Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE)

Maurizio Filippone
School of Computing Science, University of Glasgow, UK

MAURIZIO.FILIPPONE@GLASGOW.AC.UK

Raphael Engler
School of Computing Science, University of Glasgow, UK

RAPHAEL.ENGLER@WEB.DE

Abstract

In applications of Gaussian processes where quantification of uncertainty is of primary interest, it is necessary to accurately characterize the posterior distribution over covariance parameters. This paper proposes an adaptation of the Stochastic Gradient Langevin Dynamics algorithm to draw samples from the posterior distribution over covariance parameters with negligible bias and without the need to compute the marginal likelihood. In Gaussian process regression, this has the enormous advantage that stochastic gradients can be computed by solving linear systems only. A novel unbiased linear systems solver based on parallelizable covariance matrix-vector products is developed to accelerate the unbiased estimation of gradients. The results demonstrate the possibility to enable scalable and exact (in a Monte Carlo sense) quantification of uncertainty in Gaussian processes without imposing any special structure on the covariance or reducing the number of input vectors.

1. Introduction

Probabilistic kernel machines based on Gaussian Processes (GPs) (Rasmussen & Williams, 2006) are popular in a number of applied domains as they offer the possibility to flexibly model complex data and, depending on the choice of covariance function, to gain some understanding on the underlying behavior of the system under study. When quantification of uncertainty is of primary interest, it is necessary to accurately characterize the posterior distribution over covariance parameters. This has been argued in a number of papers where this is done by means of Markov chain Monte Carlo (MCMC) methods (Williams & Rasmussen, 1995; Williams & Barber, 1998; Neal, 1999; Murray & Adams, 2010; Taylor & Diggle,

2012; Filippone et al., 2013; Filippone & Girolami, 2014).

The limitation of MCMC approaches to draw samples from the posterior distribution over covariance parameters is that they need to compute the marginal likelihood at every iteration. In GP regression, a standard way to compute the marginal likelihood involves storing and factorizing an $n \times n$ matrix, leading to $O(n^3)$ time and $O(n^2)$ space complexities, where n is the size of the data set. For large data sets this becomes unfeasible, so a large number of contributions can be found in the literature on how to make these calculations tractable. For example, when the GP covariance matrix has some particular properties, e.g., it has sparse inverse (Rue et al., 2009; Simpson et al., 2013; Lyne et al., 2015), it is computed on regularly spaced inputs (Saatçi, 2011), or it is computed on univariate inputs (Gilboa et al., 2015), it is possible to considerably reduce the complexity in computing the marginal likelihood. When these properties do not hold, which is common in several Machine Learning applications, approximations are usually employed. Some examples involve the use subsets of the data (Candela & Rasmussen, 2005), the determination of a small number of surrogate input vectors that represent the full set of inputs (Titsias, 2009; Hensman et al., 2013), and the application of GPs to subsets of the data obtained by partitioning the input space (Gramacy et al., 2004), to name a few. Unfortunately, it is difficult to assess to what extent approximations affect the quantification of uncertainty in predictions, although some interesting results in this direction are reported in (Banerjee et al., 2013).

The focus of this paper are applications of GP regression where the structure of the covariance matrix is not necessarily special and quantification of uncertainty is of primary interest, so that approximations should be avoided. This paper proposes an adaptation of the Stochastic Gradient Langevin Dynamics (SGLD) algorithm (Welling & Teh, 2011) to draw samples from the posterior distribution over GP covariance parameters. SGLD does not require the computation of the marginal likelihood and yields sam-

ples from the posterior distribution of interest with negligible bias. This has the enormous advantage that stochastic gradients can be computed by solving linear systems only (Gibbs, 1997; Gibbs & MacKay, 1997; Stein et al., 2013). Linear systems can be solved by means of iterative methods, such as the Conjugate Gradient (CG) algorithm, that are based on parallelizable covariance matrix-vector products (Higham, 2008; Skilling, 1993; Seeger, 2000). Similar ideas were previously put forward to optimize GP covariance parameters (Chen et al., 2011; Anitescu et al., 2012; Stein et al., 2013). Despite the $O(n^2)$ in time and $O(n)$ in space complexities of these methods compare well with the $O(n^3)$ in time and $O(n^2)$ in space complexities of traditional MCMC-based inference, solving dense linear systems at each iteration makes the whole inference framework too slow to be of practical use. We compare a number of standard ways to speed up the solution of dense linear systems, such as fast covariance matrix-vector products (Gray & Moore, 2000; Moore, 2000) and preconditioning (Srinivasan et al., 2014), and in line with what reported in (Murray, 2009), we observe that they yield little gain in computational speed compared to the standard CG algorithm. In order to enable practical inference for GPs applied to large data sets, we therefore develop an Unbiased LInear Systems SolvEr (ULISSE) that essentially allows the CG algorithm to stop early while retaining unbiasedness of the solution.

We highlight here that (i) in (Welling & Teh, 2011), an unbiased estimate of the gradient is computed by considering small batches of data. Recent alternative contributions on scaling Bayesian inference by analyzing small batches of data can be found in (Banterle et al., 2014; Maclaurin & Adams, 2014). GPs do not lend themselves to this treatment, due to the covariance structure making all data dependent on one another. (ii) ULISSE is complementary to recent approaches in the area of probabilistic numerics that aim at infering, rather than computing, solutions to linear systems (Hennig, 2014). (iii) The proposed inference method is based on "noisy" gradients and is complementary to recent inference approaches based on noisy likelihoods (Lyne et al., 2015; Filippone, 2014). In GP regression, iterative methods akin to the CG algorithm (Higham, 2008) can be employed to obtain an unbiased estimate of the log-determinant of the covariance matrix, but this remains an extremely onerous calculation needed to get an unbiased estimate of the log-marginal likelihood. A further and perhaps more challenging issue is transforming the unbiased estimate of the log-marginal likelihood in an unbiased estimate of the marginal likelihood (Kennedy & Kuti, 1985; Liu, 2000; Lyne et al., 2015).

This paper demonstrates that employing ULISSE within SGLD makes it possible to accurately carry out inference of covariance parameters in GPs and effectively scale these computations to large data sets. We report results on a data set with about 23 thousand input vectors where we can draw ten thousand samples per day from the posterior distribution over covariance parameters on a desktop machine with standard hardware[1]. To the best of our knowledge, this paper reports the first real attempt to enable full quantification of uncertainty of covariance parameters of GPs without reducing the number of input vectors and without imposing sparsity on the GP covariance or its inverse.

