A Global-Local Approximation Framework for Large-Scale Gaussian Process Modeling

Akhil Vakayil and V. Roshan Joseph

H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract

In this work, we propose a novel framework for large-scale Gaussian process (GP) modeling. Contrary to the global, and local approximations proposed in the literature to address the computational bottleneck with exact GP modeling, we employ a combined global-local approach in building the approximation. Our framework uses a subset-of-data approach where the subset is a union of a set of global points designed to capture the global trend in the data, and a set of local points specific to a given testing location to capture the local trend around the testing location. The correlation function is also modeled as a combination of a global, and a local kernel. The performance of our framework, which we refer to as `TwinGP`, is on par or better than the state-of-the-art GP modeling methods at a fraction of their computational cost.

Keywords: Big Data, Inducing points, Kriging, Nonparametric Regression, Twinning.

1 Introduction

Gaussian process (GP) is a widely used Bayesian framework for nonparametric regression (Rasmussen and Williams, 2005), and emulating computer models (Santner et al., 2003; Gramacy, 2020). The objective is to approximate a latent function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d$, for which a functional Gaussian prior is assumed, and the posterior is obtained given the observed training data. The posterior mean is treated as the point prediction at $\mathbf{x}$, and the posterior variance gives the associated prediction uncertainty. Regardless of being the best linear unbiased predictor under the assumed model, a major drawback with GP modeling that limits its applicability to Big Data is its $\mathcal{O}(n^3)$ computational complexity, where n is the number of observations in the training data. This is because GP modeling requires the inversion of an $n \times n$ kernel (or correlation) matrix $\boldsymbol{\mathcal{R}}_{nn}$ involved in posterior estimation. Building GP approximations that scale reasonably well with large n is an active area of research, and the various methodologies to do so can be roughly grouped into two categories: (i) global, and (ii) local approximations.

Global approximations generally (i) focus on a subset of the training data with $m \ll n$ observations resulting in a much smaller $\boldsymbol{\mathcal{R}}_{mm}$ to invert (Chalupka et al., 2013), where the subset can be randomly selected, based on clustering, or with active learning (Lawrence et al., 2002; Keerthi and Chu, 2005), (ii) use a compactly supported kernel function (Gneiting, 2002) to generate sparse $\boldsymbol{\mathcal{R}}_{nn}$, then exploit the sparsity to efficiently compute $\boldsymbol{\mathcal{R}}_{nn}^{-1}$ (Kaufman et al., 2011), or (iii)

approximate $\boldsymbol{\mathcal{R}}_{nn}$ with a low m-rank matrix plus diagonal using $m \ll n$ inducing points, that can be inverted in $\mathcal{O}(m^2 n)$ (Quiñonero-Candela and Rasmussen, 2005). Local approximations essentially (i) build a local GP model for each testing location (Gramacy and Apley, 2015), or (ii) partition the training data into disjoint blocks and build independent GP for each block resulting in a block diagonal $\boldsymbol{\mathcal{R}}_{nn}$ that can be inverted efficiently (Das and Srivastava, 2010).

Global or local approximations alone can be insufficient depending upon the data. To model a complex latent function $f(\mathbf{x})$ that varies significantly locally, a larger m would be needed to adequately summarize the data, be it subset of data approach or inducing points. Local approximations where a local GP is fit in the neighborhood of the testing location ignore the global trend and often result in over-confident predictions due to local over-fitting. On the other hand, local approximations where independent GPs are fit on partitioned training data suffer from discontinuity at the boundaries of the local regions. Park and Huang (2016) show that greater the discontinuity lower the prediction accuracy, especially at the boundaries.

Snelson and Ghahramani (2007) build on the inducing points framework presented in Quiñonero-Candela and Rasmussen (2005) to incorporate local trend. To begin with, the inducing points framework makes the assumption that given the inducing points $\mathcal{I}$, the training and testing conditional distributions are independent, i.e, for any training location $\mathbf{x}$ and testing location $\mathbf{x}^*$, $p(f(\mathbf{x}), f(\mathbf{x}^*)|\mathcal{I}) = p(f(\mathbf{x})|\mathcal{I})p(f(\mathbf{x}^*)|\mathcal{I})$. The fully independent conditional approximation (`FIC`) given in Snelson and Ghahramani (2005) makes additional assumption that given $\mathcal{I}$, the conditional distribution of the latent function at any two locations (training and/or testing) $\mathbf{x}_i, \mathbf{x}_j$ are independent, i.e., $p(f(\mathbf{x}_i), f(\mathbf{x}_j)|\mathcal{I}) = p(f(\mathbf{x}_i)|\mathcal{I})p(f(\mathbf{x}_j)|\mathcal{I})$. Snelson and Ghahramani (2007) later present partially independent conditional approximation (`PIC`) where the space is partitioned into disjoint blocks and given $\mathcal{I}$, the conditional distribution at any two locations are independent only when they belong to separate blocks. The dependence within a block in `PIC` attempts to capture the local trend, with the limitation that at boundaries of a block the dependence from locations in neighboring blocks are ignored.

The discontinuity problem alluded to before with local approximations that build independent GPs on partitioned training data is generally addressed by using some form of weighted averaging of the independent GPs (Tresp, 2000; Rasmussen and Ghahramani, 2001; Gramacy and Lee, 2008; Chen and Ren, 2009; Deisenroth and Ng, 2015). Park and Apley (2018) present a patching of the independent GPs (`PK`) where they augment the training data with a set of pseudo observations located at the boundaries of neighboring regions, and impose continuity by enforcing two neighboring GPs to make identical predictions at the pseudo locations common to them. Though the discontinuity

problem can be accounted for, the global trend remains elusive for local approximations.

In this work we propose a novel methodology to capture both global and local trend in GP modeling. We first select a set of observations from the training data, independent of the testing location, to capture the global trend. The global point set is then supplemented with observations from the neighborhood of a given testing location to capture the local trend. The global trend is modeled using a power exponential kernel whose hyperparameters are learned based on the global points alone, and the local trend is modeled using a compactly supported kernel with a single hyperparameter that is predetermined based on the global points. Both kernels act on the combined set of global and local points of size $m \ll n$, resulting in an additive kernel. Vanhatalo and Vehtari (2008) also use an additive kernel where the global trend is modeled with `FIC` and local trend with a compactly supported kernel, however, contrary to our approach, their compactly supported kernel acts on the full training data and requires the training data to have lattice structure to efficiently invert the kernel matrix by exploiting its sparsity. Our framework does not impose any restrictions on the training data, and scales well in higher dimensions as we do not rely on the sparsity of the compactly supported kernel matrix for efficient inversion. Furthermore, unlike `FIC` and `PIC`, we do not make any assumptions on the conditional distributions.

We refer to our methodology as `TwinGP` owing to the twin set of training points and kernels involved. The remainder of the article is organized as follows. Section 2 provides a brief review of GP and introduces notation. Section 3 presents the new `TwinGP` framework. Section 4 gives an illustration of `TwinGP` using a $1d$ function, and in Section 5 we compare `TwinGP` with popular global, and local GP approximations for emulation as well as modeling real world datasets. Finally, in Section 6 we provide our concluding remarks.

2 Gaussian Process Review

Denote the training data as $\{\mathbf{X}_n, \mathbf{Y}_n\} \coloneqq \{\mathbf{x}_i, y_i\}_{i=1}^n$ such that the inputs $\mathbf{x}_i \in \mathbb{R}^d$ and output $y_i \in \mathbb{R}$, for all i. Let

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \text{ for } i = 1, \ldots, n, \tag{1}$$

where $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \nu^2)$ is the random noise in the output. Our aim is to estimate the latent function $f(\cdot)$ from the training data. In GP modeling, we assume that $f(\cdot)$ to be a realization from a Gaussian process:

$$f(\mathbf{x}) \sim \text{GP}(\mu, \tau^2 \mathcal{R}(\mathbf{x}, \cdot)), \tag{2}$$

where μ τ^2, and $\mathcal{R}(\cdot,\cdot)$ are the mean, variance, and correlation function of the GP, respectively. The correlation function is defined as $Cor\{f(\mathbf{u}), f(\mathbf{v})\} = \mathcal{R}(\mathbf{u},\mathbf{v})$, which is a positive definite function with $\mathcal{R}(\mathbf{u},\mathbf{u}) = 1$. We use the names correlation function and kernel function interchangeably in this paper. Please refer to the books Santner et al. (2003) and Gramacy (2020) for details on GP modeling.

