide the sampler. In this note, we present **pseudo-code** for a **Riemannian manifold Hamiltonian Monte Carlo** (RMHMC) method [1] to efficiently simulate samples from N-dimensional posterior distributions $p(\mathbf{x}|\mathbf{y})$, where $\mathbf{x} \in \mathbb{R}^N$ is drawn from a zero-mean GP prior

$$\text{prior}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{K}) \tag{1}$$

with kernel matrix $\mathbf{K}$. The aim of this note is to provide sufficient technical and algorithmic details for anyone to implement RMHMC for GPs. We assume that the likelihood for each observation y_n depends only on the latent function value x_n through

$$\ell_n(x_n) \doteq \log p(y_n|x_n) \tag{2}$$

and is not Gaussian. The resulting posterior or target distribution is

$$p(\mathbf{x}|\mathbf{y}) = \frac{\prod_n p(y_n|x_n) \cdot \text{prior}(\mathbf{x})}{p(\mathbf{y})} \,. \tag{3}$$

The variables in (3) often form a very correlated high dimensional density, which is ideally suited to RMHMC. One could also sample and infer the GP kernel hyperparameters that govern $\mathbf{K}$. Additionally sampling kernel hyperparameters is outside the scope of this note, but a sketch for how they are sampled for a fully Gaussian model is given in [2].

1.1. A formulation for obtaining normalising constants

Aside from sampling from $p(\mathbf{x}|\mathbf{y})$, we are also interested in using samples to estimate $\log Z \doteq \log p(\mathbf{y})$. We will write (3) in a slightly more general form, so that the normalising constant $Z \doteq p(\mathbf{y})$ could also be recovered from a method like Annealed Importance Sampling (AIS) or Parallel Tempering (PT).

In a general form, some other distribution $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ might also be available to us, where the choice of $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \mathbf{K}$ would simply recover the prior. For instance, if $q(\mathbf{x})$ approximates $p(\mathbf{x}|\mathbf{y})$ via some deterministic approximate inference method like Expectation Propagation (EP), then this could be used as the starting distribution at $\beta = 0$ in AIS or PT. We will touch on AIS and PT in Section **??**. For now, let the unnormalised log density of $\mathbf{x}$ be[1]

$$\begin{aligned}\mathcal{L}_\beta(\mathbf{x}) = \beta \left[\sum_{n=1}^N \ell_n(x_n) - \frac{1}{2}\mathbf{x}^T\mathbf{K}^{-1}\mathbf{x}\right] \\ - (1-\beta)\,\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\end{aligned} \tag{4}$$

for any $\beta \in [0, 1]$. The generalised target distribution is

$$p_\beta(\mathbf{x}) \doteq \frac{1}{Z(\beta)}\, \mathrm{e}^{\mathcal{L}_\beta(\mathbf{x})} \tag{5}$$

for any choice of β. In this setup $\beta = 1$ recovers $p(\mathbf{x}|\mathbf{y})$, and $\beta = 0$ recovers $q(\mathbf{x})$, or the prior if $q(\mathbf{x}) = \text{prior}(\mathbf{x})$. Methods

*Corresponding author

Email addresses: ulrich@cantab.net (Ulrich Paquet), marfra@dtu.dk (Marco Fraccaro)

[1] We include the normalising constants for $\text{prior}(\mathbf{x})$ and $q(\mathbf{x})$ when $\mathcal{L}_\beta(\mathbf{x})$ and its derivative are computed in Lines 13 and 17 in Algorithm 4, but as they're independent of $\mathbf{x}$, we omit them here to make the explanation simpler.

like AIS or PT estimate $\log Z$ by sampling from a sequence of distributions that range from p_0 to p_1.

In the rest of this note, after briefly introducing Hamiltonian Monte Carlo (HMC) in Section 2, we will introduce RMHMC as an extension of HMC that exploits local second-order statistics (Section 3). We will then discuss and evaluate a RMHMC method to simulate samples from (5). Where the β subscript is clear from the context, it will be dropped for brevity.

2. Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [3] can be used to define efficient proposal distributions for a Metropolis-Hastings sampler, that allow large moves in the parameter space while keeping a high acceptance rate. It is particularly useful for eliminating the random walk behaviour that is typical of symmetric proposal distributions, and improves poor mixing in case of highly correlated variables. The main idea behind this algorithm is to define an Hamiltonian function in terms of the target probability distribution, and move a sample from this distribution as if it was a particle in space following the corresponding Hamiltonian dynamics.

To sample a random variable $\mathbf{x} \in \mathbb{R}^N$ from the probability distribution $p(\mathbf{x})$, we introduce an independent auxiliary variable $\mathbf{p} \in \mathbb{R}^N$ with a Gaussian prior $p(\mathbf{p}) = \mathcal{N}(\mathbf{p}; \mathbf{0}, \mathbf{M})$. Due to independence, the joint density can be written as

$$p(\mathbf{x}, \mathbf{p}) = p(\mathbf{x})\, p(\mathbf{p}) \propto \mathrm{e}^{-H(\mathbf{x},\mathbf{p})} \,. \tag{6}$$

The negative joint log-probability is then

$$H(\mathbf{x}, \mathbf{p}) = -\mathcal{L}(\mathbf{x}) + \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}$$

and can be interpreted as a Hamiltonian $H(\mathbf{x}, \mathbf{p})$ with potential energy $U(\mathbf{x})$ and kinetic energy $K(\mathbf{p})$,

$$\begin{aligned} U(\mathbf{x}) &= -\mathcal{L}(\mathbf{x}) \\ K(\mathbf{p}) &= \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} \,. \end{aligned}$$

The Hamiltonian represents the total energy of a closed system, in which $\mathbf{x}$ is the position of the particle in space. Thanks to the quadratic kinetic term, the auxiliary variable $\mathbf{p}$ can be seen as a momentum variable, and the covariance matrix $\mathbf{M}$ as a mass matrix. As we will see in Section 3, RMHMC improves the sampling efficiency of HMC by making use of a position-dependent mass matrix, at the expense of complicating the overall algorithm.

To obtain samples from $p(\mathbf{x})$, HMC simulates $\{(\mathbf{x}^{(t)}, \mathbf{p}^{(t)})\}_{t=1}^{t_{\max}}$ samples from $p(\mathbf{x}, \mathbf{p})$ and discards the $\mathbf{p}^{(t)}$ samples. The remaining $\{\mathbf{x}^{(t)}\}_{t=1}^{t_{\max}}$ will then represent samples from the required marginal distribution. HMC samples from the joint distribution $p(\mathbf{x}, \mathbf{p})$ using a Gibbs sampling scheme with auxiliary variables $\mathbf{p}$.

1. Given the position and momentum $(\mathbf{x}^{(t)}, \mathbf{p}^{(t)})$ at time t, the momentum is updated by drawing a sample from the conditional distribution $p(\mathbf{p}^{(t+1)}|\mathbf{x}^{(t)}) = p(\mathbf{p}^{(t+1)}) = \mathcal{N}(\mathbf{p}^{(t+1)}; \mathbf{0}, \mathbf{M})$.