The paper is organized as follows. Section 2 briefly reviews GPs and motivates the adoption of SGLD to infer GP covariance parameters. Section 3 describes and evaluates the CG algorithm to solve linear systems and some variants based on fast covariance matrix-vector product and preconditioning. Section 4 presents ULISSE and its use to obtain an unbiased estimate of the gradient of the log-marginal likelihood in GPs. Section 5 demonstrates the ability of the proposed methodology to accurately infer covariance parameters in GPs and its scalability properties to a large data set where the marginal likelihood cannot be computed exactly. Finally, Section 6 draws the conclusions.

2. Inference of covariance parameters in GPs

In GP regression, a set of continuous labels $\mathbf{y} = \{y_1, \ldots, y_n\}$ is associated with a set of input vectors $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Throughout the paper, we will employ zero mean GPs with the following covariance function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma \exp\left(\tau \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) + \lambda \delta_{ij} \qquad (1)$$

with $\delta_{ij} = 1$ if $i = j$ and zero otherwise. The parameter τ determines the rate of decay of the covariance function, whereas σ represents the marginal variance of each Gaussian random variable comprising the GP. The parameter λ is the variance of the (Gaussian) noise on the labels. Let K be the covariance matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and denote by $\boldsymbol{\theta}$ the vector comprising all parameters of the covariance matrix K, namely $\boldsymbol{\theta} = (\sigma, \tau, \lambda)$.

In a Bayesian sense, we would like to carry any uncertainty in parameters estimates forward to predictions over the label y_* for a new input vector $\mathbf{x}_*$. In particular, this requires solving the following integral:

$$p(y_*|\mathbf{y}, X, \mathbf{x}_*) = \int p(y_*|\mathbf{y}, \boldsymbol{\theta}, X, \mathbf{x}_*)p(\boldsymbol{\theta}|\mathbf{y}, X)d\boldsymbol{\theta}. \quad (2)$$

Such an expectation, like any other expectation under the posterior over $\boldsymbol{\theta}$, is analytically intractable, so it is necessary to resort to some approximations. A standard way to tackle this intractability is to draw samples from $p(\boldsymbol{\theta}|\mathbf{y}, X)$ using MCMC methods, and approximate the expectation

---

[1]Code to reproduce all the results can be found here:
www.dcs.gla.ac.uk/~maurizio/pages/code.html

with the Monte Carlo estimate

$$p(y_*|\mathbf{y}, X, \mathbf{x}_*) \simeq \frac{1}{N} \sum_{i=1}^{N} p(y_*|\boldsymbol{\theta}^{(i)}, X, \mathbf{x}_*), \qquad (3)$$

where $\boldsymbol{\theta}^{(i)}$ denotes the ith of a set of samples from $p(\boldsymbol{\theta}|\mathbf{y}, X)$. Drawing samples from the posterior distribution can be done using several MCMC algorithms that essentially are based on a proposal mechanism and on an accept/reject step that requires the evaluation of the log-marginal likelihood:

$$\log[p(\mathbf{y}|\boldsymbol{\theta}, X)] = -\frac{1}{2} \log(|K|) - \frac{1}{2} \mathbf{y}^{\mathrm{T}} K^{-1} \mathbf{y} + \mathrm{const.} \tag{4}$$

A standard way to proceed, is to factorize the covariance matrix $K = LL^{\mathrm{T}}$ using the Cholesky algorithm (Golub & Van Loan, 1996). The factorization costs $O(n^3)$ operations and requires the storage of $O(n^2)$ entries of the covariance matrix, but after that computing the log-determinant and the inverse of K multiplied by $\mathbf{y}$ can be done using $O(n^2)$ operations.

The computational complexities above pose a constraint on the scalability of GPs to large data sets. Iterative methods based on covariance matrix-vector products (CMVPs) have been proposed to obtain an unbiased estimate of the log-marginal likelihood. Even though these methods scale with $O(n^2)$ in time and $O(n)$ in space, they are of little practical use in the task of sampling from $p(\boldsymbol{\theta}|\mathbf{y}, X)$, as the number of iterations needed to estimate the log-determinant term can be prohibitively large (see, e.g., (Chen et al., 2011)). We now illustrate our proposal to obtain samples from $p(\boldsymbol{\theta}|\mathbf{y}, X)$ with negligible bias and without having to estimate log-determinants and marginal likelihoods.

2.1. Stochastic Gradient Langevin Dynamics (SGLD)

We briefly describe how to adapt SGLD (Welling & Teh, 2011) to obtain samples from $p(\boldsymbol{\theta}|\mathbf{y}, X)$ in GPs. The idea behind SGLD is to modify the standard stochastic gradient optimization algorithm (Robbins & Monro, 1951) by injecting Gaussian noise in a way that ensures transition into a Langevin dynamics phase yielding samples from the posterior distribution of interest. In particular, the proposal of a new set of parameters is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\varepsilon_t}{2} M \{\tilde{\mathbf{g}} + \nabla_{\boldsymbol{\theta}} \log[p(\boldsymbol{\theta})]\} + \boldsymbol{\eta}_t \tag{5}$$

with $\boldsymbol{\eta}_t \sim \mathcal{N}(\boldsymbol{\eta}_t|0, \varepsilon_t M)$ and $\tilde{\mathbf{g}}$ an unbiased estimate of the gradient of $\log[p(\mathbf{y}|\boldsymbol{\theta}, X)]$. We have also introduced a preconditioning matrix M that can be chosen to improve convergence of SGLD. The update equation, except for $\boldsymbol{\eta}_t$, is the standard update used in stochastic gradient optimiza-

tion. The parameters ε_t are chosen to satisfy

$$\sum_{t=1}^{\infty} \varepsilon_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \varepsilon_t^2 < \infty \tag{6}$$

as these conditions, along with some other technical assumptions, guarantee convergence to a local maximum. The injected noise $\boldsymbol{\eta}_t$ is Gaussian with covariance $\varepsilon_t M$ ensuring that the algorithm transitions into a discretized version of a Langevin dynamics with target distribution given by the posterior over $\boldsymbol{\theta}$. In principle, it would be necessary to accept or reject the proposals, which would require evaluating the marginal likelihood. The key result in (Welling & Teh, 2011) is that when SGLD reaches the Langevin dynamics phase, the step-size ε_t is small enough to make the acceptance rate close to one. Therefore, in this phase it is possible to accept all proposals, avoiding having to evaluate the marginal likelihood, at the cost of introducing a negligible amount of bias.

