UNIVERSITY OF NICE - SOPHIA ANTIPOLIS
These d'Habilitation a Diriger des Recherches

Presented by Maurizio Filippone
Department of Data Science, EURECOM
`maurizio.filippone@eurecom.fr`

Gaussian Processes for Accurate Quantification of Uncertainty and Large-Scale Learning

Submitted in total fulfillment of the requirements
of the degree of Habilitation a Diriger des Recherches

Contents

Chapter 1

Introduction

1.1 Motivation

Drawing meaningful conclusions on the way complex real life phenomena work and being able to predict the behavior of systems of interest require developing accurate and highly interpretable mathematical models. In order to verify whether such models can actually reproduce the behavior of the system under study in a faithful way, one needs to investigate the agreement of simulations from the model with observed data. A few examples of modeling scenarios that I have been actively involved with after obtaining my Ph.D. in 2008 are the following: (i) detection of neuro-degenerative diseases from neuroimaging data [17, 28, 36] (ii) analysis of regulatory interactions and signaling processes in living cells [13, 32] (iii) analysis of perception of personality and conflict in social interactions from nonverbal cues [30, 22, 23, 41] (iv) prediction of yield of oil reservoirs [29] (v) estimation of disclosure risk [7] (vi) prediction of time series applied to prediction of user availability in network applications and tide levels [6, 12].

In general, mathematical models have parameters that need tuning to allow them to "mimic" the behavior of the system, so an important task is to estimate or infer these parameters based on observed data. The thread connecting most of my works after obtaining my Ph.D. is Bayesian inference. Within this framework, it is possible to determine the degree of uncertainty in parameter estimates and consistently propagate this to predictions and model selection tasks. Also, it is possible to include prior knowledge from domain experts and carry out model selection in a principled way.

The application domains above make a strong case as to why quantification of uncertainty is of paramount importance. For instance, just considering one of these applications, for many neurological and psychiatric disorders, making a definitive diagnosis and predicting clinical outcome are complex and difficult problems. Any decision on the clinical outcome leads to a treatment strategy that has associated costs and has an important impact on subjects' life expectancy and living conditions. Methods that accurately assign a degree of uncertainty to clinical assessments hold the potential to improve mainstream clinical practice and to provide more cost-effective and personalized approaches to treatment.

Despite the appeal of Bayesian inference, carrying out Bayesian computations for any interesting models is analytically intractable, as discussed next. Consider a supervised learning scenario. Let X be a set of n input vectors $\mathbf{x}_i \in \mathbb{R}^d (1 \leq i \leq n)$, and let $\mathbf{y}$ be the vector consisting of the corresponding labels[1] y_i. Consider a probabilistic model for the observed data parameterized by $\boldsymbol{\theta}$ so that the likelihood is $p(\mathbf{y}|X, \boldsymbol{\theta})$, and assume some prior $p(\boldsymbol{\theta})$ over the parameters. The use of Bayesian techniques can be motivated by the following expression for the predictive probability distribution of the label for a test input $\mathbf{x}_*$:

$$p(y_*|\mathbf{x}_*, \mathbf{y}, X) = \int p(y_*|\mathbf{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{y}, X) d\boldsymbol{\theta} \tag{1.1}$$

This expression is central in Bayesian data analysis, as it shows how predictions can be made by averaging out the parameters after the training set of n data has been observed. This expression also shows how crucial it is to be able to compute the posterior probability $p(\boldsymbol{\theta}|\mathbf{y}, X)$ of parameters given the observations. Using Bayes' theorem, the posterior distribution of parameters given the observed data is simply

$$p(\boldsymbol{\theta}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int p(\mathbf{y}|X, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}} \tag{1.2}$$

where $p(\mathbf{y}|X, \boldsymbol{\theta})$ is the so called likelihood, $p(\boldsymbol{\theta})$ is the prior, and the denominator is a normalization that ensures that the posterior is a proper density over the space of $\boldsymbol{\theta}$'s. Equation 1.2 can be interpreted as the update of one's prior belief about model parameters after data are observed. The intractability of Bayesian inference techniques stems from the normalization of equation 1.2 that is generally not available analytically, apart from some special cases (when likelihood and prior form a conjugate pair).

A further and perhaps more urgent challenge for modern day applications of Bayesian inference arises from the fact that it is not even possible to develop mathematical models for some classes of complex systems. Again, considering the problem of diagnosis of neurological and psychiatric disorders from neuroimaging data, it is not clear how to develop a mathematical model that relates brain structure with disease progression. In other cases, even when a mathematical model faithfully describes the evolution of a system, models implemented is computer programs might require computing times that are beyond what is needed to make quantification of uncertainty using Bayesian techniques viable. A representative example is the prediction of yield of oil reservoirs; typically, one run of the underlying models requires hours or sometimes days depending on their complexity. In all these cases, common practice is to employ *nonparametric statistical models* as a surrogate for the unknown model or

[1]Without loss of generality, we assume that the labels are univariate; the extension to multivariate labels is straightforward.

as an emulator for the response of expensive computer simulations. Such a nonparametric treatment, allows one to flexibly and expressively model data, to characterize the uncertainty in model parameters, and to account for the uncertainty due to the lack of knowledge about the underlying model. After obtaining my Ph.D. in 2008, my work has been dedicated to the development of techniques that deal with the intractability of Bayesian inference for nonparametric models, and in particular *Gaussian processes*, as discussed next.

1.2 Bayesian Gaussian Processes for Quantification of Uncertainty

Gaussian Processes (GPs) offer a distribution over functions that can be used for the purpose of determining a distribution over mappings between random variables. In most GP models, the labels $\mathbf{y}$ are assumed to be conditionally independent given a set of n latent variables. Such latent variables are modeled as realizations of a function $f(\mathbf{x})$ at the input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, i.e., $\mathbf{f} = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$. Latent variables are used to express the likelihood function, which under the assumption of independence becomes $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i)$, where $p(y_i|f_i)$ depending on the data being modeled (e.g., Gaussian for regression, Bernoulli for probit classification with probability $P(y_i = 1) = \Phi(f(\mathbf{x}_i))$ where Φ is defined as the cumulative normal distribution).

This model construction is similar to the one in generalized linear modeling; what characterizes GP models, however, is the way the latent variables are specified. In particular, we assume that the function $f(\mathbf{x})$ is distributed as a GP, which implies that the latent function values $\mathbf{f}$ are jointly distributed as a Gaussian $p(\mathbf{f}|X, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ is the covariance matrix. The entries of the covariance matrix $\mathbf{K}$ are specified by a covariance (kernel) function with hyper-parameters $\boldsymbol{\theta}$ between pairs of input vectors. A popular covariance function is the radial basis function covariance defined as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma \exp\left\{ -\frac{1}{\tau^2} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2 \right\} \tag{1.3}$$

The parameter τ defines the length-scale of the interaction between the input vectors, while σ represents the marginal variance for each latent variable. It is possible to extend this for automatic relevance determination of the feature, which takes the form:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma \exp\left\{ -\sum_{r=1}^{d} \frac{1}{{\tau_r}^2} (\mathbf{x}_{i(r)} - \mathbf{x}_{j(r)})^2 \right\} \tag{1.4}$$

The advantage of the ARD covariance is that it accounts for the influence of each feature on the mapping between inputs and labels, with larger values of parameters $(\tau_1, ..., \tau_d)$ indicating a higher influence of the corresponding features [23, 41]. For simplicity of notation, from now on we will denote by $\boldsymbol{\theta}$ the vector of all covariance parameters.

When making predictions, using a point estimate of $\boldsymbol{\theta}$ has been reported to potentially underestimate the uncertainty in predictions or yield inaccurate assessment of the relative influence of different features [17, 16, 4]. Therefore, a Bayesian approach is usually adopted to overcome these limitations, which entails characterizing the posterior distribution over covariance parameters. In order to do so, it is necessary to employ methods, such as Markov chain Monte Carlo (MCMC), that require computing the marginal likelihood $p(\mathbf{y}|X, \boldsymbol{\theta})$ every time $\boldsymbol{\theta}$ is updated. We now discuss the challenges associated with the computation of the marginal likelihood for the special case of a Gaussian likelihood and the more general case of non-Gaussian likelihoods.

Challenge #1: the non-Gaussian likelihood case

In the case of non-Gaussian likelihoods, the likelihood $p(\mathbf{y}|\mathbf{f})$ and the GP prior $p(\mathbf{f}|X, \boldsymbol{\theta})$ do not form a conjugate pair. As a consequence, it is not possible to solve the integral needed to compute the marginal likelihood

$$p(\mathbf{y}|X, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X, \boldsymbol{\theta})d\mathbf{f} \tag{1.5}$$

and this requires some form of approximation. A notable example is GP probit classification, which is what we explore in detail in this paper. In this case, the observations $\mathbf{y}$ are assumed to be Bernoulli distributed with success probability given by $p(y_i|f_i) = \Phi(y_i f_i)$ [35]. For GPs with non-Gaussian likelihoods, there have been several proposals on how to carry out deterministic approximation to integrate out the latent variables. Such approximations recover tractability but introduce bias in predictions and quantification of uncertainty. Crucially, it is not possible to quantify the extent of their impact, and this can be undesirable in applications where quantification of uncertainty is of primary interest. Over the past few years, part of my research activities focused on stochastic approximation techniques, and in particular MCMC techniques that offer guarantees of unbiasedness. Part I of this thesis is dedicated to my contributions to tackle the challenges arising from employing MCMC techniques to carry out unbiased inference of GPs.

Challenge #2: the Gaussian likelihood case

In the GP regression setting, the observations $\mathbf{y}$ are modeled to be Gaussian with mean of $\mathbf{f}$ (latent variables) and covariance $\lambda\mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix, and λ is the variance of the Gaussian noise on the observations. In this setting, the likelihood $p(\mathbf{y}|\mathbf{f})$ and the GP priors $p(\mathbf{f}|\boldsymbol{\theta})$ form a conjugate pair, so latent variables can be integrated out of the model leading to $p(\mathbf{y}|X, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$, where $\mathbf{C} = \mathbf{K} + \lambda\mathbf{I}$. This yields the log-marginal likelihood

$$\log[p(\mathbf{y}|X, \boldsymbol{\theta})] = -\frac{1}{2}\log|\mathbf{C}| - \frac{1}{2}\mathbf{y}^\top\mathbf{C}^{-1}\mathbf{y} + \text{const.} \tag{1.6}$$

in closed form. Although computable, the log-marginal likelihood requires computing the log determinant of $\mathbf{C}$ and solving a linear system involving $\mathbf{C}$. These calculations are usually carried out by factorizing the matrix $\mathbf{C}$ using the Cholesky decomposition, giving $\mathbf{C} = \mathbf{L}\mathbf{L}^\top$, with $\mathbf{L}$ being lower triangular. The Cholesky algorithm requires $O(n^3)$ operations, but subsequently computing the terms of the marginal likelihood requires at most $O(n^2)$ operations [35]. These calculations become unfeasible for n larger than a few thousand and approximations are usually employed to recover tractability. Similarly to the case of non-Gaussian likelihoods, however, the amount of bias that is introduced as a result of these approximations cannot be quantified. I devoted part of my research activities in the last few years to develop techniques that make it possible to avoid factorizations of large matrices altogether without introducing deterministic approximations. Part II of this thesis focuses on such approaches.

1.3 Plan of the thesis

The thesis is divided in two parts.

- In response to Challenge #1, Chapter 2 of the thesis deals with methods that I proposed for **Bayesian inference for Gaussian process models**
- In response to Challenge #2, Chapter 3 of the thesis reports some of my recent works on **Large-scale Gaussian process learning**

The papers included in the thesis contain experiments in various supervised learning problems. The final chapter of the thesis will draw some conclusions and discuss some of the ongoing and future work.

Chapter 2

Bayesian inference for Gaussian process models

This part of the thesis focuses on the developments that I proposed to tractably quantify uncertainty in Gaussian processes without introducing any bias. In particular, these approaches deal with the problem of inferring GP covariance parameters as well as latent variables in the case of non-Gaussian likelihood. As previously discussed, these models do not allow to analytically ingrate out latent variables to obtain the marginal likelihood, and deterministic approximations are usually proposed to recover tractability. The works reported in this part of the thesis propose ways to avoid deterministic approximations by employing MCMC techniques to draw samples from the posterior distribution over covariance parameters and latent variables.

Resources and grants: These activities involved my time and that of a number of collaborators. In particular, the works [17, 18, 16] were published soon after completing my experience as a post-doctoral researcher at University College London working in a collaborators with researchers at King's College London. Most of the funding for these activities came from my funding as a post-doctoral researcher and a three-year EPSRC project (national UK funding body).

- **A comparative evaluation of stochastic-based inference methods for Gaussian process models** In the non-Gaussian likelihood case, it is possible to think of latent variables as model parameters and attempt to employ various off-the-shelf MCMC samplers to characterize the posterior distribution over covariance parameters and latent variables. Loosely speaking, most MCMC approaches work by setting up a Markov chain in the space of model parameters whereby new values of parameters are proposed and then accepted through a given criterion. The proposal mechanisms trades-off exploration versus exploitation of the parameter space and it is based, e.g., on random walks, Hamiltonian or Langevin dynamics, whereas the acceptance of a proposal is so that the Markov chain has the posterior over model parameters as its invariant distribution. The appeal of these techniques is that convergence to the posterior follows under quite general conditions. An important feature of MCMC is ensuring that the Markov chain mixes well and that samples have a good degree of independence, so that samples from the Markov chain represent a sample from the posterior over model parameters.

The application of MCMC techniques to sample from the posterior over covariance parameters and latent variables, however, is not straightforward. The reason is that covariance parameters and latent variables are highly coupled due to the hierarchical dependency between them. As a result, proposing covariance parameters and latent variables jointly within MCMC techniques is going to make it very unlikely to accept any proposals. Sampling covariance parameters and latent variable separately, instead, introduces slow mixing of the Markov chain; the reason is that covariance parameters determine some behavior of the latent variables (e.g., length-scale of oscillations) and sampling latent variables conditioned on covariance parameters will give rise with proposals for latent variables with similar behavior. This work deals with ways to mitigate the coupling effect within MCMC techniques through reparameterization techniques. In particular, this work studies techniques that introduce transformations of latent variables to make the sampling of covariance parameters easier.

- **Probabilistic prediction of neurological disorders with a statistical assessment of neuroimaging data modalities**

  Most of the applied work I carried out on neuroimaging data involves studies on Parkinson's diseases in collaboration with neuroscientists at King's College London and University College London [17, 28, 33]. Some of the Parkinson's diseases are clinically indistinguishable in the early stages, despite having distinct characteristic patterns of molecular pathology. Finding sensitive and specific objective biomarkers for predicting disease state in these disorders is an important aim for several reasons: first, the disorders have different prognoses, where some Parkinson's diseases are characterized by relentless disease progression and carry a life expectancy of only a few years after diagnosis, others do not convey a substantial reduction in life expectancy. Second, the disorders have differential responses to treatment; some Parkinson's diseases respond moderately well to dopaminergic therapy and deep-brain stimulation, whereas others are associated with a poor response. Third, objective biomarkers predictive of early disease state may be useful to reduce the misdiagnosis rate in clinical trials of potential disease-modifying compounds. However, for any objective measure to facilitate clinical decision making in the long term, it must accurately and simultaneously discriminate between all the disorders.

  Magnetic resonance imaging (MRI) holds the potential to provide objective diagnostic markers for the disorders. However, no published studies have demonstrated an automated approach to predict diagnosis in individual subjects with accuracy that could be considered clinically useful. Existing studies have employed either manual measurements derived from radiological examination of MRI scans (rMRI) or automated approaches based on voxel-based morphometry (VBM). Both approaches have disadvantages: rMRI markers are operator-dependent and time-consuming to construct and are not sufficiently specific for discriminating between some Parkinson's diseases despite good sensitivity for discriminating between others. While VBM has been successful in identifying neuroanatomical changes associated with these disorders at the group level, it has limited ability to predict disease state at the level of individual subjects.

In the applications on neuroimaging data, I employed analysis techniques based on nonparametric statistical models. In contrast to rMRI and VBM, this line of work aims to predict disease state at the single-subject level based on distributed patterns of anatomical abnormality. Before our studies, only two studies had considered this methodology to Parkinsonian disorders and were unable to accurately discriminate all diagnostic groups. In this work, we adopted a multinomial logit model based on GP priors as a probabilistic prediction method that provides the means to incorporate measures from different imaging modalities. We aimed to characterize uncertainty without resorting to potentially inaccurate deterministic approximations, and therefore we proposed to employ MCMC methods. The structure of the model and the large number of variables involved, however, make the use of MCMC techniques seriously challenging, and we had to develop a suitable approach to make this feasible. The application of these advanced statistical modeling and inference techniques resulted in significant improvements over the state-of-the art in the discrimination between all diagnostic groups. Furthermore, we demonstrated how to combine data obtained from a variety of neuroimaging measures, and how to assess the importance of these in the discrimination task.

- **Pseudo-Marginal MCMC for Gaussian Processes** In the case of a non-Gaussian likelihood, we discussed how the marginal likelihood of the model is unavailable analytically, so it is not possible to use it directly within MCMC sampling approaches. However, when the marginal likelihood can be unbiasedly approximated, it is possible to resort to so-called Pseudo-Marginal MCMC algorithms [2], whereby the exact marginal likelihood is replaced by an estimate. Perhaps surprisingly, by replacing the exact marginal likelihood with an unbiased approximation does not affect the distribution from which the MCMC approach is sampling from. This observation has deep consequences, as it is possible to devise MCMC approach that rely exclusively on approximate and tractable computations without introducing approximations. The price pay for this is that the Markov chain can mix slowly when the variance of the estimated marginal likelihood is large. A large variance in the estimate of the marginal likelihood can eventually lead to the acceptance of a parameter within the MCMC simulation because the corresponding marginal likelihood is overestimated. If the overestimation is severe, it is unlikely that any new proposal will be accepted, resulting in slow convergence and low efficiency.

  In the case of GPs, we developed pseudo-marginal MCMC algorithms to draw samples from the posterior over covariance parameters by unbiasedly estimating the marginal likelihood using importance sampling. The key here is that the marginal likelihood can be unbiasedly estimated relying on popular approximations developed in the literature of GPs to integrate out latent variables. This so-called Pseudo-Marginal MCMC approach can be shown to yield samples from the exact posterior distribution over covariance parameters, but it is based on computations that are tractable. A key question that we aimed to address in the paper is whether the proposed "exact" MCMC actually yields any noticeable advantages in quantification of uncertainty in predictions. We addressed this by comparing the proposed fully Bayesian approach with GPs, treated with various degrees of approximations, and Support Vector Machines. The results indicate that the proposed approach outperforms the competitors in characterizing the degree of uncertainty in predictions, especially for small data sets.

A Comparative Evaluation of Stochastic-based Inference Methods for Gaussian Process Models

M. Filippone · M. Zhong · M. Girolami

Abstract Gaussian Process (GP) models are extensively used in data analysis given their flexible modeling capabilities and interpretability. The fully Bayesian treatment of GP models is analytically intractable, and therefore it is necessary to resort to either deterministic or stochastic approximations. This paper focuses on stochastic-based inference techniques. After discussing the challenges associated with the fully Bayesian treatment of GP models, a number of inference strategies based on Markov chain Monte Carlo methods are presented and rigorously assessed. In particular, strategies based on efficient parameterizations and efficient proposal mechanisms are extensively compared on simulated and real data on the basis of convergence speed, sampling efficiency, and computational cost.

Keywords Bayesian inference · Gaussian Processes · Markov chain Monte Carlo · hierarchical models · latent variable models

1 Introduction

Gaussian Process (GP) models represent a class of models that are popular in data analysis due to the associated flexibility and interpretability. Both these features are a direct consequence of their rich parameterization. Flexibility is due to the nonparametric prior over latent variables conditioning observations, whereas interpretability is due to the parameterization of the structure associated with the latent variables. Observations are conditionally independent given a set of jointly Gaussian latent variables, and are assumed to be distributed according to the particular type of data being modeled. The covariance structure of the latent variables

Maurizio Filippone
School of Computing Science, University of Glasgow, United Kingdom.
E-mail: maurizio.filippone@glasgow.ac.uk

Mingjun Zhong
Department of Biomedical Engineering, Dalian University of Technology, P.R. China
E-mail: mingjun.zhong@gmail.com

Mark Girolami
Department of Statistical Science, University College London, United Kingdom.
E-mail: girolami@stats.ucl.ac.uk

is then parameterized by a set of hyper-parameters that characterizes the covariance of the input vectors in terms of length-scales and intensity of interaction. GP models comprise a large set of models, and this paper focuses in particular on Logistic Regression with GP priors (`LRG`) (Rasmussen and Williams, 2006), Log-Gaussian Cox models (`LCX`) (Møller et al., 1998), Stochastic Volatility models with GP priors (`VLT`) (Wilson and Ghahramani, 2010), and Ordinal Regression with GP priors (`ORD`) (Chu and Ghahramani, 2005).

Exact inference in GP models is analytically intractable. Most of the work to tackle such intractability focuses on deterministic approximations to integrate out latent variables; those approaches include the Laplace Approximation (LA) (Tierney and Kadane, 1986), Expectation Propagation (EP) (Minka, 2001), and mean field approximations (Opper and Winther, 2000) (see, e.g., Rasmussen and Williams (2006) for an extensive presentation of such approximations, and Kuss and Rasmussen (2005) for their assessment on `LRG` models). Deterministic approximations provide a computationally tractable way to integrate out latent variables, but it is not possible to quantify the error that they introduce in the quantification of uncertainty in predictions (although EP for `LRG` is reported to be very accurate in Kuss and Rasmussen (2005)); also, those methods target the integration of latent variables only.

In the direction of providing a fully Bayesian treatment of GP models, it is necessary to integrate out latent variables as well as hyper-parameters, and this is usually done by quadrature methods (Cseke and Heskes, 2011; Rue et al., 2009), thus limiting the number of hyper-parameters that can be employed in GP models.

Based on those considerations, this paper focuses on non-deterministic methods to carry out inference in GP models, and in particular on stochastic based approximations based Markov Chain Monte Carlo (MCMC) methods. The use of MCMC based inference methods is appealing as it provides asymptotic guarantees of convergence to exact inference. In practice, this translates into the possibility of achieving results with the desired level of accuracy (Flegal et al., 2007). Unfortunately, the use of MCMC methods for inference in GP models is extremely difficult. The aim of this paper is to discuss the challenges associated with MCMC based inference for GP models, and compare a number of strategies that have been proposed in the literature to tackle them. A preliminary version of this work can be found in Filippone et al. (2012)[1].

To the best of our knowledge, this work (i) is the first attempt to extensively assess the state-of-the-art in stochastic-based inference methods for GP models, and (ii) sets the bar for new MCMC methods for inference in GP models. Along with those contributions, this paper presents (iii) a variant of the Hybrid Monte Carlo algorithm that outperforms state-of-the-art methods to sample from the posterior distribution of the latent variables, and (iv) tests the combination of parameterizations, as recently proposed in Yu and Meng (2011), in the case of GP models.

[1] An implementation of the methods considered in this paper can be found at: `http://www.dcs.gla.ac.uk/~maurizio/pages/code.html`

1.1 Gaussian Process Models

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of n input vectors described by a set of d covariates $\mathbf{x}_i \in \mathbb{R}^d$, associated with observed responses $\mathbf{y} = \{y_1, \ldots, y_n\}$. In GP models, the generative process modeling the observed data $\mathbf{y}$ given X is as follows. Observations are assumed to be conditionally independent given a set of n latent variables $\mathbf{f} = \{f_1, \ldots, f_n\}$, and distributed according to a certain distribution depending on the particular type of data, e.g., Bernoulli for binary labels and Poisson for observations in the form of counts. This can be translated into a likelihood function of the form $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n p(y_i|f_i)$, where for generality the distribution $p(y_i|f_i)$ is left unspecified.

In this work, latent variables are assumed to be drawn from a zero mean GP prior with covariance function k. The GP prior is a prior over functions, and the covariance structure given by k specifies the characteristics of such functions (i.e., degree of smoothness and marginal variance). Let k be parameterized by a vector of hyper-parameters $\boldsymbol{\theta} = (\sigma, \psi_{\tau_1}, \ldots, \psi_{\tau_d})$, and assume:

$$k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}) = \sigma q(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\psi}_{\boldsymbol{\tau}}) = \sigma \exp\left[-\frac{1}{2}\sum_{r=1}^{d} \frac{(\mathbf{x}_i - \mathbf{x}_j)^2_{(r)}}{\exp(\psi_{\tau_r})^2}\right] \tag{1}$$

with $\exp(\psi_{\tau_r})$ defining the length-scale of the interaction between the input vectors for the rth covariate and σ giving the marginal variance for latent variables. This type of covariance can be used for Automatic Relevance Determination (ARD) (Mackay, 1994) of the covariates, as the values $\tau_i = \exp(\psi_{\tau_i})$ can be interpreted as length-scale parameters. This definition of covariance function is adopted in many applications and is the one we will consider in the remainder of this paper. Exponentiation of the hyper-parameters is convenient, so that standard MCMC transition operators can be employed for ψ_{τ_i} thus avoiding dealing with boundary conditions or non-standard MCMC proposals (Robert and Casella, 2005). Let Q be the matrix whose entries are $q_{ij} = q(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\psi}_{\boldsymbol{\tau}})$; the covariance matrix K will then be $K = \sigma Q$. The model is fully specified by choosing a prior $p(\boldsymbol{\theta})$ for the hyper-parameters. The model structure is therefore hierarchical, with hyper-parameters conditioning the latent variables that, in turn, condition observations, so that $p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})p(\boldsymbol{\theta})$.

In a Bayesian setting, the predictive distribution for new input values $\mathbf{x}_*$ can be written in the following way (for the sake of clarity we drop the explicit conditioning on X and $\mathbf{x}_*$):

$$p(y_*|\mathbf{y}) = \int\int\int p(y_*|f_*)p(f_*|\mathbf{f}, \boldsymbol{\theta})p(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y})df_* d\mathbf{f} d\boldsymbol{\theta} \tag{2}$$

The left hand side of Eq. 2 is a full probability distribution characterizing the uncertainty in predicting y_* given the GP modeling assumption.

In this work we will focus on stochastic approximations for obtaining samples from the posterior distribution of $\mathbf{f}$ and $\boldsymbol{\theta}$, so that we can obtain a Monte Carlo estimate of the predictive distribution as follows:

$$p(y_*|\mathbf{y}) \simeq \frac{1}{N}\sum_{i=1}^{N} \int p(y_*|f_*)p(f_*|\mathbf{f}^{(i)}, \boldsymbol{\theta}^{(i)})df_* \tag{3}$$

where N denotes the number of samples used to compute the estimate. In Eq. 3 we denoted the ith samples from the posterior distribution of $\mathbf{f}$ and $\boldsymbol{\theta}$ obtained by means of MCMC methods by $\mathbf{f}^{(i)}$ and $\boldsymbol{\theta}^{(i)}$. Note that the remaining integral is univariate and it is generally easy to evaluate.

1.2 Challenges in MCMC based inference for GP models

Sampling from the posterior of latent variables and hyper-parameters by joint proposals is not feasible; it is extremely unlikely to propose a set of latent variables and hyper-parameters that are compatible with each other and observed data. This forces one to consider schemes such as Gibbs sampling, where groups of variables are updated one at time, leading to the following challenges:

(i) Due to the hierarchical structure of GP models, chains converge slowly and mix poorly if the coupling effect between the groups of variables is not dealt with properly. This requires some form of reparameterization or clever proposal mechanism that efficiently decouples the dependencies between the groups of variables. This effect has drawn a lot of attention in the case of hierarchical models in general (Yu and Meng, 2011), and recently in GP models Knorr-Held and Rue (2002); Murray and Adams (2010). In Knorr-Held and Rue (2002) a joint update of latent variables and hyper-parameters is proposed with the aim of avoiding proposals for hyper-parameters to be conditioned on the values of latent variables. In Murray and Adams (2010) a parameterization based on auxiliary data is proposed that aims at reducing the coupling between the two groups of variables. Other ideas involve the use of reparameterizations based on whitening the latent variables; in the terminology of Yu and Meng (2011), this corresponds to employing the so called Ancillary Augmentation (AA) parameterization. Recently, Yu and Meng (2011) proposed to interweave parameterizations characterized by complementary features in order to boost sampling efficiency. Parameterizations can be complementary in the sense that they offer better performance in either strong or weak data limits; the idea of combining parameterizations is to achieve high sampling efficiency in both strong and weak data scenarios. We are interested in comparing the methods in Knorr-Held and Rue (2002); Murray and Adams (2010) and Yu and Meng (2011) applied to GP models. Another possibility would be to approximately integrate out latent variables and obtain samples from the corresponding approximate posterior of hyper-parameters. For GP classification this might be a sensible thing to do, as the Expectation Propagation approximation has been reported to be very accurate Kuss and Rasmussen (2005); however, this is peculiar to GP classification and for general GP models it may not be the case.

(ii) Sampling hyper-parameters and latent variables cannot be done using exact Gibbs steps, and it requires proposals that are accepted/rejected based on a Hastings ratio, leading to a waste of expensive computations. Transition operators characterized by acceptance mechanisms embedded in a Gibbs sampler, are usually referred to as Metropolis-within-Gibbs operators. Designing proposals that guarantee high acceptance and independence between samples is extremely challenging, especially because latent variables can have dimensions in the order of hundreds or thousands. We will compare several transition operators, for different steps of the Gibbs sampler, with the aim of gaining insights about ways to strike a good balance between efficiency and computational cost. We will consider transition operators characterized by proposal mechanisms with increasing complexity, and in particular the Metropolis-Hastings (MH) operator which is based on random walk types of proposals, Hybrid Monte Carlo (HMC) which uses the gradient of the log-density of interest, and manifold methods (Girolami and Calderhead, 2011) which use curvature information (i.e., second derivatives of the log-density).

The paper is organized as follows: Sections 2 and 3 report the parameterization strategies and the transition operators considered in this work. Sections 4 and 5 report an extensive comparison of those strategies and transition operators, on simulated and real data, on the basis of efficiency, convergence speed and computational complexity; section 6 concludes the paper. For the sake of readability, most of the technical derivations can be found in the appendices.

2 Dealing with the hierarchical structure of GP models

2.1 Sufficient and Ancillary Augmentation

From a generative perspective, the model structure is hierarchical with latent variables representing sufficient statistics for the hyper-parameters. This parameterization is referred to as Sufficient Augmentation (SA) in Yu and Meng (2011) and allows one to express the joint density as

$$\text{SA} \qquad p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{4}$$

It is also possible to introduce the decomposition of the matrix Q into the product of two factors LL^{T}, and view the generation of the latent variables as $\mathbf{f} = \sqrt{\sigma}L\boldsymbol{\nu}$ with $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{\nu}|\mathbf{0}, I)$, which implies that $\mathbf{f} \sim \mathcal{N}(\mathbf{f}|\mathbf{0}, K)$. In the remainder of this paper, we will consider L to be the lower triangular Cholesky decomposition of K, but in principle any square root of K could be used. In this way, $\boldsymbol{\nu}$ is ancillary for $\boldsymbol{\theta}$ and it is possible to express the joint density as

$$\text{AA} \qquad p(\mathbf{y}, \boldsymbol{\nu}, \boldsymbol{\theta}) = p(\mathbf{y}|\boldsymbol{\nu}, \boldsymbol{\theta})p(\boldsymbol{\nu})p(\boldsymbol{\theta}) \tag{5}$$

This parameterization is called Ancillary Augmentation (AA) in the terminology of Yu and Meng (2011). In Murray and Adams (2010) SA and AA are referred to as unwhitened and whitened parameterizations respectively. Weak and strong data limits can influence the efficiency in sampling using either parameterization. For this reason, it is important to choose an efficient parameterizations for the particular problem under study and for the available amount of data, as both these aspects can dramatically influence efficiency and convergence speed of the chains.

2.2 Ancillarity-Sufficiency Interweaving Strategy - ASIS

In this section we briefly review the main results presented in Yu and Meng (2011) on the combination of parameterizations to improve convergence and efficiency of MCMC methods, and we will illustrate how these results can be applied to GP models. Intuitively, combining parameterizations seems promising to take the best from them in both weak and strong data limits, or at least, to avoid the possibility that chains do not converge because of the wrong choice of parameterization. Alternating the sampling in the SA and AA parameterizations is the most obvious way of combining the two parameterizations, but as recently investigated in Yu and Meng (2011), interweaving SA and AA is actually a more promising way forward. From a theoretical perspective, the geometric rate of convergence r of the scheme when the parameterizations are interweaved, is related to the rates of the two schemes r_1 and r_2 by $r \leq R_{1,2}\sqrt{r_1 r_2}$, where $R_{1,2}$ is the maximal correlation between the latent variables for the two schemes. Given that the former expression implies $r \leq \max(r_1, r_2)$, combining the two parameterizations leads to a scheme that is better than the worst. This is already an advantage compared to using a single scheme when one is in doubt on which scheme to use. However, the key result is the fact that $R_{1,2}$ can be very small depending on the two parameterizations, so it is possible to make the combined scheme converge quickly even if neither of the individual schemes do. In general, this result is quite remarkable, as once different reparameterizations are available, combining them using the interweaving strategy is simple to implement, and can dramatically boost sampling efficiency. In GP models, the ASIS scheme amounts to interweaving SA and AA updates, that following Yu and Meng (2011) yields:

$$\mathbf{f}|\mathbf{y}, \boldsymbol{\theta} \quad \longrightarrow \quad \boldsymbol{\theta}|\mathbf{f} \quad \longrightarrow \quad \boldsymbol{\nu} = \sigma^{-1/2} L^{-1}\mathbf{f} \quad \longrightarrow \quad \boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\nu} \tag{6}$$

2.3 Knorr-Held and Rue (KHR)

The idea underpinning KHR, is to jointly sample parameters and latent variables as follows. Firstly, a set of hyper-parameters $\boldsymbol{\theta}'|\boldsymbol{\theta}$ is proposed and secondly a set of latent variables conditioned on the new set of hyper-parameters, namely $\mathbf{f}'|\mathbf{y}, \boldsymbol{\theta}'$, is proposed. The proposal $(\boldsymbol{\theta}', \mathbf{f}')$ is then jointly accepted or rejected according to a standard Hastings ratio. The key idea is to avoid making the proposal $\boldsymbol{\theta}'$ accepted on the basis of $\mathbf{f}$ to avoid the strong coupling effect due to the hierarchical nature of the model. KHR was proposed in applications making use of Gaussian Markov Random Fields, and we will discuss the application of this idea for GP models in the section reporting the experiments. In order to avoid difficulties in devising a proposal for sampling from $\mathbf{f}'|\mathbf{y}, \boldsymbol{\theta}')$, here we set the proposal as the Gaussian obtained by constructing a Laplace approximation to $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$.

2.4 Surrogate Method (SURR)

In the SURR method (Murray and Adams, 2010), a set of auxiliary latent variables $\mathbf{g}$ is introduced as a noisy version of $\mathbf{f}$; in particular, $p(\mathbf{g}|\mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{g}|\mathbf{f}, S_{\boldsymbol{\theta}})$. This construction yields a conditional distribution for $\mathbf{f}$ of the form $p(\mathbf{f}|\mathbf{g}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, R)$, with $R = S_{\boldsymbol{\theta}} - S_{\boldsymbol{\theta}}(S_{\boldsymbol{\theta}} + K)^{-1} S_{\boldsymbol{\theta}}$ and $\mathbf{m} = R S_{\boldsymbol{\theta}}^{-1}\mathbf{g}$. After decomposing $R = DD^{\mathrm{T}}$, the sampling of $\boldsymbol{\theta}$ is then conditioned on the variables $\boldsymbol{\eta}$ defined as $\mathbf{f} = D\boldsymbol{\eta} + \mathbf{m}$. The covariance $S_{\boldsymbol{\theta}}$ is constructed to be diagonal with elements obtained by matching the posterior for each latent variable individually or by Taylor approximations (see Murray and Adams (2010) for details).

3 MCMC transition operators considered in this work

This section presents the transition operators considered in this work. We are interested in understanding whether and to what extent employing proposal mechanisms making use of gradient or curvature information of the target density improves sampling efficiency and speed of convergence with respect to computational complexity. We therefore consider transition operators with increasing complexity, and in particular the Metropolis-Hastings (MH) operator which is based on random walk types of proposals, the Hybrid Monte Carlo (HMC) operator which uses gradient information, and the Simplified Manifold Metropolis Adjusted Langevin Algorithm (SMMALA) operator which is one of the simplest manifold MCMC methods proposed in Girolami and Calderhead (2011) using curvature information.

For the sake of clarity, we will focus on the transitions operators for $\mathbf{f}$, but the same operators can be easily applied to $\boldsymbol{\theta}$. We will first present MH, HMC, and SMMALA, and we will then discuss Elliptical Slice Sampling and a few variants of MH and HMC that have been specifically proposed for sampling $\mathbf{f}$, and do not have counterparts for $\boldsymbol{\theta}$. In the case of latent variables, the operators aim to leave the posterior $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ invariant; in the remainder of this work, $W(\mathbf{f})$ is defined as $\log[p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})]$, which equals the log of the desired target density up to constants. In the case of hyper-parameters we can define the invariant distribution according to the chosen parameterization and apply the operators presented here for sampling $\boldsymbol{\theta}$ rather than $\mathbf{f}$.

3.1 Metropolis-Hastings - MH

The Metropolis-Hastings transition operator employs a proposal mechanism $g(\mathbf{f}'|\mathbf{f})$ based on a random walk (Robert and Casella, 2005). A common choice is to use a multivariate Gaussian proposal with covariance Σ centered at the former position $\mathbf{f}$, thus taking the form $g(\mathbf{f}'|\mathbf{f}) = \mathcal{N}(\mathbf{f}'|\mathbf{f}, \Sigma)$. For such a symmetric proposal mechanism, $\mathbf{f}'$ is then accepted with probability $\min\left\{1, \exp(W(\mathbf{f}') - W(\mathbf{f}))\right\}$.

3.2 Hybrid Monte Carlo - HMC

In Hybrid Monte Carlo (HMC) the proposals are based on the analogy of a physical system, where a particle is simulated moving in a potential field (Neal, 1993). An auxiliary variable $\mathbf{p}$, that plays the role of a momentum variable, is drawn from $\mathcal{N}(\mathbf{p}|\mathbf{0}, M)$, where the covariance matrix M is the so called mass matrix. The joint density of $\mathbf{f}$ and $\mathbf{p}$ factorizes as $p(\mathbf{f}, \mathbf{p}) = \exp(W(\mathbf{f}))p(\mathbf{p})$, and the negative log-joint density reads

$$H(\mathbf{f}, \mathbf{p}) = -W(\mathbf{f}) + \frac{1}{2}\log(|M|) + \frac{1}{2}\mathbf{p}^{\mathrm{T}} M^{-1}\mathbf{p} + \text{const.} \qquad (7)$$

This is the Hamiltonian of the simulated particle, where the potential field is given by $-W(\mathbf{f})$ and the kinetic energy by the quadratic form in $\mathbf{p}$. In order to draw proposals from $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$, we can simulate the particle for a certain time interval, introducing an analogous of time t and solving Hamilton's equations

$$\frac{d\mathbf{f}}{dt} = \frac{\partial H}{\partial \mathbf{p}} = M^{-1}\mathbf{p} \qquad \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{f}} = \nabla_{\mathbf{f}} W \tag{8}$$

Given that there is no friction, the energy will be conserved during the motion of the particle. Solving Hamilton's equations directly for general potential fields, however, is analytically intractable, and therefore it is necessary to resort to schemes where time is discretized. The leapfrog integrator discretizes the dynamics in λ steps, also known as leapfrog steps, and is volume preserving and reversible (see Neal (1993) for details). The leapfrog integrator yields an update of $(\mathbf{f}, \mathbf{p})$ into $(\mathbf{f}_{(\lambda)}, \mathbf{p}_{(\lambda)})$. The discretization introduces an approximation such that the total energy is not conserved, so a Metropolis accept/reject step of the form $\min\{1, \exp(-H(\mathbf{f}_{(\lambda)}, \mathbf{p}_{(\lambda)}) + H(\mathbf{f}, \mathbf{p}))\}$ is needed to ensure that HMC samples from the correct invariant distribution. The HMC transition operator is reported in Algorithm 1.

Algorithm 1 HMC transition operator when $M = L_M L_M^{\mathrm{T}}$

```
 1: $\mathbf{f}_{(0)} = \mathbf{f}$; $\mathbf{p}_{(0)} \sim \mathcal{N}(\mathbf{p}_{(0)}|\mathbf{0}, M)$        ▷ $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$; $\mathbf{p}_{(0)} = L_M \mathbf{z}$
 2: $\lambda = \mathrm{sample}[1, \ldots, \lambda_{\max}]$
 3: for ($t = 0$ to $\lambda - 1$) do
 4:     $\mathbf{p}_{(t+1/2)} = \mathbf{p}_{(t)} + \frac{\varepsilon}{2}\nabla_{\mathbf{f}} W(\mathbf{f}_{(t)})$
 5:     $\mathbf{f}_{(t+1)} = \mathbf{f}_{(t)} + \varepsilon M^{-1}\mathbf{p}_{(t+1/2)}$        ▷ $M^{-1}\mathbf{p} = \mathtt{bcksub}(L_M^{\mathrm{T}}, (\mathtt{fwdsub}(L_M, \mathbf{p})))$
 6:     $\mathbf{p}_{(t+1)} = \mathbf{p}_{(1/2)} + \frac{\varepsilon}{2}\nabla_{\mathbf{f}} W(\mathbf{f}_{(t+1)})$
 7: end for
 8: $r = \min\left\{0, H(\mathbf{f}_{(0)}, \mathbf{p}_{(0)}) - H(\mathbf{f}_{(\lambda)}, \mathbf{p}_{(\lambda)})\right\}$        ▷ $\log|M| = 2\sum_i \log(L_M)_{ii}$
                                                        ▷ $\mathbf{p}^{\mathrm{T}} M^{-1}\mathbf{p} = \|\mathtt{fwdsub}(L_M, \mathbf{p})\|^2$
 9: $u \sim \mathrm{Exp}(u|1)$
10: if ($r > -u$) then return $\mathbf{f}_{(\lambda)}$
11: else return $\mathbf{f}_{(0)}$
```

3.3 Manifold MCMC - Simplified Manifold MALA - SMMALA

Manifold MCMC methods (Girolami and Calderhead, 2011) were proposed to have an automatic mechanism to tune parameters in MALA and HMC, and are based on the use of curvature through the Fisher Information (FI) matrix. The FI matrix and the Christoffel symbols are the key quantities in information geometry as they characterize the curvature and the connection on the statistical manifold respectively. Consider a statistical model $S = \{p(\mathbf{y}|\boldsymbol{\psi})|\boldsymbol{\psi} \in \Psi\}$ where $\mathbf{y}$ denotes observed variables and $\boldsymbol{\psi}$ comprises all model parameters. Under conditions that are generally satisfied for most commonly used models (Amari and Nagaoka, 2000), S can be considered a C^{∞} manifold, and is called statistical manifold. Let $\mathcal{L} = \log[p(\mathbf{y}|\boldsymbol{\psi})]$; the FI matrix G of S at $\boldsymbol{\psi}$ is defined as:

$$G(\boldsymbol{\psi}) = \mathrm{E}_{p(\mathbf{y}|\boldsymbol{\psi})}\left[(\nabla_{\boldsymbol{\psi}}\mathcal{L})(\nabla_{\boldsymbol{\psi}}\mathcal{L})^{\mathrm{T}}\right] = -\mathrm{E}_{p(\mathbf{y}|\boldsymbol{\psi})}[\nabla_{\boldsymbol{\psi}}\nabla_{\boldsymbol{\psi}}\mathcal{L}] \tag{9}$$

By definition, the FI matrix is positive semidefinite, and can be considered as the natural metric on S.

In the case of GP models that are hierarchical we need to consider the statistical manifolds associated with the two levels of the hierarchy separately. Let's focus on the statistical manifold associated with the model for $\mathbf{y}$ given $\mathbf{f}$. The manifold MALA (MMALA) algorithm (Girolami and Calderhead, 2011) defines a Langevin diffusion with stationary distribution $p(\mathbf{f}|\boldsymbol{\theta}, \mathbf{y})$ on the Riemann manifold of density functions, characterized by a metric tensor denoted as $G_{\mathbf{f},\mathbf{f}}$. By employing a first order Euler integrator to solve the diffusion, a proposal mechanism with density $g(\mathbf{f}'|\mathbf{f}) = \mathcal{N}(\mathbf{f}'|\boldsymbol{\mu}(\mathbf{f}, \epsilon), \epsilon^2 G_{\mathbf{f},\mathbf{f}}^{-1})$ is obtained, where ϵ is the integration step size, a parameter which needs to be tuned, and the dth component of the mean function $\boldsymbol{\mu}(\mathbf{f}, \epsilon)_d$ is

$$\boldsymbol{\mu}(\mathbf{f}, \epsilon)_d = \mathbf{f}_d + \frac{\epsilon^2}{2} \left(G_{\mathbf{f},\mathbf{f}}^{-1} \nabla_{\mathbf{f}} W(\mathbf{f}) \right)_d - \epsilon^2 \sum_{i=1}^{n} \sum_{j=1}^{n} (G_{\mathbf{f},\mathbf{f}}^{-1})_{i,j} \Gamma_{i,j}^d$$

where $\Gamma_{i,j}^d$ are the Christoffel symbols of the metric in local coordinates (Amari and Nagaoka, 2000). Similarly to MALA (Roberts and Stramer, 2002), due to the discretization error introduced by the first order approximation, convergence to the stationary distribution is not guaranteed anymore and thus a standard Metropolis accept/reject step is employed to correct this bias.

In the same spirit, it is possible to extend HMC to define Hamilton's equations on the statistical manifold. This was proposed and applied in Girolami and Calderhead (2011) and called Riemann manifold Hamiltonian Monte Carlo (RM-HMC). In this work, we will not consider RM-HMC or MMALA, as they both require the derivatives of the FI matrix that would require several expensive operations. Instead, we will consider a simplified version of MMALA (SMMALA), where we assume a manifold with constant curvature, that effectively removes the term depending on the Christoffel symbols, so that the mean of the proposal of SMMALA becomes

$$\boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon) = \mathbf{f} + \frac{\epsilon^2}{2} G_{\mathbf{f},\mathbf{f}}^{-1} \nabla_{\mathbf{f}} W(\mathbf{f}) \tag{10}$$

Furthermore, in the last subsection of this section we will present two variants of HMC that bear some similarities with RM-HMC but are computationally cheaper. The SMMALA transition operator is sketched in Algorithm 2.

Algorithm 2 SMMALA transition operator

```
1: $\boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon) = \mathbf{f} + \frac{\epsilon^2}{2} G_{\mathbf{f},\mathbf{f}}^{-1} \nabla_{\mathbf{f}} W(\mathbf{f})$        ▷ $G_{\mathbf{f},\mathbf{f}} = L_G L_G^{\mathrm{T}}$
                    ▷ $G_{\mathbf{f},\mathbf{f}}^{-1} \nabla_{\mathbf{f}} W(\mathbf{f}) = \texttt{bcksub}(L_G^{\mathrm{T}}, (\texttt{fwdsub}(L_G, \nabla_{\mathbf{f}} W(\mathbf{f}))))$
2: $\mathbf{f}' \sim \mathcal{N}(\mathbf{f}'|\boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon), \epsilon^2 G_{\mathbf{f},\mathbf{f}}^{-1})$        ▷ $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$; $\mathbf{f}' = \varepsilon\, \texttt{bcksub}(L_G^{\mathrm{T}}, \mathbf{z}) + \boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon)$
3: $r = \min\{0, W(\mathbf{f}') - W(\mathbf{f}) + \log[g(\mathbf{f}|\mathbf{f}')] - \log[g(\mathbf{f}'|\mathbf{f})]\}$        ▷ $\log|G_{\mathbf{f},\mathbf{f}}| = 2\sum_i \log(L_G)_{ii}$
                    ▷ $(\mathbf{f}' - \boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon))^{\mathrm{T}} G_{\mathbf{f},\mathbf{f}}^{-1} (\mathbf{f}' - \boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon)) = \|\texttt{fwdsub}(L_G, (\mathbf{f}' - \boldsymbol{\mu}_{\mathrm{s}}(\mathbf{f}, \epsilon)))\|^2$
4: $u \sim \mathrm{Exp}(u|1)$
5: if $(r > -u)$ then return $\mathbf{f}'$
6: else return $\mathbf{f}$
```

3.4 Elliptical Slice sampling - ELL-SS

Elliptical Slice Sampling (ELL-SS) has been proposed in Murray et al. (2010) to draw samples for $\mathbf{f}$ in GP models, and is based on slice sampling (Neal, 2003). Due to the fact that latent variables are Gaussian, it is possible to derive this particular version of slice sampling, when constrained on an ellipse. For completeness, we report the transition operator in Algorithm 3 and we refer the reader to Murray et al. (2010) for further details. Note that ELL-SS is quite appealing as it returns

Algorithm 3 ELL-SS transition operator

```
 1: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, K)$
 2: $u \sim \mathrm{Exp}(u|1)$    $\eta = \log p(\mathbf{y}|\mathbf{f}) - u$    ▷ Set a threshold on the log-likelihood
 3: $\alpha \sim U[0, 2\pi]$    $[\alpha_{\min}, \alpha_{\max}] = [\alpha - 2\pi, \alpha]$    ▷ Define the bracket
 4: $\mathbf{f}' = \mathbf{f}\cos(\alpha) + \mathbf{z}\sin(\alpha)$
 5: if $(\log p(\mathbf{y}|\mathbf{f}') > \eta)$ then return $\mathbf{f}'$
 6: else    ▷ Shrink the bracket
 7:     if $(\alpha < 0)$ then $\alpha_{\min} = 0$
 8:     else $\alpha_{\max} = 0$
 9:     $\alpha \sim U[\alpha_{\min}, \alpha_{\max}]$
10:     Go to 4
```

a sample which does not need to be accepted or rejected (in fact, a rejection mechanism is implicit within step 5), and the proposal mechanism does not have any free parameters that need tuning.

3.5 Scaled versions of MH - MH v1 and MH v2

Due to the strong correlation of latent variables imposed by the GP prior, employing a MH operator with an isotropic covariance to sample latent variables leads to extremely poor efficiency. In order to overcome this problem, Neal (1999) proposed two versions of MH that we will denote by MH v1 and MH v2. In MH v1, a set of latent variables $\mathbf{z}$ is drawn from the GP prior $\mathbf{z} \sim \mathcal{N}(\mathbf{z}|0, K)$, and the proposal is constructed as follows:

$$\mathbf{f}' = \mathbf{f} + \alpha\,\mathbf{z} \tag{11}$$

where the parameter α controls the degree of update. In MH v2, instead, the proposal is as follows:

$$\mathbf{f}' = \sqrt{1-\alpha^2}\,\mathbf{f} + \alpha\,\mathbf{z} \tag{12}$$

In the latter case, given that the proposal satisfies detailed balance with respect to the prior, the acceptance has to be based on the likelihood alone.

3.6 Scaled versions of HMC - HMC v1 and HMC v2

By a similar argument as in MH, it is possible to introduce scaled versions of HMC that reduce the correlation between latent variables. This can be done by setting the mass matrix of HMC according to the precision of the posterior distribution of latent variables. Similarly, from an information geometric perspective, it is sensible to whiten latent variables according to the metric tensor of the statistical manifold. We notice that the metric tensor associated to the model for $\mathbf{y}$ given $\mathbf{f}$ is K^{-1} plus a diagonal matrix which is a function of $\mathbf{f}$ (see appendix A for full details). Whitening with respect to that metric tensor would be computationally very expensive for GP models, as it would require the simulation of the Hamiltonian dynamics on a manifold with a position-dependent curvature; this is implemented by RM-HMC which requires the derivatives of the metric tensor as well as implicit leapfrog iterations (Girolami and Calderhead, 2011). In order to reduce the computational cost, we propose the following two options: (i) to approximate the diagonal term to be independent of $\mathbf{f}$ so that $M^{-1} = (K^{-1}+C)^{-1} = C^{-1} - C^{-1}(K+C^{-1})^{-1}C^{-1}$ with C diagonal and independent of $\mathbf{f}$; we call this variant HMC v1. (ii) to ignore the diagonal part of the metric tensor and set $M^{-1} = K$; we call this variant HMC v2. In HMC v1, one simple way to make C independent of $\mathbf{f}$ is to compute it for the GP prior mean (which is zero), as proposed, e.g., in Christensen et al. (2005); Vanhatalo and Vehtari (2007).

In both cases, it is possible to employ a standard and computationally efficient HMC proposal that captures part of the curvature of the statistical manifold. This is achieved by introducing a variant of HMC that, rather than using the Cholesky decomposition of the mass matrix, uses the decomposition of its inverse. We report this variant of the HMC transition operator in Algorithm 4.

In HMC v1, employing this formulation of HMC is convenient as computing the inverse of M is more stable than computing $M = K^{-1} + C$, that requires a potentially unstable inversion of K. HMC v1 requires the computation of the inverse of the mass matrix and its factorization each time a new value of $\boldsymbol{\theta}$ is proposed. In HMC v2, instead, no extra operations in $O(n^3)$ are required given that K is already factorized, thus making it computationally very convenient.

Algorithm 4 HMC transition operator when $M^{-1} = L_{M^{-1}}L_{M^{-1}}^{\mathrm{T}}$

```
1: $\mathbf{f}_{(0)} = \mathbf{f}$; $\mathbf{p}_{(0)} \sim \mathcal{N}(\mathbf{p}_{(0)}|\mathbf{0}, M)$          $\triangleright$ $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$; $\mathbf{p}_{(0)} = \mathtt{bcksub}(L_{M^{-1}}^{\mathrm{T}}, \mathbf{z})$
2: $\lambda = \mathrm{sample}[1, \ldots, \lambda_{\max}]$
3: for ($t = 0$ to $\lambda - 1$) do
4:     $\mathbf{p}_{(t+1/2)} = \mathbf{p}_{(t)} + \frac{\varepsilon}{2}\nabla_{\mathbf{f}} W(\mathbf{f}_{(t)})$
5:     $\mathbf{f}_{(t+1)} = \mathbf{f}_{(t)} + \varepsilon M^{-1}\mathbf{p}_{(t+1/2)}$          $\triangleright$ $M^{-1}\mathbf{p} = L_{M^{-1}}(L_{M^{-1}}^{\mathrm{T}}\mathbf{p})$
6:     $\mathbf{p}_{(t+1)} = \mathbf{p}_{(1/2)} + \frac{\varepsilon}{2}\nabla_{\mathbf{f}} W(\mathbf{f}_{(t+1)})$
7: end for
8: $r = \min\left\{0, H(\mathbf{f}_{(0)}, \mathbf{p}_{(0)}) - H(\mathbf{f}_{(\lambda)}, \mathbf{p}_{(\lambda)})\right\}$          $\triangleright$ $\log|M| = -2\sum_i \log(L_{M^{-1}})_{ii}$
                                                   $\triangleright$ $\mathbf{p}^{\mathrm{T}}M^{-1}\mathbf{p} = \|L_{M^{-1}}^{\mathrm{T}}\mathbf{p}\|^2$
9: $u \sim \mathrm{Exp}(u|1)$
10: if ($r > -u$) then return $\mathbf{f}_{(\lambda)}$
11: else return $\mathbf{f}_{(0)}$
```

4 Results on simulated data

In this section, we first report a study on the efficiency and speed of convergence of different transition operators in sampling from posterior distribution of individual groups of variables in the SA and AA parameterization. Secondly, we report the same analysis to compare different parameterizations to obtain samples from the joint posterior distribution of $\mathbf{f}$ and $\boldsymbol{\theta}$.

