Elliptical slice sampling

Iain Murray
University of Toronto

Ryan Prescott Adams
University of Toronto

David J.C. MacKay
University of Cambridge

Abstract

Many probabilistic models introduce strong dependencies between variables using a latent multivariate Gaussian distribution or a Gaussian process. We present a new Markov chain Monte Carlo algorithm for performing inference in models with multivariate Gaussian priors. Its key properties are: 1) it has simple, generic code applicable to many models, 2) it has no free parameters, 3) it works well for a variety of Gaussian process based models. These properties make our method ideal for use while model building, removing the need to spend time deriving and tuning updates for more complex algorithms.

1 Introduction

The multivariate Gaussian distribution is commonly used to specify a priori beliefs about dependencies between latent variables in probabilistic models. The parameters of such a Gaussian may be specified directly, as in graphical models and Markov random fields, or implicitly as the marginals of a Gaussian process (GP). Gaussian processes may be used to express concepts of spatial or temporal coherence, or may more generally be used to construct Bayesian kernel methods for non-parametric regression and classification. Rasmussen and Williams (2006) provide a recent review of GPs.

Inferences can only be calculated in closed form for the simplest Gaussian latent variable models. Recent work shows that posterior marginals can sometimes be well approximated with deterministic methods (Kuss and Rasmussen, 2005; Rue et al., 2009). Markov chain Monte Carlo (MCMC) methods represent joint posterior distributions with samples (e.g. Neal, 1993). MCMC can be slower but applies more generally.

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

In some circumstances MCMC provides good results with minimal model-specific implementation. Gibbs sampling, in particular, is frequently used to sample from probabilistic models in a straightforward way, updating one variable at a time. In models with strong dependencies among variables, including many with Gaussian priors, Gibbs sampling is known to perform poorly. Several authors have previously addressed the issue of sampling from models containing strongly correlated Gaussians, notably the recent work of Titsias et al. (2009). In this paper we provide a technique called *elliptical slice sampling* that is simpler and often faster than other methods, while also removing the need for preliminary tuning runs. Our method provides a drop-in replacement for MCMC samplers of Gaussian models that are currently using Gibbs or Metropolis–Hastings and we demonstrate empirical success against competing methods with several different GP-based likelihood models.

2 Elliptical slice sampling

Our objective is to sample from a posterior distribution over latent variables that is proportional to the product of a multivariate Gaussian prior and a likelihood function that ties the latent variables to the observed data. We will use $\mathbf{f}$ to indicate the vector of latent variables that we wish to sample and denote a zero-mean Gaussian distribution with covariance Σ by

$$\mathcal{N}(\mathbf{f}; 0, \Sigma) \equiv |2\pi\Sigma|^{-1/2} \exp\left(-\tfrac{1}{2}\mathbf{f}^\top \Sigma^{-1}\mathbf{f}\right). \quad (1)$$

We also use $\mathbf{f} \sim \mathcal{N}(0, \Sigma)$ to state that $\mathbf{f}$ is drawn from a distribution with the density in (1). Gaussians with non-zero means can simply be shifted to have zero-mean with a change of variables; an example will be given in Section 3.3. We use $L(\mathbf{f}) = p(\text{data}\,|\,\mathbf{f})$ to denote the likelihood function so that our *target distribution* for the MCMC sampler is

$$p^\star(\mathbf{f}) = \frac{1}{Z}\,\mathcal{N}(\mathbf{f}; 0, \Sigma)\,L(\mathbf{f}), \quad (2)$$

where Z is the normalization constant, or the marginal likelihood, of the model.

Our starting point is a Metropolis–Hastings method introduced by Neal (1999). Given an initial state $\mathbf{f}$, a

new state

$$\mathbf{f}' = \sqrt{1-\epsilon^2}\,\mathbf{f} + \epsilon\,\boldsymbol{\nu}, \qquad \boldsymbol{\nu} \sim \mathcal{N}(0, \Sigma) \tag{3}$$

is proposed, where $\epsilon \in [-1, 1]$ is a step-size parameter. The proposal is a sample from the prior for $\epsilon = 1$ and more conservative for values closer to zero. The move is accepted with probability

$$p(\text{accept}) = \min(1,\, L(\mathbf{f}')/L(\mathbf{f})), \tag{4}$$

otherwise the next state in the chain is a copy of $\mathbf{f}$.

Neal reported that for some Gaussian process classifiers the Metropolis–Hastings method was many times faster than Gibbs sampling. The method is also simpler to implement and can immediately be applied to a much wider variety of models with Gaussian priors.

A drawback, identified by Neal (1999), is that the step-size ϵ needs to be chosen appropriately for the Markov chain to mix efficiently. This may require preliminary runs. Usually parameters of the covariance Σ and likelihood function L are also inferred from data. Different step-size parameters may be needed as the model parameters are updated. It would be desirable to automatically search over the step-size parameter, while maintaining a valid algorithm.

For a fixed auxiliary random draw, $\boldsymbol{\nu}$, the locus of possible proposals by varying $\epsilon \in [-1, 1]$ in (3) is half of an ellipse. A more natural parameterization is

$$\mathbf{f}' = \boldsymbol{\nu} \sin\theta + \mathbf{f} \cos\theta, \tag{5}$$

defining a full ellipse passing through the current state $\mathbf{f}$ and the auxiliary draw $\boldsymbol{\nu}$. For a fixed θ there is an equivalent ϵ that gives the same proposal distribution in the original algorithm. However, if we can search over the step-size, the full ellipse gives a richer choice of updates for a given $\boldsymbol{\nu}$.