Following (Welling & Teh, 2011), we can estimate when the algorithm reaches the Langevin dynamics phase by monitoring the ratio between the variance of the stochastic gradients and the variance of the injected noise. Defining V to be the sampling covariance of the stochastic gradients and $\lambda_{\max}(A)$ to be the largest eigenvalue of a matrix A, we can write such a ratio as

$$\frac{\varepsilon_t}{4} \lambda_{\max} \left(M^{\frac{1}{2}} V M^{\frac{1}{2}} \right) \tag{7}$$

When this ratio is small enough the algorithm is in its Langevin dynamics phase and produces samples from the posterior distribution over $\boldsymbol{\theta}$. Further theoretical analyses on the convergence properties of SGLD can be found in (Teh et al., 2014; Vollmer et al., 2015).

The motivation for employing SGLD for inference of GP covariance parameters comes from inspecting the gradient of the log-marginal likelihood that has components

$$g_i = -\frac{1}{2} \mathrm{Tr} \left(K^{-1} \frac{\partial K}{\partial \theta_i} \right) + \mathbf{y}^{\mathrm{T}} K^{-1} \frac{\partial K}{\partial \theta_i} K^{-1} \mathbf{y} \tag{8}$$

Computing the g_i's requires again $O(n^3)$ operations due to the trace term and the linear system $K^{-1}\mathbf{y}$. However, we can introduce $N_{\mathbf{r}}$ vectors $\mathbf{r}^{(i)}$ with components drawn from $\{-1, 1\}$ with probability $1/2$ and unbiasedly estimate the trace term (Gibbs, 1997), obtaining:

$$\tilde{g}_i = -\frac{1}{2N_{\mathbf{r}}} \sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)\mathrm{T}} K^{-1} \frac{\partial K}{\partial \theta_i} \mathbf{r}^{(i)} + \mathbf{y}^{\mathrm{T}} K^{-1} \frac{\partial K}{\partial \theta_i} K^{-1} \mathbf{y} \tag{9}$$

Given that $\mathrm{E}(\mathbf{r}^{(i)}\mathbf{r}^{(i)\mathrm{T}}) = I$, we can readily verify that $\mathrm{E}[\mathbf{r}^{(i)\mathrm{T}} K^{-1} \frac{\partial K}{\partial \theta_i} \mathbf{r}^{(i)}] = \mathrm{Tr} \left[K^{-1} \frac{\partial K}{\partial \theta_i} \mathrm{E}(\mathbf{r}^{(i)}\mathbf{r}^{(i)\mathrm{T}}) \right]$, which yields the trace term in eq. 8. Hence, in order to compute an

Algorithm 1 The Conjugate Gradient algorithm

Input: data X, vector $\mathbf{b}$, convergence threshold ϵ, initial vector $\mathbf{s}_0$, maximum number of iterations T

$\mathbf{e}_0 = \mathbf{b} - K\mathbf{s}_0; \quad \mathbf{d}_0 = \mathbf{e}_0;$

for $i = 0$ **to** T **do**

$\quad \alpha_i = \dfrac{\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_i}{\mathbf{d}_i^{\mathrm{T}}K\mathbf{d}_i};$

$\quad \mathbf{s}_{i+1} = \mathbf{s}_i + \alpha_i \mathbf{d}_i;$

$\quad \mathbf{e}_{i+1} = \mathbf{e}_i - \alpha_i K\mathbf{d}_i;$

$\quad$ **if** $\|\mathbf{e}_{i+1}\| < \epsilon$ **then**

$\quad\quad$ return $\mathbf{s} = \mathbf{s}_{i+1};$

$\quad$ **end if**

$\quad \beta_i = \dfrac{\mathbf{e}_{i+1}^{\mathrm{T}}\mathbf{e}_{i+1}}{\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_i};$

$\quad \mathbf{d}_{i+1} = \mathbf{e}_{i+1} + \beta_i \mathbf{d}_i;$

end for

unbiased version of the gradient of the log-marginal likelihood we need to solve one linear system for $\mathbf{y}$ and one for each of the $N_{\mathbf{r}}$ vectors $\mathbf{r}^{(i)}$ used to estimate the trace term. This consideration forms the basis of the proposed methodology. Computing an unbiased version of the gradient involves solving linear systems only, which is much easier and cheaper than estimating log-determinants.

3. Solving linear systems without storing K

We have discussed that SGLD to infer covariance parameters in GPs requires solving linear systems. Here we briefly review the Conjugate Gradient (CG) algorithm that is a popular method to iteratively solve linear systems based on Covariance Matrix Vector Product (CMVP) operations. CMVPs can be carried out without having to store K, so their time and space complexities are in $O(n^2)$ and $O(n)$, respectively. We also discuss and evaluate a few variants to speed up computations/convergence, such as preconditioning and fast CMVPs. Throughout this section we will evaluate the effectiveness of these alternatives on a GP regression task applied to the Concrete data set from the UCI repository (Asuncion & Newman, 2007). This data set contains data about the compressive strength of $n = 1030$ samples of concrete described by $d = 8$ features.