4.1 Experimental setup

We simulated data from the four GP models considered in this work, namely: `LRG`, `LCX`, `VLT`, and `ORD`. We generated 10 data sets simulating from each of the four models for all combinations of $n = 100, 400$, and $d = 2, 10$, for a total of 160 distinct data sets. In order to isolate the effect of different likelihood functions in the results, we seeded the generation of the input data matrix X, hyper-parameters, and latent variables so that these were the same across different models. Covariates were generated uniformly in the unit hyper-cube, and the parameters used to generate latent variables were $\sigma = \exp(2)$, $\psi_{\tau_i} \sim U[-3, -1]$. We imposed Gamma priors on the length-scale parameters with shape a and rate b, $p(\tau_i) = \mathrm{Gam}(\tau_i | a = 1, b = 1)$. We imposed an inverse Gamma prior $p(\sigma) = \mathrm{invGam}(\sigma | a = 1, b = 1)$, where a and b are shape and scale parameters respectively on σ to exploit conjugacy in the SA parameterization.

In all the experiments we collected 20000 samples after a burn-in phase of 5000 iterations; during the burn-in we also had an adaptive phase to allow the samplers reach recommended acceptance rates (for example around 25% for MH). The transition operators for $\mathbf{f}$ had the following tuning parameters: α for MH v1 and MH v2, and ε for SMMALA and the variants of HMC which used a maximum of 10 leapfrog steps. The transition operators for $\boldsymbol{\theta}$ employed the following proposals: MH used a covariance $\Sigma = \alpha I$, HMC used a mass matrix $M = \alpha I$ and 10 maximum leapfrog steps, and SMMALA used a step-size ε. Convergence analysis was performed using the $\hat{R}$ potential scale reduction factor (Gelman and Rubin, 1992), which is a classic score used to assess convergence of MCMC algorithms. The computation of the $\hat{R}$ value is based on the within and between chain variances; a value close to one indicates that convergence is reached. The $\hat{R}$ value was computed based on 10 chains initialized from the prior to study what efficiency can be achieved without running preliminary simulations; this is different from the initialization procedure suggested in Gelman and Rubin (1992) that requires locating the modes of the target density. Due to the fairly diffuse priors on the length-scale parameters, we noticed difficulty in achieving convergence in some cases; we therefore initialized ψ_{τ_i} randomly in the interval $[-3, -1]$. The value of $\hat{R}$ was checked at 1000, 2000, 5000, 10000, 20000 iterations. We use the following procedure to compactly visualize the speed of convergence; we threshold the median value of $\hat{R}$ across 10 data sets at each checkpoint and use the following visual coding to report speed of convergence: ▯ $< 1.1 <$ ▃ $< 1.3 <$ ▆ $< 2 <$ ▮, so that ▯ indicates that $\hat{R} < 1.1$, ▃ indicates that $1.1 < \hat{R} < 1.3$, and so on. We then stack the rectangles associated to each checkpoint where we computed the value of $\hat{R}$, thus producing a sort of histogram of the median of $\hat{R}$ over the iterations. Efficiency of MCMC methods is compared based on the minimum of the Effective Sample Size (ESS) (Robert and Casella, 2005) computed across all the sampled variables. We then report its mean and standard deviation across the 10 chains and the 10 different data sets for each combination of size of the data set, dimensionality, and type of likelihood.

Table 1 Breakdown of the number of operations in $O(n^3)$ required to apply the transition operators considered in this work. #M, #I and #C represent number of multiplication of $n \times n$ matrices, inversions of $n \times n$ matrices, and number of Cholesky decompositions respectively. Counts are reported as functions of the number of iterations T and number of covariates d. In HMC, $\bar{\lambda}$ denotes the average number of leapfrog steps in one iteration.

	$\mathbf{f}\mid\boldsymbol{\theta}, \mathbf{y}$			$\boldsymbol{\theta}\mid\mathbf{f}$			$\boldsymbol{\theta}\mid\boldsymbol{\nu}, \mathbf{y}$		
	#M	#I	#C	#M	#I	#C	#M	#I	#C
MH	0	0	1	0	0	T	0	0	T
HMC	0	0	1	0	$T\bar{\lambda}$	T	0	0	$T + Td\bar{\lambda}$
SMMALA	0	1	T	Td	T	T	0	0	$T + Td$
ELL-SS	0	0	1	–	–	–	–	–	–
MH v1	0	0	1	–	–	–	–	–	–
MH v2	0	0	1	–	–	–	–	–	–
HMC v1	0	1	2	–	–	–	–	–	–
HMC v2	0	0	1	–	–	–	–	–	–

We are also interested in statistically assessing which methods achieve faster convergence. In order to do so, we perform pairwise Mann-Whitney tests with significance level of 0.05 comparing the value of $\hat{R}$ at the last checkpoint for all the chains across 10 data sets. This allows us to obtain an ordering of methods in terms of convergence speed. In each table we include a row at the bottom reporting the result of such a test. We denote by $1|2$ situations where the method in row 1 of the corresponding table converges significantly faster than the method in row 2. Instead, the notation $1, 2$ is used when the method in row 1 does not converge significantly faster than the method in row 2.

As a measure of complexity, we counted the number of operations with complexity in $O(n^3)$, namely number of Cholesky factorizations of $n \times n$ matrices (#C), number of inversions of $n \times n$ matrices (#I)[2], and number of multiplications of $n \times n$ matrices (#M). We believe that this is a more reliable measure of complexity with respect to running time, as running time can be affected by several implementation details and other factors that are not directly related to the actual complexity of the algorithms.

4.2 Assessing the efficiency of samplers for individual groups of variables

In this section, we present an assessment of the efficiency of different transition operators for each group of variables using both SA and AA parameterizations. Computational complexity for all the operators considered in the next sections is summarized in Tab. 1, where T represents the number of iterations, d the number of covariates and $\bar{\lambda}$ the average number of leapfrog steps in HMC transition operators. In the following sub-sections we present results about the sampling of the latent variables and hyper-parameters separately.

Table 2 Comparison of transition operators to sample $\mathbf{f}|\mathbf{y},\boldsymbol{\theta}$ for data generated from models with four different likelihoods. Minimum ESS is averaged over 10 chains for 10 different data sets for each value of n and d. The last row in each sub-table reports the result of the statistical test to assess which operators achieve significantly faster convergence.

LRG

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH v1	67 (15)	▯	47 (3)	▯	22 (7)	◧	8 (1)	◧
MH v2	204 (35)	▯	151 (7)	▯	67 (17)	▯	30 (2)	◧
SMMALA	756 (284)	▯	262 (30)	▯	457 (212)	▯	48 (5)	▯
ELL-SS	321 (61)	▯	241 (11)	▯	104 (25)	▯	50 (2)	▯
HMC v1	3395 (400)	▯	5163 (268)	▯	1352 (380)	▯	2962 (155)	▯
HMC v2	4004 (577)	▯	5225 (224)	▯	1566 (342)	▯	2995 (129)	▯
	$6\|5\|3\|4\|2\|1$		$6\|5\|3,4\|2\|1$		$6\|5\|3\|4\|2\|1$		$6\|5\|4\|3\|2\|1$	

LCX

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH v1	18 (16)	◧	6 (2)	◧	6 (5)	◧	1 (0)	▮
MH v2	23 (24)	◧	6 (2)	◧	8 (7)	◧	1 (0)	◧
SMMALA	217 (155)	▯	39 (4)	◧	258 (177)	◧	7 (1)	◧
ELL-SS	39 (42)	◧	11 (4)	◧	11 (11)	◧	2 (0)	◧
HMC v1	372 (277)	▯	188 (123)	▯	199 (200)	▯	81 (30)	▯
HMC v2	254 (197)	▯	188 (125)	▯	64 (37)	◧	80 (30)	▯
	$6\|5\|3\|4\|2\|1$		$6\|5\|3\|4\|1,2$		$5\|3,6\|4\|2\|1$		$6\|5\|3\|4\|2\|1$	

VLT

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH v1	28 (13)	◧	10 (2)	◧	15 (8)	◧	2 (0)	◧
MH v2	31 (16)	◧	12 (2)	◧	16 (8)	◧	2 (0)	◧
SMMALA	424 (216)	◧	117 (13)	◧	418 (127)	◧	61 (7)	◧
ELL-SS	46 (20)	◧	18 (4)	◧	22 (10)	◧	4 (1)	◧
HMC v1	1494 (667)	◧	449 (42)	◧	1384 (392)	◧	249 (25)	◧
HMC v2	418 (68)	◧	443 (39)	◧	183 (31)	◧	245 (25)	◧
	$4\|3\|2\|1,5,6$		$3\|4\|2\|1,5,6$		$3,4\|2\|1,5,6$		$3\|6\|4,5\|2\|1$	

ORD

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH v1	14 (8)	◧	6 (2)	◧	7 (5)	◧	1 (0)	▮
MH v2	14 (9)	◧	7 (2)	◧	7 (5)	◧	2 (0)	◧
SMMALA	48 (89)	◧	2 (0)	◧	107 (156)	◧	1 (0)	▮
ELL-SS	21 (11)	◧	10 (2)	◧	9 (5)	◧	2 (0)	◧
HMC v1	539 (650)	▯	472 (39)	▯	176 (200)	◧	257 (23)	▯
HMC v2	175 (54)	▯	483 (37)	▯	61 (22)	▯	255 (24)	▯
	$6\|5\|3\|4\|1,2$		$6\|5\|4\|2\|1\|3$		$6\|5\|3\|4\|1,2$		$6\|5\|2,4\|1,3$	

4.2.1 Sampling $\mathbf{f}|\mathbf{y},\boldsymbol{\theta}$

In this section we focus on the sampling from the posterior distribution of the latent variables $\mathbf{f}$. The results can be found in Tab. 2, and they were obtained

[2] This is a shorthand notation to denote a back and forward substitution of the identity matrix using Cholesky factors.

Table 3 Comparison of transition operators to sample $\boldsymbol{\theta}|\mathbf{f}$. Minimum ESS is averaged over 10 chains for 10 different data sets for each value of n and d. The last row reports the result of the statistical test to assess which operators achieve significantly faster convergence.

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH	2024 (144)	▯	156 (37)	▯	2124 (125)	▯	77 (33)	▯
HMC	11325 (915)	▂	830 (269)	▯	12556 (661)	▄	293 (137)	▯
SMMALA	9592 (2052)	▆	61 (23)	▯	10241 (2672)	▆	47 (17)	▂
	$1\|2,3$		$2\|1\|3$		$1\|2\|3$		$2\|1\|3$	

by fixing $\boldsymbol{\theta}$ to the values used to generate the data. We notice that different likelihood functions heavily affect efficiency and convergence speed; in the examples considered here, the results show that in `LRG` it is possible to achieve efficiency one order of magnitude higher than in other models. The scaled versions of MH work well in the case of `LRG` (MH v1 is slightly better than MH v2), but do not offer guarantees of convergence on other models. ELL-SS achieves better efficiency and convergence than the scaled versions of MH. SMMALA, which uses gradient and curvature information, achieves good efficiency and faster convergence than MH v1, MH v2, and ELL-SS, but at the cost of one operation in $O(n^3)$ at each iteration, as the metric tensor is a function of $\mathbf{f}$ and needs to be factorized at each iteration. Overall, the results suggest that the scaled versions of HMC are the best sampling methods for $\mathbf{f}|\boldsymbol{\theta}, \mathbf{y}$. HMC v1 is slightly better than HMC v2, but it requires one extra inversion and one extra Cholesky decomposition compared to HMC v2 that does not require any operations in $O(n^3)$ once the covariance matrix of the GP is factorized.

4.2.2 SA parameterization - Sampling $\boldsymbol{\theta}|\mathbf{f}$

In this section we present results about the sampling of hyper-parameters from the posterior distribution $\boldsymbol{\theta}|\mathbf{f}, \mathbf{y}$ which, given the hierarchical structure of the model, is simply $\boldsymbol{\theta}|\mathbf{f}$ independent from the data model. As reported in Tab. 1, the complexity of applying SMMALA and HMC is quite high compared to MH. MH requires one Cholesky factorization of Q at each iteration. In HMC, at each leapfrog step, the gradients of Q with respect to $\boldsymbol{\theta}$ are needed and the cheapest way to do this is by inverting Q first and noticing that all the remaining operations are in $O(n^2)$; this is done $\bar{\lambda}$ times on average at every iteration of HMC. Similarly, in SMMALA the gradient can be computed by inverting Q first; by doing so, the metric tensor can then be computed by d multiplications with the derivatives of Q and no other $O(n^3)$ operations.

The results are reported in Tab. 3, and were obtained by fixing $\mathbf{f}$ to the value used to generate the data and sampling only the length-scale parameters, as σ can be efficiently sampled using exact Gibbs steps. HMC improves quite substantially on efficiency, but not on speed of convergence; it may be worth employing some rescaling of the hyper-parameters to improve on this as suggested by Neal (1996). The performance of SMMALA is highly variable in efficiency and it converges more slowly than MH and HMC. This might be due to the skewness of the target

Table 4 Comparison of transition operators to sample $\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\nu}$ for data generated from models with four different likelihoods. Minimum ESS is computed as the average over 10 chains for 10 different data sets for each value of n and d. The last row in each sub-table reports the result of the statistical test to assess which operators achieve significantly faster convergence.

`LRG`

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH	556 (201)	▥	131 (33)	▥	512 (177)	▥	56 (11)	▥
HMC	2572 (1382)	▥	859 (278)	▥	2666 (973)	▥	223 (39)	▩
SMMALA	3833 (2032)	▥	65 (42)	▥	6877 (1584)	▥	47 (21)	▩
	$1,3\|2$		$2\|1\|3$		$1\|3\|2$		$1\|2,3$	

`LCX`

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH	818 (386)	▥	6 (4)	▩	1030 (397)	▩	3 (1)	▩
HMC	5169 (3297)	▩	11 (8)	▩	7145 (3852)	▩	4 (3)	▩
SMMALA	6158 (2788)	▥	9 (6)	■	8377 (1815)	▩	6 (4)	■
	$3\|1,2$		$1,2\|3$		$1,2,3$		$1,2\|3$	

`VLT`

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH	859 (318)	▩	22 (6)	▩	795 (270)	■	8 (6)	▩
HMC	5680 (2634)	▩	48 (20)	▩	5233 (2482)	■	11 (11)	■
SMMALA	6274 (1896)	▩	14 (9)	▩	6950 (2763)	■	11 (9)	■
	$1,2,3$		$1\|2\|3$		$1,2,3$		$1\|2\|3$	

`ORD`

	$n = 100$				$n = 400$			
	$d = 2$		$d = 10$		$d = 2$		$d = 10$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
MH	689 (159)	▥	14 (7)	▩	552 (168)	▩	9 (6)	■
HMC	155 (296)	■	14 (11)	■	79 (115)	■	4 (4)	■
SMMALA	3356 (1661)	▥	11 (8)	▩	2328 (1423)	▩	19 (27)	▩
	$1,3\|2$		$1,3\|2$		$1,3\|2$		$1,3\|2$	

distribution, that is known to affect the efficiency of SMMALA (Stathopoulos and Filippone, 2011). The results indicate that MH strikes a good balance between efficiency and computational cost.

4.2.3 AA parameterization - Sampling $\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\nu}$

In this section we present the sampling of the hyper-parameters from the posterior distribution $\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\nu}$, where we fixed $\boldsymbol{\nu}$ to the values used to generate the data. The analysis of complexity shows that MH requires one Cholesky factorization at each iteration. In HMC, each leapfrog requires computing L and the gradient of L with respect to $\boldsymbol{\theta}$ and no other operations in $O(n^3)$; this can be computed using the differentiation of the Cholesky algorithm which requires d operations in $O(n^3)$ (Smith, 1995). Likewise, for SMMALA L and the d derivatives of L with respect to $\boldsymbol{\theta}$ are the only operations in $O(n^3)$ needed.

Table 5 Comparison of different strategies to sample $\mathbf{f}, \boldsymbol{\theta} | \mathbf{y}$ for data generated from a `LRG` model. The rightmost column reports the complexity of the different methods with respect to number of inversion and Cholesky decompositions. In KHR, $\bar{\kappa}$ represents the average number of iterations to run the Laplace Approximation.

	$n = 100$				$n = 400$					
	$d = 2$		$d = 10$		$d = 2$		$d = 10$			
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	#I	#C
AA	131(57)	▯▯▯▯▯	117(34)	▯▯▯▯▯	94(38)	▯▯▯▯▯	47(17)	▃▯▯▯▯	0	T
ASIS	138(63)	▯▯▯▯▯	168(49)	▯▯▯▯▯	98(39)	▯▯▯▯▯	60(25)	▯▯▯▯▯	0	$2T$
KHR	856(360)	▯▯▯▯▯	177(48)	▯▯▯▯▯	481(219)	▯▯▯▯▯	116(32)	▯▯▯▯▯	0	$\bar{\kappa}T + 2T$
SA	8(6)	▆▆▆▆▆	59(18)	▯▯▯▯▯	5(2)	█▆▆▆▆	14(6)	▂▂▂▂▂	0	T
SURR	173(95)	▯▯▯▯▯	90(32)	▯▯▯▯▯	157(51)	▯▯▯▯▯	35(15)	▃▃▃▃▃	T	$2T$
	$3 \| 5 \| 1, 2 \| 4$		$1, 2 \| 3, 4, 5$		$3 \| 5 \| 1, 2 \| 4$		$2, 3 \| 1 \| 4, 5$			

The results can be found in Tab. 4 and are again variable across different models. In general SMMALA and HMC do not seem to offer faster convergence with respect to the MH transition operator which is therefore competitive in terms of efficiency relative to computational cost.

4.3 Assessing the efficiency of different parameterizations

After analyzing the results in the previous section, we decided to combine the transition operators which achieved a good sampling efficiency with relatively low computational cost and ease of implementation. We decided that a good combination to be used in AA, SA, ASIS, and SURR could be as follows: sampling $\mathbf{f}$ using HMC v2 and $\boldsymbol{\theta}$ using MH; HMC v2 and MH where adapted during the burn-in phase and in HMC v2 we set the maximum number of leapfrog steps to 10. For the sake of brevity, we focus on the `LRG` model only; the results on efficiency and speed of convergence in sampling hyper-parameters are reported in Tab. 5.

It is striking to see how challenging it is to efficiently sample from the posterior distribution of latent variables and hyper-parameters. Sampling efficiency is generally low; this is consistent with our experience in other applications involving sampling in hierarchical models (Filippone et al., 2012). As expected, the SA parameterization is the worst among the ones we tested. The AA parameterization, ASIS, and SURR generally offer good guarantees of convergence within a few thousand iterations. SURR seems to be superior in efficiency, which is consistent with what reported in Murray and Adams (2010), but it requires more operations in $O(n^3)$ compared to AA and ASIS. ASIS slightly improves efficiency and speed of convergence with respect to the AA scheme but requires double the number of operations in $O(n^3)$. KHR seems effective in breaking the correlation between the two groups of variables, but it may require several iterations within the approximation used to sample $\mathbf{f}$. In the experiments considered here $\bar{\kappa}$ is around 8, so the best compromise between computations and efficiency seems to be given by the AA and ASIS parameterizations.

Table 6 Comparison of different strategies to sample $\mathbf{f}, \boldsymbol{\theta} | \mathbf{y}$ in four UCI data sets modeled using a `LRG` model.

	Pima $n = 768, d = 8$		Wisconsin $n = 683, d = 9$		SPECT $n = 80, d = 22$		Ionosphere $n = 351, d = 34$	
	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$
AA	34 (4)		42 (15)		99 (18)		12 (5)	
ASIS	35 (8)		47 (11)		215 (23)		24 (8)	
KHR	153 (14)		20 (10)		101 (16)		2 (2)	
SA	5 (2)		7 (3)		97 (12)		11 (7)	
SURR	76 (10)		25 (14)		84 (14)		9 (4)	

5 Results on real data

We repeated the comparison of different parameterizations on four UCI data sets (Asuncion and Newman, 2007), namely the Pima, Wisconsin, SPECT, and Ionosphere data sets, which we modeled using `LRG` models; the results are reported in Tab. 6. We used the same priors and experimental setup as in the previous sections, except that all features were transformed to have zero mean and unit standard deviation, and latent variables were sampled iterating five updates of HMC v2. Also, chains were initialized sampling from the prior. Again, the SA parameterization shows the poorest efficiency and convergence speed, and the AA parameterization improves on that. Combining the AA and SA parameterizations using ASIS slightly improves on the AA parameterization, although the improvement is not dramatic. The SURR method improves on the AA parameterization, which is consistent with what reported in Murray and Adams (2010). The results of KHR are highly variable across data sets; in cases where the approximation to sample latent variables is accurate, the chains mix well. In some cases, however, the approximation is not accurate enough to guarantee a good acceptance rate, and the chains can spend a long time in the same position before accepting the joint proposal.

6 Conclusions

In this paper we studied and compared a number of state-of-the-art strategies to carry out the fully Bayesian treatment of GP models. We focused on four GP models and performed an extensive evaluation of efficiency, convergence speed, and computational complexity of several transition operators and sampling strategies.

The results in this paper show that latent variables can be sampled quite efficiently with little computational effort once the GP covariance matrix is factorized. This can be achieved by a simple variant of HMC that we introduced in this paper. About sampling hyper-parameters in different parameterizations, the results presented here indicate that the gain in sampling efficiency given by the use of complicated proposal mechanisms does not scale as much as their computational cost. It would be interesting to investigate some recently proposed variants to slice sampling (Thompson and Neal, 2010) and Hybrid Monte Carlo (Hoffman and Gelman, 2012) on the sampling of hyper-parameters.

The analysis of the results obtained by different parameterization suggest that AA is a sensible and computationally cheap parameterization with good convergence properties. AA performs similarly to ASIS at half the computational cost. It makes sense, however, to employ ASIS when in doubt about the best parameterization to use, although GP models with full covariance matrices will generally fall into the weak data limit as the $O(n^2)$ space and $O(n^3)$ time complexities constrain the number of data that can be processed.

In general, the results show how challenging it is to efficiently sample from the posterior distribution of latent variables and hyper-parameters in GP models and motivates further research into methods to do this efficiently. Some sampling strategies, such as the one based on the AA parameterization, are capable of achieving convergence within a reasonable number of iterations, and this makes it possible to carry out the fully Bayesian treatment of GP models dealing with a small to moderate number of samples. We have recently demonstrated that this is indeed the case in Filippone et al. (2012), but more needs to be done in the direction of developing robust stochastic based inference methods for GP models.

It would be interesting to investigate how performance is affected by the choice of the design, which in the simulated data presented here was assumed uniform. Also, we studied in particular GP models with the squared exponential ARD covariance function. It would be interesting to compare the method considered here in models characterized by other covariance functions, such as the Matérn, or sparse inverse covariance functions as in Rue et al. (2009); the latter would make it possible to test the strong data limit case. Finally, in this study we have not included a mean function for the GP prior or extra parameters for the likelihood function. This would require including the sampling of other quantities that may further impact on efficiency and speed of convergence.

References

1. Amari, S. and H. Nagaoka (2000). *Methods of Information Geometry*, Volume 191 of *Translations of Mathematical monographs.* Oxford University Press.
2. Asuncion, A. and D. J. Newman (2007). UCI machine learning repository.
3. Christensen, O. F., G. O. Roberts, and J. S. Rosenthal (2005). Scaling limits for the transient phase of local MetropolisHastings algorithms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*(2), 253–268.
4. Chu, W. and Z. Ghahramani (2005). Gaussian Processes for Ordinal Regression. *Journal of Machine Learning Research 6*, 1019–1041.
5. Cseke, B. and T. Heskes (2011). Approximate Marginals in Latent Gaussian Models. *Journal of Machine Learning Research 12*, 417–454.
6. Filippone, M., A. F. Marquand, C. R. V. Blain, S. C. R. Williams, J. Mourão-Miranda, and M. Girolami (2012). Probabilistic Prediction of Neurological Disorders with a Statistical Assessment of Neuroimaging Data Modalities. *Annals of Applied Statistics 6*(4), 1883–1905.
7. Filippone, M., M. Zhong, and M. Girolami (2012). On the fully Bayesian treatment of latent Gaussian models using stochastic simulations. Technical Report TR-2012-329, School of Computing Science, University of Glasgow.
8. Flegal, J. M., M. Haran, and G. L. Jones (2007). Markov Chain Monte Carlo: Can We Trust the Third Significant Figure? *Statistical Science 23*(2), 250–260.
9. Gelman, A. and D. B. Rubin (1992). Inference from iterative simulation using multiple sequences. *Statistical Science 7*(4), 457–472.

10. Girolami, M. and B. Calderhead (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73*(2), 123–214.
11. Hoffman, M. D. and A. Gelman (2012). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, to appear.
12. Knorr-Held, L. and H. Rue (2002). On Block Updating in Markov Random Field Models for Disease Mapping. *Scandinavian Journal of Statistics 29*(4), 597–614.
13. Kuss, M. and C. E. Rasmussen (2005). Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research 6*, 1679–1704.
14. Mackay, D. J. C. (1994). Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten (Eds.), *Models of Neural Networks III*, Chapter 6, pp. 211–254. Springer.
15. Minka, T. P. (2001). Expectation Propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, San Francisco, CA, USA, pp. 362–369. Morgan Kaufmann Publishers Inc.
16. Møller, J., A. R. Syversveen, and R. P. Waagepetersen (1998). Log Gaussian Cox Processes. *Scandinavian Journal of Statistics 25*(3), 451–482.
17. Murray, I. and R. P. Adams (2010). Slice sampling covariance hyperparameters of latent Gaussian models. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (Eds.), *NIPS*, pp. 1732–1740. Curran Associates, Inc.
18. Murray, I., R. P. Adams, and D. J. C. MacKay (2010). Elliptical slice sampling. *Journal of Machine Learning Research - Proceedings Track 9*, 541–548.
19. Neal, R. (2003). Slice Sampling. *Annals of Statistics 31*, 705–767.
20. Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.
21. Neal, R. M. (1996). *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)* (1 ed.). Springer.
22. Neal, R. M. (1999). Regression and classification using Gaussian process priors (with discussion). *Bayesian Statistics 6*, 475–501.
23. Opper, M. and O. Winther (2000). Gaussian processes for classification: Mean-field algorithms. *Neural Computation 12*(11), 2655–2684.
24. Rasmussen, C. E. and C. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.
25. Robert, C. P. and G. Casella (2005). *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.
26. Roberts, G. O. and O. Stramer (2002). Langevin Diffusions and Metropolis-Hastings Algorithms. *Methodology and Computing in Applied Probability 4*(4), 337–357.
27. Rue, H., S. Martino, and N. Chopin (2009). 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.
28. Smith, S. P. (1995). Differentiation of the Cholesky Algorithm. *Journal of Computational and Graphical Statistics 4*(2), 134–147.

29. Stathopoulos, V. and M. Filippone (2011). Discussion of the paper "Riemann manifold Langevin and Hamiltonian Monte Carlo methods" by Mark Girolami and Ben Calderhead. *Journal of the Royal Statistical Society, Series B (Statistical Methodology) 73*(2), 167–168.
30. Thompson, M. and R. M. Neal (2010). Covariance-Adaptive Slice Sampling. Technical Report 1002, Department of Statistics, University of Toronto.
31. Tierney, L. and J. B. Kadane (1986). Accurate Approximations for Posterior Moments and Marginal Densities. *Journal of the American Statistical Association 81*(393), 82–86.
32. Vanhatalo, J. and A. Vehtari (2007). Sparse Log Gaussian Processes via MCMC for Spatial Epidemiology. *Journal of Machine Learning Research - Proceedings Track 1*, 73–89.
33. Wilson, A. G. and Z. Ghahramani (2010). Copula Processes. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (Eds.), *NIPS*, pp. 2460–2468. Curran Associates, Inc.
34. Yu, Y. and X.-L. Meng (2011). To Center or Not to Center: That Is Not the Question–An Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency. *Journal of Computational and Graphical Statistics 20*(3), 531–570.

A SA and AA parameterizations

A.1 Sufficient Augmentation (SA)

We derive here the quantities needed to apply the transition operators considered in this work in the SA parameterization. Let $\mathcal{L} = \log[p(\mathbf{y}|\mathbf{f})]$. The log-joint density is:

$$\log[p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta})] = \mathcal{L} - \frac{1}{2}\log(|Q|) - \frac{n}{2}\log(\sigma) - \frac{1}{2\sigma}\mathbf{f}^{\mathrm{T}}Q^{-1}\mathbf{f} + \log[p(\boldsymbol{\theta})] + \text{const.}$$

Note that σ could be marginalized out, but it would not be possible to get manageable expressions for the metric tensor with respect to $\boldsymbol{\tau}$; for $\mathbf{f}$, instead, this would be possible. We do not pursue this here, and we leave it for future investigation.

By inspecting the log-joint density, we see that we can obtain the conditional density for σ in the following form

$$\log[p(\sigma|\mathbf{y}, \mathbf{f}, \boldsymbol{\tau})] = -\frac{n}{2}\log(\sigma) - \frac{1}{2\sigma}\mathbf{f}^{\mathrm{T}}Q^{-1}\mathbf{f} + \text{const.}$$

which we recognize as an inverse Gamma. By placing an inverse Gamma prior on σ in the form $\text{invGa}(\sigma|a, b)$ with shape a and scale b, we can sample directly:

$$\sigma \sim \text{invGa}\left(\sigma \left| a + \frac{n}{2}, b + \frac{1}{2}\mathbf{f}^{\mathrm{T}}Q^{-1}\mathbf{f}\right.\right)$$

The gradients of the log-joint density needed to apply gradient based operators are:

$$\nabla_{\mathbf{f}} \log[p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta})] = \nabla_{\mathbf{f}}\mathcal{L} - \frac{1}{\sigma}Q^{-1}\mathbf{f}$$

$$\frac{\partial \log[p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta})]}{\partial \psi_{\tau_i}} = -\frac{1}{2}\mathrm{Tr}\left(Q^{-1}\frac{\partial Q}{\partial \psi_{\tau_i}}\right) + \frac{1}{2\sigma}\mathbf{f}^{\mathrm{T}}Q^{-1}\frac{\partial Q}{\partial \psi_{\tau_i}}Q^{-1}\mathbf{f} + \frac{\partial \log[p(\psi_{\boldsymbol{\tau}})]}{\partial \psi_{\tau_i}}$$

The FI for latent variables and parameters are:

$$R = \mathrm{FI}_{\mathbf{f},\mathbf{f}} = \mathrm{E}_{\mathbf{y}}\left[(\nabla_{\mathbf{f}}\mathcal{L})(\nabla_{\mathbf{f}}\mathcal{L})^{\mathrm{T}}\right] = -\mathrm{E}_{\mathbf{y}}\left[\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\mathcal{L}\right]$$

$$\mathrm{FI}_{\boldsymbol{\psi_\tau},\boldsymbol{\psi_\tau}} = \mathrm{E}_{\mathbf{f}} \left[(\nabla_{\boldsymbol{\psi_\tau}} \log[p(\mathbf{f}|\boldsymbol{\psi_\tau})])(\nabla_{\boldsymbol{\psi_\tau}} \log[p(\mathbf{f}|\boldsymbol{\psi_\tau})])^{\mathrm{T}} \right]$$

Given that the likelihood factorizes with respect to the observations, the Hessian of $\mathcal{L}$ with respect to $\mathbf{f}$ is diagonal, so $R = \mathrm{FI}_{\mathbf{f},\mathbf{f}}$ is diagonal as well. The metric tensors are the FI matrices plus the negative Hessian of the priors:

$$G_{\mathbf{f},\mathbf{f}} = R + \frac{1}{\sigma} Q^{-1}$$

$$G_{\psi_{\tau_i},\psi_{\tau_j}} = +\frac{1}{2} \mathrm{Tr} \left(Q^{-1} \frac{\partial Q}{\partial \psi_{\tau_j}} Q^{-1} \frac{\partial Q}{\partial \psi_{\tau_i}} \right) - \frac{\partial^2 \log[p(\boldsymbol{\psi_\tau})]}{\partial \psi_{\tau_i} \partial \psi_{\tau_j}}$$

A.2 Ancillary Augmentation (AA)

We derive here the quantities needed to apply the transition operators considered in this work in the AA parameterization. The expression of the log-joint density is the same as in the SA case, bearing in mind the transformation $\mathbf{f} = \sqrt{\sigma} L \boldsymbol{\nu}$; this yields:

$$\log[p(\mathbf{y}, \boldsymbol{\nu}, \boldsymbol{\theta})] = \mathcal{L}(\mathbf{y}|\boldsymbol{\nu}, \boldsymbol{\theta}) - \frac{1}{2} \boldsymbol{\nu}^{\mathrm{T}} \boldsymbol{\nu} + \log[p(\boldsymbol{\theta})] + \mathrm{const.}$$

The gradient with respect to the hyper-parameters can be computed by using the chain rule of derivation and standard properties of derivatives of vector valued functions:

$$\frac{\partial \log[p(\mathbf{y}, \boldsymbol{\nu}, \boldsymbol{\theta})]}{\partial \psi_{\tau_i}} = \sqrt{\sigma} (\nabla_{\mathbf{f}} \mathcal{L}(\mathbf{y}|\mathbf{f}))^{\mathrm{T}} \frac{\partial L}{\partial \theta_i} \boldsymbol{\nu} + \frac{\partial \log[p(\boldsymbol{\psi_\tau})]}{\partial \psi_{\tau_i}}$$

The FI matrix is readily obtained as:

$$\mathrm{FI}_{\theta_i,\theta_j} = \sigma \boldsymbol{\nu}^{\mathrm{T}} \frac{\partial L^{\mathrm{T}}}{\partial \theta_i} R \frac{\partial L}{\partial \theta_j} \boldsymbol{\nu}$$

With the contribution (negative Hessian) of the prior, the metric tensor used in the manifold methods results in:

$$G_{\theta_i,\theta_j} = \sigma \boldsymbol{\nu}^{\mathrm{T}} \frac{\partial L^{\mathrm{T}}}{\partial \theta_i} R \frac{\partial L}{\partial \theta_j} \boldsymbol{\nu} - \frac{\partial^2 \log[p(\boldsymbol{\theta})]}{\partial \theta_i \partial \theta_j}$$

B GP models considered in this paper

B.1 Logistic regression with GP priors (`LRG`)

Let:

$$l^+(f) = \mathrm{logistic}(f) = \frac{1}{1 + \exp(-f)} \qquad l^-(f) = 1 - l^+(f) = \mathrm{logistic}(-f)$$

In logistic regression, observations follow a Bernoulli distribution with success probability given by a sigmoid transformation of the associated latent variables:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i) = \prod_{i=1}^{n} \mathrm{Bern}(y_i|l^+(f_i)) = \prod_{i=1}^{n} l^+(f_i)^{y_i} l^-(f_i)^{(1-y_i)}$$

The gradient with respect to $\mathbf{f}$ results in:

$$(\nabla_{\mathbf{f}} \mathcal{L})_j = y_j - l^+(f_j)$$

The computation of diagonal elements of the FI matrix for $\mathbf{f}$ requires the expectations of y_i^2 which are the same as the expectations of y_i, that are l_i^+; this leads to $R_{ii} = l^+(f_i) l^-(f_i)$.

B.2 Log-Gaussian Cox model (LCX)

In this model, observations follow a Poisson distribution with mean computed as an exponentially transformed version of the corresponding latent variables:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i) = \prod_{i=1}^{n} \mathrm{Poisson}(y_i|\exp(f_i))$$

The gradient with respect to $\mathbf{f}$ and the diagonal elements of R result in:

$$(\nabla_{\mathbf{f}}\mathcal{L})_j = y_j - \exp(f_j) \qquad R_{ii} = \exp(f_i)$$

B.3 Stochastic Volatility model with GP priors (VLT)

In this model, observations follow a zero mean Gaussian distribution with standard deviation computed as an exponentially transformed version of the corresponding latent variable:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i) = \prod_{i=1}^{n} \mathcal{N}(y_i|0, \exp(f_i)^2)$$

The gradient with respect to $\mathbf{f}$ and the diagonal elements of R result in:

$$(\nabla_{\mathbf{f}}\mathcal{L})_j = \exp(f_i)^{-2} y_j^2 - 1 \qquad R_{ii} = 2$$

B.4 Ordinal Regression with GP priors (ORD)

In this model, latent variables are thresholded at r points that will be denoted by $b_0, \ldots, b_r$, with $b_0 = -\infty$ and $b_r = +\infty$. Then, y is the index of the interval where the corresponding latent variable f falls. The likelihood of an observed label y_i associated to the ith latent variable f_i is then:

$$\bar{p}(y_i|f_i) = 1 \qquad \text{if } b_{y_i-1} < f \leq b_{y_i}$$

and zero otherwise. This model is usually modified to allow for a noise term δ (distributed as $\mathcal{N}(\delta|0, \sigma_\delta^2)$) in the latent variables so that:

$$p(y_i|f_i) = \int \bar{p}(y_i|f_i + \delta)\mathcal{N}(\delta|0, \sigma_\delta^2)d\delta = \Phi(z_i^{(y_i)}) - \Phi(z_i^{(y_i-1)})$$

where:

$$z_i^{(s)} = \frac{b_s - f_i}{\sigma_\delta}$$

In particular:

$$\mathcal{L} = \sum_{i=1}^{n} \log\left[\Phi(z_i^{(y_i)}) - \Phi(z_i^{(y_i-1)})\right]$$

$$(\nabla_{\mathbf{f}}\mathcal{L})_i = \frac{1}{\sigma_\delta} \frac{\mathcal{N}(z_i^{(y_i-1)}|0,1) - \mathcal{N}(z_i^{(y_i)}|0,1)}{\Phi(z_i^{(y_i)}) - \Phi(z_i^{(y_i-1)})}$$

By writing the diagonal elements of Hessian of the log-likelihood computed for $y_i = s$

$$(\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\mathcal{L})_{ii}^{(s)} = \frac{1}{\sigma_\delta^2} \frac{z_i^{(s)}\mathcal{N}(z_i^{(s)}|0,1) - z_i^{(s-1)}\mathcal{N}(z_i^{(s-1)}|0,1)}{\Phi(z_i^{(s)}) - \Phi(z_i^{(s-1)})} - \frac{1}{\sigma_\delta^2}\left(\frac{\mathcal{N}(z_i^{(s-1)}|0,1) - \mathcal{N}(z_i^{(s)}|0,1)}{\Phi(z_i^{(s)}) - \Phi(z_i^{(s-1)})}\right)^2$$

it is possible to compute the expectation of the negative Hessian as:

$$R_{ii} = -\sum_{s=1}^{r} (\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\mathcal{L})_{ii}^{(s)}\, p(s|f_i) = \sum_{s=1}^{r} (\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\mathcal{L})_{ii}^{(s)} \left[\Phi(z_i^{(s-1)}) - \Phi(z_i^{(s)})\right]$$

Note that the formulation in this paper is slightly different from the one in Chu and Ghahramani (2005), where σ is dropped and thresholds are inferred instead.

PROBABILISTIC PREDICTION OF NEUROLOGICAL DISORDERS WITH A STATISTICAL ASSESSMENT OF NEUROIMAGING DATA MODALITIES*

By M. Filippone, A.F. Marquand C.R.V. Blain S.C.R. Williams J. Mourão-Miranda and M. Girolami

*Abstract*For many neurological disorders, prediction of disease state is an important clinical aim. Neuroimaging provides detailed information about brain structure and function from which such predictions may be statistically derived. A multinomial logit model with Gaussian process priors is proposed to: (i) predict disease state based on whole-brain neuroimaging data and (ii) analyze the relative informativeness of different image modalities and brain regions. Advanced Markov chain Monte Carlo methods are employed to perform posterior inference over the model. This paper reports a statistical assessment of multiple neuroimaging modalities applied to the discrimination of three Parkinsonian neurological disorders from one another and healthy controls, showing promising predictive performance of disease states when compared to non probabilistic classifiers based on multiple modalities. The statistical analysis also quantifies the relative importance of different neuroimaging measures and brain regions in discriminating between these diseases and suggests that for prediction there is little benefit in acquiring multiple neuroimaging sequences. Finally, the predictive capability of different brain regions is found to be in accordance with the regional pathology of the diseases as reported in the clinical literature.

1. Introduction.

For many neurological and psychiatric disorders, making a definitive diagnosis and predicting clinical outcome are complex and difficult problems. Difficulties arise due to many factors including overlapping symptom profiles, comorbidities in clinical populations and individual variation in disease phenotype or disease course. In addition, for many neurological disorders the diagnosis can only be confirmed via analysis of brain tissue post-mortem. Thus, technological advances that improve the efficiency or accuracy of clinical assessments hold the potential to

*MG gratefully acknowledges support from the EPSRC grants EP/E052029/1 and EP/H024875/1. AM gratefully acknowledges support from the KCL Centre of Excellence in Medical Engineering, funded by the Wellcome Trust and EPSRC under grant no. WT088641/Z/09/Z and JMM gratefully acknowledges support from the Wellcome Trust under grant no. WT086565/Z/08/Z.

Keywords and phrases: multi-modality multinomial logit model, Gaussian process, hierarchical model, high dimensional data, Markov chain Monte Carlo, Parkinsonian diseases, prediction of disease state

improve mainstream clinical practice and to provide more cost-effective and personalized approaches to treatment.

In this regard, combining data obtained from neuroimaging measures with statistical discriminant analysis has recently attracted substantial interest amongst the neuroimaging community (e.g., Klöppel et al. (2008); Marquand et al. (2008)). Neuroimaging data present particular statistical challenges in that they are often extremely high dimensional (in the order of hundreds of thousands to millions of variates) with very few samples (tens to hundreds). Further, multiple imaging sequences may be acquired for each participant, each aiming to measure different properties of brain tissue. Each sequence may in turn give rise to several different measurements. In the present work, we will use the term 'modality' to describe such a set of measurements. In response to those challenges, most attempts to predict disease state from neuroimaging data employ the Support Vector Machine (SVM; see e.g. Schölkopf and Smola (2001)) based on information obtained from a single modality. Such an approach however is not able, in any statistical manner, to fully address questions related to the importance of different modalities. As we will show in the experimental section of this paper, even the multi-modality SVM-based classifier proposed in Rakotomamonjy et al. (2008) lacks a systematic way of characterizing the uncertainty in the predictions and in the assessment of the relative importance of different modalities.

In this work, we adopt a multinomial logit model based on Gaussian process (GP) priors (Williams and Barber, 1998) as a probabilistic prediction method that provides the means to incorporate measures from different imaging modalities. We apply this approach to discriminate between three relatively common neurological disorders of the motor system based on data modalities derived from three distinct neuroimaging sequences. In this application we aim to characterize uncertainty without resorting to potentially inaccurate deterministic approximations to the integrals involved in the inference process. Therefore, we propose to employ Markov chain Monte Carlo (MCMC) methods to estimate the analytically intractable integrals as they provide guarantees of asymptotic convergence to the correct results. The particular structure of the model and the large number of variables involved, however, make the use of MCMC techniques seriously challenging (Filippone, Zhong and Girolami, 2012; Murray and Adams, 2010). In this work, we make use of reparametrization techniques (Yu and Meng, 2011) and state-of-the-art sampling methods based on the geometry of the underlying statistical model (Girolami and Calderhead, 2011) to achieve efficient sampling.

The remainder of the paper is structured as follows: in section 2, we describe the motivating application of statistical discrimination of movement disorders from brain images. In section 3 we introduce the multinomial logit model with GP priors and in section 4 we present the associated MCMC methodology. In section 5 we report a comparison of MCMC strategies applied to our brain imaging data and in section 6 we investigate the predictive ability of different data sources and brain regions, comparing the results with a non-probabilistic multi-modality classifier based on SVMs. Section 7 shows how predictive probabilities can be used to refine predictions, and section 8 draws conclusions commenting on the questions that this methodology addresses in this particular application.

2. Discriminating among Parkinsonian Disorders.

2.1. *Introduction to the disorders.* For this application, we aim to discriminate between healthy control subjects (HCs) and subjects with either multiple system atrophy (MSA), progressive supranuclear palsy (PSP) or idiopathic Parkinson's disease (IPD), which are behaviorally diagnosed motor conditions collectively referred to as 'Parkinsonian disorders'. MSA, PSP and IPD can be difficult to distinguish clinically in the early stages (Litvan et al., 2003) and carry a high rate of misdiagnosis, even though early diagnosis is important in predicting clinical outcome and formulating a treatment strategy (Seppi, 2007). For example, MSA and PSP have a much more rapid disease progression relative to IPD and carry a shorter life expectancy after diagnosis. Further, IPD responds relatively well to pharmacotherapy, while MSA and PSP are both associated with a modest to poor response. Thus, automated diagnostic tools to discriminate between the disorders is of clinical relevance where they may help to reduce the rate of misdiagnosis and ultimately lead to more favourable outcomes for patients.

2.2. *The clinical problem of discriminating among Parkinsonian Disorders.* In this study, we employed magnetic resonance imaging (MRI) as it is non-invasive, widely available and, unlike alternative measures such as positron emission tomography, does not involve exposing subjects to ionizing radiation. A detailed discussion of the imaging modalities employed in this study is beyond the scope of the present work but see Farrall (2006) for an overview. Briefly, the different imaging modalities employed here measure different properties of brain tissue: T1-weighted imaging is well-suited to visualizing anatomical structure, while T2-weighted structural imaging often shows focal tissue abnormalities more clearly. Diffusion tensor imaging (DTI) does not measure brain structure directly, but instead measures the diffusion of water molecules along fibre tracts in the brain, thus quantifying the integrity of the fibre bundles that connect different brain regions (see Basser and Jones (2002) for an introduction to DTI).

A review of the neuropathology of the Parkinsonian disorders is also beyond the scope of this work but briefly we note that MSA and PSP are characterized by distinct cellular pathologies and subsequent degeneration of widespread and partially overlapping brain regions. For MSA, affected regions include the brainstem, basal ganglia (e.g. caudate and putamen), cerebellum and cerebral cortex (Wenning et al., 1997). Note that MSA is sometimes subdivided into Parkinsonian and cerebellar subtypes (MSA-P and MSA-C respectively) but for the present work we included both variants in the same class. For PSP the brainstem and basal ganglia both undergo severe degeneration (Hauw et al., 1994) although cortical areas are also affected. In contrast, IPD is characterized in the early stages by relatively focal pathology in the substantia nigra (a pair of small nuclei in the brainstem), which is difficult to detect using conventional structural MRI where the scans of IPD patients can appear effectively normal (Seppi, 2007).

There is however, some evidence that changes in IPD can be detected using DTI (see, e.g., Yoshikawa et al. (2004)). Thus, it is of interest to investigate which data modalities are best suited to discriminating between MSA, PSP, IPD and HCs, which has practical implications in planning future diagnostic studies: MRI scans are costly, so it is desirable to know which data modalities provide the best discrimination of the diagnostic groups and which scans can be omitted from a scanning protocol to avoid wasting money acquiring scans that do not provide additional predictive value.

2.3. *State of the art in diagnosis and prediction.* We are aware of only one existing study that employed a discriminant approach to diagnose these diseases based on whole-brain neuroimaging measures (Focke et al., 2011). This study employed binary SVM classifiers to discriminate MSA-P from IPD, PSP and HCs based on a similar sample to the present study. The authors reported that (i) PSP could be accurately discriminated from IPD, (ii) that separation of the MSA-P group from IPS and from controls was only marginally better than chance and (iii) that no separation of the IPD group from HCs was possible. The authors did not attempt to combine the distinct binary classifiers to provide multi-class predictions.

The problem of combining different modalities in classification models can be seen as a Multiple Kernel Learning (MKL) problem (Lanckriet et al., 2004; Sonnenburg et al., 2006). A recent MKL approach to classification referred to as 'simpleMKL' has been proposed by Rakotomamonjy et al. (2008). SimpleMKL is based on a SVM learning algorithm and shows good performance relative to other MKL approaches; for this reason we will consider it as a baseline against which we aim to compare the performance of the proposed approach.

2.4. *Data acquisition and preprocessing.* Eighteen subjects with MSA, 16 subjects with PSP, 14 subjects with IPD (all in mid disease stage) and 14 HCs were recruited according to clinical and experimental criteria described in Blain et al. (2006). For each subject, a T2-weighted structural image, a T1-weighted spoiled gradient recalled (SPGR) structural image and a DTI sequence were acquired and preprocessed (see the appendix for the details on acquisition.)

All images were screened by a trained radiologist and were examined for gross

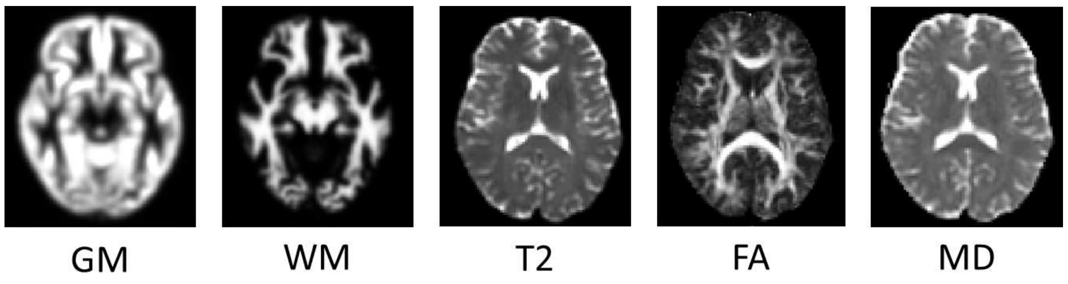

FIGURE 1. *Examples of each data source (after preprocessing), taken from the same slice and subject*

structural abnormalities, including white matter abnormalities. Diffusion tensor images were then preprocessed according to an in-house protocol and were summarized by measures of fractional anisotropy (FA) and mean diffusivity (MD) at every brain location (see Basser and Jones (2002)). SPGR images were preprocessed using the DARTEL toolbox included in the SPM software package (www.fil.ion.ucl.ac.uk/spm), which involved non-linear registration to a common reference space, segmentation into grey matter (GM), white matter (WM) and cerebrospinal fluid (CSF) in addition to smoothing with a 6mm isotropic Gaussian kernel.

For this analysis, whole-brain (unmodulated) GM and WM images derived from the SPGR scans, the T2 structural images plus the FA and MD images derived from the DTI sequence were used for classification, yielding a total of five distinct modalities for each subject. For illustrative purposes, an example of each type of image after preprocessing is provided in figure 1.

3. Multinomial logit model with GP priors. The aim of this study is twofold: the first is to reliably estimate the probability that new subjects have MSA, PSP, IPD, or none of them and the second is to assess the importance of different sources of information in the discrimination among these diseases. We cast this problem as a multi-modality classification problem, whereby we associate class labels corresponding to MSA, PSP, IPD, and HCs to n subjects described by $s = 1, \dots, q$ distinct representations (i.e. modalities), each defined by d_s covariates.

Denote each modality by an $n \times d_s$ matrix X_s. Let $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ be the set of observed labels for the n subjects. Assume a 1-of-m coding scheme ($m = 4$ in our application), whereby the membership of subject i to class c is represented by a vector $\mathbf{y}_i$ of length m where $(\mathbf{y}_i)_r = 1$ if $r = c$ and zero otherwise.

In this work we propose a probabilistic multinomial logit classification model based on Gaussian process (GP) priors to model the probability $\pi_{ic} := p(y_{ic} = 1|X_1, \dots, X_q)$ that subject i belongs to class c. The multinomial logit model assumes that the class labels $\mathbf{y}_i$ are conditionally independent given a set of m latent functions

$\mathbf{f}_c$ that model the π_{ic} using the following transformation:

$$\pi_{ic} = \frac{[\exp(\mathbf{f}_c)]_i}{[\sum_{r=1}^{m} \exp(\mathbf{f}_r)]_i} \; . \qquad (1)$$

We assume that the latent variables $\mathbf{f}_c$ are independent across classes and drawn from GPs, so that $\mathbf{f}_c \sim \mathcal{N}(\mathbf{0}, K_c)$. The assumption of independence between variables belonging to different classes can be relaxed in cases where there is prior knowledge about that. Note that the assumption of conditional independence of the class labels given the latent variables is not restrictive as the prior over the latent variables imposes a covariance structure that is reflected on the class labels.

In order to assess the importance of each modality in the classification task, we propose to model each covariance K_c as a linear combination of covariances obtained from the q modalities, say C_s with $s = 1, \ldots, q$. We constrain the linear combination of covariance functions to be positive definite by modeling $K_c = \sum_{s=1}^{q} \exp[\theta_{cs}] C_s$. Note that given the additivity of the linear model under the GP it is possible to interpret this model as one where each latent function is a linear combination of basis functions with covariances C_s. Since the data modalities employed in this study potentially have different numbers of features which are are also scaled differently, we employed two simple operations to normalize the images prior to classification. First we divided each feature vector by its Euclidean norm then standardised each feature to have zero mean and unit variance across all scans. We then chose a covariance for each modality to be $C_s = X_s X_s^T$. Given that the modalities are normalized and that the covariances are linear in the data sources X_s, the inference of the corresponding weights allows to draw conclusions on their relative importance in the classification task. In this work we imposed Gamma priors on the weights $\exp[\theta_{cs}]$, but for the sake of completeness and to rule out any dependencies of the results from the parametrization of the weights and the specification of the prior, we have also explored the possibility to use a Dirichlet prior inferring the concentration parameter; we will discuss this in more detail in the sections reporting the results.

We note here that the representation in eq. 1 is redundant as class probabilities are defined up to a scaling factor of the exponential of the latent variables. Choosing a model in which $m - 1$ latent functions are modeled and one is fixed would remove any redundancy, but, as described in Neal (1999) “forcing the latent values for one of the classes to be zero would introduce an arbitrary asymmetry into the prior”. Also, modeling $m - 1$ latent functions would not allow a direct interpretation of the importance of different modalities given by the hyper-parameters.

We used all features to perform the classification because in our experience feature selection does not provide a benefit in terms of increasing the accuracy of classification models for neuroimaging data but does increase their complexity. In line with this, a recent comparative analysis of alternative data preprocessing methods on a publicly available neuroimaging dataset indicated that feature selection did not improve classification performance but did substantially increase the computation time (Cuingnet et al., 2011).

We now discuss how to make inference for the proposed model. In order to keep the notation uncluttered, we will drop the explicit conditioning on X_s. We will denote by $\mathbf{f}$ the $(nc) \times 1$ vector obtained by concatenating the class specific latent functions $\mathbf{f}_c$, and similarly by $\mathbf{y}$ and $\boldsymbol{\pi}$ the vectors obtained by concatenating the vectors $\mathbf{y}_{\cdot c}$ and $\pi_{\cdot c}$. Finally, let the $m \times q$ matrix $\boldsymbol{\theta}$ denote the set of hyper-parameters, and K be the matrix obtained by block-concatenating the covariance matrices K_c.

Given the likelihood for the observed labels, the prior over the latent functions, and the prior over the hyper-parameters, we can write the log-joint density as

$$\mathcal{L} = \log[p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta})] = \log[p(\mathbf{y}|\mathbf{f})] + \log[p(\mathbf{f}|\boldsymbol{\theta})] + \log[p(\boldsymbol{\theta})] \ .$$

One of the goals of our analysis is to obtain predictive distributions for new subjects. Let $\mathbf{y}_*$ denote the corresponding label; the predictive density is obtained by marginalizing out parameters and latent functions via

$$p(\mathbf{y}_*|\mathbf{y}) = \int p(\mathbf{y}_*|\mathbf{f}_*) p(\mathbf{f}_*|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y}) d\mathbf{f}_* d\mathbf{f} d\boldsymbol{\theta} \ .$$

In this work, we propose to estimate this integral by obtaining posterior samples from $p(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y})$ using MCMC methods. The Appendix gives details of how Monte-Carlo estimates of this predictive distribution may be obtained. Obtaining samples from $p(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y})$ is complex because of the structure of the model that makes $\mathbf{f}$ and $\boldsymbol{\theta}$ strongly coupled a posteriori. Also, there is no closed form for updating $\mathbf{f}$ and $\boldsymbol{\theta}$ using a standard Gibbs sampler, so samplers based on an accept/reject mechanism need to be employed (Metropolis-within-Gibbs samplers) with the effect of reducing efficiency. This motivates the use of efficient samplers to alleviate this problem as discussed next.

