Large Linear Multi-output Gaussian Process Learning for Time Series

Anonymous Author(s)
Affiliation
Address
email

Abstract

Gaussian processes, or distributions over arbitrary functions in a continuous domain, can be generalized to the multi-output case: a linear model of coregionalization (LMC) is one approach [1]. LMCs estimate and exploit correlations across the multiple outputs. While model estimation can be performed efficiently for single-output GPs [2], these assume stationarity, but in the multi-output case the cross-covariance interaction is not stationary. We propose Large Linear GPs (LLGPs), which circumvent the need for stationarity by using LMC's structure, enabling optimization of GP hyperparameters for multi-dimensional outputs and one-dimensional inputs. When applied to real time series data, we find our theoretical improvement relative to the current state of the art is realized with LLGP being generally an order of magnitude faster while improving or maintaining predictive accuracy.

1 Introduction

Gaussian processes (GPs) are a nonlinear regression method that capture function smoothness across inputs through a response covariance function [3]. GPs extend to multi-output regression, where the objective is to build a probabilistic regression model over vector-valued observations by identifying latent cross-output processes. Multi-output GPs frequently appear in time-series contexts, such as the problem of imputing missing temperature readings for sensors in different locations or missing foreign exchange rates and commodity prices given rates and prices for other goods over time [4, 5]. Efficient model estimation would enable researchers to quickly explore large spaces of parameterizations to find an appropriate one for their task.

For n input points, exact GP inference requires maintaining an n^2 matrix of covariances between response variables at each input and performing $O(n^3)$ inversions with that matrix [3]. Some single-output GP methods exploit structure in this matrix to reduce runtime [2]. In the multi-output setting, the same structure does not exist. Approximations developed for multi-output methods instead reduce the dimensionality of the GP estimation problem from n to $m < n$, but still require m to scale with n to retain accuracy [6]. The polynomial dependence on m is still cubic: the matrix inversion underlying the state-of-the-art multi-output GP estimation method ignores LMC's structure. Here, we exploit this structure to avoid direct matrix inversion.

Our paper is organized as follows. In Sec. 2 we give background on single-output and multi-output GPs, as well as some history in exploiting structure for matrix inversions in GPs. Sec. 3 details both related work that was built upon in LLGP and existing methods for multi-output GPs, followed by Sec. 4 describing our contributions. Sec. 5 describes our method. Then, in Sec. 6 we compare the performance of LLGP to existing methods and offer concluding remarks in Sec. 7.

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

2 Background

2.1 Gaussian processes (GPs)

A GP is a set of random variables (RVs) $\{y_{\mathbf{x}}\}_{\mathbf{x}}$ indexed by $\mathbf{x} \in \mathcal{X}$, with the property that, for any finite collection $X = \{\mathbf{x}_i\}_{i=1}^n$ of $\mathcal{X}$, the RVs are jointly Gaussian [3]. With zero mean without loss of generality and a prespecified covariance $K : \mathcal{X}^2 \to \mathbb{R}$, $\mathbf{y}_X \sim N(\mathbf{0}, K_{X,X})$, where $(\mathbf{y}_X)_i = y_{\mathbf{x}_i}$ and $(K_{X,X})_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Given observations of $\mathbf{y}_X$, inference at a single point $* \in \mathcal{X}$ of an $\mathbb{R}$-valued RV y_* is performed by the marginalization $y_*|\mathbf{y}_X$ [3]. Predictive accuracy is sensitive to a particular parameterization of our kernel, and model estimation is performed by maximizing data log likelihood with respect to parameters $\boldsymbol{\theta}$ of K, $\mathcal{L}(\boldsymbol{\theta}) = \log p(\mathbf{y}_X|X, \boldsymbol{\theta})$. Gradient-based optimization methods then require the gradient with respect to every parameter θ_j of $\boldsymbol{\theta}$:

$$\partial_{\theta_j}\mathcal{L} = \frac{1}{2}\boldsymbol{\alpha}^\top \partial_{\theta_j} K_{|\boldsymbol{\theta}}\boldsymbol{\alpha} - \frac{1}{2}\operatorname{tr}\left(K_{|\boldsymbol{\theta}}^{-1}\partial_{\theta_j}K_{|\boldsymbol{\theta}}\right); \quad \boldsymbol{\alpha} = K_{|\boldsymbol{\theta}}^{-1}\mathbf{y}. \tag{1}$$

2.2 Multi-output linear GPs

We build multi-output GP models as instances of general GPs, where a multi-output model explicitly represents correlations between outputs through a shared input space [1]. Here, for D outputs, we write our indexing set as $\mathcal{X}' = [D] \times \mathcal{X}$, a point from a shared domain coupled with an output index. Then, if we make observations at $X_d \subset \mathcal{X}$ for output $d \in [D]$, we can set:

$$\mathbf{X} = \{(d, x) \,|\, d \in [D], x \in X_d\} \subset \mathcal{X}'; \quad n = |\mathbf{X}|\,.$$