3.1. The Conjugate Gradient (CG) algorithm

Given a linear system of the form $K\mathbf{s} = \mathbf{b}$ with K and $\mathbf{b}$ given, the CG algorithm (Golub & Van Loan, 1996) yields the solution $\mathbf{s}$ without having to invert or factorize the matrix K. The idea is to calculate the solution $\mathbf{s}$ as the minimizer of

$$\phi(\mathbf{s}) = \frac{1}{2}\mathbf{s}^{\mathrm{T}}K\mathbf{s} - \mathbf{s}^{\mathrm{T}}\mathbf{b} \qquad (10)$$

which can be obtained by employing gradient-based optimization. The CG algorithm is initialized from an initial
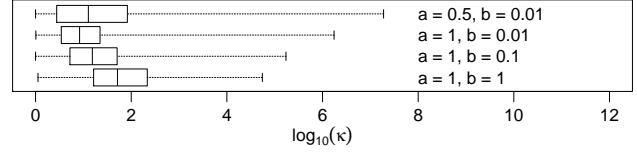

Figure 1. Distribution of the condition number κ of the covariance matrix for different choices of shape and rate parameters of a Gamma prior on each covariance parameter θ.

guess $\mathbf{s}_0$. After that, the iterations refine the solution $\mathbf{s}$ by updates in directions $\mathbf{d}_i$. The CG algorithm, in comparison with the standard gradient descent, is characterized by the fact that K-orthogonality (or conjugacy with respect to K) of the search directions is imposed, namely $\mathbf{d}_i^{\mathrm{T}}K\mathbf{d}_j = 0$ when $i \neq j$. This condition yields a sequence of residuals $\mathbf{e}_i = \mathbf{b} - K\mathbf{s}_i$ that are mutually orthogonal, and guarantees convergence in at most n iterations. Remarkably, the CG algorithm can be implemented in a way that requires a single CMVP ($K\mathbf{d}_i$) at each iteration (see Algorithm 1).

The trade-off between accuracy and speed is governed by the threshold ϵ, which in this paper is set to $\epsilon = 10^{-8}$. Theoretically, the CG algorithm is guaranteed to converge in at most n iterations, but in practice, due to the representation in finite numerical precision, orthogonality of the directions can be lost, especially in badly conditioned systems, and the CG algorithm can take more than n iterations to converge. The condition number of a matrix is defined as the ratio between its largest and smallest eigenvalues:

$$\kappa = \frac{\lambda_{\max}(K)}{\lambda_{\min}(K)}$$

Fig. 1 shows the distribution of the condition number when each covariance parameter θ_i is sampled form a Gamma distribution with shape and rate parameters a and b. The distributions are reasonably vague and give a rough idea of the typical condition numbers encountered during the inference of GP covariance parameters for the Concrete data set. We can expect slower convergence speed when the condition number is large due to numerical instabilities; we are interested in quantifying to what extent this applies to GPs and what is the impact of cheap CMVPs and preconditioning on convergence speed. In the remainder of this section, we will consider the problem of solving the linear system $K\mathbf{s} = \mathbf{y}$ that is needed in the calculation of part of the gradient in eq. 9. The results pertaining to the solution of the linear systems $K\mathbf{s} = \mathbf{r}^{(i)}$ are quite similar, so for the sake of brevity we will omit them.

3.2. Fast CMVPs

We consider here the use of two fast CMVPs based on efficient representation of input data that we will call "kdtree" (Gray & Moore, 2000) and "anchors" (Moore,

2000)[2]. These methods yield fast CMVPs at the price of a lower accuracy.

In the top row of Fig. 2 we show the number of iterations required by the CG algorithm to reach convergence versus the condition number and the error in the solution versus the condition number. The error is defined as the norm of the difference between the solution obtained by the CG algorithm and the one obtained by factorizing K using the Cholesky algorithm and carrying out forward and back substitutions with $\mathbf{y}$. We compare a baseline CG algorithm with CMVPs performed in double precision with CG algorithms implemented with (i) single precision ("float") CMVPs, (ii) "kdtree" CMVPs and (iii) "anchors" CMVPs. The convergence threshold of the CG algorithm was set to 10^{-8}, so in order to be able to satisfy this criterion when employing "kdtree" and "anchors" CMVPs, we selected the relative and absolute tolerance parameters to be 10^{-10}.

The results indicate that double precision calculations lead to the lowest number of iterations compared to the other methods, especially when κ is large. Double precision calculations also offer the lowest error. Single precision calculations lead to a very poor error compared to the other methods. The CG algorithm with "kdtree" CMVPs seems to take longer to converge than the one with "anchor" CMVPs, but it achieves a lower error.

Drawing definitive conclusions on whether fast CMVPs yield any gain in computing time is far from trivial, as this very much depends on implementation details and hardware where the code is run. What we can say, however, is that gaining orders of magnitude speed-ups would require reducing the accuracy of fast CMVPs, but this would require loosening up the convergence criterion in order for the CG algorithm to converge. As a result, we would be able to obtain solutions of linear systems faster but at the cost of a reduced accuracy in the solution, which in turn would bias the estimation of gradients.

3.3. Preconditioned CG

The Preconditioned CG (PCG) is a variant of the CG algorithm that aims at mitigating the issues associated with the rate of convergence of the CG algorithm when the condition number κ is large. A (right) preconditioning matrix J operates on the linear system yielding

$$KJ^{-1}(J\mathbf{s}) = \mathbf{b}$$

The success of PCG is based on the possibility to construct J so that KJ^{-1} is well conditioned. This can be achieved when J^{-1} well approximates K^{-1}, and a complication immediately arises on how to do so for general kernel matrices

---

[2]code implementing these methods can be found here: www.cs.ubc.ca/~awll/nbody_methods.html
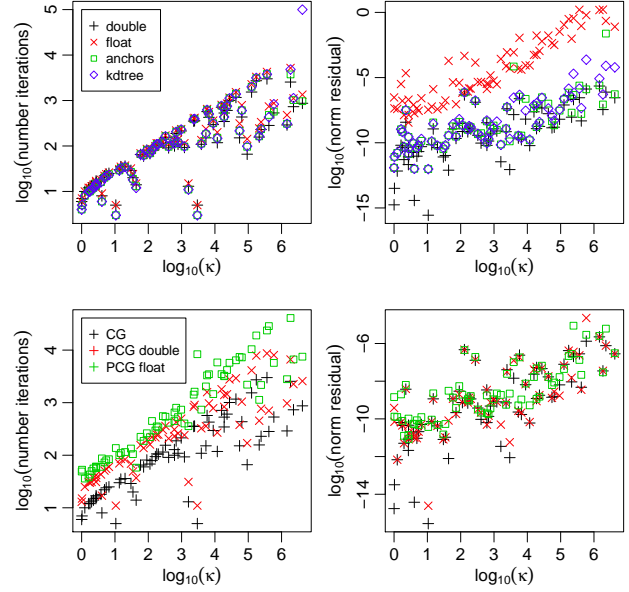

Figure 2. Top row: Comparison of fast CMVPs on number of iterations and error versus condition number. Bottom row: Comparison of the CG algorithm and two PCG algorithms using double and single precision CMVPs to solve inner linear systems.

without carrying out expensive operations (in $O(n^3)$).