2. Given $\mathbf{p}^{(t+1)}$, the variable $\mathbf{x}^{(t+1)}$ is sampled from the conditional distribution $p(\mathbf{x}^{(t+1)}|\mathbf{p}^{(t+1)})$ with the Metropolis-Hastings algorithm, using a deterministic proposal distribution defined by simulating the behaviour of the physical system evolving under Hamiltonian dynamics and with initial position $(\mathbf{x}^{(t)}, \mathbf{p}^{(t+1)})$. Given the Hamiltonian, we numerically integrate Hamilton's equations

$$\begin{aligned} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} &= \frac{\partial H}{\partial \mathbf{p}} = \frac{\partial K}{\partial \mathbf{p}} = \mathbf{M}^{-1}\mathbf{p} \\ \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}\tau} &= -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial U}{\partial \mathbf{x}} = \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}) \,. \end{aligned} \tag{7}$$

and follow a trajectory to obtain a new pair $(\mathbf{x}^*, \mathbf{p}^*)$, that is accepted with probability

$$\alpha = \min\left(1, \frac{\mathrm{e}^{-H(\mathbf{x}^*,\mathbf{p}^*)}}{\mathrm{e}^{-H(\mathbf{x}^{(t)},\mathbf{p}^{(t+1)})}}\right) \,. \tag{8}$$

The accept-step appears in Lines 30 to 35 in Algorithm 1.

The validity of this sampler relies on the *reversibility* and *volume preservation* properties of Hamiltonian mechanics [3]. When an analytic solution to the system of nonlinear differential equations is available, the proposed trajectory moves along the isocontours of the Hamiltonian in the phase space (and the acceptance rate in therefore one), whereas the random draws of the momentum $\mathbf{p}$ from the exact conditional distribution will change the energy levels. For practical applications of interest, Hamilton's equations do not have an analytic solution, and it is therefore necessary to discretise time and resort to numerical approximations. It is common to use the *Stormer-Verlet leapfrog integrator*, that retains the reversibility and volume preservation properties required to obtain an exact sampler, and computes the updates (in vector form) as

$$\begin{aligned} \mathbf{p}(\tau + \tfrac{\epsilon}{2}) &= \mathbf{p}(\tau) - \tfrac{\epsilon}{2}\nabla_{\mathbf{x}}U(\mathbf{x}(\tau)) \\ \mathbf{x}(\tau + \epsilon) &= \mathbf{x}(\tau) + \epsilon\nabla_{\mathbf{p}}K(\mathbf{p}(\tau + \tfrac{\epsilon}{2})) \\ \mathbf{p}(\tau + \epsilon) &= \mathbf{p}(\tau + \tfrac{\epsilon}{2}) - \tfrac{\epsilon}{2}\nabla_{\mathbf{x}}U(\mathbf{x}(\tau + \epsilon)) \,. \end{aligned} \tag{9}$$

After half a step for the momentum variables, a full step for the position variables is done using the new momentum variables, which is finally followed by the missing half step for the momentum variables using the updated position variables. This procedure is repeated $l_{\max}$ times for each sample to avoid random walk behaviour. Due to the integration errors caused by the discretisation, the Hamiltonian is not exactly conserved with the leapfrog method, but it remains close to the true value to give a high acceptance probability. This error can be also controlled by careful tuning (manual or automatic) of the step size ϵ and the maximum number of integration steps $l_{\max}$.

2.1. Tuning HMC

The performance of Hamiltonian Monte Carlo is highly dependent on the correct tuning of the step size ϵ, the number of leapfrog steps $l_{\max}$ done at each iteration and the mass matrix $\mathbf{M}$ of the momentum variable. A too low step size ϵ makes it difficult to completely explore the whole space unless a high

$l_{\max}$ is used, but this causes an increase in computational time. On the other hand, a too big ϵ can lead to an unstable algorithm that suffers from a low acceptance rate. Furthermore, if $l_{\max}$ is too small there will be slow mixing due to random-walk behaviour, whereas a too high value for $l_{\max}$ may cause *double-back behaviour*, where the integrator returns to its starting point. Picking the right mass matrix $\mathbf{M}$ is essential for optimised performance, as its diagonal terms have to reflect the scale of the sampled momentum variables and the off-diagonal terms their correlation: a simple default choice using a (possibly scaled) identity matrix will give poor results for highly correlated variables.

It is finally worth noting that an acceptance rate of 1 is not the optimal choice, as this would mean for example that ϵ could be increased to move even further in the phase space. [3] shows that the optimal acceptance rate should be around 0.65. One can also consider a number of burn-in samples to avoid highly correlated proposed samples, not too high though for an efficient implementation.

3. Riemannian Manifold Hamiltonian Monte Carlo

As argued in the previous section, one of the main issues that arise when Hamiltonian Monte Carlo is used, is the difficult tuning of the mass matrix $\mathbf{M}$, which is essential for good convergence of the sampler. Girolami *et al.* show that when, for instance, the position dependent expected Fisher information matrix $\mathbf{G}(\mathbf{x})$ is used instead of a fixed mass matrix $\mathbf{M}$, many of the shortcomings of HMC can be addressed [2]. The introduced algorithm – *Riemannian manifold Hamiltonian Monte Carlo (RMHMC)* – can be seen as an extension of Hamiltonian Monte Carlo where the local geometry of the distribution we want to sample from is taken into account through the metric tensor $\mathbf{G}(\mathbf{x})$.

In RMHMC, the covariance structure of the auxiliary Gaussian momentum variable is set as $\mathbf{G}(\mathbf{x})$, so that its distribution is shaped by the position of $\mathbf{x}$, i.e. $p(\mathbf{p}|\mathbf{x}) = \mathcal{N}(\mathbf{p}; \mathbf{0}, \mathbf{G}(\mathbf{x}))$. While HMC factorises in (6) and uses $p(\mathbf{p}) = \mathcal{N}(\mathbf{p}; \mathbf{0}, \mathbf{M})$ for some fixed mass matrix $\mathbf{M}$, for RMHMC the joint density

$$p(\mathbf{x}, \mathbf{p}) = p(\mathbf{x})\, p(\mathbf{p}|\mathbf{x}) \propto \mathrm{e}^{-H(\mathbf{x},\mathbf{p})} \tag{10}$$

is no longer factorisable. As a result, the Hamiltonian

$$H(\mathbf{x}, \mathbf{p}) = -\mathcal{L}(\mathbf{x}) + \frac{1}{2}\log\left((2\pi)^N \det(\mathbf{G}(\mathbf{x}))\right) + \frac{1}{2}\mathbf{p}^T\mathbf{G}(\mathbf{x})^{-1}\mathbf{p} \tag{11}$$

is not separable. Unlike HMC, the term coming from the Gaussian normalising constant depends on $\mathbf{x}$, and it needs to be included in the potential energy term.

RMHMC uses the same Gibbs sampler as HMC. First, the momentum is sampled from the conditional distribution $\mathbf{p}|\mathbf{x}$ and then a new proposal for a Metropolis-Hastings sampler is found following a trajectory that is obtained by solving Hamilton's equations, that are in this case

$$\begin{aligned}\frac{\mathrm{d}x_n}{\mathrm{d}\tau} &= \frac{\partial H}{\partial p_n} = \{\mathbf{G}(\mathbf{x})^{-1}\mathbf{p}\}_n \\ \frac{\mathrm{d}p_n}{\mathrm{d}\tau} &= -\frac{\partial H}{\partial x_n} = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial x_n} - \frac{1}{2}\,\mathsf{tr}\left\{\mathbf{G}(\mathbf{x})^{-1}\frac{\partial \mathbf{G}(\mathbf{x})}{\partial x_n}\right\} \\ &\quad + \frac{1}{2}\mathbf{p}^T\mathbf{G}(\mathbf{x})^{-1}\frac{\partial \mathbf{G}(\mathbf{x})}{\partial x_n}\mathbf{G}(\mathbf{x})^{-1}\mathbf{p}\,. \end{aligned} \tag{12}$$