2.1 Sampling an alternative model

'Slice sampling' (Neal, 2003) provides a way to sample along a line with an adaptive step-size. Proposals are drawn from an interval or 'bracket' which, if too large, is shrunk automatically until an acceptable point is found. There are also ways to automatically enlarge small initial brackets. Naively applying these adaptive algorithms to select the value of ϵ in (3) or θ in (5) does not result in a Markov chain transition operator with the correct stationary distribution. The locus of states is defined using the current position $\mathbf{f}$, which upsets the reversibility and correctness of the update.

We would like to construct a valid Markov chain transition operator on the ellipse of states that uses slice sampling's existing ability to adaptively pick step sizes.

Input: current state $\mathbf{f}$, a routine that samples from $\mathcal{N}(0, \Sigma)$, log-likelihood function $\log L$.

Output: a new state $\mathbf{f}'$. When $\mathbf{f}$ is drawn from $p^\star(\mathbf{f}) \propto \mathcal{N}(\mathbf{f}; 0, \Sigma)\, L(\mathbf{f})$, the marginal distribution of $\mathbf{f}'$ is also $p^\star$.

1. Sample from $p(\,\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta \,|\, (\boldsymbol{\nu}_0 \sin\theta + \boldsymbol{\nu}_1 \cos\theta = \mathbf{f})\,)$:
$$\begin{aligned} \theta &\sim \text{Uniform}[0, 2\pi] \\ \boldsymbol{\nu} &\sim \mathcal{N}(0, \Sigma) \\ \boldsymbol{\nu}_0 &\leftarrow \mathbf{f}\sin\theta + \boldsymbol{\nu}\cos\theta \\ \boldsymbol{\nu}_1 &\leftarrow \mathbf{f}\cos\theta - \boldsymbol{\nu}\sin\theta \end{aligned}$$
2. Update $\theta \in [0, 2\pi]$ using slice sampling (Neal, 2003) on:
$$p^\star(\theta \,|\, \boldsymbol{\nu}_0, \boldsymbol{\nu}_1) \propto L(\boldsymbol{\nu}_0 \sin\theta + \boldsymbol{\nu}_1 \cos\theta)$$
3. **return** $\mathbf{f}' = \boldsymbol{\nu}_0 \sin\theta + \boldsymbol{\nu}_1 \cos\theta$

Figure 1: Intuition behind elliptical slice sampling. This is a valid algorithm, but will be adapted (Figure 2).

We will first intuitively construct a valid method by positing an augmented probabilistic model in which the step-size is a variable. Standard slice sampling algorithms then apply to that model. We will then adjust the algorithm for our particular setting to provide a second, slightly tidier algorithm.

Our augmented probabilistic model replaces the original latent variable with prior $\mathbf{f} \sim \mathcal{N}(0, \Sigma)$ with

$$\begin{aligned} \boldsymbol{\nu}_0 &\sim \mathcal{N}(0, \Sigma) \\ \boldsymbol{\nu}_1 &\sim \mathcal{N}(0, \Sigma) \\ \theta &\sim \text{Uniform}[0, 2\pi] \\ \mathbf{f} &= \boldsymbol{\nu}_0 \sin\theta + \boldsymbol{\nu}_1 \cos\theta. \end{aligned} \tag{6}$$

The marginal distribution over the original latent variable $\mathbf{f}$ is still $\mathcal{N}(0, \Sigma)$, so the distribution over data is identical. However, we can now sample from the posterior over the new latent variables:

$$p^\star(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta) \propto \mathcal{N}(\boldsymbol{\nu}_0; 0, \Sigma)\, \mathcal{N}(\boldsymbol{\nu}_1; 0, \Sigma)\, L(\mathbf{f}(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta)),$$

and use the values of $\mathbf{f}$ deterministically derived from these samples. Our first approach applies two Monte Carlo transition operators that leave the new latent posterior invariant.

Operator 1: jointly resample the latents $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta$ given the constraint that $\mathbf{f}(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \theta)$ is unchanged. Because the effective variable of interest doesn't change, the likelihood does not affect this conditional distribution, so the update is generic and easy to implement.

Operator 2: use a standard slice sampling algorithm to update the step-size θ given the other variables.

The resulting algorithm is given in Figure 1. The auxiliary model construction makes the link to slice sampling explicit, which makes it easy to understand the validity of the approach. However, the algorithm

can be neater and the $[0, 2\pi]$ range for slice sampling is unnatural on an ellipse. The algorithm that we will present in detail results from eliminating $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ and a different way of setting slice sampling's initial proposal range. The precise connection will be given in Section 2.4. A more direct, technical proof that the equilibrium distribution of the Markov chain is the target distribution is presented in Section 2.3.

Elliptical slice sampling, our proposed algorithm is given in Figure 2, which includes the details of the slice sampler. An example run is illustrated in Figure 3(a–d). Even for high-dimensional problems, the states considered within one update lie in a two-dimensional plane. In high dimensions $\mathbf{f}$ and $\boldsymbol{\nu}$ are likely to have similar lengths and be an angle of $\pi/2$ apart. Therefore the ellipse will typically be fairly close to a circle, although this is not required for the validity of the algorithm.

