Tighter Bounds on the Log Marginal Likelihood of Gaussian Process Regression using Conjugate Gradients

Artem Artemev [*1 2] **David R. Burt** [*3] **Mark van der Wilk** [1]

Abstract

We propose a lower bound on the log marginal likelihood of Gaussian process regression models that can be computed without matrix factorisation of the full kernel matrix. We show that approximate maximum likelihood learning of model parameters by maximising our lower bound retains many benefits of the sparse variational approach while reducing the bias introduced into hyperparameter learning. The basis of our bound is a more careful analysis of the log-determinant term appearing in the log marginal likelihood, as well as using the method of conjugate gradients to derive tight lower bounds on the term involving a quadratic form. Our approach is a step forward in unifying methods relying on lower bound maximisation (e.g. variational methods) and iterative approaches based on conjugate gradients for training Gaussian processes. In experiments, we show improved predictive performance with our model for a comparable amount of training time compared to other conjugate gradient based approaches.

1. Introduction

Scaling models involving Gaussian process priors to large datasets is an important and well-researched problem in Bayesian statistics and machine learning. In order for a method to succeed in this task, it should provide high-quality approximations to 1) the posterior mean and (co-)variance over functions, 2) the log marginal likelihood (LML). While only the former is needed for making predictions, the latter is a useful tool for model selection; when the kernel is differentiable with respect to hyperparameters, the LML or an approximation to it can be optimised using gradient based methods in order to automatically select model hyperparameters (Rasmussen & Williams, 2006, section 5).

We derive a lower bound on the LML of regression with a Gaussian process prior and a Gaussian likelihood. We refer to this bound as CGLB. Parameter learning with CGLB combines many of the strengths of sparse variational Gaussian process regression (SGPR) (Titsias, 2009) with the strengths of conjugate gradient (CG) methods (Gibbs & Mackay, 1997) for GP inference. We use an improved bound on the log-determinant of the covariance matrix as well as CG to tighten the SGPR evidence lower bound (ELBO). We show empirically that this leads to less bias in parameter selection than maximisation of the ELBO. This reduced bias leads to improved performance on several benchmark tasks involving large datasets.

Maximisation of our lower bound provides an alternative to current CG-based GP inference schemes Gibbs & Mackay (1997); Wang et al. (2019), which simplifies hyperparameter learning. Recently, Wang et al. (2019) showed that CG-based GP inference can scale impressively to large datasets, and provide excellent performance on many regression tasks. While Wang et al. (2019) argued that their method should be considered 'exact', there are two caveats which complicate training. First, gradient estimates provided by these methods are biased if conjugate gradients is stopped too early; while this bias can be reduced by running more iterations of CG, this comes at an additional computational cost. Second, the estimates of the LML are stochastic. The variance of the estimator provided can be reduced at the cost of needing to solve more systems of equations. Following Davies (2015), we refer to these approaches as 'Iterative GPs', to contrast them with implementations using deterministic, direct methods (e.g. Cholesky decomposition) for computing the LML.

Building on work in Gibbs & Mackay (1997) and Davies (2015) we derive a stopping criterion for our application of CG that ensures more iterations of CG would not improve the bound on the LML significantly. We also exploit old solutions to CG from previous iterations of hyperparameter learning. This approach allows us to often run zero or one steps of CG per iteration of hyperparameter learning without significantly impacting the approximate LML, even on large datasets. Using a similar approach to Gardner et al.

[*]Equal contribution [1]Department of Computing, Imperial College London, London, UK [2]Secondmind, Cambridge, UK [3]Department of Engineering, University of Cambridge, Cambridge, UK. Correspondence to: David R. Burt <drb62@cam.ac.uk>, Artem Artemev <a.artemev20@imperial.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

(2018); Wang et al. (2019) and Meanti et al. (2020) CGLB can be implemented with memory complexity that is linear in the number of training examples, n, by splitting up (and parallelising) matrix-vector products (Charlier et al., 2021).

We empirically show that the combination of a deterministic objective function and reduced number of CG steps per iteration of hyperparameter optimisation leads to improved stability and performance when performing model selection (for comparable computational times) compared to existing approximate methods for GP regression.

2. Background

In this section, we review Gaussian process regression with a Gaussian likelihood. We then discuss the conjugate gradient method for solving linear systems of equations, and how this can be applied to scaling Gaussian process regression. We conclude the section with a discussion of sparse variational inference as a method for scalable approximate inference in GP regression models.

2.1. Gaussian Process Regression

We assume a dataset has been observed and denote it by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathcal{X}$ where $\mathcal{X}$ denotes the set of possible observed features and $y_i \in \mathbb{R}$. Let $\mathbf{y} \in \mathbb{R}^n$ denote the vector formed by concatenating the y_i and $\mathbf{x} \in \mathcal{X}^n$ denote the tuple $(x_i)_{i=1}^n$.

We take a Bayesian approach with prior $f \sim \mathcal{GP}(0, k)$, i.e. f is a Gaussian process with zero mean and covariance function k, and likelihood $Y|f(\mathbf{x}) \sim \mathcal{N}(f(\mathbf{x}), \sigma^2\mathrm{I})$, where Y is an $\mathbb{R}^n$-valued random variable and by an abuse of notation we use $f(\mathbf{x}) \in \mathbb{R}^n$ to denote the $\mathbb{R}^n$-valued random variable formed by indexing f at each x_i. Inference involves computing the distribution of $f|(Y = \mathbf{y})$. This is again a Gaussian process with mean and covariance,

$$\tilde{\mu}(x) = \mathbf{k_{fs}}\mathrm{K}^{-1}\mathbf{y}, \quad \tilde{k}(x, x') = k(x, x') - \mathbf{k_{fs}}^\mathsf{T}\mathrm{K}^{-1}\mathbf{k_{fs'}},$$

with $x, x' \in \mathcal{X}$, $s = f(x)$, $s' = f(x')$, $\mathrm{K} = \mathrm{K_{ff}} + \sigma^2\mathrm{I}$, $\mathrm{K_{ff}} \in \mathbb{R}^{n\times n}$ with entries $(\mathrm{K_{ff}})_{ij} = k(x_i, x_j)$, and $\mathbf{k_{fs}}, \mathbf{k_{fs'}} \in \mathbb{R}^n$ take values $(\mathbf{k_{fs}})_i = k(x_i, x)$ and $(\mathbf{k_{fs'}})_i = k(x_i, x')$.

We assume k is parameterised, and denote these hyperparameters together with σ^2 as θ. The choice of θ has a significant impact on the generalisation properties of the posterior (Rasmussen & Williams, 2006, Chapter 5).

Type-II maximum likelihood is a heuristic that automates selection of θ by maximising the LML, defined as the log density of the prior probability distribution over Y evaluated at $Y = \mathbf{y}$, with respect to the hyperparameters θ. For this model the LML is

$$\log p_Y(\mathbf{y}; \theta) = c - \frac{1}{2}\underbrace{\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}}_{\text{quad. term}} - \frac{1}{2}\underbrace{\log|\mathrm{K}|}_{\text{log-det. term}}. \quad (1)$$

with $c = -\frac{n}{2}\log 2\pi$. Here and elsewhere, we suppress the dependence of kernel matrices on θ. Common implementations of Gaussian process inference and hyperparameter selection rely on a Cholesky factorisation of K in order to evaluate $\log|\mathrm{K}|$ and $\mathrm{K}^{-1}\mathbf{y}$. The Cholesky decomposition is usually implemented in a way that requires roughly $n^3/3$ floating point operations and stores a matrix with $n(n+1)/2$ distinct entries in memory. This can be a prohibitive cost for regression problems with many observations.