Due to this dependence of the kinetic energy on the position (through $G(\mathbf{x})$), the proposals generated from the leapfrog integrator will not satisfy detailed balance in a Hamiltonian Monte Carlo scheme. To overcome this problem, [2] uses a more general leapfrog integrator, which is semi-explicit (i.e. the update equations are defined implicitly and need to be solved with some fixed point iterations) but that satisfies reversibility and volume preservation, therefore giving a correct sampler. This *generalised leapfrog integrator* leads to the following updates of the position and momentum variables:

$$\mathbf{p}(\tau + \tfrac{\epsilon}{2}) = \mathbf{p}(\tau) - \tfrac{\epsilon}{2}\nabla_{\mathbf{x}} H\Big(\mathbf{x}(\tau), \mathbf{p}(\tau + \tfrac{\epsilon}{2})\Big) \tag{13}$$

$$\mathbf{x}(\tau + \epsilon) = \mathbf{x}(\tau) + \tfrac{\epsilon}{2}\Big[\nabla_{\mathbf{p}} H\Big(\mathbf{x}(\tau), \mathbf{p}(\tau + \tfrac{\epsilon}{2})\Big) + \nabla_{\mathbf{p}} H\Big(\mathbf{x}(\tau + \epsilon), \mathbf{p}(\tau + \tfrac{\epsilon}{2})\Big)\Big] \tag{14}$$

$$\mathbf{p}(\tau + \epsilon) = \mathbf{p}(\tau + \tfrac{\epsilon}{2}) - \tfrac{\epsilon}{2}\nabla_{\mathbf{x}} H\Big(\mathbf{x}(\tau + \epsilon), \mathbf{p}(\tau + \tfrac{\epsilon}{2})\Big)\,. \tag{15}$$

It is simple to show that for separable Hamiltonians the generalised leapfrog integrator coincides with the one defined for HMC in Equations (9). When the Hamiltonian is non-separable, (13) and (14) are implicitly defined. In Algorithm 1, we solve them through *fixed point iterations*: Lines 13-17 for Equation (13), and Lines 20-25 for Equation(14).

Figure 1 shows an example of a trajectory followed by HMC and RMHMC to obtain one sample from a mixture of three Gaussians, starting from the same initial position. We can see that RMHMC is better than HMC at taking into account the local geometry of the correlated Gaussian distributions, and can therefore do bigger moves in parameters space accepted with high probability.

4. RMHMC for Gaussian Process Classification

In this section, we take GP classification (GPC) as a working example of a GP model with non-Gaussian likelihood terms. For a GPC problem, every latent function value x_n supports a binary observation $y_n \in \{-1, +1\}$. The probit function $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(z; 0, 1)\,\mathrm{d}z$ is often used to model the likelihood,

$$\ell_n(x_n) \doteq \log p(y_n|x_n) = \log \Phi(y_n x_n)\,.$$

The resulting function $\mathcal{L}_\beta(\mathbf{x})$ in (4) is concave, and $p(\mathbf{x}|\mathbf{y})$ is usually very correlated.

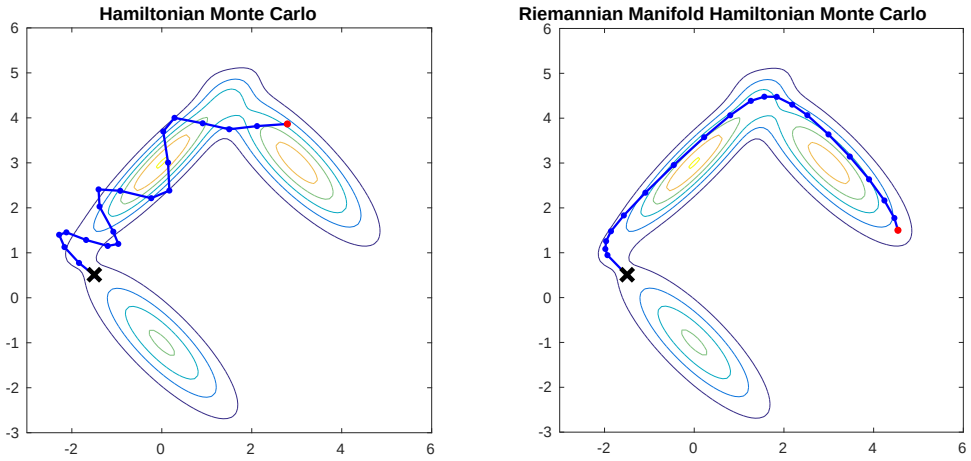

Figure 1: Comparison between trajectories followed by HMC and RMHMC starting from the same initial position (the black cross).

The Markov chain Monte Carlo (MCMC) algorithm that draws samples from (5) via the joint density in (10) is given in pseudocode in Algorithm 1. We'll first discuss the its inputs in Section 4.1, and then turn to its choice of metric tensor in Section 4.2. The resulting Hamiltonian and its gradients are given in Section 4.3. Section 4.4 gives the first, second, and third derivatives for a probit log likelihood, which are all required in $\partial H / \partial \mathbf{x}$.

4.1. Inputs

Algorithm 1 requires the parameters of $\mathcal{L}_\beta(\mathbf{x})$ in (4), with minimal other external settings:

- The input observations $\mathbf{y}$ from (2) and kernel matrix $\mathbf{K}$ from the GP prior in (1) are required. It is also useful to precompute the Cholesky decomposition $\mathbf{K} = \mathbf{L}^{\mathrm{K}}(\mathbf{L}^{\mathrm{K}})^T$ and log determinant $\log|\mathbf{K}|$ of the kernel matrix.
- If $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is an approximation obtained by EP, then the precision matrix of q decomposes as $\mathbf{K}^{-1}$ plus a *diagonal* matrix $\widetilde{\boldsymbol{\Sigma}}$ containing the contributions from N approximate factors corresponding to the likelihood terms,

$$\boldsymbol{\Sigma}^{-1} = \mathbf{K}^{-1} + \widetilde{\boldsymbol{\Sigma}}^{-1} .$$

  We choose the above notation to match that of the GP classification approximation in Rasmussen and Williams's *Gaussian Processes for Machine Learning* book [4]. Of course $\boldsymbol{\Sigma}$ needn't have this form, but as we consider an EP approximation in Section 5, use this form for numerical stability. In addition to taking $\boldsymbol{\mu}$ and $\widetilde{\boldsymbol{\Sigma}}$ as inputs, we also precompute the Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{L}^{\Sigma}(\mathbf{L}^{\Sigma})^T$ and its log determinant $\log|\boldsymbol{\Sigma}|$.
- Inverse temperature β, a starting point $\mathbf{x}_0$, the number of leapfrog steps $l_{\max}$ per sample, a step-size ϵ, and a maximum number of samples $t_{\max}$ are also required. We implicitly assume that a burn-in sample would be discarded from $\{\mathbf{x}_t\}_{t=1}^{t_{\max}}$, and that the number of fixed point iterations is pre-set to, say, $f_{\max} = 5$.