In (Srinivasan et al., 2014) it was proposed to define $J = K + \delta I$ with $\delta > 0$. Compared to the standard CG algorithm, the PCG algorithm introduces the solution of an "inner" linear system of the form $J^{-1}\mathbf{z}$ at each iteration, that can be solved again using the CG algorithm. A large value of δ makes $K + \delta I$ well conditioned and makes convergence speed of the inner CG algorithm faster, whereas it makes J^{-1} and K^{-1} considerably different leading to the necessity to run the outer CG algorithm for several iterations. For small values of δ the situation is reversed, so δ needs to be tuned to find an optimal compromise.

In the bottom row of Fig. 2, we compare the standard CG algorithm with two versions of the PCG algorithm on number of iterations and accuracy of the solution. In the first version of the PCG algorithm we used double precision calculations when solving the inner linear systems, whereas in the second version we used single precision calculations. In both versions of the PCG algorithm we set δ to yield the lowest number of iterations in order to show whether it is possible to reduce the number of computations.

The results show that the standard CG algorithm takes less iterations to converge than the PCG algorithm (counting both inner and outer iterations). Even in the case of single precision calculations in the inner CG algorithm, we did not experience any gain in computing time due to the increased number of iterations. For other data and in dif-

ferent experimental conditions there might be a computational advantage in using a preconditioner, as shown in (Srinivasan et al., 2014), but the gain is generally modest.

4. Unbiased LInear System SolvEr (ULISSE)

From the analysis in the previous sections it is evident that none of the standard ways to speedup calculations and convergence of the CG algorithm offer substantial gains in computing time. As a result, employing iterative methods as an alternative to traditional factorization techniques seems beyond practicality as pointed out, e.g., in (Murray, 2009). One of the novel contributions of this paper is to accelerate the CG algorithm at the expenses of obtaining an (unbiased) estimate of the solution. The idea is to stop the CG algorithm before the convergence criterion is satisfied and apply some corrections to ensure unbiasedness of the solution. We note here that our proposal can be applied to any of the variants of the CG algorithm presented earlier and to dense as well as sparse linear systems.

We can rewrite the final solution of a linear system obtained by the CG algorithm as a sum of incremental updates

$$\mathbf{s} = \mathbf{s}_0 + \boldsymbol{\delta}_1 + \ldots + \boldsymbol{\delta}_T \tag{11}$$

assuming that it takes T iterations to satisfy the convergence threshold ϵ. We can define an "early stop" threshold $\alpha > \epsilon$ that will be reached after $l < T$ iterations, and rewrite the final solution by introducing a series of coefficients as follows

$$\begin{aligned} \mathbf{s} &= \mathbf{s}_0 + \sum_{i=1}^{l-1} \boldsymbol{\delta}_i + \frac{1}{w_0}\left(w_0\boldsymbol{\delta}_{l+0} + \frac{1}{w_1}\left(w_0 w_1 \boldsymbol{\delta}_{l+1} + \right. \right. \\ &\quad \left. \left. + \frac{1}{w_2}\left(w_0 w_1 w_2 \boldsymbol{\delta}_{l+2} + \ldots\right)\right)\right) \end{aligned} \tag{12}$$

We will focus on coefficients defined as $w_r = \exp(\beta r)$, but this choice is not restrictive. We can now obtain an unbiased estimate of the solution of the linear system by adding these instructions to the standard CG algorithm: set $\tilde{\mathbf{s}} = \mathbf{s}_0 + \sum_{i=1}^{l-1} \boldsymbol{\delta}_i$ and iterate for $j = 0, 1, \ldots$ the following two steps (i) draw $u_j \sim U[0,1]$ (ii) if $u_j < \frac{1}{w_j}$ then $\tilde{\mathbf{s}} = \tilde{\mathbf{s}} + \prod_{r=0}^{j} w_r \boldsymbol{\delta}_{l+j}$, else return $\tilde{\mathbf{s}}$ and stop the CG algorithm. The expectation of $\tilde{\mathbf{s}}$ is clearly $\mathbf{s}$ and the rate of decay β in the expression of w_r determines the average number of steps that are carried out after the convergence threshold α is reached.

For simplicity, we set the early stop threshold to $\alpha = q\sqrt{n}$ as q gives a rough indication of the average error that we are expecting in each element of the solution. In Fig. 3 we report number of iterations and average standard deviation across the elements of the solution. ULISSE with two different values of β and q is compared with the baseline CG
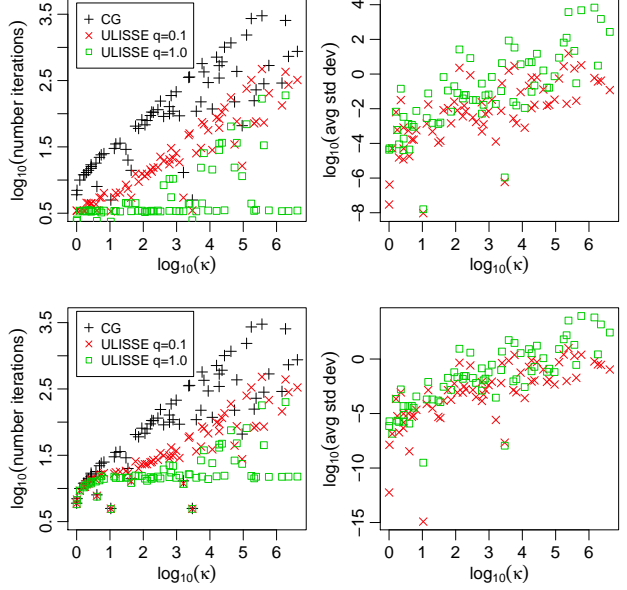

Figure 3. Comparison of the CG algorithm and ULISSE with early stop thresholds α calculated with $q = 0.1$ and $q = 1$ on number of iterations and standard deviation of the solution. The top row corresponds to $\beta = 1$ in the calculation of the weights w_r, whereas the bottom row corresponds to $\beta = 100$.

algorithm without early stop ("CG"). We stress again that the error is such that the solution is unbiased.