From (1) and (2), we have $\mathbf{Y}_n \sim \mathcal{N}_n(\mu\mathbb{1}_n, \tau^2\boldsymbol{\mathcal{R}}_{nn} + \nu^2\mathbf{I}_n)$, where $\boldsymbol{\mathcal{R}}_{nn}$ is the $n \times n$ correlation matrix whose ij^{th} element is $\mathcal{R}(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbb{1}_n \coloneqq [1,\ldots,1]'$, and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Let $\eta = \nu^2/\tau^2$, known as nugget. Given an arbitrary testing location $\mathbf{x}^* \in \mathbb{R}^d$, we are interested in finding the conditional distribution of $f(\mathbf{x}^*)|\mathbf{Y}_n$.

Define the $1\times n$ correlation vector as $\boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_n) \coloneqq [\mathcal{R}(\mathbf{x}^*, \mathbf{x}_1), \ldots, \mathcal{R}(\mathbf{x}^*, \mathbf{x}_n)]$. In keeping with the notation, we have $\boldsymbol{\mathcal{R}}(\mathbf{X}_n, \mathbf{x}^*) = \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_n)'$ and $\boldsymbol{\mathcal{R}}_{nn} = \boldsymbol{\mathcal{R}}(\mathbf{X}_n, \mathbf{X}_n) \coloneqq [\mathbf{R}(\mathbf{x}_1, \mathbf{X}_n); \ldots; \boldsymbol{\mathcal{R}}(\mathbf{x}_n, \mathbf{X}_n)]$. The joint distribution of $f(\mathbf{x}^*)$ and $\mathbf{f}(\mathbf{X}_n) \coloneqq [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]'$ is given by

$$\begin{bmatrix} f(\mathbf{x}^*) \\ \mathbf{f}(\mathbf{X}_n) \end{bmatrix} \sim \mathcal{N}_{n+1}\left(\begin{bmatrix} \mu \\ \mu\mathbb{1}_n \end{bmatrix}, \ \tau^2 \begin{bmatrix} 1 & \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_n) \\ \boldsymbol{\mathcal{R}}(\mathbf{X}_n, \mathbf{x}^*) & \boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n \end{bmatrix} \right).$$

Then, the conditional distribution of $f(\mathbf{x}^*)|\mathbf{Y}_n$, is given by

$$f(\mathbf{x}^*)|\mathbf{Y}_n \sim \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)),$$

where

$$\mu(\mathbf{x}^*) = \mu + \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_n)[\boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n]^{-1}(\mathbf{Y}_n - \mu\mathbb{1}_n), \tag{3}$$

$$\sigma^2(\mathbf{x}^*) = \tau^2\Big\{1 - \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_n)[\boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n]^{-1}\boldsymbol{\mathcal{R}}(\mathbf{X}_n, \mathbf{x}^*)\Big\}. \tag{4}$$

There exists a multitude of correlation functions in the literature (Rasmussen and Williams, 2005, Ch.4), each with their own set of correlation parameters $\boldsymbol{\theta}$ that are estimated from the training data. The empirical Bayes estimate of μ, τ^2, η, and $\boldsymbol{\theta}$ are given as follows (Santner et al., 2003):

$$\hat{\mu} = \frac{\mathbb{1}_n'[\boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n]^{-1}\mathbf{Y}_n}{\mathbb{1}_n'\,[\boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n]^{-1}\,\mathbb{1}_n}, \tag{5}$$

$$\hat{\tau}^2 = \frac{1}{n}(\mathbf{Y}_n - \hat{\mu}\mathbb{1}_n)'[\boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n]^{-1}(\mathbf{Y}_n - \hat{\mu}\mathbb{1}_n), \tag{6}$$

$$\hat{\boldsymbol{\theta}}, \hat{\eta} = \underset{\boldsymbol{\theta},\ \eta\ >\ 0}{\arg\min}\ \ n\log\hat{\tau}^2 + \log|\boldsymbol{\mathcal{R}}_{nn} + \eta\mathbf{I}_n|. \tag{7}$$

As evident from the above equations, much of the complexity in GP modeling stems from the

computation of $[\boldsymbol{\mathcal{R}}_{nn} + \eta \mathbf{I}_n]^{-1}$ and $|\boldsymbol{\mathcal{R}}_{nn} + \eta \mathbf{I}_n|$, both requiring $\mathcal{O}(n^3)$ operations. In the next section, we propose our methodology to address this computational bottleneck for large n.

3 Global-Local Gaussian Process

In order to circumvent the $\mathcal{O}(n^3)$ computational complexity, we build the GP using only $m \ll n$ points form the training data. Contrary to the global approximation methodologies in the literature where the m points are chosen from the training data or artificially generated, so as to summarize the whole data, we use a combination of g global points and l local points such that $m = g + l$. The g global points are independent of the testing location, while the l local points are specific to the testing location $\mathbf{x}^*$.

To capture the global trend, and local trend with respect to $\mathbf{x}^*$, the kernel function $\mathcal{R}(\cdot, \cdot)$ is also modeled as a combination of two kernel functions (Ba and Joseph, 2012; Harari and Steinberg, 2014), i.e.,

$$\mathcal{R}(\mathbf{x}_a, \mathbf{x}_b) = (1 - \lambda)\mathcal{G}(\mathbf{x}_a, \mathbf{x}_b) + \lambda \mathcal{L}(\mathbf{x}_a, \mathbf{x}_b), \ \lambda \in [0, 1]. \tag{8}$$

$\mathcal{G}$ is designed to capture the global trend in the data, while $\mathcal{L}$ captures the local trend around $\mathbf{x}^*$. The hyperparameter λ controls the proportion of global and local trends used in building $\mathcal{R}$.

Let $\mathbf{X}_m \subset \mathbf{X}_n$ represent the m training points, then $\boldsymbol{\mathcal{G}}_{mm} = \mathcal{G}(\mathbf{X}_m, \mathbf{X}_m)$, and $\boldsymbol{\mathcal{L}}_{mm} = \mathcal{L}(\mathbf{X}_m, \mathbf{X}_m)$. With this setup, instead of inverting $\boldsymbol{\mathcal{R}}_{nn} + \eta \mathbf{I}_n$ in (3)-(7), we need only invert $\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m$ where

$$\boldsymbol{\mathcal{R}}_{mm} = (1 - \lambda)\boldsymbol{\mathcal{G}}_{mm} + \lambda \boldsymbol{\mathcal{L}}_{mm}. \tag{9}$$

Thus, the computational complexity of fitting the GP reduces to $\mathcal{O}(m^3)$. In the subsections that follow, we present how to efficiently locate the m training points, the final predictive equations, and how the kernel function $\mathcal{R}(\cdot, \cdot)$ in (8) is determined.

3.1 Global and Local Points

Denote the global training points as $\mathbf{X}_g$, and the local training points as $\mathbf{X}_l$, we have $\mathbf{X}_m = \mathbf{X}_g \cup \mathbf{X}_l$. Ideally, for a given testing location $\mathbf{x}^*$, we should be denoting $\mathbf{X}_m$ as $\mathbf{X}_m(\mathbf{x}^*) = \mathbf{X}_g \cup \mathbf{X}_l(\mathbf{x}^*)$, a technicality we omit for ease of notation.