4.2. Metric tensor

There are many choices of a metric for a specific manifold, and a more detailed discussion beyond the scope of this note is presented by Girolami and Calderhead [1]. We choose the negative second derivative of $\mathcal{L}(\mathbf{x})$ in (4), evaluated at $\mathbf{x}$, as metric tensor:

$$\mathbf{G}(\mathbf{x}) \doteq -\nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}) .$$

Therefore $\mathbf{G}(\mathbf{x})$ is simply a Hessian matrix,

$$-\nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}) = \underbrace{-\beta \,\mathsf{diag}\Big[\nabla^2_{x_n} \ell_n(x_n)\Big]}_{\boldsymbol{\Lambda}(\mathbf{x})} + \beta\mathbf{K}^{-1} + (1-\beta)\boldsymbol{\Sigma}^{-1}.$$

Notation diag indicates a diagonal matrix formed by its arguments. We deliberately set aside a definition of $\boldsymbol{\Lambda}(\mathbf{x})$ as the only component of $\mathbf{G}(\mathbf{x})$ that depends on $\mathbf{x}$. This simplifies later derivations, for which another derivative (the third) is required in $\partial \mathbf{G}(\mathbf{x}) / \partial x_n$ in (12) when simulating the Hamiltonian dynamics. Hence

$$\mathbf{G}(\mathbf{x}) = \boldsymbol{\Lambda}(\mathbf{x}) + \beta\mathbf{K}^{-1} + (1-\beta)\boldsymbol{\Sigma}^{-1}. \tag{16}$$

4.3. The Hamiltonian and its gradients

First consider $-\partial H / \partial \mathbf{x}$ in Equation (12). A fast way to compute it is given in function **hamiltonian-and-gradient** in Algorithm 2. It relies on the metric tensor, computed by the **riemann-metric** function in Algorithm 3, and the derivatives $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x})$, computed in the **derivatives** function in Algorithm 4. Considering the latter, the derivative of $\mathcal{L}$ with respect to $\mathbf{x}$ is

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}) = \beta \,\mathsf{vec}\Big[\nabla_{x_n} \ell_n(x_n)\Big] - \beta\mathbf{K}^{-1}\mathbf{x} - (1-\beta)\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$$

where vec gives a column vector of its arguments. Function **derivatives** uses pre-computed Cholesky decompositions of $\mathbf{K} = \mathbf{L}^{\mathrm{K}}(\mathbf{L}^{\mathrm{K}})^T$ and $\boldsymbol{\Sigma} = \mathbf{L}^{\Sigma}(\mathbf{L}^{\Sigma})^T$ to stably determine $\mathbf{K}^{-1}\mathbf{x}$ and $\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$ through back-solving. The Cholesky

Algorithm 1 Riemannian Manifold Hamiltonian Monte Carlo

1: **input:** $\mathbf{y}$, $\mathbf{K}$, $\mathbf{L}^{\mathrm{K}}$, $\log|\mathbf{K}|$, $\boldsymbol{\mu}$, $\widetilde{\boldsymbol{\Sigma}}$, $\mathbf{L}^{\Sigma}$, $\log|\boldsymbol{\Sigma}|$, β, $\mathbf{x}_0$, $l_{\max}$, ϵ, $t_{\max}$
2: $\mathbf{x} := \mathbf{x}_0$
3: $\{\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}\} :=$ **riemann-metric**$(\mathbf{x}, \mathbf{y}, \mathbf{K}, \mathbf{L}^{\mathrm{K}}, \log|\mathbf{K}|, \boldsymbol{\mu}, \widetilde{\boldsymbol{\Sigma}}, \mathbf{L}^{\Sigma}, \log|\boldsymbol{\Sigma}|, \beta)$
4: **for** $t = 1$ to $t_{\max}$ **do**
5: $\quad \mathbf{L}^{\mathrm{Ginv}} = \mathsf{chol}(\mathbf{G}^{-1})$
6: $\quad \mathbf{p} \sim \mathcal{N}(\mathbf{p}; \mathbf{0}, \mathbf{I})$
7: $\quad \mathbf{p} := (\mathbf{L}^{\mathrm{Ginv}})^T \backslash \mathbf{p}$ *// initial momentum is* $\mathbf{p} \sim \mathcal{N}(\mathbf{p}; \mathbf{0}, \mathbf{G}(\mathbf{x})^{-1})$
8: $\quad \{H, \frac{\partial H}{\partial \mathbf{x}}\} :=$ **hamiltonian-and-gradient**$(\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}, \mathbf{p})$
9: $\quad H^{\mathsf{old}} := H$; $\mathbf{x}^{\mathsf{old}} := \mathbf{x}$; $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}^{\mathsf{old}} := \frac{\partial \mathcal{L}}{\partial \mathbf{x}}$; $\mathcal{L}^{\mathsf{old}} := \mathcal{L}$; $(\mathbf{G}^{-1})^{\mathsf{old}} := \mathbf{G}^{-1}$; $(\log|\mathbf{G}|)^{\mathsf{old}} := \log|\mathbf{G}|$; $(\mathrm{d}\mathbf{g})^{\mathsf{old}} := \mathrm{d}\mathbf{g}$
10: $\quad$ *// take* $l_{\max}$ *leapfrog steps:*
11: $\quad$ **for** $l = 1$ to $l_{\max}$ **do**
12: $\quad\quad$ *// the fixed point loops* $f = 1, \ldots, f_{\max}$ *below are a time-reversible volume preserving numerical integrator for solving the non-separable Hamiltonian to ensure a correct MCMC algorithm*
13: $\quad\quad$ **for** $f = 1$ to $f_{\max}$ **do**
14: $\quad\quad\quad \mathbf{p}' := \mathbf{p} - \frac{1}{2}\epsilon \frac{\partial H}{\partial \mathbf{x}}$ *// equation (13)*
15: $\quad\quad\quad \{H, \frac{\partial H}{\partial \mathbf{x}}\} :=$ **hamiltonian-and-gradient**$(\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}, \mathbf{p}')$
16: $\quad\quad$ **end for**
17: $\quad\quad \mathbf{p} := \mathbf{p}'$
18: $\quad\quad \frac{\partial H}{\partial \mathbf{p}} := \mathbf{G}^{-1}\mathbf{p}$
19: $\quad\quad \frac{\partial H}{\partial \mathbf{p}}' := \frac{\partial H}{\partial \mathbf{p}}$
20: $\quad\quad$ **for** $f = 1$ to $f_{\max}$ **do**
21: $\quad\quad\quad \mathbf{x}' := \mathbf{x} + \frac{1}{2}\epsilon(\frac{\partial H}{\partial \mathbf{p}} + \frac{\partial H}{\partial \mathbf{p}}')$ *// equation (14)*
22: $\quad\quad\quad \{\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}\} :=$ **riemann-metric**$(\mathbf{x}', \mathbf{y}, \mathbf{K}, \mathbf{L}^{\mathrm{K}}, \log|\mathbf{K}|, \boldsymbol{\mu}, \widetilde{\boldsymbol{\Sigma}}, \mathbf{L}^{\Sigma}, \log|\boldsymbol{\Sigma}|, \beta)$
23: $\quad\quad\quad \frac{\partial H}{\partial \mathbf{p}}' := \mathbf{G}^{-1}\mathbf{p}$
24: $\quad\quad$ **end for**
25: $\quad\quad \mathbf{x} := \mathbf{x}'$
26: $\quad\quad \{H, \frac{\partial H}{\partial \mathbf{x}}\} :=$ **hamiltonian-and-gradient**$(\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}, \mathbf{p})$
27: $\quad\quad \mathbf{p} := \mathbf{p} - \frac{1}{2}\epsilon \frac{\partial H}{\partial \mathbf{x}}$ *// equation (15)*
28: $\quad\quad \{H, \frac{\partial H}{\partial \mathbf{x}}\} :=$ **hamiltonian-and-gradient**$(\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}, \mathbf{p})$
29: $\quad$ **end for**
30: $\quad$ **if** $\mathrm{rand} < \exp\{H^{\mathsf{old}} - H\}$ **then**
31: $\quad\quad$ *// accept; do nothing*
32: $\quad$ **else**
33: $\quad\quad$ *// reject*
34: $\quad\quad \mathbf{x} := \mathbf{x}^{\mathsf{old}}$; $\frac{\partial \mathcal{L}}{\partial \mathbf{x}} := \frac{\partial \mathcal{L}}{\partial \mathbf{x}}^{\mathsf{old}}$; $\mathcal{L} := \mathcal{L}^{\mathsf{old}}$; $\mathbf{G}^{-1} := (\mathbf{G}^{-1})^{\mathsf{old}}$; $\log|\mathbf{G}| := (\log|\mathbf{G}|)^{\mathsf{old}}$; $\mathrm{d}\mathbf{g} := (\mathrm{d}\mathbf{g})^{\mathsf{old}}$
35: $\quad$ **end if**
36: $\quad \mathrm{samples}(t) = \mathbf{x}$
37: $\quad \mathrm{energies}(t) = -\mathcal{L}$
38: **end for**
39: **return** samples, energies

factors are lower-diagonal, and $\circ$ indicates the element-wise or Hadamard product between a pair of vectors or matrices.