4.1. Impact on the calculation of stochastic gradients

We conclude this section by showing the impact of ULISSE in the calculation of stochastic gradients in GPs. Applying the proposed unbiased solver to the first term of $\tilde{g}_i$ in eq. 9 is straightforward and it requires solving $N_\mathbf{r}$ linear systems, one for each of the $\mathbf{r}^{(i)}$ vectors. For the quadratic term in $\mathbf{y}$, instead, we need to obtain two independent unbiased estimates of $K^{-1}\mathbf{y}$ in order for the expectation of the whole term to be unbiased. This can be implemented by running a single instance of the CG algorithm and keeping track of two solutions obtained by independent draws of the uniform variables u_j used to early stop the CG algorithm. We remark that the unbiased estimation of gradients involves now two sources of stochasticity: one due to the stochastic estimate of the trace term in eq. 8, and one due to the proposed way to unbiasedly solve all linear systems in eq. 9.

Fig. 4 reports the average, taken with respect to 100 repetition of the $\log_{10}$ of the relative square norm of the error:

$$\frac{\|\mathbf{g}(\boldsymbol{\theta}) - \tilde{\mathbf{g}}(\boldsymbol{\theta})\|^2}{\|\mathbf{g}(\boldsymbol{\theta})\|^2} \tag{13}$$

as a function of the condition number κ. We used one vector $\mathbf{r}^{(1)}$ to estimate the gradient in eq. 9. The figure shows
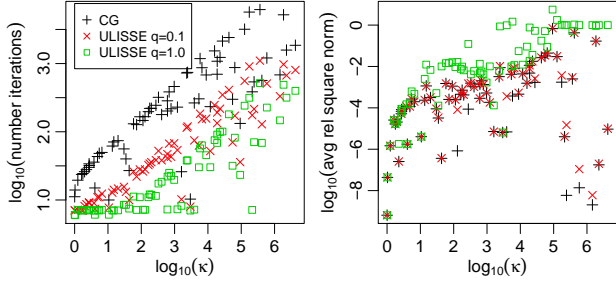
Figure 4. Comparison of the CG algorithm and ULISSE and early stop thresholds computed with $q = 0.1$ and $q = 1$ to estimate the gradient of the log-marginal likelihood in eq. 9. In ULISSE, the weights w_r are calculated with $\beta = 1$.

Figure 5. Concrete data ($n = 1030$) - Left panel: Comparison of MCMC (red) and SGLD with ULISSE (black) on running mean and plus/minus two standard deviations. The trace of one chain of SGLD is shown in gray. Right panel: Convergence analysis of SGLD with ULISSE with PSRF computed over ten chains.

that the estimate in eq. 9 ("CG" in the figure) is quite accurate, as the relative error is small in a wide range of values of κ. Also, at the expenses of a larger variance in the estimate of the gradient, ULISSE yields orders of magnitude improvements in the number of iterations.

5. Experimental validation

In this section, we infer covariance parameters of GP regression models using SGLD with ULISSE. We start by considering the Concrete data set where it is possible to compare our proposal with the Metropolis-Hastings (MH) algorithm. We then demonstrate the scalability of the proposed methodology by considering a data set with $n = 22,784$ and $d = 8$.

5.1. Comparison with MCMC

We ran the MH algorithm for fifty-thousand iterations to the GP regression model with covariance in eq. 1 applied to the Concrete data set. We allowed for an initial adaptive phase to reach an average acceptance rate between 0.2 and 0.4, and we discarded the first ten-thousand samples. Fig. 5 shows the running mean and the interval corresponding to plus/minus twice the running standard deviation of the posterior over the three parameters (solid red lines) computed over the remaining forty-thousand samples.

We compare the run from the MH algorithm with SGLD, where we made the following design choices. We employed ULISSE within the CG algorithm with double precision CMVPs. We set the early stop threshold α to $\sqrt{n}$ and the parameter β in the computation of the weights w_r to 1. Stochastic gradients were computed using $N_{\mathbf{r}} = 4$ vectors $\mathbf{r}^{(i)}$. We ran SGLD for forty-thousand iterations; the step-size was set to $\varepsilon_t = a(b + t)^{-\gamma}$, with $\gamma = 1$, and it was chosen to start from 10^{-1} and reduce to 10^{-4} on the last iteration. During the execution of SGLD we monitored the
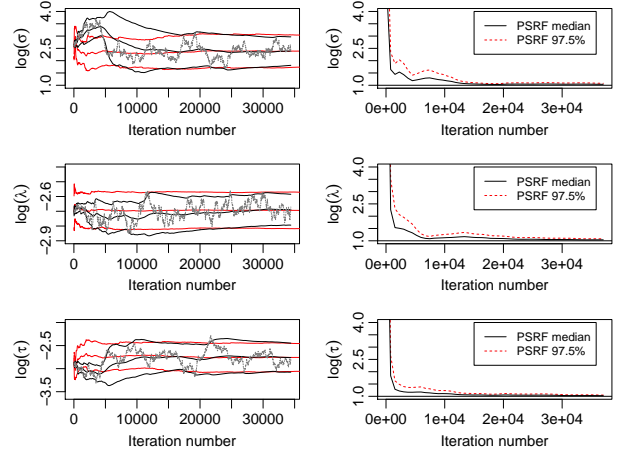
quantity $\frac{\varepsilon_t}{4} \lambda_{\max} \left(M^{\frac{1}{2}} V M^{\frac{1}{2}} \right)$ as discussed in Section 4, and we froze the value of ε_t when it was less than 0.002; the covariance of the gradients V was estimated on batches of one-hundred iterations. In order to speed up computations, we decided to redraw the vectors $\mathbf{r}^{(i)}$ every twenty iterations and to keep them fixed in between. The advantage of this is that the solutions of the linear systems $K\mathbf{r}^{(i)}$ can be used to initialize the same systems when proposing new $\boldsymbol{\theta}$'s thus speeding up convergence. Finally, we set the preconditioning matrix M in SGLD as the inverse of the negative Hessian of the log of the posterior density at its mode computed on a subset of five hundred input vectors, as this is cheap way to obtain a rough idea of the covariance structure of the posterior distribution for the full data set.

SGLD yields an effective sample size of about 0.1% and it draws one independent sample every 27 sec. In Fig. 5 we report the running statistics for the three parameters (solid black lines), and the trace-plot of one run of SGLD (solid gray lines), where we discarded all iterations prior to the freezing of the step-size ε_t. The figure shows a striking match between the results obtained by a standard MCMC approach and SGLD with ULISSE. This demonstrates that our proposal is a valid alternative to other MCMC approaches to reliably quantify uncertainty in GPs.