An LMC kernel K is of the form

$$K([i, \mathbf{x}_i], [j, \mathbf{x}_j]) = \sum_{q=1}^{Q} b_{ij}^{(q)} k_q(\|\mathbf{x}_i - \mathbf{x}_j\|) + \epsilon_i 1_{i=j}, \tag{2}$$

where $k_q : \mathbb{R} \to \mathbb{R}$ is a stationary kernel on $\mathcal{X}$. Typically, the positive semi-definite (PSD) matrices $B_q \in \mathbb{R}^{D\times D}$ formed by $b_{ij}^{(q)}$ are parameterized as $A_q A_q^\top + \boldsymbol{\kappa}_q I_D$, with $A_q \in \mathbb{R}^{D\times R_q}$, $\boldsymbol{\kappa}_q \in \mathbb{R}_+^D$ and R_q a preset rank. Importantly, even though each k_q is stationary, K is only stationary on $\mathcal{X}'$ if B_q is Toeplitz. In practice, where we wish to capture covariance across outputs as a D^2-dimensional latent process, B_q is not Toeplitz, so $K([i, \mathbf{x}_i], [j, \mathbf{x}_j]) \neq K([i+1, \mathbf{x}_i + 1], [j+1, \mathbf{x}_j + 1])$.

The LMC kernel provides a flexible way of specifying multiple additive contributions to the covariance between two inputs for two different outputs. The contribution of the qth kernel k_q to the covariance between the ith and jth outputs is then specified by the multiplicative factor $b_{ij}^{(q)}$. By choosing B_q to have rank $R_q = D$, the corresponding LMC model can have any positive contribution between two outputs that best fits the data, so long as B_q remains PSD. By reducing the rank R_q, the interactions of the outputs have lower-rank latent processes, with rank 0 indicating no interaction (i.e., if $A = 0$, then we have an independent GP for each output).

2.3 Structured covariance matrices

If we can identify structure in the covariance K, then we can develop fast in-memory representations and efficient matrix-vector multiplications (MVMs) for K—this has been used in the past to accelerate GP model estimation [7, 8]. The Kronecker product $A \otimes B$ of matrices of order a, b is a block matrix of order ab with ijth block $A_{ij}B$. We can represent the product by keeping representations of A and B separately, rather than the product. Then, the corresponding MVMs can be computed in time $O(a\,\mathrm{MVM}(B) + b\,\mathrm{MVM}(A))$, where $\mathrm{MVM}(\cdot)$ is the runtime of a MVM. For GPs on uniform dimension-P grids, this reduces the runtime of finding $\mathcal{L}$ from $O(n^3)$ to $O(n^{3/P})$ [7].

Symmetric Toeplitz matrices T are constant along their diagonal and fully determined by their top row $\{T_{1j}\}_{j=1}^n$, yielding an $O(n)$ representation. Such matrices arise naturally when we examine the covariance matrix induced by a stationary kernel k applied to a one-dimensional grid of inputs. Since the difference in adjacent inputs $t_{i+1} - t_i$ is the same for all i, we have the Toeplitz property that:

$$T_{(i+1)(j+1)} = k(|t_{i+1} - t_{j+1}|) = k(|t_i - t_j|) = T_{ij}.$$

Furthermore, we can embed T in the upper-left corner of a circulant matrix C of twice its size, which enables MVMs $C\begin{pmatrix}\mathbf{x} & \mathbf{0}\end{pmatrix}^\top = \begin{pmatrix}T\mathbf{x} & \mathbf{0}\end{pmatrix}^\top$ in $O(n \log n)$ time. This approach has been used for fast inference in single-output GP time series with uniformly spaced inputs [8].

3 Related work

3.1 Approximate inference methods

Inducing point approaches create a tractable model to approximate the exact GP. For example, the deterministic training conditional (DTC) for a single-output GP fixes inducing points $T \subset \mathcal{X}$ and estimates kernel hyperparameters for $\mathbf{y}_X|\mathbf{y}_T \sim N(K_{X,T}K_{T,T}^{-1}\mathbf{y}_T, \sigma^2)$ [9]. This approach is agnostic to kernel stationarity, so one may use inducing points for all outputs $\mathbf{T} \subset \mathcal{X}'$, with the model equal to the original GP model when $\mathbf{T} = \mathbf{X}$ [5]. Computationally, these approaches resemble making rank-m approximations to the $n \times n$ covariance matrix.