As intended, our slice sampling approach selects a new location on the randomly generated ellipse in (5). There are no rejections: the new state $\mathbf{f}'$ is never equal to the current state $\mathbf{f}$ unless that is the only state on the ellipse with non-zero likelihood. The algorithm proposes the angle θ from a bracket $[\theta_{\min}, \theta_{\max}]$ which is shrunk exponentially quickly until an acceptable state is found. Thus the step size is effectively adapted on each iteration for the current $\boldsymbol{\nu}$ and Σ.

2.2 Computational cost

Drawing $\boldsymbol{\nu}$ costs $\mathcal{O}(N^3)$, for N-dimensional $\mathbf{f}$ and general Σ. The usual implementation of a Gaussian sampler would involve caching a (Cholesky) decomposition of Σ, such that draws on subsequent iterations cost $\mathcal{O}(N^2)$. For some problems with special structure drawing samples from the Gaussian prior can be cheaper.

In many models the Gaussian prior distribution captures dependencies: the observations are independent conditioned on $\mathbf{f}$. In these cases, computing $L(\mathbf{f})$ will cost $\mathcal{O}(N)$ computation. As a result, drawing the $\boldsymbol{\nu}$ random variate will be the dominant cost of the update in many high-dimensional problems. In these cases the extra cost of elliptical slice sampling over Neal's Metropolis–Hastings algorithm will be small.

As a minor performance improvement, our implementation optionally accepts the log-likelihood of the initial state, if known from a previous update, so that it doesn't need to be recomputed in step 2.

2.3 Validity

Elliptical slice sampling considers settings of an angle variable, θ. Figure 2 presented the algorithm as it would be used: there is no need to index or remember the visited angles. For the purposes of analysis we

Input: current state $\mathbf{f}$, a routine that samples from $\mathcal{N}(0, \Sigma)$, log-likelihood function $\log L$.

Output: a new state $\mathbf{f}'$. When $\mathbf{f}$ is drawn from $p^\star(\mathbf{f}) \propto \mathcal{N}(\mathbf{f}; 0, \Sigma)\, L(\mathbf{f})$, the marginal distribution of $\mathbf{f}'$ is also $p^\star$.

1. Choose ellipse: $\boldsymbol{\nu} \sim \mathcal{N}(0, \Sigma)$
2. Log-likelihood threshold:
$$u \sim \text{Uniform}[0, 1]$$
$$\log y \leftarrow \log L(\mathbf{f}) + \log u$$
3. Draw an initial proposal, also defining a bracket:
$$\theta \sim \text{Uniform}[0, 2\pi]$$
$$[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi,\ \theta]$$
4. $\mathbf{f}' \leftarrow \mathbf{f}\cos\theta + \boldsymbol{\nu}\sin\theta$
5. **if** $\log L(\mathbf{f}') > \log y$ **then:**
6. Accept: **return** $\mathbf{f}'$
7. **else:**
Shrink the bracket and try a new point:
8. **if** $\theta < 0$ **then:** $\theta_{\min} \leftarrow \theta$ **else:** $\theta_{\max} \leftarrow \theta$
9. $\theta \sim \text{Uniform}[\theta_{\min}, \theta_{\max}]$
10. **GoTo** 4.

Figure 2: The elliptical slice sampling algorithm.

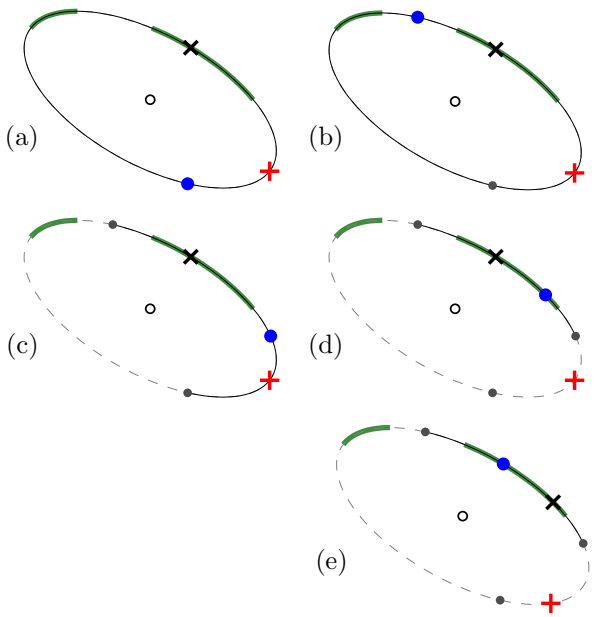

Figure 3: **(a)** The algorithm receives $\mathbf{f}$=✕ as input. Step 1 draws auxiliary variate $\boldsymbol{\nu}$=+, defining an ellipse centred at the origin (○). Step 2: a likelihood threshold defines the 'slice' (▬). Step 3: an initial proposal ● is drawn, in this case not on the slice. **(b)** The first proposal defined both edges of the $[\theta_{\min}, \theta_{\max}]$ bracket; the second proposal (●) is also drawn from the whole range. **(c)** One edge of the bracket (—) is moved to the last rejected point such that ✕ is still included. Proposals are made with this shrinking rule until one lands on the slice. **(d)** The proposal here (●) is on the slice and is returned as $\mathbf{f}'$. **(e)** Shows the reverse configuration discussed in Section 2.3: ✕ is the input $\mathbf{f}'$, which with auxiliary $\boldsymbol{\nu}'$=+ defines the same ellipse. The brackets and first three proposals (•) are the same. The final proposal (●) is accepted, a move back to $\mathbf{f}$.

will denote the ordered sequence of angles considered during the algorithm by $\{\theta_k\}$ with $k=1..K$.