2.2. Conjugate Gradients

The conjugate gradient (CG) algorithm (Hestenes & Stiefel, 1952) is a method for solving systems of equations using only matrix-vector multiplication and elementary vector operations. Given a symmetric positive definite matrix K and a vector $\mathbf{y}$ the goal of conjugate gradients is to find an vector $\mathbf{v}$ satisfying $\mathrm{K}\mathbf{v} = \mathbf{y}$. Starting with an initial guess for $\mathbf{v}$, each iteration of the conjugate gradient method chooses a search direction and updates the current guess for $\mathbf{v}$ by adding a vector in the chosen direction. In exact arithmetic, CG is guaranteed to solve an $n \times n$ system of equations in n iterations, each of which involves a $\Theta(n^2)$ matrix-vector multiplication. Viewed as an iterative algorithm CG has strong guarantees on the rate at which the error decreases, at least for well-conditioned matrices (Hackbusch, 1994, Section 10.2.3). The convergence of conjugate gradient is often practically assessed by examining the residual $\mathbf{r} = \mathbf{y} - \mathrm{K}\mathbf{v}$, which is computed in each iteration of CG. If the residual is $\mathbf{0}$, then the algorithm has converged. For computational reasons, CG is often stopped when the residual has a sufficiently small Euclidean norm.

2.3. Gaussian Process Regression with the Conjugate Gradient Method

CG has been proposed as an efficient method for directly approximating the gradient of eq. (1) (Gibbs & Mackay, 1997). An obstacle to this approach is the evaluation of the gradient of the log-determinant, $\frac{\partial}{\partial\theta_i}\log|\mathrm{K}| = \mathrm{tr}(\mathrm{K}^{-1}\frac{\partial\mathrm{K}}{\partial\theta_i})$. Evaluating this trace directly is computationally expensive, as it requires solving n, $n \times n$ systems of linear equations. Hutchinson's trace estimator (Hutchinson, 1989) provides a stochastic estimate of this gradient. The estimator is formed by first noting that for any $\mathbb{R}^n$-valued random variable $\mathbf{p}$ such that $\mathbb{E}_\mathbf{p}\left[\mathbf{p}\mathbf{p}^\mathsf{T}\right] = \mathrm{I}$, $\mathrm{tr}(\mathrm{K}^{-1}\frac{\partial\mathrm{K}}{\partial\theta_i}) = \mathbb{E}_\mathbf{p}\left[\mathbf{p}^\mathsf{T}\mathrm{K}^{-1}\frac{\partial\mathrm{K}}{\partial\theta_i}\mathbf{p}\right]$. The expectation can be estimated with Monte Carlo, using CG to approximate $\mathrm{K}^{-1}\mathbf{p}_i$, where $\mathbf{p}_i$ is a sample of $\mathbf{p}$.

This procedure results in a biased, stochastic estimate of the gradient. The bias in this estimator can be decreased, and practically removed, at the cost of increasing the number of iterations of CG. The variance can be reduced by increasing the number of samples of $\mathbf{p}$, at the cost of needing to solve more systems of equation. The variance of this estimator for

various distributions of $\mathbf{p}$ as well as high probability bounds on the relative error are known (Avron & Toledo, 2011).

In cases when the log marginal likelihood itself is of interest, for example if model comparison is performed with discrete hyperparameters, approximations to the LML using CG and related ideas have also been proposed (Ubaru et al., 2017). Iterative GPs have been shown to be highly scalable with modern computational architectures (Gardner et al., 2018; Wang et al., 2019).

2.4. Gaussian Process Regression with Sparse Methods

An alternative approach, which avoids the computation of K entirely, relies on sparsity assumptions in the data-domain that lead to a low-rank approximation of K_{ff}, (e.g. Williams & Seeger, 2001; Snelson & Ghahramani, 2005). This approach reduces the computational requirement to $\mathcal{O}(nm^2)$ where m is a parameter that controls the rank of the approximation to K_{ff}. Titsias (2009) proposed an interpretation of sparse methods as variational inference with a structured family of posterior distributions. This framework defines an evidence lower bound (ELBO), $L(\mathbf{y};\theta) \leq \log p_Y(\mathbf{y};\theta)$, where

$$L(\mathbf{y};\theta) = c - \frac{1}{2}\underbrace{\mathbf{y}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{y}}_{\substack{\text{bound on}\\ \text{quad. term}}} - \frac{1}{2}\underbrace{\left(\log|\mathrm{Q}| + \frac{\mathrm{tr}(\mathrm{K}-\mathrm{Q})}{\sigma^2}\right)}_{\text{bound on log-det. term}} \quad (2)$$

with $\mathrm{Q} = \mathrm{Q}_{\mathrm{ff}} + \sigma^2\mathrm{I}$, $\mathrm{Q}_{\mathrm{ff}} = \mathrm{K}_{\mathrm{uf}}^\mathsf{T}\mathrm{K}_{\mathrm{uu}}^{-1}\mathrm{K}_{\mathrm{uf}}$, K_{uu} an $m \times m$ matrix with $(\mathrm{K}_{\mathrm{uu}})_{ij} = k(z_i, z_j)$, K_{uf} an $m \times n$ matrix with $(\mathrm{K}_{\mathrm{uf}})_{ij} = k(z_i, x_j)$, $z_i \in \mathcal{X}$ and $u_i = f(z_i)$.[1] The z_i are variational parameters. The ELBO can be evaluated in $O(nm^2)$, and is frequently jointly maximised with respect to variational parameters $\mathbf{z} = (z_i)_{i=1}^m$ and hyperparameters θ as a form of approximate maximum likelihood learning. The variational posterior is then used for prediction. This is a Gaussian process, with mean and covariance,

$$\widehat{m}(x) = \mathbf{k_{us}}^\mathsf{T}\mathrm{K}_{\mathrm{uu}}^{-1}\mathrm{K}_{\mathrm{uf}}\mathrm{Q}^{-1}\mathbf{y} \quad \text{and} \quad (3)$$

$$\hat{k}(x,x') = k(x,x') - \mathbf{k_{us}}^\mathsf{T}\mathrm{K}_{\mathrm{uu}}^{-1}\mathrm{K}_{\mathrm{uf}}\mathrm{Q}^{-1}\mathrm{K}_{\mathrm{uf}}\mathrm{K}_{\mathrm{uu}}^{-1}\mathbf{k_{us'}} \quad (4)$$

where $\mathbf{k_{us}}, \mathbf{k_{us'}} \in \mathbb{R}^m$ have entries $(\mathbf{k_{us}})_i = k(\mathbf{z}_i, x)$ and $(\mathbf{k_{us'}})_i = k(\mathbf{z}_i, x')$. When the number of training examples is large, the kernel is sufficiently smooth, and the data is not too spread out $\log p_Y(\mathbf{y};\theta) - L(\mathbf{y};\theta)$ is small for some $m \ll n$ (Burt et al., 2020), in which case one expects that similar hyperparameters would be selected by maximising the ELBO as would be selected by maximising the LML. However, certain settings of θ, for example those for which σ^2 is very small, lead to large discrepancies between the LML and ELBO, which results in significant bias in parameter selection and underfitting (Bauer et al., 2016).