In Collaborative Multi-output Gaussian Processes (COGP), multi-output GP algorithms further share an internal representation of the covariance structure among all outputs at once [6]. COGP fixes inducing points $\mathbf{T} = [D] \times T$ for some m-sized $T \subset \mathcal{X}$ and puts a GP prior on $\mathbf{y}_\mathbf{T}$ with a restricted LMC kernel that matches the Semiparametric Latent Factor Model (SLFM) [10]. Applying the COGP prior to $\mathbf{y}_X$ corresponds to an LMC kernel (Eq. 2) where $\boldsymbol{\kappa}_q$ is set to 0 and $A_q = \mathbf{a}_q \in \mathbb{R}^{D\times 1}$. Moreover, SLFM and COGP models include an independent GP for each output, represented in LMC as additional kernels $\{k_d\}_{d=1}^D$, where $A_d = 0$ and $\boldsymbol{\kappa}_d = \mathbf{e}_d \in \mathbb{R}^D$. COGP uses its shared structure to derive efficient expressions for variational inference for parameter estimation.

3.2 Approaches for stationary kernels

3.2.1 Structured Kernel Interpolation (SKI)

SKI abandons the inducing-point approach: instead of using an intrinsically sparse model, SKI approximates the original $K_{X,X}$ directly [11]. To do this efficiently, SKI relies on the differentiability of K. For $\mathbf{x}, \mathbf{z}$ within a grid U, $|U| = m$, and $W_{\mathbf{x},U} \in \mathbb{R}^{1\times m}$ as the cubic interpolation weights [12], $|K_{\mathbf{x},\mathbf{z}} - W_{\mathbf{x},U}K_{U,\mathbf{z}}| = O(m^{-3})$. The simultaneous interpolation $W \triangleq W_{X,U} \in \mathbb{R}^{n\times m}$ then yields the SKI approximation: $K_{X,X} \approx WK_{U,U}W^\top$. W has only $4^d n$ nonzero entries, with $\mathcal{X} = \mathbb{R}^d$. Even without relying on structure, SKI reduces the representation of $K_{X,X}$ to an m-rank matrix.

Massively Scalable Gaussian Processes (MSGP) exploits structure as well: the kernel $K_{U,U}$ on a grid has Kronecker and Toeplitz matrix structure [2]. Drawing on previous work on structured GPs [8, 7], MSGP uses linear conjugate gradient descent as a method for evaluating $K_{|\boldsymbol{\theta}}^{-1}\mathbf{y}$ efficiently for Eq. 1. In addition, an efficient eigendecomposition that carries over to the SKI kernel for the remaining $\log\left|K_{|\boldsymbol{\theta}}\right|$ term in Eq. 1 has been noted previously [13].

Although evaluating $\log\left|K_{|\boldsymbol{\theta}}\right|$ is not feasible in the LMC setting because the LMC sum breaks Kronecker and Toeplitz structure, the approach of creating structure with SKI carries over to LLGP.

4 Contributions of LLGP

First, we identify a block-Toeplitz structure induced by the LMC kernel evaluated on a uniformly spaced grid of inputs. Next, we show how an LMC kernel can be decomposed into two block diagonal components, one of which has structure similar to that of SLFM [10]. Both of these structures coordinate for fast matrix-vector multiplication $K\mathbf{z}$ with the covariance matrix K for any vector $\mathbf{z}$.

With multiple outputs, non-differentiable cross-covariance interactions violate the assumptions of SKI's cubic interpolation. We show a modification to SKI is compatible with the piecewise-differentiable LMC kernel. This approximation induces the previously-identified block-Toeplitz structure of the LMC kernel, enabling acceleration of GP estimation for non-uniform inputs.

For low-dimensional inputs, the above contributions offer an asymptotic and practical runtime improvement in hyperparameter estimation while also expanding the feasible kernel families to any differentiable LMC kernels, relative to COGP (Tab. 1) [6].

5 Large Linear GP

We propose a linear model of coregionalization (LMC) method based on recent structure-based optimizations for GP estimation instead of variational approaches. Critically, the accuracy of the method need not be reduced by keeping the number of interpolation points m low, because its reliance on structure allows better asymptotic performance. For simplicity, our work focuses on multi-dimensional outputs, one-dimensional inputs, and Gaussian likelihoods.

For a given $\boldsymbol{\theta}$, we construct an operator $\tilde{K}_{|\boldsymbol{\theta}}$ which approximates MVMs with the covariance matrix, $K_{|\boldsymbol{\theta}}\mathbf{z} \approx \tilde{K}_{|\boldsymbol{\theta}}\mathbf{z}$. Using only the action of MVM with the covariance operator, we derive $\nabla\mathcal{L}(\boldsymbol{\theta})$. Critically, we cannot access $\mathcal{L}$ itself, only $\nabla\mathcal{L}$, so we choose AdaDelta as a gradient-only high-level optimization routine for $\mathcal{L}$ [14].