4. MCMC sampling strategies.

4.1. *Riemann Manifold MCMC methods.* The proposed model comprises a set of m latent functions, each of which has dimension n, and a set of $q \times m$ weights. Given the large number of strongly correlated variables involved in the model, we need to employ statistically efficient sampling methods to characterize the posterior distribution $p(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y})$.

Recently, a set of novel Monte Carlo methods for efficient posterior sampling has been proposed in Girolami and Calderhead (2011) which provides promising capability for challenging and high dimensional problems such as the one considered in this paper. In most sampling methods (with the exception of Gibbs sampling) it is crucial to tune any parameters of the proposal distributions in order to avoid strong dependency within the chain or the possibility that the chain does not move at all. As the dimensionality increases, this becomes a hugely challenging issue, given that several parameters need to be tuned and are crucial to the effectiveness of the sampler.

The sampling methods developed in Girolami and Calderhead (2011) aim at providing a systematic way of designing such proposals by exploiting the differential geometry of the underlying statistical model. The main quantity in this differential geometric approach to MCMC is the local Riemannian metric tensor which is the expected Fisher Information (FI) that defines the statistical manifold; see Girolami and Calderhead (2011) for full details. The intuition behind manifold MCMC methods is that the statistical manifold provides a structure that is suitable for making efficient proposals based on Langevin diffusion or Hamiltonian dynamics. In the case of the sampling of $\mathbf{f}$ using RM-HMC, let $G_{\mathbf{f}}$ be the metric tensor computed as the FI for the statistical manifold of $p(\mathbf{y}|\mathbf{f})$ plus the negative Hessian of $p(\mathbf{f}|\boldsymbol{\theta})$ (see the appendix for further details). Introducing an auxiliary momentum variable $\mathbf{p} \sim \mathcal{N}(\mathbf{p}|\mathbf{0}, G_{\mathbf{f}})$ as in HMC, RM-HMC can be derived by solving the dynamics associated with the Hamiltonian:

$$H(\mathbf{f}, \mathbf{p}) = -\log[p(\mathbf{y}, \mathbf{f}|\boldsymbol{\theta})] + \frac{1}{2}\log|G_{\mathbf{f}}| + \frac{1}{2}\mathbf{p}^{\mathrm{T}} G_{\mathbf{f}}^{-1}\mathbf{p} + \text{const.}$$

Given that the metric tensor is dependent on the value of $\mathbf{f}$, the Hamiltonian is therefore non-separable between $\mathbf{p}$ and $\mathbf{f}$, and a generalized leapfrog integrator must be employed (Girolami and Calderhead, 2011). RM-HMC can be seen as a generalization of Hybrid Monte Carlo (HMC) (Neal, 1993), where the mass matrix is now substituted by the metric tensor.

4.2. *Ancillary and Sufficient augmentation.* The proposed classification model is hierarchical, and the application of a Metropolis-within-Gibbs style scheme, sampling $\mathbf{f}|\boldsymbol{\theta}, \mathbf{y}$ then $\boldsymbol{\theta}|\mathbf{f}, \mathbf{y}$, leads to poor efficiency and slow convergence. This effect has drawn a lot of attention in the case of hierarchical models in general (Papaspiliopoulos, Roberts and Sköld, 2007; Yu and Meng, 2011), and recently in latent Gaussian models (Murray and Adams, 2010; Filippone, Zhong and Girolami, 2012). In order to decouple the strong posterior dependency between $\boldsymbol{\theta}$ and $\mathbf{f}$ we can apply reparametrization techniques, whereby we introduce a new set of variables $\boldsymbol{\nu}_c$ related to the old set of latent variables by a transformation $\mathbf{f}_c = g(\boldsymbol{\nu}_c, \boldsymbol{\theta}_c)$. This transformation can be chosen to achieve faster convergence as studied, e.g., in Papaspiliopoulos, Roberts and Sköld (2007), and should be designed to handle both strong and weak data limits, namely situations where data overwhelm the prior or not. In the terminology of Yu and Meng (2011), we identify two particular cases, namely Sufficient augmentation (SA), and Ancillary augmentation (AA). In the SA scheme the sampling of $\boldsymbol{\theta}$ is done by proposing $\boldsymbol{\theta}'|\boldsymbol{\theta}, \mathbf{f}, \mathbf{y}$. In the case of weak data, as it is the case in this application, the SA parametrization is inefficient, given the strong coupling between $\mathbf{f}$ and $\boldsymbol{\theta}$. In contrast, in the AA scheme, the new set of latent variables $\boldsymbol{\nu}_c$ is constructed to be a priori independent from $\boldsymbol{\theta}$; this is a good candidate to provide an efficient parametrization in cases of weak data. To see this, in the case of no data, the posterior over hyper-parameters and the newly defined latent variables $\boldsymbol{\nu}_c$ corresponds to the prior which is factorized and easy to explore.

This parametrization makes the $\boldsymbol{\nu}_c$ ancillary for $\mathbf{y}$. We propose to realize this by defining $\mathbf{f}_c = L_c\boldsymbol{\nu}_c$, where L_c is any square root of K_c (in the remainder of this paper we will consider L_c as the Cholesky factor of K_c). This sampling scheme amounts to sampling $\boldsymbol{\theta}'|\boldsymbol{\theta}, \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_m, \mathbf{y}$. In the next section we will report experiments showing the relative merits of SA and AA combined with manifold methods, with the ultimate goal of achieving efficiency in inferring latent functions and hyper-parameters in our application. All implemented methods were tested for correctness as proposed by Geweke (2004), and convergence analysis was performed using the $\hat{R}$ potential scale reduction factor (Gelman and Rubin, 1992).

5. Comparison of MCMC sampling strategies.

In this section we investigate the efficiency of various MCMC sampling strategies in our application. Table 1 lists the sampling approaches that we considered for this study. All approaches make use of RM-HMC for sampling the latent variables with different metrics. Approach (a) uses a simple isotropic metric so that RM-HMC is effectively HMC with an identity mass matrix. Approaches (c), (d), and (f) use the metric derived from the FI (see appendix), while approaches (b) and (e) use an alternative homogeneous metric, defined as $\hat{F} = K^{-1} + \mathrm{diag}(\boldsymbol{\pi}_p) - \Phi_p\Phi_p^{\mathrm{T}}$. Note that this is similar to the definition of the metric tensor outlined in the appendix, except $\boldsymbol{\pi}_p$ and Φ_p are defined by the prior frequency of the classes in the training set instead of by the likelihood. Employing this homogeneous metric is less efficient than employing a position specific metric but it holds two practical advantages: (i) it has a substantially lower computational cost since it does not recompute and invert the metric tensor at every step and (ii) the explicit leapfrog integrator may be used in place of the generalized (implicit) leapfrog integrator used in Girolami and Calderhead (2011).

In sampling the hyper-parameters, approaches (a)-(c) effectively employ an HMC proposal with identity mass matrix with SA parametrization. Approach (d) uses RM-HMC with metric derived from the FI as shown in appendix, whereas approaches (e) and (f) make use of Metropolis-Hastings (MH) with an identity covariance with AA parametrization.

Approach (a) can be viewed as a simple baseline approach since it does not incorporate any knowledge on the curvature of the target distribution and attempts to explore the parameter space by isotropic proposals. It is presented primarily as a ref-

TABLE 1
Sampling schemes evaluated

Approach	$p(\mathbf{f}'\|\mathbf{f},\boldsymbol{\theta})$		$p(\boldsymbol{\theta}'\|\mathbf{f},\boldsymbol{\theta})$		
	Sampler	Metric	Sampler	Metric	Scheme
(a)	RM-HMC	I	RM-HMC	I	SA
(b)	RM-HMC	$\hat{F}$	RM-HMC	I	SA
(c)	RM-HMC	$G_{\mathbf{f}}$	RM-HMC	I	SA
(d)	RM-HMC	$G_{\mathbf{f}}$	RM-HMC	$G_{\boldsymbol{\theta}}$	SA
(e)	RM-HMC	$\hat{F}$	MH	–	AA
(f)	RM-HMC	$G_{\mathbf{f}}$	MH	–	AA

erence for the other approaches. Approaches (b) and (c) employ manifold methods to efficiently sample the latent variables only while approaches (d) also applies them to sample the hyper-parameters. Approaches (e) and (f) employ manifold methods for the latent variables and an MH sampler with AA for the hyperparameters.

For all the experiments that follow, we applied an independent Gamma prior to each weight $\exp(\theta_{cs})$, with $a = b = 2$, where a and b refer respectively to shape and rate parameters. This prior is relatively vague but nevertheless constrained the sampler to a plausible parameter range.

We tuned each of the sampling approaches described above using pilot runs and assessed convergence by recording when all sampled variables had $\hat{R} < 1.1$. According to this criterion, sampling approaches (e) and (f) converged after 1,000 iterations for the latent function variables and after a few thousands of iterations for the hyperparameters. Sampling approaches (a-d) did not converge even after 100,000 iterations, so will not be considered further. This demonstrates that the structure of the model poses a serious challenge in efficiently sampling $\mathbf{f}$ and $\boldsymbol{\theta}$, no matter how efficient are the individual samplers employed in the Metropolis-within-Gibbs sampler. For all subsequent analysis, we discarded all samples acquired prior to convergence (burn-in). A plot reporting the evolution of Gelman and Rubin's shrink factor vs the number of iterations (for the first 10,000 iterations) for the slowest variable to converge is reported in figure 2. The left and right panel of figure 2 correspond to the slowest variable in the approach (e) for the multi-modality and multi-region classifiers (see next section) respectively; in both cases the slowest variable was one of the hyper-parameters.

For the latent function variables, we used an RM-HMC trajectory length of 10 leapfrog steps and a step size of 0.5 for sampling approaches (e) and (f). This appeared to be near optimal and yielded an acceptance rate in the range of 60-70%, while keeping correlation between successive samples relatively low. For the hyperparameters, a step size of 0.2 yielded an acceptance rate in the range of 60-70% although correlation between successive samples remained high (see below).

We report the Effective Sample Size (ESS) (Geyer, 1992) for each method in ta-

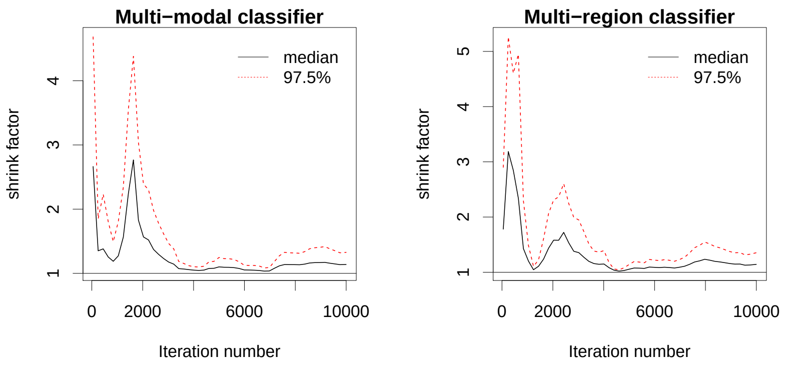

Figure 2. *Convergence analysis: plot reporting the evolution of Gelman and Rubin's shrink factor vs the number of iterations for the slowest variable to converge in approach (e) for the multi-modality classifier (left panel) and the multi-region classifier (right panel).*

Table 2
Efficiency of converged sampling schemes (Multi-source classifier) Min and max refer to the minimum and maximum ESS across all sampled variables

Approach	mean % ESS_f (min, max)	mean % ESS_θ (min, max)
(e)	27.04 (5.31, 48.06)	0.34 (0.21, 0.48)
(f)	24.11 (5.71, 42.86)	0.31 (0.19, 0.42)

ble 2, expressed as a percentage of the total number of samples. The ESS is an autocorrelation based method that is used to estimate the number of independent samples within a set of samples obtained from an MCMC method. Both approaches (e) and (f) sampled the latent function variables relatively efficiently although there was some variability between different variables. Sampling of the hyperparameters was much more challenging than the sampling of the latent function variables, and the MH samplers achieved an ESS less than 0.5% for all variables. Thus, for subsequent analysis we ran a relatively long Markov chain (5 million iterations) which we thinned by a factor of 500, ensuring independent sampling for all variables. Note that RM-HMC with metric $\hat{F}$ and RM-HMC with matrix $G_{\mathbf{f}}$ (approaches (e) and (f)) performed approximately equivalently for sampling the latent function variables. Thus, for the remainder of this paper we focus on the results obtained from the sampler that employed the homogeneous metric $\hat{F}$ for the latent functions and MH for the hyperparameters (i.e. approach (e)) owing to its lower computational cost. The results reported in this section are in line with what observed in a recent extensive study on the fully Bayesian treatment of models involving GP priors (Filippone, Zhong and Girolami, 2012). In particular, it has been reported that the sampling of the latent variables can be done efficiently using RM-HMC and a variant with the homogeneous metric $\hat{F}$, and that for the hyper-parameters the MH proposal with the AA parametrization is a good compromise between efficiency and computational cost.

6. Predictive accuracy and assessment of neuroimaging data modalities. In this section, we have three main objectives: first, we aim to demonstrate that the predictive approach we propose can accurately discriminate between multiple neurological conditions. Second, we investigate which neuroimaging data modalities carry discriminating information for these disorders and whether greater predictive performance can be achieved by combining multiple modalities. Finally, we investigate the predictive ability of different brain regions for discriminating between each of the disorders.

For estimating the predictive ability of the classifiers we performed four-fold cross-validation (CV) In the CV procedure, we randomly partitioned the data into four folds so that each CV fold contained approximately the same frequency of classes as in the entire data set. We then carried out the inference leaving out one fold that we used to assess the accuracy of the proposed method; leaving out one fold at a time it is possible to obtain an estimate of performance on unseen data.

We compared the performance of the proposed multinomial logit model with simpleMKL. Similar to the proposed method, simpleMKL allows an optimal linear combination of data sources or brain regions to be inferred from the data but unlike the proposed approach, simpleMKL is not a probabilistic model. In the MKL literature, each data source is referred to as a 'kernel' which corresponds to a covariance function for the proposed multinomial logit model. Since SVMs do not support true multi-class classification, we employed a 'one-vs-all' approach to combine multiple binary classifiers to provide a multi-class decision function. This has the consequence that simpleMKL estimates a linear combination of kernels that provides optimal accuracy across all binary classification decisions, and is therefore not able to estimate an independent set of kernel weighting factors for each class. To ensure the comparison with the multinomial logit model was as fair as possible we used nested cross-validation to find an optimal value for the SVM regularization parameter C. We achieved this by performing an inner 'leave-one-out' cross validation cycle ('validation') within each outer four fold cycle ('test') while we varied C logarithmically across a wide range of values (10^{-5} to 10^{5} in steps of 10). We selected the value of C that provided the optimal accuracy on the validation set, before applying it to the test set. To further examine whether any performance difference could be attributed to the extension of simpleMKL to multiclass classification, we also compared the accuracy of simpleMKL and the proposed model on all possible binary classification decisions. For simpleMKL, we used the implementation provided by Rakotomamonjy et al. (2008) where we used the 'weight variation' stopping criterion and the default options.

We employed two measures of predictive performance (i) balanced classification accuracy, which measures the mean number of correct predictions across all classes assuming a zero-one loss and (ii) a multi-class Brier score, which also quantifies the confidence of classifier predictions on w unseen samples and can be computed as the following error measure: $B = \frac{1}{w}\sum_{i=1}^{w}\sum_{c=1}^{m}(\pi_{ic}^{*} - y_{ic}^{*})^2$. Note that SimpleMKL does not provide probabilistic predictions, so the Brier score is not appropriate to evaluate the performance of this algorithm. Gneiting and Raftery (2007) reported studies on the connections between the Brier score and predictive accuracy in the case of two class classification, reporting that simple accuracy is a proper score unlike the Brier score which is strictly proper. We are unaware of any results on the connections between the two scores in the case of multi-class classification.

For comparison we also present predictive accuracy measures derived from classifiers using each data source independently, and a classifier using an unweighted linear sum of data sources (i.e. $K_c = \sum_{s=1}^{q} C_s$ for all c).

6.1. *Multi-modal classifier.* We first studied the classification problem based on the five data sources obtained from the three modalities, namely GM, WM, T2, FA, and MD, as explained in section 2, so that $q = 5$. The overall performance of each model is summarized in table 3. Note that all classifiers exceeded the predictive accuracy that would be expected by chance (i.e. $p < 0.05$, χ^2 test).

TABLE 3
Predictive accuracy (multi-source classifier). Min and max values refer to minimum and maximum values across CV folds

	Input data	Accuracy (min, max)	Brier score (min, max)
1	GM only	0.627 (0.321, 0.854)	0.667 (0.636, 0.712)
2	WM only	0.603 (0.350, 0.771)	0.653 (0.609, 0.710)
3	T2 only	0.545 (0.500, 0.604)	0.663 (0.619, 0.695)
4	FA only	0.569 (0.442, 0.688)	0.675 (0.645, 0.703)
5	MD only	0.623 (0.533, 0.750)	0.631 (0.588, 0.680)
6	Weighted sum	0.598 (0.350, 0.708)	0.588 (0.517, 0.662)
7	Unweighted sum	0.610 (0.400, 0.708)	0.553 (0.469, 0.646)
8	SimpleMKL	0.418 (0.143, 0.625)	-

From table 3, it is apparent that classifiers based on the T2 and FA data sources achieved lower classification accuracy than all the other data sources, suggesting they are not ideally suited to discriminating between these disorders. Further, the linear combinations of sources did not achieve higher accuracy than the best individual data source and the highest accuracy was obtained using the GM images only, although the difference is relatively small. The SimpleMKL classifier produced lower accuracy than either linear combination derived from the multinomial logit model. The mean accuracy for the binary classifiers over all pairs of classes was

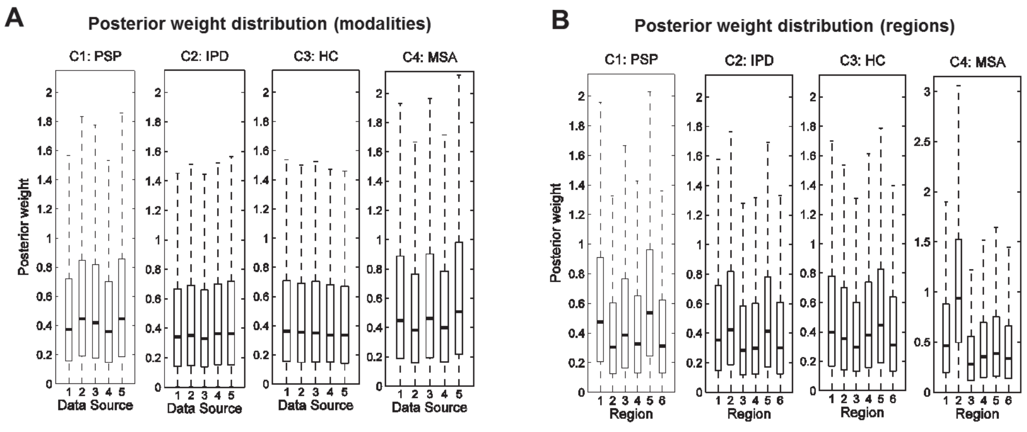

FIGURE 3. *Posterior distributions for the predictive weights for the multi-modality classifier (A) and the multi-region classifier (B). Panel A: Data sources: (1) GM, (2) WM, (3) T2, (4) FA, (5) MD. Panel B: Regions: (1) brainstem, (2) cerebellum, (3) caudate, (4) middle occipital gyrus, (5) putamen, (6) all other regions*

slightly higher for the GP classifiers (0.807) relative to simpleMKL (0.741), suggesting that most of the performance difference between simpleMKL and the proposed multinomial logit approach can be attributed to the extension of simpleMKL to the multiclass case.

In contrast to the outcomes for classification accuracy, the linear combinations of data sources produced more accurate probabilistic predictions than any of the individual modalities, indicative of a disjunction between categorical classification accuracy and accurate quantification of predictive uncertainty (table 3). This is probably a result of this model having greater flexibility to scale the magnitude of the latent function variables.

6.1.1. *Covariance parameters for the latent functions.* The posterior distribution of the weights is an important secondary outcome from this model and is summarized in figure 3A. These hyper-parameters collectively describe the relative contribution (or weighting factors) for each modality in deriving the prediction for each class.

The posterior class distribution for covariance weights in this model is relatively flat across all modalities for each class although each weight has slightly greater magnitude for the PSP and MSA classes relative to the other classes. Overall, the results from this section provide evidence that all imaging modalities contain similar information for discriminating disease groups. In other words, we found little benefit from combining multiple neuroimaging sequences. This has the important implica-

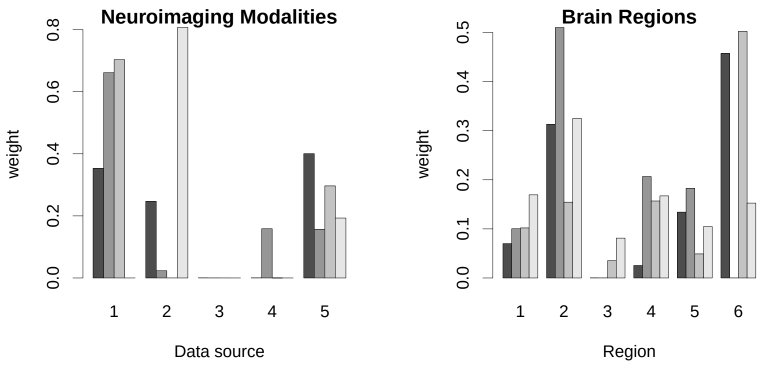

FIGURE 4. *Weights of the neuroimaging modalities (left panel) and brain regions (right panel) obtained by SimpleMKL across the four folds. Left panel: Data sources: (1) GM, (2) WM, (3) T2, (4) FA, (5) MD. Right panel: Regions: (1) brainstem, (2) cerebellum, (3) caudate, (4) middle occipital gyrus, (5) putamen, (6) all other regions*

tion that for the purposes of discrimination it appears sufficient to acquire a single structural MRI scan (i.e. SPGR image), which is comparatively rapid and inexpensive to acquire. Although the T2 images are also relatively inexpensive, they do not offer the same discriminative value and the DTI images, which are time-consuming and expensive to acquire, and appear to offer little additional benefit.

In the left panel of Fig. 4 we report the kernel weights obtained by SimpleMKL across the four folds. We can see how the values of the weights are not consistent across the folds; for example, GM is given zero weight in the fourth fold, whereas it seems to be important for the other three cases. Modality T2, instead, is consistently given zero weight across the four folds suggesting that this might not add any information to the other modalities. This is in contrast with the results of the probabilistic classifier that suggests that there is not much evidence in the data to completely ignore the information from one of the modalities.

6.2. *Multi-region classifier.* In this section, we illustrate how the proposed methodology may be used to estimate the predictive value of different brain regions for classification although we will investigate the relative contribution of different brain regions in greater detail and comment on the clinical significance of these findings in a separate report. For neurological applications, it is primarily important to assist interpretation, since it is desirable to identify differential patterns of regional pathology for each disease. While there are other methods to achieve this goal, an advantage of the proposed approach is that it provides a full posterior distribution over regional weighting parameters. For this analysis we used only the GM data modality, since it showed the highest discrimination accuracy, and used an anatomical template (Shattuck et al., 2008) to parcellate the GM images into six regions:

Table 4
Efficiency of converged sampling schemes (multi-region classifier). Min and max refer to the minimum and maximum ESS across all sampled variables

Approach	mean % ESS_f (min, max)	mean % ESS_θ (min, max)
(e)	8.52 (2.37, 34.85)	0.28 (0.11, 0.64)
(f)	9.42 (2.32, 35.44)	0.31 (0.11, 0.50)

(i) brainstem, (ii) bilateral cerebellum, (iii) bilateral caudate, (iv) bilateral middle occipital gyrus, (v) bilateral putamen and (vi) all other regions, so that now $q = 6$. As described above, the cerebellum, brainstem, caudate and putamen are affected to varying degrees in MSA, PSP or IPD. The middle occipital gyrus region was selected as a control region, as this is hypothesized to contain minimal discriminatory information.

All sampling approaches performed similarly for this problem in that none of the sampling approaches (a-d) converged after 100,000 iterations and sampling approaches (e) and (f) converged after 1,000 iterations for the latent function variables and after a few thousands of iterations for the hyperparameters (see the right panel of figure 2). Table 4 reports the efficiency of the sampling approaches (e) and (f) as they were the only ones that converged in a reasonable number of iterations.

The sampling efficiency for the latent function values was somewhat lower for this problem than for the multi-modal prediction problem described in the previous section. On average, the sampling efficiency for the hyper-parameters was approximately equivalent to the values reported above, but the minimum ESS was slightly lower. To accommodate this, we thinned all Markov chains by a factor of 1,000 ensuring approximately independent sampling for all variables.

Predictive accuracy measures for the multi-region classifier are presented in table 5. All classifiers exceeded the predictive accuracy that would be expected by chance ($p < 0.05$, χ^2 test) except the simpleMKL classifier which performed very poorly for this dataset. As for the multi-modal classifier, we compared the predictive accuracy of simpleMKL to the logit model across all binary classification decisions. In this case, the models produced similar accuracy (0.765 for the GP classifiers, 0.780 for simpleMKL. This provides further evidence that the suboptimal performances of simpleMKL can be traced to the extension of the binary SVM to the multi-class setting. In particular, it is likely that the suboptimal performance of simpleMKL in the multi-class context is due to the fact that it does not support different weighting factors for each class. In this case, the classifiers using weighted and unweighted covariance sums of brain regions produced the most accurate predictions and quantified predictive confidence most accurately. Again, there was negligible difference between the classifiers using the weighted and unweighted sums.

Table 5
Predictive accuracy (multi-region classifier). Min and max values refer to minimum and maximum values across CV folds

	Input data	Accuracy (min, max)	Brier score (min, max)
1	Brainstem only	0.578 (0.500, 0.688)	0.595 (0.555, 0.643)
2	Cerebellum only	0.478 (0.333, 0.562)	0.634 (0.593, 0.643)
3	Caudate only	0.349 (0.221, 0.520)	0.737 (0.693, 0.764)
4	Mid. Occipital Gyrus only	0.419 (0.333, 0.479)	0.741 (0.677, 0.773)
5	Putamen only	0.438 (0.354, 0.521)	0.668 (0.604, 0.718)
6	All other regions	0.424 (0.321, 0.563)	0.753 (0.724, 0.779)
7	Weighted sum	0.614 (0.500, 0.708)	0.547 (0.499, 0.593)
8	Unweighted sum	0.624 (0.500, 0.708)	0.546 (0.501, 0.592)
9	SimpleMKL	0.229 (0.111, 0.375)	-

6.2.1. *Covariance parameters for the latent functions.* The posterior distribution of the weights for the multi-region classifieris summarized in figure 3B. The posterior means of the weighting factors were again relatively constant between brain regions and also showed a high variance. This indicates that the relative contribution of different brain regions was not strongly determined by the data and that we should be cautious about interpreting the relative contributions of the different brain regions using this approach. Nevertheless, the clearest differential effect among regions was for the cerebellum, where the lower quartile of the posterior distribution for the MSA class was greater than the mean of all other regions. In addition, the brainstem also made a small positive contribution towards predicting the MSA class. As described above, both the cerebellum and brainstem are known to undergo severe degeneration in MSA. The strongest positive contributions to predicting the PSP class were obtained from the brainstem, caudate and putamen, which once again are regions known to show the extensive degeneration in PSP. The regional weighting factors for the IPD and control groups were somewhat flatter, which is consistent with focal nature of the degeneration in early-mid IPD and with the observation that the brain scans of these groups are difficult to discriminate from one another. However, the posterior suggests that the cerebellum showed a relatively increased weighting relative to other regions for the IPD class, and that the putamen was assigned a relatively increased weighting for the IPD and HC classes, which is congruent with the expectation that these classes are characterized by greater GM concentration in those regions relative to the PSP and MSA classes respectively. From the current analysis, it is difficult to determine the regions having the greatest predictive value for discriminating the PD from the HC group. As future work, separate binary classifiers trained to discriminate these classes directly may be beneficial in this respect. We notice also that the control region (i.e. the middle occipital gyrus) was assigned comparatively low weighting for every class.

Overall, the results from this section suggest that distributed patterns of abnormality across multiple brain regions are necessary to accurately discriminate between classes. The neurodegenerative disorders studied in the present work have relatively well-defined regional pathology, but even in this case the most accurate predictions were obtained from the classifiers using all brain regions.

Again, the analysis of the weights obtained by simpleMKL (right panel of Fig. 4) shows that the non-probabilistic classifier obtains sparse solutions for the weights that are not consistent across the folds, thus preventing one from being able to properly assess the role played by each region in the classification task.

6.3. *Results with the Dirichlet prior.* Here we briefly discuss the results obtained when imposing a Dirichlet prior on the weights $\exp(\theta_{cs})$, focusing only on the multi-region classifier for the sake of brevity. The sampling strategy was as in approach (e), with the difference that the update of the hyper-parameters followed a MH sampling with proposal based on Dirichlet distributions. In order to test the robustness to prior specification, we added a further level in the hierarchy of the model by imposing a prior over the concentration parameter of the Dirichlet distribution, so that the model had a joint density $p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta}, \alpha) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\alpha)p(\alpha)$. Including a hyper-prior over the concentration parameter has the effect of averaging out the choice of the prior and inferring the concentration parameter allows inference of levels of sparsity from the data.

From the computational perspective, the sampling of α induces a further level of correlation in the chains. We ran thorough convergence tests, and we obtained similar convergence and efficiency results as in the previous analysis. We chose a fairly diffuse hyper-prior $p(\alpha)$ as exponential with unit rate, so that $\mathrm{E}[\alpha] = 1$, which corresponds to a uniform prior over the simplex for the weights.

Note that data has quite a weak effect in informing the posterior over the concentration parameter, as they are three levels apart in the hierarchy (Goel and DeGroot, 1981). Nevertheless, comparing prior and posterior over α, we notice a slight reduction in the interquartile range from $[0.29, 1.39]$ to $[0.51, 1.49]$ and a shift of the mean from 1 to 1.16, thus supporting a diffuse (non-sparse) prior over the weights. In terms of questions addressed in this particular application, the results obtained by adding a hyper-prior lead to the same conclusions.

7. Refining predictions using predictive probabilities.

We have discussed how a probabilistic approach allows us to assess the importance of different neuroimaging modalities in disease classification. Another advantage of employing a probabilistic classification model is that predictive probabilities quantify the uncertainty in the outcome, which allow a ”reject option” to be specified. Under this framework, the researcher specifies in advance a confidence threshold below which a prediction is considered to be inconclusive. In cases where the maximum class probability does not exceed this threshold, the final decision may be deferred to

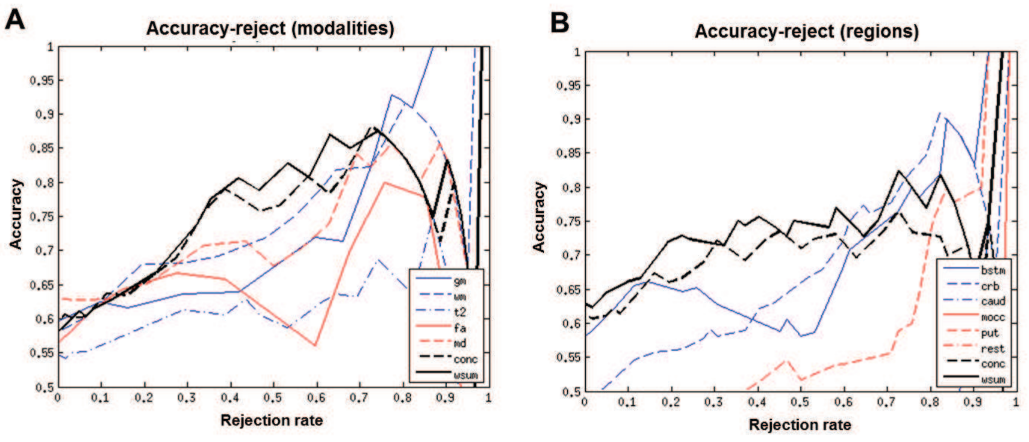

FIGURE 5. *A: Accuracy-reject curve for multi-modality classifiers. B: Accuracy-reject curve for multi-region classifiers.*

another classification model or a human expert. To investigate the suitability of the proposed classifier for this approach and to assess the accuracy of the classifier across varying rejection thresholds, we plotted accuracy-reject curves for each of the classifiers investigated in this work (figure 5). These were constructed by varying the rejection threshold monotonically from 0 to 1 in steps of 0.01. At each threshold, we computed the rejection rate as the proportion of samples for which the most confident class prediction did not exceed the rejection threshold and measured the accuracy of the remaining samples. The accuracy-reject curves were then generated by plotting accuracy as a function of rejection rate.

The accuracy-reject curves show that: (i) predictive performance increases monotonically across most rejection rates and (ii) the multi-source classifiers perform better than any of the individual modalities or brain regions across most rejection rates. This implies that the multi-source classifiers not only make fewer errors, but also quantify predictive uncertainty more accurately than any of the individual regions. At high rejection thresholds, the multi-modality classifier is outperformed by the GM modality, owing to two confident misclassifications deriving from the FA and MD modalities, suggesting the possibility of atypical white matter pathology in these subjects.

8. Conclusions. In this paper we presented the application of a multinomial logit model based on GP priors to the problem of classification of neurological disorders based on neuroimaging measures. The proposed model is flexible and highly descriptive, and it can be employed in scenarios where the focus is on gaining insights into the relative importance of different data modalities or brain regions in the application under study. Also, it allows accurate quantification of the uncertainty in the predictions, which is crucial in several applications and especially for predicting disease state in clinical applications.

From a statistical perspective, carrying out the inference task in the model presented in this paper and in latent Gaussian models in general represents a serious challenge. This paper presented a combination of advanced inference techniques based on MCMC methods that allowed us to tackle this problem in an efficient way. Predictions for unseen data were obtained by integrating out all the parameters in the model, thus capturing the uncertainty in the inferred parameters. We also investigated the use of a hyper-prior to integrate out the choice of the prior.

The motivating application for this study aimed to use neuroimaging measures to classify a cohort of 62 participants, consisting of both healthy controls and patients affected by one of three variants of Parkinsonian disorder. We demonstrated accurate classification of disease state that compares favourably with the only existing study of which we are aware of employing whole-brain neuroimaging measures to discriminate between these disorders (Focke et al., 2011). For future work it will be important to: (i) validate how well the predictive accuracy obtained here generalizes to earlier disease stages and (ii) investigate methods to improve the predictive accuracy beyond what was reached here, which will become increasingly important when the proposed method is evaluated in early stage disease. Construction of classification features from brain images that better reflect the underlying pathology may be particularly beneficial in this regard. We showed how the results of the inference allowed us to draw conclusions regarding the relative importance of neuroimaging measures and brain regions in discriminating between classes. We also compared the results with SimpleMKL, a non-probabilistic multi-modality classifier based on SVMs, which shows lower accuracy and most importantly is not able to address questions regarding the relative importance of neuroimaging measures and brain regions in a statistically consistent way. In contrast, the proposed method was able to give insights into the predictive ability of the different neuroimaging sequences, and suggested that all the modalities investigated in this study carry similar discriminative information. This has important implications for planning future studies, and suggests that there is little benefit in acquiring multiple neuroimaging sequences. Instead, for the purposes of prediction, acquiring a single structural brain image is probably the most cost-effective approach. Another level of the analysis showed that the proposed method was able to quantify the predictive ability of different brain regions for discriminating between classes. Similar to the previous analysis, this analysis showed that all regions carry some discriminative information, but at the same time seems to indicate that some of them have greater predictive ability than others for different classes. Further, the regional distribution of these regions is in accordance with the known pathology of the disorders based on the clinical literature.

APPENDIX A: DATA ACQUISITION DETAILS

For each subject, a T2-weighted structural image, a T1-weighted spoiled gradient recalled (SPGR) structural image and a DTI sequence were acquired using a 1.5T GE Signa LX NVi scanner (General Electric, WI, USA). All images had whole brain coverage and imaging parameters for the T2 weighted images and DTI sequence have been described previously (Blain et al., 2006). Imaging parameters for the SPGR imaging sequence were: repetition time = 18ms, echo time = 5.1 ms, inversion time = 450 ms, matrix size = 256×152, field of view (FOV) = 240×240. SPGR Images were reconstructed over a 240×240 FOV, yielding an in-plane resolution of 0.94×0.94mm and 124 1.5 mm thick slices. Subjects provided informed written consent, and the study was approved by the local Research Ethics Committee.

APPENDIX B: MCMC ADDITIONAL DETAILS

Let $K_{*\cdot}$ be an $m \times (mn)$ block diagonal rectangular matrix where entries in the r-th diagonal block contain the covariance of the test sample with the training samples corresponding to the r-th covariance K_r. Also, let K_{**} be an $m \times m$ matrix where the i, j entry is the covariance of the test sample corresponding to the covariances K_i and K_j. A priori we assumed zero covariance across latent functions, so K_{**} will be diagonal. Using the properties of GPs, given $\mathbf{f}$ and $\boldsymbol{\theta}$, then $p(\mathbf{f}_*|\mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \Sigma_*)$ with $\boldsymbol{\mu}_* = K_{*\cdot}K^{-1}\mathbf{f}$ and $\Sigma_* = K_{**} - K_{*\cdot}K^{-1}K_{\cdot *}$. Given N_1 independent posterior samples for $\mathbf{f}$ and $\boldsymbol{\theta}$, we can estimate the integral by

$$p(\mathbf{y}_*|\mathbf{y}) \approx \frac{1}{N_1} \sum_{i=1}^{N_1} \int p(\mathbf{y}_*|\mathbf{f}_*) p(\mathbf{f}_*|\mathbf{f}_{(i)}, \boldsymbol{\theta}_{(i)}) d\mathbf{f}_* \; .$$

Each of the former integrals can be estimated again by a Monte Carlo sum, by drawing N_2 independent samples from $p(\mathbf{f}_*|\mathbf{f}_{(i)}, \boldsymbol{\theta}_{(i)})$ which is Gaussian:

$$\int p(\mathbf{y}_*|\mathbf{f}_*) p(\mathbf{f}_*|\mathbf{f}_{(i)}, \boldsymbol{\theta}_{(i)}) d\mathbf{f}_* \approx \frac{1}{N_2} \sum_{j=1}^{N_2} p(\mathbf{y}_*|(\mathbf{f}_*)_{(j)}) \; .$$

The required gradients of the joint log-density follow as $\nabla_{\mathbf{f}} \mathcal{L} = -K^{-1}\mathbf{f} + \mathbf{y} - \boldsymbol{\pi}$ and

$$\frac{\partial \mathcal{L}}{\partial \theta_{cj}} = -\frac{1}{2} \exp[\theta_{cj}] \mathrm{Tr}\left(K_c^{-1} C_j\right) + \frac{1}{2} \exp[\theta_{cj}] \mathbf{f}_c^{\mathrm{T}} K_c^{-1} C_j K_c^{-1} \mathbf{f}_c + \frac{\partial p(\boldsymbol{\theta}_c)}{\partial \theta_{cj}} \; .$$

and the FI for the two groups of variables, along with the negative Hessian of the priors are $G_{\mathbf{f}} = K^{-1} + \mathrm{diag}(\boldsymbol{\pi}) - \Phi\Phi^{\mathrm{T}}$ and $(G_{\boldsymbol{\theta}})_{cjr} = \frac{1}{2} \exp[\theta_{cr} + \theta_{cj}] \mathrm{Tr}\left(K_c^{-1} C_r K_c^{-1} C_j\right)$

where Φ is a $(mn) \times n$ matrix obtained by stacking by row the matrices $\mathrm{diag}(\boldsymbol{\pi}_c)$. The derivatives of the two metric tensors needed to apply RM-HMC can be computed by using standard properties of derivatives of expressions involving matrices.

REFERENCES

Basser, P. J. and Jones, D. K. (2002). Diffusion-tensor MRI: theory, experimental design and data analysis - a technical review. *NMR in Biomedicine* **15** 456–467.

Blain, C. R. V., Barker, G. J., Jarosz, J. M., Coyle, N. A., Landau, S., Brown, R. G., Chaudhuri, K. R., Simmons, A., Jones, D. K., Williams, S. C. R. and Leigh, P. N. (2006). Measuring brain stem and cerebellar damage in Parkinsonian syndromes using diffusion tensor MRI. *Neurology* **67** 2199-2205.

Cuingnet, R., Gerardin, E., Tessieras, J., Auzias, G., Lehéricy, S., Habert, M.-O. O., Chupin, M., Benali, H. and Colliot, O. (2011). Automatic classification of patients with Alzheimer's disease from structural MRI: A comparison of ten methods using the ADNI database. *NeuroImage* **56** 766-781.

Farrall, A. J. (2006). Magnetic resonance imaging. *Practical Neurology* **6** 318-325.

Filippone, M., Zhong, M. and Girolami, M. (2012). On the fully Bayesian treatment of latent Gaussian models using stochastic simulations Technical Report No. TR-2012-329, School of Computing Science, University of Glasgow.

Focke, N. K., Helms, G., Scheewe, S., Pantel, P. M., Bachmann, C. G., Dechent, P., Ebentheuer, J., Mohr, A., Paulus, W. and Trenkwalder, C. (2011). Individual voxel-based subtype prediction can differentiate progressive supranuclear palsy from idiopathic Parkinson syndrome and healthy controls. *Human Brain Mapping* **32** 1905–1915.

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

Geweke, J. (2004). Getting it right: joint distribution tests of posterior simulators. *Journal of the American Statistical Association* **99** 799–804.

Geyer, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science* **7** 473–483.

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

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* **102** 359–378.

Goel, P. K. and DeGroot, M. H. (1981). Information about hyperparamters in hierarchical models. *Journal of the American Statistical Association* **76** 140–147.

Hauw, J., Daniel, S., Dickson, D., Horoupian, D., Jellinger, K., Lantos, P., McKee, A., Tabaton, M. and Litvan, I. (1994). Preliminary NINDS neuropathologic criteria for Steele-Richardson-Olszewski syndrome (progressive supranuclear palsy). *Neurology* **44** 2015-9.

Klöppel, S., Stonnington, C. M., Chu, C., Draganski, B., Scahill, R. I., Rohrer, J. D., Fox, N. C., Jack, C. R., Ashburner, J. and Frackowiak, R. S. J. (2008). Automatic classification of MR scans in Alzheimer's disease. *Brain* **131** 681-689.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P. L., Ghaoui, L. E. and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research* **5** 27-72.

Litvan, I., Bhatia, K. P., Burn, D. J., Goetz, C. G., Lang, A. E., McKeith, I., Quinn, N., Sethi, K. D., Shults, C. and Wenning, G. K. (2003). SIC Task Force appraisal of clinical diagnostic criteria for Parkinsonian disorders. *Movement Disorders* **18** 467–486.

Marquand, A. F., Mourão Miranda, J., Brammer, M. J., Cleare, A. J. and Fu, C. H. (2008). Neuroanatomy of verbal working memory as a diagnostic biomarker for depression. *Neuroreport*

19 1507–1511.

Murray, I. and Adams, R. P. (2010). Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems 23* (J. Lafferty, C. K. I. Williams, R. Zemel, J. Shawe-Taylor and A. Culotta, eds.) 1723–1731.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report No. CRG-TR-93-1, Dept. of Computer Science, University of Toronto.

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

Papaspiliopoulos, O., Roberts, G. O. and Sköld, M. (2007). A general framework for the parametrization of hierarchical models. *Statistical Science* **22** 59–73.

Rakotomamonjy, A., Bach, F. R., Canu, S. and Grandvalet, Y. (2008). SimpleMKL. *Journal of Machine Learning Research* **9** 2491–2521.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Seppi, K. (2007). MRI for the differential diagnosis of neurodegenerative Parkinsonism in clinical practice. *Parkinsonism & Related Disorders* **13** S400 - S405. Proceedings of the XVII WFN World Congress on Parkinson's Disease and Related Disorders.

Shattuck, D. W., Mirza, M., Adisetiyo, V., Hojatkashani, C., Salamon, G., Narr, K. L., Poldrack, R. A., Bilder, R. M. and Toga, A. W. (2008). Construction of a 3D probabilistic atlas of human cortical structures. *NeuroImage* **39** 1064–1080.

Sonnenburg, S., Rätsch, G., Schäfer, C. and Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research* **7** 1531-1565.

Wenning, G., Tison, F., Ben-Shlomo, Y., Daniel, S. and Quinn, N. (1997). Multiple system atrophy: a review of 203 pathologically proven cases. *Movement Disorders* **12** 133-47.

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

Yoshikawa, K., Nakata, Y., Yamada, K. and Nakagawa, M. (2004). Early pathological changes in the Parkinsonian brain demonstrated by diffusion tensor MRI. *Journal of Neurology, Neurosurgery & Psychiatry* **75** 481-484.

Yu, Y. and Meng, X.-L. (2011). To center or not to center: that is not the question – an Ancillarity–Sufficiency Interweaving Strategy (ASIS) for boosting MCMC efficiency. *Journal of Computational and Graphical Statistics* **20** 531–570.

School of Computing Science
University of Glasgow
E-mail: maurizio.filippone@glasgow.ac.uk

Institute of Psychiatry
King's College London
E-mail: andre.marquand@kcl.ac.uk
camillahertlein@googlemail.com
steve.williams@kcl.ac.uk

University College and
King's College London
E-mail: j.mourao-miranda@cs.ucl.ac.uk

Department of Statistical Science
Centre for Computational Statistics
and Machine Learning
University College London
E-mail: mark@stats.ucl.ac.uk

Pseudo-Marginal Bayesian Inference for Gaussian Processes

Maurizio Filippone and Mark Girolami

Abstract—The main challenges that arise when adopting Gaussian Process priors in probabilistic modeling are how to carry out exact Bayesian inference and how to account for uncertainty on model parameters when making model-based predictions on out-of-sample data. Using probit regression as an illustrative working example, this paper presents a general and effective methodology based on the pseudo-marginal approach to Markov chain Monte Carlo that efficiently addresses both of these issues. The results presented in this paper show improvements over existing sampling methods to simulate from the posterior distribution over the parameters defining the covariance function of the Gaussian Process prior. This is particularly important as it offers a powerful tool to carry out full Bayesian inference of Gaussian Process based hierarchic statistical models in general. The results also demonstrate that Monte Carlo based integration of all model parameters is actually feasible in this class of models providing a superior quantification of uncertainty in predictions. Extensive comparisons with respect to state-of-the-art probabilistic classifiers confirm this assertion.

Index Terms—Hierarchic Bayesian Models, Gaussian Processes, Markov chain Monte Carlo, Pseudo-Marginal Monte Carlo, Kernel Methods, Approximate Bayesian Inference.

1 Introduction

Non-parametric or kernel based models represent a successful class of statistical modelling and prediction methods. To focus ideas throughout the paper we employ the working example of predictive classification problems; the methodology presented, however, is applicable to all hierarchic Bayesian models in general and those employing Gaussian Process (GP) priors in particular. Important examples of kernel-based classifiers are the Support Vector Machine (SVM) [1], [2], the Relevance Vector Machine (RVM) [3], [4], and the Gaussian Process classifier [5]. Although these classifiers are based on different modeling assumptions and paradigms of statistical inference, they are characterized by a kernel function or covariance operator that allows one to build nonlinear classifiers able to tackle challenging problems [6], [7], [8], [9], [10], [11].

In order to allow these classifiers to be flexible, it is necessary to parameterize the kernel (or covariance) function by a set of so called *hyper-parameters*. After observing a set of training data, the aim is to estimate or infer such hyper-parameters. In the case of SVMs point estimates of hyper-parameters are obtained by optimizing a cross-validation error. This makes optimization viable only in the case of very few hyper-parameters, as grid search is usually employed, and is limited by the available amount of data. In GP classification, instead, the probabilistic nature of the model provides a means (usually after approximately integrating out latent variables) to obtain an approximate marginal likelihood that offers the possibility to optimize the hyper-parameters; this is known as type II Maximum Likelihood (ML) [12], [5]. Deterministic approximations for integrating out the latent variables include the Laplace Approximation (LA) [13], Expectation Propagation (EP) [14], Variational Bayes [15], Integrated Nested Laplace Approximations [16], and mean field approximations [17]; see [18], [19] for extensive assessments of the relative merits of different approximation schemes for GP classification.

From a fully Bayesian perspective, in a GP classifier one would like to be able to (i) infer all model parameters and latent variables from data and (ii) integrate out latent variables and hyper-parameters with respect to their posterior distribution when making predictions accounting for their uncertainty; this in particular, would effectively make the classifier *parameter free*. To date, the literature lacks a systematic way to efficiently tackle both of these questions. The main limitations are due the fact that it is not possible to obtain any of the quantities needed in the inference in closed form because of analytical intractability. This requires the use of some form of approximation, and deterministic approximations have been proposed to integrate out latent variables only; in order to integrate out the hyper-parameters most of the approaches propose quadrature methods [20], [16], that can only be employed in the case of a small number of hyper-parameters.

Recently, there have been a few attempts to carry out inference using stochastic approximations based on Markov chain Monte Carlo (MCMC) methods [21], [22], [23], the idea being to leverage asymptotic guarantees of convergence of Monte Carlo estimates to the true values. Unfortunately, employing MCMC methods for inferring latent variables and hyper-parameters is extremely chal-

- *M. Filippone is with the School of Computing Science, University of Glasgow, UK. E-mail: maurizio.filippone@glasgow.ac.uk*
- *M. Girolami is with the Department of Statistics, University of Warwick, UK. E-mail: m.girolami@warwick.ac.uk*

lenging, and state-of-the-art methods for doing so are still inefficient and difficult to use in practice.

This paper aims at providing a straightforward to implement methodology that is effective in the direction of bridging this gap, by proposing an MCMC method that addresses most of the difficulties that one is faced with when applying stochastic based inference in GP modeling, such as discrete label classification. The main issue in applying MCMC methods to carry out inference in GP classification is in sampling the hyper-parameters form the full posterior. This is due to the structure of the model that makes latent variables and hyper-parameters strongly coupled a posteriori; as a result, chains are characterized by low efficiency and poor mixing. The key idea of the proposed methodology is to break the correlation in the sampling between latent variables and hyper-parameters by approximately integrating out the latent variables while retaining a correct MCMC procedure; namely, maintaining the exact posterior distribution over hyper-parameters as the invariant distribution of the chains and ergodicity properties. This can be achieved by means of the so called *Pseudo Marginal* (PM) approach to Monte Carlo sampling [24], [25]. This work shows that the use of the PM approach leads to remarkable efficiency in the sampling of hyper-parameters, thus making the fully Bayesian treatment viable and simple to implement and employ.

The importance of integrating out the hyper-parameters to achieve a sound quantification of uncertainty in predictions is well known and has been highlighted, for example, in [12], [26], [16], [27]; employing this in practice, however, is notoriously challenging. The main motivation for this work, is to demonstrate that this marginalization can be done exactly, in a Monte Carlo sense, by building upon deterministic approximations already proposed in the GP and Machine Learning literature. This work reports a thorough empirical comparison in this direction, showing the ability of the fully Bayesian treatment to achieve a better quantification of uncertainty compared to the standard practice of optimization of the hyper-parameters in GP classification. Furthermore, the results report a comparison with a probabilistic version of the SVM classifier [28]. The results on GP classification support the argument that hyper-parameters should be integrated out to achieve a reliable quantification of uncertainty in applications and this paper provides a practical means to achieve this[1].

The paper is organized as follows: section 2 reviews the Gaussian Process approach to classification, and section 3 presents the proposed MCMC approach to obtain samples from the posterior distribution over both latent variables and hyper-parameters. Section 4 reports an assessment of the sampling efficiency achieved by the PM approach compared to other MCMC approaches, and section 5 reports a study on the performance of the fully Bayesian GP classifier compared to other probabilistic classifiers. Finally, section 6 concludes the paper.

1. The code to reproduce all the experiments is available at: http://www.dcs.gla.ac.uk/~maurizio/pages/code_pm/

2 Gaussian Process Classification

In this section, we briefly review GP classification based on a probit likelihood (see [5] for an extensive presentation of GPs). Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of n input vectors described by d covariates and associated with observed univariate responses $\mathbf{y} = \{y_1, \ldots, y_n\}$ with $y_i \in \{-1, +1\}$. Let $\mathbf{f} = \{f_1, \ldots, f_n\}$ be a set of latent variables. From a generative perspective, GP classifiers assume that the class labels have a Bernoulli distribution with success probability given by a transformation of the latent variables:

$$p(y_i|f_i) = \Phi(y_i f_i). \tag{1}$$

Here Φ denotes the cumulative function of the Gaussian density; based on this modeling assumption, the likelihood function is:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i). \tag{2}$$

The latent variables $\mathbf{f}$ are given a zero mean GP prior with covariance K:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{f}|\mathbf{0}, K). \tag{3}$$

Let $k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})$ be the function modeling the covariance between latent variables evaluated at the input vectors, parameterized by a vector of hyper-parameters $\boldsymbol{\theta}$. In this paper we will adopt a covariance function defined as follows:

$$k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}) = \sigma \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} A (\mathbf{x}_i - \mathbf{x}_j)\right]. \tag{4}$$

The parameter σ is the variance of the marginal prior distribution for each of the latent variables f_i. The matrix A, instead, defines the type of covariance between the values of the function at different input vectors. By defining a matrix A with a global parameter as follows,

$$A^{-1} = \tau^2 I, \tag{5}$$

an isotropic covariance function is obtained. Alternatively, A can be defined to assign a different parameter to each covariate

$$A^{-1} = \mathrm{diag}\left(\tau_1^2, \ldots, \tau_d^2\right). \tag{6}$$

The latter choice yields the so called *Automatic Relevance Determination* (ARD) prior [29]. In this formulation, the hyper-parameters τ_i can be interpreted as length-scale parameters. Let $\boldsymbol{\theta}$ be a vector comprising σ and all the length-scale parameters, and K be the matrix whose entries are $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})$.

The GP classification model is hierarchical, as $\mathbf{y}$ is conditioned on $\mathbf{f}$, and $\mathbf{f}$ is conditioned on $\boldsymbol{\theta}$ and the inputs X. In order to keep the notation uncluttered, in the remainder of this paper we will not report explicitly the conditioning on the inputs in any of the equations.

We now briefly review the types of approximations that have been proposed in the literature to employ GP classifiers.

2.1 Deterministic approximations for integrating out latent variables

One of the difficulties encountered in GP classification is that, unlike GP regression, the prior on the latent variables and the likelihood do not form a conjugate pair; therefore, it is not possible to analytically integrate out the latent variables. As a consequence, it is not possible to directly sample from or optimize the distribution of hyper-parameters given the labels, nor directly evaluate predictive probabilities. This has motivated a large body of research that attempts to approximate the posterior distribution over the latent variables $p(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$ with a Gaussian $q(\mathbf{f}|\mathbf{y},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mu_q,\Sigma_q)$ in order to exploit conjugacy. By doing so, it is possible to analytically integrate out latent variables to obtain an approximate marginal likelihood, and compute the predictive distribution for new data, as discussed in the following. The Gaussian approximation yields an approximate marginal likelihood $\hat{p}(\mathbf{y}|\boldsymbol{\theta})$ that can then be optimized with respect to the hyper-parameters, or used to obtain samples from the approximate posterior distribution over the hyper-parameters, say $\hat{p}(\boldsymbol{\theta}|\mathbf{y})$, using MCMC techniques. We now briefly discuss how this can be achieved.