We first identify the joint distribution over a state drawn from the target distribution (2) and the other random quantities generated by the algorithm:

$$p(\mathbf{f}, y, \boldsymbol{\nu}, \{\theta_k\}) = p^\star(\mathbf{f})\, p(y\,|\,\mathbf{f})\, p(\boldsymbol{\nu})\, p(\{\theta_k\}\,|\,\mathbf{f}, \boldsymbol{\nu}, y)$$
$$= \frac{1}{Z}\mathcal{N}(\mathbf{f}; 0, \Sigma)\,\mathcal{N}(\boldsymbol{\nu}; 0, \Sigma)\, p(\{\theta_k\}\,|\,\mathbf{f}, \boldsymbol{\nu}, y), \quad (7)$$

where the vertical level y was drawn uniformly in $[0, L(\mathbf{f})]$, that is, $p(y\,|\,\mathbf{f}) = 1/L(\mathbf{f})$. The final term, $p(\{\theta_k\}\,|\,\mathbf{f}, \boldsymbol{\nu}, y)$, is a distribution over a random-sized set of angles, defined by the stopping rule of the algorithm.

Given the random variables in (7) the algorithm deterministically computes positions, $\{\mathbf{f}_k\}$, accepting the first one that satisfies a likelihood constraint. More generally each angle specifies a rotation of the two a priori Gaussian variables:

$$\begin{aligned} \boldsymbol{\nu}_k &= \boldsymbol{\nu}\cos\theta_k - \mathbf{f}\sin\theta_k \\ \mathbf{f}_k &= \boldsymbol{\nu}\sin\theta_k + \mathbf{f}\cos\theta_k, \quad k = 1..K. \end{aligned} \quad (8)$$

For any choice of θ_k this deterministic transformation has unit Jacobian. Any such rotation also leaves the joint prior probability invariant,

$$\mathcal{N}(\boldsymbol{\nu}_k; 0, \Sigma)\,\mathcal{N}(\mathbf{f}_k; 0, \Sigma) = \mathcal{N}(\boldsymbol{\nu}; 0, \Sigma)\,\mathcal{N}(\mathbf{f}; 0, \Sigma) \quad (9)$$

for all k, which can easily be verified by substituting values into the Gaussian form (1).

It is often useful to consider how an MCMC algorithm could make a *reverse transition* from the final state $\mathbf{f}'$ back to the initial state $\mathbf{f}$. The final state $\mathbf{f}' = \mathbf{f}_K$ was the result of a rotation by θ_K in (8). Given an initial state of $\mathbf{f}' = \mathbf{f}_K$, the algorithm could generate $\boldsymbol{\nu}' = \boldsymbol{\nu}_K$ in step 1. Then a rotation of $-\theta_K$ would return back to the original $(\mathbf{f}, \boldsymbol{\nu})$ pair. Moreover, the same ellipse of states is accessible and rotations of $\theta_k - \theta_K$ will reproduce any intermediate $\mathbf{f}_{k<K}$ locations visited by the initial run of the algorithm.

In fact, the algorithm is *reversible*:

$$p(\mathbf{f}, y, \boldsymbol{\nu}, \{\theta_k\}) = p(\mathbf{f}', y, \boldsymbol{\nu}', \{\theta'_k\}), \quad (10)$$

the equilibrium probability of a forwards draw (7) is the same as the probability of starting at $\mathbf{f}'$, drawing the same y (possible because $L(\mathbf{f}') > y$), $\boldsymbol{\nu}' = \boldsymbol{\nu}_K$ and

$$\text{angles, } \theta'_k = \begin{cases} \theta_k - \theta_K & k < K \\ -\theta_K & k = K, \end{cases} \quad (11)$$

resulting in the original state $\mathbf{f}$ being returned. The reverse configuration corresponding to the result of a forwards run in Figure 3(d) is illustrated in Figure 3(e). Substituting (9) into (7) shows that ensuring that the forward and reverse angles are equally probable,

$$p(\{\theta_k\}\,|\,\mathbf{f}, \boldsymbol{\nu}, y) = p(\{\theta'_k\}\,|\,\mathbf{f}', \boldsymbol{\nu}', y), \quad (12)$$

results in the reversible property (10).

The algorithm does satisfy (12): The probability of the first angle is always $1/2\pi$. If more angles were considered before an acceptable state was found, these angles were drawn with probabilities $1/(\theta_{\max} - \theta_{\min})$. Whenever the bracket was shrunk in step 8, the side to shrink must have been chosen such that $\mathbf{f}_K$ remained selectable as it was selected later. The reverse transition uses the same intermediate proposals, making the same rejections with the same likelihood threshold, y. Because the algorithm explicitly includes the initial state, which in reverse is $\mathbf{f}_K$ at $\theta' = 0$, the reverse transition involves the same set of shrinking decisions as the forwards transitions. As the same brackets are sampled, the $1/(\theta_{\max} - \theta_{\min})$ probabilities for drawing angles are the same for the forwards and reverse transitions.