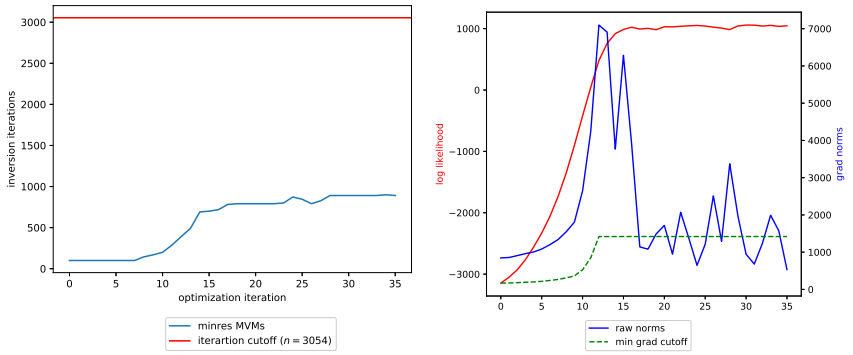
5.1 Gradient construction

Gibbs and MacKay [15] describe the algorithm for GP model estimation in terms of only MVMs with the covariance matrix. In particular, they note that we can solve for $\boldsymbol{\alpha}$ satisfying $K_{|\boldsymbol{\theta}}\boldsymbol{\alpha} = \mathbf{y}$ in Eq. 1 using linear conjugate gradient descent (LCG). Moreover, they develop a stochastic approximation by introducing RV $\mathbf{r}$ with $\operatorname{cov}\mathbf{r} = I$:

$$\operatorname{tr}\left(K_{|\boldsymbol{\theta}}^{-1}\partial_{\theta_j}K_{|\boldsymbol{\theta}}\right) = \mathbb{E}\left[(K_{|\boldsymbol{\theta}}^{-1}\mathbf{r})^\top\partial_{\theta_j}K_{|\boldsymbol{\theta}}\mathbf{r}\right]. \tag{3}$$

This approximation improves as the size of $K_{|\boldsymbol{\theta}}$ increases, so, as in other work [16], we let $\mathbf{r} \sim \operatorname{Unif}\{\pm 1\}$ and take a fixed number N samples from $\mathbf{r}$.

We depart from Gibbs and MacKay in two important ways (Algorithm 1). First, we do not construct $K_{|\boldsymbol{\theta}}$, but instead keep a low-memory representation $\tilde{K}_{|\boldsymbol{\theta}}$ (Sec. 5.2). Second, we use MINRES instead of LCG as the Krylov-subspace inversion method used to compute inverses from MVMs. Iterative MINRES solutions to numerically semidefinite matrices monotonically improve in practice, as opposed to LCG [17]. This is essential in GP optimization, where the diagonal noise matrix $\boldsymbol{\epsilon}$, iid for each output, shrinks throughout learning. Inversion-based methods then require additional iterations because κ_2, the spectral condition number of $K_{|\boldsymbol{\theta}}$, increases as $K_{|\boldsymbol{\theta}}$ becomes less diagonally dominant (Fig. 1).



(a) (b)

Figure 1: Trace of (a) the number of MVMs that MINRES must perform to invert $K_{|\boldsymbol{\theta}}^{-1}\mathbf{y}$ and (b) $\mathcal{L}$, $\|\nabla\mathcal{L}\|$ given $\boldsymbol{\theta}$ at each optimization iteration for a GP applied to the dataset in Sec. 6.1. In (b), in green, we have 20% of the rolling maximum ∞-norm of previous gradients.

Every AdaDelta iteration (invoking Algorithm 1) then takes total time $\tilde{O}(\mathrm{MVM}(\tilde{K}_{|\boldsymbol{\theta}})\sqrt{\kappa_2})$ [18]. This analysis holds as long as the error in the gradients is fixed and we can compute MVMs with the matrix $\partial_{\theta_j}K_{|\boldsymbol{\theta}}$ for each j at least as fast as $\mathrm{MVM}(\tilde{K}_{|\boldsymbol{\theta}})$. Indeed, we assume a differentiable kernel and then recall that applying the linear operator ∂_{θ_j} will maintain the structure of $K_{|\boldsymbol{\theta}}$.

For a gradient-only stopping heuristic, we maintain the running maximum gradient ∞-norm. If gradient norms drop below a proportion of the running max norm more than a pre-set number of times, we terminate (Fig. 1).