To obtain an approximate predictive distribution, conditioned on a value of the hyper-parameters $\boldsymbol{\theta}$, we can compute:

$$p(y_*|\mathbf{y},\boldsymbol{\theta}) = \int p(y_*|f_*)p(f_*|\mathbf{f},\boldsymbol{\theta})q(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})df_*d\mathbf{f}. \quad (7)$$

Here $\boldsymbol{\theta}$ can be a ML estimate that maximizes the approximate likelihood or one sample from the approximate posterior $\hat{p}(\boldsymbol{\theta}|\mathbf{y})$. For simplicity of notation, let K be the covariance matrix evaluated at $\boldsymbol{\theta}$, $\mathbf{k}_*$ the vector whose ith element is $k(\mathbf{x}_i,\mathbf{x}_*|\boldsymbol{\theta})$ and $k_{**} = k(\mathbf{x}_*,\mathbf{x}_*|\boldsymbol{\theta})$. Given the properties of multivariate normal variables, f_* is distributed as $\mathcal{N}(f_*|\mu_*,\beta_*^2)$ with $\mu_* = \mathbf{k}_*^{\mathrm{T}}K^{-1}\mathbf{f}$ and $\beta_*^2 = k_{**} - \mathbf{k}_*^{\mathrm{T}}K^{-1}\mathbf{k}_*$. Approximating $p(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$ with a Gaussian $q(\mathbf{f}|\mathbf{y},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}_q,\Sigma_q)$ makes it possible to analytically perform integration with respect to $\mathbf{f}$ in equation 7. In particular, the integration with respect to $\mathbf{f}$ yields $\mathcal{N}(f_*|m_*,s_*^2)$ with

$$m_* = \mathbf{k}_*^{\mathrm{T}}K^{-1}\boldsymbol{\mu}_q,$$

and

$$s_*^2 = k_{**} - \mathbf{k}_*^{\mathrm{T}}K^{-1}\mathbf{k}_* + \mathbf{k}_*^{\mathrm{T}}K^{-1}\Sigma_q K^{-1}\mathbf{k}_*.$$

The univariate integration with respect to f_* follows exactly in the case of a probit likelihood, as it is a convolution of a Gaussian and a cumulative Gaussian

$$\int p(y_*|f_*)\mathcal{N}(f_*|m_*,s_*^2)df_* = \Phi\left(\frac{m_*}{\sqrt{1+s_*^2}}\right). \quad (8)$$

We now briefly review two popular approximation methods for integrating out latent variables, namely the Laplace Approximation and Expectation Propagation.

2.1.1 Laplace Approximation

The Laplace Approximation (LA) is based on the assumption that the distribution of interest can be approximated by a Gaussian centered at its mode and with the same curvature. By analyzing the Taylor expansion of the logarithm of target and approximating densities, the latter requirement is satisfied by imposing an inverse covariance for the approximating Gaussian equal to the negative Hessian of the logarithm of the target density [30]. For a given value of the hyper-parameters $\boldsymbol{\theta}$, define

$$\Psi(\mathbf{f}) = \log[p(\mathbf{y}|\mathbf{f})] + \log[p(\mathbf{f}|\boldsymbol{\theta})] + \text{const}. \quad (9)$$

as the logarithm of the target density up to terms independent of $\mathbf{f}$. Performing a Laplace approximation amounts in defining a Gaussian $q(\mathbf{f}|\mathbf{y},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}},\hat{\Sigma})$, such that

$$\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} \Psi(\mathbf{f}) \qquad \text{and} \qquad \hat{\Sigma}^{-1} = -\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\hat{\mathbf{f}}). \quad (10)$$

As it is not possible to directly solve the maximization problem in equation 10, an iterative procedure based on the following Newton-Raphson formula is usually employed:

$$\mathbf{f}_{\text{new}} = \mathbf{f} - (\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f}))^{-1}\nabla_{\mathbf{f}}\Psi(\mathbf{f}) \quad (11)$$

starting from $\mathbf{f} = \mathbf{0}$ until convergence. The gradient and the Hessian of the log of the target density are:

$$\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\log[p(\mathbf{y}|\mathbf{f})] - K^{-1}\mathbf{f}, \quad (12)$$

$$\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\log[p(\mathbf{y}|\mathbf{f})] - K^{-1}. \quad (13)$$

Note that if $\log[p(\mathbf{y}|\mathbf{f})]$ is concave, such as in probit classification, $\Psi(\mathbf{f})$ has a unique maximum. Practically, the Newton-Raphson update in equation 11 is implemented by employing Woodbury identities to avoid inverting K directly (see section 3.4 of [5] for full details). In such an implementation, one $n \times n$ matrix factorization is needed at each iteration and no other $O(n^3)$ operations.

2.1.2 Expectation Propagation

The Expectation Propagation (EP) algorithm is based on the assumption that each individual term of the likelihood can be approximated by an unnormalized Gaussian

$$p(y_i|f_i) \simeq \tilde{Z}_i\mathcal{N}(f_i|\tilde{\mu}_i,\tilde{\sigma}_i^2). \quad (14)$$

Approximating each term in the likelihood by a Gaussian implies that the approximate likelihood, as a function of $\mathbf{f}$, is multivariate Gaussian

$$\mathcal{N}(\mathbf{f}|\tilde{\boldsymbol{\mu}},\tilde{\Sigma})\prod_{i=1}^{n}\tilde{Z}_i \quad (15)$$

with $\tilde{\boldsymbol{\mu}}_i = \tilde{\mu}_i$ and $\tilde{\Sigma}_{ii} = \tilde{\sigma}_i$.

Under this approximation, the posterior $p(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$ is approximated by a Gaussian $q(\mathbf{f}|\mathbf{y},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}},\hat{\Sigma})$ with:

$$\hat{\Sigma}^{-1} = K^{-1} + \tilde{\Sigma}^{-1} \qquad \text{and} \qquad \hat{\mathbf{f}} = \hat{\Sigma}\,\tilde{\Sigma}\,\tilde{\boldsymbol{\mu}} \tag{16}$$

The EP algorithm is characterized by the way the parameters $\tilde{Z}_i$, $\tilde{\mu}_i$, and $\tilde{\sigma}_i^2$ are optimized. The EP algorithm loops through the n factors approximating the likelihood updating those three parameters for each factor in turn. First, the so called *cavity distribution* is computed

$$q^{\backslash i}(f_i|\boldsymbol{\theta}) \propto \int p(\mathbf{f}|\boldsymbol{\theta}) \prod_{j\neq i} \tilde{Z}_j \mathcal{N}(f_j|\tilde{\mu}_j, \tilde{\sigma}_j^2), \tag{17}$$

which is obtained by leaving out the ith factor from $q(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$. Second, a revised Gaussian $q'(f_i|\boldsymbol{\theta})$, which closely approximates the product of the cavity distribution and the exact ith likelihood term, is sought. In particular, this is performed by minimizing the following Kullback-Leibler divergence:

$$\mathrm{KL}\left(q^{\backslash i}(f_i|\boldsymbol{\theta}) p(y_i|f_i) \middle\| q'(f_i|\boldsymbol{\theta}) \right), \tag{18}$$

which in practice boils down to matching the moments of the two distributions. Third, once the mean and variance of $q'(f_i|\boldsymbol{\theta})$ are computed, it is possible to derive the updated parameters $\tilde{Z}_i$, $\tilde{\mu}_i$, and $\tilde{\sigma}_i^2$ for the ith factor. The derivation of those equations is rather involved, and the reader is referred to [5] for full details; EP requires five operations in $O(n^3)$ at each iteration. Note that convergence of the EP algorithm is not guaranteed in general; however, for GP classification, no convergence issues have been reported in the literature. Furthermore, EP for GP classification has been reported to offer superior accuracy in approximations compared to other methods [18], [19].

2.2 Fully Bayesian treatment

In a fully Bayesian treatment, the aim is to integrate out latent variables as well as hyper-parameters:

$$p(y_*|\mathbf{y}) = \int p(y_*|f_*) p(f_*|\mathbf{f},\boldsymbol{\theta}) p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y}) df_* d\mathbf{f} d\boldsymbol{\theta}. \tag{19}$$

Again, the integration with respect to f_* can be done analytically, whereas the integration with respect to latent variables and hyper-parameters requires the posterior distribution $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$. One way to tackle the intractability in characterizing $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$ is to draw samples from $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$ using MCMC methods, so that a Monte Carlo estimate of the predictive distribution can be used

$$p(y_*|\mathbf{y}) \simeq \frac{1}{N} \sum_{i=1}^{N} \int p(y_*|f_*) p(f_*|\mathbf{f}^{(i)},\boldsymbol{\theta}^{(i)}) df_*, \tag{20}$$

where $\mathbf{f}^{(i)},\boldsymbol{\theta}^{(i)}$ denotes the ith sample from $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$. This estimate will asymptotically converge to the exact expectation $p(y_*|\mathbf{y})$.

3 MCMC Sampling From $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$

Sampling from the posterior over $\mathbf{f}$ and $\boldsymbol{\theta}$ by joint proposals is not feasible; it is extremely unlikely to propose a set of latent variables and hyper-parameters that are compatible with each other and observed data. In order to draw samples from $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$, it is therefore necessary to resort to a Gibbs sampler, whereby $\mathbf{f}$ and $\boldsymbol{\theta}$ are updated in turn. We now briefly review the state of the art in Gibbs sampling techniques for GP models, and propose a new Gibbs sampler based on the PM approach.

3.1 Drawing samples from $p(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$

Efficient sampling of the latent variables can be achieved by means of Elliptical Slice Sampling (ELL-SS) [31]. ELL-SS is based on an adaptation of the Slice Sampling algorithm [32] to propose new values of the latent variables. ELL-SS has the very appealing property of requiring no tuning, so that minimum user intervention is needed, and by the fact that once K is factorized the complexity of iterating ELL-SS is in $O(n^2)$. Recently, the efficiency of ELL-SS has been extensively demonstrated on several models involving GP priors [21].

Another way to efficiently sample latent variables in GP models is by means of a variant of Hybrid Monte Carlo (HMC) [33], [34] where the inverse mass matrix is set to the GP covariance K, as described in detail in [21]. This variant of HMC can be interpreted as a simplified version of Riemann manifold Hamiltonian Monte Carlo (RMHMC) [35] which makes it possible to obtain samples from the posterior distribution over $\mathbf{f}$ in $O(n^2)$ once K is factorized. Owing to its simplicity, in the remainder of this paper we will use ELL-SS to sample from the posterior over latent variables $p(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$.

3.2 Drawing samples from the posterior over θ employing reparameterization techniques

3.2.1 SA and AA parameterizations

In GP classification, efficiently sampling from the posterior distribution over the latent variables and hyper-parameters is complex because of their strong coupling [21], [22], [26]. The result of this strong coupling is that fixing $\mathbf{f}$ induces a sharply peaked posterior over $\boldsymbol{\theta}$ that makes the chain converge slowly and mix very poorly. This effect is illustrated in Figure 1. In particular, conditioning the sampling of $\boldsymbol{\theta}$ on $\mathbf{f}$ corresponds to considering the standard parameterization of GP models $\mathbf{y}|\mathbf{f}$ and $\mathbf{f}|\boldsymbol{\theta}$, which is also known as Sufficient Augmentation (SA) [36].

A better parameterization can be given by introducing a set of transformed (whitened) latent variables $\boldsymbol{\nu}$ [37]. The way $\boldsymbol{\nu}$ is defined is by $\mathbf{f} = L\boldsymbol{\nu}$, L being the Cholesky factor of K. In this parameterization, that is also known as Ancillary Augmentation (AA) [36], $\boldsymbol{\nu}$ are constructed to be a priori independent from the hyper-parameters (using L is convenient as it is needed also to evaluate the GP prior density). In the AA parameterization $\boldsymbol{\theta}$ is sampled from $p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\nu})$. The effect of conditioning on $\boldsymbol{\nu}$ makes the conditional posterior over $\boldsymbol{\theta}$ larger, as illustrated in Figure 1.

3.2.2 The Surrogate data model

In the Surrogate (SURR) data model proposed in [22], a set of auxiliary latent variables $\mathbf{g}$ is introduced as a noisy version of $\mathbf{f}$; in particular, $p(\mathbf{g}|\mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{g}|\mathbf{f}, S_{\boldsymbol{\theta}})$. This construction yields a conditional for $\mathbf{f}$ of the form $p(\mathbf{f}|\mathbf{g}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, R)$, with $R = S_{\boldsymbol{\theta}} - S_{\boldsymbol{\theta}}(S_{\boldsymbol{\theta}} + K)^{-1} S_{\boldsymbol{\theta}}$ and $\mathbf{m} = R S_{\boldsymbol{\theta}}^{-1}\mathbf{g}$. After decomposing $R = DD^{\mathrm{T}}$, the sampling of $\boldsymbol{\theta}$ is then conditioned on the "whitened" variables $\boldsymbol{\eta}$, defined as $\mathbf{f} = D\boldsymbol{\eta} + \mathbf{m}$. The covariance $S_{\boldsymbol{\theta}}$ is constructed by matching the posterior distribution over each of the latent variables individually (see [22] for further details). Figure 1 shows that the SURR parameterization is characterized by a conditional posterior over $\boldsymbol{\theta}$ larger than SA and slightly larger than the AA parameterization.

3.3 Drawing samples from $p(\boldsymbol{\theta}|\mathbf{y})$: the Pseudo Marginal approach

The use of reparameterization techniques mitigates the problems due to the coupling of latent variables and hyper-parameters, but sampling efficiency for GP models is still an issue (for example, [7] reports simulations of ten parallel chains comprising five millions samples each). Intuitively, the best strategy to break the correlation between latent variables and hyper-parameters in sampling from the posterior over the hyper-parameters would be to integrate out the latent variables altogether. As we discussed, this is not possible, but here we present a strategy that uses an unbiased estimate of the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ to devise an MCMC strategy that produces samples from the correct posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$. For the sake of clarity, in this work we will focus on the Metropolis-Hastings algorithm with proposal $\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})$. We are interested in sampling from the posterior distribution

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{21}$$

In order to do that, we would need to integrate out the latent variables:

$$p(\mathbf{y}|\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})d\mathbf{f} \tag{22}$$

and use this along with the prior $p(\boldsymbol{\theta})$ in the Hastings ratio:

$$z = \frac{p(\mathbf{y}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})} \frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})} \tag{23}$$

As already discussed, analytically integrating out $\mathbf{f}$ is not possible.

The results in [25], [24] show that we can plug into the Hastings ratio an estimate $\tilde{p}(\mathbf{y}|\boldsymbol{\theta})$ of the marginal $p(\mathbf{y}|\boldsymbol{\theta})$, and as long as this is unbiased, then the sampler will draw samples from the correct posterior $p(\boldsymbol{\theta}|\mathbf{y})$.

$$\tilde{z} = \frac{\tilde{p}(\mathbf{y}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{\tilde{p}(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})} \frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})} \tag{24}$$

This result is remarkable as it gives a simple recipe to be used in hierarchical models to tackle the problem of strong coupling between groups of variables when using MCMC algorithms.

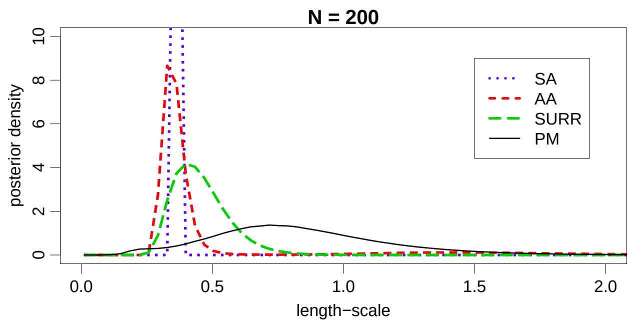

Fig. 1. Comparison of the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ with the posterior $p(\boldsymbol{\theta}|\mathbf{f})$ in the SA parameterization, the posterior $p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\nu})$ in the AA parameterization, and the parameterization used in the SURR method.

Figure 1 shows the effect of conditioning the sampling of $\boldsymbol{\theta}$ on different transformations of the latent variables given by SA (blue line), AA (red line), and SURR (green line). The conditional variance for the three approaches is still way lower than the variance of the marginal posterior $p(\boldsymbol{\theta}|\mathbf{y})$ that can be obtained by the PM approach. This motivates the use of the PM approach to effectively break the correlation between latent variables and hyper-parameters in an MCMC scheme.

Note that if the goal is quantifying uncertainty in the parameters only, and no predictions are needed, one could just iterate the sampling of $\boldsymbol{\theta}|\mathbf{y}$, as this is done regardless of $\mathbf{f}$. For predictions, instead, samples from the joint posterior $p(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y})$ are needed in the Monte Carlo integral in equation 20, so both steps are necessary. We consider this as a Gibbs sampler despite the fact that in principle interleaving of the two steps is not needed; one could obtain samples from the posterior distribution over $\mathbf{f}$ in a second stage, once samples from $p(\boldsymbol{\theta}|\mathbf{y})$ are available. This would come at an extra cost given that sampling $\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}$ requires the factorization of K for each MCMC sample $\boldsymbol{\theta}$. Therefore, when predictions are needed, we prefer to interleave the two steps, and still interpret the proposed sampling strategy as a Gibbs sampler.

3.3.1 Unbiased estimation of $p(\mathbf{y}|\boldsymbol{\theta})$ using importance sampling

In order to obtain an unbiased estimator $\tilde{p}(\mathbf{y}|\boldsymbol{\theta})$ for the marginal $p(\mathbf{y}|\boldsymbol{\theta})$, we propose to employ importance sampling. We draw N_{imp} samples $\mathbf{f}_i$ from the approximating distribution $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$, so that we can approximate the marginal $p(\mathbf{y}|\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})d\mathbf{f}$ by:

$$\tilde{p}(\mathbf{y}|\boldsymbol{\theta}) \simeq \frac{1}{N_{\text{imp}}} \sum_{i=1}^{N_{\text{imp}}} \frac{p(\mathbf{y}|\mathbf{f}_i)p(\mathbf{f}_i|\boldsymbol{\theta})}{q(\mathbf{f}_i|\mathbf{y}, \boldsymbol{\theta})} \tag{25}$$

It is easy to verify that equation 25 yields an unbiased estimate of $p(\mathbf{y}|\boldsymbol{\theta})$, as its expectation is the exact marginal $p(\mathbf{y}|\boldsymbol{\theta})$. Therefore, this estimate can be used in the Hastings ratio to construct an MCMC approach that samples from the correct invariant distribution $p(\boldsymbol{\theta}|\mathbf{y})$. Algorithm 1 sketches the MH algorithm that we propose to sample the hyper-parameters.

Algorithm 1 Pseudo-marginal MH transition operator to sample $\boldsymbol{\theta}$.

```
Input: The current pair $(\theta, \tilde{p}(\mathbf{y}|\boldsymbol{\theta}))$, a routine to approximate $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ by $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$, and number of importance samples $N_{\text{imp}}$
Output: A new pair $(\theta, \tilde{p}(\mathbf{y}|\boldsymbol{\theta}))$
1: Draw $\boldsymbol{\theta}'$ from the proposal distribution $\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})$
2: Approximate $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$ by $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$
3: Draw $N_{\text{imp}}$ samples from $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$
4: Compute $\tilde{p}(\mathbf{y}|\boldsymbol{\theta}')$ using eq. 25
5: Compute $A = \min\left\{1, \frac{\tilde{p}(\mathbf{y}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{\tilde{p}(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})} \frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right\}$
6: Draw $u$ from $U_{[0,1]}$
7: if $A > u$ then return $(\boldsymbol{\theta}', \tilde{p}(\mathbf{y}|\boldsymbol{\theta}'))$
8: else return $(\boldsymbol{\theta}, \tilde{p}(\mathbf{y}|\boldsymbol{\theta}))$
```

From the theory of importance sampling [38], the variance of the estimator is zero when $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ is proportional to $p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})$, which is proportional to the posterior distribution over $\mathbf{f}$ that we do not know how to sample from in the first place. In our case, the more accurate the Gaussian approximation the smaller the variance of the estimator. Given that EP has been reported to be more accurate in approximating $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$, it is reasonable to expect that EP will lead to a smaller estimator variance compared to LA. This will be assessed in the next section.

The motivation for using an importance sampling estimator rather than other simulation based methods for estimating marginal likelihoods, is the following. Even though it is possible to sample $\mathbf{f}$ relatively efficiently, the estimation of marginal likelihoods from MCMC simulations is generally challenging [39], [40] and only guarantees of estimator consistency are available. Obtaining estimates based on samples from $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ would require some form of user intervention (assessment of convergence and estimation of efficiency) every time a new value of $\boldsymbol{\theta}$ is proposed; this is clearly not practical or useful for the PM scheme. This work reports an extensive assessment of LA and EP to obtain Gaussian approximations to $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ within the importance sampling estimate of $p(\mathbf{y}|\boldsymbol{\theta})$.

3.3.2 *Analysis of correctness*

We show here why the proposed method yields an MCMC approach that produces samples from the correct invariant distribution $p(\boldsymbol{\theta}|\mathbf{y})$. The easiest way to see this is by considering $N_{\text{imp}} = 1$; showing correctness for larger numbers of importance samples is similar but notationally heavier (see [25] for further details). By substituting the importance sampling estimate $\tilde{p}(\mathbf{y}|\boldsymbol{\theta})$ with $N_{\text{imp}} = 1$ into $\tilde{z}$ and rearranging the terms, we obtain

$$\tilde{z} = \frac{p(\mathbf{y}|\mathbf{f}')p(\mathbf{f}'|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})p(\boldsymbol{\theta})} \times \left[\frac{q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})}{q(\mathbf{f}'|\mathbf{y}, \boldsymbol{\theta}')} \frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right] \tag{26}$$

Isolating the terms in the squared bracket allows us to interpret $\tilde{z}$ as a Hastings ratio with a joint proposal for $\boldsymbol{\theta}$ and for the importance sample $\mathbf{f}$ given by

$$\pi(\mathbf{f}', \boldsymbol{\theta}'|\mathbf{f}, \boldsymbol{\theta}) = q(\mathbf{f}'|\mathbf{y}, \boldsymbol{\theta}')\pi(\boldsymbol{\theta}'|\boldsymbol{\theta}). \tag{27}$$

The remaining term in $\tilde{z}$ indicates that the target distribution this approach is sampling from is

$$p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta}). \tag{28}$$

If we concentrate on $\boldsymbol{\theta}$, regardless of $\mathbf{f}$, the target distribution is exactly what we are aiming to sample from, as it is proportional to the posterior $p(\boldsymbol{\theta}|\mathbf{y})$. The extension to more than one importance sample follows from a similar argument, except that the approximating density $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ appears in the expression of the target distribution; however, this does not cause any problems as marginalizing latent variables leads to the same conclusion as in the case $N_{\text{imp}} = 1$. The analysis for $N_{\text{imp}} = 1$ also reveals an interesting similarity with the approach proposed in [23], where a joint update of $\boldsymbol{\theta}$ and $\mathbf{f}$ was performed as follows: proposing $\boldsymbol{\theta}'|\boldsymbol{\theta}$, proposing $\mathbf{f}'|\mathbf{y}, \boldsymbol{\theta}'$, and accepting/rejecting the joint proposal $\boldsymbol{\theta}', \mathbf{f}'$. However in this case the PM transition kernel will still target the desired marginal posterior irrespective of the value of importance samples N_{imp}.

4 Assessing importance distributions

In this section, we present simulations to assess the ability of the PM approach to characterize the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ in GP classification. First, we aim to assess the quality of the estimate given by the importance sampler based on LA and EP on simulated data with respect to the number of importance samples. Second, we will evaluate the efficiency of the sampler on simulated data with respect to the approximation used to draw importance samples and with respect to their number. Third, we will compare the PM approach with the AA and SURR parameterizations that are the most efficient sampling schemes proposed in the literature for sampling hyper-parameters in models involving GP priors [21]. In all the experiments, in both LA and EP we imposed a convergence criterion on the change in squared norm of $\mathbf{f}$ being less than $n/10^4$.

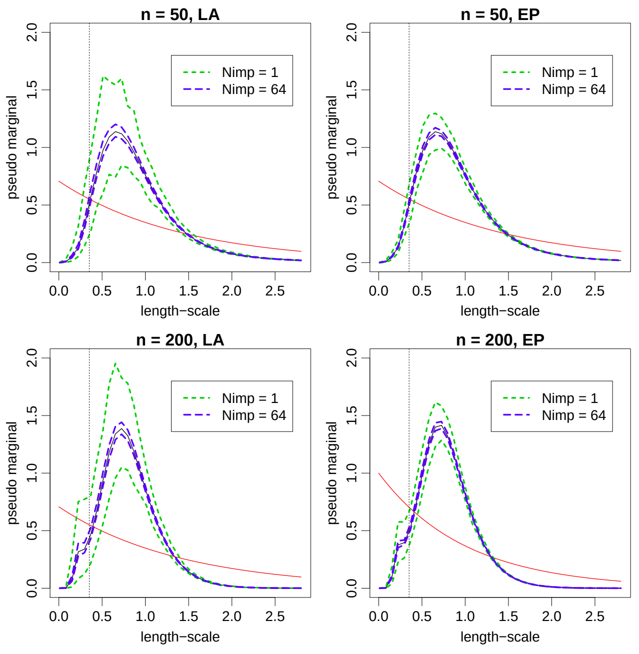

Fig. 2. Plot of the PM as a function of the length-scale τ; black solid lines represent the average over 500 repetitions and dashed lines represent 2.5th and 97.5th quantiles for $N_{\text{imp}} = 1$ and $N_{\text{imp}} = 64$. The solid red line is the prior density.

4.1 Analysis of the variance of the estimator

In this section, we present an assessment of the variance of the estimator $\tilde{p}(\boldsymbol{\theta}|\mathbf{y})$ with respect to the global length-scale parameter τ in equation 5. In particular, we are interested in comparing the quality of the approximation given by the importance sampling approach based on the two different approximations employed in this work.

Given that the dimensionality of the space where the approximation is performed grows linearly with the number of input vectors n, we are also interested in studying the effect of n on the quality of the approximation. Based on these considerations, we simulated data from the GP classification model with $d = 2$ and $n = 50$ and $n = 200$, with an isotropic covariance function with $\tau = 0.35$ and $\sigma = 2.08$. We fixed the value of σ to the value used to generate the data, and we imposed a Gamma prior on the length-scale $p(\tau) = \mathcal{G}(\tau|a_\tau, b_\tau)$ with shape $a_\tau = 1$ and rate $b_\tau = 1/\sqrt{d}$. We then computed the posterior over $\boldsymbol{\theta}$ based on $\tilde{p}(\mathbf{y}|\boldsymbol{\theta})$ for different values of τ and over 500 repetitions, with different number of importance samples N_{imp}. The results are reported in figure 2.

As expected, for larger sets of importance samples, the estimates are more accurate. We can also see that EP leads to a smaller variance compared to LA, which is not surprising given that the approximation achieved by EP is more accurate [18], [19]. The experiment suggests that there is little increase in the variance of the estimator for the larger data set.

4.2 Effect of the pseudo marginal on the efficiency of the sampler

In this section we report an analysis on simulated data showing how the choice of the approximation and the number of importance samples affect the efficiency in sampling from $p(\boldsymbol{\theta}|\mathbf{y})$. We generated data sets from the GP classification model with different combinations of number of input vectors and number of covariates. The covariates were generated in the unit hypercube and data were selected to have an equal number of input vectors in each class. We chose Gamma priors for the hyper-parameters as follows: $p(\tau_i) = \mathcal{G}(\tau_i|a_\tau, b_\tau)$ with shape $a_\tau = 1$ and rate $b_\tau = 1/\sqrt{d}$, and $p(\sigma) = \mathcal{G}(\sigma|a_\sigma, b_\sigma)$ with shape $a_\sigma = 1.2$ and rate $b_\sigma = 0.2$. In the formulation of the GP classification model, all hyper-parameters have to be positive; for the sake of convenience, we reparameterized them introducing the variables $\psi_{\tau_i} = \log(\tau_i)$ and $\psi_\sigma = \log(\sigma)$.

For each method, we ran 10 parallel chains for 5000 burn-in iterations followed by 10000 iterations; convergence speed of the samplers was monitored using the Potential Scale Reduction Factor (PSRF) ($\hat{R}$ statistics) as described in [41]. The chains were initialized from the prior, rather than using the procedure suggested in [41] to make the convergence test more challenging. Also, correctness of the code was checked by using the idea presented in [42], that indirectly shows that the Markov chains have indeed $p(\boldsymbol{\theta}|\mathbf{y})$ as their invariant distribution.

The proposal mechanism $\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})$ was the same for all the PM approaches for a given combination of n and d, so that it is meaningful to analyze the effect of N_{imp} on sampling efficiency, convergence speed, and acceptance rate. In particular, a large variance for the estimator of the marginal likelihood can eventually lead to the acceptance of $\boldsymbol{\theta}$ because $p(\mathbf{y}|\boldsymbol{\theta})$ is largely overestimated leading to a difficulty for the chain to move away from there. In this case, the chain can get stuck and take several iterations before moving again; this effect has been reported in [24], [25]. To isolate the effect of N_{imp} and the type of approximation on sampling efficiency and acceptance rate, we tuned the chains using preliminary runs for EP and $N_{\text{imp}} = 64$ to achieve about 25% acceptance rate and used the same proposal for LA and other values of N_{imp}.

The results are reported in table 1 for isotropic and ARD covariances. As a measure of efficiency, we used the minimum Effective Sample Size (ESS) [43] across the hyper-parameters. The tables also report the median of $\hat{R}$ achieved by the chains at different iterations, namely 1000, 2000, 5000, and 10000. This gives an idea of the convergence as the iterations progress. Finally, in table 1 we report the acceptance rate; a low acceptance rate compared to the one obtained by PM EP (64) indicates that the chains are more likely to get stuck due to a large variance of the estimator of the marginal likelihood.

The results indicate that sampling efficiency when employing EP to approximate the posterior distribution

TABLE 1
Analysis of convergence and efficiency of a MH algorithm sampling the hyper-parameters using the PM approach. The results show the dependency of the effective sample size (ESS) and speed of convergence (measured through the $\hat{R}$ statistics after $1e3$, $2e3$, $5e3$, and $1e4$ iterations) with respect to the type of approximation (LA or EP) and the number of importance samples used to compute an unbiased estimate of the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta})$.

			Isotropic						ARD					
n	d	Scheme	ESS	$\hat{R}$	$\hat{R}$	$\hat{R}$	$\hat{R}$	Acc	ESS	$\hat{R}$	$\hat{R}$	$\hat{R}$	$\hat{R}$	Acc
		(N_{imp})		$1e3$	$2e3$	$5e3$	$1e4$	rate		$1e3$	$2e3$	$5e3$	$1e4$	rate
50	2	PM LA (1)	749 (73)	1.00	1.00	1.00	1.00	23.9 (0.4)	131 (41)	1.04	1.05	1.04	1.02	13.0 (3.3)
		PM LA (16)	778 (61)	1.00	1.00	1.00	1.00	25.1 (0.4)	206 (21)	1.05	1.04	1.03	1.01	16.7 (1.2)
		PM LA (64)	752 (76)	1.00	1.00	1.00	1.00	24.8 (0.5)	212 (38)	1.03	1.03	1.02	1.01	16.6 (1.9)
		PM EP (1)	747 (100)	1.00	1.00	1.00	1.00	24.4 (0.5)	208 (22)	1.01	1.00	1.01	1.00	16.1 (1.0)
		PM EP (16)	736 (163)	1.00	1.00	1.00	1.00	25.0 (0.6)	246 (23)	1.00	1.00	1.00	1.00	19.2 (0.7)
		PM EP (64)	793 (54)	1.00	1.00	1.00	1.00	24.9 (0.5)	252 (24)	1.00	1.00	1.00	1.00	19.9 (0.8)
		AA	287 (58)	1.00	1.00	1.00	1.00	22.7 (0.9)	20 (9)	1.12	1.12	1.05	1.03	20.3 (2.6)
		SURR	154 (11)	1.01	1.00	1.00	1.00	22.4 (1.0)	21 (6)	1.11	1.08	1.07	1.05	21.6 (3.3)
50	10	PM LA (1)	237 (147)	1.25	1.26	1.19	1.08	18.2 (4.6)	19 (12)	1.08	1.09	1.13	1.17	7.1 (4.4)
		PM LA (16)	238 (194)	1.01	1.02	1.03	1.02	16.9 (6.0)	62 (30)	1.04	1.04	1.06	1.11	17.6 (4.9)
		PM LA (64)	348 (126)	1.01	1.01	1.01	1.01	21.8 (2.3)	78 (23)	1.03	1.03	1.01	1.01	20.2 (2.8)
		PM EP (1)	282 (47)	1.02	1.01	1.01	1.00	19.2 (1.4)	63 (18)	1.03	1.03	1.02	1.01	15.6 (1.2)
		PM EP (16)	507 (36)	1.00	1.00	1.00	1.00	24.7 (1.0)	107 (12)	1.01	1.01	1.01	1.01	24.2 (0.5)
		PM EP (64)	583 (51)	1.00	1.00	1.00	1.00	26.1 (0.6)	108 (28)	1.02	1.01	1.01	1.01	26.3 (0.7)
		AA	71 (23)	1.02	1.01	1.02	1.01	22.8 (2.5)	74 (6)	1.03	1.03	1.02	1.02	22.4 (2.7)
		SURR	52 (15)	1.04	1.02	1.02	1.02	21.8 (2.9)	45 (5)	1.03	1.03	1.02	1.01	21.7 (1.7)
200	2	PM LA (1)	717 (31)	1.00	1.00	1.00	1.00	31.7 (0.5)	318 (29)	1.00	1.00	1.00	1.00	38.3 (0.8)
		PM LA (16)	739 (46)	1.00	1.00	1.00	1.00	32.6 (0.5)	364 (20)	1.00	1.00	1.00	1.00	43.1 (0.5)
		PM LA (64)	730 (26)	1.00	1.00	1.00	1.00	32.6 (0.6)	355 (42)	1.00	1.00	1.00	1.00	43.6 (0.7)
		PM EP (1)	736 (47)	1.00	1.00	1.00	1.00	32.8 (0.5)	349 (26)	1.00	1.00	1.00	1.00	42.0 (0.4)
		PM EP (16)	736 (48)	1.00	1.00	1.00	1.00	32.7 (0.4)	365 (21)	1.00	1.00	1.00	1.00	43.5 (0.5)
		PM EP (64)	721 (43)	1.00	1.00	1.00	1.00	32.6 (0.7)	354 (37)	1.00	1.00	1.00	1.00	43.9 (0.6)
		AA	112 (49)	1.01	1.01	1.01	1.01	21.6 (1.0)	41 (8)	1.07	1.07	1.06	1.04	23.0 (2.3)
		SURR	54 (8)	1.01	1.02	1.01	1.01	21.5 (1.5)	61 (9)	1.02	1.02	1.01	1.01	22.0 (1.9)
200	10	PM LA (1)	115 (53)	1.03	1.03	1.02	1.04	27.1 (11.3)	27 (10)	1.39	1.33	1.19	1.11	12.3 (2.9)
		PM LA (16)	117 (42)	1.05	1.04	1.03	1.03	26.8 (9.0)	53 (18)	1.08	1.06	1.03	1.02	18.0 (1.9)
		PM LA (64)	145 (62)	1.01	1.01	1.01	1.02	28.7 (8.0)	37 (31)	1.14	1.15	1.17	1.17	13.2 (8.6)
		PM EP (1)	75 (35)	1.03	1.03	1.03	1.02	18.2 (3.7)	26 (14)	1.12	1.09	1.13	1.08	10.3 (3.1)
		PM EP (16)	137 (38)	1.01	1.01	1.01	1.01	27.6 (4.2)	52 (13)	1.20	1.15	1.08	1.04	17.1 (2.1)
		PM EP (64)	130 (64)	1.09	1.09	1.09	1.12	25.5 (10.0)	45 (21)	1.03	1.03	1.03	1.09	17.4 (3.4)
		AA	26 (6)	1.05	1.04	1.03	1.03	23.7 (1.9)	22 (7)	1.10	1.08	1.05	1.03	21.6 (5.5)
		SURR	18 (7)	1.27	1.28	1.24	1.16	21.5 (8.6)	10 (3)	1.07	1.08	1.06	1.05	20.3 (3.9)

over $\mathbf{f}$ is higher than when employing the LA algorithm. It is striking to see that evaluating the PM with as little as one importance sample seems already able to offer acceptable performance in terms of ESS compared to larger values of N_{imp}. However, a low acceptance rate when N_{imp} is small suggests that the corresponding chains can take several iterations before accepting any proposals.

4.3 Comparison with reparameterization techniques

Table 1 also reports a comparison of the PM method with the AA and SURR sampling schemes with a Metropolis-Hastings transition operator so that results are meaningfully comparable. The proposal mechanism was tuned during the burn-in phase to achieve about 25% acceptance rate. Table 1 shows that the PM approach achieves faster convergence and higher sampling efficiency compared to the AA scheme. The SURR method has comparable convergence behavior and efficiency compared to the AA scheme. This is somehow different from what presented in [22], where a component-wise slice sampling transition operator was employed. In [22], the SURR method achieved higher efficiency per covariance construction, likelihood evaluation and running time compared to the AA method. In the experiment reported here ESS is comparable.

Table 2 reports the average number of operations in $O(n^3)$ needed for one iteration of the PM approach with LA and EP approximations and the AA and SURR parameterizations. The table shows that EP is more expensive than the LA algorithm, and that the PM approaches require more $O(n^3)$ operations compared to AA and SURR. Normalization of the ESS by the number of operations suggests that the cost of obtaining independent samples using the PM approach is generally better than AA and SURR. In the PM approach we also tried stopping the approximation algorithms using a looser convergence criterion (results not reported); especially for $N_{\text{imp}} = 64$ this yielded similar efficiency with much lower computational cost.

We conducted a further test of convergence with the aim of mitigating the effect of the random walk exploration and highlighting the advantages offered by different parameterizations. We ran the AA, SURR and PM approaches by repeating each step of the Gibbs

TABLE 3
Analysis of convergence and efficiency of the AA and SURR parameterizations compared to the PM approach. Each of the Gibbs sampling updates was repeated 20 times in order to highlight the effect of the parameterization alone. The PM approach is based on the EP approximation and $N_{\text{imp}} = 64$. The column next to the one reporting the ESS shows the ESS normalized by the number of operations in $O(n^3)$.

			Isotropic						ARD					
n	d	Scheme	ESS	ESS $/O(n^3)$	$\hat{R}$ $1e3$	$\hat{R}$ $2e3$	$\hat{R}$ $5e3$	$\hat{R}$ $1e4$	ESS	ESS $/O(n^3)$	$\hat{R}$ $1e3$	$\hat{R}$ $2e3$	$\hat{R}$ $5e3$	$\hat{R}$ $1e4$
50	2	PM	9057 (243)	556 (15)	1.00	1.00	1.00	1.00	4301 (169)	243 (10)	1.00	1.00	1.00	1.00
		AA	1571 (378)	1571 (378)	1.00	1.00	1.00	1.00	57 (11)	57 (11)	1.01	1.01	1.01	1.01
		SURR	702 (98)	234 (33)	1.00	1.00	1.00	1.00	64 (9)	21 (3)	1.02	1.02	1.02	1.01
50	10	PM	7478 (333)	575 (26)	1.00	1.00	1.00	1.00	2207 (141)	161 (10)	1.00	1.00	1.00	1.00
		AA	211 (22)	211 (22)	1.01	1.00	1.00	1.00	464 (40)	464 (40)	1.00	1.00	1.00	1.00
		SURR	133 (23)	44 (8)	1.01	1.01	1.01	1.00	207 (18)	69 (6)	1.01	1.01	1.01	1.00
200	2	PM	8850 (301)	571 (19)	1.00	1.00	1.00	1.00	5844 (166)	338 (10)	1.00	1.00	1.00	1.00
		AA	327 (142)	327 (142)	1.01	1.00	1.01	1.00	143 (22)	143 (22)	1.01	1.01	1.00	1.00
		SURR	234 (24)	78 (8)	1.00	1.00	1.00	1.00	339 (47)	113 (16)	1.00	1.00	1.00	1.00
200	10	PM	737 (533)	55(40)	1.01	1.01	1.01	1.01	363 (250)	28 (19)	1.03	1.03	1.01	1.01
		AA	37 (7)	37 (7)	1.05	1.05	1.05	1.03	42 (8)	42 (8)	1.03	1.03	1.02	1.01
		SURR	25 (6)	8 (2)	1.05	1.05	1.01	1.00	22 (5)	7 (2)	1.09	1.07	1.05	1.03

TABLE 2
Average number of operations in $O(n^3)$ required for each iteration of the PM approaches with the LA and EP approximations and for each iteration in the AA and SURR parameterizations.

n	d	Scheme	Isotropic	ARD
50	2	PM LA	8.2 (0.2)	8.8 (0.2)
		PM EP	16.3 (0.0)	17.7 (0.1)
		AA	1.0 (0.0)	1.0 (0.0)
		SURR	3.0 (0.0)	3.0 (0.0)
50	10	PM LA	7.8 (0.3)	8.3 (0.2)
		PM EP	13.0 (0.1)	13.7 (0.1)
		AA	1.0 (0.0)	1.0 (0.0)
		SURR	3.0 (0.0)	3.0 (0.0)
200	2	PM LA	8.7 (0.0)	9.3 (0.0)
		PM EP	15.5 (0.0)	17.3 (0.0)
		AA	1.0 (0.0)	1.0 (0.0)
		SURR	3.0 (0.0)	3.0 (0.0)
200	10	PM LA	7.6 (0.2)	7.9 (0.2)
		PM EP	13.3 (0.1)	13.2 (0.2)
		AA	1.0 (0.0)	1.0 (0.0)
		SURR	3.0 (0.0)	3.0 (0.0)

sampler 20 times, so that we can roughly consider the new sample drawn in a Gibbs sampling updates independent with respect to the previous. The behavior of the $\hat{R}$ statistics with respect to the number of iterations, reported in table 3, reveals some interesting features. All methods are characterized by fast convergence. In the case of the SURR and AA methods, however, efficiency is much lower than what can be achieved by the PM method when repeating the Gibbs sampling update $\boldsymbol{\theta}|\mathbf{y}$ 20 times. This is an indication that the parameterizations of SURR and AA methods are not fully capable of breaking the correlation between hyper-parameters and latent variables. Finally, note that in the case of ARD covariances, the low efficiency in all methods is due to the random walk type of exploration; in those cases 20 iterations of the Gibbs sampling steps were not enough to ensure independence between consecutive samples.

4.4 Convergence speed and efficiency on real data

This section reports a comparison of classification performance on three UCI data sets [44], so as to verify the capability of the proposed approach to effectively carry out inference for parameters in general GP classification problems. The Breast and Pima data sets are two-class classification problems, described by $d = 9$ and $d = 8$ covariates, comprising $n = 682$ and $n = 768$ input vectors respectively. The Abalone data set has three classes; in this paper, we considered the task of inferring parameters of a GP probit classifier for the two classes "M" and "F", resulting in a data set of $n = 2835$ input vectors and $d = 8$ covariates. All covariates were transformed to have zero mean and unit standard deviation. We selected the isotropic covariance function in equation 5 and chose the following priors for their parameters: $p(\tau) = \mathcal{G}(\tau|a_\tau, b_\tau)$ with shape $a_\tau = 1$ and rate $b_\tau = 1/\sqrt{d}$, and $p(\sigma) = \mathcal{G}(\sigma|a_\sigma, b_\sigma)$ with shape $a_\sigma = 1.1$ and rate $b_\sigma = 0.1$.

We compared the proposed sampling PM approach with the AA and SURR parameterizations. In the latter two, we alternated the sampling of $\boldsymbol{\theta}$ using the MH algorithm and the sampling of $\mathbf{f}$ iterating ELL-SS ten times. In the PM approach we selected an approximation based on the LA algorithm and chose the number of importance samples to be $N_{\text{imp}} = 1$.

We ran chains for 12000 iterations, where the first 2000 were used to tune the proposal mechanisms to achieve an acceptance rate between 20% and 30%. In the case of the PM approach, in the adaptive phase we used the approximate marginal likelihood obtained by the LA algorithm. This was to overcome the problems that may arise when chains get trapped due to a largely overestimated value of the marginal likelihood. After 2000 iterations, we then switched to the estimate of the marginal likelihood obtained by importance sampling. For each sampling approach, we ran 10 parallel chains initialized from the prior, so that we could compare convergence speed.

The results for the three UCI data sets are reported in figures 3 and 4. The plots show efficiency and convergence speed in the sampling of the logarithm of the length-scale parameter τ. The left and middle panels of these figures show the trace and the auto-correlation plots of one chain after the burn-in period of 2000 iterations. The auto-correlation plot gives an indication of efficiency; the faster the auto-correlation of a chain reaches values close to zero, the higher the efficiency of the corresponding MCMC approach. In order to facilitate visualization, the traces were thinned by a factor of 10 and the auto-correlation plots were computed on the thinned chains. The right panel shows the evolution of the PSRF ($\hat{R}$) after the burn-in period without thinning the chains.

The results indicate that the SURR parameterization yields better performance compared to the AA parameterization, where convergence can be extremely slow. The PM approach achieves impressive efficiency and convergence speed compared to the AA and SURR parameterizations. Remarkably, in these experiments, this is achieved using only one importance sample and a relatively cheap approximation based on the LA algorithm. We note here that this might not be the case in general. The key to the success of the PM approach is the possibility to obtain a low-variance estimator for the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta})$. Some approximations might not be good enough to ensure that this is the case; in such situations, more importance samples or better approximations should be employed. This has recently been investigated in more detail in [45], where unbiased estimates of the marginal likelihood based on Annealed Importance Sampling [46] have been proposed.

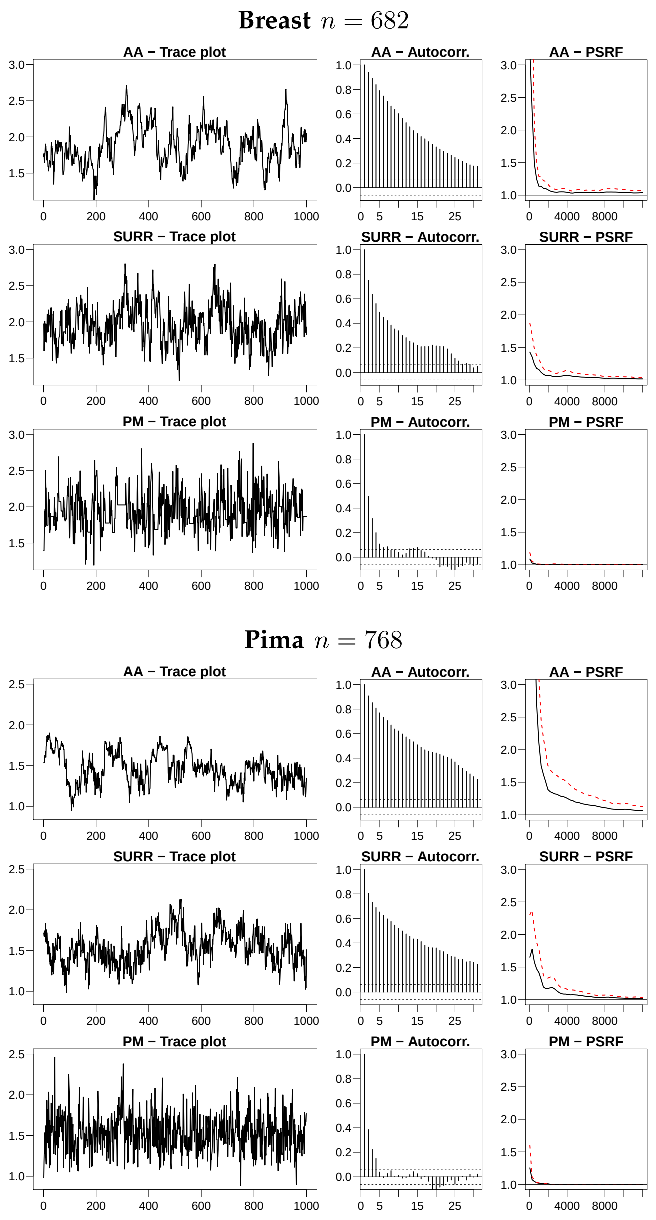

Fig. 3. Summary of efficiency and convergence speed on Breast and Pima data sets for the AA and SURR parameterization and the proposed PM approach. All plots show the sampling of the logarithm of the length-scale parameter τ. The figure shows trace plots (left panels) and corresponding auto-correlation plots (middle panels) for one chain thinned by a factor of 10 after burn-in. The right panel reports the evolution of the PSRF after burn-in; in this plot the solid line and the red dashed line represent the median and the 97.5% percentile respectively.

5 Comparison with other probabilistic classifiers

5.1 Data sets and experimental setup

This section reports a comparison of classification performance on five UCI data sets. In the Glass data set we considered the two classes “window glass” and “non-window glass”. In the USPS data set we considered the task of classification of “3” vs “5” as in [18].

We constructed increasingly larger training sets comprising an equal number of input vectors per class. For each value of number of input vectors n, we constructed 40 training sets across which we evaluated the performance of the proposed PM approach.

5.2 Comparing methods

We compared the PM approach (MCMC PM EP) with (i) a probabilistic version of an SVM classifier [28] (ii) the GP classifier using the EP approximation [18], [5] optimizing $\boldsymbol{\theta}$ using type II Maximum Likelihood [5], [12] (EP ML) and (iii) with the classifier obtained by sampling $\boldsymbol{\theta}$ based on the marginal likelihood computed by EP (MCMC EP). Predictions in EP ML and MCMC EP were carried out by approximately integrating out latent variables according to the Gaussian approximation given

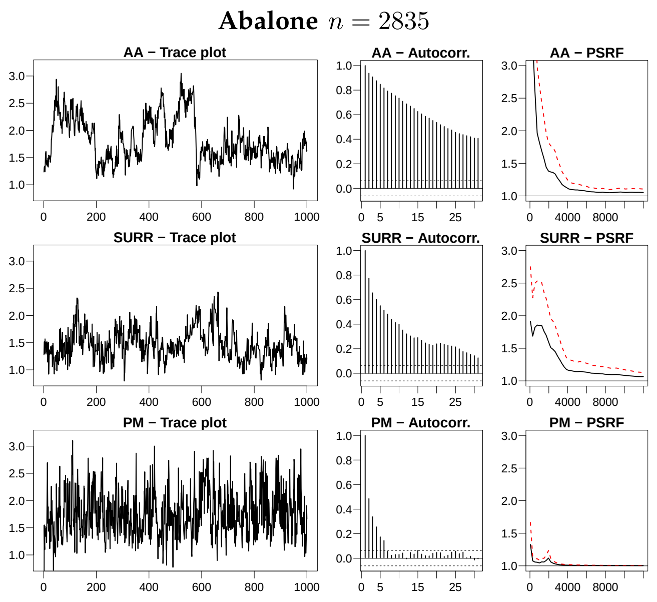

Fig. 4. Summary of efficiency and convergence speed on the Abalone data set.

by the EP algorithm.

We used the SVM code in the `e1071` R package, which provides a front-end to the `LIBSVM` library where non-linear SVMs employ a squared exponential isotropic kernel. In order to meaningfully compare the four classifiers, we used the isotropic kernel/covariance for all of them. In the MCMC PM EP method we set $N_{\text{imp}} = 64$.

5.3 Performance scores

We are interested in comparing the ability of the classifiers to reliably quantify uncertainty in predicting class labels. Predictive probabilities give a means to do so; the more confident the classifier is about a correct class label, the better. Also, predictive probabilities make it possible to avoid making decisions when predictive probabilities are below a given threshold.

Following [47], we propose to summarize the ability of a classifier to reliably quantify uncertainty in predictions by analyzing classification accuracy and AUC (which denotes the area under the Receiver Operating Characteristic (ROC) curve for the classifier) versus the degree of abstention. In particular, we propose to measure the area under the two curves obtained by plotting accuracy versus degree of abstention and AUC versus degree of abstention. We will denote such scores by “capacity accuracy” and “capacity AUC”, respectively. A value of capacity close to one suggests that a classifier is capable of correctly classifying test data with a high degree of confidence.

For a probabilistic classifier, we compute accuracy and AUC versus degree of abstention as follows. Denote by ρ a threshold for the predictive probabilities. According to a threshold value ρ, we compute the degree of abstention as the proportion of test data for which the predictive probability p satisfies the following condition $0.5 - \rho < p < 0.5 + \rho$. For the rest of the test data, namely data for which the classifier is most confident, we compute accuracy and AUC. We repeat this procedure for different values of ρ, starting from 0.00 going up to 0.50 at increments of 0.01, so that we obtain the plots of accuracy and AUC with respect to degree of abstention. We finally compute the area of the two curves to obtain “capacity accuracy” and “capacity AUC” for the classifier. Given that the degree of abstention might not reach the value of one, we propose to divide the area of the curves by the largest degree of abstention, so that the two capacity scores are normalized to one. Figure 5 shows two exemplar curves of accuracy and AUC with respect to the degree of abstention that are used to compute the capacity scores.

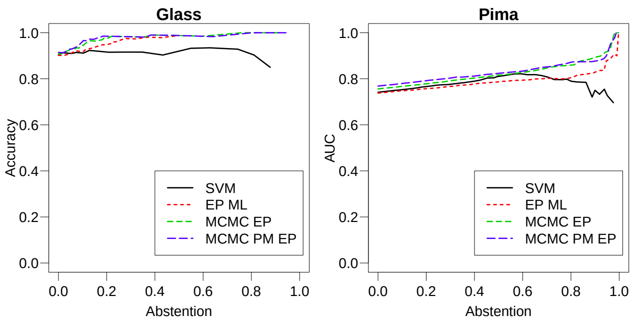

Fig. 5. Left panel: Plot of accuracy vs degree of abstention for one of the folds in the Glass data set for $n = 50$. Right panel: Plot of AUC vs degree of abstention for one of the folds in the Pima data set for $n = 20$.

5.4 Results

The results are reported in figures 6 and 7 for the five data sets considered in this work. In general, the probabilistic version of the SVM classifier leads to a worse quantification of uncertainty compared to the GP classifiers. Figure 5 shows how SVMs tend to assign high confidence to wrong decisions more often than GP classifiers. Also, the GP classifier optimizing the hyper-parameters (EP ML) yields worse performance compared to the GP classifiers where hyper-parameters are integrated out. Finally, the general trend is that the PM approach is the one achieving the best quantification of uncertainty compared to all the classifiers considered in this work.

A closer look at the posterior distribution obtained by MCMC EP and MCMC PM EP reveals the following insights. In the case of classification, the quality of the approximation of the marginal likelihood given by EP is generally accurate enough that the posterior distribution over the hyper-parameters for the MCMC PM EP approach is similar to the one obtained by employing MCMC EP. Differences in predictions obtained by the two methods are mostly due to the different way latent

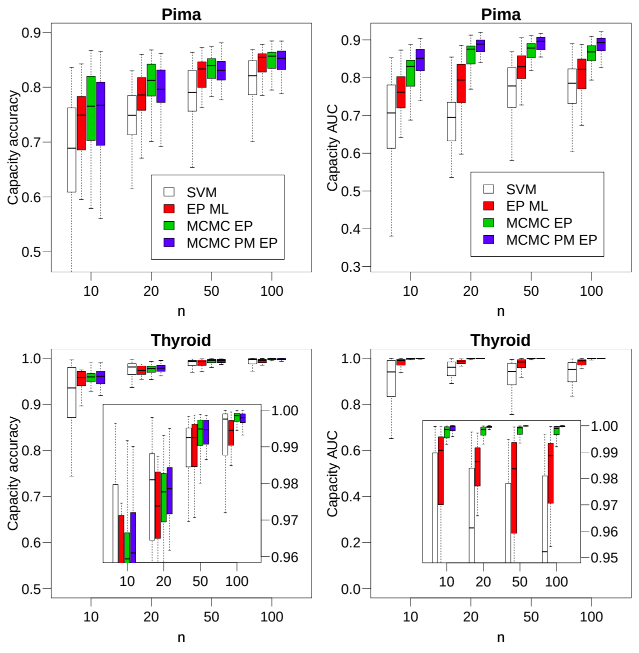

Fig. 6. Plots of performance scores with respect to size of training set for the Pima (first row) and the Thyroid (second row) data sets. The legend is reported in the first row only and it applies to all the plots. In the remaining plots, a closeup is reported to make it easier to compare the results.

variables are integrated out when making predictions. The situation can be different for other GP models where, for example, EP cannot be derived or EP exhibits convergence issues, so those considerations are peculiar to GP classification. The aim of this study is to demonstrate that it is important to account for the uncertainty in the hyper-parameters when predicting labels on unseen data in order to achieve a sound quantification of uncertainty. Although this has been pointed out in previous work [12], [26], [16], [27], the proposed approach makes exact Bayesian computations, in a Monte Carlo sense, actually feasible by building upon deterministic approximations.