The reversibility of the transition operator (10) implies that the target posterior distribution (2) is a stationary distribution of the Markov chain. Drawing $\mathbf{f}$ from the stationary distribution and running the algorithm draws a sample from the joint auxiliary distribution (7). The deterministic transformations in (8) and (11) have unit Jacobian, so the probability density of obtaining a joint draw corresponding to $(\mathbf{f}', y, \boldsymbol{\nu}', \{\theta'_k\})$ is equal to the probability given by (7) for the original variables. The reversible property in (10) shows that this is the same probability as generating the variables by first generating $\mathbf{f}'$ from the target distribution and generating the remaining quantities using the algorithm. Therefore, the marginal probability of $\mathbf{f}'$ is given by the target posterior (2).

Given the first angle, the distribution over the first proposed move is $\mathcal{N}(\mathbf{f}\cos\theta,\, \Sigma\sin^2\theta)$. Therefore, there is a non-zero probability of transitioning to any region that has non-zero probability under the posterior. This is enough to ensure that, formally, the chain is irreducible and aperiodic (Tierney, 1994). Therefore, the Markov chain has a unique stationary distribution and repeated applications of elliptical slice sampling to an arbitrary starting point will asymptotically lead to points drawn from the target posterior distribution (2).

2.4 Slice sampling variants

There is some amount of choice in how the slice sampler on the ellipse could be set up. Other methods for proposing angles could have been used, as long as they satisfied the reversible condition in (12). The particular algorithm proposed in Figure 2 is appealing because it is simple and has no free parameters.

The algorithm must choose the initial edges of the bracket $[\theta_{\min}, \theta_{\max}]$ randomly. It would be aesthetically pleasing to place the edges of the bracket at the opposite side of the ellipse to the current position, at $\pm\pi$. However this deterministic bracket placement would not be reversible and gives an invalid algorithm.

The edge of a randomly-chosen bracket could lie on the 'slice', the acceptable region of states. Our recommended elliptical slice sampling algorithm, Figure 2, would accept this point. The initially-presented algorithm, Figure 1, effectively randomly places the endpoints of the bracket but without checking this location for acceptability. Apart from this small change, it can be shown that the algorithms are equivalent.

In typical problems the slice will not cover the whole ellipse. For example, if $\mathbf{f}$ is a representative sample from a posterior, often $-\mathbf{f}$ will not be. Increasing the probability of proposing points close to the current state may increase efficiency. One way to do this would be to shrink the bracket more aggressively (Skilling and MacKay, 2003). Another would be to derive a model from the auxiliary variable model (6), but with a non-uniform distribution on θ. Another way would be to randomly position an initial bracket of width less than 2π — the code that we provide optionally allows this. However, as explained in section 2.2, for high-dimensional problems such tuning will often only give small improvements. For smaller problems we have seen it possible to improve the cpu-time efficiency of the algorithm by around two times.

Another possible line of research is methods for biasing proposals away from the current state. For example the 'over-relaxed' methods discussed by Neal (2003) have a bias towards the opposite side of the slice from the current position. In our context it may be desirable to encourage moves close to $\theta = \pi/2$, as these moves are independent of the previous position. These proposals are only likely to be useful when the likelihood terms are very weak, however. In the limit of sampling from the prior due to a constant likelihood, the algorithm already samples reasonably efficiently. To see this, consider the distribution over the outcome after N iterations initialized at $\mathbf{f}^0$:

$$\mathbf{f}^N = \mathbf{f}^0 \prod_{n=1}^{N} \cos\theta^n + \sum_{m=1}^{N} \boldsymbol{\nu}^m \sin\theta^m \prod_{n=m+1}^{N} \cos\theta^n,$$

where $\boldsymbol{\nu}^n$ and θ^n are values drawn at iteration n. Only one angle is drawn per iteration when sampling from the prior, because the first proposal is always accepted. The only dependence on the initial state is the first term, the coefficient of which shrinks towards zero exponentially quickly.

2.5 Limitations

A common modeling situation is that an unknown constant offset, $c \sim \mathcal{N}(0, \sigma_m^2)$, has been added to the entire latent vector $\mathbf{f}$. The resulting variable, $\mathbf{g} = \mathbf{f} + c$, is still Gaussian distributed, with the constant σ_m^2 added to every element of the covariance matrix. Neal (1999) identified that this sort of covariance will not tend to produce useful auxiliary draws $\boldsymbol{\nu}$. An iteration of the Markov chain can only add a nearly-constant shift to the current state. Indeed, covariances with large constant terms are generally problematic as they tend to be poorly conditioned. Instead, large offsets should be modeled and sampled as separate variables.

No algorithm can sample effectively from arbitrary distributions. As any distribution can be factored as in (2), there exist likelihoods $L(\mathbf{f})$ for which elliptical slice sampling is not effective. Many Gaussian process applications have strong prior smoothness constraints and relatively weak likelihood constraints. This important regime is where we focus our empirical comparison.

3 Related work

Elliptical slice sampling builds on a Metropolis–Hastings (M–H) update proposed by Neal (1999). Neal reported that the original update performed moderately better than using a more obvious M–H proposal,

$$\mathbf{f}' = \mathbf{f} + \epsilon\,\boldsymbol{\nu}, \qquad \boldsymbol{\nu} \sim \mathcal{N}(0, \Sigma), \tag{13}$$

and much better than Gibbs sampling for Gaussian process classification. Neal also proposed using Hybrid/Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 1993), which can be very effective, but requires tuning and the implementation of gradients. We now consider some other alternatives that have similar requirements to elliptical slice sampling.