In order to check convergence speed of SGLD, we ran ten parallel chains and computed the Potential Scale Reduction Factor (PSRF) (Gelman & Rubin, 1992). The chains were initialized by drawing from a Gaussian with mean on the MAP solution over a subset of five hundred input vectors and covariance M, so as to ensure enough dispersion to reliably report the PSRF. Fig. 5 shows the median and the
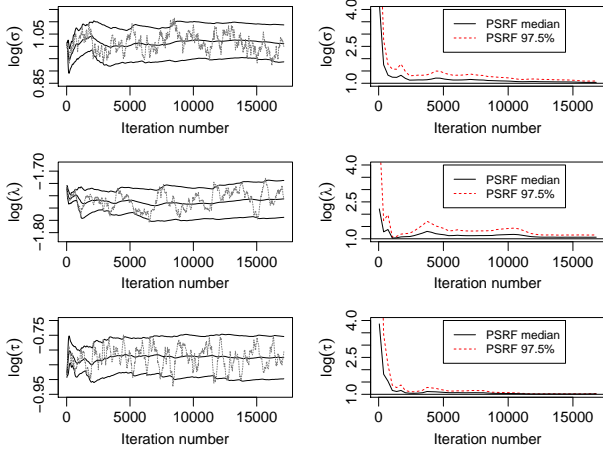
Figure 6. Census data ($n = 22,784$) - Left panel: Running mean and plus/minus two standard deviations (black) and trace of one chain of SGLD (gray). Right panel: Convergence analysis of SGLD with ULISSE reporting the PSRF computed over five chains.

97.5th percentile of the PSRF across the ten chains. The analysis of these plots reveals that SGLD achieves convergence after few thousand iterations.

5.2. Demonstration on a larger dataset

We now present the application of SGLD with ULISSE to a data set where it is not possible to run any MCMC algorithm with exact computation of the marginal likelihood on a conventional desktop machine. This data set contains data collected as part of the 1990 US census. In this study, we used the 8L data set[3] where the regression task associates the median house price in a given region with demographic composition and housing market features ($n = 22,784$ and $d = 8$). We kept the same experimental conditions as in the case of the Concrete data, except that ε_t was chosen to decrease from $5 \cdot 10^{-2}$ to $5 \cdot 10^{-6}$ to cope with the larger gradients obtained for this data set, and the preconditioner M was estimated based on the MAP on one-thousand data points. The running statistics for the three parameters for one chain are reported in Fig. 6, along with the PSRF computed across five chains, which shows that convergence was reached after few thousand iterations.

SGLD with ULISSE was run on a desktop machine with an eight core (i7-2600 CPU at 3.40GHz) processor, and an NVIDIA GeForce GTX 590 graphics card (released in 2011). The two GPUs in the graphics card are used to carry out CMVPs. With this arrangement, we were able to draw roughly ten thousand samples per day from the posterior distribution over covariance parameters. SGLD yields an

[3]`www.cs.toronto.edu/~delve/data/`

effective sample size of roughly 0.1%, and it can draw one independent sample every 2.4 hours.

6. Conclusions

This paper presented a novel way to accurately infer covariance parameters in GPs. The novelty stems from the combination of stochastic gradient-based inference and a fast unbiased solver of linear systems. The results demonstrate that it is possible to carry out inference of GP covariance parameters over a data set comprising about 23 thousand input vectors in a day on a desktop machine with standard hardware. The proposed methodology can exploit parallelism in computing covariance matrix-vector products, so there is an opportunity to scale "exact" inference (in a Monte Carlo sense) to even larger data sets. We are not aware of any method that is capable of carrying out full quantification of uncertainty of GP covariance parameters on such large data sets without imposing special structures on the covariance or reducing the number of input vectors. These results are important not only in Machine Learning, but also in areas where quantification of uncertainty is of primary interest and GPs are routinely employed, such as calibration of computer models (Kennedy & O'Hagan, 2001) and optimization (Jones et al., 1998).

The results reported in this paper, although promising, indicate some directions for improvements. SGLD requires the tuning of a preconditioning matrix M. Choosing M to be similar to the covariance of the posterior speeds up convergence of SGLD when it reaches the Langevin dynamics phase. However, M also affects the scaling of the gradient in the proposal. During the first phase of SGLD this might not be optimal, and ideally, gradients should be scaled in a way similar to AdaGrad (Duchi et al., 2011). In (Welling & Teh, 2011), it was possible to establish a connection between the covariance of the gradients, the Fisher Information, and M due to the fact that stochastic gradients are computed on subsets of the data. We were unable to do so for GPs due to the different way stochasticity is introduced in the computation of the gradients. Despite this complication, we demonstrated that it is still possible to obtain convergence to the posterior distribution over covariance parameters in a reasonable number of iterations, which is of ultimate importance in any inference task.

We are currently investigating the application of SGLD to automatic relevance determination covariances and the possibility to extend our proposal to scale inference for other GP models, e.g., GP classification and GPs for spatio-temporal data. Other interesting aspects to explore would be the introduction of mixed precision calculations within the CG algorithm to improve convergence and computation speed as presented, e.g., in (Jang et al., 2011; Cevahir et al., 2009; Baboulin et al., 2009).

Acknowledgments

References

Anitescu, M., Chen, J., and Wang, L. A Matrix-free Approach for Solving the Parametric Gaussian Process Maximum Likelihood Problem. *SIAM Journal on Scientific Computing*, 34(1):A240–A262, 2012.

Asuncion, A. and Newman, D. J. UCI machine learning repository, 2007.

Baboulin, M., Buttari, A., Dongarra, J., Kurzak, J., Langou, J., Langou, J., Luszczek, P., and Tomov, S. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12): 2526–2533, 2009.

Banerjee, A., Dunson, D. B., and Tokdar, S. T. Efficient Gaussian process regression for large datasets. *Biometrika*, 100(1):75–89, 2013.

Banterle, M., Grazian, C., and Robert, C. P. Accelerating Metropolis-Hastings algorithms: Delayed acceptance with prefetching, June 2014. arXiv:1406.2660.