The objective we desire to achieve with $\mathbf{X}_g$ is to sufficiently model the global trend in the

training data. Mak and Joseph (2018) proposed a nonparametric and model-free data reduction technique known as support points that can produce a small set of points to represent the full dataset. Support points are obtained by minimizing the energy distance (Székely and Rizzo, 2013) between its empirical distribution and that of the dataset using difference-of-convex programming. However, support points need not be a subset of the full dataset. Given the support points, Joseph and Vakayil (2022) use sequential nearest neighbor assignment to sample from the dataset. Even still, computing support points as given in Mak and Joseph (2018) is $\mathcal{O}(n^2)$. Later, Vakayil and Joseph (2022) present an efficient sampling algorithm named `Twinning` that minimizes the energy distance in $\mathcal{O}(n \log n)$. Since we are dealing with large datasets in this work, we will use `Twinning` to find $\mathbf{X}_g$.

Given the testing location $\mathbf{x}^*$, $\mathbf{X}_l$ is obtained as the l nearest neighbors to $\mathbf{x}^*$ in $\mathbf{X}_n \setminus \mathbf{X}_g$, a task that can be performed efficiently with kd-tree in $\mathcal{O}(l \log l)$. Figure 1 gives a depiction of the global and local training points identified for a given testing location, for a $1d$ example. One can observe from the figure that fitting a GP on the local points alone, shown as green diamonds, will clearly underpredict at the testing location, and it is shown to be not optimal (Emery, 2009). In `laGP` proposed by Gramacy and Apley (2015), the nearest neighbors are supplemented with active learning, however, an optimization is required at each testing location that adds to the computational complexity. On the other hand, our method makes use of the predetermined global points (blue circles) to capture the global trend, and therefore, nearest neighbors alone as the local points will suffice. This reduces the computational burden in our framework.

3.2 Predictive equations

At this stage, given $\mathbf{x}^*$, we have identified $\mathbf{X}_m$. Assume that the kernel function $\mathcal{R}$ is fully determined. Let $\mathbf{Y}_m$ be the response vector corresponding to $\mathbf{X}_m$, i.e., $\mathbf{Y}_m = [\mathbf{Y}_g; \mathbf{Y}_l]$ where $\mathbf{Y}_g$ and $\mathbf{Y}_l$ are the response vectors corresponding to $\mathbf{X}_g$ and $\mathbf{X}_l$, respectively. The conditional distribution of $f(\mathbf{x}^*)|\mathbf{Y}_m$ is given by $\mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*))$, where

$$\mu(\mathbf{x}^*) = \hat{\mu}_m + \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_m)[\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1}(\mathbf{Y}_m - \mu \mathbb{1}_m), \tag{10}$$

$$\sigma^2(\mathbf{x}^*) = \hat{\tau}_m^2 \Big\{ 1 - \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_m)[\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1} \boldsymbol{\mathcal{R}}(\mathbf{X}_m, \mathbf{x}^*) \Big\}, \tag{11}$$

$$\hat{\mu}_m = \frac{\mathbb{1}_m' [\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1} \mathbf{Y}_m}{\mathbb{1}_m' [\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1} \mathbb{1}_m}, \tag{12}$$

$$\hat{\tau}_m^2 = \frac{1}{m} (\mathbf{Y}_m - \hat{\mu}_m \mathbb{1}_m)' [\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1} (\mathbf{Y}_m - \hat{\mu}_m \mathbb{1}_m). \tag{13}$$

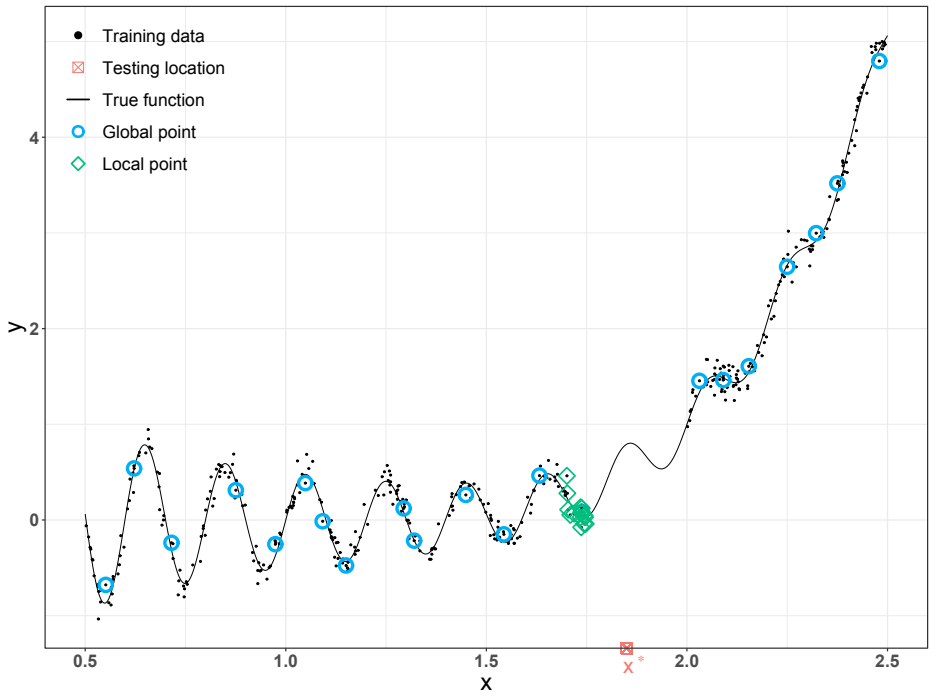

Figure 1: *Illustration of the training points (global and local) identified for a given training data and testing location with* `TwinGP`.

For the prediction of a noisy observation, we have $y^*|\mathbf{Y}_m \sim \mathcal{N}(\mu(\mathbf{x}^*), \sigma_\eta^2(\mathbf{x}^*))$, where $\mu(\mathbf{x}^*)$ is the same as given in (10), but $\sigma_\eta^2(\mathbf{x}^*)$ changes to

$$\sigma_\eta^2(\mathbf{x}^*) = \hat{\tau}_m^2 \Big\{ 1 + \eta - \boldsymbol{\mathcal{R}}(\mathbf{x}^*, \mathbf{X}_m)[\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1} \boldsymbol{\mathcal{R}}(\mathbf{X}_m, \mathbf{x}^*) \Big\}. \tag{14}$$

Furthermore, $\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m$ can be deconstructed as follows,

$$\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m = \begin{bmatrix} \boldsymbol{\mathcal{R}}_{gg} + \eta \mathbf{I}_g & \boldsymbol{\mathcal{R}}_{gl} \\ \boldsymbol{\mathcal{R}}_{lg} & \boldsymbol{\mathcal{R}}_{ll} + \eta \mathbf{I}_l \end{bmatrix} \tag{15}$$

$$= \begin{bmatrix} (1-\lambda)\boldsymbol{\mathcal{G}}_{gg} + \lambda \boldsymbol{\mathcal{L}}_{gg} + \eta \mathbf{I}_g & (1-\lambda)\boldsymbol{\mathcal{G}}_{gl} + \lambda \boldsymbol{\mathcal{L}}_{gl} \\ (1-\lambda)\boldsymbol{\mathcal{G}}_{lg} + \lambda \boldsymbol{\mathcal{L}}_{lg} & (1-\lambda)\boldsymbol{\mathcal{G}}_{ll} + \lambda \boldsymbol{\mathcal{L}}_{ll} + \eta \mathbf{I}_l \end{bmatrix}. \tag{16}$$