In terms of complexity of the three GP classifiers, the following considerations can be made. All the methods employ EP that requires three $O(n^3)$ operations at each iteration. In the case of ML, the approximation of the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ given by EP is optimized with respect to $\boldsymbol{\theta}$. When the number of hyper-parameters is small, as in the cases of the isotropic RBF covariance function considered here, the optimization can be performed by grid search. In the case of the ARD covariance, optimization can be performed, for instance, by employing the conjugate gradient algorithm, that requires the derivatives of the approximate marginal likelihood with respect to the hyper-parameters. Computing such derivatives involves an extra $O(n^3)$ operation (see section 5.5.2 of [5]). In MCMC EP the approximation of the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ given by EP is used

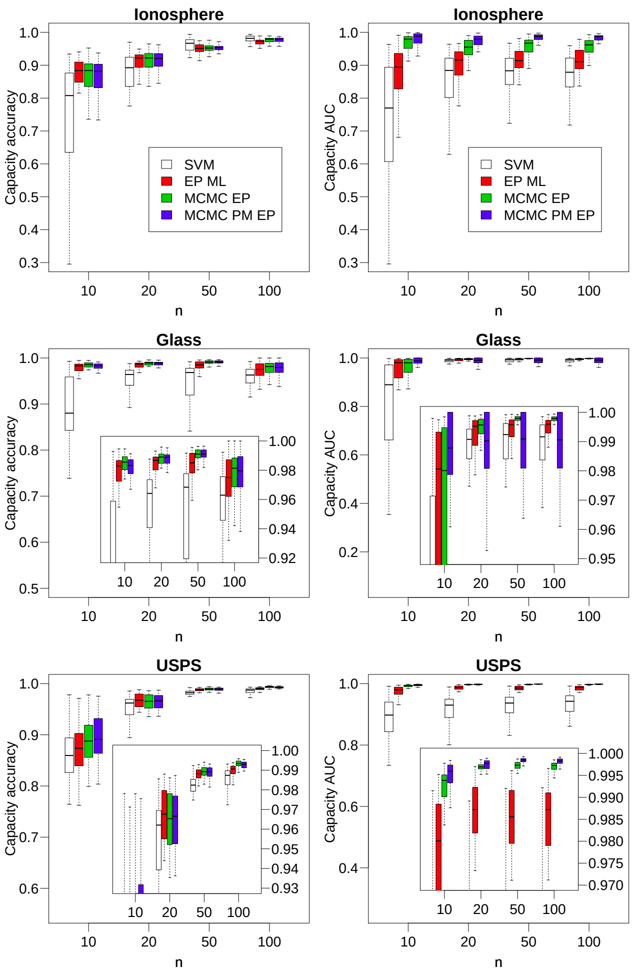

Fig. 7. Plots of performance scores with respect to size of training set for the Ionosphere (first row), the Glass (second row) and the USPS (third row) data sets. The legend is reported in the first row only and it applies to all the plots. In the remaining plots, a closeup is reported to make it easier to compare the results.

directly to obtain samples form $p(\boldsymbol{\theta}|\mathbf{y})$. In this case, each iteration requires running EP to obtain an approximation to $p(\mathbf{y}|\boldsymbol{\theta})$, so the overall complexity is still in $O(n^3)$, but the number of operations depends on the number of the iterations the MCMC approach is run for. Running MCMC PM EP requires exactly the same number of operations as MCMC EP, except that the Cholesky decomposition of the covariance of the approximating Gaussian is needed to draw the importance samples, adding an extra $O(n^3)$ operation.

To compute the mean of the predictive distribution for one test sample, EP ML requires only operations in $O(n^2)$ and none in $O(n^3)$, as the expensive elements needed to compute it are already available from running the EP approximation. In the case of MCMC EP, the mean of the predictive distribution is an average of means computed for several draws from the posterior over $\boldsymbol{\theta}$, so the complexity scales linearly with the number of MCMC samples and quadratically with n. MCMC PM EP, instead, requires samples from the posterior distribution over latent variables in addition to hyper-parameters. Drawing samples from $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ requires computations in $O(n^2)$ as previously discussed. For each sample from the posterior distribution over latent variables and hyper-parameters, all the other computations are again in $O(n^2)$ following similar arguments as in the case of EP ML; therefore, similarly to MCMC EP, the complexity scales linearly with the number of MCMC samples and quadratically with n.

6 Conclusions

This paper presented a methodology that enables the fully Bayesian treatment of GP models, using probit regression as a working example, and builds upon existing approximate methods for integrating out latent variables. The key element in this paper is the adoption of the pseudo marginal approach to devise a correct MCMC sampling scheme for the hyper-parameters of the covariance of the Gaussian Process prior from an approximation of the marginal density of the observation given the hyper-parameters. The resulting sampling scheme is simple, and it is based on approximate methods that are currently very popular.

The results indicate that the proposed methodology leads to an MCMC approach where chains are characterized by high convergence speed and high efficiency. This is an important feature that yields a step forward toward making fully Bayesian inference based on Gaussian Processes a concrete possibility for many applications with little user intervention. The overall efficiency is driven by the MH proposal for the hyper-parameters that can be inefficient for models with several hyper-parameters; a matter of current investigation is to study alternative proposal mechanisms that avoid the erratic behavior of random walk exploration.

In support vector based classifiers hyper-parameters are optimized by minimizing the cross-validation error across a set of candidate values. It is clear that for small data sets or for covariance functions with a large number of hyper-parameters, this procedure becomes unfeasible. The proposed approach, instead, yields a natural way to integrate the uncertainty in the hyper-parameters when making predictions and infer them from data. The comparison with other state-of-the-art probabilistic classifiers that are commonly employed in the Machine Learning community shows that accounting for the posterior over the hyper-parameters is extremely beneficial, especially for small data sets.

In terms of scalability, the main computational bottleneck is in the computation of the GP prior density that requires the factorization of the covariance matrix, which is in $O(n^3)$. The same considerations apply to all GP classifiers that use approximate methods to integrate out latent variables, so we argue that by running an efficient sampling procedure for the hyper-parameters rather than an optimization strategy, the computational overhead will not be dramatically higher, but the classification performance will be much more reliable.

We believe that the results presented here can be extended to other latent Gaussian models, such as Log-Gaussian Cox process models [48] and Ordinal Regression with GP priors [49]. Finally, it is possible to extend the proposed PM MCMC approach to deal with GP models characterized by a sparse inverse covariance, which are popular when analyzing spatio-temporal data. In this case, it is possible to exploit sparsity in the inverse covariance of the GP, yielding a fast mixing and efficient MCMC approach capable of processing large amounts of data.

Acknowledgments

The authors would like to thank the anonymous reviewers for their critical and constructive comments and suggestions. Mark Girolami is supported by an EPSRC Established Career Research Fellowship EP/J016934/2, a Royal Society Wolfson Research Merit Award, and EPSRC Project grants ENGAGE EP/K015664/2 and EQUIP EP/K034154/1. This work is dedicated to Stefano Filippone.

References

[1] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[2] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[3] C. E. Rasmussen and J. Q. Candela, "Healing the relevance vector machine through augmentation," in *Proceedings of the 22nd international conference on Machine learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 689–696.

[4] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

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

[6] A. Bosch, A. Zisserman, and X. Muoz, "Scene Classification Using a Hybrid Generative/Discriminative Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 712–727, 2008.

[7] M. Filippone, A. F. Marquand, C. R. V. Blain, S. C. R. Williams, J. Mourão-Miranda, and M. Girolami, "Probabilistic Prediction of Neurological Disorders with a Statistical Assessment of Neuroimaging Data Modalities," *Annals of Applied Statistics*, vol. 6, no. 4, pp. 1883–1905, 2012.

[8] T. Jaakkola, M. Diekhans, and D. Haussler, "A Discriminative Framework for Detecting Remote Protein Homologies," *Journal of Computational Biology*, vol. 7, no. 1-2, pp. 95–114, 2000.

[9] T. Joachims, "Text Categorization with Suport Vector Machines: Learning with Many Relevant Features," in *ECML*, ser. Lecture Notes in Computer Science, C. Nedellec and C. Rouveirol, Eds., vol. 1398. Springer, 1998, pp. 137–142.

[10] G. Rätsch, S. Sonnenburg, and C. Schäfer, "Learning Interpretable SVMs for Biological Sequence Classification," *BMC Bioinformatics*, vol. 7, no. S-1, 2006.

[11] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian learning for efficient visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1292–1304, Aug. 2005.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, Aug. 2007.

[13] C. K. I. Williams and D. Barber, "Bayesian classification with Gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1342–1351, 1998.

[14] T. P. Minka, "Expectation Propagation for approximate Bayesian inference," in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, ser. UAI '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 362–369.

[15] M. N. Gibbs and D. J. C. MacKay, "Variational Gaussian process classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 11, no. 6, pp. 1458–1464, 2000.

[16] 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 (Statistical Methodology)*, vol. 71, no. 2, pp. 319–392, 2009.

[17] M. Opper and O. Winther, "Gaussian processes for classification: Mean-field algorithms," *Neural Computation*, vol. 12, no. 11, pp. 2655–2684, 2000.

[18] M. Kuss and C. E. Rasmussen, "Assessing Approximate Inference for Binary Gaussian Process Classification," *Journal of Machine Learning Research*, vol. 6, pp. 1679–1704, 2005.

[19] H. Nickisch and C. E. Rasmussen, "Approximations for Binary Gaussian Process Classification," *Journal of Machine Learning Research*, vol. 9, pp. 2035–2078, Oct. 2008.

[20] B. Cseke and T. Heskes, "Approximate Marginals in Latent Gaussian Models," *Journal of Machine Learning Research*, vol. 12, pp. 417–454, 2011.

[21] M. Filippone, M. Zhong, and M. Girolami, "A comparative evaluation of stochastic-based inference methods for Gaussian process models," *Machine Learning*, vol. 93, no. 1, pp. 93–114, 2013.

[22] I. Murray and R. P. Adams, "Slice sampling covariance hyperparameters of latent Gaussian models," in *NIPS*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, 2010, pp. 1732–1740.

[23] L. Knorr-Held and H. Rue, "On Block Updating in Markov Random Field Models for Disease Mapping," *Scandinavian Journal of Statistics*, vol. 29, no. 4, pp. 597–614, Dec. 2002.

[24] C. Andrieu and G. O. Roberts, "The pseudo-marginal approach for efficient Monte Carlo computations," *The Annals of Statistics*, vol. 37, no. 2, pp. 697–725, Apr. 2009.

[25] M. A. Beaumont, "Estimation of Population Growth or Decline in Genetically Monitored Populations," *Genetics*, vol. 164, no. 3, pp. 1139–1160, Jul. 2003.

[26] R. M. Neal, "Regression and classification using Gaussian process priors (with discussion)," *Bayesian Statistics*, vol. 6, pp. 475–501, 1999.

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

[28] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 2011.

[29] D. J. C. Mackay, "Bayesian methods for backpropagation networks," in *Models of Neural Networks III*, E. Domany, J. L. van Hemmen, and K. Schulten, Eds. Springer, 1994, ch. 6, pp. 211–254.

[30] L. Tierney and J. B. Kadane, "Accurate Approximations for Posterior Moments and Marginal Densities," *Journal of the American Statistical Association*, vol. 81, no. 393, pp. 82–86, 1986.

[31] I. Murray, R. P. Adams, and D. J. C. MacKay, "Elliptical slice sampling," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 541–548, 2010.

[32] R. M. Neal, "Slice Sampling," *Annals of Statistics*, vol. 31, pp. 705–767, 2003.

[33] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[34] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," Dept. of Computer Science, University of Toronto, Tech. Rep. CRG-TR-93-1, Sep. 1993.

[35] M. Girolami and B. Calderhead, "Riemann manifold Langevin and Hamiltonian Monte Carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 2, pp. 123–214, Mar. 2011.

[36] Y. Yu and X.-L. Meng, "To Center or Not to Center: That Is Not the Question–An Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency," *Journal of Computational and Graphical Statistics*, vol. 20, no. 3, pp. 531–570, 2011.

[37] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld, "A general framework for the parametrization of hierarchical models," *Statistical Science*, vol. 22, no. 1, pp. 59–73, 2007.

[38] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[39] N. Friel and A. N. Pettitt, "Marginal likelihood estimation via power posteriors," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 3, pp. 589–607, 2008.

[40] J. Skilling, "Nested sampling for general Bayesian computation," *Bayesian Analysis*, vol. 1, no. 4, pp. 833–860, 2006.

[41] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical Science*, vol. 7, no. 4, pp. 457–472, 1992.

[42] J. Geweke, "Getting it right: joint distribution tests of posterior simulators," *Journal of the American Statistical Association*, vol. 99, no. 467, pp. 799–804, 2004.

[43] W. R. Gilks and D. J. Spiegelhalter, *Markov chain Monte Carlo in practice*. Chapman & Hall/CRC, 1996.

[44] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007.

[45] M. Filippone, "Bayesian inference for Gaussian process classifiers with annealing and exact-approximate MCMC," 2013, arXiv:1311.7320.

[46] R. M. Neal, "Annealed importance sampling," *Statistics and Computing*, vol. 11, no. 2, pp. 125–139, Apr. 2001.

[47] C. Ferri and J. Hernández-Orallo, "Cautious Classifiers," in *ROCAI*, J. Hernández-Orallo, C. Ferri, N. Lachiche, and P. A. Flach, Eds., 2004, pp. 27–36.

[48] J. Møller, A. R. Syversveen, and R. P. Waagepetersen, "Log Gaussian Cox Processes," *Scandinavian Journal of Statistics*, vol. 25, no. 3, pp. 451–482, 1998.

[49] W. Chu and Z. Ghahramani, "Gaussian Processes for Ordinal Regression," *Journal of Machine Learning Research*, vol. 6, pp. 1019–1041, Dec. 2005.

Maurizio Filippone Maurizio Filippone received a Master's degree in Physics and a Ph.D. in Computer Science from the University of Genova, Italy, in 2004 and 2008, respectively.

In 2007, he was a Research Scholar with George Mason University, Fairfax, VA. From 2008 to 2011, he was a Research Associate with the University of Sheffield, U.K. (2008-2009), with the University of Glasgow, U.K. (2010), and with University College London, U.K (2011). He is currently a Lecturer with the University of Glasgow, U.K. His current research interests include statistical methods for pattern recognition.

Dr Filippone serves as an Associate Editor for Pattern Recognition and the IEEE Transactions on Neural Networks and Learning Systems.

Mark Girolami Mark Girolami FRSE is Professor of Statistics in the Department of Statistics at the University of Warwick. He also holds a professorial post in the Department of Computer Science at Warwick and is Director of the EPSRC funded Research Network on Computational Statistics and Machine Learning. He is currently Editor-in-Chief of the journal Statistics and Computing, an Associate Editor for J. R. Statist. Soc. C, Journal of Computational and Graphical Statistics and Area Editor for Pattern Recognition Letters. He currently holds a Royal Society Wolfson Research Merit Award and an EPSRC Established Career Research Fellowship.

Chapter 3

Large-Scale Gaussian process learning

In this part of the thesis I present my work on the development of scalable approximate inference methods for GPs, keeping accurate quantification of uncertainty as a desirable property. The starting point of this line of work is based on the core idea behind the Pseudo-Marginal MCMC work for GPs with non-Gaussian likelihoods [16]. In this work, we showed how to leverage unbiased computations of the likelihood to obtain samples from the posterior distribution over covariance parameters without introducing any bias. Taking the same idea one step further, it is possible to think that fast approximate computations, provided that they satisfy some properties, are enough to carry out "exact" inference. This is the idea underpinning the paper [15], later developed to carry out stochastic gradient-based optimization of covariance parameters using preconditioning techniques [10]. IN parallel, we also explored various alternative directions to carry out approximations for GP models that enable scaling to large-scale supervised learning problems [20, 9, 25].

Resources and grants: These activities involved my time, the time of a number of students, and that of a number of collaborators. The works [15, 10] involved my time as well as that of two students; for the former, I was helped by a Master's student, while the latter was led by one of my Ph.D. students, and is in collaboration with Oxford and Columbia universities. The work [20] involved my time in a collaboration with Cambridge and Lancaster universities (one PhD student from Cambridge and two academics). The work [9] is led by one of my Ph.D. students, and is in collaboration with another colleague within the Department of Data Science at EURECOM and the University of New South Wales, Australia. Most of the funding for these activities came from a three-year EPSRC project and the AXA Chair.

- **Preconditioning Kernel Matrices** and **Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE)** Instead of approximating the marginal likelihood in an unbiased fashion as in [16], these works are based on the development of an unbiased estimator of the gradient of the log-marginal likelihood. In GPs, working with the gradient of the log-marginal likelihood rather than the log-marginal likelihood itself has some important advantages. The log-marginal likelihood of GP models involves the computation of the log-determinant of the (generally dense) $n \times n$ covariance matrix $\mathbf{C} = \mathbf{K} + \lambda \mathbf{I}$, where n is the number of data, as well as the solution of linear systems of this kind $\mathbf{C}^{-1}\mathbf{v}$ with $\mathbf{v} \in \mathcal{R}^n$. The standard way to proceed is to compute determinants and inverses based on a factorization of the matrix $\mathbf{C}$, which costs $O(n^3)$ operations; storing the matrix $\mathbf{C}$ and its factorization requires $O(n^2)$ space. By employing stochastic linear algebra techniques, it is possible to obtain an unbiased version of the gradient of the log-likelihood relying exclusively on the solution of linear systems involving $\mathbf{C}$. The motivation for employing SGLD for inference of GP covariance parameters comes from inspecting the gradient of the log-marginal likelihood that has components $g_i = \frac{\partial \log[p(\mathbf{y}|\boldsymbol{\theta},X)]}{\partial \theta_i}$:

$$g_i = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) + \mathbf{y}^{\top}\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y} \tag{3.1}$$

Computing the g_i's requires again $O(n^3)$ operations due to the trace term and the linear system $\mathbf{C}^{-1}\mathbf{y}$. 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 [19], obtaining:

$$\tilde{g}_i = -\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)\top}\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{r}^{(i)} + \mathbf{y}^{\top}\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y} \tag{3.2}$$

Given that $\mathrm{E}(\mathbf{r}^{(i)}\mathbf{r}^{(i)\top}) = I$, we can readily verify that

$$\mathrm{E}[\mathbf{r}^{(i)\top}\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{r}^{(i)}] = \mathrm{Tr}\left[\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathrm{E}(\mathbf{r}^{(i)}\mathbf{r}^{(i)\top})\right]$$

which yields the trace term in eq. 3.1. Hence, in order to compute an 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. We investigated methods to solve such systems that avoid the necessity to store the matrix $\mathbf{C}$ and that are based on iterating matrix-vector products, such as Conjugate Gradient and Preconditioned Conjugate Gradient algorithms. In this case, the elements of $\mathbf{C}$ can be computed on the fly when needed in the computation of the matrix-vector multiplication. Matrix-vector multiplications cost $O(n^2)$ operations and are easily parallelizable.

Inference or optimization of covariance parameters $\boldsymbol{\theta}$ can then proceed by relying exclusively on stochastic gradients. For instance, it is possible to employ stochastic gradient-based MCMC samplers, such as the one in [39], and this is what is presented in [15]. In

[9], instead, we demonstrated the scalability of this proposal when optimizing covariance parameters using stochastic gradient-based optimization techniques. In particular, in [9] we showed how popular approximations to scale GP computations can effectively be used to precondition the linear systems involved in the calculation of stochastic gradients to significantly accelerate computations. In all, these works demonstrate the potential of these techniques to scale GP computations from $O(n^3)$ time and $O(n^2)$ space complexities to $O(n^2)$ and $O(n)$, respectively, while maintaining a sound quantification of uncertainty as the GP model is not approximated.

- **MCMC for Variationally Sparse Gaussian Processes** Another recent work that aims to scale nonparametric modeling to large datasets appears in [20]. In this work, we borrow ideas from the literature on scalable GPs using inducing points [38] and develop an MCMC-based inference algorithm that has many interesting properties. The proposed inference algorithm assumes that a limited set of cleverly positioned inducing points is sufficient to accurately model the available data, therefore making computations cheaper. This work differs from previous work on scalable GPs in the way it infers all variables in the model. In particular, we show that the optimal posterior distribution over the inducing points is a Dirac's delta, yielding a theoretical justification for the optimization of the position of the inducing points. Furthermore, we develop an efficient MCMC algorithm, based on Hybrid Monte Carlo, that is capable of drawing samples from the posterior distribution of all variables in the model, both latent variables and covariance parameters. Finally, we demonstrate in several applications that a limited number of inducing points and the proposed inference method yield an accurate quantification of uncertainty in predictions.

- **Random Feature Expansions for Deep Gaussian processes** Deep Gaussian Processes (DGPs) [11] are deep probabilistic models obtained by stacking multiple layers implemented through GPs. This construction to model composition of functions naturally resembles that of Deep Neural Networks (DNNs). The connection between DGPs and DNNs has been extensively investigated in the literature, and DGPs with a variety of kernel functions can be interpreted as DNNs with an infinite number of neurons at each layer and specific activation functions (see, e.g., [14, 31]). In contrast to DNNs, DGPs provide an elegant way to deal with the model-selection problem of determining a suitable number of neurons, as they are inherently nonparametric learning machines. Furthermore, they allow for a principled probabilistic framework to carry out learning of latent representations and covariance parameters.

  Although attractive from a theoretical point of view, learning DGPs poses some significant computational challenges that arguably hinder their application to a wider variety of problems. In contrast, DNNs have been extremely successful in areas such as computer vision because of their amenability to GPU and distributed computations, automatic differentiation tools, and mature developments of regularization techniques, such as low-rank weight representations and dropout [37]. In this work, we aimed to bridge the gap between DGPs and DNNs by showing how DGPs can be learned at scale by borrowing the key strengths of DNNs, while retaining a probabilistic formulation for accurate quantification of uncertainty. In particular, we show how random feature approximations of covariance functions applied to DGPs lead to DNNs with activation functions induced by the choice of the GP covariance function and low-rank weight matrices. We conveniently implemented stochastic variational inference [24] in TensorFlow [1] to infer the parameters of such approximate DGPs. Through extensive experimental comparisons, we demonstrate that our proposal significantly advances the state-of-the-art in inference of DGP models, and can be applied to millions of observations with moderately deep architectures.

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

Maurizio Filippone MAURIZIO.FILIPPONE@GLASGOW.AC.UK
School of Computing Science, University of Glasgow, UK

Raphael Engler RAPHAEL.ENGLER@WEB.DE
School of Computing Science, University of Glasgow, UK

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

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

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 samples 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

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

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 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:

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

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

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

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 optimization. The parameters ε_t are chosen to satisfy

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

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

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

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

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

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:

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

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

Algorithm 1 The Conjugate Gradient algorithm

```
  Input: data $X$, vector $\mathbf{b}$, convergence threshold $\epsilon$, initial vector $\mathbf{s}_0$, maximum number of iterations $T$
  $\mathbf{e}_0 = \mathbf{b} - K\mathbf{s}_0$;    $\mathbf{d}_0 = \mathbf{e}_0$;
  for $i = 0$ to $T$ do
    $\alpha_i = \frac{\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_i}{\mathbf{d}_i^{\mathrm{T}}K\mathbf{d}_i}$;
    $\mathbf{s}_{i+1} = \mathbf{s}_i + \alpha_i\mathbf{d}_i$;
    $\mathbf{e}_{i+1} = \mathbf{e}_i - \alpha_i K\mathbf{d}_i$;
    if $\|\mathbf{e}_{i+1}\| < \epsilon$ then
      return $\mathbf{s} = \mathbf{s}_{i+1}$;
    end if
    $\beta_i = \frac{\mathbf{e}_{i+1}^{\mathrm{T}}\mathbf{e}_{i+1}}{\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_i}$;
    $\mathbf{d}_{i+1} = \mathbf{e}_{i+1} + \beta_i\mathbf{d}_i$;
  end for
```

yields the trace term in eq. 8. Hence, in order to compute an 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

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

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

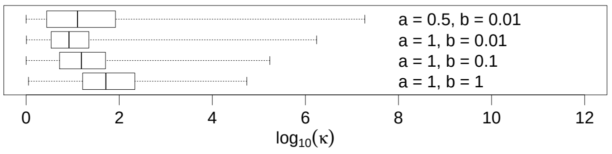

Figure 1. Distribution of the condition number κ of the covariance matrix for different choices of shape and rate parameters of a Gamma prior on each covariance parameter θ.

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

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

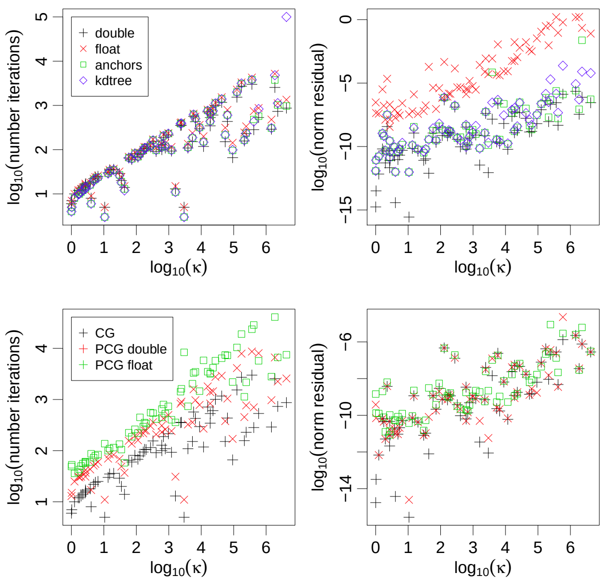

Figure 2. Top row: Comparison of fast CMVPs on number of iterations and error versus condition number. Bottom row: Comparison of the CG algorithm and two PCG algorithms using double and single precision CMVPs to solve inner linear systems.

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 different 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

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

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

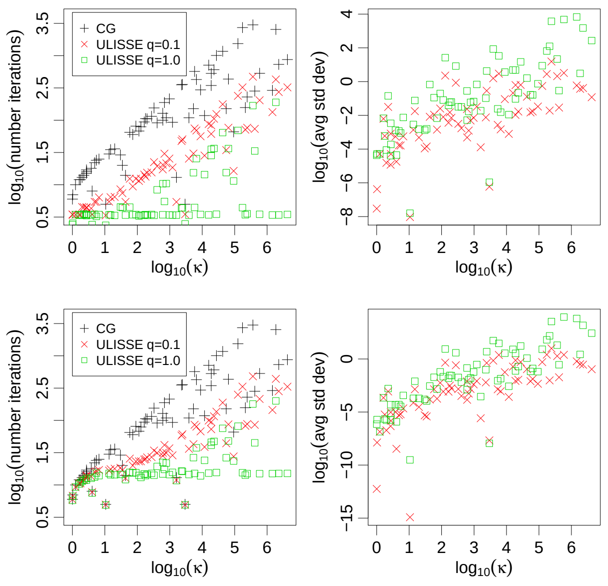

Figure 3. Comparison of the CG algorithm and ULISSE with early stop thresholds α calculated with $q = 0.1$ and $q = 1$ on number of iterations and standard deviation of the solution. The top row corresponds to $\beta = 1$ in the calculation of the weights w_r, whereas the bottom row corresponds to $\beta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

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

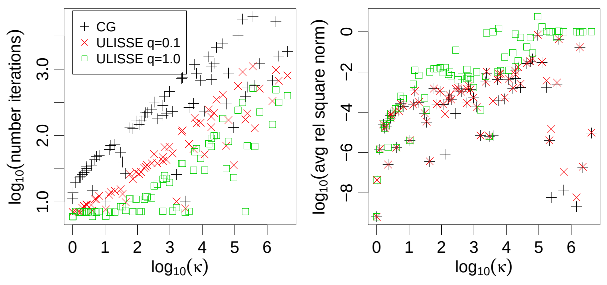

Figure 4. Comparison of the CG algorithm and ULISSE and early stop thresholds computed with $q = 0.1$ and $q = 1$ to estimate the gradient of the log-marginal likelihood in eq. 9. In ULISSE, the weights w_r are calculated with $\beta = 1$.

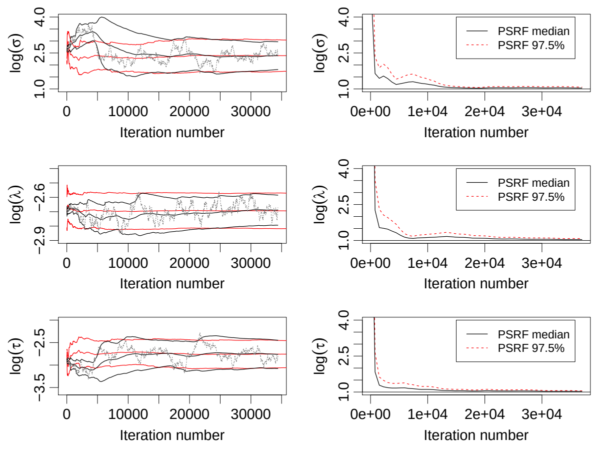

Figure 5. Concrete data ($n = 1030$) - Left panel: Comparison of MCMC (red) and SGLD with ULISSE (black) on running mean and plus/minus two standard deviations. The trace of one chain of SGLD is shown in gray. Right panel: Convergence analysis of SGLD with ULISSE with PSRF computed over ten chains.

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

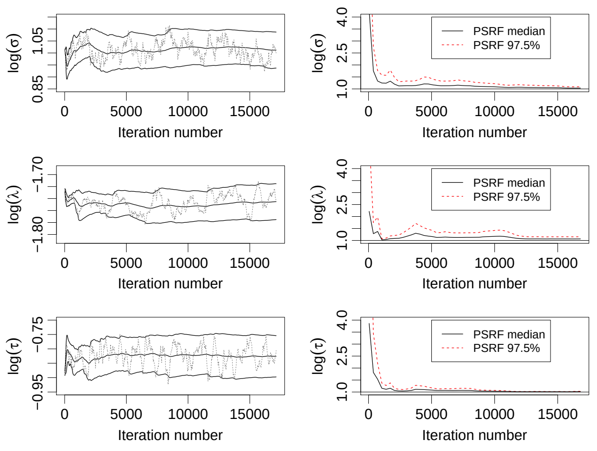

Figure 6. Census data ($n = 22,784$) - Left panel: Running mean and plus/minus two standard deviations (black) and trace of one chain of SGLD (gray). Right panel: Convergence analysis of SGLD with ULISSE reporting the PSRF computed over five chains.

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

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

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

MF gratefully acknowledges support from EPSRC grant EP/L020319/1.

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.

Preconditioning Kernel Matrices

Kurt Cutajar KURT.CUTAJAR@EURECOM.FR
EURECOM, Department of Data Science

Michael A. Osborne MOSB@ROBOTS.OX.AC.UK
University of Oxford, Department of Engineering Science

John P. Cunningham JPC2181@COLUMBIA.EDU
Columbia University, Department of Statistics

Maurizio Filippone MAURIZIO.FILIPPONE@EURECOM.FR
EURECOM, Department of Data Science

Abstract

The computational and storage complexity of kernel machines presents the primary barrier to their scaling to large, modern, datasets. A common way to tackle the scalability issue is to use the conjugate gradient algorithm, which relieves the constraints on both storage (the kernel matrix need not be stored) and computation (both stochastic gradients and parallelization can be used). Even so, conjugate gradient is not without its own issues: the conditioning of kernel matrices is often such that conjugate gradients will have poor convergence in practice. Preconditioning is a common approach to alleviating this issue. Here we propose preconditioned conjugate gradients for kernel machines, and develop a broad range of preconditioners particularly useful for kernel matrices. We describe a scalable approach to both solving kernel machines and learning their hyperparameters. We show this approach is exact in the limit of iterations and outperforms state-of-the-art approximations for a given computational budget.

1. Introduction

Kernel machines, in enabling flexible feature space representations of data, comprise a broad and important class of tools throughout machine learning and statistics; prominent examples include support vector machines (Schölkopf & Smola, 2001) and Gaussian processes (GPs) (Rasmussen & Williams, 2006). At the core of most kernel machines is the need to solve linear systems involving the Gram matrix $K = \{k(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta})\}_{i,j=1,\ldots,n}$, where the kernel function k, parameterized by $\boldsymbol{\theta}$, implicitly specifies the feature space representation of data points $\mathbf{x}_i$. Because K grows with the number of data points n, a fundamental computational bottleneck exists: storing K is $\mathcal{O}(n^2)$, and solving a linear system with K is $\mathcal{O}(n^3)$. As the need for large-scale kernel machines grows, much work has been directed towards this scaling issue.

Standard approaches to kernel machines involve a factorization (typically Cholesky) of K, which is efficient and exact but maintains the quadratic storage and cubic runtime costs. This cost is particularly acute when adapting (or learning) hyperparameters $\boldsymbol{\theta}$ of the kernel function, as K must then be factorized afresh for each $\boldsymbol{\theta}$. To alleviate this burden, numerous works have turned to approximate methods (Candela & Rasmussen, 2005; Snelson & Ghahramani, 2007; Rahimi & Recht, 2008) or methods that exploit structure in the kernel (Gilboa et al., 2015). Approximate methods can achieve attractive scaling, often through the use of low-rank approximations to K, but they can incur a potentially severe loss of accuracy. An alternative to factorization is found in the conjugate gradient method (CG), which is used to directly solve linear systems via a sequence of matrix-vector products. Any kernel structure can then be exploited to enable fast multiplications, driving similarly attractive runtime improvements, and eliminating the storage burden (neither K nor its factors need be represented in memory). Unfortunately, in the absence of special structure that accelerates multiplications, CG performs no better than $\mathcal{O}(n^3)$ in the worst case, and in practice finite numerical precision often results in a degradation of run-

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

time performance compared to factorization approaches.

Throughout optimization, the typical approach to the slow convergence of CG is to apply preconditioners to improve the geometry of the linear system being solved (Golub & Van Loan, 1996). While preconditioning has been explored in domains such as spatial statistics (Chen, 2005; Stein et al., 2012; Ashby & Falgout, 1996), the application of preconditioning to kernel matrices in machine learning has received little attention. Here we design and study preconditioned conjugate gradient methods (PCG) for use in kernel machines, and provide a full exploration of the use of approximations of K as preconditioners.

Our contributions are as follows. (i) Extending the work in (Davies, 2014), we apply a broad range of kernel matrix approximations as preconditioners. Interestingly, this step allows us to exploit the important developments of *approximate* kernel machines to accelerate the *exact* computation that PCG offers. (ii) As a motivating example used throughout the paper, we analyze and provide a general framework to both learn kernel parameters and make predictions in GPs. (iii) We extend stochastic gradient learning for GPs (Filippone & Engler, 2015; Anitescu et al., 2012) to allow any likelihood that factorizes over the data points by developing an unbiased estimate of the gradient of the approximate log-marginal likelihood. We demonstrate this contribution in making the first use of PCG for GP classification. (iv) We evaluate datasets over a range of problem size and dimensionality. Because PCG is exact in the limit of iterations (unlike approximate techniques), we demonstrate a tradeoff between accuracy and computational effort that improves beyond state-of-the-art approximation and factorization approaches.

In all, we show that PCG, with a thoughtful choice of preconditioner, is a competitive strategy which is possibly even superior than existing approximation and CG-based techniques for solving general kernel machines[1].

2. Motivating example – Gaussian Processes

Gaussian processes (GPs) are the fundamental building block of many probabilistic kernel machines that can be applied in a large variety of modeling scenarios (Rasmussen & Williams, 2006). Throughout the paper, we will denote by $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ a set of n input vectors and use $\mathbf{y} = (y_1, \dots, y_n)^\top$ for the corresponding labels. GPs are formally defined as collections of random variables characterized by the property that any finite number of them is jointly Gaussian distributed. The specification of a kernel function determines the covariance structure of such random variables

$$\mathrm{cov}\big(f(\mathbf{x}), f(\mathbf{x}')\big) = k(\mathbf{x}, \mathbf{x}' \mid \boldsymbol{\theta}).$$

In this work we focus in particular on the popular Radial Basis Function (RBF) kernel

$$k(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta}) = \sigma^2 \exp\left[-\frac{1}{2}\sum_{r=1}^{d}\frac{(\mathbf{x}_i - \mathbf{x}_j)_r^2}{l_r^2}\right], \qquad (1)$$

where $\boldsymbol{\theta}$ represents the collection of the kernel parameters σ^2 and l_r^2. Defining $f_i = f(\mathbf{x}_i)$ and $\mathbf{f} = (f_1, \dots, f_n)^\top$, and assuming a zero mean GP, we have

$$\mathbf{f} \sim \mathcal{N}(\mathbf{f} \mid \mathbf{0}, K),$$

where K is the $n \times n$ Gram matrix with elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta})$. Note that the kernel above and many popular kernels in machine learning give rise to dense kernel matrices. Observations are then modeled through a transformation h of a set of GP-distributed latent variables, specifying the model

$$y_i \sim p\big(y_i \mid h(f_i)\big), \qquad \mathbf{f} \sim \mathcal{N}(\mathbf{f} \mid \mathbf{0}, K).$$

2.1. The need for preconditioning

The success of nonparametric models based on kernels hinges on the adaptation of kernel parameters $\boldsymbol{\theta}$. The motivation for preconditioning begins with an inspection of the log-marginal likelihood of GP models with prior $\mathcal{N}(\mathbf{f} \mid \mathbf{0}, K)$. In Gaussian processes with a Gaussian likelihood $y_i \sim \mathcal{N}(y_i \mid f_i, \lambda)$, we have analytic forms for

$$\log[p(\mathbf{y} \mid \boldsymbol{\theta}, X)] = -\frac{1}{2}\log\left(|K_{\mathbf{y}}|\right) - \frac{1}{2}\mathbf{y}^\top K_{\mathbf{y}}^{-1}\mathbf{y} + \mathrm{const},$$

and its derivatives with respect to kernel parameters θ_i,

$$g_i = -\frac{1}{2}\mathrm{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{y}^\top K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial \theta_i}K_{\mathbf{y}}^{-1}\mathbf{y}. \quad (2)$$

where $K_{\mathbf{y}} = K + \lambda I$. The traditional approach involves factorizing the kernel matrix $K_{\mathbf{y}} = LL^\top$ using the Cholesky algorithm (Golub & Van Loan, 1996) which costs $\mathcal{O}(n^3)$ operations. After that, all other operations cost $\mathcal{O}(n^2)$ except for the trace term in the calculation of g_i which once again requires $\mathcal{O}(n^3)$ operations. Similar computations are required for computing mean and variance predictions for test data (Rasmussen & Williams, 2006). Note that the solution of a linear system is required for computing the variance at every test point.

This approach is not viable for large n and, consequently, many approaches have been proposed to approximate these computations, thus leading to approximate optimal values for $\boldsymbol{\theta}$ and approximate predictions. Here we investigate the

[1]Code to replicate all results in this paper is available at http://github.com/mauriziofilippone/preconditioned_GPs

possibility of avoiding approximations altogether, by arguing that for parameter optimization it is sufficient to obtain an unbiased estimate of the gradient g_i. In particular, when such an estimate is available, it is possible to employ stochastic gradient optimization that has strong theoretical guarantees (Robbins & Monro, 1951). In the case of GPs, the problematic terms in eq. 2 are the solution of the linear system $K_{\mathbf{y}}^{-1}\mathbf{y}$ and the trace term. In this work we make use of a stochastic linear algebra result that allows for an approximation of the trace term,

$$\operatorname{Tr}\left(K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial\theta_i}\right) \approx \frac{1}{N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)\top} K_{\mathbf{y}}^{-1}\frac{\partial K_{\mathbf{y}}}{\partial\theta_i}\mathbf{r}^{(i)},$$

where the $N_{\mathbf{r}}$ vectors $\mathbf{r}^{(i)}$ have components drawn from $\{-1, 1\}$ with probability $1/2$. Verifying that this is an unbiased estimate of the trace term is straightforward considering that $\mathrm{E}\left(\mathbf{r}^{(i)}\mathbf{r}^{(i)\top}\right) = I$ (Gibbs, 1997).

This result shows that all it takes to calculate stochastic gradients is the ability to efficiently solve linear systems. Linear systems can be iteratively solved using *conjugate gradient* (CG) (Golub & Van Loan, 1996). The advantage of this formulation is that we can attempt to optimize kernel parameters using stochastic gradient optimization without having to store $K_{\mathbf{y}}$ and, given that the most expensive operation is now multiplying the kernel matrix by vectors, only $\mathcal{O}(n^2)$ computations are required. However, it is well known that the convergence of the CG algorithm depends on the condition number $\kappa(K_{\mathbf{y}})$ (ratio of largest to smallest eigenvalues), so the suitability of this approach may also be curtailed if $K_{\mathbf{y}}$ is badly conditioned. To this end, a well-known approach for improving the conditioning of a matrix, which in turn accelerates convergence, is *preconditioning*. This necessitates the introduction of a preconditioning matrix, P, which should be chosen in such a way that $P^{-1}K_{\mathbf{y}}$ approximates the identity matrix, I. Intuitively, this can be obtained by setting $P = K_{\mathbf{y}}$; however, given that in Preconditioned CG (PCG) we are required to solve linear systems involving P, this choice would be no easier than solving the original system. Thus we must choose P which approximates $K_{\mathbf{y}}$ as closely as possible, but which can also be easily inverted. The PCG algorithm is shown in Algorithm 1.

Algorithm 1 The Preconditioned CG Algorithm, adapted from (Golub & Van Loan, 1996)

```
Require: data $X$, vector $\mathbf{v}$, convergence threshold $\epsilon$, initial
  vector $\mathbf{x}_0$, maximum no. of iterations $T$
  $\mathbf{r}_0 = \mathbf{v} - K_{\mathbf{y}}\mathbf{x}_0$;  $\mathbf{z}_0 = P^{-1}\mathbf{r}_0$;  $\mathbf{p}_0 = \mathbf{z}_0$
  for i = 0 : $T$ do
    $\alpha_i = \frac{\mathbf{r}_i^T\mathbf{z}_i}{\mathbf{r}_i^T K_{\mathbf{y}}\mathbf{z}_i}$
    $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\mathbf{p}_i$
    $\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha_i K_{\mathbf{y}}\mathbf{p}_i$
    if $\|\mathbf{r}_{i+1}\| < \epsilon$ then
      return $\mathbf{x} = \mathbf{x}_{i+1}$
    end if
    $\mathbf{z}_{i+1} = P^{-1}\mathbf{r}_{i+1}$
    $\beta_i = \frac{\mathbf{r}_{i+1}^T\mathbf{r}_{i+1}}{\mathbf{r}_i^T\mathbf{r}_i}$
    $\mathbf{p}_{i+1} = \mathbf{p}_{i+1} + \beta_i\mathbf{p}_i$
  end for
```

2.2. Non-Gaussian Likelihoods

When the likelihood $p(y_i \mid f_i)$ is not Gaussian, it is no longer possible to analytically integrate out latent variables. Instead, techniques such as Gaussian approximations (see, e.g., (Kuss & Rasmussen, 2005; Nickisch & Rasmussen, 2008)) and methods attempting to characterize the full posterior $p(\mathbf{f}, \boldsymbol{\theta} \mid \mathbf{y})$ (Murray et al., 2010; Filippone et al., 2013) may be required. Among the various schemes to recover tractability in the case of models with a non-Gaussian likelihood, we choose the Laplace approximation, as we can formulate it in a way that only requires the solution of linear systems. The GP models we consider assume that the likelihood factorizes across all data points $p(\mathbf{y} \mid \mathbf{f}) = \prod_{i=1}^{n} p(y_i \mid f_i)$. The use of CG for computing the Laplace approximation has been proposed elsewhere (Flaxman et al., 2015), but we make the first use of preconditioning and stochastic gradient estimation within the Laplace approximation to compute stochastic gradients for non-conjugate models.

Defining $W = -\nabla_{\mathbf{f}}\nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})]$ (a diagonal matrix), carrying out the Laplace approximation algorithm, computing its derivatives wrt $\boldsymbol{\theta}$, and making predictions, all possess the same computational bottleneck: the solution of linear systems involving the matrix $B = I + W^{\frac{1}{2}}KW^{\frac{1}{2}}$ (Rasmussen & Williams, 2006). For a given $\boldsymbol{\theta}$, each iteration of the Laplace approximation algorithm requires solving one linear system involving B and two matrix-vector multiplications involving K; the linear system involving B can be solved using CG or PCG. The Laplace approximation yields the mode $\hat{\mathbf{f}}$ of the posterior over latent variables and offers an approximate log-marginal likelihood in the form:

$$\log[\hat{p}(\mathbf{y} \mid \boldsymbol{\theta}, X)] = -\frac{1}{2}\log|B| - \frac{1}{2}\hat{\mathbf{f}}^\top K^{-1}\hat{\mathbf{f}} + \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]$$

which poses the same computational challenges as the regression case. Once again, we therefore seek an alternative way to learn kernel parameters by stochastic gradient optimization based on computing unbiased estimates of the gradient of the approximate log-marginal likelihood. This is complicated further by the inclusion of an additional "implicit" term accounting for the change in the solution given by the Laplace approximation for a change in $\boldsymbol{\theta}$. The full derivation of the gradient is rather lengthy and is deferred to the supplementary material. Nonetheless, it is worth noting that the calculation of the exact gradient involves trace terms similar to the regression case that cannot be computed for large n, and we unbiasedly estimate these using the stochastic approximation of the trace.

3. Preconditioning Kernel Matrices

Here we consider choices for kernel preconditioners, and for the sake of clarity we focus on preconditioners for $K_{\mathbf{y}}$. Unless stated otherwise, we shall consider standard *left* preconditioning, whereby the original problem of solving $K_{\mathbf{y}}\mathbf{z} = \mathbf{v}$ is transformed by applying a preconditioner, P, to both sides of this equation. This formulation may thus be expressed as $P^{-1}K_{\mathbf{y}}\mathbf{z} = P^{-1}\mathbf{v}$.

3.1. Nyström type approximations

The Nyström method was originally proposed to approximate the eigendecomposition of kernel matrices (Williams & Seeger, 2001); as a result, it offers a way to obtain a low rank approximation of K. This method selects a subset of $m \ll n$ data (inducing points) collected in the set U which are intended for approximating the spectrum of K. The resulting approximation is $\hat{K} = K_{XU}K_{UU}^{-1}K_{UX}$ where K_{UU} denotes the evaluation of the kernel function over the inducing points, and K_{XU} denotes the evaluation of the kernel function between the input points and the inducing points. The resulting preconditioner $P = K_{XU}K_{UU}^{-1}K_{UX} + \lambda I$ can be inverted using the matrix inversion lemma

$$P^{-1}\mathbf{v} = \lambda^{-1}\left[I - K_{XU}\left(K_{UU} + K_{UX}K_{XU}\right)^{-1}K_{UX}\right]\mathbf{v},$$

which has $\mathcal{O}(m^3)$ complexity.

3.1.1. Fully and Partially Independent Training Conditional

The use of a subset of data for approximating a GP kernel has also been utilized in the fully and partially independent training conditional approaches (FITC and PITC, respectively) for approximating GP regression (Candela & Rasmussen, 2005). In the former case, the prior covariance of the approximation can be written as follows:

$$P = K_{XU}K_{UU}^{-1}K_{UX} + \mathbf{diag}\left(K - K_{XU}K_{UU}^{-1}K_{UX}\right) + \lambda I.$$

As the name implies, this formulation enforces that the latent variables associated with U are taken to be completely conditionally independent. On the other hand, the PITC method extends on this approach by enforcing that although inducing points assigned to a designated block are conditionally dependent on each other, there is no dependence between points placed in different blocks:

$$P = K_{XU}K_{UU}^{-1}K_{UX} + \mathbf{bldiag}\left(K - K_{XU}K_{UU}^{-1}K_{UX}\right) + \lambda I.$$

For the FITC preconditioner, the diagonal resulting from the training conditional can be added to the diagonal noise matrix, and the inversion lemma can be invoked as for the Nyström case. Meanwhile, for the PITC preconditioner, the noise diagonal can be added to the block diagonal matrix, which can then be inverted block-by-block. Once again, matrix inversion can then be carried out as before, where the inverted block diagonal matrix takes the place of λI in the original formulation.

3.2. Approximate factorization of kernel matrices

This group of preconditioners relies on approximations to K that factorize as $\hat{K} = \Phi\Phi^{\top}$. We shall consider different ways of determining Φ such that P can be inverted at a lower cost than the original kernel matrix K. Once again, this enables us to employ the matrix inversion lemma, and express the linear system:

$$P^{-1}\mathbf{v} = (\Phi\Phi^{\top} + \lambda I)^{-1}\mathbf{v} = \lambda^{-1}[I - \Phi(I + \Phi^{\top}\Phi)^{-1}\Phi^{\top}]\mathbf{v}.$$

We now review a few methods to approximate the kernel matrix K in the form $\Phi\Phi^{\top}$.

3.2.1. Spectral Approximation

The spectral approach uses random Fourier features for deriving a sparse approximation of a GP (Rahimi & Recht, 2008). This approach for GPs was introduced in (Lázaro-Gredilla et al., 2010), and relies on the assumption that stationary kernel functions can be represented as the Fourier transform of non-negative measures. As such, the elements of K can be approximated as follows:

$$\hat{K}_{ij} = \frac{\sigma_0^2}{m}\phi(\mathbf{x}_i)^{\top}\phi(\mathbf{x}_j) = \frac{\sigma_0^2}{m}\sum_{r=1}^{m}\cos\left[2\pi\mathbf{s}_r^{\top}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right].$$

In the equation above, the vectors $\mathbf{s}_r$ denote the *spectral points* (or *frequencies*) which in the case of the RBF kernel can be sampled from $\mathcal{N}\left(\mathbf{0}, \frac{1}{4\pi^2}\Lambda\right)$, where $\Lambda = \left[1/l_1^2, \ldots, 1/l_n^2\right]$. To the best of our knowledge, this is the first time such an approximation has been considered for the purpose of preconditioning kernel matrices.

3.2.2. Partial SVD

Another factorization approach that we consider in this work is the partial singular value decomposition (SVD) method (Golub & Van Loan, 1996). The SVD method factorizes the original kernel matrix K into $A\Lambda A^{\top}$, where A is a unitary matrix and Λ is a diagonal matrix of singular values. Here, we shall consider a variation of this technique called *randomized truncated* SVD (Halko et al., 2011), which constructs an approximate low rank SVD factorization of K using random sampling to accelerate computations.

3.2.3. Structured Kernel Interpolation (SKI)

Some recent work on approximating GPs has exploited the fast computation of Kronecker matrix-vector multiplications when inputs are located on a Cartesian grid (Gilboa et al., 2015). Unfortunately, not all datasets meet this requirement, thus limiting the widespread application of Kronecker inference. To this end, SKI (Wilson & Nickisch, 2015) is an approximation technique which exploits the benefits of the Kronecker product without imposing any requirements on the structure of the training data. In particular, a grid of inducing points, U, is constructed, and the covariance between the training data and U is then represented as $K_{XU} = WK_{UU}$. In this formulation, W denotes a sparse interpolation matrix for assigning weights to the elements of K_{UU}. In this manner, a preconditioner exploiting Kronecker structure can be constructed as $P = WK_{UU}W^\top + \lambda I$. If we consider $V = W/\sqrt{\lambda}$, we can rewrite the (inverse) preconditioner as $P^{-1} = \lambda^{-1}(VK_{\mathrm{UU}}V^\top + I)^{-1}$. Since this can no longer be solved directly, we solve this (inner-loop) linear system using the CG algorithm (all within one iteration of the outer-loop PCG). For badly conditioned systems, although the complexity of the required matrix-vector multiplications is now much less than $\mathcal{O}(n^2)$, the number of iterations to solve linear systems involving the preconditioner is potentially very large, and could diminish the benefits of preconditioning.

3.3. Other approaches

3.3.1. Block Jacobi

An alternative to using a single subset of data involves constructing local GPs over segments of the original data (Snelson & Ghahramani, 2007). An example of such an approach is the *Block Jacobi* approximation, whereby the preconditioner is constructed by taking a block diagonal of K and discarding all other elements in the kernel matrix. In this manner, covariance is only expressed for points within the same block, as $P = \textbf{bldiag}\left(K_{\mathbf{y}} + \lambda I\right)$. The inverse of this block diagonal matrix is computationally cheap (also block diagonal). However, given that a substantial amount of information contained in the original covariance matrix is ignored, this choice is intrinsically a rather crude approach.

3.3.2. Regularization

An appealing feature shared by the aforementioned preconditioners (aside from SKI) is that their structure enables us to directly solve $P^{-1}\mathbf{v}$. An alternative technique for constructing a preconditioner involves adding a positive regularization parameter, δI, to the original kernel matrix, such that $P = K_{\mathbf{y}} + \delta I$ (Srinivasan et al., 2014). This follows from the fact that adding noise to the diagonal of $K_{\mathbf{y}}$ makes it better-conditioned, and the condition number is expected to decrease further as δ increases. Nonetheless, for the purpose of preconditioning, this parameter should be tuned in such a way that P remains a sensible approximation of $K_{\mathbf{y}}$. As opposed to the previous preconditioners, this is an instance of *right* preconditioning, which has the following general form $K_{\mathbf{y}}P^{-1}(P\mathbf{x}) = \mathbf{v}$.

Given that it is no longer possible to evaluate $P^{-1}\mathbf{v}$ analytically, this linear system is solved yet again using CG, such that a linear system of equations is solved at every outer iteration of the PCG algorithm. Due to the potential loss of accuracy incurred while solving the inner linear systems, a variation of the standard PCG algorithm, referred to as *flexible* PCG (Notay, 2000), is used instead. Using this approach, a re-orthogonalization step is introduced such that the search directions remain orthogonal even when the inner system is not solved to high precision.

4. Comparison of Preconditioners

In this section, we provide an empirical exploration of these preconditioners in a practical setting. We begin by considering three datasets for regression from the UCI repository (Asuncion & Newman, 2007), namely the Concrete dataset ($n = 1030, d = 8$), the Power Plant dataset ($n = 9568, d = 4$), and the Protein dataset ($n = 45730, d = 9$). In particular, we evaluate the convergence in solving $K_{\mathbf{y}}\mathbf{z} = \mathbf{y}$ using iterative methods, where $\mathbf{y}$ denotes the labels of the designated dataset, and $K_{\mathbf{y}}$ is constructed using different configurations of kernel parameters.

With this experiment, we aim to assess the quality of different preconditioners based on how many matrix-vector products they require, which, for most approaches, corresponds to the number of iterations taken by PCG to converge. The convergence threshold is set to $\epsilon^2 = n \cdot 10^{-10}$ so as to roughly accept an average error of 10^{-5} on each element of the solution.