Algorithm 1 Compute an approximation of ∇L. Assume MINRES is the inversion routine. We also assume we have access to linear operators D_{θ_j}, representing matrices $\partial_{\theta_j} \tilde{K}_{|\boldsymbol{\theta}}$.

procedure LLGP($\tilde{K}_{|\boldsymbol{\theta}}$, $\mathbf{y}$, N, $\{D_{\theta_j}\}$)
  $R \leftarrow \{\mathbf{r}_i\}_{i=1}^N$, sampling $\mathbf{r} \sim \mathrm{Unif}\{\pm 1\}$.
  for $\mathbf{z}$ in $\{\mathbf{y}\} \cup R$, in parallel **do**
    $K^{-1}\mathbf{z} \leftarrow \text{MINRES}(\tilde{K}_{|\boldsymbol{\theta}}, \mathbf{z})$.
  end for
  $g \leftarrow \mathbf{0}$
  for θ_j in $\boldsymbol{\theta}$ **do** ▷ Compute $\partial_{\theta_j} \mathcal{L}$
    $t \leftarrow \frac{1}{N} \sum_{i=1}^N \left(K^{-1}\mathbf{r}_i\right) \cdot D_{\theta_j}(\mathbf{r}_i)$ ▷ t approximates the trace term of Eq. 3
    $g_j \leftarrow \frac{1}{2} \left(K^{-1}\mathbf{y}\right) \cdot \tilde{K}_{|\boldsymbol{\theta}} \left(K^{-1}\mathbf{y}\right) - \frac{1}{2} t$ ▷ Eq. 1
  end for
  return g
end procedure

5.2 Fast MVMs and parsimonious kernels

The bottleneck of Algorithm 1 is the iterative MVM operations in MINRES. Since $K_{|\boldsymbol{\theta}}$ only enters computation as an MVM operator, the required memory is dictated by its representation $\tilde{K}_{|\boldsymbol{\theta}}$, which need not be dense as long as we can perform MVM with any vector to arbitrary, fixed precision.

When LMC kernels are evaluated on a grid of points for each output, so $X_d = U$, the simultaneous covariance matrix equation without noise over $\mathbf{U}$ (Eq. 4) holds for Toeplitz matrices K_q formed by the stationary kernels k_q evaluated the shared interpolating points U for all outputs:

$$K_{\mathbf{U},\mathbf{U}} = \sum_q (A_q A_q^\top + \operatorname{diag} \boldsymbol{\kappa}_q) \otimes K_q. \tag{4}$$

To adapt SKI to our context of multi-output GPs, we build a grid $\mathbf{U} \subset \mathcal{X}'$ out of a common subgrid $U \subset \mathcal{X}$ that extends to all outputs with $\mathbf{U} = [D] \times U$. Since the LMC kernel evaluated between two sets of outputs K_{X_i,X_j} is differentiable, as long as U spans a range larger than each $\{X_d\}_{d\in[D]}$, the corresponding SKI approximation (Eq. 5) holds with the same asymptotic convergence cubic in $1/m$.

$$K_{\mathbf{X},\mathbf{X}} \approx W K_{\mathbf{U},\mathbf{U}} W^\top + \boldsymbol{\epsilon}. \tag{5}$$

We cannot fold $\boldsymbol{\epsilon}$ into the interpolated term $K_{\mathbf{U},\mathbf{U}}$ since it does not correspond to a differentiable kernel. However, since $\boldsymbol{\epsilon}$ is diagonal, it is efficient to represent and multiply with. Then, the MVM $K_{\mathbf{X},\mathbf{X}}\mathbf{z}$ can be approximated by $W K_{\mathbf{U},\mathbf{U}} W^\top \mathbf{z} + \boldsymbol{\epsilon}\mathbf{z}$, where matrix multiplication by the sparse matrices $\boldsymbol{\epsilon}, W, W^\top$ requires $O(n)$ space and time.

We consider different representations of $K_{\mathbf{U},\mathbf{U}}$ (Eq. 5) to reduce the memory and runtime requirements for performing the multiplication $K_{\mathbf{U},\mathbf{U}}\mathbf{z}$ in the following sections.

5.2.1 SUM: sum representation

In SUM, we represent $K_{\mathbf{U},\mathbf{U}}$ with a Q-length list. At each index q, B_q is a dense matrix of order D and K_q is a Toeplitz matrix of order m, represented using only the top row. In turn, multiplication $K_{\mathbf{U},\mathbf{U}}\mathbf{z}$ is performed by multiplying each matrix in the list with $\mathbf{z}$ and summing the results. The Kronecker MVM $(B_q \otimes K_q)\mathbf{z}$ may be expressed as D fast Toeplitz MVMs with K_q and m dense MVMs with B_q. In turn, assuming $D \ll m$, the runtime for each of the Q terms is $O(Dm \log m)$.