Turning to the **riemann-metric** function, quantities like

$$\mathsf{tr}\left\{\mathbf{G}(\mathbf{x})^{-1}\frac{\partial \mathbf{G}(\mathbf{x})}{\partial x_n}\right\}$$

are required in Equation (12), whilst the evaluation of H also needs $\log|\mathbf{G}(\mathbf{x})|$. Notice that the derivative

$$\left\{\frac{\partial \mathbf{G}}{\partial x_n}\right\}_{n,n} = \frac{\partial \Lambda_{nn}}{\partial x_n} \tag{17}$$

is non-zero in position (n, n), and zero elsewhere, and $\mathbf{G}$ has no other dependence on $\mathbf{x}$ than through the log likelihood derivatives. We therefore only keep the non-zero entries, given by (17), in a vector $\mathrm{d}\mathbf{g}$ in Algorithm 3.

To evaluate the log determinant $\log|\mathbf{G}(\mathbf{x})|$ and the inverse $\mathbf{G}(\mathbf{x})^{-1}$, we'll consider the cases where $q(\mathbf{x}) = p(\mathbf{x})$ and where $q(\mathbf{x}) \neq p(\mathbf{x})$ separately. Looking at **riemann-metric**, their evaluations are under the two branches, starting from Lines 13 and 19, of its only if-then-else fork. To speed up computation, we introduce an additional operator: Let $\star$ indicate a row-wise product between a matrix and a vector, e.g. $\mathbf{K}\star\mathbf{s}$ mul-

Algorithm 2 Riemann Hamiltonian and Gradient

function hamiltonian-and-gradient
input: $\mathcal{L}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$, $\mathbf{G}^{-1}$, $\log|\mathbf{G}|$, dg, $\mathbf{p}$
$H = \frac{1}{2}\mathbf{p}^T\mathbf{G}^{-1}\mathbf{p} + \frac{1}{2}\log|\mathbf{G}| - \mathcal{L}$
$\mathbf{w} = \mathbf{G}^{-1}\mathbf{p}$
$\frac{\partial H}{\partial \mathbf{x}} = \frac{1}{2}(\mathrm{dg} \circ \mathsf{diag}(\mathbf{G}^{-1})) - \frac{1}{2}(\mathbf{w} \circ \mathbf{w} \circ \mathrm{dg}) - \frac{\partial \mathcal{L}}{\partial \mathbf{x}}$
return $\{H, \frac{\partial H}{\partial \mathbf{x}}\}$

$\mathrm{dg} \doteq \mathsf{vec}[\frac{\partial \mathbf{G}(\mathbf{x})_{nn}}{\partial x_n}]$
// equation (12), but ignoring $\frac{N}{2}\log(2\pi)$ *as a cancelling constant in (8)*

tiplies s_1 with the first row of $\mathbf{K}$, s_2 with the second row, etc. If $\mathbf{S} \doteq \mathsf{diag}(\mathbf{s})$, the result is equal to (but faster to compute than) the matrix product $\mathbf{SK}$.

4.3.1. The inverse $\mathbf{G}(\mathbf{x})^{-1}$ *and determinant* $\log|\mathbf{G}(\mathbf{x})|$ *when* $q(\mathbf{x}) = p(\mathbf{x})$

If q is equal to the prior, then

$$\mathbf{G}(\mathbf{x})^{-1} = (\mathbf{\Lambda}(\mathbf{x}) + \mathbf{K}^{-1})^{-1} ,$$

for which we'll use the Sherman-Morrison-Woodbury formula. We'll expand the steps in more detail in Section 4.3.2 below. Briefly, the steps are to let $\mathbf{s} := \mathsf{diag}(\mathbf{\Lambda}^{\frac{1}{2}})$, to determine a stable Cholesky decomposition in $\mathbf{L}^{\mathrm{s}} = \mathsf{chol}(\mathbf{I} + \mathbf{K} \circ (\mathbf{s}\mathbf{s}^T))$ (note that $\mathbf{K} \circ (\mathbf{s}\mathbf{s}^T) = \mathbf{SKS}$, but that the former is faster to compute), and to use $\mathbf{L}^{\mathrm{s}}$ to back-solve $\mathbf{V} = \mathbf{L}^{\mathrm{s}} \backslash (\mathbf{K} \star \mathbf{s})$. Then[2]

$$\mathbf{G}(\mathbf{x})^{-1} := \mathbf{K} - \mathbf{V}^T\mathbf{V} ,$$

and using the precomputed Cholesky decomposition $\mathbf{L}^{\mathrm{K}} = \mathsf{chol}(\mathbf{K})$ of the kernel matrix,

$$\log|\mathbf{G}(\mathbf{x})| := -2\sum_{n=1}^{N}\log L^{\mathrm{K}}_{nn} + 2\sum_{n=1}^{N}\log L^{\mathrm{s}}_{nn} .$$

4.3.2. The inverse $\mathbf{G}(\mathbf{x})^{-1}$ *and determinant* $\log|\mathbf{G}(\mathbf{x})|$ *when* $q(\mathbf{x}) \neq p(\mathbf{x})$

To determine the determinant when q is not equal to the prior, we define

$$\mathbf{A}^{-1} \doteq \beta\mathbf{K}^{-1} + (1-\beta)\mathbf{\Sigma}^{-1}$$

so that $\mathbf{G}(\mathbf{x})^{-1} = (\mathbf{\Lambda}(\mathbf{x}) + \mathbf{A}^{-1})^{-1}$. If q is given by an EP approximation, then the precision matrix of q decomposes as $\mathbf{K}^{-1}$ plus a *diagonal* matrix $\widetilde{\mathbf{\Sigma}}$ containing the contributions from N approximate factors corresponding to the likelihood terms,

$$\mathbf{\Sigma}^{-1} = \mathbf{K}^{-1} + \widetilde{\mathbf{\Sigma}}^{-1} .$$

We choose the above notation to match that of the GPC approximation in Rasmussen and Williams's *Gaussian Processes for Machine Learning* book. Of course $\mathbf{\Sigma}$ needn't have this form, but as we use an EP approximation, we utilize it in aid of numerical stability. Then

$$\mathbf{A}^{-1} \doteq \mathbf{K}^{-1} + (1-\beta)\widetilde{\mathbf{\Sigma}}^{-1} = \mathbf{K}^{-1} + \mathbf{B}$$

for diagonal positive definite matrix $\mathbf{B} = (1-\beta)\widetilde{\mathbf{\Sigma}}^{-1}$. In Line 21 in Algorithm 3 we use