Since $\boldsymbol{\mathcal{R}}_{gg} = (1-\lambda)\boldsymbol{\mathcal{G}}_{gg} + \lambda \boldsymbol{\mathcal{L}}_{gg}$ is independent of the testing location $\mathbf{x}^*$, $\boldsymbol{\mathcal{R}}_{gg}$ and $[\boldsymbol{\mathcal{R}}_{gg} + \eta \mathbf{I}_g]^{-1}$ can be predetermined before testing. For a given $\mathbf{x}^*$, the computational effort in inverting $\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m$

can be significantly reduced by using block matrix inversion as follows,

$$[\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{gg}^{-1} + \boldsymbol{\Sigma}_{gg}^{-1}\boldsymbol{\mathcal{R}}_{gl}\mathbf{S}^{-1}\boldsymbol{\mathcal{R}}_{lg}\boldsymbol{\Sigma}_{gg}^{-1} & -\boldsymbol{\Sigma}_{gg}^{-1}\boldsymbol{\mathcal{R}}_{gl}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\boldsymbol{\mathcal{R}}_{lg}\boldsymbol{\Sigma}_{gg}^{-1} & \mathbf{S}^{-1} \end{bmatrix}, \tag{17}$$

where $\boldsymbol{\Sigma}_{gg} = \boldsymbol{\mathcal{R}}_{gg} + \eta \mathbf{I}_g$, and $\mathbf{S} = \boldsymbol{\mathcal{R}}_{ll} - \boldsymbol{\mathcal{R}}_{lg}\boldsymbol{\Sigma}_{gg}^{-1}\boldsymbol{\mathcal{R}}_{gl}$, thereby, per testing location we need only invert $\mathbf{S}$, a small $l \times l$ matrix.

3.3 Correlation Functions

To model the global trend, we use the popular power exponential kernel function (Sacks et al., 1989),

$$\mathcal{G}(\mathbf{x}_a, \mathbf{x}_b) = \exp\left(-\sum_{i=1}^{d} \frac{|\mathbf{x}_a^i - \mathbf{x}_b^i|^\alpha}{\theta_g^i}\right), \ \alpha \in (0, 2], \theta_g^i > 0. \tag{18}$$

Denote the hyperparameters of $\mathcal{G}$ as $\boldsymbol{\theta}_g$, we have $\boldsymbol{\theta}_g = \{\theta_g^1, \ldots, \theta_g^d, \alpha\}$. They can be estimated with respect to $\mathbf{X}_g$ alone, i.e., independent of the testing location,

$$\hat{\mu}_g = \frac{\mathbb{1}_g'[\boldsymbol{\mathcal{G}}_{gg} + \eta_g \mathbf{I}_g]^{-1}\mathbf{Y}_g}{\mathbb{1}_g'[\boldsymbol{\mathcal{G}}_{gg} + \eta_g \mathbf{I}_g]^{-1}\mathbb{1}_g}, \tag{19}$$

$$\hat{\tau}_g^2 = \frac{1}{g}(\mathbf{Y}_g - \hat{\mu}_g \mathbb{1}_g)'[\boldsymbol{\mathcal{G}}_{gg} + \eta_g \mathbf{I}_g]^{-1}(\mathbf{Y}_g - \hat{\mu}_g \mathbb{1}_g), \tag{20}$$

$$\hat{\boldsymbol{\theta}}_g = \arg\min_{\boldsymbol{\theta}_g} \ g \log \hat{\tau}_g^2 + \log|\boldsymbol{\mathcal{G}}_{gg} + \eta_g \mathbf{I}_g|. \tag{21}$$

We have constrained $\alpha \in [1, 2]$ in the optimization assuming the latent function $f(\cdot)$ to be reasonably smooth and not too wiggly.

To model the local trend, we use compactly supported correlation functions (Gneiting, 2002), which ensures identifiability of the local correlation parameters with respect to the global correlation parameters. Specifically, we use Wendlands's compactly supported radial function (Wendland, 2004),

$$\mathcal{L}(\mathbf{x}_a, \mathbf{x}_b) = \left((q+1)\frac{\|\mathbf{x}_a - \mathbf{x}_b\|_2}{\theta_l} + 1\right) \max\left\{0, \left(1 - \frac{\|\mathbf{x}_a - \mathbf{x}_b\|_2}{\theta_l}\right)\right\}^{q+1}, \ q = \lfloor \frac{d}{2} \rfloor + 2 \tag{22}$$

where $\|\cdot\|_2$ is the ℓ_2 norm and $\lfloor u \rfloor$ is the largest integer lesser than or equal to u. $\mathcal{L}$ has a single hyperparameter θ_l which we set as the covering radius (Fasshauer, 2007, p. 22) for $\mathbf{X}_g$ to cover $\mathbf{X}_n$,

i.e.,

$$\hat{\theta}_l = \min \{\rho : \mathbf{X}_n \subseteq \cup_{i=1}^{g} \mathcal{B}_\rho(\mathbf{x}_i),\ \mathbf{x}_i \in \mathbf{X}_g,\ \forall i\}, \tag{23}$$

where $\mathcal{B}_\rho(\mathbf{x})$ is a closed ball of radius ρ centered at $\mathbf{x}$. Setting θ_l as in (23) makes it independent of the testing location, thereby, allowing us to precompute $\boldsymbol{\mathcal{L}}_{gg}$ in (16) leading to the computational gains with block matrix inversion (17). In addition, almost surely for any testing location $\mathbf{x}^*$ there exists at least one global training point such that the local kernel is active between them, i.e., there exists $\mathbf{x}_g \in \mathbf{X}_g : \mathcal{L}(\mathbf{x}^*, \mathbf{x}_g) > 0$. The motivation here is that we do not wish $\mathcal{L}$ to neglect the correlation between $\mathbf{x}^*$ and $\mathbf{X}_g$.

We need to specify two more parameters: λ and η_l. We could estimate them using empirical Bayes methods as before, but since we have unused data in the training set, we can do the estimation in a different and more robust fashion. We sample $\{\mathbf{X}_v, \mathbf{Y}_v\}$ from $\{\mathbf{X}_n, \mathbf{Y}_n\} \setminus \{\mathbf{X}_g, \mathbf{Y}_g\}$ by `Twinning` to create a validation set. Now, we can estimate λ and η_l by minimizing the prediction error:

$$\begin{aligned} \hat{\lambda},\ \hat{\eta_l} &= \underset{\lambda \in [0,1],\ \eta_l > 0}{\arg\min}\ \text{MSE}(\lambda) \\ &= \underset{\lambda \in [0,1],\ \eta_l > 0}{\arg\min} \sum_{\mathbf{x} \in \mathbf{X}_v} (y_{\mathbf{x}} - \mu(\mathbf{x}))^2, \end{aligned} \tag{24}$$

where $\mu(\mathbf{x})$ is obtained as given in (10) with $\eta = (1-\lambda)\eta_g + \lambda\eta_l$.

3.4 Computational complexity

The `TwinGP` procedure is summarized in Algorithm 1. The computational complexity of `TwinGP` can be deconstructed as follows,

Testing

(i) Identifying $\mathbf{X}_l$ for a given $\mathbf{x}^*$ is on average $\mathcal{O}(\log n)$ using kd-tree.

(ii) The complexity in computing $\mu(\mathbf{x}^*)$ and $\sigma^2(\mathbf{x}^*)$ is dictated by inversion of $\boldsymbol{\mathcal{R}}_{mm}$. With block matrix inversion as given in (17), computing $[\boldsymbol{\mathcal{R}}_{mm} + \eta \mathbf{I}_m]^{-1}$ is $\mathcal{O}(g^2 l)$.

Training

(i) Obtaining a twinning sample to identify $\mathbf{X}_g$ is on average $\mathcal{O}(n \log n)$.

(ii) The complexity in estimating $\boldsymbol{\theta}_g$ is dictated by inversion of $\boldsymbol{\mathcal{G}}_{gg} + \eta_g \mathbf{I}_g$ that is $\mathcal{O}(g^3)$.

(iii) θ_l can be estimated efficiently with a kd-tree in $\mathcal{O}(n \log g)$.

(iv) The complexity in estimating λ and η_l is similar to testing, i.e., $\mathcal{O}(g^2 l)$ per validation point.