For every variation, we set the parameters of the preconditioners so as to have a complexity lower than the $\mathcal{O}(n^2)$ cost associated with matrix-vector products; by doing so, we can assume that the latter computations are the dominant cost for large n. In particular, for Nyström-type methods, we set $m = \sqrt{n}$ inducing points, so that when we invert the preconditioner using the matrix inversion lemma, the cost is in $\mathcal{O}(m^3) = \mathcal{O}(n^{3/2})$. Similarly, for the Spectral preconditioner, we set $m = \sqrt{n}$ random features. For the SKI preconditioner, we take an equal number of elements on the grid for each dimension; under this assumption, Kronecker products have $\mathcal{O}(dn^{\frac{d+1}{d}})$ cost (Gilboa et al., 2015), and we set the size of the grid so that the complexity of applying the preconditioner matches $\mathcal{O}(n^{3/2})$, so as to be consistent with the other preconditioners. For the Regularized approach, each iteration needed to apply the precondi-

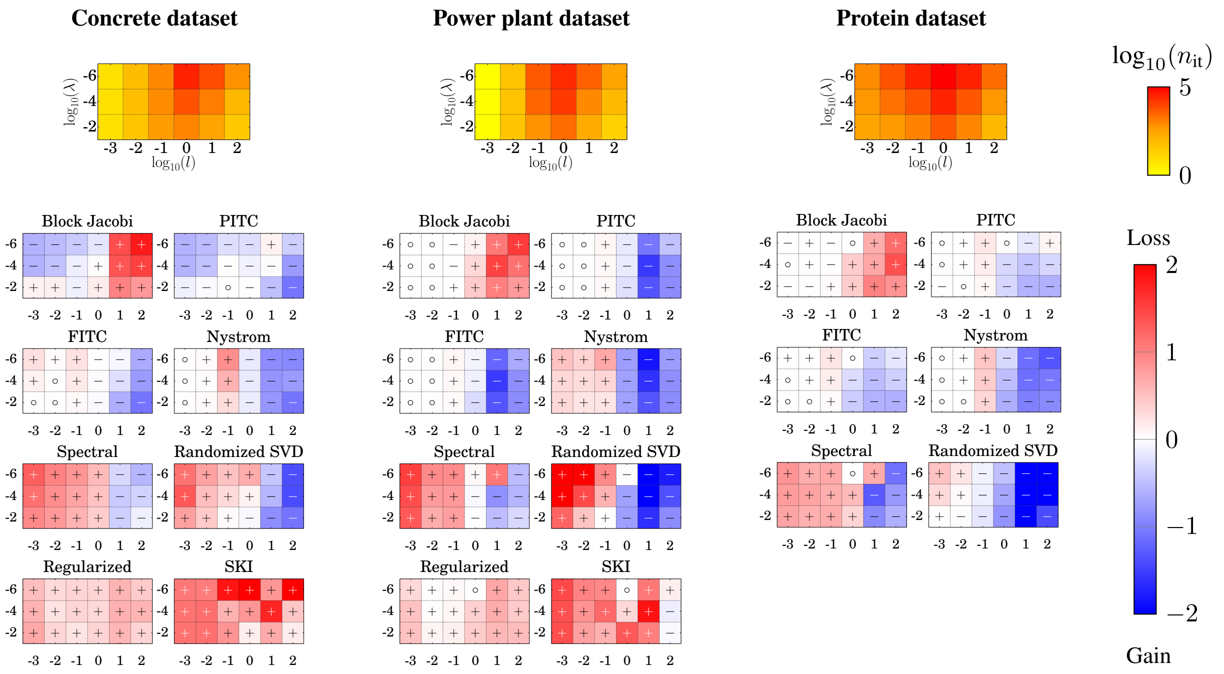

Figure 1. Comparison of preconditioners for different settings of kernel parameters. The lengthscale l and the noise variance λ are shown on the x and y axes respectively. The top figure indicates the number of iterations required to solve the corresponding linear system using CG, whilst the bottom part of the figure shows the rate of improvement (negative - blue) or degradation (positive - red) achieved by using PCG to solve the same linear system. Parameters and results are reported in $\log_{10}$. Symbols added to facilitate reading in B/W print.

tioner requires one matrix-vector product, and we add this to the overall count of such computations. For this preconditioner, we add a diagonal offset δ to the original matrix, equivalent to two orders of magnitude greater than the noise of the process. In general, although the complexity of PCG is indeed no different from that of CG, we emphasize that experiencing a 2-fold or 5-fold (in some cases even an order of magnitude) improvement can be very substantial when plain CG takes very long to converge or when the dataset is large.

We focus on an isotropic RBF variant of the kernel in eq. 1, fixing the marginal variance σ^2 to one. We vary the lengthscale parameter l and the noise variance λ in $\log_{10}$ scale. The top part of fig. 1 shows the number of iterations that the standard CG algorithm takes, where we have capped the number of iterations to 100,000.

The bottom part of the figure reports the improvement offered by various preconditioners measured as

$$\log_{10}\left(\frac{\#\text{ PCG iterations}}{\#\text{ CG iterations}}\right).$$

It is worth noting that when both CG and PCG fail to converge within the upper bound, the improvement will be marked as 0, i.e. neither a gain or a loss within the given bound. The results plotted in fig. 1 indicate that the low-rank preconditioners (PITC, FITC and Nyström) achieve significant reductions in the number of iterations for each dataset, and all approaches work best when the lengthscale is longer, characterising smoother processes. In contrast, preconditioning seems to be less effective when the lengthscale is shorter, corresponding to a kernel matrix that is more sparse. However, for cases yielding positive results, the improvement is often in the range of an order of magnitude, which can be substantial when a large number of iterations is required by the CG algorithm.

The results also confirm that, as alluded to in the previous section, Block Jacobi preconditioning is generally a poor preconditioner, particularly when the corresponding kernel matrix is dense. The only minor improvements were observed when CG itself converges quickly, in which case preconditioning serves very little purpose either way.

The regularization approach with flexible conjugate gradient does not appear to be effective in any case either, particularly due to the substantial amount of iterations required for solving an inner system at every iteration of the PCG algorithm. This implies that introducing additional small jitter to the diagonal does not necessarily make the system much easier to solve, whilst adding an overly large offset would negatively impact convergence of the outer algorithm. One could assume that tuning the value of this parameter could result in slightly better results; however, preliminary experiments in this regard yielded only minor improvements.

The results for SKI preconditioning are similarly discouraging at face value. When the matrix $K_{\mathbf{y}}$ is very badly conditioned, an excessive number of inner iterations are required for every iteration of outer PCG. This greatly increases the duration of solving such systems, and as a result, this method was not included in the comparison for the Protein dataset, where it was evident that preconditioning the matrix in this manner would not yield satisfactory improvements. Notwithstanding that these experiments depict a negative view of SKI preconditioning, it must be said that we assumed a fairly simplistic interpolation procedure in our experiments, where each data point was mapped to nearest grid location. The size of the constructed grid is also hindered considerably by the constraint imposed by our upper bound on complexity. Conversely, more sophisticated interpolation strategies or even grid formulation procedures could possibly speed up the convergence of CG for the inner systems. In line with this thought, however, one could argue that the preconditioner would no longer be straightforward to construct, which goes against our innate preference towards easily derived preconditioners.

5. Impact of preconditioning on GP learning

One of the primary objectives of this work is to reformulate GP regression and classification in such a way that preconditioning can be effectively exploited. In section 2, we demonstrated how preconditioning can indeed be applied to GP regression problems, and also proposed a novel way of rewriting GP classification in terms of solving linear systems (where preconditioning can thus be employed). We can now evaluate how the proposed preconditioned GP techniques compare to other state of the art methods.

To this end, in this section, we empirically report on the generalization ability of GPs as a function of the time taken to optimize parameters $\boldsymbol{\theta}$ and compute predictions. In particular, for each of the methods featured in our comparison, we iteratively run the optimization of kernel parameters for a few iterations and predict on unseen data, and assess how prediction accuracy varies over time for different methods.

The analysis provided in this report is inspired by (Chalupka et al., 2013), although we do not propose an *approximate* method to learn GP kernel parameters. Instead, we put forward a means of accelerating the optimization of kernel parameters *without any approximation*[2]. Given the predictive mean and variance for the N_{test} test points, say m_{*i} and s_{*i}^2, we report two error measures, namely the Root Mean Square Error, $\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}(m_{*i} - y_{*i})^2}$, along with the negative log-likelihood on the test data, $-\sum_{i=1}^{N_{\text{test}}}\log[p(y_{*i} \mid m_{*i}, s_{*i}^2)]$, where y_{*i} denotes the label of the ith of N_{test} data points. For classification, instead of the RMSE we report the error rate of the classifier.

[2]The one proviso to this statement is that, for non-Gaussian likelihood, stochastic gradients target the approximate log-marginal likelihood obtained by the Laplace approximation.

We can make use of stochastic gradients for GP models to optimize kernel parameters using off-the-shelf stochastic gradient optimization algorithms. In order to reduce the number of parameters to tune, we employ ADAGRAD (Duchi et al., 2011) – an optimization algorithm having a single step-size parameter. For the purpose of this experiment, we do not attempt to optimize this parameter, since this would require additional computations. Nonetheless, our experience with training GP models indicates that the choice of this parameter is not critical: we set the step-size to one.

Fig. 2 shows the two error measures over time for a selection of approaches. In the figure, PCG and CG refer to stochastic gradient optimization of kernel parameters using ADAGRAD, where linear systems are solved with PCG and CG, respectively. In view of the results obtained in our comparison of preconditioners, we decide to proceed with the Nyström preconditioning method. Furthermore, we construct the preconditioner with $m = 4\sqrt{n}$ points randomly selected from the input data at each iteration, such that the overall complexity of the PCG method matches plain CG. For these methods, stochastic estimates of trace terms are carried out using $N_{\mathbf{r}} = 4$ random vectors. The baseline CHOL method refers to the optimization of kernel parameters using the L-BFGS algorithm, where the exact log-marginal likelihood and its gradient are calculated using the full Cholesky decomposition of $K_{\mathbf{y}}$ or B.

Alongside these approaches for optimizing kernel parameters without approximation, we also evaluate the performance of approximate GP methods. For this experiment, we chose to compare against approximations found in the software package GPstuff (Vanhatalo et al., 2013), namely the fully and partial independent training conditional approaches (FITC, PITC), and the sparse variational GP (VAR) (Titsias, 2009). In order to match the computational cost of CG/PCG, which is in $\mathcal{O}(n^2)$, we set the number of inducing points for the approximate methods to be $n^{2/3}$.

All methods are initialized from the same set of kernel parameters, and the curves are averaged over 5 folds (3 for the Protein and EEG datasets). For the sake of integrity, we ran each method in the comparison individually on a workstation with Intel Xeon E5-2630 CPU having 16 cores and 128GB RAM. We also ensured that all methods reported

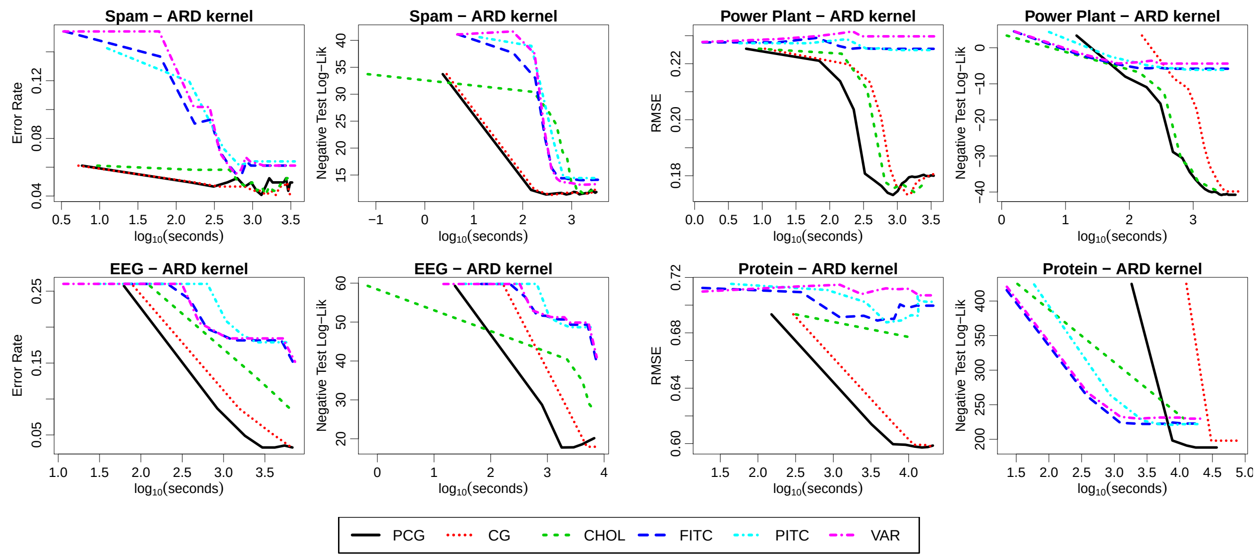

Figure 2. RMSE and negative log of the likelihood on $\sqrt{n}$ held out test data over time. GP models employ the ARD kernel in eq. 1. GP classification: Spam dataset ($n = 4601, d = 57$) and EEG dataset ($n = 14979, d = 14$). GP regression: Power Plant dataset ($n = 9568, d = 4$) and Protein dataset ($n = 45730, d = 9$). Curves are averaged over multiple repetitions.

in the comparison used optimized linear algebra routines exploiting the multi-core architecture. This diligence for ensuring fairness gives credence to our assumption that the timings are not affected by external factors other than the actual implementation of the algorithms. The CG, PCG and CHOL approaches have been implemented in R; the fact that the approximate methods were implemented in a different environment (GPstuff is written in Matlab/Octave) and by a different developer may cast some doubt on the correctness of directly comparing results. However, we believe that the key point emerging from this comparison is that preconditioning feasibly enables the use of iterative approaches for optimization of kernel parameters in GPs, and the results are competitive with those achieved using popular GP software packages.

For the reported experiments, it was possible to store the kernel matrix K for all datasets, making it possible to compare methods against the baseline GP where computations use Cholesky decompositions. We stress, however, that iterative approaches based on CG/PCG can be implemented without the need to store K, whereas this is not possible for approaches that attempt to factorize K exactly. It is also worth noting that for the CG/PCG approach, calculating the log-likelihood on test data requires solving one linear system for each test point; this clearly penalizes the speed of these methods given the set-up of the experiment, where predictions are carried out every fixed number of iterations.

6. Discussion and Conclusions

Careful attention to numerical properties is essential in scaling machine learning to large and realistic datasets. Here we have introduced the use of preconditioning to the implementation of kernel machines, specifically, prediction and learning of kernel parameters for GPs. Our novel scheme permits the use of any likelihood that factorizes over the data points, allowing us to tackle both regression and classification. We have shown robust performance improvements, in both accuracy and computational cost, over a host of state-of-the-art approximation methods for kernel machines. Notably, our method is exact in the limit of iterations, unlike approximate alternatives. We have also shown that the use of PCG is competitive with exact Cholesky decomposition in modestly sized datasets, when the Cholesky factors can be feasibly computed. When data and thus the kernel matrix grow large enough, Cholesky factorization becomes unfeasible, leaving PCG as the optimal choice.

One of the key features of a PCG implementation is that it does not require storage of any $\mathcal{O}(n^2)$ objects. We plan to extend our implementation to compute the elements of K on the fly in one case, and in another case store K in a distributed fashion (e.g. in TensorFlow/Spark). Furthermore, while we have focused on solving linear systems, we can also use preconditioning for other iterative algorithms involving the K matrix, e.g., those to solve $\log(K)\mathbf{v}$ and $K^{1/2}\mathbf{v}$ (Chen et al., 2011), as is often useful in estimating marginal likelihoods for probabilistic kernel models like GPs.

Acknowledgements

KC and MF are grateful to Pietro Michiardi and Daniele Venzano for assisting the completion of this work by providing additional computational resources for running the experiments. JPC acknowledges support from the Sloan Foundation, The Simons Foundation (SCGB#325171 and SCGB#325233), and The Grossman Center at Columbia University.

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.

Ashby, S. F. and Falgout, R. D. A Parallel Multigrid Preconditioned Conjugate Gradient algorithm for Groundwater Flow Simulations. *Nuclear Science and Engineering*, 124(1):145–159, 1996.

Asuncion, A. and Newman, D. J. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2007.

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.

Chalupka, K., Williams, C. K. I., and Murray, I. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14, 2013.

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.

Chen, K. *Matrix Preconditioning Techniques and Applications*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

Davies, A. *Effective Implementation of Gaussian Process Regression for Machine Learning*. PhD thesis, University of Cambridge, 2014.

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. and Engler, R. Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE). In Blei, D. and Bach, F. (eds.), *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1015–1024. JMLR Workshop and Conference Proceedings, 2015.

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.

Flaxman, S., Wilson, A., Neill, D., Nickisch, H., and Smola, A. Fast Kronecker inference in Gaussian processes with non-Gaussian likelihoods. In Blei, D. and Bach, F. (eds.), *Proceedings of The 32nd International Conference on Machine Learning*, pp. 607–616. JMLR Workshop and Conference Proceedings, 2015.

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 Transactions on Pattern Analysis and Machine Intelligence*, 37(2):424–436, 2015.

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

Halko, N., Martinsson, P. G., and Tropp, J. A. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, May 2011.

Kuss, M. and Rasmussen, C. E. Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

Murray, I., Adams, R. P., and MacKay, D. J. C. Elliptical slice sampling. *Journal of Machine Learning Research - Proceedings Track*, 9:541–548, 2010.

Nickisch, H. and Rasmussen, C. E. Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9:2035–2078, October 2008.

Notay, Y. Flexible Conjugate Gradients. *SIAM Journal on Scientific Computing*, 22(4):1444–1460, 2000.

Rahimi, A. and Recht, B. Random Features for Large-Scale Kernel Machines. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1177–1184. Curran Associates, Inc., 2008.

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.

Schölkopf, B. and Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

Snelson, E. and Ghahramani, Z. Local and global sparse Gaussian process approximations. In Meila, M. and Shen, X. (eds.), *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2 of *JMLR Proceedings*, pp. 524–531. JMLR.org, 2007.

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. Difference Filter Preconditioning for Large Covariance Matrices. *SIAM Journal on Matrix Analysis Applications*, 33(1):52–72, 2012.

Titsias, M. K. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In Dyk, D. A. and Welling, M. (eds.), *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5 of *JMLR Proceedings*, pp. 567–574. JMLR.org, 2009.

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., and Vehtari, A. Gpstuff: Bayesian modeling with gaussian processes. *Journal of Machine Learning Research*, 14(1):1175–1179, 2013.

Williams, C. K. I. and Seeger, M. Using the nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems 13*, pp. 682–688. MIT Press, 2001.

Wilson, A. and Nickisch, H. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In Blei, D. and Bach, F. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1775–1784. JMLR Workshop and Conference Proceedings, 2015.

A. Other results not included in the paper

In fig. 3 we report some of the runs that we did not include in the main text for lack of space. The figure reports plots on the error vs. time for the same regression cases considered in the main text but with an isotropic kernel, and results on the concrete dataset with isotropic and ARD kernels.

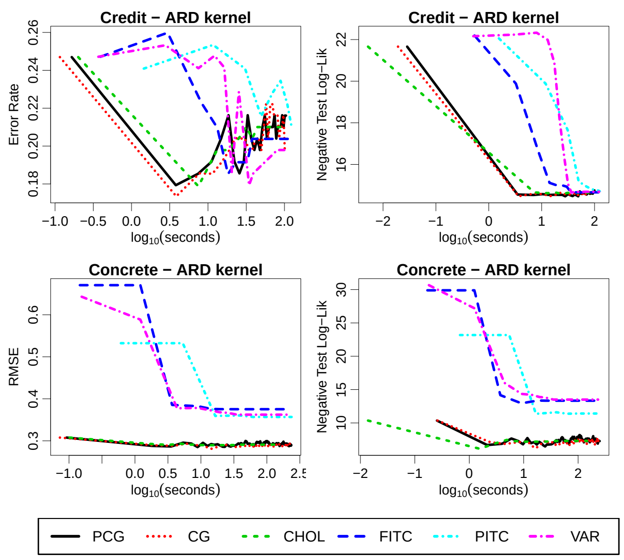

Figure 3. RMSE and negative log of the likelihood on $\sqrt{n}$ held out test data over time. GP models employ the ARD kernel in eq. 1. GP classification: Credi dataset ($n = 1000, d = 24$). GP regression: Concrete dataset ($n = 1029, d = 8$). Curves are averaged over multiple repetitions.

B. Gaussian Processes with non-Gaussian likelihood functions

In this section we report the derivations of the quantities needed to compute an unbiased estimate of the log-marginal likelihood given by the Laplace approximation for GP models with non-Gaussian likelihood functions. Throughout this section, we assume a factorizing likelihood

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i).$$

and we specialize the equations to the probit likelihood

$$p(y_i \mid f_i) = \Phi(y_i f_i). \tag{3}$$

where Φ denotes the cumulative function of the Gaussian density. The latent variables $\mathbf{f}$ are given a zero mean GP prior $\mathbf{f} \sim \mathcal{N}(\mathbf{f}|\mathbf{0}, K)$.

For a given value of the hyperparameters $\boldsymbol{\theta}$, define

$$\Psi(\mathbf{f}) = \log[p(\mathbf{y} \mid \mathbf{f})] + \log[p(\mathbf{f} \mid \boldsymbol{\theta})] + \text{const.} \tag{4}$$

as the logarithm of the posterior density over $\mathbf{f}$. Performing a Laplace approximation amounts in defining a Gaussian $q(\mathbf{f} \mid \mathbf{y}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f} \mid \hat{\mathbf{f}}, \hat{\Sigma})$, such that

$$\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} \Psi(\mathbf{f}) \qquad \text{and} \qquad \hat{\Sigma}^{-1} = -\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\hat{\mathbf{f}}). \tag{5}$$

As it is not possible to directly solve the maximization problem in equation 5, an iterative procedure based on the following Newton-Raphson formula is usually employed,

$$\mathbf{f}^{\text{new}} = \mathbf{f} - (\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f}))^{-1}\nabla_{\mathbf{f}}\Psi(\mathbf{f}), \tag{6}$$

starting from some initial $\mathbf{f}$ until convergence. The gradient and the Hessian of the log of the target density are

$$\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})] - K^{-1}\mathbf{f} \quad \text{and} \tag{7}$$

$$\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})] - K^{-1} = -W - K^{-1}, \tag{8}$$

where we have defined $W = -\nabla_{\mathbf{f}}\nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})]$, which is diagonal because the likelihood factorizes over observations. Note that if $\log[p(\mathbf{y} \mid \mathbf{f})]$ is concave, such as in probit classification, $\Psi(\mathbf{f})$ has a unique maximum.

Standard manipulations lead to

$$\mathbf{f}^{\text{new}} = (K^{-1} + W)^{-1}(W\mathbf{f} + \nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})]).$$

We can rewrite the inverse of the negative Hessian using the matrix inversion lemma:

$$\left(K^{-1} + W\right)^{-1} = K - KW^{\frac{1}{2}}B^{-1}W^{\frac{1}{2}}K,$$

where

$$B = I + W^{\frac{1}{2}}KW^{\frac{1}{2}}.$$

This means that each iteration becomes:

$$\mathbf{f}^{\text{new}} = (K - KW^{\frac{1}{2}}B^{-1}W^{\frac{1}{2}}K)(W\mathbf{f} + \nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})]).$$

We can define $\mathbf{b} = (W\mathbf{f} + \nabla_{\mathbf{f}} \log[p(\mathbf{y} \mid \mathbf{f})])$ and rewrite this expression as:

$$\mathbf{f}^{\text{new}} = K(\mathbf{b} - W^{\frac{1}{2}}B^{-1}W^{\frac{1}{2}}K\mathbf{b}).$$

From this, we see that at convergence

$$\mathbf{a} = K^{-1}\hat{\mathbf{f}} = (\mathbf{b} - W^{\frac{1}{2}}B^{-1}W^{\frac{1}{2}}K\mathbf{b}).$$

As we will see later, the definition of $\mathbf{a}$ is useful for the calculation of the gradient and for predictions.

Proceeding with the calculations from right to left we see that in order to complete a Newton-Raphson iteration the expensive operations are: (i) carry out one matrix-vector multiplication $K\mathbf{b}$, (ii) solve a linear system involving the

Algorithm 2 Laplace approximation for GPs

```
1: Input: data $X$, labels $\mathbf{y}$, likelihood function $p(\mathbf{y} \mid \mathbf{f})$
2: $\mathbf{f} = \mathbf{0}$
3: repeat
4:    Compute $\mathrm{diag}(W)$, $\mathbf{b}$, $W^{\frac{1}{2}}K\mathbf{b}$
5:    $\mathrm{solve}(B, W^{\frac{1}{2}}K\mathbf{b})$
6:    Compute $\mathbf{a}$, $K\mathbf{a}$
7:    Compute $\mathbf{f}^{\mathrm{new}}$
8: until convergence
9: return $\hat{\mathbf{f}}$, $\mathbf{a}$
```

Algorithm 3 Stochastic gradients for GPs

```
1: Input: data $X$, labels $\mathbf{y}$, $\hat{\mathbf{f}}$, $\mathbf{a}$
2: $\mathrm{solve}(B, \mathbf{r}^{(i)})$ for $i = 1, \ldots, N_{\mathbf{r}}$
3: Compute first term of $\tilde{g}_i$
4: Compute second term of $\tilde{g}_i$
5: $\mathrm{solve}(B, W^{\frac{1}{2}}K\mathbf{r}^{(i)})$ for $i = 1, \ldots, N_{\mathbf{r}}$
6: Compute $\tilde{\mathbf{u}}$
7: $\mathrm{solve}(B, W^{\frac{1}{2}}\frac{\partial K}{\partial \theta_i}\nabla_{\hat{\mathbf{f}}}\log[p(\mathbf{y} \mid \hat{\mathbf{f}})])$
8: Compute third term of $\tilde{g}_i$
9: return $\tilde{\mathbf{g}}$
```

B matrix, and (iii) carry out one matrix-vector multiplication involving K and the vector in the parenthesis. Calculating $\mathbf{b}$ and performing any multiplications of $W^{\frac{1}{2}}$ with vectors cost $\mathcal{O}(n)$.

All these operations can be carried out without the need to store K or any other $n \times n$ matrices. The linear system in (ii) can be solved using the CG algorithm that involves repeatedly multiplying B (and therefore K) with vectors.

B.1. Stochastic gradients

The Laplace approximation yields an approximate log-marginal likelihood in the following form:

$$\log[\hat{p}(\mathbf{y} \mid \boldsymbol{\theta}, X)] = -\frac{1}{2}\log|B| - \frac{1}{2}\hat{\mathbf{f}}^{\top}K^{-1}\hat{\mathbf{f}} + \log[p(\mathbf{y} \mid \hat{\mathbf{f}})] \quad (9)$$

Handy relationships that we will be using in the remainder of this section are:

$$\log|B| = \log|I + W^{\frac{1}{2}}KW^{\frac{1}{2}}| = \log|I + KW|;$$

$$(I + KW)^{-1} = W^{-\frac{1}{2}}B^{-1}W^{\frac{1}{2}}.$$

The gradient of the log-marginal likelihood with respect to the kernel parameters $\boldsymbol{\theta}$ requires differentiating the terms that explicitly depend on $\boldsymbol{\theta}$ and those that implicitly depend on it because a change in the parameters reflects in a change in $\hat{\mathbf{f}}$. Denoting by g_i the ith component of the gradient of $\frac{\partial \log[\hat{p}(\mathbf{y}|\boldsymbol{\theta})]}{\partial \theta_i}$, we obtain

$$\begin{aligned} g_i = & -\frac{1}{2}\mathrm{Tr}\left(B^{-1}\frac{\partial B}{\partial \theta_i}\right) \\ & + \frac{1}{2}\hat{\mathbf{f}}^{\top}K^{-1}\frac{\partial K}{\partial \theta_i}K^{-1}\hat{\mathbf{f}} \\ & + \left[\nabla_{\hat{\mathbf{f}}}\log[\hat{p}(\mathbf{y}|\boldsymbol{\theta})]\right]^{\top}\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i} \end{aligned} \quad (10)$$

The trace term cannot be computed exactly for large n so we propose a stochastic estimate:

$$-\frac{1}{2}\left[\widetilde{\mathrm{Tr}\left(B^{-1}\frac{\partial B}{\partial \theta_i}\right)}\right] = -\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}(\mathbf{r}^{(i)})^{\top}B^{-1}\frac{\partial B}{\partial \theta_i}\mathbf{r}^{(i)}.$$

By noticing that the derivative of B is $W^{\frac{1}{2}}\frac{\partial K}{\partial \theta_i}W^{\frac{1}{2}}$, this simplifies to

$$-\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}(\mathbf{r}^{(i)})^{\top}B^{-1}W^{\frac{1}{2}}\frac{\partial K}{\partial \theta_i}W^{\frac{1}{2}}\mathbf{r}^{(i)},$$

so we need to solve $N_{\mathbf{r}}$ linear systems involving B.

The second term contains the linear system $K^{-1}\hat{\mathbf{f}}$ that we already have from the Laplace approximation and is $\mathbf{a}$.

The third term is slightly more involved and will be dealt with in the next sub-section.

B.1.1. Implicit derivatives

The last (implicit) term in the last equation can be simplified by noticing that:

$$\log[\hat{p}(\mathbf{y} \mid \boldsymbol{\theta})] = \Psi(\hat{\mathbf{f}}) - \frac{1}{2}\log|B|$$

and that the derivative of the first term wrt $\hat{\mathbf{f}}$ is zero because $\hat{\mathbf{f}}$ maximizes $\Psi(\hat{\mathbf{f}})$. Therefore:

$$\left[\nabla_{\hat{\mathbf{f}}}\log[\hat{p}(\mathbf{y} \mid \boldsymbol{\theta})]\right]^{\top}\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i} = -\frac{1}{2}\left[\nabla_{\hat{\mathbf{f}}}\log|B|\right]^{\top}\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i}$$

The components of $\left[\nabla_{\hat{\mathbf{f}}}\log|B|\right]$ can be obtained by considering the identity $\log|B| = \log|I + KW|$, so differentiating $\log|B|$ wrt the components of $\hat{\mathbf{f}}$ becomes:

$$\frac{\partial \log|I + KW|}{\partial(\hat{\mathbf{f}})_j} = \mathrm{Tr}\left((I + KW)^{-1}K\frac{\partial W}{\partial(\hat{\mathbf{f}})_j}\right)$$

We can rewrite this by gathering K inside the inverse and, due to the inversion of the matrix product, K cancels out:

$$\frac{\partial \log|I + KW|}{\partial(\hat{\mathbf{f}})_j} = \mathrm{Tr}\left((K^{-1} + W)^{-1}\frac{\partial W}{\partial(\hat{\mathbf{f}})_j}\right)$$

We notice here that the resulting trace contains the inverse of the same matrix needed in the iterations of the Laplace approximation and that the matrix $\frac{\partial W}{\partial(\hat{\mathbf{f}})_j}$ is zero everywhere except in the jth diagonal element where it attains the value:

$$\frac{\partial W}{\partial (\hat{\mathbf{f}})_j} = \frac{\partial^3 \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]}{\partial (\hat{\mathbf{f}})_j^3}$$

For this reason, it would be possible to simplify the trace term as the product between the jth diagonal element of $(K^{-1} + W)^{-1}$ and $\frac{\partial^3 \log[p(\mathbf{y}|\hat{\mathbf{f}})]}{\partial (\hat{\mathbf{f}})_j^3}$. Bearing in mind that we need n of these quantities, we could define

$$D = \mathrm{diag}\left[\mathrm{diag}\left[(K^{-1} + W)^{-1}\right]\right]$$

$$(\mathbf{d})_j = \frac{\partial^3 \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]}{\partial (\hat{\mathbf{f}})_j^3}$$

and rewrite

$$-\frac{1}{2}\left[\nabla_{\hat{\mathbf{f}}} \log |B|\right] = -\frac{1}{2} D\mathbf{d}$$

which is the standard way to proceed when computing the gradient of the approximate log-marginal likelihood using the Laplace approximation (Rasmussen & Williams, 2006). However, this would be difficult to compute exactly for large n, as this would require inverting $K^{-1} + W$ first and then compute its diagonal. Using the matrix inversion lemma would not simplify things as there would still be an inverse of B to compute explicitly. We therefore aim for a stochastic estimate of this term starting from:

$$\begin{aligned}\frac{\partial \log |I + KW|}{\partial (\hat{\mathbf{f}})_j} &= \mathrm{Tr}\left((K^{-1} + W)^{-1} \frac{\partial W}{\partial (\hat{\mathbf{f}})_j}\right) \\ &= \mathrm{Tr}\left((K^{-1} + W)^{-1} \frac{\partial W}{\partial (\hat{\mathbf{f}})_j} \mathrm{E}[\mathbf{r}\mathbf{r}^\top]\right)\end{aligned} \quad (11)$$

where we have introduced the $\mathbf{r}$ vectors with the property $\mathrm{E}[\mathbf{r}\mathbf{r}^\top] = I$. So an unbiased estimate of the trace for each component of $\hat{\mathbf{f}}$ is:

$$\begin{aligned}(\tilde{\mathbf{u}})_j &= \left[\frac{\partial \log \widetilde{|I + KW|}}{\partial (\hat{\mathbf{f}})_j}\right] \\ &= \frac{1}{N_{\mathbf{r}}} \sum_{i=1}^{N_{\mathbf{r}}} (\mathbf{r}^{(i)})^\top (K^{-1} + W)^{-1} \frac{\partial W}{\partial (\hat{\mathbf{f}})_j} \mathbf{r}^{(i)}\end{aligned} \quad (12)$$

which requires solving $N_{\mathbf{r}}$ linear systems involving the B matrix:

$$(K^{-1} + W)^{-1}\mathbf{r}^{(i)} = K(\mathbf{r}^{(i)} - W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} K \mathbf{r}^{(i)})$$

The derivative of $\hat{\mathbf{f}}$ wrt θ_i can be obtained by differentiating the expression $\hat{\mathbf{f}} = K \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]$:

$$\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i} = \frac{\partial K}{\partial \theta_i} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})] + K \nabla_{\hat{\mathbf{f}}} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})] \frac{\partial \hat{\mathbf{f}}}{\partial \theta_i}$$

Algorithm 4 Prediction for GPs with Laplace approximation without Cholesky decompositions

```
1: Input: data $X$, labels $\mathbf{y}$, test input $\mathbf{x}_*$, $\hat{\mathbf{f}}$, $\mathbf{a}$
2: Compute $\mu_*$
3: solve$(B, W^{\frac{1}{2}}\mathbf{k}_*)$
4: Compute $s_*^2$, $\Phi\left(\frac{m_*}{\sqrt{1+s_*^2}}\right)$
5: return $\Phi\left(\frac{m_*}{\sqrt{1+s_*^2}}\right)$
```

Given that $\nabla_{\hat{\mathbf{f}}} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})] = -W$ we can rewrite:

$$(I + KW)\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i} = \frac{\partial K}{\partial \theta_i} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]$$

which yields:

$$\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i} = (I + KW)^{-1} \frac{\partial K}{\partial \theta_i} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]$$

So an unbiased estimate of the implicit term in the gradient of the approximate log-marginal likelihood becomes:

$$-\frac{1}{2}\tilde{\mathbf{u}}^\top (I + KW)^{-1} \frac{\partial K}{\partial \theta_i} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]$$

Rewriting the inverse in terms of B yields:

$$-\frac{1}{2}\tilde{\mathbf{u}}^\top W^{-\frac{1}{2}} B^{-1} W^{\frac{1}{2}} \frac{\partial K}{\partial \theta_i} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y} \mid \hat{\mathbf{f}})]$$

Putting everything together, the components of the stochastic gradient are:

$$\begin{aligned}\tilde{g}_i = &-\frac{1}{2N_{\mathbf{r}}} \sum_{i=1}^{N_{\mathbf{r}}} (\mathbf{r}^{(i)})^\top B^{-1} W^{\frac{1}{2}} \frac{\partial K}{\partial \theta_i} W^{\frac{1}{2}} \mathbf{r}^{(i)} \\ &+\frac{1}{2}\mathbf{a}^\top \frac{\partial K}{\partial \theta_i} \mathbf{a} \\ &-\frac{1}{2}\tilde{\mathbf{u}}^\top W^{-\frac{1}{2}} B^{-1} W^{\frac{1}{2}} \frac{\partial K}{\partial \theta_i} \nabla_{\hat{\mathbf{f}}} \log[p(\mathbf{y}|\hat{\mathbf{f}})]\end{aligned} \quad (13)$$

B.2. Predictions

To obtain an approximate predictive distribution, conditioned on a value of the hyperparameters $\boldsymbol{\theta}$, we can compute:

$$p(y_* \mid \mathbf{y}, \boldsymbol{\theta}) = \int p(y_* \mid f_*) p(f_* \mid \mathbf{f}, \boldsymbol{\theta}) q(\mathbf{f} \mid \mathbf{y}, \boldsymbol{\theta}) df_* d\mathbf{f}. \quad (14)$$

Given the properties of multivariate normal variables, f_* is distributed as $\mathcal{N}(f_* \mid \mu_*, \beta_*^2)$ with $\mu_* = \mathbf{k}_*^\top K^{-1}\mathbf{f}$ and $\beta_*^2 = k_{**} - \mathbf{k}_*^\top K^{-1}\mathbf{k}_*$. Approximating $p(\mathbf{f} \mid \mathbf{y}, \boldsymbol{\theta})$ with a Gaussian $q(\mathbf{f} \mid \mathbf{y}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}_q, \Sigma_q)$ makes it possible to analytically perform integration with respect to $\mathbf{f}$ in eq. 14. In particular, the integration with respect to $\mathbf{f}$ yields $\mathcal{N}(f_* \mid m_*, s_*^2)$ with

$$m_* = \mathbf{k}_*^\top K^{-1}\hat{\mathbf{f}}$$

and

$$s_*^2 = k_{**} - \mathbf{k}_*^\top (K + W^{-1})^{-1}\mathbf{k}_*$$

These quantities can be rewritten as:

$$m_* = \mathbf{k}_*^\top \mathbf{a}$$

and

$$s_*^2 = k_{**} - \mathbf{k}_*^\top W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} \mathbf{k}_*$$

This shows that the mean is cheap to compute, whereas the variance requires solving another linear system involving B for each test point.

The univariate integration with respect to f_* follows exactly in the case of a probit likelihood, as it is a convolution of a Gaussian and a cumulative Gaussian

$$\int p(y_* \mid f_*)\mathcal{N}(f_* \mid m_*, s_*^2) df_* = \Phi\left(\frac{m_*}{\sqrt{1+s_*^2}}\right). \tag{15}$$

B.3. Low rank preconditioning

When a low rank approximation of the matrix K is available, say $\hat{K} = \Phi\Phi^\top$, the inverse of the preconditioner can be rewritten as:

$$(I + W^{\frac{1}{2}}\hat{K}W^{\frac{1}{2}})^{-1} = (I + W^{\frac{1}{2}}\Phi\Phi^\top W^{\frac{1}{2}})^{-1}$$

By using the matrix inversion lemma we obtain:

$$(I+W^{\frac{1}{2}}\Phi\Phi^\top W^{\frac{1}{2}})^{-1} = I - W^{\frac{1}{2}}\Phi(I+\Phi^\top W\Phi)^{-1}\Phi^\top W^{\frac{1}{2}}$$

Similarly to the GP regression case, the application of this preconditioner is in $\mathcal{O}(m^3)$, where m is the rank of Φ.

MCMC for Variationally Sparse Gaussian Processes

James Hensman
CHICAS, Lancaster University
james.hensman@lancaster.ac.uk

Alexander G. de G. Matthews
University of Cambridge
am554@cam.ac.uk

Maurizio Filippone
EURECOM
maurizio.filippone@eurecom.fr

Zoubin Ghahramani
University of Cambridge
zoubin@cam.ac.uk

Abstract

Gaussian process (GP) models form a core part of probabilistic machine learning. Considerable research effort has been made into attacking three issues with GP models: how to compute efficiently when the number of data is large; how to approximate the posterior when the likelihood is not Gaussian and how to estimate covariance function parameter posteriors. This paper simultaneously addresses these, using a variational approximation to the posterior which is sparse in support of the function but otherwise free-form. The result is a Hybrid Monte-Carlo sampling scheme which allows for a non-Gaussian approximation over the function values and covariance parameters simultaneously, with efficient computations based on inducing-point sparse GPs. Code to replicate each experiment in this paper is available at github.com/sparseMCMC.

1 Introduction

Gaussian process models are attractive for machine learning because of their flexible nonparametric nature. By combining a GP prior with different likelihoods, a multitude of machine learning tasks can be tackled in a probabilistic fashion [1]. There are three things to consider when using a GP model: approximation of the posterior function (especially if the likelihood is non-Gaussian), computation, storage and inversion of the covariance matrix, which scales poorly in the number of data; and estimation (or marginalization) of the covariance function parameters. A multitude of approximation schemes have been proposed for efficient computation when the number of data is large. Early strategies were based on retaining a sub-set of the data [2]. Snelson and Ghahramani [3] introduced an inducing point approach, where the model is augmented with additional variables, and Titsias [4] used these ideas in a variational approach. Other authors have introduced approximations based on the spectrum of the GP [5, 6], or which exploit specific structures within the covariance matrix [7, 8], or by making unbiased stochastic estimates of key computations [9]. In this work, we extend the variational inducing point framework, which we prefer for general applicability (no specific requirements are made of the data or covariance function), and because the variational inducing point approach can be shown to minimize the KL divergence to the posterior process [10].

To approximate the posterior function and covariance parameters, Markov chain Monte-Carlo (MCMC) approaches provide asymptotically exact approximations. Murray and Adams [11] and Filippone et al. [12] examine schemes which iteratively sample the function values and covariance parameters. Such sampling schemes require computation and inversion of the full covariance matrix at each iteration, making them unsuitable for large problems. Computation may be reduced somewhat by considering variational methods, approximating the posterior using some fixed family of distributions [13, 14, 15, 16, 1, 17], though many covariance matrix inversions are generally required. Recent works [18, 19, 20] have proposed inducing point schemes which can reduce the

Table 1: Existing variational approaches

Reference	$p(\mathbf{y} \mid \mathbf{f})$	**Sparse**	**Posterior**	**Hyperparam.**
Williams & Barber[21] [also 14, 17]	probit/logit	✗	Gaussian (assumed)	point estimate
Titsias [4]	Gaussian	✓	Gaussian (optimal)	point estimate
Chai [18]	softmax	✓	Gaussian (assumed)	point estimate
Nguyen and Bonilla [1]	any factorized	✗	Mixture of Gaussians	point estimate
Hensman et al. [20]	probit	✓	Gaussian (assumed)	point estimate
This work	any factorized	✓	free-form	free-form

computation required substantially, though the posterior is assumed Gaussian and the covariance parameters are estimated by (approximate) maximum likelihood. Table 1 places our work in the context of existing variational methods for GPs.

This paper presents a general inference scheme, with the only concession to approximation being the variational inducing point assumption. Non-Gaussian posteriors are permitted through MCMC, with the computational benefits of the inducing point framework. The scheme jointly samples the inducing-point representation of the function with the covariance function parameters; with sufficient inducing points our method approaches full Bayesian inference over GP values and the covariance parameters. We show empirically that the number of required inducing points is substantially smaller than the dataset size for several real problems.

2 Stochastic process posteriors

The model is set up as follows. We are presented with some data inputs $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ and responses $\mathbf{y} = \{y_n\}_{n=1}^N$. A latent function is assumed drawn from a GP with zero mean and covariance function $k(\mathbf{x}, \mathbf{x}')$ with (hyper-) parameters $\boldsymbol{\theta}$. Consistency of the GP means that only those points with data are considered: the latent vector $\mathbf{f}$ represents the values of the function at the observed points $\mathbf{f} = \{f(\mathbf{x}_n)\}_{n=1}^N$, and has conditional distribution $p(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{ff})$, where $\mathbf{K}_{ff}$ is a matrix composed of evaluating the covariance function at all pairs of points in $\mathbf{X}$. The data likelihood depends on the latent function values: $p(\mathbf{y} \mid \mathbf{f})$. To make a prediction for latent function value test points $\mathbf{f}^\star = \{f(\mathbf{x}^\star)\}_{\mathbf{x}^\star \in \mathbf{X}^\star}$, the posterior function values and parameters are integrated:

$$p(\mathbf{f}^\star \mid \mathbf{y}) = \iint p(\mathbf{f}^\star \mid \mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}, \boldsymbol{\theta} \mid \mathbf{y}) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\mathbf{f} \,. \tag{1}$$

In order to make use of the computational savings offered by the variational inducing point framework [4], we introduce additional input points to the function $\mathbf{Z}$ and collect the responses of the function at that point into the vector $\mathbf{u} = \{\mathbf{u}_m = f(\mathbf{z}_m)\}_{m=1}^M$. With some variational posterior $q(\mathbf{u}, \boldsymbol{\theta})$, new points are predicted similarly to the exact solution

$$q(\mathbf{f}^\star) = \iint p(\mathbf{f}^\star \mid \mathbf{u}, \boldsymbol{\theta}) q(\mathbf{u}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\mathbf{u} \,. \tag{2}$$

This makes clear that the approximation is a stochastic process in the same fashion as the true posterior: the length of the predictions vector $\mathbf{f}^\star$ is potentially unbounded, covering the whole domain.

To obtain a variational objective, first consider the support of $\mathbf{u}$ under the true posterior, and for $\mathbf{f}$ under the approximation. In the above, these points are subsumed into the prediction vector $\mathbf{f}^\star$: from here we shall be more explicit, letting $\mathbf{f}$ be the points of the process at $\mathbf{X}$, $\mathbf{u}$ be the points of the process at $\mathbf{Z}$ and $\mathbf{f}^\star$ be a large vector containing all other points of interest[1]. All of the free parameters of the model are then $\mathbf{f}^\star, \mathbf{f}, \mathbf{u}, \boldsymbol{\theta}$, and using a variational framework, we aim to minimize the Kullback-Leibler divergence between the approximate and true posteriors:

$$\mathcal{K} \triangleq \mathrm{KL}[q(\mathbf{f}^\star, \mathbf{f}, \mathbf{u}, \boldsymbol{\theta}) || p(\mathbf{f}^\star, \mathbf{f}, \mathbf{u}, \boldsymbol{\theta} \mid \mathbf{y})] = -\mathop{\mathbb{E}}_{q(\mathbf{f}^\star, \mathbf{f}, \mathbf{u}, \boldsymbol{\theta})} \left[\log \frac{p(\mathbf{f}^\star \mid \mathbf{u}, \mathbf{f}, \boldsymbol{\theta}) p(\mathbf{u} \mid \mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}, \boldsymbol{\theta} \mid \mathbf{y})}{p(\mathbf{f}^\star \mid \mathbf{u}, \mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f} \mid \mathbf{u}, \boldsymbol{\theta}) q(\mathbf{u}, \boldsymbol{\theta})} \right] \tag{3}$$

[1]The vector $\mathbf{f}^\star$ here is considered finite but large enough to contain any point of interest for prediction. The infinite case follows Matthews et al. [10], is omitted here for brevity, and results in the same solution.

where the conditional distributions for $\mathbf{f}^\star$ have been expanded to make clear that they are the same under the true and approximate posteriors, and $\mathbf{X}, \mathbf{Z}$ and $\mathbf{X}^\star$ have been omitted for clarity. Straightforward identities simplify the expression,

$$\begin{aligned}\mathcal{K} &= -\mathbb{E}_{q(\mathbf{f},\mathbf{u},\boldsymbol{\theta})}\left[\log \frac{p(\mathbf{u}\,|\,\mathbf{f},\boldsymbol{\theta})p(\mathbf{f}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})p(\mathbf{y}\,|\,\mathbf{f})/p(\mathbf{y})}{p(\mathbf{f}\,|\,\mathbf{u},\boldsymbol{\theta})q(\mathbf{u},\boldsymbol{\theta})}\right] \\ &= -\mathbb{E}_{q(\mathbf{f},\mathbf{u},\boldsymbol{\theta})}\left[\log \frac{p(\mathbf{u}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})p(\mathbf{y}\,|\,\mathbf{f})}{q(\mathbf{u},\boldsymbol{\theta})}\right] + \log p(\mathbf{y})\,,\end{aligned} \tag{4}$$

resulting in the variational inducing-point objective investigated by Titsias [4], aside from the inclusion of $\boldsymbol{\theta}$. This can be rearranged to give the following informative expression

$$\mathcal{K} = \text{KL}\left[q(\mathbf{u},\boldsymbol{\theta})||\frac{p(\mathbf{u}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})\,\exp\{\mathbb{E}_{p(\mathbf{f}\,|\,\mathbf{u},\boldsymbol{\theta})}[\log p(\mathbf{y}\,|\,\mathbf{f})]\}}{C}\right] - \log C + \log p(\mathbf{y}). \tag{5}$$

Here C is an intractable constant which normalizes the distribution and is independent of q. Minimizing the KL divergence on the right hand side reveals that the optimal variational distribution is

$$\log \hat{q}(\mathbf{u},\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{f}\,|\,\mathbf{u},\boldsymbol{\theta})}\left[\log p(\mathbf{y}\,|\,\mathbf{f})\right] + \log p(\mathbf{u}\,|\,\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log C. \tag{6}$$

For general likelihoods, since the optimal distribution does not take any particular form, we intend to sample from it using MCMC, thus combining the benefits of variationally-sparse Gaussian processes with a free-form posterior. Sampling is feasible using standard methods since $\log \hat{q}$ is computable up to a constant, using $\mathcal{O}(NM^2)$ computations. After completing this work, it was brought to our attention that a similar suggestion had been made in [22], though the idea was dismissed because "prediction in sparse GP models typically involves some additional approximations". Our presentation of the approximation consisting of the entire stochastic process makes clear that no additional approximations are required. To sample effectively, the following are proposed.

Whitening the prior Noting that the problem (6) appears similar to a standard GP for $\mathbf{u}$, albeit with an interesting 'likelihood', we make use of an ancillary augmentation $\mathbf{u} = \mathbf{R}\mathbf{v}$, with $\mathbf{R}\mathbf{R}^\top = \mathbf{K}_{uu}$, $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This results in the optimal variational distribution

$$\log \hat{q}(\mathbf{v},\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{f}\,|\,\mathbf{u}=\mathbf{R}\mathbf{v})}\left[\log p(\mathbf{y}\,|\,\mathbf{f})\right] + \log p(\mathbf{v}) + \log p(\boldsymbol{\theta}) - \log C \tag{7}$$

Previously [11, 12] this parameterization has been used with schemes which alternate between sampling the latent function values (represented by $\mathbf{v}$ or $\mathbf{u}$) and the parameters $\boldsymbol{\theta}$. Our scheme uses HMC across $\mathbf{v}$ and $\boldsymbol{\theta}$ jointly, whose effectiveness is examined throughout the experiment section.

Quadrature The first term in (6) is the expected log-likelihood. In the case of factorization across the data-function pairs, this results in N one-dimensional integrals. For Gaussian or Poisson likelihood these integrals are tractable, otherwise they can be approximated by Gauss-Hermite quadrature. Given the current sample $\mathbf{v}$, the expectations are computed w.r.t. $p(f_n\,|\,\mathbf{v},\boldsymbol{\theta}) = \mathcal{N}(\mu_n, \gamma_n)$, with:

$$\boldsymbol{\mu} = \mathbf{A}^\top\mathbf{v};\;\; \boldsymbol{\gamma} = \text{diag}(\mathbf{K}_{ff} - \mathbf{A}^\top\mathbf{A});\;\; \mathbf{A} = \mathbf{R}^{-1}\mathbf{K}_{uf};\;\; \mathbf{R}\mathbf{R}^\top = \mathbf{K}_{uu}, \tag{8}$$

where the kernel matrices $\mathbf{K}_{uf}, \mathbf{K}_{uu}$ are computed similarly to $\mathbf{K}_{ff}$, but over the pairs in $(\mathbf{X},\mathbf{Z}), (\mathbf{Z},\mathbf{Z})$ respectively. From here, one can compute the expected likelihood and it is subsequently straightforward to compute derivatives in terms of $\mathbf{K}_{uf}$, diag$(\mathbf{K}_{ff})$ and $\mathbf{R}$.

Reverse mode differentiation of Cholesky To compute derivatives with respect to $\boldsymbol{\theta}$ and $\mathbf{Z}$ we use reverse-mode differentiation (backpropagation) of the derivative through the Cholesky matrix decomposition, transforming $\partial \log \hat{q}(\mathbf{v},\boldsymbol{\theta})/\partial \mathbf{R}$ into $\partial \log \hat{q}(\mathbf{v},\boldsymbol{\theta})/\partial \mathbf{K}_{uu}$, and then $\partial \log \hat{q}(\mathbf{v},\boldsymbol{\theta})/\partial \boldsymbol{\theta}$. This is discussed by Smith [23], and results in a $\mathcal{O}(M^3)$ operation; an efficient Cython implementation is provided in the supplement.

3 Treatment of inducing point positions & inference strategy

A natural question is, what strategy should be used to select the inducing points $\mathbf{Z}$? In the original inducing point formulation [3], the positions $\mathbf{Z}$ were treated as parameters to be optimized. One could interpret them as parameters of the *approximate prior covariance* [24]. The variational formulation

[4] treats them as parameters of the variational approximation, thus protecting from over-fitting as they form part of the variational posterior. In this work, since we propose a Bayesian treatment of the model, we question whether it is feasible to treat $\mathbf{Z}$ in a Bayesian fashion.

Since $\mathbf{u}$ and $\mathbf{Z}$ are auxiliary parameters, the form of their distribution does not affect the marginals of the model. The term $p(\mathbf{u} \,|\, \mathbf{Z})$ has been defined by the consistency with the GP in order to preserve the posterior-process interpretation above (i.e. $\mathbf{u}$ should be points on the GP), but we are free to choose $p(\mathbf{Z})$. Omitting dependence on $\boldsymbol{\theta}$ for clarity, and choosing w.l.o.g. $q(\mathbf{u}, \mathbf{Z}) = q(\mathbf{u} \,|\, \mathbf{Z})q(\mathbf{Z})$, the bound on the marginal likelihood, similarly to (4) is given by

$$\mathcal{L} = \mathbb{E}_{p(\mathbf{f} \,|\, \mathbf{u}, \mathbf{Z})q(\mathbf{u} \,|\, \mathbf{Z})q(\mathbf{Z})} \left[\log \frac{p(\mathbf{y} \,|\, \mathbf{f})p(\mathbf{u} \,|\, \mathbf{Z})p(\mathbf{Z})}{q(\mathbf{u} \,|\, \mathbf{Z})q(\mathbf{Z})} \right] . \tag{9}$$

The bound can be maximized w.r.t $p(\mathbf{Z})$ by noting that the term only appears inside a (negative) KL divergence: $-\mathbb{E}_{q(\mathbf{Z})}[\log q(\mathbf{Z})/p(\mathbf{Z})]$. Substituting the optimal $p(\mathbf{Z}) = q(\mathbf{Z})$ reduces (9) to

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z})} \left[\mathbb{E}_{p(\mathbf{f} \,|\, \mathbf{u}, \mathbf{Z})q(\mathbf{u} \,|\, \mathbf{Z})} \left[\log \frac{p(\mathbf{y} \,|\, \mathbf{f})p(\mathbf{u} \,|\, \mathbf{Z})}{q(\mathbf{u} \,|\, \mathbf{Z})} \right] \right] , \tag{10}$$

which can now be optimized w.r.t. $q(\mathbf{Z})$. Since no entropy term appears for $q(\mathbf{Z})$, the bound is maximized when the distribution becomes a Dirac's delta. In summary, since we are free to choose a prior for $\mathbf{Z}$ which maximizes the amount of information captured by $\mathbf{u}$, the optimal distribution becomes $p(\mathbf{Z}) = q(\mathbf{Z}) = \delta(\mathbf{Z} - \hat{\mathbf{Z}})$. This formally motivates optimizing the inducing points $\mathbf{Z}$.