[1] We assume here and throughout that K_{uu} is invertible.

3. Lower Bounds on the Log Marginal Likelihood

In this section we present two new lower bounds on the log marginal likelihood one of which can be computed in $O(nm^2 + (t+1)n^2)$, and one that can be computed in $O(n^2m + (t+1)n^2)$ where t is the number of steps of CG run. We also present a mean function that can be used for prediction in conjunction with these bounds.

3.1. Conjugate Gradient Lower Bound

We now state the lower bound:

Lemma 1. *Let* K *as in eq.* (1)*,* Q *as in eq.* (2)*,* $c = -\frac{n}{2}\log 2\pi$*, then for any* $\mathbf{v} \in \mathbb{R}^n$

$$\log p_Y(\mathbf{y};\theta) \geq c - \frac{1}{2}\left(\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r} + 2\mathbf{y}^\mathsf{T}\mathbf{v} - \mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}\right) - \frac{1}{2}\left(\log|\mathrm{Q}| + n\log\left(\frac{\mathrm{tr}(\mathrm{Q}^{-1}\mathrm{K})}{n}\right)\right), \quad (5)$$

$$\geq c - \frac{1}{2}\left(\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r} + 2\mathbf{y}^\mathsf{T}\mathbf{v} - \mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}\right) - \frac{1}{2}\left(\log|\mathrm{Q}| + n\log\left(1 + \frac{\mathrm{tr}(\mathrm{K}-\mathrm{Q})}{n\sigma^2}\right)\right), \quad (6)$$

where $\mathbf{r} = \mathbf{y} - \mathrm{K}\mathbf{v}$.

Remark 1. *When* $\mathbf{v} = 0$ *or* $\mathbf{v} = \mathrm{K}^{-1}\mathbf{y}$*, the bounds in lemma 1 are at least as tight as eq.* (2)*. For general* $\mathbf{v}$*, they may be smaller than the lower bound in eq.* (2)*. We address this in section 3.5 by ensuring* $\mathbf{v} \approx \mathrm{K}^{-1}\mathbf{y}$ *in such a way that the bounds in lemma 1 is no more than* ϵ *smaller than eq.* (2) *(and in practice nearly always larger), where* $\epsilon > 0$ *is a user-specified parameter.*

We refer to eq. (6) as CGLB. The right hand side of eqs. (5) and (6) can be maximised with respect to $\{\theta, \mathbf{v}, (z_i)_{i=1}^m\}$ for hyperparameter learning. For fixed $\mathbf{v}$, the right hand side of eq. (5) can be computed in $O(n^2 + n^2m)$, while eq. (6) can be computed in $O(n^2 + nm^2)$ using similar computations to eq. (2) but with an additional $n \times n$ matrix-vector product to compute $\mathrm{K}\mathbf{v}$. If $\mathbf{v} = \mathrm{K}^{-1}\mathbf{y}$, then $\mathbf{r} = 0$, and the first term in the bounds becomes $-\frac{1}{2}\mathbf{y}\mathrm{K}^{-1}\mathbf{y}$. This choice of $\mathbf{v}$ maximises the lower bounds (CGLB) for any choice of $\{\theta, (z_i)_{i=1}^m\}$, is independent of $(z_i)_{i=1}^m$, and can be approximated by running conjugate gradients on the system of equation $\mathrm{K}\mathbf{v} = \mathbf{y}$.

In order to derive eqs. (5) and (6) on the log marginal likelihood, it suffices to upper bound the quadratic term and the log-determinant term from eq. (1).

3.2. Bounds on the Log-Determinant Term

Unlike previous Iterative GPs (Gibbs & Mackay, 1997; Gardner et al., 2018), we aim for a deterministic estimate of

the LML and its gradient. A simple approach is to combine estimates of the term $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}$ based on CG with the bound $\log|\mathrm{K}| \leq \log|\mathrm{Q}| + \frac{1}{\sigma^2}\mathrm{tr}(\mathrm{K}-\mathrm{Q})$ from SGPR. This bound is tight when $\mathrm{tr}(\mathrm{K}-\mathrm{Q})/\sigma^2 \approx 0$, but is loose otherwise. We derive tighter bounds on $\log|\mathrm{K}|$ using properties of Q.

First, we recall several matrix properties.

Proposition 1 (Horn & Johnson, 2012, page 11, 51)**.** *Let* $\mathrm{A}, \mathrm{B} \in \mathbb{R}^{n\times n}$ *and* $\{\lambda_i\}_{i=1}^n$ *denote the eigenvalues of* A *(counted with multiplicity). Then 1.* $|\mathrm{A}| = \prod_{i=1}^n \lambda_i$ *2.* $|\mathrm{AB}| = |\mathrm{A}||\mathrm{B}|$*, 3.* $\mathrm{tr}(\mathrm{A}) = \sum_{i=1}^n \lambda_i$*, 4.* $\mathrm{tr}(\mathrm{AB}) = \mathrm{tr}(\mathrm{BA})$.

We say a symmetric matrix $\mathrm{A} \in \mathbb{R}^{n\times n}$ is *positive semi-definite* (PSD) if for all $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{w}^\mathsf{T}\mathrm{A}\mathbf{w} \geq 0$, and positive definite if the inequality is strict. For any PSD matrix, we have $\lambda_i \geq 0$ where λ_i denotes an eigenvalue of A; for positive definite matrices, this inequality is again strict.

Proposition 2 (Horn & Johnson, 2012, page 495)**.** *Let* H *be a symmetric real matrix with* $\mathrm{H} = \left[\begin{smallmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{B}^\mathsf{T} & \mathrm{C} \end{smallmatrix}\right]$ *with* A *non-singular. Then* H *is PSD if and only if* A *and* $\mathrm{C} - \mathrm{B}^\mathsf{T}\mathrm{A}^{-1}\mathrm{B}$ *are both PSD.*

Consider $\mathrm{H} = \left[\begin{smallmatrix} \mathrm{K_{uu}} & \mathrm{K_{uf}} \\ \mathrm{K_{uf}^T} & \mathrm{K_{ff}} \end{smallmatrix}\right]$. Since the kernel is PSD, and H is formed by evaluating the kernel at $\{\mathbf{z}, \mathbf{x}\}$ H is PSD. Proposition 2 implies that $\mathrm{K_{ff}} - \mathrm{Q_{ff}} = \mathrm{K} - \mathrm{Q}$ is PSD.

Proposition 3 (Horn & Johnson, 2012, page 431)**.** *Let* $\mathrm{A} \in \mathbb{R}^{n\times n}$ *be positive definite and* $\mathrm{B} \in \mathbb{R}^{n\times m}$*. Then* $\mathrm{B}^\mathsf{T}\mathrm{AB}$ *is positive definite.*

Proposition 4 (Horn & Johnson, 2012, Corollary 7.7.4)**.** *Let* $\mathrm{A}, \mathrm{B} \in \mathbb{R}^{n\times n}$ *such that* $\mathrm{A} - \mathrm{B}$ *is PSD. If* A *and* B *are invertible,* $\mathrm{B}^{-1} - \mathrm{A}^{-1}$ *is PSD.*

Proposition 5. *Let* $\mathrm{A} \in \mathbb{R}^{n\times n}$ *PSD, then* $\log|\mathrm{I} + \mathrm{A}| \leq \mathrm{tr}(\mathrm{A})$.