Overall complexity

Thus, the overall computational complexity of `TwinGP` is $\mathcal{O}(g^3 + tg^2 l)$, where t is the number of testing locations. If we select g to be order of $\sqrt{n}$, the training complexity reduces to $\mathcal{O}(n^{1.5})$, a substantial improvement over $\mathcal{O}(n^3)$.

Algorithm 1 `TwinGP`

1: Input training set $\{\mathbf{X}_n, \mathbf{Y}_n\}$ and testing locations $\mathbf{X}_t^*$
2: Identify global training points $\mathbf{X}_g$ (Section 3.1)
3: Estimate $\boldsymbol{\theta}_g$ as in (21)
4: Estimate θ_l as in (23)
5: Estimate λ and η_l as in (24)
6: **for** $\mathbf{x}^* \in \mathbf{X}_t^*$ **do**
7: Identify local training points $\mathbf{X}_l$ (Section 3.1)
8: Compute $\mu(\mathbf{x}^*)$ and $\sigma^2(\mathbf{x}^*)$ (Section 3.2)
9: **end for**
10: **return** $\{\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)\}, \forall \mathbf{x}^* \in \mathbf{X}_t^*$

4 $1d$ Illustration

Here we illustrate how `TwinGP` overcomes the shortcomings of purely local or global GP approximations. Consider the following function from Gramacy and Lee (2012),

$$f(x) = \frac{\sin 10\pi x}{2x} + (x-1)^4, \ x \in [0.5, 2.5]. \tag{25}$$

We first generate the training dataset with $n = 500$ observations, where $x_1, \ldots, x_n$ are selected on a uniform grid in $[0.5, 2.5]$, and we add a Gaussian noise to each observation as follows,

$$y_i = f(x_i) + \epsilon_i, \ \epsilon_i \sim \mathcal{N}(0, 0.01), \ \forall i = 1, \ldots, 500. \tag{26}$$

The testing set at $2,000$ locations are also selected on a uniform grid in $[0.5, 2.5]$. Plot (a) in Figure 2 depicts $f(x)$, the 500 training locations, and the prediction from a full GP modeled using the `mlegp` (Dancik and Dorman, 2008) package. As evident, $f(x)$ is recovered extremely well with a full GP, in addition, the confidence intervals are tight.

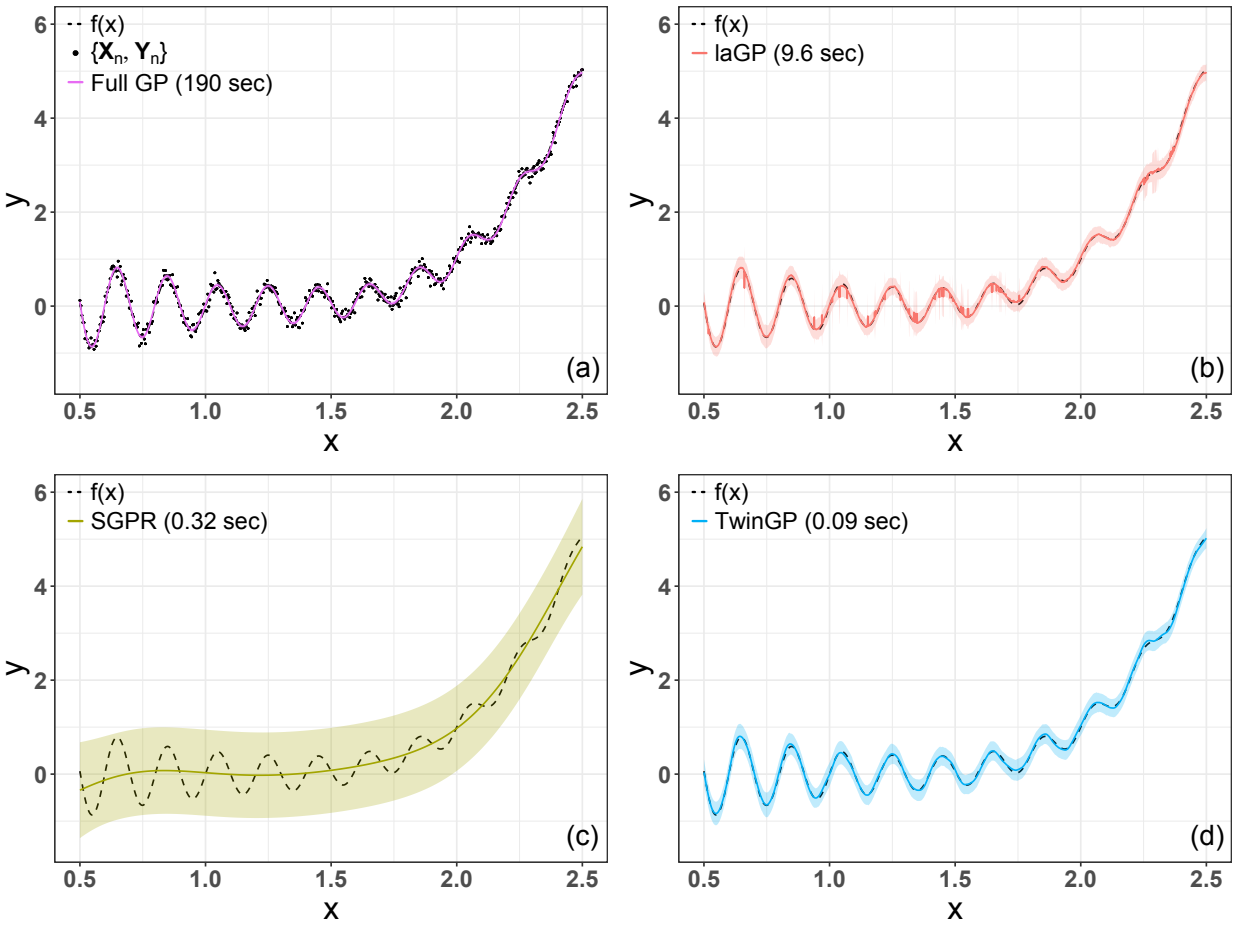

Figure 2: *Prediction and 2σ confidence interval (shaded region) with full GP, `laGP`, `SGPR`, and `TwinGP` to model the 1d function in (25).*

To demonstrate the global GP approximation method, we use `SGPR` (Titsias, 2009) implemented in `GPyTorch` (Gardner et al., 2018). `SGPR` is a low-rank GP approximation where the inducing points and kernel hyperparameters are estimated with a variational learning approach. For local approximation we use `laGP` (Gramacy and Apley, 2015) implemented in the `laGP` (Gramacy, 2016) package. For each testing location, `laGP` starts with a set of training points in its neighborhood and then sequentially adds more training points so as to minimize the predictive variance.

For `TwinGP` we set the number of global points $g = 22$ and local points $l = 25$. For `SGPR` the number of inducing points is set to be $m = g + l$, and for `laGP` $l + 10$ training locations are considered per testing location, i.e., start with l neighborhood points and then sequentially add 10 more points. Plots (b), (c), and (d) in Figure 2 give the predictions and 2σ confidence intervals from `laGP`, `SGPR`, and `TwinGP` respectively. As expected, `laGP` predictions are discontinuous owing to its local nature, while `SGPR` produces smooth prediction neglecting the local oscillations of $f(x)$.

On the other hand, `TwinGP` gives comparable predictions to the full GP. The total computation time for training and testing with the four methods are given in Figure 2. We can clearly see the computational saving with `TwinGP` over the full GP, which will be even more substantial with large datasets as we demonstrate in the next section.

5 Experiments

In this section, we make an extensive analysis of `TwinGP` performance on several emulation, and real world datasets, compared to other scalable GP modeling frameworks. Similar to Section 4, in our experiments we include `SGPR` for global approximation, `laGP` for local approximation, and patchwork kriging (`PK`) by Park and Apley (2018) introduced in Section 1. All the experiments presented in this section are carried out on a 2.6 GHz 6-Core Intel i7-9750H processor with 16 GB memory. We apply the following general settings for the different modeling frameworks, unless stated otherwise.