5.2.2 BT: block-Toeplitz representation

In BT, we note that $K_{\mathbf{U},\mathbf{U}}$ is a block matrix with blocks T_{ij}:

$$\sum_q B_q \otimes K_q = (T_{ij})_{i,j\in[D]^2}, \quad T_{ij} = \sum_q b_{ij}^{(q)} K_q.$$

On a one-dimensional grid U, these matrices are Toeplitz because they are linear combinations of Toeplitz matrices. BT requires D^2 m-sized rows to represent each T_{ij}. Then, using the usual block matrix multiplication, an MVM $K_{\mathbf{U},\mathbf{U}}\mathbf{z}$ takes $O(D^2 m \log m)$ time.

On a grid of inputs with $\mathbf{X} = \mathbf{U}$, the SKI interpolation becomes $W = I$. In this case, using BT alone leads to a faster algorithm—applying the Chan block-Toeplitz preconditioner in reduces the number of MVMs necessary to iteratively find an inverse [19].

5.2.3 SLFM: SLFM representation

For the rank-based SLFM representation, let $R \triangleq \sum_q {}^{R_q}/_Q$ be the average rank, $R \leq D$, and re-write the kernel:

$$K_{\mathbf{U},\mathbf{U}} = \sum_q \sum_{r=1}^{R_q} \mathbf{a}_q^{(r)} \mathbf{a}_q^{(r)\top} \otimes K_q + \sum_q \operatorname{diag} \boldsymbol{\kappa}_q \otimes K_q.$$

Note $\mathbf{a}_q^{(r)} \mathbf{a}_q^{(r)\top}$ is rank 1. Under some re-indexing $q' \in [RQ]$, which flattens the double summation such that each q' corresponds to a unique (r, q), the term $\sum_q \sum_{r=1}^{R_q} \mathbf{a}_q^{(r)} \mathbf{a}_q^{(r)\top} \otimes K_q$ may be rewritten as

$$\sum_{q'} \mathbf{a}_{q'} \mathbf{a}_{q'}^\top \otimes K_{q'} = \mathbf{A} \operatorname{blockdiag}_{q'} \left(K_{q'}\right) \mathbf{A}^\top,$$

where $\mathbf{A} = \left(\mathbf{a}_{q'}\right)_{q'} \otimes I_m$ with $\left(\mathbf{a}_{q'}\right)_{q'}$ a matrix of horizontally stacked column vectors [10]. Next, we rearrange the remaining term $\sum_q \operatorname{diag} \boldsymbol{\kappa}_q \otimes K_q$ as $\operatorname{blockdiag}_d(T_d)$, where $T_d = \sum_q \kappa_{qd} K_q$ is Toeplitz. Thus, SLFM represents $K_{\mathbf{U},\mathbf{U}}$ as the sum of two block diagonal matrices of block order QR and D, where each block is a Toeplitz order m matrix; thus, MVMs run in $O((QR + D)m \log m)$.

Note that BT and SLFM each have a faster run time than the other depending on whether $D^2 > QR$. An algorithm that uses this condition to decide between representations can minimize runtime (Tab. 1). We found that SUM is efficient in practice for $Q = 1$.

Table 1: For both LLGP and COGP, m is a configurable parameter that increases up to n to improve accuracy. Q, R, D, κ_2 depend on the LMC kernel, which has $O(QRD)$ hyperparameters (Eq. 2). The asymptotic performance is given in the table. COGP is only independent of R because it cannot represent models for $R \neq 1$. Computing $\nabla \mathcal{L}$ at $\boldsymbol{\theta}$ requires an up-front cost in addition to the per-hyperparameter cost for each $\theta_j \in \boldsymbol{\theta}$. Multiplicative log terms in κ_2, m are hidden.

METHOD	UP-FRONT COST FOR $\nabla \mathcal{L}$	ADDITIONAL COST PER HYPERPARAMETER
EXACT	n^3	n^2
COGP	Qm^3	nm
LLGP	$\sqrt{\kappa_2}\left(n + \min(QR + D, D^2)m\right)$	$n + Dm$

5.3 GP mean and variance prediction

The predictive mean can be computed in $O(1)$ time by $K_{*,X}\boldsymbol{\alpha} \approx W_{*,\mathbf{U}} K_{\mathbf{U},\mathbf{U}} \boldsymbol{\alpha}$ [2].

The full predictive covariance estimate requires finding a new term $K_{*,X} K_{X,X}^{-1} K_{X,*}$. This is done by solving the linear system in a matrix-free manner on-the-fly; in particular, $K_{X,X}^{-1} K_{X,*}$ is computed via MINRES for every new test point $K_{X,*}$. Over several test points, this process is parallelizable.