This is immediate from writing the log determinant as a sum of the log-eigenvalues, and applying the inequality $\log(1+x) \leq x$ to each term in the sum.

Proposition 6 (Tao, 2012, Exercise 1.3.9)**.** *Let* $\mathrm{A}, \mathrm{B} \in \mathbb{R}^{n\times n}$ *PSD, then* $\mathrm{tr}(\mathrm{AB}) \leq \lambda_1(\mathrm{A})\mathrm{tr}(\mathrm{B})$*, where* $\lambda_1(\mathrm{A})$ *denotes the largest eigenvalue of* A.

We now derive the SGPR bound $\log|\mathrm{K}| \leq \log|\mathrm{Q}| + \frac{1}{\sigma^2}\mathrm{tr}(\mathrm{K}-\mathrm{Q})$; along the way we re-derive a tighter lower bound on $\log|\mathrm{K}|$ given in Shi et al. (2020, Appendix A) with computational complexity $O(n^2 m)$.

$$\begin{aligned}\log|\mathrm{K}| &= \log|\mathrm{Q}| + \log|\mathrm{Q}^{-1/2}\mathrm{K}\mathrm{Q}^{-1/2}| \\ &= \log|\mathrm{Q}| + \log|\mathrm{I} + \mathrm{Q}^{-1/2}(\mathrm{K}-\mathrm{Q})\mathrm{Q}^{-1/2}| \\ &\leq \log|\mathrm{Q}| + \mathrm{tr}\left(\mathrm{Q}^{-1}(\mathrm{K}-\mathrm{Q})\right). \end{aligned} \tag{7}$$

The first equality uses that the determinant is multiplicative. The inequality combines proposition 5 with the cyclic property of trace (proposition 1). Because, $\log(1+x) \leq x$ is only tight when $x \approx 0$, this is only tight when $\mathrm{Q}^{-1/2}(\mathrm{K}-\mathrm{Q})\mathrm{Q}^{-1/2}$ has exclusively small eigenvalues. Equation (7) coincides with the lower bound given in Shi et al. (2020). We then can use proposition 6,

$$\begin{aligned}\mathrm{tr}\left(\mathrm{Q}^{-1}(\mathrm{K}-\mathrm{Q})\right) &\leq \lambda_1(\mathrm{Q}^{-1})\mathrm{tr}\left(\mathrm{K}-\mathrm{Q}\right) \\ &\leq \frac{1}{\sigma^2}\mathrm{tr}\left(\mathrm{K}-\mathrm{Q}\right). \end{aligned} \tag{8}$$

Equation (8) is the SGPR bound stated above, and can be computed in $O(nm^2)$.

The bound on the log-determinant term we used in lemma 1 is derived by a modification of above argument. We would like to replace the inequality in eq. (7) with a tighter inequality. We use a version of the arithmetic-geometric mean inequality for positive definite matrices.

Proposition 7 (AM-GM inequality for log determinant; Vakili et al., 2020, Lemma 1)**.** *For* $\mathrm{A} \in \mathbb{R}^{n\times n}$ *a positive definite matrix,*

$$\log\det(\mathrm{A}) \leq n\log\left(\mathrm{tr}(\mathrm{A})/n\right) \tag{9}$$

This proposition can be proved by writing the log determinant in terms of eigenvalues, then applying the arithmetic-geometric mean inequality.

We now state the bound on the log-determinant, which is a corollary of proposition 7.

Lemma 2. *For* $\mathrm{K}, \mathrm{Q} \in \mathbb{R}^{n\times n}$ *PSD such that* $\mathrm{K}-\mathrm{Q}$ *is PSD and* $\mathrm{Q} - \sigma^2\mathrm{I}$ *is PSD,*

$$\log|\mathrm{K}| \leq \log|\mathrm{Q}| + n\log\left(\frac{\mathrm{tr}(\mathrm{Q}^{-1}\mathrm{K})}{n}\right) \tag{10}$$

$$\leq \log|\mathrm{Q}| + n\log\left(1 + \frac{\mathrm{tr}(\mathrm{K}-\mathrm{Q})}{n\sigma^2}\right). \tag{11}$$

Remark 2. *Equation (10) does not assume* $\mathrm{Q} - \sigma^2\mathrm{I}$ *is PSD, but eq. (11) does. In our application* $\mathrm{Q} - \sigma^2\mathrm{I} = \mathrm{Q_{ff}}$ *is PSD.*

Remark 3. *Equation (10) improves upon eq. (7), while eq. (11) improves upon eq. (8). This can be seen by applying* $\log(1+x) \leq x$ *in both cases.*

Proof. We begin as in the proof of the proof of the SGPR bound, but then apply proposition 7 in place of proposition 5:

$$\begin{aligned}\log|\mathrm{K}| &= \log|\mathrm{Q}| + \log|\mathrm{Q}^{-1/2}\mathrm{K}\mathrm{Q}^{-1/2}| \\ &\leq \log|\mathrm{Q}| + n\log\left(\mathrm{tr}(\mathrm{Q}^{-1/2}\mathrm{K}\mathrm{Q}^{-1/2})/n\right) \\ &= \log|\mathrm{Q}| + n\log\left(\mathrm{tr}(\mathrm{Q}^{-1}\mathrm{K})/n\right). \end{aligned} \tag{12}$$

This proves the first statement in the lemma. As in the derivation of eq. (8) from eq. (7), we can apply the bound $\mathrm{tr}(\mathrm{Q}^{-1}\mathrm{K}) = n + \mathrm{tr}(\mathrm{Q}^{-1}(\mathrm{K}-\mathrm{Q})) \leq n + \frac{1}{\sigma^2}\mathrm{tr}(\mathrm{K}-\mathrm{Q})$. This yields the second bound in the lemma. □

Like the SGPR bound on the log-determinant, in order to compute eq. (11), we only need to compute $\log|\mathrm{Q}|$ and $\mathrm{tr}(\mathrm{K}-\mathrm{Q})$. We could use lemma 2 together with the bound on the quadratic form from SGPR, giving a lower bound on the LML that can be computed in $O(nm^2)$, with only minor modifications to standard implementations of SGPR. This bound is included in fig. 1, under the name 'CGLB-v_0-1024'.

In the supplement, we show that eq. (11) can be improved while still maintaining a computational cost of $O(nm^2)$, by phrasing the problem of upper bounding the log-determinant of K as a constrained convex optimisation problem, subject to the constraints that $\mathrm{K}-\mathrm{Q}$ is PSD, and $\mathrm{tr}(\mathrm{K})$ as well as the eigenvalues of Q are known.

3.3. Bounds on the Quadratic Term

We now turn our attention to an upper bound on the quadratic form $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}$. In order to combine the benefits of SGPR and iterative methods we derive an upper bound on $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}$ that is tight if either: 1) $\mathrm{Q}\approx\mathrm{K}$, or 2) we are able to find a $\mathbf{v}\in\mathbb{R}^n$ such that $\mathbf{v}\approx\mathrm{K}^{-1}\mathbf{y}$.