$$\mathbf{T} \doteq \mathbf{B}^{\frac{1}{2}} ,$$

and will do so below. To determine the inverse of $\mathbf{A}$, the Woodbury identity states

$$\begin{aligned}\mathbf{A} &= (\mathbf{K}^{-1} + \mathbf{TT})^{-1} \\ &= \mathbf{K} - \mathbf{KT}(\underbrace{\mathbf{I} + \mathbf{TKT}}_{\mathbf{L}^{\mathrm{t}}(\mathbf{L}^{\mathrm{t}})^T})^{-1}\mathbf{TK} ,\end{aligned}$$

with a Cholesky decomposition $\mathbf{L}^{\mathrm{t}} \doteq \mathsf{chol}(\mathbf{I} + \mathbf{TKT})$. The steps are to set $\mathbf{t} := \sqrt{1-\beta}\,\mathsf{diag}(\widetilde{\mathbf{\Sigma}}^{-\frac{1}{2}})$ as the vector, and then to determine $\mathbf{L}^{\mathrm{t}} := \mathsf{chol}(\mathbf{I} + \mathbf{K} \circ (\mathbf{t}\mathbf{t}^T))$, $\mathbf{V} := \mathbf{L}^{\mathrm{t}} \backslash (\mathbf{K} \star \mathbf{t})$, and $\mathbf{A} = \mathbf{K} - \mathbf{V}^T\mathbf{V}$. The log determinant of $\mathbf{A}$'s inverse follows from a similar identity,

$$\begin{aligned}\log|\mathbf{A}^{-1}| &= \log|\mathbf{K}^{-1} + \mathbf{TT}| \\ &= \log|\mathbf{I}| + \log|\mathbf{K}^{-1}| + \log|\mathbf{I} + \mathbf{TKT}| ,\end{aligned}$$

which will re-use the Cholesky decomposition of $\mathbf{I} + \mathbf{TKT}$ to give

$$\log|\mathbf{A}^{-1}| = -2 * \sum_{n=1}^{N}\log L^{\mathrm{K}}_{nn} + 2\sum_{n=1}^{N}\log L^{\mathrm{t}}_{nn} .$$

The same set of steps can be repeated to find $\mathbf{G}(\mathbf{x})^{-1}$ and its determinant. As $\mathbf{G}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{x}) + \mathbf{A}^{-1}$, let $\mathbf{L}^{\mathrm{a}} := \mathsf{chol}(\mathbf{I} + \mathbf{A} \circ (\mathbf{s}\mathbf{s}^T))$, with $\mathbf{s}$ defined as before. With $\mathbf{V} := \mathbf{L}^{\mathrm{a}} \backslash (\mathbf{A} \star \mathbf{s})$, we obtain $\mathbf{G}(\mathbf{x})^{-1} := \mathbf{A} - \mathbf{V}^T\mathbf{V}$ and

$$\log|\mathbf{G}(\mathbf{x})| := \log|\mathbf{A}^{-1}| + 2\sum_{n=1}^{N}\log L^{\mathrm{a}}_{nn} .$$

4.4. Derivatives of a probit log likelihood

The first, second, and third derivatives of the log likelihood are required in RMHMC. All derivatives appears in the gradient of $H(\mathbf{x}, \mathbf{p})$ with respect to $\mathbf{x}$ in Equation (12): the first derivate in the gradient of $\mathcal{L}(\mathbf{x})$, the second derivative in the metric tensor $\mathbf{G}(\mathbf{x}) \doteq -\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x})$, and the third derivative in the gradient of the metric tensor $\partial\mathbf{G}(\mathbf{x})/\partial x_n$.

The log likelihood for x_n, on observing $y_n \in \{-1, +1\}$, is the log probit function

$$\ell(x) = \log p(y|x) = \log\Phi(yx) ,$$

[2] As in the riemann-metric function in Algorithm 3, we use $\mathbf{V}$ as a "local variable" in this section, with a similar "local" use in Section 4.3.2.

Algorithm 3 Metric Tensor

1: **function** riemann-metric
2: **input:** $\mathbf{x}$, $\mathbf{y}$, $\mathbf{K}$, $\mathbf{L}^{\mathrm{K}}$, $\log|\mathbf{K}|$, $\boldsymbol{\mu}$, $\widetilde{\boldsymbol{\Sigma}}$, $\mathbf{L}^{\Sigma}$, $\log|\boldsymbol{\Sigma}|$, β
3: $\{\mathcal{L}, \frac{\partial\mathcal{L}}{\partial\mathbf{x}}\} :=$ **derivatives**$(\mathbf{x}, \mathbf{y}, \mathbf{L}^{\mathrm{K}}, \log|\mathbf{K}|, \boldsymbol{\mu}, \mathbf{L}^{\Sigma}, \log|\boldsymbol{\Sigma}|, \beta)$
4: *// compute the non-zero (diagonal) entries of* $\frac{\partial\mathbf{G}(\mathbf{x})}{\partial x_n}$ *as a vector* $\mathrm{d}\mathbf{g}$*:*
5: $\mathbf{z} := \mathbf{y} \circ \mathbf{x}$
6: $\mathbf{r} := -\frac{1}{2}\mathbf{z}^2 - \frac{1}{2}\log(2\pi) - \log\Phi(\mathbf{z})$ *// $\mathbf{r}$ is the log ratio of $\mathcal{N}(yx)/\Phi(yx)$, and $\mathbf{z}^2 \doteq \mathbf{z} \circ \mathbf{z}$, i.e. element-wise square*
7: $\boldsymbol{\lambda} := \mathbf{z} \circ \exp\{\mathbf{r}\} + \exp\{2\mathbf{r}\}$ *// $\exp\{\mathbf{r}\} \doteq \mathsf{vec}[\exp\{r_n\}]$, i.e. applied element-wise*
8: $\boldsymbol{\lambda} := \beta\boldsymbol{\lambda}$
9: $\mathrm{d}\mathbf{g} := \mathbf{y} \circ (1 - \mathbf{x}^2) \circ \exp\{\mathbf{r}\} - 3\mathbf{x} \circ \exp\{2\mathbf{r}\} - 2\mathbf{y} \circ \exp\{3\mathbf{r}\}$
10: $\mathrm{d}\mathbf{g} := \beta(\mathrm{d}\mathbf{g})$
11: *// compute the inverse $\mathbf{G}(\mathbf{x})^{-1}$ and its determinant:*
12: $\mathbf{s} := \sqrt{\boldsymbol{\lambda}}$ *// element-wise root*
13: **if** $\boldsymbol{\Sigma} = \mathbf{K}$ (i.e. $\widetilde{\boldsymbol{\Sigma}} = \mathbf{0}$) and $\boldsymbol{\mu} = \mathbf{0}$ **then**
14: $\quad\mathbf{L} := \mathsf{chol}(\mathbf{I} + \mathbf{K} \circ (\mathbf{s}\mathbf{s}^T))$ *// slower is $\mathbf{S} := \mathsf{diag}(\mathbf{s})$ and then $\mathbf{L} := \mathsf{chol}(\mathbf{I} + \mathbf{SKS})$*
15: $\quad\mathbf{V} = \mathbf{L}\backslash(\mathbf{K} \star \mathbf{s})$ *// slower is $\mathbf{V} = \mathbf{L}\backslash(\mathbf{SK})$*
16: $\quad$*// $\star$ indicates the row-wise product between $\mathbf{K}$ and $\mathbf{s}$, multiplying s_1 with the first row of $\mathbf{K}$, s_2 with $\mathbf{K}$'s second row, etc.*
17: $\quad\mathbf{G}^{-1} := \mathbf{K} - \mathbf{V}^T\mathbf{V}$;
18: $\quad\log|\mathbf{G}| := -2\sum_{n=1}^{N}\log L^{\mathrm{K}}_{nn} + 2\sum_{n=1}^{N}\log L_{nn}$
19: **else**
20: $\quad\mathbf{t} := \sqrt{1-\beta}\,\mathsf{diag}(\widetilde{\boldsymbol{\Sigma}}^{-\frac{1}{2}})$
21: $\quad\mathbf{L} := \mathsf{chol}(\mathbf{I} + \mathbf{K} \circ (\mathbf{t}\mathbf{t}^T))$ *// slower is $\mathbf{T} := \mathsf{diag}(\mathbf{t})$ and then $\mathbf{L} := \mathsf{chol}(\mathbf{I} + \mathbf{TKT})$*
22: $\quad\mathbf{V} := \mathbf{L}\backslash(\mathbf{K} \star \mathbf{t})$ *// slower is $\mathbf{V} = \mathbf{L}\backslash(\mathbf{TK})$*
23: $\quad\mathbf{A} = \mathbf{K} - \mathbf{V}^T\mathbf{V}$ *// using $\mathbf{A}^{-1} = \mathbf{K}^{-1} + (1-\beta)\widetilde{\boldsymbol{\Sigma}}^{-1}$*
24: $\quad\log|\mathbf{A}^{-1}| = -2 * \sum_{n=1}^{N}\log L^{\mathrm{K}}_{nn} + 2\sum_{n=1}^{N}\log L_{nn}$
25: $\quad\mathbf{L} := \mathsf{chol}(\mathbf{I} + \mathbf{A} \circ (\mathbf{s}\mathbf{s}^T))$ *// slower is $\mathbf{S} := \mathsf{diag}(\mathbf{s})$ and then $\mathbf{L} := \mathsf{chol}(\mathbf{I} + \mathbf{SAS})$*
26: $\quad\mathbf{V} := \mathbf{L}\backslash(\mathbf{A} \star \mathbf{s})$ *// slower is $\mathbf{V} = \mathbf{L}\backslash(\mathbf{SA})$*
27: $\quad\mathbf{G}^{-1} := \mathbf{A} - \mathbf{V}^T\mathbf{V}$ *// using $\mathbf{G} = \boldsymbol{\Lambda} + \mathbf{A}^{-1}$*
28: $\quad\log|\mathbf{G}| := \log|\mathbf{A}^{-1}| + 2\sum_{n=1}^{N}\log L_{nn}$
29: **end if**
30: **return** $\{\mathcal{L}, \frac{\partial\mathcal{L}}{\partial\mathbf{x}}, \mathbf{G}^{-1}, \log|\mathbf{G}|, \mathrm{d}\mathbf{g}\}$