Candela, J. Q. and Rasmussen, C. E. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Cevahir, A., Nukada, A., and Matsuoka, S. Fast Conjugate Gradients with Multiple GPUs. In Allen, G., Nabrzyski, J., Seidel, E., van Albada, G., Dongarra, J., and Sloot, P. (eds.), *Computational Science ICCS 2009*, volume 5544 of *Lecture Notes in Computer Science*, pp. 893–903. Springer Berlin Heidelberg, 2009.

Chen, J., Anitescu, M., and Saad, Y. Computing f(A)b via Least Squares Polynomial Approximations. *SIAM Journal on Scientific Computing*, 33(1):195–222, 2011.

Duchi, J., Hazan, E., and Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, July 2011.

Filippone, M. Bayesian inference for Gaussian process classifiers with annealing and pseudo-marginal MCMC. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pp. 614–619. IEEE, 2014.

Filippone, M. and Girolami, M. Pseudo-marginal Bayesian inference for Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11): 2214–2226, 2014.

Filippone, M., Zhong, M., and Girolami, M. A comparative evaluation of stochastic-based inference methods for Gaussian process models. *Machine Learning*, 93(1):93–114, 2013.

Gelman, A. and Rubin, D. B. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7 (4):457–472, 1992.

Gibbs, M. and MacKay, D. J. C. Efficient Implementation of Gaussian Processes. Technical report, Cavendish Laboratory, Cambridge, UK, 1997.

Gibbs, M. N. *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge, 1997.

Gilboa, E., Saatci, Y., and Cunningham, J. P. Scaling Multidimensional Inference for Structured Gaussian Processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(2): 424–436, 2015.

Golub, G. H. and Van Loan, C. F. *Matrix computations*. The Johns Hopkins University Press, 3rd edition, October 1996.

Gramacy, R. B., Lee, H. K. H., and Macready, W. G. Parameter space exploration with Gaussian process trees. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. ACM, 2004.

Gray, A. G. and Moore, A. W. 'N-Body' Problems in Statistical Learning. In *Advances in Neural Information Processing Systems 13, NIPS, 2000, Denver, CO, USA*, pp. 521–527. MIT Press, 2000.

Hennig, P. Probabilistic Interpretation of Linear Solvers, October 2014. arXiv:1402.2058.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian Processes for Big Data, September 2013. arXiv:1309.6835.

Higham, N. J. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

Jang, Y.-C., Kim, H.-J., and Lee, W. Multi GPU Performance of Conjugate Gradient Solver with Staggered Fermions in Mixed Precision, November 2011. arXiv:1111.0125.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Kennedy, A. D. and Kuti, J. Noise without Noise: A New Monte Carlo Method. *Physical Review Letters*, 54:2473–2476, 1985.

Kennedy, M. C. and O'Hagan, A. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

Liu, K.-F. A Noisy Monte Carlo Algorithm with Fermion Determinant. In Frommer, A., Lippert, T., Medeke, B., and Schilling, K. (eds.), *Numerical Challenges in Lattice Quantum Chromodynamics*, volume 15 of *Lecture Notes in Computational Science and Engineering*, pp. 142–152. Springer Berlin Heidelberg, 2000.

Lyne, A.-M., Girolami, M., Atchade, Y., Strathmann, H., and Simpson, D. On Russian Roulette Estimates for Bayesian inference with Doubly-Intractable Likelihoods, February 2015. arXiv:1306.4032.

Maclaurin, D. and Adams, R. P. Firefly Monte Carlo: Exact MCMC with Subsets of Data, March 2014. arXiv:1403.5693.

Moore, A. The Anchors Hierarchy: Using the Triangle Inequality to Survive High-Dimensional Data. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 397–405. AAAI Press, 2000.

Murray, I. Gaussian processes and fast matrix-vector multiplies, 2009. Presented at the Numerical Mathematics in Machine Learning workshop at the 26th International Conference on Machine Learning (ICML 2009), Montreal, Canada.

Murray, I. and Adams, R. P. Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems 23, NIPS, Vancouver, BC, Canada, 6-9 December 2010*, pp. 1732–1740. Curran Associates, Inc., 2010.

Neal, R. M. Regression and classification using Gaussian process priors (with discussion). *Bayesian Statistics*, 6: 475–501, 1999.

Rasmussen, C. E. and Williams, C. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Robbins, H. and Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

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

Saatçi, Y. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

Seeger, M. Skilling techniques for Bayesian analysis. Technical report, Institute for ANC, Edinburgh, UK, 2000.

Simpson, D. P., Turner, I. W., Strickland, C. M., and Pettitt, A. N. Scalable iterative methods for sampling from massive Gaussian random vectors, December 2013. arXiv:1312.1476.

Skilling, J. Bayesian Numerical Analysis. In Grandy, W. T. and Milonni, P. W. (eds.), *Physics and Probability*, pp. 207–222. Cambridge University Press, 1993. Cambridge Books Online.

Srinivasan, B. V., Hu, Q., Gumerov, N. A., Murtugudde, R., and Duraiswami, R. Preconditioned Krylov solvers for kernel regression, August 2014. arXiv:1408.1237.

Stein, M. L., Chen, J., and Anitescu, M. Stochastic approximation of score functions for Gaussian processes. *The Annals of Applied Statistics*, 7(2):1162–1191, 2013.

Taylor, M. B. and Diggle, J. P. INLA or MCMC? A Tutorial and Comparative Evaluation for Spatial Prediction in log-Gaussian Cox Processes, 2012. arXiv:1202.1738.

Teh, Y. W., Thiéry, A., and Vollmer, S. Consistency and fluctuations for stochastic gradient Langevin dynamics, September 2014. arXiv:1409.0578.

Titsias, M. K. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, AISTATS, Clearwater Beach, FL, USA, April 16-18, 2009*, pp. 567–574. JMLR.org, 2009.

Vollmer, S. J., Zygalakis, K. C., , and Teh, Y. W. (Non-) asymptotic properties of Stochastic Gradient Langevin Dynamics, January 2015. arXiv:1501.00438.

Welling, M. and Teh, Y. W. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 681–688. Omnipress, 2011.

Williams, C. K. I. and Barber, D. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351, 1998.

Williams, C. K. I. and Rasmussen, C. E. Gaussian Processes for Regression. In *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*, pp. 514–520. MIT Press, 1995.