Lemma 3. *Let* $\mathrm{K},\mathrm{Q}\in\mathbb{R}^{n\times n}$ *PD such that* $\mathrm{K}-\mathrm{Q}$ *is PSD. Then, for any* $\mathbf{v},\mathbf{y}\in\mathbb{R}^n$,

$$2\mathbf{y}^\mathsf{T}\mathbf{v}-\mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}\leq\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y} \tag{13}$$
$$\leq\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r}+2\mathbf{y}^\mathsf{T}\mathbf{v}-\mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}, \tag{14}$$

where $\mathbf{r}=\mathbf{y}-\mathrm{K}\mathbf{v}$.

Remark 4. *The lower bound is well-known, and yields the popular interpretation of CG as optimisation of a quadratic function, (e.g. Hackbusch, 1994, Section 9.1.1)*

Choosing $\mathbf{v}=\mathbf{0}$ results in the upper bound $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}\leq\mathbf{y}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{y}$, which corresponds to the quadratic term in the SGPR bound. If we have $m=0$ so that $\mathrm{Q}=\sigma^2\mathrm{I}$, after some rearranging we recover the upper bound considered in Gibbs & Mackay (1997, eq. 50) for monitoring the convergence of CG. On the other hand if $\mathbf{v}=\mathrm{K}^{-1}\mathbf{y}$, the upper and lower bound coincide, implying we have recovered $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}$. This is reminiscent of the approach taken in Van der Wilk et al. (2020), in which an auxiliary matrix $\mathbf{T}$ is introduced yielding a lower bound on the variational bound of Hensman et al. (2015), that is tight when $\mathbf{T}=\mathrm{K}_{\mathrm{uu}}^{-1}$.

Proof of lemma 3. We begin by expanding out the quadratic form,

$$\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}=(\mathbf{r}+\mathrm{K}\mathbf{v})^\mathsf{T}\mathrm{K}^{-1}(\mathbf{r}+\mathrm{K}\mathbf{v})$$
$$=\mathbf{r}^\mathsf{T}\mathrm{K}^{-1}\mathbf{r}+2\mathbf{r}^\mathsf{T}\mathbf{v}+\mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}. \tag{15}$$

Let $\mathbf{w}=\mathrm{K}^{-1}\mathbf{y}$. Then $\mathbf{r}^\mathsf{T}\mathrm{K}^{-1}\mathbf{r}=\mathbf{w}^\mathsf{T}\mathrm{K}\mathbf{w}\geq 0$. This proves the lower bounds in lemma 3, as $2\mathbf{r}^\mathsf{T}\mathbf{v}+\mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}=2\mathbf{y}^\mathsf{T}\mathbf{v}-\mathbf{v}^\mathsf{T}\mathrm{K}\mathbf{v}$.

Adding 0 to the term involving an inverse,

$$\mathbf{r}^\mathsf{T}\mathrm{K}^{-1}\mathbf{r}=\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r}-\mathbf{r}^\mathsf{T}(\mathrm{Q}^{-1}-\mathrm{K}^{-1})\mathbf{r}. \tag{16}$$

From proposition 4 and since $\mathrm{K}-\mathrm{Q}$ is PSD $\mathrm{Q}^{-1}-\mathrm{K}^{-1}$ is PSD. Hence, $\mathbf{r}^\mathsf{T}\mathrm{K}^{-1}\mathbf{r}\leq\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r}$. □

In fig. 1 we select hyperparameters for a Gaussian process regression model on the `poletele` dataset using various lower bounds on the LML formed by combining bounds presented in this and the previous section, with $m=1024$ inducing points. 'SGPR' and 'CGLB-v_0' use the bound $\mathbf{y}\mathrm{K}^{-1}\mathbf{y}\leq\mathbf{y}\mathrm{Q}^{-1}\mathbf{y}$, while the other methods use eq. (14). The figure indicates that introducing the auxiliary vector $\mathbf{v}$ can lead to significantly improvements in the bound. Additionally, comparing the three 'CGLB' bounds shows the improvement from using eq. (10) or eq. (11) instead of eq. (8).

3.4. Approximate Predictive Posterior

After selecting model hyperparameters, we must compute a mean and (co-)variance at test points. We use the same covariance calculation as in SGPR (eq. (4)). For the mean, we would like to recover the exact GP mean if $\mathrm{K}=\mathrm{Q}$ or $\mathbf{v}=\mathrm{K}^{-1}\mathbf{y}$, as in both cases our bound estimates $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}$ exactly. We propose the estimator

$$m(x)=\mathbf{k_{fs}}^\mathsf{T}\mathbf{v}+\mathbf{k_{us}}^\mathsf{T}\mathrm{K}_{\mathrm{uu}}^{-1}\mathrm{K}_{\mathrm{uf}}\mathrm{Q}^{-1}(\mathbf{y}-\mathrm{K}\mathbf{v}), \tag{17}$$

where $\mathbf{k_{us}}$ is as in eq. (4) and $\mathbf{k_{fs}}\in\mathbb{R}^n$ with $\mathbf{k_{fs}}_i=k(x_i,x)$. The first term is the estimate given by running conjugate gradient and replacing $\mathrm{K}^{-1}\mathbf{y}$ with its approximation. The second term is a correction that is equivalent to the mean of SGPR, but replacing $\mathbf{y}$ with the residual of the CG computation. The mean predictor in eq. (17) is not sparse, but the covariance we use is induced in the same way as in the sparse variational framework. The CGLB predictive posterior closely resembles 'decoupled' variational Gaussian

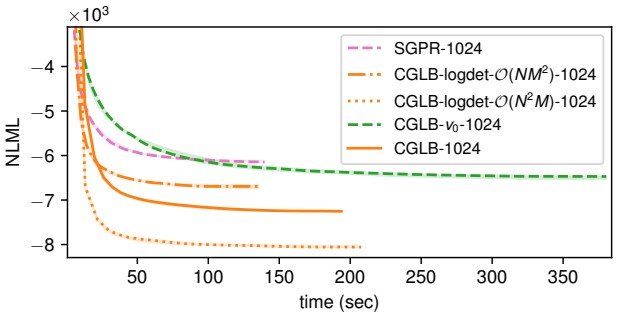

Figure 1. Comparison of the SGPR bound, the CGLB bound with fixed zero v vector (CGLB-v_0), and CGLB bounds with log determinants from eq. (11) (CGLB-1024), eq. (12) (CGLB-logdet-$O(N^2M)$), and eq. (8) (CGLB-logdet-$O(NM^2)$) on the `poletele` dataset.

process inference Cheng & Boots (2017), with the inducing points used to determine the mean function $\mathbf{z}' = \{\mathbf{x}, \mathbf{z}\}$ and the inducing points, $\mathbf{z}$, used to determine the covariance function.

A worthwhile question is whether CGLB has a variational interpretation, i.e. whether the gap between CGLB and the LML is a Kullback-Leibler divergence between an approximate posterior and the posterior. A variational formulation of the bounds would suggest an alternative, likely preferable, predictive distribution to the one we describe in this section.

3.5. Optimisation with Conjugate Gradients

The vector $\mathbf{v} \in \mathbb{R}^n$ is an auxiliary parameter in our lower bound on the LML. From lemma 3, for fixed θ, CGLB is maximized by $\mathbf{v} = \mathrm{K}^{-1}\mathbf{y}$. A simple approach is to treat $\mathbf{v}$ as an additional parameter, and optimise the bound jointly with respect to $\{\theta, \mathbf{v}, (z_i)_{i=1}^m\}$. However, this introduces n additional dimensions to the optimisation problem, and we find it leads to slow parameter learning (fig. 2).