`TwinGP`: The number of global points g is set as $\min\{50d,\ \max\{\lfloor\sqrt{n}\rfloor,\ 10d\}\}$, i.e., at least $10d$ points are chosen to model the global trend, with an upper bound of $50d$. The local trend is modeled with $l = \max\{25,\ 3d\}$ points. The number of validation points used for estimating λ and η is set as $2g$. The $\boldsymbol{\theta}_g$ hyperparameter optimization is done with grid initialization and multi-starts as outlined in Basak et al. (2022).

`SGPR`: For a fair comparison with `TwinGP`, we set the number of inducing points to be $m = g + l$. We follow the implementation provided in `GPyTorch` documentation[1]. The number of iterations in hyperparameter optimization is modified from constant 100 to $\min\{250,\ \lfloor 50 \log(1+d) \rfloor\}$, and a learning rate of 0.05 is used instead of 0.01. In addition, separate lengthscales are learnt per dimension.

`laGP`: For each testing location, we start with l training points in its neighborhood and sequentially add 10 more points to the design that minimize the predictive variance. The `aGPsep` function provided in the `R` package by Gramacy (2016) is used to execute `laGP` in parallel. For emulation datasets where there is no observation noise, the nugget parameter is set to 10^{-7}, and for real world datasets the nugget is set to `NULL` in which case it is estimated.

`PK`: The training locations are partitioned into K disjoint blocks such that each block contains at least $10d$ points. The spatial tree algorithm used to make the partition benefits from K being

[1] `https://docs.gpytorch.ai/en/latest/examples/02_Scalable_Exact_GPs/SGPR_Regression_CUDA.html`

a power of 2, hence, we set $K = 2^{\lfloor \log_2 \frac{n}{10d} \rfloor}$. The number of pseudo observations B introduced at the boundaries of neigboring blocks to enforce continuity between the neighboring GPs is set to be 3. Our choice of B is conservative as the complexity of `PK` scales proportionally to B^3. We use the `MATLAB` implementation of `PK` provided by the author[2].

5.1 Evaluation criteria

The performance of the different modeling frameworks is assessed based on their prediction accuracy and total computation time. The prediction accuracy is quantified with root mean squared error (RMSE) and negative log predictive density (NLPD). RMSE measures the quality of point predictions from the model, while NLPD measures how well the predicted posterior distribution fits the testing data. Lower the RMSE and NLPD values, better the performance of the model. Given the testing set $\{\mathbf{X}_t^*, \mathbf{Y}_t^*\}$ with t testing locations, we have

$$\text{RMSE} = \sqrt{\frac{1}{t}\sum_{i=1}^{t}\{y_i^* - \mu(\mathbf{x}_i^*)\}^2}, \tag{27}$$

$$\text{NLPD} = \frac{1}{2t}\sum_{i=1}^{t}\left[\frac{\{y_i^* - \mu(\mathbf{x}_i^*)\}^2}{\sigma^2(\mathbf{x}_i^*)} + \log\{2\pi\sigma^2(\mathbf{x}_i^*)\}\right]. \tag{28}$$

5.2 Emulation

We consider three emulation problems here, the piston simulation function (Kenett and Zacks, 2021), the borehole function (Morris et al., 1993), and the Dette & Pepelyshev function (Dette and Pepelyshev, 2010). For a given emulation experiment, the input dimensions are scaled to $[0, 1]$ and $n = 10,000$ training locations are sampled uniformly from a d-dimensional hypercube, i.e., $\mathbf{X}_n \sim \text{Unif}(0,1)^{n\times d}$, and $2,000$ deterministic Sobol sequence in d dimensions are the testing locations. A GP is modeled on the training locations using `TwinGP`, `SGPR`, `laGP`, and `PK`. The performance of the fitted models is assessed based on the criteria given in Section 5.1. The procedure is repeated for 50 iterations with different training locations sampled similary, while the testing locations remain unchanged. The distribution of RMSE and NLPD over the 50 iterations are presented as box-and-whisker plots. Further information and code for the emulations problems can be obtained from Simon Fraser University's virtual library of simulation experiments[3].

[2] `https://www.chiwoopark.net/code-and-dataset`
[3] `https://www.sfu.ca/~ssurjano/emulat.html`

5.2.1 Piston simulation function

The piston simulation funtion models the cirular motion of a piston within a cylinder (Kenett and Zacks, 2021). There are $d = 7$ inputs, which are piston weight, piston surface area, initial gas volume, spring coefficient, atmospheric pressure, ambient temperature, and filling gas temperature. The response is the time taken to complete one cycle in seconds. The experiment results are given in Figure 3. We see that `TwinGP` performs the best in terms of RMSE, and for NLPD, on average, its performance is comparable to that of `PK`. Compuation time is the least for `TwinGP`, around 2 seconds, while `PK` took around 80 seconds.

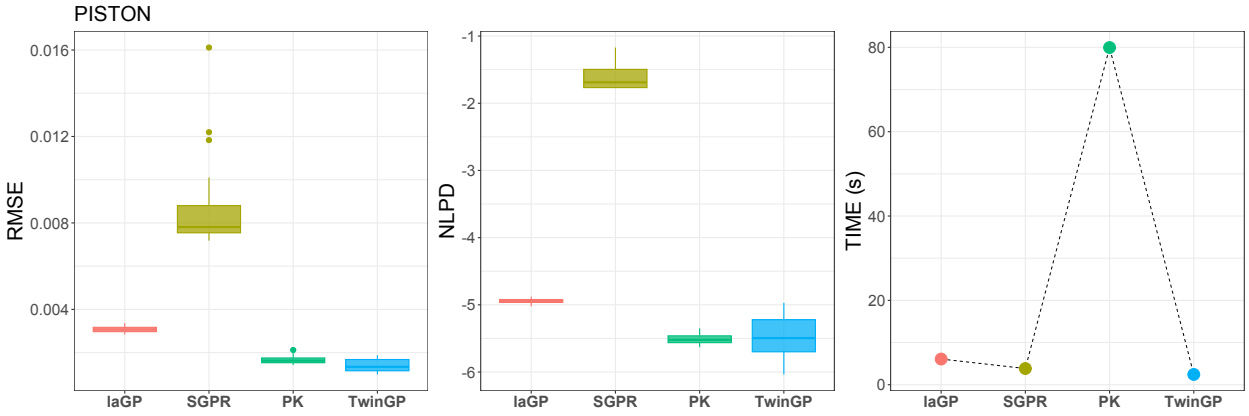

Figure 3: *Distribution of RMSE and NLPD over 50 iterations of modeling the piston simulation function with `laGP`, `SGPR`, `PK`, and `TwinGP`. The corresponding average computation time per iteration is given in the right most plot.*

5.2.2 Borehole function

The borehole function models water flow through a borehole (Morris et al., 1993). There are $d = 8$ inputs, which are radius of the borehole, radius of influence, transmissivity of upper aquifer, potentiometric head of upper aquifer, transmissivity of lower aquifer, potentiometric head of lower aquifer, length of borehole, and hydraulic conductivity of borehole. The response is the water flow rate in m^3/yr. The experiment results are given in Figure 4. We see that `PK` is the best performing modeling framework in terms of both RMSE and NLPD, with `TwinGP` being the close second. However, in terms of compuation time, `TwinGP` performance is far superior.