Derivatives for Z For completeness we also include the derivative of the free form objective with respect to the inducing point positions. Substituting the optimal distribution $\hat{q}(\mathbf{u}, \boldsymbol{\theta})$ into (4) to give $\hat{\mathcal{K}}$ and then differentiating we obtain

$$\frac{\partial \hat{\mathcal{K}}}{\partial \mathbf{Z}} = -\frac{\partial \log C}{\partial \mathbf{Z}} = -\mathbb{E}_{\hat{q}(\mathbf{v}, \boldsymbol{\theta})} \left[\frac{\partial}{\partial \mathbf{Z}} \mathbb{E}_{p(\mathbf{f} \,|\, \mathbf{u} = \mathbf{R}\mathbf{v})} [\log p(\mathbf{y} \,|\, \mathbf{f})] \right] . \tag{11}$$

Since we aim to draw samples from $\hat{q}(\mathbf{v}, \boldsymbol{\theta})$, evaluating this free form inducing point gradient using samples seems plausible but challenging. Instead we use the following strategy.

1. Fit a Gaussian approximation to the posterior. We follow [20] in fitting a Gaussian approximation to the posterior. The positions of the inducing points are initialized using k-means clustering of the data. The values of the latent function are represented by a mean vector (initialized randomly) and a lower-triangular matrix $\mathbf{L}$ forms the approximate posterior covariance as $\mathbf{L}\mathbf{L}^\top$. For large problems (such as the MNIST experiment), stochastic optimization using AdaDelta is used. Otherwise, LBFGS is used. After a few hundred iterations with the inducing points positions fixed, they are optimized in free-form alongside the variational parameters and covariance function parameters.
2. Initialize the model using the approximation. Having found a satisfactory approximation, the HMC strategy takes the optimized inducing point positions from the Gaussian approximation. The initial value of $\mathbf{v}$ is drawn from the Gaussian approximation, and the covariance parameters are initialized at the (approximate) MAP value.
3. Tuning HMC. The HMC algorithm has two free parameters to tune, the number of leapfrog steps and the step-length. We follow a strategy inspired by Wang et al. [25], where the number of leapfrog steps is drawn randomly from 1 to L_{max}, and Bayesian optimization is used to maximize the expected square jump distance (ESJD), penalized by $\sqrt{L_{max}}$. Rather than allow an adaptive (but convergent) scheme as [25], we run the optimization for 30 iterations of 30 samples each, and use the best parameters for a long run of HMC.
4. Run tuned HMC to obtain predictions. Having tuned the HMC, it is run for several thousand iterations to obtain a good approximation to $\hat{q}(\mathbf{v}, \boldsymbol{\theta})$. The samples are used to estimate the integral in equation (2). The following section investigates the effectiveness of the proposed sampling scheme.

4 Experiments

4.1 Efficient sampling using Hamiltonian Monte Carlo

This section illustrates the effectiveness of Hamiltonian Monte Carlo in sampling from $\hat{q}(\mathbf{v}, \boldsymbol{\theta})$. As already pointed out, the form assumed by the optimal variational distribution $\hat{q}(\mathbf{v}, \boldsymbol{\theta})$ in equation (6) resembles the joint distribution in a GP model with a non-Gaussian likelihood.

For a fixed $\boldsymbol{\theta}$, sampling $\mathbf{v}$ is relatively straightforward, and this can be done efficiently using HMC [12, 26, 27] or Elliptical Slice Sampling [28]. A well tuned HMC has been reported to be extremely efficient in sampling the latent variables, and this motivates our effort into trying to extend this efficiency to the sampling of hyper-parameters as well. This is also particularly appealing due to the convenience offered by the proposed representation of the model.

The problem of drawing samples from the posterior distribution over $\mathbf{v}, \boldsymbol{\theta}$ has been investigated in detail in [11, 12]. In these works, it has been advocated to alternate between the sampling of $\mathbf{v}$ and $\boldsymbol{\theta}$ in a Gibbs sampling fashion and condition the sampling of $\boldsymbol{\theta}$ on a suitably chosen transformation of the latent variables. For each likelihood model, we compare efficiency and convergence speed of the proposed HMC sampler with a Gibbs sampler where $\mathbf{v}$ is sampled using HMC and $\boldsymbol{\theta}$ is sampled using the Metropolis-Hastings algorithm. To make the comparison fair, we imposed the mass matrix in HMC and the covariance in MH to be isotropic, and any parameters of the proposal were tuned using Bayesian optimization. Unlike in the proposed HMC sampler, for the Gibbs sampler we did not penalize the objective function of the Bayesian optimization for large numbers of leapfrog steps, as in this case HMC proposals on the latent variables are computationally cheaper than those on the hyper-parameters. We report efficiency in sampling from $\hat{q}(\mathbf{v}, \boldsymbol{\theta})$ using Effective Sample Size (ESS) and Time Normalized (TN)-ESS. In the supplement we include convergence plots based on the Potential Scale Reduction Factor (PSRF) computed based on ten parallel chains; in these each chain is initialized from the VB solution and individually tuned using Bayesian optimization.

4.2 Binary Classification

We first use the *image* dataset [29] to investigate the benefits of the approach over a Gaussian approximation, and to investigate the effect of changing the number of inducing points, as well as optimizing the inducing points under the Gaussian approximation. The data are 18 dimensional: we investigated the effect of our approximation using both ARD (one lengthscale per dimension) and an isotropic RBF kernel. The data were split randomly into 1000/1019 train/test sets; the log predictive density over ten random splits is shown in Figure 1.

Following the strategy outlined above, we fitted a Gaussian approximation to the posterior, with $\mathbf{Z}$ initialized with k-means. Figure 1 investigates the difference in performance when $\mathbf{Z}$ is optimized using the Gaussian approximation, compared to just using k-means for $\mathbf{Z}$. Whilst our strategy is not guaranteed to find the global optimum, it is clear that it improves the performance.

The second part of Figure 1 shows the performance improvement of our sampling approach over the Gaussian approximation. We drew 10,000 samples, discarding the first 1000: we see a consistent improvement in performance once M is large enough. For small M, The Gaussian approximation appears to work very well. The supplement contains a similar Figure for the case where a single lengthscale is shared: there, the improvement of the MCMC method over the Gaussian approximation is smaller but consistent. We speculate that the larger gains for ARD are due to posterior uncertainty in the lengthscales, which is poorly represented by a point in the Gaussian/MAP approximation.

The ESS and TN-ESS are comparable between HMC and the Gibbs sampler. In particular, for 100 inducing points and the RBF covariance, ESS and TN-ESS for HMC are 11 and $1.0 \cdot 10^{-3}$ and for the Gibbs sampler are 53 and $5.1 \cdot 10^{-3}$. For the ARD covariance, ESS and TN-ESS for HMC are 14 and $5.1 \cdot 10^{-3}$ and for the Gibbs sampler are 1.6 and $1.5 \cdot 10^{-4}$. Convergence, however, seems to be faster for HMC, especially for the ARD covariance (see the supplement).

4.3 Log Gaussian Cox Processes

We apply our methods to Log Gaussian Cox processes [30]: doubly stochastic models where the rate of an inhomogeneous Poisson process is given by a Gaussian process. The main difficulty for inference lies in that the likelihood of the GP requires an integral over the domain, which is typically intractable. For low dimensional problems, this integral can be approximated on a grid; assuming that the GP is constant over the width of the grid leads to a factorizing Poisson likelihood for each of the grid points. Whilst some recent approaches allow for a grid-free approach [19], these usually require concessions in the model, such as an alternative link function, and do not approach full Bayesian inference over the covariance function parameters.

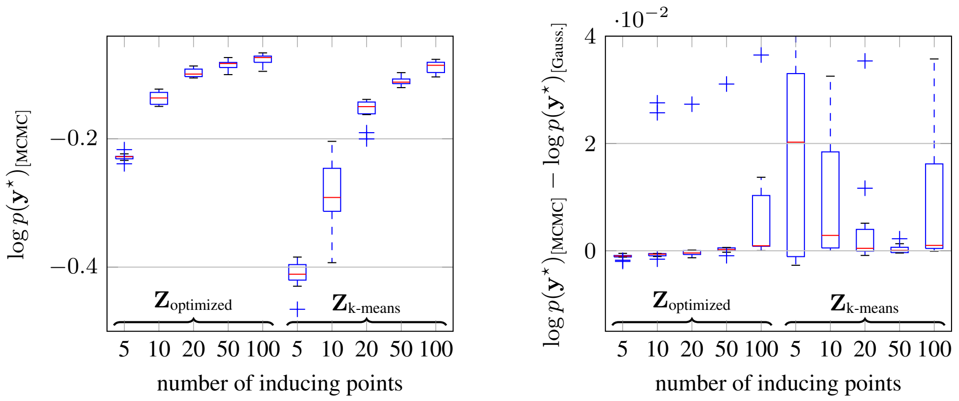

Figure 1: Performance of the method on the *image* dataset, with one lengthscale per dimension. Left, box-plots show performance for varying numbers of inducing points and $\mathbf{Z}$ strategies. Optimizing $\mathbf{Z}$ using the Gaussian approximation offers significant improvement over the k-means strategy. Right: improvement of the performance of the Gaussian approximation method, with the same inducing points. The method offers consistent performance gains when the number of inducing points is larger. The supplement contains a similar figure with only a single lengthscale.

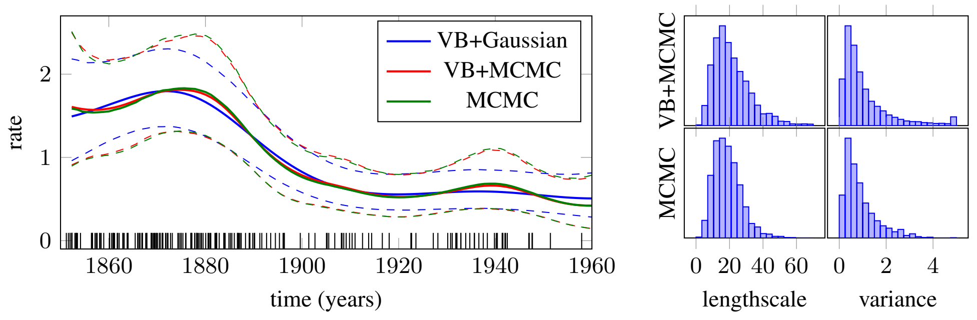

Figure 2: The posterior of the rates for the coal mining disaster data. Left: posterior rates using our variational MCMC method and a Gaussian approximation. Data are shown as vertical bars. Right: posterior samples for the covariance function parameters using MCMC. The Gaussian approximation estimated the parameters as (12.06, 0.55).

Coal mining disasters On the one-dimensional coal-mining disaster data. We held out 50% of the data at random, and using a grid of 100 points with 30 evenly spaced inducing points $\mathbf{Z}$, fitted both a Gaussian approximation to the posterior process with an (approximate) MAP estimate for the covariance function parameters (variance and lengthscale of an RBF kernel). With Gamma priors on the covariance parameters we ran our sampling scheme using HMC, drawing 3000 samples. The resulting posterior approximations are shown in Figure 2, alongside the true posterior using a sampling scheme similar to ours (but without the inducing point approximation). The free-form variational approximation matches the true posterior closely, whilst the Gaussian approximation misses important detail. The approximate and true posteriors over covariance function parameters are shown in the right hand part of Figure 2, there is minimal discrepancy in the distributions.

Over 10 random splits of the data, the average held-out log-likelihood was -1.229 for the Gaussian approximation and -1.225 for the free-form MCMC variant; the average difference was 0.003, and the MCMC variant was always better than the Gaussian approximation. We attribute this improved performance to marginalization of the covariance function parameters.

Efficiency of HMC is greater than for the Gibbs sampler; ESS and TN-ESS for HMC are 6.7 and $3.1 \cdot 10^{-2}$ and for the Gibbs sampler are 9.7 and $1.9 \cdot 10^{-2}$. Also, chains converge within few thousand iterations for both methods, although convergence for HMC is faster (see the supplement).

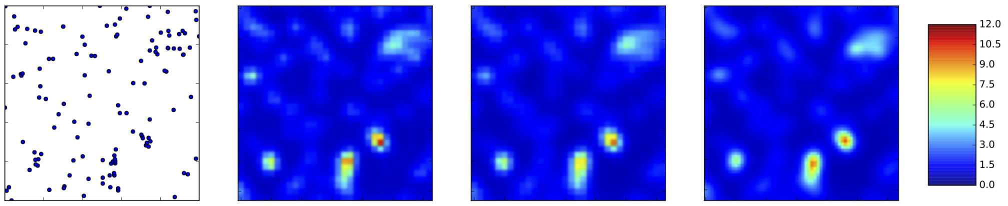

Figure 3: Pine sapling data. From left to right: reported locations of pine saplings; posterior mean intensity on a 32x32 grid using full MCMC; posterior mean intensity on a 32x32 grid (with sparsity using 225 inducing points), posterior mean intensity on a 64x64 grid (using 225 inducing points).

Pine saplings The advantages of the proposed approximation are prominent as the number of grid points become higher, an effect emphasized with increasing dimension of the domain. We fitted a similar model to the above to the pine sapling data [30].

We compared the sampling solution obtained using 225 inducing points on a 32 x 32 grid to the gold standard full MCMC run with the same prior and grid size. Figure 3 shows that the agreement between the variational sampling and full sampling is very close. However the variational method was considerably faster. Using a single core on a desktop computer required 3.4 seconds to obtain 1 effective sample for a well tuned variational method whereas it took 554 seconds for well tuned full MCMC. This effect becomes even larger as we increase the resolution of the grid to 64 x 64, which gives a better approximation to the underlying smooth function as can be seen in figure 3. It took 4.7 seconds to obtain one effective sample for the variational method, but now gold standard MCMC comparison was computationally extremely challenging to run for even a single HMC step. This is because it requires linear algebra operations using $\mathcal{O}(N^3)$ flops with $N = 4096$.

4.4 Multi-class Classification

To do multi-class classification with Gaussian processes, one latent function is defined for each of the classes. The functions are defined a-priori independent, but covary a posteriori because of the likelihood. Chai [18] studies a sparse variational approximation to the softmax multi-class likelihood restricted to a Gaussian approximation. Here, following [31, 32, 33], we use a robust-max likelihood. Given a vector $\mathbf{f}_n$ containing K latent functions evaluated at the point $\mathbf{x}_n$, the probability that the label takes the integer value y_n is $1 - \epsilon$ if $y_n = \operatorname{argmax} \mathbf{f}_n$ and $\epsilon/K - 1$ otherwise. As Girolami and Rogers [31] discuss, the 'soft' probit-like behaviour is recovered by adding a diagonal 'nugget' to the covariance function. In this work, ϵ was fixed to 0.001, though it would also be possible to treat this as a parameter for inference. The expected log-likelihood is $\mathbb{E}_{p(\mathbf{f}_n \,|\, \mathbf{v}, \boldsymbol{\theta})}[\log p(y_n \,|\, \mathbf{f}_n)] = p \log(\epsilon) + (1-p) \log(\epsilon/(K-1))$, where p is the probability that the labelled function is largest, which is computable using one-dimensional quadrature. An efficient Cython implementation is contained in the supplement.

Toy example To investigate the proposed posterior approximation for the multivariate classification case, we turn to the toy data shown in Figure 4. We drew 750 data points from three Gaussian distributions. The synthetic data was chosen to include non-linear decision boundaries and ambiguous decision areas. Figure 4 shows that there are differences between the variational and sampling solutions, with the sampling solution being more conservative in general (the contours of 95% confidence are smaller). As one would expect at the decision boundary there are strong correlations between the functions which could not be captured by the Gaussian approximation we are using. Note the movement of inducing points away from k-means and towards the decision boundaries.

Efficiency of HMC and the Gibbs sampler is comparable. In the RBF case, ESS and TN-ESS for HMC are 1.9 and $3.8 \cdot 10^{-4}$ and for the Gibbs sampler are 2.5 and $3.6 \cdot 10^{-4}$. In the ARD case, ESS and TN-ESS for HMC are 1.2 and $2.8 \cdot 10^{-3}$ and for the Gibbs sampler are 5.1 and $6.8 \cdot 10^{-4}$. For both cases, the Gibbs sampler struggles to reach convergence even though the average acceptance rates are similar to those recommended for the two samplers individually.

MNIST The MNIST dataset is a well studied benchmark with a defined training/test split. We used 500 inducing points, initialized from the training data using k-means. A Gaussian approximation

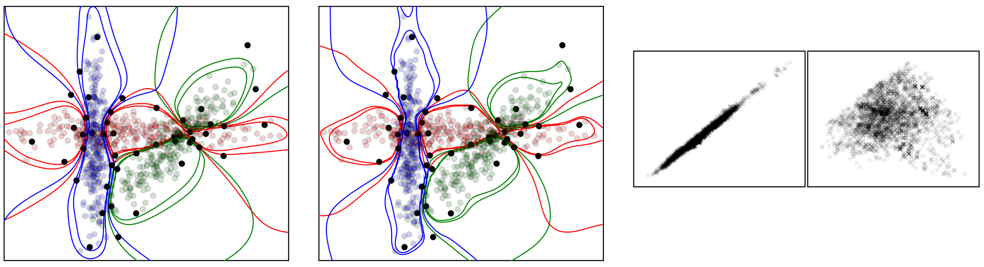
Figure 4: A toy multiclass problem. Left: the Gaussian approximation, colored points show the simulated data, lines show posterior probability contours at 0.3, 0.95, 0.99. Inducing points positions shows as black points. Middle: the free form solution with 10,000 posterior samples. The free-form solution is more conservative (the contours are smaller). Right: posterior samples for $\mathbf{v}$ at the same position but across different latent functions. The posterior exhibits strong correlations and edges.

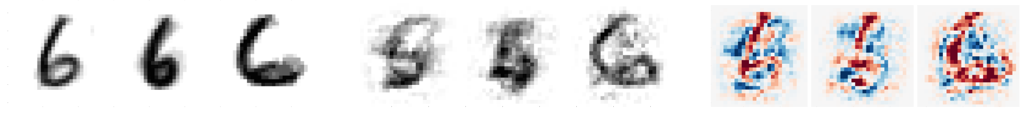
Figure 5: Left: three k-means centers used to initialize the inducing point positions. Center: the positions of the same inducing points after optimization. Right: difference.

was optimized using minibatch-based optimization over the means and variances of $q(\mathbf{u})$, as well as the inducing points and covariance function parameters. The accuracy on the held-out data was 98.04%, significantly improving on previous approaches to classify these digits using GP models.

For binary classification, Hensman et al. [20] reported that their Gaussian approximation resulted in movement of the inducing point positions toward the decision boundary. The same effect appears in the multivariate case, as shown in Figure 5, which shows three of the 500 inducing points used in the MNIST problem. The three examples were initialized close to the many six digits, and after optimization have moved close to other digits (five and four). The last example still appears to be a six, but has moved to a more 'unusual' six shape, supporting the function at another extremity. Similar effects are observed for all inducing-point digits. Having optimized the inducing point positions with the approximate $q(\mathbf{v})$, and estimate for $\boldsymbol{\theta}$, we used these optimal inducing points to draw samples from $\mathbf{v}$ and $\boldsymbol{\theta}$. This did not result in an increase in accuracy, but did improve the log-density on the test set from -0.068 to -0.064. Evaluating the gradients for the sampler took approximately 0.4 seconds on a desktop machine, and we were easily able to draw 1000 samples. This dataset size has generally be viewed as challenging in the GP community and consequently there are not many published results to compare with. One recent work [34] reports a 94.05% accuracy using variational inference and a GP latent variable model.

5 Discussion

We have presented an inference scheme for general GP models. The scheme significantly reduces the computational cost whilst approaching exact Bayesian inference, making minimal assumptions about the form of the posterior. The improvements in accuracy in comparison with the Gaussian approximation of previous works has been demonstrated, as has the quality of the approximation to the hyper-parameter distribution. Our MCMC scheme was shown to be effective for several likelihoods, and we note that the automatic tuning of the sampling parameters worked well over hundreds of experiments. This paper shows that MCMC methods are feasible for inference in large GP problems, addressing the unfair sterotype of 'slow' MCMC.

Acknowledgments JH was funded by an MRC fellowship, AM and ZG by EPSRC grant EP/I036575/1 and a Google Focussed Research award.

References

[1] T. V. Nguyen and E. V. Bonilla. Automated variational inference for Gaussian process models. In *NIPS*, pages 1404–1412, 2014.
[2] L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural comp.*, 14(3):641–668, 2002.
[3] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, pages 1257–1264, 2005.
[4] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, pages 567–574, 2009.
[5] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *JMLR*, 11:1865–1881, 2010.
[6] A. Solin and S. Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *arXiv preprint 1401.5508*, 2014.
[7] A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *NIPS*, pages 3626–3634. 2014.
[8] S. Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.
[9] M. Filippone and R. Engler. Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE). *ICML 2015*, 2015.
[10] A. G. D. G. Matthews, J. Hensman, R. E. Turner, and Z. Ghahramani. On sparse variational methods and the KL divergence between stochastic processes. *arXiv preprint 1504.07027*, 2015.
[11] I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *NIPS*, pages 1732–1740, 2010.
[12] M. Filippone, M. Zhong, and M. Girolami. A comparative evaluation of stochastic-based inference methods for Gaussian process models. *Mach. Learn.*, 93(1):93–114, 2013.
[13] M. N. Gibbs and D. J. C. MacKay. Variational Gaussian process classifiers. *IEEE Trans. Neural Netw.*, 11(6):1458–1464, 2000.
[14] M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural comp.*, 21(3): 786–792, 2009.
[15] M. Kuss and C. E. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *JMLR*, 6:1679–1704, 2005.
[16] H. Nickisch and C. E. Rasmussen. Approximations for binary Gaussian process classification. *JMLR*, 9: 2035–2078, 2008.
[17] E. Khan, S. Mohamed, and K. P. Murphy. Fast Bayesian inference for non-conjugate Gaussian process regression. In *NIPS*, pages 3140–3148, 2012.
[18] K. M. A. Chai. Variational multinomial logit Gaussian process. *JMLR*, 13(1):1745–1808, June 2012.
[19] C. Lloyd, T. Gunter, M. A. Osborne, and S. J. Roberts. Variational inference for Gaussian process modulated poisson processes. *ICML 2015*, 2015.
[20] J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. In *AISTATS*, pages 351–360, 2014.
[21] C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1342–1351, 1998.
[22] Michalis K Titsias, Neil Lawrence, and Magnus Rattray. Markov chain monte carlo algorithms for gaussian processes. In D. Barber, A. T. Chiappa, and S. Cemgil, editors, *Bayesian time series models*. 2011.
[23] S. P. Smith. Differentiation of the cholesky algorithm. *J. Comp. Graph. Stat.*, 4(2):134–147, 1995.
[24] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6:1939–1959, 2005.
[25] Z. Wang, S. Mohamed, and N. De Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo. In *ICML*, volume 28, pages 1462–1470, 2013.
[26] J. Vanhatalo and A. Vehtari. Sparse Log Gaussian Processes via MCMC for Spatial Epidemiology. In *Gaussian processes in practice*, volume 1, pages 73–89, 2007.
[27] O. F. Christensen, G. O. Roberts, and J. S. Rosenthal. Scaling limits for the transient phase of local MetropolisHastings algorithms. *JRSS:B*, 67(2):253–268, 2005.
[28] I. Murray, R. P. Adams, and D. J. C. MacKay. Elliptical slice sampling. In *AISTATS*, volume 9, 2010.
[29] G. Rätsch, T. Onoda, and K-R Müller. Soft margins for adaboost. *Mach. Learn.*, 42(3):287–320, 2001.
[30] J. Møller, A. R. Syversveen, and R. P. Waagepetersen. Log Gaussian Cox processes. *Scand. stat.*, 25(3): 451–482, 1998.
[31] M. Girolami and S. Rogers. Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Comp.*, 18:2006, 2005.
[32] H. Kim and Z. Ghahramani. Bayesian Gaussian Process Classification with the EM-EP Algorithm. *IEEE TPAMI*, 28(12):1948–1959, 2006.
[33] D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont. Robust multi-class Gaussian process classification. In *NIPS*, pages 280–288, 2011.
[34] Y. Gal, M. Van der Wilk, and Rasmussen C. E. Distributed variational inference in sparse Gaussian process regression and latent variable models. In *NIPS*. 2014.

Supplementary material for:

MCMC for Variationally Sparse GPs

5.1 Coal data

Figure 6 replicates Figure 1, but with a single lengthscale shared across each input.

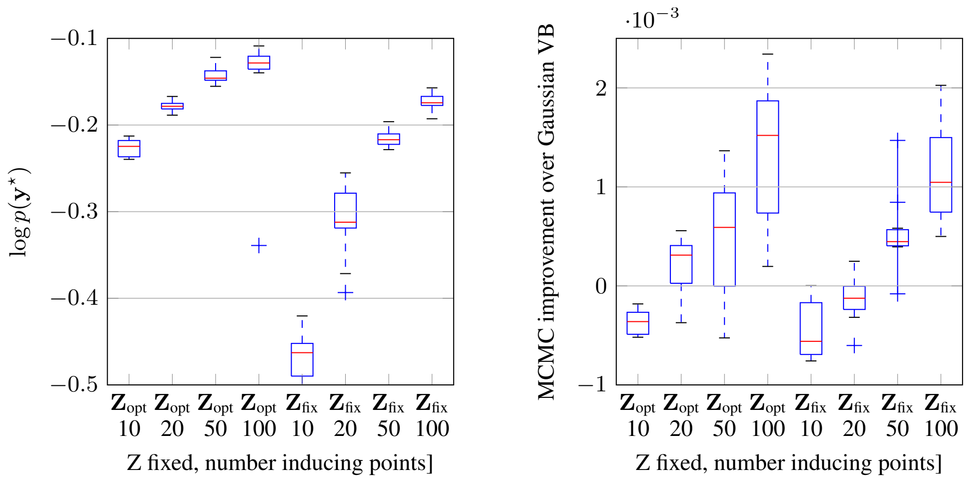

Figure 6: Performance of the method on the *image* dataset, with a single lengthscale.

5.2 Convergence plots

Convergence of the samplers on the Image dataset is reported in fig. 7 and shows the evolution of the PSRF for the twenty slowest parameters for HMC and the Gibbs sampler in the case of RBF and ARD covariances. The figure shows that HMC consistently converges faster than the Gibbs sampler for both covariances, even when the ESS of the slowest variable is comparable.

Fig. 7 shows the convergence analysis on the coal dataset. In this case, HMC converges faster than the Gibbs sampler and efficiency is comparable.

Convergence of the samplers on the toy multi-class dataset is reported in fig. 9. HMC converges much faster than the Gibbs sampler even though efficiency measured through ESS is comparable.

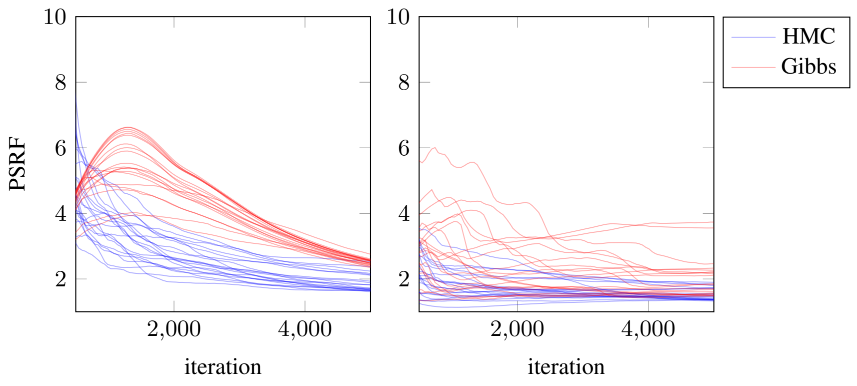

Figure 7: Image dataset - Evolution of the PSRF of the twenty least efficient parameter traces for HMC (blue) and the Gibbs sampler (red). Left panel: RBF case - minimum ESS and TN-ESS for HMC are 11 and $1.0 \cdot 10^{-3}$ and for the Gibbs sampler are 53 and $5.1 \cdot 10^{-3}$. Right panel: ARD case - minimum ESS and TN-ESS for HMC are 14 and $5.1 \cdot 10^{-3}$ and for the Gibbs sampler are 1.6 and $1.5 \cdot 10^{-4}$.

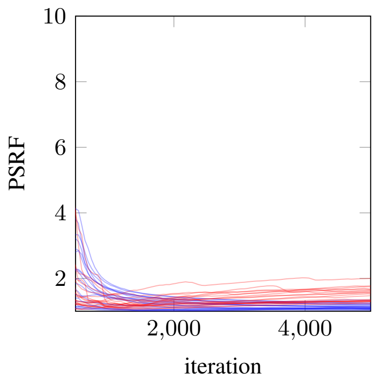

Figure 8: Coal dataset - Evolution of the PSRF of the twenty least efficient parameter traces for HMC (blue) and the Gibbs sampler (red). Minimum ESS and TN-ESS for HMC are 6.7 and $3.1 \cdot 10^{-2}$ and for the Gibbs sampler are 9.7 and $1.9 \cdot 10^{-2}$.

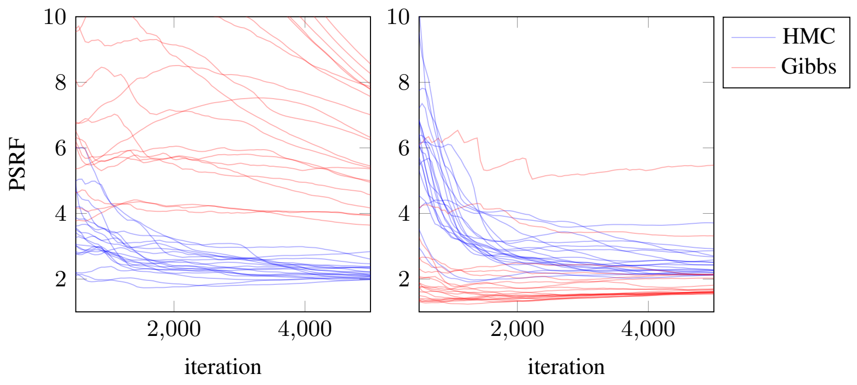

Figure 9: Multiclass dataset - Evolution of the PSRF of the twenty least efficient parameter traces for HMC (blue) and the Gibbs sampler (red). Left panel: RBF case - minimum ESS and TN-ESS for HMC are 1.9 and $3.8 \cdot 10^{-4}$ and for the Gibbs sampler are 2.5 and $3.6 \cdot 10^{-4}$. Right panel: ARD case - minimum ESS and TN-ESS for HMC are 1.2 and $2.8 \cdot 10^{-3}$ and for the Gibbs sampler are 5.1 and $6.8 \cdot 10^{-4}$.

5.3 Pine saplings

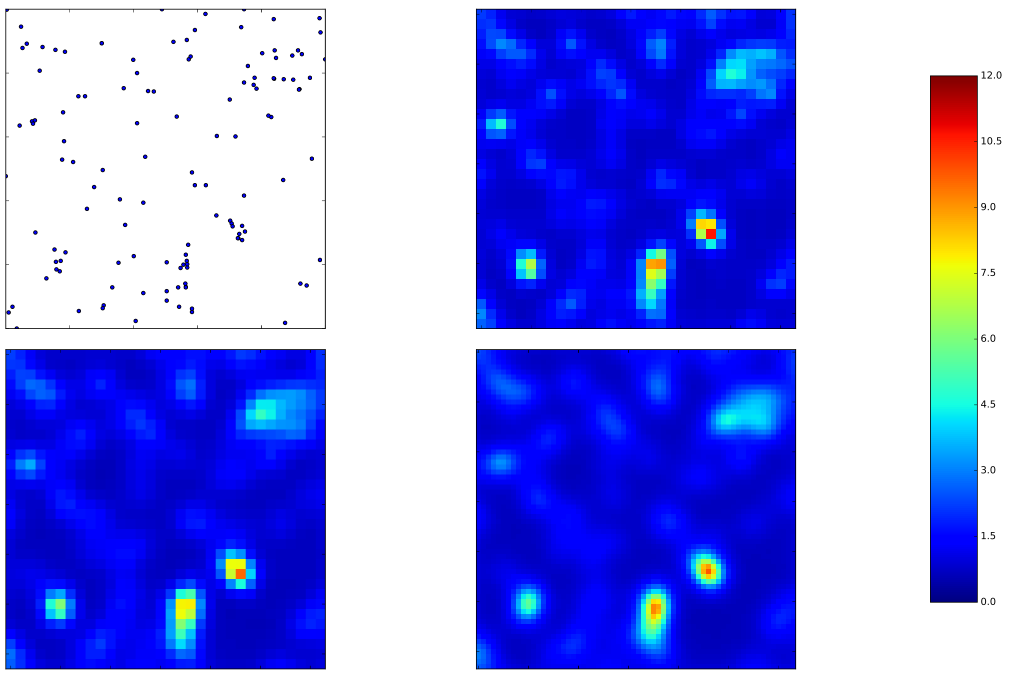

Figure 10: A larger version of Figure 3. Top right: gold standard MCMC 32x32 grid. Bottom left: Variational MCMC 32x32 grid. Bottom right: Variational MCMC 64x64 grid, with 225 inducing points in the non-exact case.

Random Feature Expansions for Deep Gaussian Processes

Kurt Cutajar [1] Edwin V. Bonilla [2] Pietro Michiardi [1] Maurizio Filippone [1]

Abstract

The composition of multiple Gaussian Processes as a Deep Gaussian Process (DGP) enables a deep probabilistic nonparametric approach to flexibly tackle complex machine learning problems with sound quantification of uncertainty. Existing inference approaches for DGP models have limited scalability and are notoriously cumbersome to construct. In this work we introduce a novel formulation of DGPs based on random feature expansions that we train using stochastic variational inference. This yields a practical learning framework which significantly advances the state-of-the-art in inference for DGPs, and enables accurate quantification of uncertainty. We extensively showcase the scalability and performance of our proposal on several datasets with up to 8 million observations, and various DGP architectures with up to 30 hidden layers.

1. Introduction

Given their impressive performance on machine learning and pattern recognition tasks, Deep Neural Networks (DNNs) have recently attracted a considerable deal of attention in several applied domains such as computer vision and natural language processing; see, e.g., LeCun et al. (2015) and references therein. Deep Gaussian Processes (DGPs; Damianou & Lawrence, 2013) alleviate the outstanding issue characterizing DNNs of having to specify the number of units in hidden layers by implicitly working with infinite representations at each layer. From a generative perspective, DGPs transform the inputs using a cascade of Gaussian Processes (GPs; Rasmussen & Williams, 2006) such that the output of each layer of GPs forms the input to the GPs at the next layer, effectively implementing a deep probabilistic nonparametric model for compositions of functions (Neal, 1996; Duvenaud et al., 2014).

Because of their probabilistic formulation, it is natural to approach the learning of DGPs through Bayesian inference techniques; however, the application of such techniques to learn DGPs leads to various forms of intractability. A number of contributions have been proposed to recover tractability, extending or building upon the literature on approximate methods for GPs. Nevertheless, only few works leverage one of the key features that arguably make DNNs so successful, that is being scalable through the use of mini-batch-based learning (Hensman & Lawrence, 2014; Dai et al., 2016; Bui et al., 2016). Even among these works, there does not seem to be an approach that is truly applicable to large-scale problems, and practical beyond only a few hidden layers.

In this paper, we develop a practical learning framework for DGP models that significantly improves the state-of-the-art on those aspects. In particular, our proposal introduces two sources of approximation to recover tractability, while (i) scaling to large-scale problems, (ii) being able to work with moderately deep architectures, and (iii) being able to accurately quantify uncertainty. The first is a model approximation, whereby the GPs at all layers are approximated using random feature expansions (Rahimi & Recht, 2008); the second approximation relies upon stochastic variational inference to retain a probabilistic and scalable treatment of the approximate DGP model.

We show that random feature expansions for DGP models yield Bayesian DNNs with low-rank weight matrices, and the expansion of different covariance functions results in different DNN activation functions, namely trigonometric for the Radial Basis Function (RBF) covariance, and Rectified Linear Unit (ReLU) functions for the ARC-COSINE covariance. In order to retain a probabilistic treatment of the model we adapt the work on variational inference for DNNs and variational autoencoders (Graves, 2011; Kingma & Welling, 2014) using mini-batch-based stochastic gradient optimization, which can exploit GPU and distributed computing. In this respect, we can view the probabilistic treatment of DGPs approximated through random feature expansions as a means to specify sensible and interpretable priors for probabilistic DNNs. Furthermore, unlike popular inducing points-based approximations for DGPs, the resulting learning framework does not involve any matrix decompositions in the size of the number of inducing points,

[1]Department of Data Science, EURECOM, France [2]School of Computer Science and Engineering, University of New South Wales, Australia. Correspondence to: Kurt Cutajar <kurt.cutajar@eurecom.fr>, Pietro Michiardi <pietro.michiardi@eurecom.fr>, Edwin V. Bonilla Cutajar <e.bonilla@unsw.edu.au>, Maurizio Filippone <maurizio.filippone@eurecom.fr>.

but only matrix products. We implement our model in TensorFlow (Abadi et al., 2015), which allows us to rely on automatic differentiation to apply stochastic variational inference.

Although having to select the appropriate number of random features goes against the nonparametric formulation favored in GP models, the level of approximation can be tuned based on constraints on running time or hardware. Most importantly, the random feature approximation enables us to develop a learning framework for DGPs which significantly advances the state-of-the-art. We extensively demonstrate the effectiveness of our proposal on a variety of regression and classification problems by comparing it with DNNs and other state-of-the-art approaches to infer DGPs. The results indicate that for a given DGP architecture, our proposal is consistently faster at achieving lower errors compared to the competitors. Another key observation is that the proposed DGP outperforms DNNs trained with dropout on quantification of uncertainty metrics.

We focus part of the experiments on large-scale problems, such as MNIST8M digit classification and the AIRLINE dataset, which contain over 8 and 5 million observations, respectively. Only very recently there have been attempts to demonstrate performance of GP models on such large data sets (Wilson et al., 2016; Krauth et al., 2017), and our proposal is on par with these latest GP methods. Furthermore, we obtain impressive results when employing our learning framework to DGPs with moderate depth (few tens of layers) on the AIRLINE dataset. We are not aware of any other DGP models having such depth that can achieve comparable performance when applied to datasets with millions of observations. Crucially, we obtain all these results by running our algorithm on a single machine without GPUs, but our proposal is designed to be able to exploit GPU and distributed computing to significantly accelerate our deep probabilistic learning framework (see supplement for experiments in distributed mode).

In summary, the most significant contributions of this work are as follows: (i) we propose a novel approximation of DGPs based on random feature expansions that we study in connection with DNNs; (ii) we demonstrate the ability of our proposal to systematically outperform state-of-the-art methods to carry out inference in DGP models, especially for large-scale problems and moderately deep architectures; (iii) we validate the superior quantification of uncertainty offered by DGPs compared to DNNs.

1.1. Related work

Following the original proposal of DGP models in Damianou & Lawrence (2013), there have been several attempts to extend GP inference techniques to DGPs. Notable examples include the extension of inducing point approximations (Hensman & Lawrence, 2014; Dai et al., 2016) and Expectation Propagation (Bui et al., 2016). Sequential inference for training DGPs has also been investigated in Wang et al. (2016). A recent example of a DGP "natively" formulated as a variational model appears in Tran et al. (2016). Our work is the first to employ random feature expansions to approximate DGPs as DNNs. The expansion of the squared exponential covariance for DGPs leads to trigonometric DNNs, whose properties were studied in Sopena et al. (1999). Meanwhile, the expansion of the arc-cosine covariance is inspired by Cho & Saul (2009), and it allows us to show that DGPs with such covariance can be approximated with DNNs having ReLU activations.

The connection between DGPs and DNNs has been pointed out in several papers, such as Neal (1996) and Duvenaud et al. (2014), where pathologies with deep nets are investigated. The approximate DGP model described in our work becomes a DNN with low-rank weight matrices, which have been used in, e.g., Novikov et al. (2015); Sainath et al. (2013); Denil et al. (2013) as a regularization mechanism. Dropout is another technique to speed-up training and improve generalization of DNNs that has recently been linked to variational inference (Gal & Ghahramani, 2016).

Random Fourier features for large scale kernel machines were proposed in Rahimi & Recht (2008), and their application to GPs appears in Lázaro-Gredilla et al. (2010). In the case of squared exponential covariances, variational learning of the posterior over the frequencies was proposed in Gal & Turner (2015) to avoid potential overfitting caused by optimizing these variables. These approaches are special cases of our DGP model when using no hidden layers.

In our work, we learn the proposed approximate DGP model using stochastic variational inference. Variational learning for DNNs was first proposed in Graves (2011), and later extended to include the reparameterization trick to clamp randomness in the computation of the gradient with respect to the posterior over the weights (Kingma & Welling, 2014; Rezende et al., 2014), and to include a Gaussian mixture prior over the weights (Blundell et al., 2015).

2. Preliminaries

Consider a supervised learning scenario where a set of input vectors $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ is associated with a set of (possibly multivariate) labels $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top$, where $\mathbf{x}_i \in R^{D_{\text{in}}}$ and $\mathbf{y}_i \in R^{D_{\text{out}}}$. We assume that there is an underlying function $f_o(\mathbf{x}_i)$ characterizing a mapping from the inputs to a latent representation, and that the labels are a realization of some probabilistic process $p(y_{io}|f_o(\mathbf{x}_i))$ which is based on this latent representation.

In this work, we consider modeling the latent functions using Deep Gaussian Processes (DGPs; Damianou &

Lawrence, 2013). Let variables in layer l be denoted by the (l) superscript. In DGP models, the mapping between inputs and labels is expressed as a composition of functions

$$\mathbf{f}(\mathbf{x}) = \left(\mathbf{f}^{(N_\mathrm{h}-1)} \circ \ldots \circ \mathbf{f}^{(0)}\right)(\mathbf{x}),$$

where each of the N_h layers, is composed of a (possibly transformed) multivariate Gaussian process (GP). Formally, a GP is a collection of random variables such that any subset of these are jointly Gaussian distributed (Rasmussen & Williams, 2006). In GPs, the covariance between variables at different inputs is modeled using the so-called *covariance function*.

Given the relationship between GPs and single-layered neural networks with an infinite number of hidden units (Neal, 1996), the DGP model has an obvious connection with DNNs. In contrast to DNNs, where each of the hidden layers implements a parametric function of its inputs, in DGPs these functions are assigned a GP prior, and are therefore nonparametric. Furthermore, because of their probabilistic formulation, it is natural to approach the learning of DGPs through Bayesian inference techniques that lead to principled approaches for both determining the optimal settings of architecture-dependent parameters, such as the number of hidden layers, and quantification of uncertainty.

While DGPs are attractive from a theoretical standpoint, inference is extremely challenging. Denote by $F^{(l)}$ the set of latent variables with entries $f_{io}^{(l)} = f_o^{(l)}(\mathbf{x}_i)$, and let $p(Y|F^{(N_\mathrm{h})})$ be the conditional likelihood. Learning and making predictions with DGPs requires solving integrals that are generally intractable. For example, computing the marginal likelihood to optimize covariance parameters $\boldsymbol{\theta}^{(l)}$ at all layers entails solving

$$p(Y|X,\boldsymbol{\theta}) = \int p\left(Y|F^{(N_\mathrm{h})}\right) p\left(F^{(N_\mathrm{h})}|F^{(N_\mathrm{h}-1)}, \boldsymbol{\theta}^{(N_\mathrm{h}-1)}\right) \times \ldots \times p\left(F^{(1)}|X,\boldsymbol{\theta}^{(0)}\right) dF^{(N_\mathrm{h})} \ldots dF^{(1)}.$$

In the following section we use random feature approximations to the covariance function in order to develop a scalable algorithm for inference in DGPs.

2.1. Random Feature Expansions for GPs

We start by describing how random feature expansions can be used to approximate the covariance of a single GP model. Such approximations have been considered previously, for example by Rahimi & Recht (2008) in the context of non-probabilistic kernel machines. Here we focus on random feature expansions for the radial basis function (RBF) covariance and the ARC-COSINE covariance, which we will use in our experiments.

For the sake of clarity, we will present the covariances without any explicit scaling of the features or the covariance itself. After explaining the random feature expansion associated with each covariance, we will generalize these results in the context of DGPs to include scaling the covariance by a factor σ^2, and scaling the features for Automatic Relevance Determination (ARD) (Mackay, 1994).

2.1.1. Radial Basis Function Covariance

A popular example of a covariance function, which we consider here, is the Radial Basis Function (RBF) covariance

$$k_\mathrm{rbf}(\mathbf{x},\mathbf{x}') = \exp\left[-\frac{1}{2}\left\|\mathbf{x}-\mathbf{x}'\right\|^\top\right]. \quad (1)$$

Appealing to Bochner's theorem, any continuous shift-invariant normalized covariance function $k(\mathbf{x}_i,\mathbf{x}_j) = k(\mathbf{x}_i - \mathbf{x}_j)$ is positive definite if and only if it can be rewritten as the Fourier transform of a non-negative measure $p(\boldsymbol{\omega})$ (Rahimi & Recht, 2008). Denoting the spectral frequencies by $\boldsymbol{\omega}$, while assigning $\iota = \sqrt{-1}$ and $\boldsymbol{\delta} = \mathbf{x}_i - \mathbf{x}_j$, in the case of the RBF covariance in equation 1, this yields:

$$k_\mathrm{rbf}(\boldsymbol{\delta}) = \int p(\boldsymbol{\omega}) \exp\left(\iota\boldsymbol{\delta}^\top\boldsymbol{\omega}\right) d\boldsymbol{\omega}, \quad (2)$$

with a corresponding non-negative measure $p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, I)$. Because the covariance function and the non-negative measures are real, we can drop the unnecessary complex part of the argument of the expectation, keeping $\cos(\boldsymbol{\delta}^\top\boldsymbol{\omega}) = \cos((\mathbf{x}_i - \mathbf{x}_j)^\top\boldsymbol{\omega})$ that can be rewritten as $\cos(\mathbf{x}_i^\top\boldsymbol{\omega})\cos(\mathbf{x}_j^\top\boldsymbol{\omega}) + \sin(\mathbf{x}_i^\top\boldsymbol{\omega})\sin(\mathbf{x}_j^\top\boldsymbol{\omega})$.

The importance of the expansion above is that it allows us to interpret the covariance function as an expectation that can be estimated using Monte Carlo. Defining $\mathbf{z}(\mathbf{x}|\boldsymbol{\omega}) = [\cos(\mathbf{x}^\top\boldsymbol{\omega}), \sin(\mathbf{x}^\top\boldsymbol{\omega})]^\top$, the covariance function can be therefore unbisedly approximated as

$$k_\mathrm{rbf}(\mathbf{x}_i,\mathbf{x}_j) \approx \frac{1}{N_\mathrm{RF}} \sum_{r=1}^{N_\mathrm{RF}} \mathbf{z}(\mathbf{x}_i|\tilde{\boldsymbol{\omega}}_r)^\top \mathbf{z}(\mathbf{x}_j|\tilde{\boldsymbol{\omega}}_r), \quad (3)$$

with $\tilde{\boldsymbol{\omega}}_r \sim p(\boldsymbol{\omega})$. This has an important practical implication, as it provides the means to access an approximate explicit representation of the mapping induced by the covariance function that, in the RBF case, we know is infinite dimensional (Shawe-Taylor & Cristianini, 2004). Various results have been established on the accuracy of the random Fourier feature approximation; see, e.g., Rahimi & Recht (2008).

2.1.2. Arc-cosine Covariance

We also consider the ARC-COSINE covariance of order p, which is defined as:

$$k_\mathrm{arc}^{(p)}(\mathbf{x},\mathbf{x}') = \frac{1}{\pi}\left(\|\mathbf{x}\|\,\|\mathbf{x}'\|\right)^p J_p\left(\cos^{-1}\left(\frac{\mathbf{x}^\top\mathbf{x}'}{\|\mathbf{x}\|\|\mathbf{x}'\|}\right)\right), \quad (4)$$

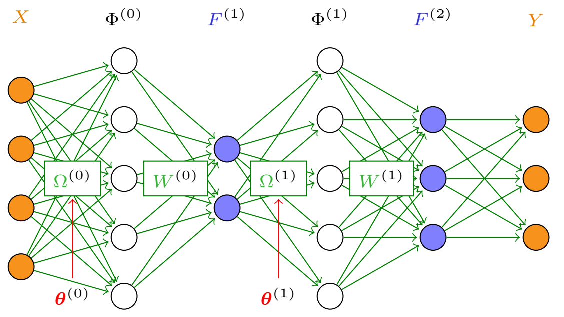

Figure 1. The proposed DGP approximation. At each hidden layer GPs are replaced by their two-layer weight-space approximation. Random-features $\Phi^{(l)}$ are obtained using a weight matrix $\Omega^{(l)}$. This is followed by a linear transformation parameterized by weights $W^{(l)}$. The prior over $\Omega^{(l)}$ is determined by the covariance parameters $\boldsymbol{\theta}^{(l)}$ of the original GPs.

where we have defined

$$J_p(\alpha) = (-1)^p (\sin\alpha)^{2p+1} \left(\frac{1}{\sin\alpha}\frac{\partial}{\partial\alpha}\right)^p \left(\frac{\pi-\alpha}{\sin\alpha}\right).$$

Let $H(\cdot)$ be the Heaviside function. Following Cho & Saul (2009), an integral representation of this covariance is:

$$k_{\text{arc}}^{(p)}(\mathbf{x},\mathbf{x}') = 2\int H(\boldsymbol{\omega}^\top \mathbf{x}) \left(\boldsymbol{\omega}^\top \mathbf{x}\right)^p H(\boldsymbol{\omega}^\top \mathbf{x}') \left(\boldsymbol{\omega}^\top \mathbf{x}'\right)^p \times \mathcal{N}(\boldsymbol{\omega}|0,I)d\boldsymbol{\omega}. \quad (5)$$

This integral formulation immediately suggests a random feature approximation for the ARC-COSINE covariance in equation (4), noting that it can be seen as an expectation of the product of the same function applied to the inputs to the covariance. As before, this provides an approximate explicit representation of the mapping induced by the covariance function. Interestingly, for the ARC-COSINE covariance of order $p = 1$, this yields an approximation based on popular Rectified Linear Unit (ReLU) functions. We note that for the the ARC-COSINE covariance with degree $p = 0$, the resulting Heaviside activations are unsuitable for our inference scheme, given that they yield systematically zero gradients.

3. Random Feature Expansions for DGPs

In this section, we present our approximate formulation of DGPs which, as we illustrate in the experiments, leads to a practical learning algorithm for these deep probabilistic nonparametric models. We propose to employ the random feature expansion at each layer, and by doing so we obtain an approximation to the original DGP model as a DNN (Figure 1).

Assume that the GPs have zero mean, and define $F^{(0)} := X$. Also, assume that the GP covariances at each layer are parameterized through a set of parameters $\boldsymbol{\theta}^{(l)}$. The parameter set $\boldsymbol{\theta}^{(l)}$ comprises the layer-wise GP marginal variances $(\sigma^2)^{(l)}$ and lengthscale parameters $\Lambda^{(l)} = \operatorname{diag}((\ell_1^2)^{(l)}, \ldots, (\ell_{D_F^{(l)}}^2)^{(l)})$.

Considering a DGP with RBF covariances, taking a "weight-space view" of the GPs at each layer, and extending the results in the previous section, we have that

$$\Phi_{\text{rbf}}^{(l)} = \sqrt{\frac{(\sigma^2)^{(l)}}{N_{\text{RF}}^{(l)}}} \left[\cos\left(F^{(l)}\Omega^{(l)}\right), \sin\left(F^{(l)}\Omega^{(l)}\right)\right], \quad (6)$$

and $F^{(l+1)} = \Phi_{\text{rbf}}^{(l)} W^{(l)}$. At each layer, the priors over the weights are $p\left(\Omega_{\cdot j}^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, \left(\Lambda^{(l)}\right)^{-1}\right)$ and $p\left(W_{\cdot i}^{(l)}\right) = \mathcal{N}(\mathbf{0}, I)$.

Each matrix $\Omega^{(l)}$ has dimensions $D_{F^{(l)}} \times N_{\text{RF}}^{(l)}$. On the other hand, the weight matrices $W^{(l)}$ have dimensions $2N_{\text{RF}}^{(l)} \times D_{F^{(l+1)}}$ (weighting of sine and cosine random features), with the constraint that $D_{F^{(N_\text{h})}} = D_{\text{out}}$.

Similarly, considering a DGP with ARC-COSINE covariances of order $p = 1$, the application of the random feature approximation leads to DNNs with ReLU activations:

$$\Phi_{\text{arc}}^{(l)} = \sqrt{\frac{2(\sigma^2)^{(l)}}{N_{\text{RF}}^{(l)}}} \max\left(0, F^{(l)}\Omega^{(l)}\right), \quad (7)$$

with $\Omega_{\cdot j}^{(l)} \sim \mathcal{N}\left(\mathbf{0}, \left(\Lambda^{(l)}\right)^{-1}\right)$, which are cheaper to evaluate and differentiate than the trigonometric functions required in the RBF case. As in the RBF case, we allowed the covariance and the features to be scaled by $(\sigma^2)^{(l)}$ and $\Lambda^{(l)}$, respectively. The dimensions of the weight matrices $\Omega^{(l)}$ are the same as in the RBF case, but the dimensions of the $W^{(l)}$ matrices are $N_{\text{RF}}^{(l)} \times D_{F^{(l+1)}}$.

3.1. Low-rank weights in the resulting DNN

Our formulation of an approximate DGP using random feature expansions reveals a close connection with DNNs. In our formulation, the design matrices at each layer are $\Phi^{(l+1)} = \gamma\left(\Phi^{(l)} W^{(l)} \Omega^{(l+1)}\right)$, where $\gamma(\cdot)$ denotes the element-wise application of covariance-dependent functions, i.e., sine and cosine for the RBF covariance and ReLU for the ARC-COSINE covariance. Instead, for the DNN case, the design matrices are computed as $\Phi^{(l+1)} = g(\Phi^{(l)}\Omega^{(l)})$, where $g(\cdot)$ is a so-called activation function. In light of this, we can view our approximate DGP model as a DNN. From a probabilistic standpoint, we can interpret our approximate DGP model as a DNN with specific Gaussian priors over the $\Omega^{(l)}$ weights controlled by the covariance parameters $\boldsymbol{\theta}^{(l)}$, and standard Gaussian priors over the $W^{(l)}$ weights. Covariance parameters act as hyper-priors over the weights $\Omega^{(l)}$, and the objective is to optimize these during training.

Another observation about the resulting DGP approximation is that, for a given layer l, the transformations given by $W^{(l)}$ and $\Omega^{(l+1)}$ are both linear. If we collapsed the two transformations into a single one, by introducing weights $\Xi^{(l)} = W^{(l)}\Omega^{(l+1)}$, we would have to learn $O\left(N_{RF}^{(l)} \times N_{RF}^{(l+1)}\right)$ weights at each layer, which is considerably more than learning the two separate sets of weights. As a result, we can view the proposed approximate DGP model as a way to impose a low-rank structure on the weights of DNNs, which is a form of regularization proposed in the literature of DNNs (Novikov et al., 2015; Sainath et al., 2013; Denil et al., 2013).

3.2. Variational inference

In order to keep the notation uncluttered, let $\mathbf{\Theta}$ be the collection of all covariance parameters $\boldsymbol{\theta}^{(l)}$ at all layers. Also, consider the case of a DGP with fixed spectral frequencies $\Omega^{(l)}$ collected into $\mathbf{\Omega}$, and let $\mathbf{W}$ be the collection of the weight matrices $W^{(l)}$ at all layers. For $\mathbf{W}$ we have a product of standard normal priors stemming from the approximation of the GPs at each layer $p(\mathbf{W}) = \prod_{l=0}^{N_{\mathrm{h}}-1} p(W^{(l)})$, and we propose to treat $\mathbf{W}$ using variational inference following Kingma & Welling (2014) and Graves (2011), and optimize all covariance parameters $\mathbf{\Theta}$. We will consider $\mathbf{\Omega}$ to be fixed here, but we will discuss alternative ways to treat $\mathbf{\Omega}$ in the next section. In the supplement we also assess the quality of the variational approximation over $\mathbf{W}$, with $\mathbf{\Omega}$ and $\mathbf{\Theta}$ fixed, by comparing it with MCMC techniques.