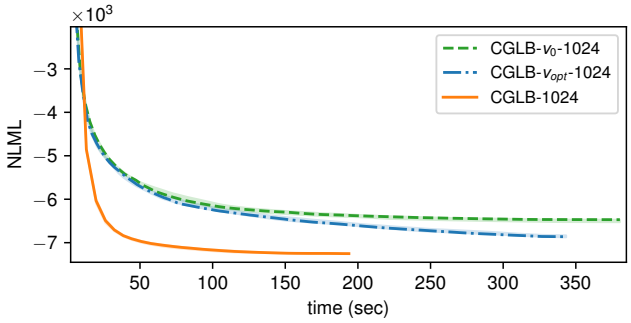

Figure 2. Negative log marginal likelihoods (NLML) for different learning regimes of auxiliary v vector on the `poletele` dataset. The CGLB-v_0 corresponds to the constant zero vector, such that SGPR quadratic term is recovered. The CGLB-v_{opt} includes v vector to the optimisation parameter set. The CGLB (without v suffix) uses CG to tune v vector.

We instead select $\mathbf{v}$ by running conjugate gradient on the system of equations $\mathrm{K}\mathbf{v} = \mathbf{y}$ each time we evaluate the lower bound on the LML. We make several design choices when running conjugate gradients to improve and assess convergence of the approximation.

Preconditioner We use Q as a preconditioner. This preconditioner has been used previously in kernel methods (Cutajar et al., 2016). In our case, this has computational advantages: we can reuse some of the calculations used in bounding the log-determinant.

Initializing CG We initialize CG using the value of $\mathbf{v}$ found in the previous evaluation of the lower bound. As the optimiser often evaluates similar settings of kernel hyperparameters in sequence and, as long as σ^2 is not very close to zero similar parameter settings result in similar optimal values for $\mathbf{v}$, this is often a good guess for the the solution to $\mathrm{K}^{-1}\mathbf{y}$. We show in section 4 for many datasets, after the first handful of optimisation steps the old solution for $\mathbf{v}$ is good enough and no iterations of CG are needed during most steps of parameter optimisation.

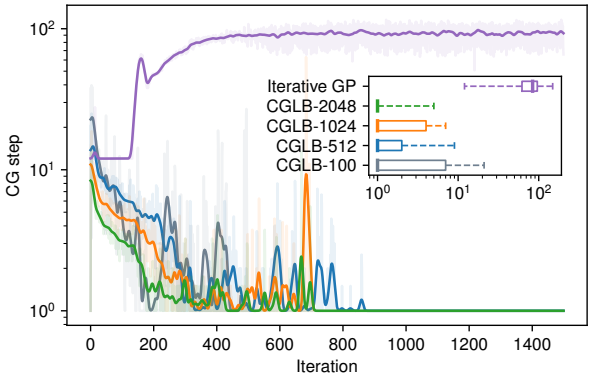

Figure 3. The number of CG iterations spent in CGLB and Iterative GP to achieve a pre-set residual error (note the stopping criteria for the methods is different) on the `protein` dataset. For CGLB this number goes to zero (the plot uses $\log(x+1)$ scaling for the CG step axis), and CGLB reuses the vector v in the subsequent iterations. The figure shows the average number of steps (shaded lines) spent per function evaluation during optimisation for five experiments on a different data splits; solid lines are the smoothing applied to the average number of steps for each model. The inner box-whisker plot depicts IQRs of CG steps that CGLB and Iterative GP models ran throughout 1500 iterations. The whiskers set to 95 percentile. The initial flat line for Iterative GP at 10 iterations is due to a hard-coded constraint in the GPytorch code.

Stopping Criterion We monitor upper and lower bounds on $\mathbf{y}^\mathsf{T}\mathrm{K}^{-1}\mathbf{y}$ in order to decide when to terminate CG, as advocated in Gibbs & Mackay (1997). Subtracting the upper and lower bounds in lemma 3, we derive the stopping criterion $\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r} \leq 2\epsilon$. This ensures that the slack in our bound introduced from not exactly computing the quadratic term is at most ϵ. Because of our choice of preconditioner, we compute $\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r}$ in every iteration, which allows this stopping criterion used with almost no additional computation.

Previous methods (Gardner et al., 2018; Wang et al., 2019) have used eq. (13) have used $\|\mathbf{r}\|_2$ as a stopping criterion. However, this can lead to significant biases in hyperparameter selection. In particular, the bias in the lower bound in eq. (13) is $\mathbf{r}^\mathsf{T}\mathrm{K}^{-1}\mathbf{r}$, which in the worst case can be as large as $\|\mathbf{r}\|_2^2/(2\sigma^2)$. Similarly, in the worst case and when $m = 0$, the upper bound in lemma 3 can have a bias of a similar magnitude in the opposite direction. By instead ensuring $\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r} \leq 2\epsilon$, we ensure the bias we introduce into estimation of the LML through this term in CGLB is

uniformly bounded by ϵ over all hyperparameter settings. We therefore expect this bias to have a smaller impact on the estimation of hyperparameters, in particular σ^2.

3.6. Selecting Optimization Parameters with CGLB versus Iterative Methods

An advantage of our approach compared to existing iterative methods is that the quality of design and optimization choices can be assessed directly through the approximation to the log marginal likelihood, as is done in Cholesky-based implementations of GPR and SGPR. Good choices of the parameters $\{\mathbf{v}, (z_i)_{i=1}^m\}$ result in tighter lower bounds on the log marginal likelihood.

In contrast, when using Iterative GP any bias introduced due to an insufficient number of iterations of CG may lead to either over or under estimation of the log marginal likelihood. In this case, changes to optimization parameters that increase the estimated log marginal likelihood could be indicative of additional bias instead of improved hyperparameter selection. While methods considered in Gibbs & Mackay (1997); Ubaru et al. (2017); Gardner et al. (2018); Wang et al. (2019) are capable of having less bias when estimating the LML than our method (at least for fixed m, θ), any bias introduced is harder to assess. In the next section, we find empirically this makes achieving good performance with these methods more challenging.

4. Experiments

We now compare hyperparameter selection with CGLB (lemma 1, eq. (6)), with sparse methods (SGPR) as well as the conjugate gradient method considered in Wang et al. (2019) (Iterative GP). We would like to determine which method finds better hyperparameters, and how these hyperparameters effect predictive performance. We consider several regression UCI datasets, and compare the methods on the basis of root mean square error (RMSE) and negative log predictive density (NLPD) on held-out data to assess predictive performance. In order to assess the quality of the hyperparameters found by the methods, for datasets where it is computationally feasible, we compare the LML of the model with the hyperparameters selected by each method, computed directly with Cholesky decomposition. Our implementation of CGLB, as well as code for experiments can be found at `https://github.com/awav/CGLB`.

4.1. Data Preparation

We randomly split each dataset into a training set consisting of 2/3 of examples, and a test set consisting of the remaining 1/3. We run 5 seeds in all experiments, with each seed corresponding to a different random split of the dataset. Each input dimension is normalised to have mean 0 and variance 1 within the training set. Similarly, the training outputs are normalised to have 0 mean and variance 1. We apply the same normalisation to test data when making predictions, using the statistics computed on the training data. All metrics reported are on the standardised data, so that we would expect a model predicting a constant mean function to have RMSE near 1.0. Datasets are downloaded using the Bayesian benchmarks package (Salimbeni, 2019).