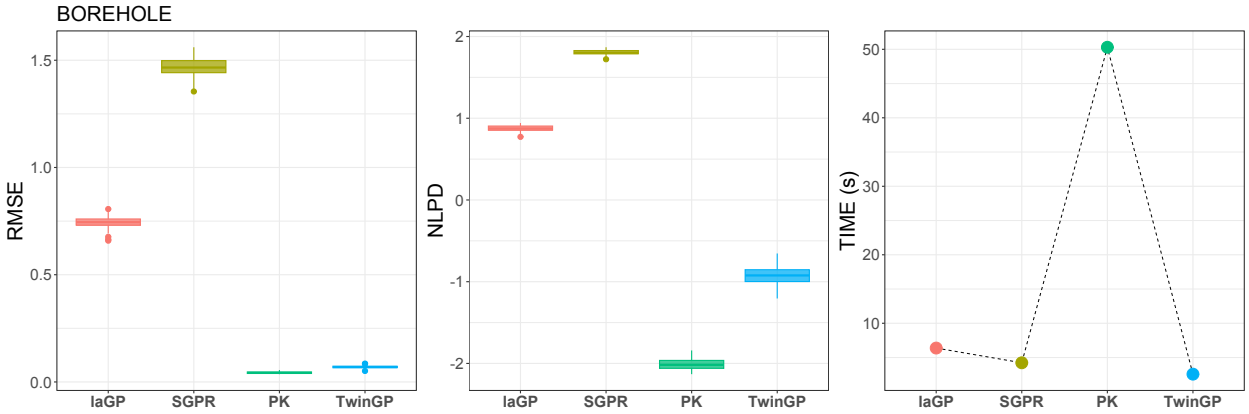

Figure 4: *Distribution of RMSE and NLPD over 50 iterations of modeling the borehole function with* `laGP`*,* `SGPR`*,* `PK`*, and* `TwinGP`*. The corresponding average computation time per iteration is given in the right most plot.*

5.2.3 Dette-Pepelyshev function

The Dette-Pepelyshev function from Dette and Pepelyshev (2010) with $d = 8$ input variables is heavily curved in some variables, and less so in others. Following the general settings described in the beginning of Section 5, running `PK` with $K = 64$ encountered numerical instabilites, hence, we increased K to 128. The experiment results are given in figure 5. In terms of RMSE, both `TwinGP` and `PK` perform the best with comparable results. `PK` performance is the best when it comes to NLPD, though at a very high computational cost.

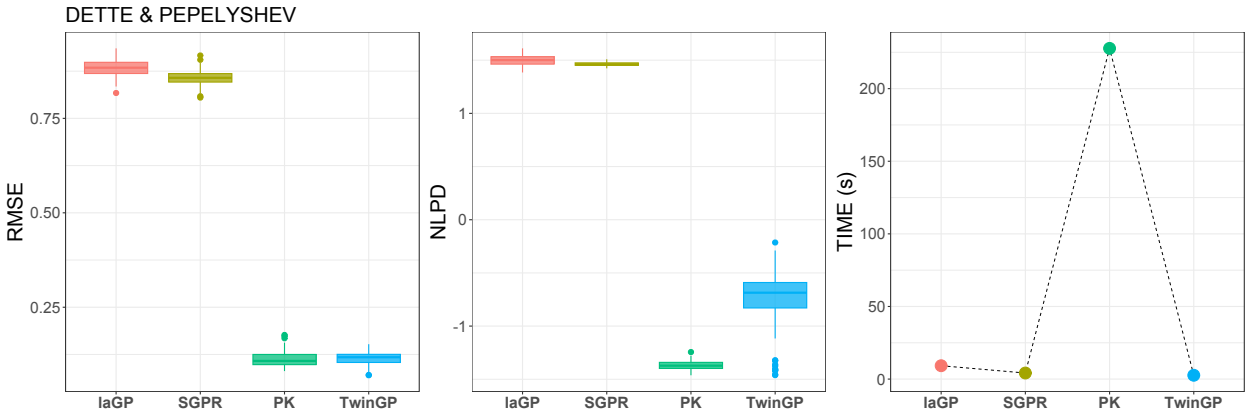

Figure 5: *Distribution of RMSE and NLPD over 50 iterations of modeling the Dette-Pepelyshev function with* `laGP`*,* `SGPR`*,* `PK`*, and* `TwinGP`*. The corresponding average computation time per iteration is given in the right most plot.*

5.3 Real world data

We consider four real world datasets here: the ozone column spatial dataset, protein tertiary structure dataset, Sarcos robotics dataset, and the flight delays dataset. The same datasets are considered in Park and Apley (2018). For a given dataset, we first randomly split the dataset in 90-10 proportion and a GP is modeled with `TwinGP`, `SGPR`, `laGP`, and `PK` on 90% of the dataset, and the remaining 10% is used for testing the performance of the fitted models. The experiment is repeated for 50 iterations using different random splits. The distribution of RMSE and NLPD over the 50 iterations are presented as box-and-whisker plots.

5.3.1 Ozone column

The ozone column dataset contains measurement of total column of ozone by the NIMBUS-7/TOMS satellite on October 1, 1988, at different latitudes and longitudes over the globe. There are $182,591$ observations with $d = 2$ inputs, latitude and longitude. The response is the total column of ozone. The experiment results are given in Figure 6. We omit `SGPR` in NLPD plot since it produced negative variances. In terms of RMSE, both `TwinGP` and `PK` provide the best performance. For NLPD, `TwinGP` on average performs better than `PK`, though `PK` is more consistent. Furthermore, `TwinGP` computation time is only 5 seconds, and is the fastest compared to the rest, while `PK` is the slowest at around 208 seconds. We used $K = 1024$ and $B = 3$ for `PK`, following the settings described at the beginning of Section 5. Decreasing K to 512 reduced the computation time to about 150 seconds with near identical RMSE and NLPD performance, even still, `PK` remained the slowest amongst rest of the models. Further decreasing K or increasing B for `PK` was computationally more expensive with no significant difference in RMSE and NLPD.

5.3.2 Protein tertiary structure

The protein tertiary structure dataset can be obtained from the UCI machine learning repository[4]. The dataset has $d = 9$ input variables relating to the physiochemical properties of protein tertiary structure, and the response is size of the protein residue. There are $45,730$ observations in total. The experiment results are given in Figure 7. `TwinGP` is the best performing model in terms of RMSE. For NLPD, both `TwinGP` and `PK` performance is comparable, and are better than the rest. Though `laGP` is the fastest at around 13 seconds with `TwinGP` being the close second at about 17

[4] `https://archive.ics.uci.edu/ml/datasets`

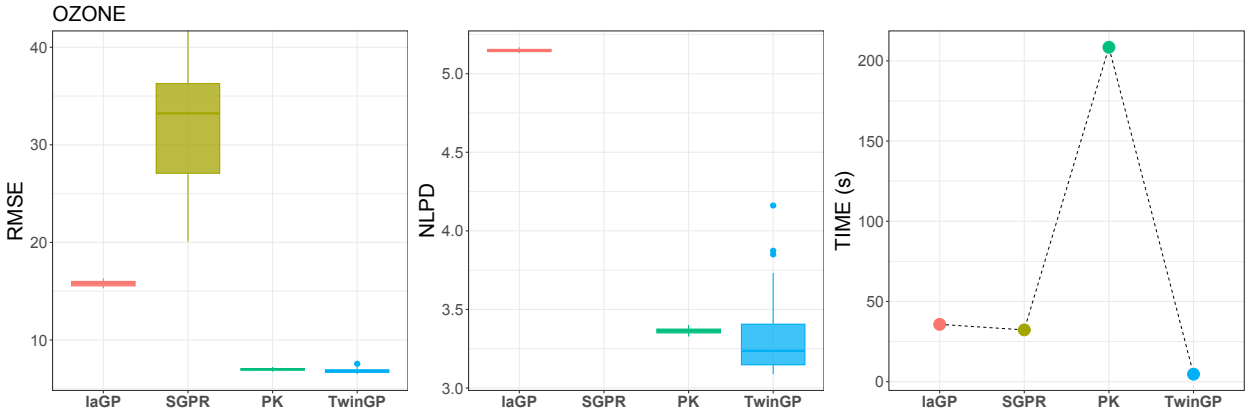

Figure 6: *Distribution of RMSE and NLPD over 50 iterations of modeling the ozone column dataset with* `laGP`*,* `SGPR`*,* `PK`*, and* `TwinGP`*. The corresponding average computation time per iteration is given in the right most plot.*

seconds, `laGP` is the worst performing model in terms of RMSE.