The marginal likelihood $p(Y|X, \mathbf{\Omega}, \mathbf{\Theta})$ involves intractable integrals, but we can obtain a tractable lower bound using variational inference. Defining $\mathcal{L} = \log\left[p(Y|X, \mathbf{\Omega}, \mathbf{\Theta})\right]$ and $\mathcal{E} = \mathrm{E}_{q(\mathbf{W})}\left(\log\left[p\left(Y|X, \mathbf{W}, \mathbf{\Omega}, \mathbf{\Theta}\right)\right]\right)$, we obtain

$$\mathcal{L} \geq \mathcal{E} - \mathrm{DKL}\left[q(\mathbf{W}) \| p\left(\mathbf{W}\right)\right], \tag{8}$$

where $q(\mathbf{W})$ acts as an approximation to the posterior over all the weights $p(\mathbf{W}|Y, X, \mathbf{\Omega}, \mathbf{\Theta})$.

We are interested in optimizing $q(\mathbf{W})$, i.e. finding an optimal approximate distribution over the parameters according to the bound above. The first term can be interpreted as a model fitting term, whereas the second as a regularization term. In the case of a Gaussian distribution $q(\mathbf{W})$ and a Gaussian prior $p(\mathbf{W})$, it is possible to compute the DKL term analytically (see supplementary material), whereas the remaining term needs to be estimated. Assume a Gaussian approximating distribution that factorizes across layers and weights:

$$q(\mathbf{W}) = \prod_{ijl} q\left(W_{ij}^{(l)}\right) = \prod_{ijl} \mathcal{N}\left(m_{ij}^{(l)}, (s^2)_{ij}^{(l)}\right). \tag{9}$$

The variational parameters are the mean and the variance of each of the approximating factors $m_{ij}^{(l)}, (s^2)_{ij}^{(l)}$, and we aim to optimize the lower bound with respect to these as well as all covariance parameters $\mathbf{\Theta}$.

In the case of a likelihood that factorizes across observations, an interesting feature of the expression of the lower bound is that it is amenable to fast stochastic optimization. In particular, we derive a doubly-stochastic approximation of the expectation in the lower bound as follows. First, $\mathcal{E}$ can be rewritten as a sum over the input points, which allows us to estimate it in an unbiased fashion using mini-batches, selecting m points from the entire dataset:

$$\mathcal{E} \approx \frac{n}{m} \sum_{k \in \mathcal{I}_m} \mathrm{E}_{q(\mathbf{W})}(\log[p(\mathbf{y}_k|\mathbf{x}_k, \mathbf{W}, \mathbf{\Omega}, \mathbf{\Theta})]). \tag{10}$$

Second, each of the elements of the sum can be estimated using Monte Carlo, yielding:

$$\mathcal{E} \approx \frac{n}{m} \sum_{k \in \mathcal{I}_m} \frac{1}{N_{\mathrm{MC}}} \sum_{r=1}^{N_{\mathrm{MC}}} \log[p(\mathbf{y}_k|\mathbf{x}_k, \tilde{\mathbf{W}}_r, \mathbf{\Omega}, \mathbf{\Theta})], \tag{11}$$

with $\tilde{\mathbf{W}}_r \sim q(\mathbf{W})$. In order to facilitate the optimization, we reparameterize the weights as follows:

$$(\tilde{W}_r^{(l)})_{ij} = s_{ij}^{(l)} \epsilon_{rij}^{(l)} + m_{ij}^{(l)}. \tag{12}$$

By differentiating the lower bound with respect to $\mathbf{\Theta}$ and the mean and variance of the approximate posterior over $\mathbf{W}$, we obtain an unbiased estimate of the gradient for the lower bound. The reparameterization trick ensures that the randomness in the computation of the expectation is fixed when applying stochastic gradient ascent moves to parameters of $q(\mathbf{W})$ and $\mathbf{\Theta}$ (Kingma & Welling, 2014). Automatic differentiation tools enabled us to compute stochastic gradients automatically, which is why we opted to implement our model in TensorFlow (Abadi et al., 2015).

3.3. Treatment of the spectral frequencies $\mathbf{\Omega}$

So far, we have assumed the spectral frequencies $\mathbf{\Omega}$ to be sampled from the prior and fixed throughout, whereby we employ the reparameterization trick to obtain $\Omega_{ij}^{(l)} = (\beta^2)_{ij}^{(l)} \varepsilon_{rij}^{(l)} + \mu_{ij}^{(l)}$, with $(\beta^2)_{ij}^{(l)}$ and $\mu_{ij}^{(l)}$ determined by the prior $p\left(\Omega_{\cdot j}^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, \left(\Lambda^{(l)}\right)^{-1}\right)$. We then draw the $\varepsilon_{rij}^{(l)}$'s and fix them from the outset, such that covariance parameters $\mathbf{\Theta}$ can be optimized along with $q(\mathbf{W})$. We refer to this variant as PRIOR-FIXED.

Inspired by previous work on random feature expansions for GPs, we can think of alternative ways to treat these parameters, e.g., Lázaro-Gredilla et al. (2010); Gal & Turner (2015). In particular, we study a variational treatment of $\mathbf{\Omega}$; we refer the reader to the supplementary material for details on the derivation of the lower bound in this case.

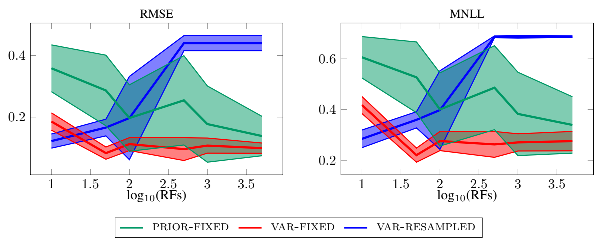

Figure 2. Performance of different strategies for dealing with $\mathbf{\Omega}$ as a function of the number of random features. These can be fixed (PRIOR-FIXED), or treated variationally (with fixed randomness VAR-FIXED and resampled at each iteration VAR-RESAMPLED).

When being variational about $\mathbf{\Omega}$ we introduce an approximate posterior $q(\mathbf{\Omega})$ which also has a factorized form. We use the reparameterization trick once again, but the coefficients $(\beta^2)_{ij}^{(l)}$ and $\mu_{ij}^{(l)}$ to compute $\Omega_{ij}^{(l)}$ are now determined by $q(\mathbf{\Omega})$. We report two variations of this treatment, namely VAR-FIXED and VAR-RESAMPLED. In VAR-FIXED, we fix $\varepsilon_{rij}^{(l)}$ in computing $\mathbf{\Omega}$ throughout the learning of the model, whereas in VAR-RESAMPLED we resample these at each iteration. We note that one can also be variational about $\mathbf{\Theta}$, but we leave this for future work.

In Figure 2, we illustrate the differences between the strategies discussed in this section; we report the accuracy of the proposed one-layer DGP with RBF covariances with respect to the number of random features on one of the datasets that we consider in the experiment section (EEG dataset). For PRIOR-FIXED, more random features result in a better approximation of the GP priors at each layer, and this results in better generalization. When we resample $\mathbf{\Omega}$ from the approximate posterior (VAR-RESAMPLED), we notice that the model quickly struggles with the optimization when increasing the number of random features. We attribute this to the fact that the factorized form of the posterior over $\mathbf{\Omega}$ and $\mathbf{W}$ is unable to capture posterior correlations between the coefficients for the random features and the weights of the corresponding linearized model. Being deterministic about the way spectral frequencies are computed (VAR-FIXED) offers the best performance among the three learning strategies, and this is what we employ throughout the rest of this paper.

3.4. Computational complexity

When estimating the lower bound, there are two main operations performed at each layer, that is $F^{(l)}\Omega^{(l)}$ and $\Phi^{(l)}W^{(l)}$. Recalling that this matrix product is done for samples from the posterior over $\mathbf{W}$ (and $\mathbf{\Omega}$ when treated variationally) and given the mini-batch formulation, the former costs $\mathcal{O}\left(mD_F^{(l)}N_{\mathrm{RF}}^{(l)}N_{\mathrm{MC}}\right)$, while the latter costs $\mathcal{O}\left(mN_{\mathrm{RF}}^{(l)}D_F^{(l)}N_{\mathrm{MC}}\right)$.

Because of feature expansions and stochastic variational inference, the resulting algorithm does not involve any Cholesky decompositions. This is in sharp contrast with stochastic variational inference using inducing-point approximations (see e.g. Dai et al., 2016; Bui et al., 2016), where such operations could significantly limit the number of inducing points that can be employed.

4. Experiments

We evaluate our model by comparing it against relevant alternatives for both regression and classification, and assess its performance when applied to large-scale datasets. We also investigate the extent to which such deep compositions continue to yield good performance when the number of layers is significantly increased.

4.1. Model Comparison

We primarily compare our model to the state-of-the-art DGP inference method presented in the literature, namely DGPs using expectation propagation (DGP-EP; Bui et al., 2016). We originally intended to include results for the variational auto-encoded DGP (Damianou & Lawrence, 2013); however, the results obtained using the available code were not competitive with DGP-EP and we thus decided to exclude them from the figures. We also omitted DGP training using sequential inference (Wang et al., 2016) given that we could not find an implementation of the method and, in any case, the performance reported in the paper is inferior to more recent approaches. We also compare against DNNs in order to present the results in a wider context, and demonstrate that DGPs lead to better quantification of uncertainty. Finally, to substantiate the benefits of using a deep model, we compare against the shallow sparse variational GP (Hensman et al., 2015b) implemented in GPflow (Matthews et al., 2016).

We use the same experimental set-up for both regression and classification tasks using datasets from the UCI repository (Asuncion & Newman, 2007), for models having one hidden layer. The results for architectures with two hidden layers are included in the supplementary material. The specific configurations for each model are detailed below:

DGP-RBF, DGP-ARC : In the proposed DGP with an RBF kernel, we use 100 random features at every hidden layer to construct a multivariate GP with $D_F^{(l)} = 3$, and set the batch size to $m = 200$. We initially only use a single Monte Carlo sample, and halfway through the allocated optimization time, this is then increased to 100 samples. We employ the Adam optimizer with a learning rate of 0.01, and in order to stabilize the optimization procedure, we fix the parameters $\mathbf{\Theta}$ for 12,000 iterations, before jointly optimizing all parameters. As discussed in Sec-

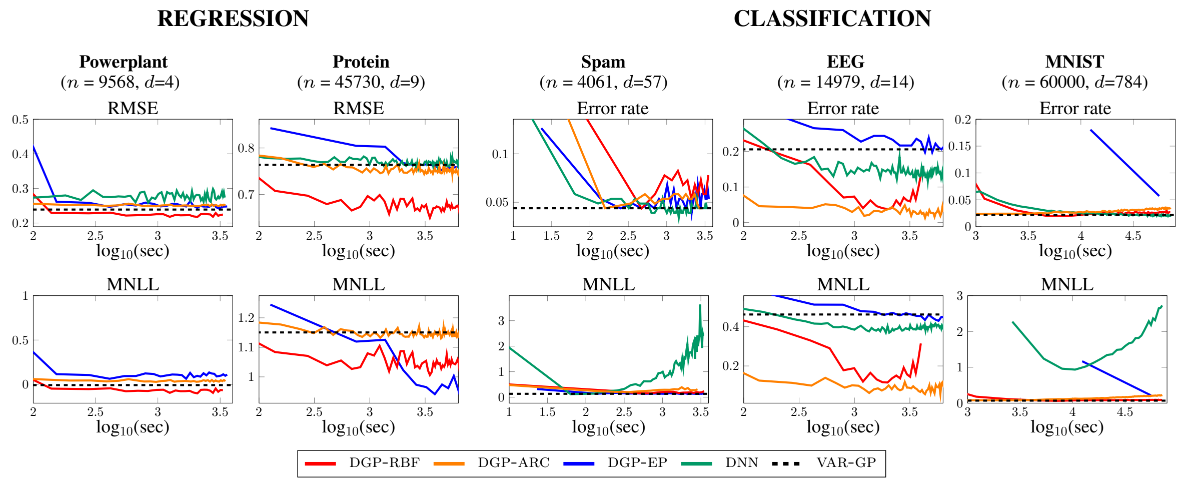

Figure 3. Progression of error rate (RMSE in the regression case) and MNLL over time for competing models. Results are shown for configurations having 1 hidden layer, while the results for models having 2 such layers may be found in the supplementary material.

tion 3.3, Ω are optimized variationally with fixed randomness. The same set-up is used for DGP-ARC, the variation of our model implementing the ARC-COSINE kernel;

DGP-EP [1]: For this technique, we use the same architecture and optimizer as for DGP-RBF and DGP-ARC, a batch size of 200 and 100 inducing points at each hidden layer. For the classification case, we use 100 samples for approximating the Softmax likelihood;

DNN : We construct a DNN configured with a dropout rate of 0.5 at each hidden layer in order to provide regularization during training. In order to preserve a degree of fairness, we set the number of hidden units in such a way as to ensure that the number of weights to be optimized match those in the DGP-RBF and DGP-ARC models when the random features are taken to be fixed.

We assess the performance of each model using the error rate (RMSE in the regression case) and mean negative log-likelihood (MNLL) on withheld test data. The results are averaged over 3 folds for every dataset. The experiments were launched on single nodes of a cluster of Intel Xeon E5-2630 CPUs having 32 cores and 128GB RAM.

Figure 3 shows that DGP-RBF and DGP-ARC consistently outperform competing techniques both in terms of convergence speed and predictive accuracy. This is particularly significant for larger datasets where other techniques take considerably longer to converge to a reasonable error rate, although DGP-EP converges to superior MNLL for the PROTEIN dataset. The results are also competitive (and sometimes superior) to those obtained by the variational GP (VAR-GP) in Hensman et al. (2015b). It is striking to see how inferior uncertainty quantification provided by the DNN (which is inherently limited to the classification case, so no MNLL reported on regression datasets) is compared to DGPs, despite the error rate being comparable.

By virtue of its higher dimensionality, larger configurations were used for MNIST. For DGP-RBF and DGP-ARC, we use 500 random features, 50 GPs in the hidden layers, batch size of 1000, and Adam with a 0.001 learning rate. Similarly for DGP-EP, we use 500 inducing points, with the only difference being a slightly smaller batch size to cater for issues with memory requirements. Following Simard et al. (2003), we employ 800 hidden units at each layer of the DNN. The DGP-RBF peaks at 98.04% and 97.93% for 1 and 2 hidden layers respectively. It was observed that the model performance degrades noticeably when more than 2 hidden layers are used (without feeding forward the inputs). This is in line with what is reported in the literature on DNNs (Neal, 1996; Duvenaud et al., 2014). By simply re-introducing the original inputs in the hidden layer, the accuracy improves to 98.2% for the one hidden layer case.

Recent experiments on MNIST using a variational GP with MCMC report overall accuracy of 98.04% (Hensman et al., 2015a), while the AutoGP architecture has been shown to give 98.45% accuracy (Krauth et al., 2017). Using a finer-tuned configuration, DNNs were also shown to obtain 98.5% accuracy (Simard et al., 2003), whereas 98.6% has been reported for SVMs (Schölkopf, 1997). In view of this wider scope of inference techniques, it can be confirmed that the results obtained using the proposed architecture are comparable to the state-of-the-art, even if further extensions may be required for obtaining a proper edge. Note that this comparison focuses on approaches without preprocessing, and excludes convolutional neural nets.

[1]Code obtained from: github.com/thangbui/deepGP_approxEP

Table 1. Performance of our proposal on large-scale datasets.

Dataset	Accuracy		MNLL	
	RBF	ARC	RBF	ARC
MNIST8M	99.14%	99.04%	0.0454	0.0465
AIRLINE	78.55%	72.76%	0.4583	0.5335

4.2. Large-scale Datasets

One of the defining characteristics of our model is the ability to scale up to large datasets without compromising on performance and accuracy in quantifying uncertainty. As a demonstrative example, we evaluate our model on two large-scale problems which go beyond the scale of datasets to which GPs and especially DGPs are typically applied.

We first consider MNIST8M, which artificially extends the original MNIST dataset to 8+ million observations. We trained this model using the same configuration described for standard MNIST, and we obtained 99.14% accuracy on the test set using one hidden layer. Given the size of this dataset, there are only few reported results for other GP models. Most notably, Krauth et al. (2017) recently obtained 99.11% accuracy with the AutoGP framework, which is comparable to the result obtained by our model.

Meanwhile, the AIRLINE dataset contains flight information for 5+ million flights in the US between Jan and Apr 2008. Following the procedure described in Hensman et al. (2013) and Wilson et al. (2016), we use this 8-dimensional dataset for classification, where the task is to determine whether a flight has been delayed or not. We construct the test set using the scripts provided in Wilson et al. (2016), where $100,000$ data points are held-out for testing. We construct our DGP models using 100 random features at each layer, and set the dimensionality to $D_{F^{(l)}} = 3$. As shown in Table 1, our model works significantly better when the RBF kernel is employed. In addition, the results are also directly comparable to those obtained by Wilson et al. (2016), which reports accuracy and MNLL of 78.1% and 0.457, respectively. These results give further credence to the tractability, scalability, and robustness of our model.

4.3. Model Depth

In this final part of the experiments, we assess the scalability of our model with respect to additional hidden layers in the constructed model. In particular, we re-consider the AIRLINE dataset and evaluate the performance of DGP-RBF models constructed using up to 30 layers. In order to cater for the increased depth in the model, we feed-forward the original input to each hidden layer, as suggested in Duvenaud et al. (2014).

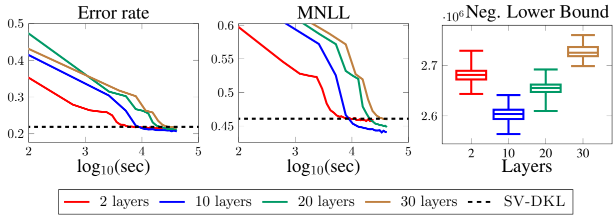

Figure 4. Left and central panels - Performance of our model on the AIRLINE dataset as function of time for different depths. The baseline (SV-DKL) is taken from Wilson et al. (2016). Right panel - The box plot of the negative lower bound, estimated over 100 mini-batches of size $50,000$, confirms that this is a suitable objective for model selection.

Figure 4 reports the progression of error rate and MNLL over time for different number of hidden layers, using the results obtained in Wilson et al. (2016) as a baseline (reportedly obtained in about 3 hours). As expected, the model takes longer to train as the number of layers increases. However, the model converges to an optimal state in every case in less than a couple of hours, with an improvement being noted in the case of 10 and 20 layers over the shallower 2-layer model. The box plot within the same figure indicates that the negative lower bound is a suitable objective function for carrying out model selection.

5. Conclusions

In this work, we have proposed a novel formulation of DGPs which exploits the approximation of covariance functions using random features, as well as stochastic variational inference for preserving the probabilistic representation of a regular GP. We demonstrated how inference using this model is not only faster, but also frequently superior to other state-of-the-art methods, with particular emphasis on competing DGP models. The results obtained for both the AIRLINE dataset and the MNIST8M digit recognition problem are particularly impressive since such large datasets have been generally considered to be beyond the computational scope of DGPs. We perceive this to be a considerable step forward in the direction of scaling and accelerating DGPs.

The results obtained on higher-dimensional datasets strongly suggest that approximations such as Fastfood (Le et al., 2013) could be instrumental in the interest of using more random features. We are also currently investigating ways to mitigate the decline in performance observed when optimizing $\mathbf{\Omega}$ variationally with resampling. The obtained results also encourage the extension of our model to include residual learning or convolutional layers suitable for computer vision applications.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

Asuncion, Arthur and Newman, David J. UCI machine learning repository, 2007.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight Uncertainty in Neural Network. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1613–1622. JMLR.org, 2015.

Bui, Thang D., Hernández-Lobato, Daniel, Hernández-Lobato, José M., Li, Yingzhen, and Turner, Richard E. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1472–1481. JMLR.org, 2016.

Chen, Jianmin, Monga, Rajat, Bengio, Samy, and Jozefowicz, Rafal. Revisiting distributed synchronous sgd. `https://arxiv.org/abs/1604.00981`, 2016.

Chilimbi, T., Suzue, Y., Apacible, J., and Kalyanaraman, K. Project adam: Building an efficient and scalable deep learning training system. In *USENIX Symposium on Operating Systems Design and Implementation, October 6-8, 2014, Broomfield, Colorado, USA*, 2014.

Cho, Youngmin and Saul, Lawrence K. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pp. 342–350, 2009.

Dai, Zhenwen, Damianou, Andreas, González, Javier, and Lawrence, Neil. Variational auto-encoded deep gaussian processes. In *Proceedings of the Fourth International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2-4 May, 2016*, 2016.

Damianou, Andreas C. and Lawrence, Neil D. Deep Gaussian Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Proceedings*, pp. 207–215. JMLR.org, 2013.

Denil, Misha, Shakibi, Babak, Dinh, Laurent, Ranzato, Marc'Aurelio, and de Freitas, Nando. Predicting Parameters in Deep Learning. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 2148–2156, 2013.

Duvenaud, David K., Rippel, Oren, Adams, Ryan P., and Ghahramani, Zoubin. Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, pp. 202–210. JMLR.org, 2014.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1050–1059. JMLR.org, 2016.

Gal, Yarin and Turner, Richard. Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 655–664. JMLR.org, 2015.

Graves, Alex. Practical Variational Inference for Neural Networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2348–2356. Curran Associates, Inc., 2011.

Hensman, James and Lawrence, Neil D. Nested Variational Compression in Deep Gaussian Processes, December 2014.

Hensman, James, Fusi, Nicoló, and Lawrence, Neil D. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*, 2013.

Hensman, James, de G. Matthews, Alexander G., Filippone, Maurizio, and Ghahramani, Zoubin. MCMC for variationally sparse gaussian processes. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1648–1656, 2015a.

Hensman, James, de G. Matthews, Alexander G., and Ghahramani, Zoubin. Scalable variational Gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, pp. 351–360, 2015b.

Kingma, Diederik P. and Welling, Max. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations, ICLR 2014, Banff, Canada, April 14-16, 2014*, 2014.

Krauth, Karl, Bonilla, Edwin V., Cutajar, Kurt, and Filippone, Maurizio. AutoGP: Exploring the capabilities and limitations of Gaussian process models. In *AISTATS*, 2017.

Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

Le, Quoc V., Sarls, Tams, and Smola, Alexander J. Fastfood - computing hilbert space expansions in loglinear time. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 244–252. JMLR.org, 2013.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

Mackay, D. J. C. Bayesian methods for backpropagation networks. In Domany, E., van Hemmen, J. L., and Schulten, K. (eds.), *Models of Neural Networks III*, chapter 6, pp. 211–254. Springer, 1994.

Matthews, Alexander G. de G., van der Wilk, Mark, Nickson, Tom, Fujii, Keisuke., Boukouvalas, Alexis, León-Villagrá, Pablo, Ghahramani, Zoubin, and Hensman, James. GPflow: A Gaussian process library using TensorFlow. *arXiv preprint 1610.08733*, October 2016.

Murray, Iain, Adams, Ryan P., and MacKay, David J. C. Elliptical slice sampling. *Journal of Machine Learning Research - Proceedings Track*, 9:541–548, 2010.

Neal, Radford M. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, August 1996. ISBN 0387947248.

Novikov, Alexander, Podoprikhin, Dmitry, Osokin, Anton, and Vetrov, Dmitry P. Tensorizing Neural Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 442–450, 2015.

Rahimi, Ali and Recht, Benjamin. Random Features for Large-Scale Kernel Machines. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1177–1184. Curran Associates, Inc., 2008.

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

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286. JMLR.org, 2014.

Sainath, Tara N., Kingsbury, Brian, Sindhwani, Vikas, Arisoy, Ebru, and Ramabhadran, Bhuvana. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pp. 6655–6659. IEEE, 2013. doi: 10.1109/ICASSP.2013.6638949.

Schölkopf, Bernhard. *Support vector learning*. PhD thesis, Berlin Institute of Technology, 1997.

Shawe-Taylor, John and Cristianini, Nello. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

Simard, Patrice Y., Steinkraus, Dave, and Platt, John C. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, Washington, DC, USA, 2003. IEEE Computer Society.

Sopena, J. M., Romero, E., and Alquezar, R. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, 1999. doi: 10.1049/cp:19991129.

Tran, Dustin, Ranganath, Rajesh, and Blei, David M. The Variational Gaussian Process. In *Proceedings of the Fourth International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2-4 May, 2016*, 2016.

Wang, Yali, Brubaker, Marcus A., Chaib-draa, Brahim, and Urtasun, Raquel. Sequential inference for deep gaussian process. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, pp. 694–703, 2016.

Wilson, Andrew Gordon, Hu, Zhiting, Salakhutdinov, Ruslan, and Xing, Eric P. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2586–2594, 2016.

A. Additional Experiments

Using the experimental set-up described in Section 4, Figure 5 demonstrates how the competing models perform with regards to the RMSE (or error rate) and MNLL metric when two hidden layers are incorporated into the competing models. The results follow a similar progression to those reported in Figure 3 of the main paper. The DGP-ARC and DGP-RBF models both continue to perform well after introducing this additional layer. However, the results for the regularized DNN are notably inferior, and the degree of overfitting is also much greater. To this end, the MNLL obtained for the MNIST dataset is not shown in the plot as it was vastly inferior to the values obtained using the other methods. DGP-EP was also observed to have low scalability in this regard whereby it was not possible to obtain sensible results for the MNIST dataset using this configuration.

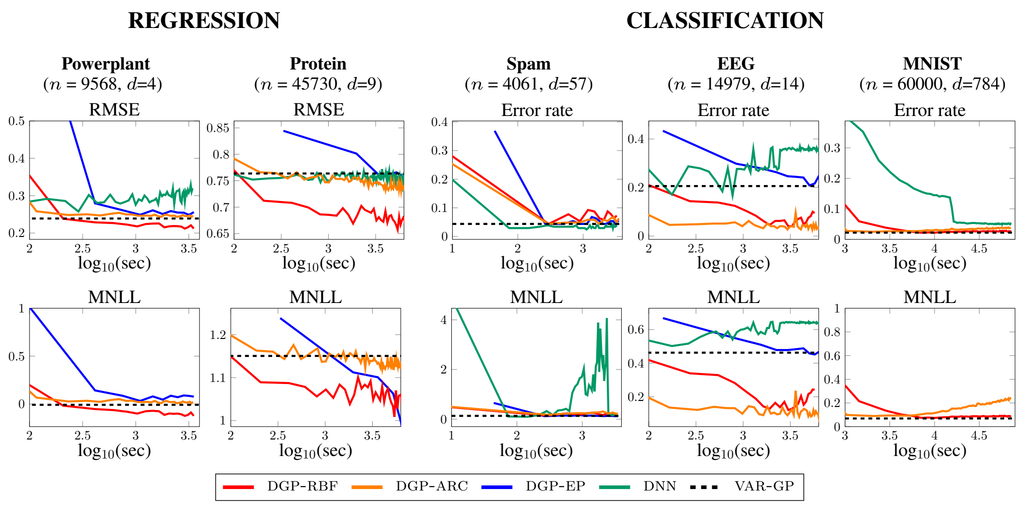

Figure 5. Progression of RMSE and MNLL over time for competing models. Results are shown for configurations having 2 hidden layers. There is no plot for DGP-EP on MNIST because the model did not produce sensible results within the allocated time.

In Section 3.3, we outlined the different strategies for treating $\mathbf{\Omega}$, namely fixing them or treating them variationally, where we observed that the constructed DGP model appears to work best when these are treated variationally while fixing the randomness in their computation throughout the learning process (VAR-FIXED). In Figures 6 and 7, we compare these three approaches on the complete set of datasets reported in the main experiments for one and two hidden layers, respectively. Once again, we confirm that the performance obtained using the VAR-FIXED strategy yields more consistent results than the alternatives. This is especially pertinent to the classification datasets, where the obtained error rate is markedly superior. However, the variation of the model constructed with the ARC-COSINE kernel and optimized using VAR-FIXED appears to be susceptible to some overfitting for higher dimensional datasets (SPAM and MNIST), which is expected given that we are optimizing several covariance parameters (ARD). This would motivate trying to be variational about $\mathbf{\Theta}$ too.

B. Comparison with MCMC

Figure 8 shows a comparison between the variational approximation and MCMC for a two-layer DGP model applied to a regression dataset. The dataset has been generated by drawing 50 data points from $\mathcal{N}(y|h(h(x)), 0.01)$, with $h(x) = 2x\exp(-x^2)$. We experiment with two different levels of precision in the DGP approximation by using 10 and 50 fixed spectral frequencies, respectively, so as to assess the impact on the number of random features on the results. For MCMC, covariance parameters are fixed to the values obtained by optimizing the variational lower bound on the marginal likelihood in the case of 50 spectral frequencies.

We obtained samples from the posterior over the latent variables at each layer using MCMC techniques. In the case of a Gaussian likelihood, it is possible to integrate out the GP at the last layer, thus obtaining a model that only depends on the GP at the first. As a result, the collapsed DGP model becomes a standard GP model whose latent variables can be sampled using various MCMC samplers developed in the literature of MCMC for GPs. Here we employ Elliptical Slice Sampling (Murray et al., 2010) to draw samples from the posterior over the latent variables at the first layer, whereas latent variables at the second can be sampled directly from a multivariate Gaussian distribution. More details on the MCMC sampler are reported

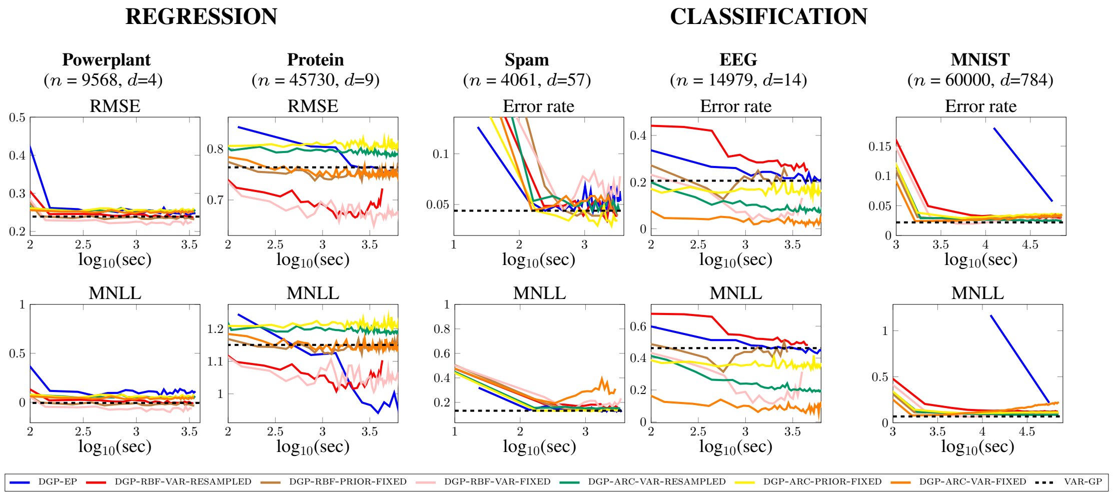

Figure 6. Progression of RMSE and MNLL over time for different optimisation strategies for DGP-ARC and DGP-RBF models. Results are shown for configurations having 1 hidden layer.

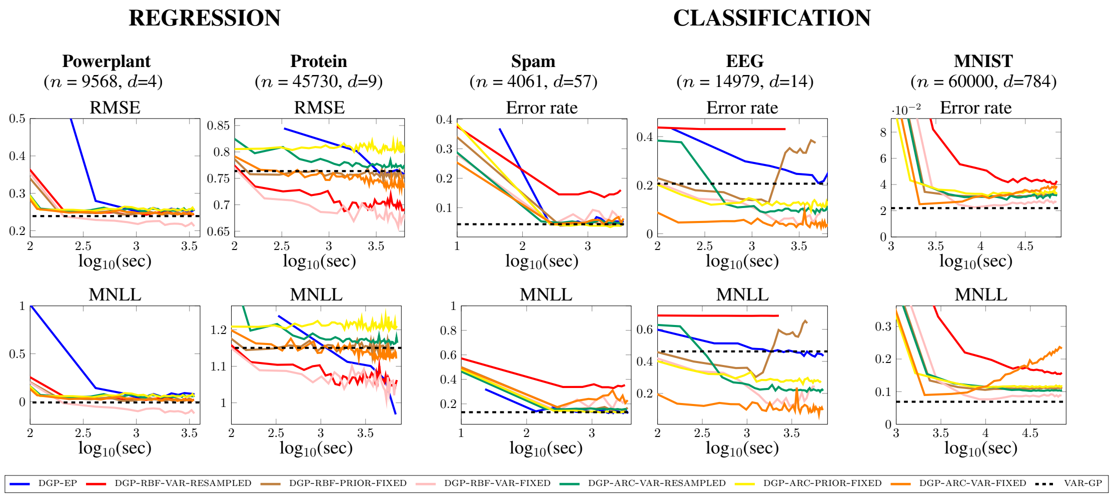

Figure 7. Progression of RMSE and MNLL over time for different optimisation strategies for DGP-ARC and DGP-RBF models. Results are shown for configurations having 2 hidden layers.

at the end of this section.

The plots depicted in Figure 8 illustrate how the MCMC approach explores two modes of the posterior of opposite sign. This is due to the output function being invariant to the flipping of the sign of the weights at the two layers. Conversely, the variational approximation over $\mathbf{W}$ accurately identifies one of the two modes of the posterior. The overall approximation is accurate in the case of more random Fourier features, whereas in the case of less, the approximation is unsurprisingly characterized by out-of-sample oscillations. The variational approximation seems to result in larger uncertainty in predictions compared to MCMC; we attribute this to the factorization of the posterior over all the weights.

B.1. Details of MCMC sampler for a two-layer DGP with a Gaussian likelihood

We give details of the MCMC sampler that we used to draw samples from the posterior over latent variables in DGPs. In the experiments, we regard this as the gold-standard against which we compare the quality of the proposed DGP approximation and inference. For the sake of tractability, we assume a two-layer DGP with a Gaussian likelihood, and we fix the hyperparameters of the GPs. Without loss of generality, we assume Y to be univariate and the hidden layer to be composed of a

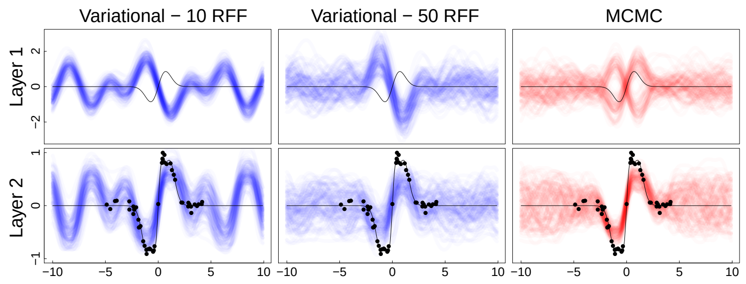

Figure 8. Comparison of MCMC and variational inference of a two-layer DGP with a single GP in the hidden layer and a Gaussian likelihood. The posterior over the latent functions is based on 100 MCMC samples and 100 samples from the variational posterior.

single GP. The model is therefore as follows:

$$\begin{aligned} p\left(Y \middle| F^{(2)}, \lambda\right) &= \mathcal{N}\left(Y \middle| F^{(2)}, \lambda I\right) \\ p\left(F^{(2)} \middle| F^{(1)}, \boldsymbol{\theta}^{(1)}\right) &= \mathcal{N}\left(F^{(2)} \middle| \mathbf{0}, K\left(F^{(1)}, \boldsymbol{\theta}^{(1)}\right)\right) \\ p\left(F^{(1)} \middle| X, \boldsymbol{\theta}^{(0)}\right) &= \mathcal{N}\left(F^{(1)} \middle| \mathbf{0}, K\left(X, \boldsymbol{\theta}^{(0)}\right)\right) \end{aligned}$$

with λ, $\boldsymbol{\theta}^{(1)}$, and $\boldsymbol{\theta}^{(0)}$ fixed. In the model specification above, we denoted by $K\left(F^{(1)}, \boldsymbol{\theta}^{(1)}\right)$ and $K\left(X, \boldsymbol{\theta}^{(0)}\right)$ the covariance matrices obtained by applying the covariance function with parameters $\boldsymbol{\theta}^{(1)}$, and $\boldsymbol{\theta}^{(0)}$ to all pairs of $F^{(1)}$ and X, respectively.

Given that the likelihood is Gaussian, it is possible to integrate out $F^{(2)}$ analytically

$$p\left(Y \middle| F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}\right) = \int p\left(Y \middle| F^{(2)}, \lambda\right) p\left(F^{(2)} \middle| F^{(1)}, \boldsymbol{\theta}^{(1)}\right) dF^{(2)}$$

obtaining the more compact model specification:

$$\begin{aligned} p\left(Y \middle| F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}\right) &= \mathcal{N}\left(Y \middle| \mathbf{0}, K\left(F^{(1)}, \boldsymbol{\theta}^{(1)}\right) + \lambda I\right) \\ p\left(F^{(1)} \middle| X, \boldsymbol{\theta}^{(0)}\right) &= \mathcal{N}\left(F^{(1)} \middle| \mathbf{0}, K\left(X, \boldsymbol{\theta}^{(0)}\right)\right) \end{aligned}$$

For fixed hyper-parameters, these expressions reveal that the observations are distributed as in the standard GP regression case, with the only difference that the covariance is now parameterized by GP distributed random variables $F^{(1)}$. We can interpret these variables as some sort of hyper-parameters, and we can attempt to use standard MCMC methods to samples from their posterior.

In order to develop a sampler for all latent variables, we factorize their full posterior as follows:

$$p\left(F^{(2)}, F^{(1)} \middle| Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)}\right) = p\left(F^{(2)} \middle| Y, F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}\right) p\left(F^{(1)} \middle| Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)}\right)$$

which suggest a Gibbs sampling strategy to draw samples from the posterior where we iterate

1. sample from $p\left(F^{(1)} \middle| Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)}\right)$
2. sample from $p\left(F^{(2)} \middle| Y, F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}\right)$

Step 1. can be done by setting up a Markov chain with invariant distribution given by:

$$p\left(F^{(1)} \middle| Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)}\right) \propto p\left(Y \middle| F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}\right) p\left(F^{(1)} \middle| X, \boldsymbol{\theta}^{(0)}\right)$$

We can interpret this as a GP model, where the likelihood now assumes a complex form because of the nonlinear way in which the likelihood depends on $F^{(1)}$. Because of this interpretation, we can attempt to use any of the samplers developed in the literature of GPs to obtain samples from the posterior over latent variables in GP models.

Step 2. can be done directly given that the posterior over $F^{(2)}$ is available in closed form and it is Gaussian:

$$p\left(F^{(2)}\middle|Y, F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}\right) = \mathcal{N}\left(F^{(2)}\middle|K^{(1)}\left(K^{(1)} + \lambda I\right)^{-1} Y, K^{(1)} - K^{(1)}\left(K^{(1)} + \lambda I\right)^{-1} K^{(1)}\right)$$

where we have defined

$$K^{(1)} := K\left(F^{(1)}, \boldsymbol{\theta}^{(1)}\right)$$

C. Derivation of the lower bound

For the sake of completeness, here is a detailed derivation of the lower bound that we use in variational inference to learn the posterior over $\mathbf{W}$ and optimize $\boldsymbol{\Theta}$, assuming $\boldsymbol{\Omega}$ fixed:

$$\begin{aligned}
\log[p(Y|X, \boldsymbol{\Omega}, \boldsymbol{\Theta})] &= \log\left[\int p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta}) p(\mathbf{W}) d\mathbf{W}\right] \\
&= \log\left[\int \frac{p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta}) p(\mathbf{W})}{q(\mathbf{W})} q(\mathbf{W}) d\mathbf{W}\right] \\
&= \log\left[\mathrm{E}_{q(\mathbf{W})} \frac{p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta}) p(\mathbf{W})}{q(\mathbf{W})}\right] \\
&\geq \mathrm{E}_{q(\mathbf{W})}\left(\log\left[\frac{p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta}) p(\mathbf{W})}{q(\mathbf{W})}\right]\right) \\
&= \mathrm{E}_{q(\mathbf{W})}\left(\log[p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta})]\right) + \mathrm{E}_{q(\mathbf{W})}\left(\log\left[\frac{p(\mathbf{W})}{q(\mathbf{W})}\right]\right) \\
&= \mathrm{E}_{q(\mathbf{W})}\left(\log[p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta})]\right) - \mathrm{DKL}[q(\mathbf{W})||p(\mathbf{W})]
\end{aligned}$$

D. Learning Ω variationally

Defining $\boldsymbol{\Psi} = \{\mathbf{W}, \boldsymbol{\Omega}\}$, we can attempt to employ variational inference to treat the spectral frequencies $\boldsymbol{\Omega}$ variationally as well as $\mathbf{W}$. In this case, the detailed derivation of the lower bound is as follows:

$$\begin{aligned}
\log\left[p(Y|X, \boldsymbol{\Theta})\right] &= \log\left[\int p(Y|X, \boldsymbol{\Psi}, \boldsymbol{\Theta}) p(\boldsymbol{\Psi}|\boldsymbol{\Theta}) d\boldsymbol{\Psi}\right] \\
&= \log\left[\int \frac{p(Y|X, \boldsymbol{\Psi}, \boldsymbol{\Theta}) p(\boldsymbol{\Psi}|\boldsymbol{\Theta})}{q(\boldsymbol{\Psi})} q(\boldsymbol{\Psi}) d\boldsymbol{\Psi}\right] \\
&= \log\left[\mathrm{E}_{q(\boldsymbol{\Psi})} \frac{p(Y|X, \boldsymbol{\Psi}, \boldsymbol{\Theta}) p(\boldsymbol{\Psi}|\boldsymbol{\Theta})}{q(\boldsymbol{\Psi})}\right] \\
&\geq \mathrm{E}_{q(\boldsymbol{\Psi})}\left(\log\left[\frac{p(Y|X, \boldsymbol{\Psi}, \boldsymbol{\Theta}) p(\boldsymbol{\Psi}|\boldsymbol{\Theta})}{q(\boldsymbol{\Psi})}\right]\right) \\
&= \mathrm{E}_{q(\boldsymbol{\Psi})}\left(\log[p(Y|X, \boldsymbol{\Psi}, \boldsymbol{\Theta})]\right) + \mathrm{E}_{q(\boldsymbol{\Psi})}\left(\log\left[\frac{p(\boldsymbol{\Psi}|\boldsymbol{\Theta})}{q(\boldsymbol{\Psi})}\right]\right) \\
&= \mathrm{E}_{q(\boldsymbol{\Psi})}\left(\log[p(Y|X, \boldsymbol{\Psi}, \boldsymbol{\Theta})]\right) - \mathrm{DKL}[q(\boldsymbol{\Psi})||p(\boldsymbol{\Psi}|\boldsymbol{\Theta})]
\end{aligned}$$

Again, assuming a factorized prior over all weights across layers

$$p(\boldsymbol{\Psi}|\boldsymbol{\theta}) = \prod_{l=0}^{N_\mathrm{h}-1} p(\Omega^{(l)}|\boldsymbol{\theta}^{(l)}) p(W^{(l)}) = \prod_{ijl} q\left(\Omega_{ij}^{(l)}\right) \prod_{ijl} q\left(W_{ij}^{(l)}\right), \tag{13}$$

we optimize the variational lower bound using variational inference following the mini-batch approach with the reparameterization trick explained in the main paper. The variational parameters then become the mean and the variance of each of the approximating factors

$$q\left(W_{ij}^{(l)}\right) = \mathcal{N}\left(m_{ij}^{(l)}, (s^2)_{ij}^{(l)}\right), \tag{14}$$

$$q\left(\Omega_{ij}^{(l)}\right) = \mathcal{N}\left(\mu_{ij}^{(l)}, (\beta^2)_{ij}^{(l)}\right), \tag{15}$$

and we optimize the lower bound with respect to the variational parameters $m_{ij}^{(l)}, (s^2)_{ij}^{(l)}, \mu_{ij}^{(l)}, (\beta^2)_{ij}^{(l)}$.

E. Expression for the DKL divergence between Gaussians

Given $p_1(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $p_2(x) = \mathcal{N}(\mu_2, \sigma_2^2)$, the KL divergence between the two is:

$$\text{DKL}\left(p_1(x) \| p_2(x)\right) = \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1 - \mu_2)^2}{\sigma_2^2}\right]$$

F. Distributed Implementation

Our model is easily amenable to a distributed implementation using asynchronous distributed stochastic gradient descent Chilimbi et al. (2014). Our distributed setting, based on TensorFlow, includes one or more *Parameter servers* (PS), and a number of *Workers*. The latter proceed asynchronously using randomly selected batches of data: they fetch fresh model parameters from the PS, compute the gradients of the lower bound with respect to these parameters, and push those gradients back to the PS, which update the model accordingly. Given that workers compute gradients and send updates to PS asynchronously, the discrepancy between the model used to compute gradients and the model actually updated can degrade training quality. This is exacerbated by a large number of asynchronous workers, as noted in Chen et al. (2016).

We focus our experiments on the MNIST dataset, and study how training time and error rates evolve with the number of workers introduced in our system. The parameters for the model are identical to those reported for the previous experiments.

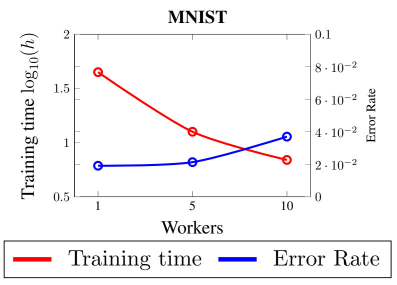

Figure 9. Comparison of training time and error rate for asynchronous DGP-RBF with 1, 5 and 10 workers.

We report the results in Figure 9, and as expected, the training time decreases in proportion to the number of workers, albeit sub-linearly. On the other hand, the increasing error rate confirms our intuition that imprecise updates of the gradients negatively impact the optimization procedure. The work in Chen et al. (2016) corroborates our findings, and motivates efforts in the direction of alleviating this issue.

Chapter 4

Conclusions

The topic of nonparametric modeling through kernel methods has gained considerable attention in Machine Learning since the development of Support Vector Machines [8] and Gaussian processes [40] in the '90s. After that, a huge number of contributions have been proposed to scale kernel-based learning. While these approaches have proven successful and offer an elegant framework for nonparametric modeling and, in the case of GPs, for quantification of uncertainty, recent years have seen the rise of deep learning as the preferred way to tackle large-scale Machine Learning problems, given the impressive performance in various tasks [26]. Regarding several model approximations and lack of scalability as the limiting factors for GPs, the contributions gathered in this thesis aim to render the applicability of GP modeling more practical and scalable. By developing these techniques further, the aim is to be able to use GP models on large-scale problems where an effective comparison with deep learning techniques can be attempted. Some works are already pointing to the ability of kernel methods to perform on-par with deep neural networks [27, 21], so there is hope that this is actually possible. My research agenda will explore these ideas in the context of GPs and quantification of uncertainty.

My current ongoing research is investigating a number of extensions as follows:

- **Probabilistic numerics for large scale gp learning**: One line of work that I am currently pursuing in collaboration with the University of Oxford is in probabilistic numerics, that is an emerging area in Machine Learning and Statistics. The key idea is to reinterpret *calculation* as *estimation*, so that expensive or intractable computations are replaced by estimates based on evidence that is easier to gather. As a result, these methods offer a way to quantify the uncertainty in the calculation due to the lack of computational resources, and when these calculations are part of some statistical models, this uncertainty can be propagated forward to model predictions and analysis. We recently applied this to the Bayesian inference of log-determinants of large dense matrices showing the impact of using a small number of matrix-vector products on its uncertainty.

- **Structured Gaussian matrices**: Kernel methods using random feature expansions struggle to cope effectively with large-dimensional input space due to the introduction of $\mathcal{O}(DN_{\mathrm{RF}})$ parameters, where D is the dimension of the input space and N_{RF} is the number of random features. For simplicity, assuming $N_{\mathrm{RF}} = D$, this means dealing with $\mathcal{O}(D^2)$ parameters leading to time complexity that is also $\mathcal{O}(D^2)$. In the literature there has been work to deal with the quadratic scaling in D by using structured random matrices [34, 42], leading to space complexity of $\mathcal{O}(D)$ and time complexity $\mathcal{O}(D \log(D))$. We are starting a collaboration with the authors of [42] at Google Research in NYC to apply these ideas to the approximation of DGPs, that should therefore provide us with the possibility to obtain better GP and DGP approximations.

- **Better variational approximations**: The framework for large-scale inference of DGPs is based on stochastic variational inference, where the assumption is that the posterior over model parameters factorizes. This is a restrictive assumption that affects the quality of the inference, and it would be sensible to obtain better variational approximations that are still amenable to mini-batch learning. There have been some works that yield tighter bounds for the marginal likelihood [5], and the idea is to study these in the context of large-scale learning of GPs and DGPs.

- **Distributed linear algebra**: In the context of inference or optimization of covariance parameters of GPs, it would be interesting to be able to solve these without any approximations. In particular, this would entail being able to solve large dense linear systems exactly. We are currently investigating how to do this exploiting large-scale computing facilities, by extending previous work on parallel algebra. We are currently looking into the development of efficient distributed Cholesky factorization algorithms, as well as iterative (preconditioned) algorithms to solve large dense linear systems.

I envisage immediate impact of the results of my research in a number of areas where nonparametric models are commonly employed, such as the emulation of the output of expensive models. The applicability of Gaussian processes is currently limited by the scalability issues discussed above, and scaling computations without sacrificing accuracy would make an enormous impact in this community. I am starting a collaboration with a former colleague at UCL that uses statistical models to emulate the output of simulators implementing climate and Tsunami models, and I am planning to capitalize on this soon, as the work on scaling Bayesian computations for Gaussian processes is mature enough.

I also envisage immediate impact in life sciences. Currently, inferring parameters of mathematical models describing regulatory interactions and signaling processes in living cells is extremely challenging, and accurately modeling the development of neurological disorders from neuroimaging data is very difficult. Challenges arise because of the necessity to simulate expensive models forward in time, and because of the huge dimensionality of the data. By scaling Bayesian computations for Gaussian processes it will be truly possible to draw meaningful conclusions from the analysis of data in these domains. I am currently involved in a collaboration with biologists at Glasgow, UK, and, as an example, this research could yield a deeper understanding of potential factors in the development of cardiovascular diseases and diabetes. I am also collaborating with scientists at the Donders Institute of Neuroscience, The Netherlands, that is a center of excellence in the analysis of neuroimaging data. I've recently been awarded an AXA Chair for the duration of seven years to investigate these topics in greater detail.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, Apr. 2009.

[3] N. Babaguchi, K. Aizawa, J. R. Smith, S. Satoh, T. Plagemann, X.-S. Hua, and R. Yan, editors. *Proceedings of the 20th ACM Multimedia Conference, MM '12, Nara, Japan, October 29 - November 02, 2012*. ACM, 2012.

[4] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 1st ed. 2006. corr. 2nd printing 2011 edition, Aug. 2006.

[5] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015.

[6] E. Canestrelli, P. Canestrelli, M. Corazza, M. Filippone, S. Giove, and F. Masulli. Local learning of tide level time series using a fuzzy approach. In *IJCNN*, pages 1813–1818. IEEE, 2007.

[7] C. Carota, M. Filippone, R. Leombruni, and S. Polettini. Bayesian nonparametric disclosure risk estimation via mixed effects log-linear models. *Annals of Applied Statistics*, 9(1):525–546, 2015.

[8] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20:273–297, 1995.

[9] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random feature expansions for deep Gaussian processes, 2016. arXiv:1610.04386.

[10] K. Cutajar, M. A. Osborne, J. P. Cunningham, and M. Filippone. Preconditioning kernel matrices. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2016, New York City, USA, June 19-24, 2016*, 2016.

[11] A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Proceedings*, pages 207–215. JMLR.org, 2013.

[12] M. Dell'Amico, M. Filippone, P. Michiardi, and Y. Roudier. On user availability prediction and network applications. *IEEE Transactions on Networking*, 2014.

[13] F. Dondelinger, M. Filippone, S. Rogers, and D. Husmeier. ODE parameter inference using adaptive gradient matching with Gaussian processes. In *AISTATS*, 2013.

[14] D. K. Duvenaud, O. Rippel, R. P. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 202–210. JMLR.org, 2014.

[15] M. Filippone and R. Engler. Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE). In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015*, 2015.

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

[17] M. Filippone, A. F. Marquand, C. R. V. Blain, S. C. R. Williams, J. Mourão-Miranda, and M. Girolami. Probabilistic Prediction of Neurological Disorders with a Statistical Assessment of Neuroimaging Data Modalities. *Annals of Applied Statistics*, 6(4):1883–1905, 2012.

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

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

[20] J. Hensman, A. G. de G. Matthews, M. Filippone, and Z. Ghahramani. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12 2015, Montreal, Quebec, Canada*, 2015.

[21] P. Huang, H. Avron, T. N. Sainath, V. Sindhwani, and B. Ramabhadran. Kernel methods match deep neural networks on TIMIT. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[22] S. Kim, M. Filippone, F. Valente, and A. Vinciarelli. Predicting the conflict level in television political debates: an approach based on crowdsourcing, nonverbal communication and Gaussian processes. In Babaguchi et al. [3], pages 793–796.

[23] S. Kim, F. Valente, M. Filippone, and A. Vinciarelli. Predicting continuous conflict perception with Bayesian Gaussian processes. *IEEE Transactions on Affective Computing*, 5(2):187–200, 2014.

[24] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations, ICLR 2014, Banff, Canada, April 14-16, 2014*, 2014.

[25] K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone. AutoGP: Exploring the capabilities and limitations of Gaussian process models, 2016. arXiv:1610.05392.

[26] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[27] Z. Lu, A. May, K. Liu, A. B. Garakani, D. Guo, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, and F. Sha. How to scale up kernel methods to be as good as deep neural nets. *CoRR*, abs/1411.4000, 2014.

[28] A. F. Marquand, M. Filippone, J. Ashburner, M. Girolami, J. Mourao-Miranda, G. J. Barker, S. C. R. Williams, P. N. Leigh, and C. R. V. Blain. Automated, High Accuracy Classification of Parkinsonian Disorders: A Pattern Recognition Approach. *PLoS ONE*, 8(7):e69237+, July 2013.

[29] L. Mohamed, B. Calderhead, M. Filippone, M. Christie, and M. Girolami. Population MCMC methods for history matching and uncertainty quantification. *Computational Geosciences*, 16(2):423–436, 2012.

[30] G. Mohammadi, A. Origlia, M. Filippone, and A. Vinciarelli. From speech to personality: mapping voice quality and intonation into personality differences. In Babaguchi et al. [3], pages 789–792.

[31] R. M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, Aug. 1996.

[32] M. Niu, S. Rogers, M. Filippone, and D. Husmeier. Fast inference in nonlinear dynamical systems using gradient matching. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2016, New York City, USA, June 19-24, 2016*, 2016.

[33] A. D. O'Harney, A. Marquand, K. Rubia, K. Chantiluke, A. B. Smith, A. Cubillo, C. Blain, and M. Filippone. Pseudo-marginal Bayesian multiple-class multiple-kernel learning for neuroimaging data. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pages 3185–3190. IEEE, 2014.

[34] A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.

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

[36] J. M. Rondina, M. Filippone, M. Girolami, and N. S. Ward. Decoding post-stroke motor function from structural brain imaging. *NeuroImage: Clinical*, 12:372–380, 2016.

[37] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[38] M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In D. A. Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, volume 5 of *JMLR Proceedings*, pages 567–574. JMLR.org, 2009.

[39] M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011.

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

[41] X. Xiong, M. Filippone, and A. Vinciarelli. Looking good with flickr faves: Gaussian processes for finding difference makers in personality impressions. In *ACM Multimedia*, 2016.

[42] F. X. Yu, A. T. Suresh, K. M. Choromanski, D. Holtmann-Rice, and S. Kumar. Orthogonal random features. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1975–1983, 2016.