4.2. Model Class and Initialisation of Parameters

We run experiments with a Matérn 3/2 kernel, with independently learned lengthscales along each input dimension (i.e. automatic relevance detection). While in the derivations we have assume the prior mean is 0, in experiments we take the prior mean to be a constant function that is learned as a hyperparameter. The changes to the predictive distribution and lower bound presented in the previous section to account for this are straightforward. This is the same experimental setup considered in Wang et al. (2019).

All kernel lengthscales, the kernel variance and the likelihood variance are initialised at 1.0. The prior mean is initialised at 0. We use soft-plus constraints on the lengthscales and likelihood, lower bounding them at $1\mathrm{e}{-6}$ for SGPR and CGLB and $1\mathrm{e}{-4}$ for iterative GP, as we found this was necessary for numerical stability. Experiments are run using double precision.

4.3. Training Procedure

For CGLB and SGPR we vary the number of inducing points between 512 and 4096. We train CGLB and SGPR with the L-BFGS optimiser (Liu & Nocedal, 1989) for either 2000 steps or until the optimiser stops due to a small projected gradient norm or being unable to find a point that improves upon the current value during line-search. We initialise $(z_i)_{i=1}^m$ using the 'Greedy' method advocated for in Burt et al. (2020), then optimise them jointly with hyperparameters. Alternatives to joint optimisation, such as reinitialisation of inducing points as suggested in Burt et al. (2020) may lead to further improvements in the training procedure, though we do not explore those here. For CGLB we stop CG when the residual $\mathbf{r}$ satisfied $\frac{1}{2}\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r} \leq \epsilon = 1.0$ during training and when $\frac{1}{2}\mathbf{r}^\mathsf{T}\mathrm{Q}^{-1}\mathbf{r} \leq \epsilon = 1\mathrm{e}{-3}$ when selecting $\mathbf{v}$ for use in predictions. The criteria $\epsilon = 1.0$ used during training ensures that the additional slack introduced into CGLB from estimating the quadratic term is at most 1.0, regardless of the current value of θ.

We follow the procedure described in Wang et al. (2019) to train Iterative GP. We first pre-train the model on a subset of 10000 observations with 10 iterations of L-BFGS. Then we run 10 iterations of Adam optimiser (Kingma & Ba, 2015) on the same subset, followed by 2000 iterations of

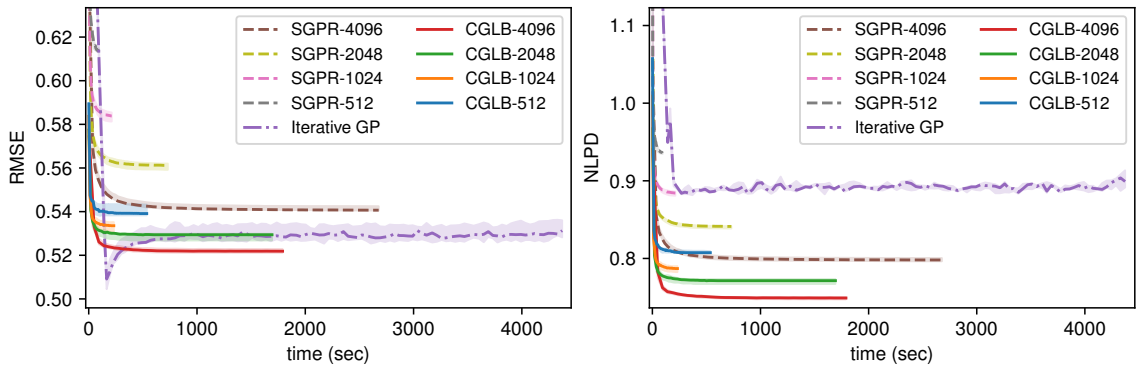

Figure 4. Test root mean square error (RMSE) and negative log predictive density (NLPD) metrics of CGLB, SGPR and Iterative GP models computed on the `protein` dataset. The shaded area is IQR region, and the line is median over five experiment trials with different dataset splits.

Adam with 0.1 learning rate on the full dataset. For forming Hutchinson's trace estimation, 10 vectors $\{\mathbf{p}_i\}_{i=1}^{10}$ are sampled. In order to run L-BFGS, the vectors $\{\mathbf{p}_i\}_{i=1}^{10}$ are fixed along each line search, as suggested in the documentation for GPytorch (Gardner et al., 2018).

For experiments with SGPR we use GPflow (Matthews et al., 2017). For small datasets ($n < 20000$), we use a GPflow implementation of CGLB as we found this to be faster than our GPytorch implementation. However, for larger datasets we use GPytorch kernel abstractions with KeOps (Charlier et al., 2021) to perform matrix-vector operations to reduce memory requirements. For Iterative GPs we use the GPytorch implementation (Gardner et al., 2018). Following Wang et al. (2019), we use the Lanczos variance estimator (without kernel interpolation) introduced in Pleiss et al. (2018) for computing the predictive NLPD for Iterative GPs. We run all experiments on a single Tesla V100-32GB GPU.

4.4. Results

In fig. 3, we show that CGLB typically only needs a single matrix-vector multiply with the kernel matrix, and reuses $\mathbf{v}$ in most training steps. Adding more inducing points, which results in a better preconditioner, decreases the number of steps of optimisation before we use zero CG steps in most subsequent parameter updates. Iterative GP requires a considerably greater number of CG steps, particularly in later training steps.

In fig. 4, we demonstrate the predictive performance of CGLB during training and compare it with Iterative GP and SGPR on the `protein` UCI dataset (Dua & Graff, 2017). After splitting, the `protein` dataset has 29267 training examples, 16463 testing examples and input dimensionality 9. The predictive RMSE of CGLB is either better or equal to Iterative GP and consistently outperforms SGPR. Moreover, CGLB shows improved predictive uncertainty estimations as measured by NLPD compared to Iterative GP and SGPR. For CGLB and SGPR, increasing the number of inducing points leads to large performance gains. SGPR is faster per iteration than CGLB for a fixed m; however, achieving comparable performance to CGLB requires many more inducing points with SGPR on several datasets (`bike`, `kin40k`, `protein`).

Results for other datasets are shown in table 1. For small datasets where we can compute the log marginal likelihood using Cholesky-based methods, we see that CGLB often finds settings of hyperparameters with higher LML than competing methods, suggesting that the gains in predictive performance are attributable to improved hyperparameter selection. On the `bike` dataset, Iterative GP significantly overestimates the LML. We hypothesise this is due to the interaction between bias and optimisation that can occur when maximising an estimate that is not a lower bound. The supplementary material shows model performance over time for several more regression datasets.

For several datasets, we found that Iterative GP achieved good test performance as measured by RMSE after a handful of iterations, but that continuing to optimise the objective led to a small drop in performance, e.g. fig. 4. We initially thought this might be due to a high learning rate. However, in the supplement, we show that even if the learning rate for Adam is reduced from 0.1 to 0.01 the same phenomenon occurs, but convergence is slower. In contrast, we do not see this behaviour for SGPR or CGLB.