6 Results

We evaluate the methods on held out data by using standardized mean square error (SMSE) of the test points with the predicted mean, and the negative log predictive density (NLPD) of the Gaussian likelihood of the inferred model. NLPD takes confidence into account, while SMSE only evaluates the mean prediction. In both cases, lower values represent better performance. We evaluated the

performance of our flexible representations of the kernel, SUM, BT, and SLFM, by computing exact gradients using the standard Cholesky algorithm for a grid of different kernels (see Supplement).[1]

6.1 Foreign exchange rate prediction (FX2007)

We replicate the medium-sized dataset from COGP as an application to evaluate LLGP performance. The dataset includes ten foreign currency exchange rates—CAD, EUR, JPY, GBP, CHF, AUD, HKD, NZD, KRW, and MXN—and three precious metals—XAU, XAG, and XPT—implying that $D = 13$.[2] In LLGP, we set $Q = 1$, $R = 2$, as recommended for LMC models on this dataset [5]. COGP roughly corresponds to the the SLFM model, which has a total of 94 hyperparameters, compared to 53 for LLGP. All kernels are squared exponential. The data used in this example are from 2007, and include $n = 3054$ training points and 150 test points. The test points include 50 contiguous points extracted from each of the CAD, JPY, and AUD exchanges. For this application, LLGP uses $m = n/D = 238$ interpolating points. We used the COGP settings from the paper.[3] LLGP outperforms COGP in terms of predictive mean and variance estimation as well as runtime (Tab. 2).

Table 2: Average predictive performance and training time over 10 runs for LLGP and COGP on the FX2007 dataset. Parenthesized values are standard error. LLGP was run with LMC set to $Q = 1$, $R = 2$, and 238 interpolating points. COGP used a $Q = 2$ kernel with 100 inducing points.

METRIC	LLGP	COGP
SECONDS	**64** (**8**)	296 (2)
SMSE	**0.21** (**0.01**)	0.26 (0.03)
NLPD	**-3.62** (**0.07**)	14.52 (3.10)

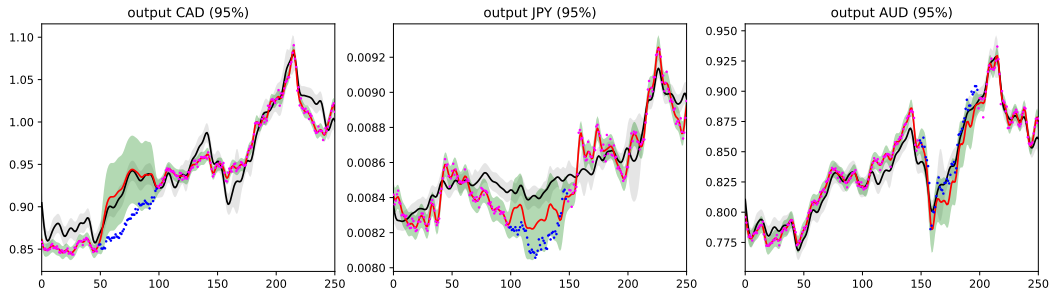


Figure 2: Test outputs for the FX2007 dataset. COGP mean is black, with 95% confidence intervals shaded in grey. LLGP mean is a solid red curve, with light green 95% confidence intervals. Magenta points are in the training set, while blue ones are in the test set. Notice LLGP variance corresponds to an appropriate level of uncertainty on the test set and certainty on the training set, as opposed to the uniform variance from COGP.

6.2 Weather dataset

Next, we replicate results from a weather dataset, a large time series used to validate COGP. Here, $D = 4$ weather sensors Bramblemet, Sotonmet, Cambermet, and Chimet record air temperature over five days in five minute intervals, with some dropped records due to equipment failure. Parts of Cambernet and Chimet are dropped for imputation, yielding $n = 15789$ training measurements and 374 test measurements.

[1] Hyperparameters for the stopping condition, code, and benchmarking scripts are available in <anonymous repository>.

[2] Data are from `http://fx.sauder.ubc.ca/data.html`

[3] COGP hyperparameters for FX2007 were 100 inducing points, 500 iterations, 200 mini-batch size.

We use the default COGP parameters[4] LLGP was run with the same parameters as in FX2007, simulating the SLFM model. We tested LLGP models on $m = \{500, 1000\}$ interpolating points.

Table 3: Average predictive performance and training time over 10 runs of LLGP and COGP on the weather dataset. Parenthesized values are standard error. Both LLGP and COGP trained the SLFM model. We show LLGP with 500 and 1000 interpolating points and COGP with 200 inducing points.