where subscripts n are dropped for brevity. Let $\mathcal{N}(yx) \doteq \mathcal{N}(yx; 0, 1)$ denote a centred unit-variance Gaussian. Then the first derivative

$$\nabla_x \ell(x) = \frac{\mathcal{N}(yx)}{\Phi(yx)}$$

is given in Line 9 in the **derivatives** function in Algorithm 4.

Taking the derivative again (and multiplying by -1), the diagonal values of $\boldsymbol{\Lambda}(\mathbf{x})$ are

$$\Lambda_{nn} = \beta\left[yx\,\frac{\mathcal{N}(yx)}{\Phi(yx)} + \left(\frac{\mathcal{N}(yx)}{\Phi(yx)}\right)^2\right],$$

and are computed as a vector $\boldsymbol{\lambda}$ in Line 8 in the **riemann-metric** function in Algorithm 3.

In the derivative of the metric tensor $\partial\mathbf{G}(\mathbf{x})/\partial x_n$, the log likelihood contributes

$$\frac{\partial\Lambda_{nn}}{\partial x_n} = \beta\left[y(1-x^2)\,\frac{\mathcal{N}(yx)}{\Phi(yx)} - 3x\left(\frac{\mathcal{N}(yx)}{\Phi(yx)}\right)^2 - 2y\left(\frac{\mathcal{N}(yx)}{\Phi(yx)}\right)^3\right],$$

and is computed as a vector $\mathrm{d}\mathbf{g}$ in Line 10 in the **riemann-metric** function.

5. Results

The results presented here are form a baseline for the *Adaptive Resample-Move* algorithm in [5]. The evaluation is on the USPS 3-vs.-5 data set [6], using a covariance function $K_{mn} = k(\boldsymbol{\xi}_m, \boldsymbol{\xi}_n) = \sigma^2\exp(-\frac{1}{2}\|\boldsymbol{\xi}_m - \boldsymbol{\xi}_n\|^2/\ell^2)$ that correlates inputs $\boldsymbol{\xi}_m$ and $\boldsymbol{\xi}_n$ through a length scale $\ell = \exp(4.85)$ and amplitude parameter $\sigma = \exp(5.1)$.[3]

We ran Annealed Importance Sampling (AIS) [7] using different versions of Hamiltonian Monte Carlo (HMC) methods for the transition kernel. Such a highly correlated high-dimensional prior highlights some deficiencies in a basic HMC method, where mixing can be slow due to a sample's leapfrog trajectory oscillating up and down the sides of a valley of $\log\{p(\mathbf{y}|\mathbf{x})^\beta\mathrm{prior}(\mathbf{x})\}$, without actually progressing through it. To further aid AIS with different HMC methods, we additionally let AIS anneal from a Gaussian approximation $q(\mathbf{x})$ to the GPC posterior, instead of the prior. The approximation $q(\mathbf{x})$ was obtained with Expectation Propagation (EP).

Figure 2 compares the estimates of $\log Z$ obtained with AIS to the required computation time. The details of the methods are:

[3] On [6]'s entire $(\log\ell, \log\sigma)$-grid, this setting proved to be the hardest.

Algorithm 4 Derivatives of $\mathcal{L}_\beta(\mathbf{x})$ in Equation (4)