Further, we observed unusual behaviour of the Iterative GP method on some datasets when the positive constraint of likelihood noise is set to a low number. In Iterative GP experiments, we used the GPytorch default noise value $1\mathrm{e}{-4}$. We attempted to lower this constraint to $1\mathrm{e}{-6}$ as the noise variance on several datasets selected by other methods, particularly `bike` was generally below $1\mathrm{e}{-4}$, and this low noise variance led to improved predictive performance. However, Iterative GP training and predictive performance

Table 1. Median LML, predictive NLPD and predictive RMSE over five datasets splits for Iterative GP, SGPR and CGLB. Cholesky subcolumns represent the same metrics evaluated by using Cholesky-based GPR implementation with hyperparameters found by Iterative GP, SGPR and CGLB models accordingly. The value of n listed in the right column, is the total number of datapoints, including both the test and train set. CGLB with $m = 4096$ is missing for `keggundirected` due to the high memory requirement.

		LML		NLPD		RMSE	
		Approx	Cholesky	Approx	Cholesky	Approx	Cholesky
`bike` n=17379, d=17	Iterative GP	30992.8	31319.1	-2.016	-3.257	0.020	0.014
	SGPR-4096	30502.5	32814.2	-3.280	-3.336	0.010	0.010
	CGLB-4096	37732.7	**42023.0**	**-4.216**	-4.329	0.004	0.004
	CGLB-2048	34102.8	38936.7	-3.811	-3.972	**0.003**	0.003
	CGLB-1024	30493.9	35351.8	-3.403	-3.615	0.005	0.005
`elevators` n=16599, d=18	Iterative GP	-4709.0	-4705.1	0.407	0.384.	**0.353**	0.353
	SGPR-4096	-4675.3	**-4653.3**	**0.386**	0.386	0.354	0.354
	CGLB-4096	-4669.8	-4659.1	**0.386**	0.386	0.354	0.354
	CGLB-2048	-4677.9	-4656.4	0.387	0.387	0.355	0.355
	CGLB-1024	-4712.0	-4670.0	0.392	0.391	0.356	0.356
`poletele` n=15000, d=26	Iterative GP	13552.5	-7641.5	-0.935	1.217	0.079	0.078
	SGPR-4096	9057.7	9624.0	-1.172	-1.180	0.078	0.078
	CGLB-4096	9377.1	**9862.2**	**-1.201**	-1.203	**0.077**	0.077
	CGLB-2048	8248.6	9248.0	-1.126	-1.145	0.080	0.080
	CGLB-1024	7250.7	8694.2	-1.057	-1.098	0.083	0.083
`kin40k` n=40000, d=8	Iterative GP	23859.0	—	-0.454	—	0.087	—
	SGPR-4096	7486.0	—	-0.705	—	0.107	—
	CGLB-4096	12244.2	—	**-0.919**	—	**0.086**	—
	CGLB-2048	10028.6	—	-0.826	—	0.088	—
	CGLB-1024	7260.0	—	-0.714	—	0.093	—
`protein` n=45730, d=9	Iterative GP	-25703.9	—	0.897	—	0.531	—
	SGPR-4096	-27714.6	—	0.798	—	0.541	—
	CGLB-4096	-26570.7	—	**0.749**	—	**0.522**	—
	CGLB-2048	-27442.0	—	0.771	—	0.529	—
	CGLB-1024	-28243.7	—	0.790	—	0.535	—
`keggundirected` n=63608, d=27	Iterative GP	-21659.7	—	1.310	—	**0.117**	—
	SGPR-4096	-29837.5	—	**-0.710**	—	0.118	—
	CGLB-4096	—	—	—	—	—	—
	CGLB-2048	-29751.7	—	-0.708	—	0.119	—
	CGLB-1024	-29728.9	—	-0.707	—	0.120	—

became unstable in later optimisation steps. Neither CGLB nor SGPR exhibited degradation in predictive performance during optimisation, even in cases where the noise level was near the lower bound of $1\mathrm{e}{-6}$. We hypothesise this is further evidence that lower bound maximisation is more robust as an optimisation procedure than other biased estimates of the LML, especially methods that optimise an objective that can overestimate the LML in ways that are non-uniform in θ (c.f. discussion of stopping criteria in section 3.5).

5. Conclusion

CGLB combines benefits from sparse variational methods with iterative approaches for hyperparameter selection. Maximising CGLB alleviates some of the hyperparameter bias that results from ELBO maximisation with sparse methods. As CGLB is deterministic, optimisation is frequently easier than using the stochastic approximations to the LML given by existing Iterative GP methods. Additionally, as the bias in CGLB results in a lower bound on the LML, we can assess choices made regarding optimisation using only training data, as higher lower bounds correlate well with better hyperparameter selection.

Acknowledgements

The authors would like to thank an anonymous reviewer for pointing us to the lower bound presented in Shi et al. (2020), which led to our proof of eq. (10). We would also like to thank Ke Alexander Wang for clarifying several points regarding the training procedure for Iterative GPs and Andrew Y.K. Foong and David Janz for providing feedback on an earlier draft of this work. DRB acknowledges funding from the Qualcomm Innovation Fellowship and the Williams College Herchel Smith Fellowship.

References

Avron, H. and Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-

definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34, 2011.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*, 2016.

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. Convergence of sparse variational inference in Gaussian processes regression. *Journal of Machine Learning Research*, 21(131):1–63, 2020.

Charlier, B., Feydy, J., Glaunès, J. A., Collin, F.-D., and Durif, G. Kernel operations on the GPU, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 20, 2021.

Cheng, C.-A. and Boots, B. Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, 2017.

Cutajar, K., Osborne, M., Cunningham, J., and Filippone, M. Preconditioning kernel matrices. In *International Conference on Machine Learning*, 2016.

Davies, A. *Effective Implementation of Gaussian Process Regression for Machine Learning*. PhD Thesis, University of Cambridge, 2015.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. Physicochemical Properties of Protein Tertiary Structure Data Set.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Gibbs, M. and Mackay, D. Efficient implementation of Gaussian processes. Technical report, Cavendish Laboratory, University of Cambridge, 1997.

Hackbusch, W. *Iterative solution of large sparse systems of equations*. Springer, 1994.

Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360, 2015.

Hestenes, M. R. and Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

Horn, R. A. and Johnson, C. R. *Matrix analysis*. Cambridge University Press, 2012.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

Meanti, G., Carratino, L., Rosasco, L., and Rudi, A. Kernel methods through the roof: Handling billions of points efficiently. In *Advances in Neural Information Processing Systems*, 2020.

Pleiss, G., Gardner, J. R., Weinberger, K. Q., and Wilson, A. G. Constant-time predictive distributions for gaussian processes. In *International Conference on Machine Learning*, 2018.

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

Salimbeni, H. Bayesian benchmarks. https://github.com/hughsalimbeni/bayesian_benchmarks, 2019.

Shi, J., Titsias, M., and Mnih, A. Sparse orthogonal variational inference for gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, 2005.

Tao, T. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, 2009.

Ubaru, S., Chen, J., and Saad, Y. Fast estimation of $\mathrm{tr}(f(a))$ via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Vakili, S., Khezeli, K., and Picheny, V. On information gain and regret bounds in Gaussian process bandits. *arXiv preprint arXiv:2009.06966*, 2020.

van der Wilk, M., John, S. T., Artemev, A., and Hensman, J. Variational Gaussian process models without matrix inverses. In *Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference*, Proceedings of Machine Learning Research, 2020.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact Gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, 2019.

Williams, C. K. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, 2001.