METRIC	LLGP $m = 500$	LLGP $m = 1000$	COGP
SECONDS	**60** (**14**)	259 (62)	1380 (12)
SMSE	0.09 (0.01)	0.09 (0.01)	**0.08** (**0.00**)
NLPD	2.14 (0.58)	**1.54** (**0.03**)	98.48 (1.30)

LLGP performed slightly worse than COGP in SMSE, but both NLPD and runtime indicate significant improvements (Tab. 3). Varying the number of interpolating points m from 500 to 1000 demonstrates the runtime versus NLPD tradeoff (Fig. 3). While NLPD improvement diminishes as m increases, LLGP still improves upon COGP for a wide range of m by an order of magnitude in runtime and almost two orders of magnitude in NLPD.

7 Conclusion

In this paper, we present LLGP, which we show adapts and accelerates SKI [11] for the problem of multi-output GP regression. LLGP exploits structure unique to LMC kernels, enabling a parsimonious representation of the covariance matrix, and gradient computations in $\tilde{O}\left(\sqrt{\kappa_2}(m+n)\right)$.

LLGP provides an efficient means to approximate the log-likelihood gradients using interpolation. We have shown on several datasets that this can be done in a way that is faster and leads to more accurate results than variational approximations. Because LLGP scales well with increases in m, capturing complex interactions in the covariance with an accurate interpolation is cheap, as demonstrated by performance on both the FX2007 and weather datasets (Fig. 2, Tab. 2, Tab. 3).

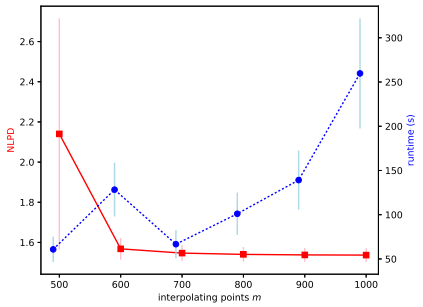


Figure 3: Average and standard error of NLPD and runtime of the SLFM model on LLGP across over varying interpolating points. Every setting was run 10 times.

Future work would extend the inputs to accept multiple dimensions. This can be done without losing internal structure in the kernel [2]: Toeplitz covariance matrices become block-Toeplitz with Toeplitz-blocks (BTTB). The cubic interpolation W requires a number of terms exponential in the dimension, so projection into lower dimensions estimated in a supervised manner would be essential. Another useful line for investigation would be more informed stopping heuristics. Finally, an extension to non-Gaussian noise is also feasible in a matrix-free manner by following prior work [16].

References

[1] Mauricio Alvarez, Lorenzo Rosasco, Neil Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

[2] Andrew Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable Gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.

[4] `https://github.com/trungngv/cogp`, commit `3b07f621ff11838e89700cfb58d26ca39b119a35`. The weather dataset was run on 1500 iterations, mini-batch size 1000, 200 interpolating points.

[3] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, pages 514–520, 1996.

[4] Michael Osborne, Stephen Roberts, Alex Rogers, Sarvapali Ramchurn, and Nicholas Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *7th international conference on Information processing in sensor networks*, pages 109–120. IEEE Computer Society, 2008.

[5] Mauricio Alvarez, David Luengo, Michalis Titsias, and Neil D Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In *AISTATS*, volume 9, pages 25–32, 2010.

[6] Trung Nguyen, Edwin Bonilla, et al. Collaborative multi-output Gaussian processes. In *UAI*, pages 643–652, 2014.

[7] Elad Gilboa, Yunus Saatçi, and John Cunningham. Scaling multidimensional inference for structured Gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):424–436, 2015.

[8] John Cunningham, Krishna Shenoy, and Maneesh Sahani. Fast Gaussian process methods for point process intensity estimation. In *25th international conference on Machine learning*, pages 192–199. ACM, 2008.

[9] Joaquin Quiñonero-Candela and Carl Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[10] Matthias Seeger, Yee-Whye Teh, and Michael Jordan. Semiparametric latent factor models. In *Eighth Conference on Artificial Intelligence and Statistics*, 2005.

[11] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (kiss-gp). In *The 32nd International Conference on Machine Learning*, pages 1775–1784, 2015.

[12] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981.

[13] Andrew Wilson, Elad Gilboa, John Cunningham, and Arye Nehorai. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*, pages 3626–3634, 2014.

[14] Matthew Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[15] Mark Gibbs and David MacKay. Efficient implementation of Gaussian processes, 1996.

[16] Kurt Cutajar, Michael Osborne, John Cunningham, and Maurizio Filippone. Preconditioning kernel matrices. In *ICML*, pages 2529–2538, 2016.

[17] David Fong and Michael Saunders. CG versus MINRES: an empirical comparison. *SQU Journal for Science*, 17(1):44–62, 2012.

[18] Vikas Raykar and Ramani Duraiswami. Fast large scale Gaussian process regression using approximate matrix-vector products. In *Learning workshop*, 2007.

[19] Tony Chan and Julia Olkin. Circulant preconditioners for toeplitz-block matrices. *Numerical Algorithms*, 6(1):89–101, 1994.