3.1 'Conventional' slice sampling

Elliptical slice sampling builds on the family of methods introduced by Neal (2003). Several of the existing slice sampling methods would also be easy to apply: they only require point-wise evaluation of the posterior up to a constant. These methods do have step-size parameters, but unlike simple Metropolis methods, typically the performance of slice samplers does not crucially rely on carefully setting free parameters.

The most popular generic slice samplers use simple univariate updates, although applying these directly to $\mathbf{f}$ would suffer the same slow convergence problems as Gibbs sampling. While Agarwal and Gelfand (2005) have applied slice sampling for sampling parameters in Gaussian spatial process models, they assumed a

linear-Gaussian observation model. For non-Gaussian data it was suggested that "there seems to be little role for slice sampling."

Elliptical slice sampling changes all of the variables in $\mathbf{f}$ at once, although there are potentially better ways of achieving this. An extensive search space of possibilities includes the suggestions for multivariate updates made by Neal (2003).

One simple possible slice sampling update performs a univariate update along a random line traced out by varying ϵ in (13). As the M–H method based on the line worked less well than that based on an ellipse, one might also expect a line-based slice sampler to perform less well. Intuitively, in high dimensions much of the mass of a Gaussian distribution is in a thin ellipsoidal shell. A straight line will more rapidly escape this shell than an ellipse passing through two points within it.

3.2 Control variables

Titsias et al. (2009) introduced a sampling method inspired by sparse Gaussian process approximations. M control variables $\mathbf{f}_c$ are introduced such that the joint prior $p(\mathbf{f}, \mathbf{f}_c)$ is Gaussian, and that $\mathbf{f}$ still has marginal prior $\mathcal{N}(0, \Sigma)$. For Gaussian process models a parametric family of joint covariances was defined, and the model is optimized so that the control variables are informative about the original variables: $p(\mathbf{f} \,|\, \mathbf{f}_c)$ is made to be highly peaked. The optimization is a pre-processing step that occurs before sampling begins.

The idea is that the small number of control variables $\mathbf{f}_c$ will be less strongly coupled than the original variables, and so can be moved individually more easily than the components of $\mathbf{f}$. A proposal involves resampling one control variable from the conditional prior and then resampling $\mathbf{f}$ from $p(\mathbf{f} \,|\, \mathbf{f}_c)$. This move is accepted or rejected with the Metropolis–Hastings rule.

Although the method is inspired by an approximation used for large datasets, the accept/reject step uses the full model. After $\mathcal{O}(N^3)$ pre-processing it costs $\mathcal{O}(N^2)$ to evaluate a proposed change to the N-dimensional vector $\mathbf{f}$. One 'iteration' in the paper consisted of an update for each control variable and so costs $\mathcal{O}(MN^2)$ — roughly M elliptical slice sampling updates. The control method uses fewer likelihood evaluations per iteration, although has some different minor costs associated with book-keeping of the control variables.

3.3 Local updates

In some applications it may make sense to update only a subset of the latent variables at a time. This might help for computational reasons given the $\mathcal{O}(N^2)$ scaling for drawing samples of subsets of size N. Titsias et al. (2009) also identified suitable subsets for local updates and then investigated sampling proposals from the conditional Gaussian prior.

In fact, local updates can be combined with any transition operator for models with Gaussian priors. If $\mathbf{f}_A$ is a subset of variables to update and $\mathbf{f}_B$ are the remaining variables, we can write the prior as:

$$\begin{bmatrix}\mathbf{f}_A \\ \mathbf{f}_B\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}\Sigma_{A,A} & \Sigma_{A,B} \\ \Sigma_{B,A} & \Sigma_{B,B}\end{bmatrix}\right) \tag{14}$$

and the conditional prior is:

$$p(\mathbf{f}_A \,|\, \mathbf{f}_B) = \mathcal{N}(\mathbf{f}_A;\, \mathbf{m}, S), \text{ where}$$
$$\mathbf{m} = \Sigma_{A,B}\Sigma_{B,B}^{-1}\mathbf{f}_B, \text{ and } S = \Sigma_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A}.$$

A change of variables $\mathbf{g} = \mathbf{f}_A - \mathbf{m}$ allows us to express the conditional posterior as: $p^\star(\mathbf{g}) \propto \mathcal{N}(\mathbf{g};\, 0,\, S)\, L\big(\left[\begin{smallmatrix}\mathbf{g}+\mathbf{m} \\ \mathbf{f}_B\end{smallmatrix}\right]\big)$. We can then apply elliptical slice sampling, or any alternative, to update $\mathbf{g}$ (and thus $\mathbf{f}_A$). Updating groups of variables according to their conditional distributions is a standard way of sampling from a joint distribution.

4 Experiments

We performed an empirical comparison on three Gaussian process based probabilistic modeling tasks. Only a brief description of the models and methods can be given here. Full code to reproduce the results is provided as supplementary material.

4.1 Models

Each of the models associates a dimension of the latent variable, f_n, with an 'input' or 'feature' vector $\mathbf{x}_n$. The models in our experiments construct the covariance from the inputs using the most common method,

$$\Sigma_{ij} \;=\; \sigma_f^2 \exp\big(-\tfrac{1}{2}\textstyle\sum_{d=1}^{D}(x_{d,i} - x_{d,j})^2/\ell^2\big), \tag{15}$$

the squared-exponential or "Gaussian" covariance. This covariance has "lengthscale" parameter ℓ and an overall "signal variance" σ_f^2. Other covariances may be more appropriate in many modeling situations, but our algorithm would apply unchanged.