function derivatives
input: $\mathbf{x}$, $\mathbf{y}$, $\mathbf{L}^{\mathrm{K}}$, $\log|\mathbf{K}|$, $\boldsymbol{\mu}$, $\mathbf{L}^{\Sigma}$, $\log|\boldsymbol{\Sigma}|$, β
// compute the value of the log likelihood for the probit classification model and its derivative with respect to $\mathbf{x}$
$\mathbf{z} := \mathbf{y} \circ \mathbf{x}$
$L := \sum_{n=1}^{N} \log \Phi(z_n)$
$\mathcal{I} := \mathsf{indexes}(z_n > -15)$ *// normal regime; $\neg\mathcal{I}$ indexes asymptotic regime for numeric stability*
$\frac{\partial L}{\partial \mathbf{x}}(\mathcal{I}) := (2\pi)^{-1/2} \exp\{-\frac{1}{2}\mathbf{z}_{\mathcal{I}}^2\}/\Phi(\mathbf{z}_{\mathcal{I}})$ *// normal regime; element-wise division*
$\frac{\partial L}{\partial \mathbf{x}}(\neg\mathcal{I}) := -\mathbf{z}_{\neg\mathcal{I}} - 1/\mathbf{z}_{\neg\mathcal{I}} + 2/\mathbf{z}_{\neg\mathcal{I}}^3$ *// asymptotic regime; element-wise division*
$\frac{\partial L}{\partial \mathbf{x}} := \mathbf{y} \circ \frac{\partial L}{\partial \mathbf{x}}$
// add the derivatives with respect to the log prior
$\mathbf{f} = \mathbf{L}^{\mathrm{K}} \backslash \mathbf{x}$
if $\mathbf{L}^{\Sigma} = \mathbf{L}^{\mathrm{K}}$ and $\boldsymbol{\mu} = \mathbf{0}$ **then**
  $\mathcal{L} := \beta L - \frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\mathbf{f}^T\mathbf{f}$ *// equation (4)*
  $\frac{\partial \mathcal{L}}{\partial \mathbf{x}} := \beta \frac{\partial L}{\partial \mathbf{x}} - ((\mathbf{L}^{\mathrm{K}})^T \backslash \mathbf{f})$
else
  $\mathbf{f}' := \mathbf{L}^{\Sigma} \backslash (\mathbf{x} - \boldsymbol{\mu})$
  $\mathcal{L} := \beta L - \frac{N}{2}\log(2\pi) - \beta(\frac{1}{2}\log|\mathbf{K}| + \frac{1}{2}\mathbf{f}^T\mathbf{f}) - (1-\beta)(\frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\mathbf{f}'^T\mathbf{f}')$ *// equation (4)*
  $\frac{\partial \mathcal{L}}{\partial \mathbf{x}} := \beta \frac{\partial L}{\partial \mathbf{x}} + \beta((\mathbf{L}^{\mathrm{K}})^T \backslash \mathbf{f}) - (1-\beta)((\mathbf{L}^{\Sigma})^T \backslash \mathbf{f}')$
end if
return $\{\mathcal{L}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}}\}$

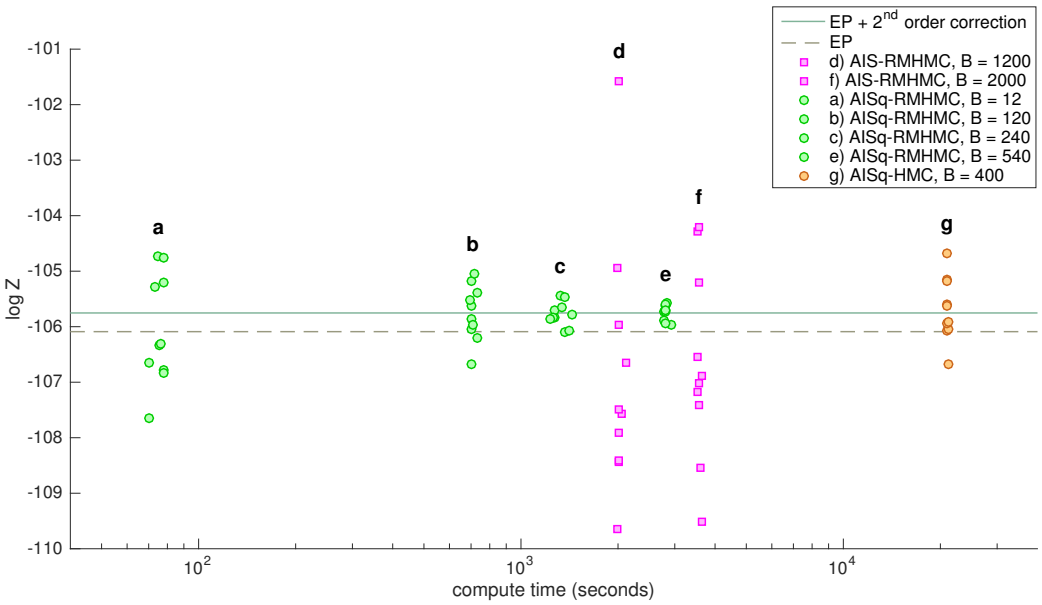

Figure 2: An extensive time-based comparison for GPC on the USPS 3-vs.-5 data set, using a highly correlated prior. From left to right, the evaluations are for (a) AISq-RMHMC using $B = 12$; (b) AISq-RMHMC using $B = 120$; (c) AISq-RMHMC using $B = 240$; (d) AIS-RMHMC using $B = 1200$; (e) AISq-RMHMC using $B = 540$; (f) AIS-RMHMC using $B = 2000$; (g) AISq-HMC using $B = 400$. The dotted line indicates EP's approximation, and the solid line a second order correction to the EP solution. A label that is starred indicates that an AIS method was aided by annealing from EP's $q(\mathbf{x})$ to $p(\mathbf{x}|\mathbf{y})$, and not from the prior, as is more commonly done.

AISq+HMC in (g) runs AIS from the EP's $q(\mathbf{x})$ at $\beta = 0$ to $p(\mathbf{x}|\mathbf{y})$ at $\beta = 1$ using intermediate distributions

$$p_\beta(\mathbf{x}) = \frac{1}{Z(\beta)} \left(\prod_n \Phi(y_n x_n) \cdot \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{K}) \right)^{\beta} q(\mathbf{x})^{1-\beta} . \tag{18}$$

Note that a starred label indicates that the estimates were aided by $q(\mathbf{x})$. A HMC transition kernel with $l_{\max} = 200$ *leapfrog* steps is used at each $\beta \in [0, 1]$ value. AIS's β-grid is a geometric progression over $B = 400$ β-values. Plot (g) used a step size $\epsilon = 0.02$ per proposal; both $l_{\max}$ and ϵ were carefully tuned to the problem. The simplest AIS-HMC version, which anneals from $p(\mathbf{x})$ and not $q(\mathbf{x})$, didn't obtain estimates inside the bounds of Figure 2, and

is excluded.

AIS+RMHMC in (j) and (l) anneals from $p(\mathbf{x})$, and replaces HMC with a more advanced RMHMC that uses $\epsilon = 0.1$ and $l_{\max} = 10$ leapfrog steps per proposal at each β value.

AISq+RMHMC in (c*), (f*), (g*) and (k*) anneals from $q(\mathbf{x})$ using a RMHMC kernel ($\epsilon = 0.1, l_{\max} = 10$).

It is known that the EP estimate of $\log Z$ is remarkably accurate for this problem [6], hence EP's $\log Z$ estimate and its a second-order corrected estimate [9] are given for reference.

References

[1] M. Girolami, B. Calderhead, Riemann manifold Langevin and Hamiltonian Monte Carlo methods, Journal of the Royal Statistical Society, Series B 73 (2) (2011) 123–214.

[2] M. Girolami, B. Calderhead, S. A. Chin, Riemannian manifold Hamiltonian Monte Carlo, arXiv:0907.1100 (2009).

[3] R. M. Neal, MCMC using Hamiltonian dynamics, Handbook of Markov Chain Monte Carlo 54 (2010) 113–162.

[4] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.

[5] M. Fraccaro, U. Paquet, O. Winther, An adaptive resample-move algorithm for estimating normalizing constants (2016).

[6] M. Kuss, C. E. Rasmussen, Assessing approximate inference for binary Gaussian process classification, Journal of Machine Learning Research 6 (2005) 1679–1704.

[7] R. M. Neal, Annealed importance sampling, Statistics and Computing 11 (2) (2001) 125–139.

[8] F. Hamze, N. de Freitas, Hot coupling: A particle approach to inference and normalization on pairwise undirected graphs of arbitrary topology, in: Advances in Neural Information Processing Systems 18, 2005.

[9] M. Opper, U. Paquet, O. Winther, Perturbative corrections for approximate inference in Gaussian latent variable models, Journal of Machine Learning Research 14 (Sep) (2013) 2857–2898.

[10] M. D. Hoffman, A. Gelman, The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo, Journal of Machine Learning Research 15 (Apr) (2014) 1593–1623.