Gaussian regression: given observations $\mathbf{y}$ of the latent variables with Gaussian noise of variance σ_n^2,

$$L_r(\mathbf{f}) = \textstyle\prod_n \mathcal{N}(y_n; f_n, \sigma_n^2), \tag{16}$$

the posterior is Gaussian and so fully tractable. We use this as a simple test that the method is working correctly. Differences in performance on this task will also give some indication of performance with a simple log-concave likelihood function.

We generated ten synthetic datasets with input feature dimensions from one to ten. Each dataset was of size $N=200$, with inputs $\{\mathbf{x}_n\}_{n=1}^N$ drawn uniformly from a D-dimensional unit hypercube and function values drawn from a Gaussian prior, $\mathbf{f}\sim\mathcal{N}(0,\Sigma)$, using covariance (15) with lengthscale $\ell=1$ and unit signal variance, $\sigma_f^2=1$. Noise with variance $\sigma_n^2=0.3^2$ was added to generate the observations.

Gaussian process classification: a well-explored application of Gaussian processes with a non-Gaussian noise model is binary classification:

$$L_c(\mathbf{f}) = \textstyle\prod_n \sigma\left(y_n f_n\right), \tag{17}$$

where $y_n\in\{-1,+1\}$ are the label data and $\sigma(a)$ is a sigmoidal function: $1/(1+e^{-a})$ for the logistic classifier; a cumulative Gaussian for the probit classifier.

We ran tests on the USPS classification problem as set up by Kuss and Rasmussen (2005). We used $\log\sigma_f=3.5$, $\log\ell=2.5$ and the logistic likelihood.

Log Gaussian Cox process: an inhomogeneous Poisson process with a non-parametric rate can be constructed by using a shifted draw from a Gaussian process as the log-intensity function. Approximate inference can be performed by discretizing the space into bins and assuming that the log-intensity is uniform in each bin (Møller et al., 1998). Each bin contributes a Poisson likelihood:

$$L_p(\mathbf{f}) = \prod_n \frac{{\lambda_n}^{y_n}\exp(-\lambda_n)}{y_n!}, \quad \lambda_n = e^{f_n+m}, \tag{18}$$

where the model explains the y_n counts in bin n as drawn from a Poisson distribution with mean λ_n. The offset to the log mean, m, is the mean log-intensity of the Poisson process plus the log of the bin size.

We perform inference for a Cox process model of the dates of mining disasters taken from a standard data set for testing point processes (Jarrett, 1979). The 191 events were placed into 811 bins of 50 days each. The Gaussian process parameters were fixed to $\sigma_f^2=1$ and $\ell=13516$ days (a third of the range of the dataset). The offset m in (18) was set to $m=\log(191/811)$, to match the empirical mean rate.

4.2 Results

A trace of the samples' log-likelihoods, Figure 4, shows that elliptical slice sampling and control variables sampling have different behavior. The methods make different types of moves and only control variables sampling contains rejections. Using long runs of either method to estimate expectations under the target distribution is valid. However, sticking in a state due to many rejections can give a poor estimator as can always making small moves. It can be difficult to judge overall sampling quality from trace plots alone.

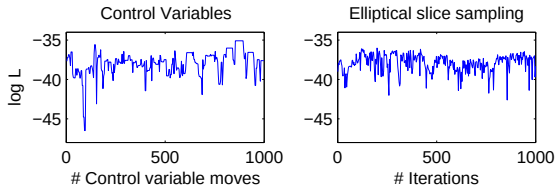

Figure 4: Traces of log-likelihoods for the 1-dimensional GP regression experiment. Both lines are made with 333 points plotted after each sweep through $M=3$ control variables and after every 3 iterations of elliptical slice sampling.

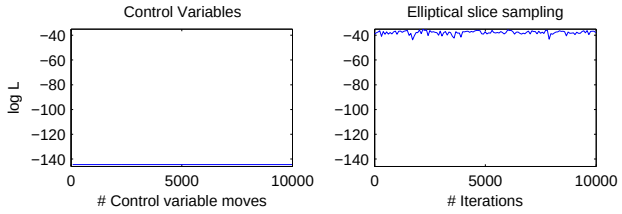

Figure 5: As in Figure 4 but for 10-dimensional regression and plotting every $M=78$ iterations. (Control variables didn't move on this run.)

As a quantitative measure of quality we estimated the "effective number of samples" from log-likelihood traces using R-CODA (Cowles et al., 2006). Figure 6 shows these results along with computer time taken. The step size for Neal's Metropolis method was chosen using a grid search to maximize performance. Times are for the provided implementations under Matlab v7.8 on a single 64 bit, 3 GHz Intel Xeon CPU. Comparing runtimes is always problematic, due to implementation-specific details. Our numbers of effective samples are primarily plotted for the same number of $\mathcal{O}(N^2)$ updates with the understanding that some correction based loosely on runtime should be applied.

The control variables approach was particularly recommended for Gaussian processes with low-dimensional input spaces. On our particular low-dimensional synthetic regression problems using control variables clearly outperforms all the other methods. On the model of mining disasters, control variable sampling has comparable performance to elliptical slice sampling with about 50% less run time. On higher-dimensional problems more control variables are required; then other methods cost less. Control variables failed to sample in high-dimensions (Figure 5). On the USPS classification problem control variables ran exceedingly slowly and we were unable to obtain any meaningful results.

Elliptical slice sampling obtained more effective samples than Neal's M–H method *with the best possible step size*, although at the cost of increased run time. On the problems involving real data, elliptical slice sampling

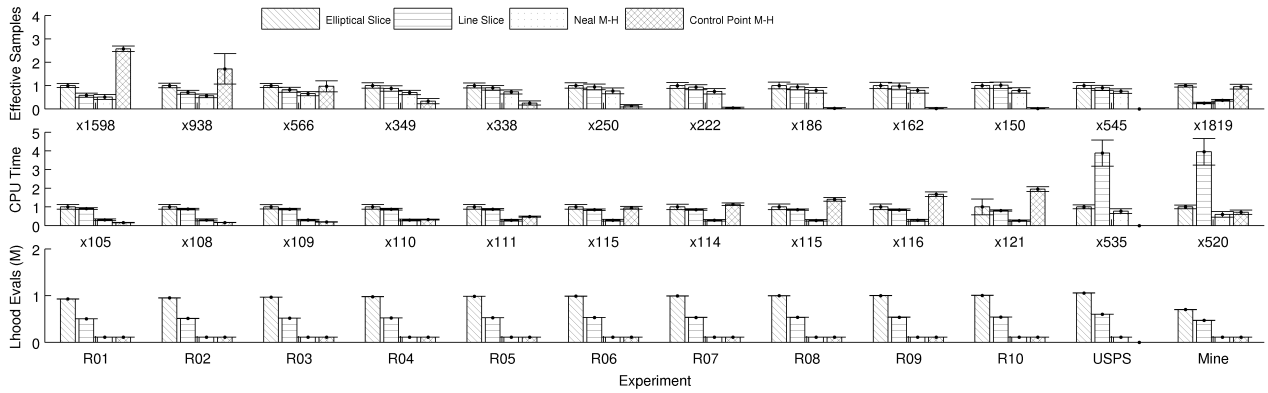

Figure 6: Number of effective samples from 10^5 iterations after 10^4 burn in, with time and likelihood evaluations required. The means and standard deviations for 100 runs are shown (divide the "error bars" by 10 to get standard errors on the mean, which are small). Each iteration involves one $\mathcal{O}(N^2)$ operation (e.g. one $\boldsymbol{\nu}$ draw or updating one control variable). Each group of bars in the top two rows has been rescaled for readability: the numbers beneath each group show the number of effective samples or CPU time in seconds for elliptical slice sampling, which always has bars of height 1.

was better overall whereas M–H has more effective samples per unit time (in our implementation) on the synthetic problems. The performance differences aren't huge; either method would work well enough.

Elliptical slice sampling takes less time than slice sampling along a straight line (line sampling involves additional prior evaluations) and usually performs better.

5 Discussion

The slice samplers use many more likelihood evaluations than the other methods. This is partly by choice: our code can take a step-size parameter to reduce the number of likelihood evaluations (Section 2.4). On these problems the time for likelihood computations isn't completely negligible: speedups of around $\times 2$ may be possible by tuning elliptical slice sampling. Our default position is that ease-of-use and human time is important and that the advantage of having no free parameters should often be taken in exchange for a factor of two in runtime.

We fixed the parameters of Σ and L in our experiments to simplify the comparison. Fixing the model potentially favors the methods that have adjustable parameters. In problems were Σ and L change dramatically, a single step-size or optimized set of control variables could work very poorly.

Elliptical slice sampling is a simple generic algorithm with no tweak parameters. It performs similarly to the best possible performance of a related M–H scheme, and could be applied to a wide variety of applications in both low and high dimensions.

Acknowledgements

Thanks to Michalis Titsias for code, Sinead Williamson and Katherine Heller for a helpful discussion, and to Radford Neal, Sam Roweis, Christophe Andrieu and the reviewers for useful suggestions. RPA is funded by the Canadian Institute for Advanced Research.

References

D. K. Agarwal and A. E. Gelfand. Slice sampling for simulation based fitting of spatial data models. *Statistics and Computing*, 15(1):61–69, 2005.

M. K. Cowles, N. Best, K. Vines, and M. Plummer. R-CODA 0.10-5, 2006. `http://www-fis.iarc.fr/coda/`.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, September 1987.

R. G. Jarrett. A note on the intervals between coal-mining disasters. *Biometrika*, 66(1):191–193, 1979.

M. Kuss and C. E. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

J. Møller, A. R. Syversveen, and R. P. Waagepetersen. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

R. M. Neal. Regression and classification using Gaussian process priors. In J. M. Bernardo et al., editors, *Bayesian Statistics 6*, pages 475–501. OU Press, 1999.

R. M. Neal. Slice sampling. *Annals of Statistics*, 31(3): 705–767, 2003.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for machine learning*. MIT Press, 2006.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2):319–392, 2009.

J. Skilling and D. J. C. MacKay. Slice sampling — a binary implementation. *Annals of Statistics*, 31(3), 2003.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

M. Titsias, N. D. Lawrence, and M. Rattray. Efficient sampling for Gaussian process inference using control variables. In *Advances in Neural Information Processing Systems 21*, pages 1681–1688, 2009.