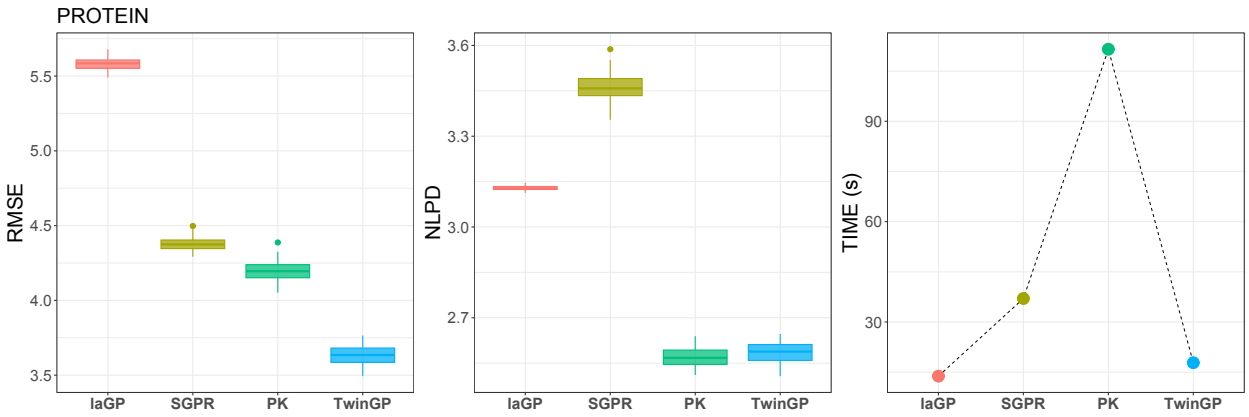

Figure 7: *Distribution of RMSE and NLPD over 50 iterations of modeling the protein structure dataset with* `laGP`*,* `SGPR`*,* `PK`*, and* `TwinGP`*. The corresponding average computation time per iteration is given in the right most plot.*

5.3.3 Sarcos robotics

The Sarcos robotics dataset[5] (Vijayakumar and Schaal, 2000) has $d = 21$ input variables representing positions, velocities, and accelerations of a seven degrees-of-freedom Sarcos anthropomorphic robot arm, and there are 7 responses corresponding to the 7 joint torques. Similar to Park and Apley (2018), we only consider the first response in modeling. The dataset is provided as training and

[5] `http://gaussianprocess.org/gpml/data`

testing sets which we combine, resulting in 48,933 total observations, before making the 90-10 random splits. The experiment results are given in Figure 8. Here we see that `TwinGP` is the best performing model in terms of all the evaluation criteria. `TwinGP` computation time is only about 47 seconds.

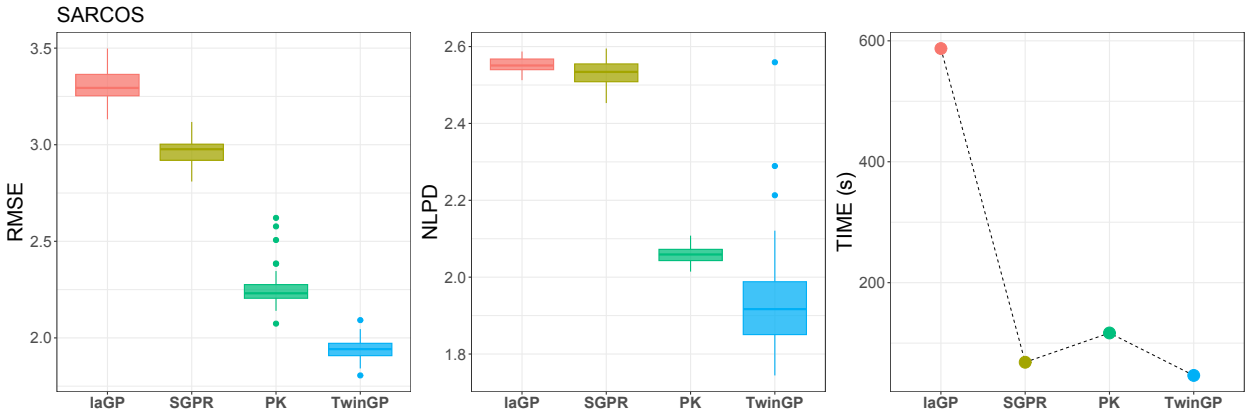

Figure 8: *Distribution of RMSE and NLPD over 50 iterations of modeling the sarcos robotics dataset with `laGP`, `SGPR`, `PK`, and `TwinGP`. The corresponding average computation time per iteration is given in the right most plot.*

5.3.4 Flight delays

The flight delays dataset[6] consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. In keeping with previous studies involving this dataset, 800,000 observations are randomly selected for this study from a total of about 120 million observations. Following Park and Apley (2018), $d = 8$ input variables are used in modeling, they are the age of the aircraft, distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month, and month. The response is the arrival time delay. The experiment results are given in Figure 9. `SGPR` ran out of memory on our machine, and a single iteration with `PK` did not finish in one hour, hence, we have excluded both `SGPR` and `PK` from the plots. Here, `laGP` performs better than `TwinGP` in terms of RMSE and NLPD, while `TwinGP` is twice as fast.

[6] `https://community.amstat.org/jointscsg-section/dataexpo/dataexpo2009`

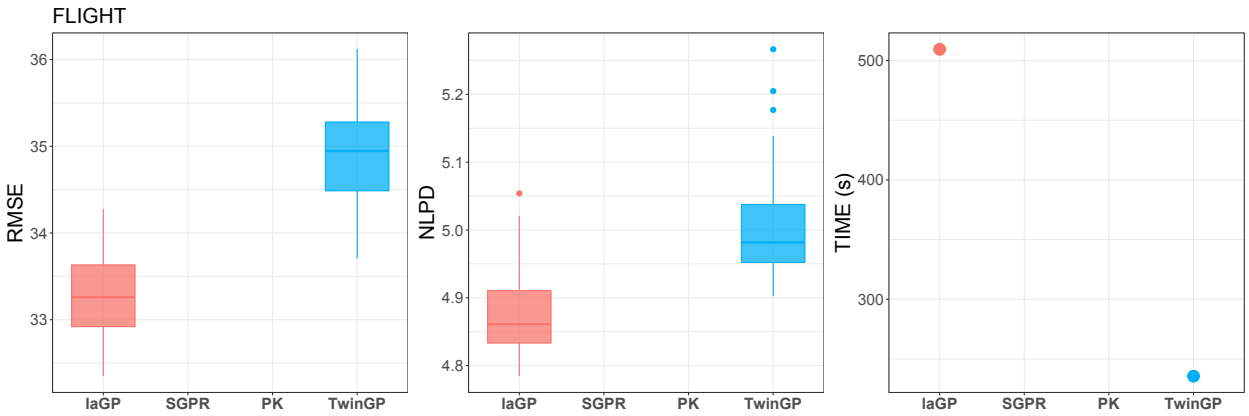

Figure 9: *Distribution of RMSE and NLPD over 50 iterations of modeling the flight delays dataset with* `laGP` *and* `TwinGP`*. The corresponding average computation time per iteration is given in the right most plot.*

6 Conclusions

In this article, we presented a unified global-local GP approximation that addresses the drawbacks of purely global, or local approximations in the literature. With our approximation framework, referred to as `TwinGP`, GP modeling on a million data points can be performed in just a few minutes on an ordinary personal computer. The two main features of `TwinGP` are: (i) the set of design points considered per testing location is a union of global points and local points, and (ii) the correlation function is modeled as sum of two kernels, one each to capture the global and local trend. The set of global points are identified with `Twinning`, and are independent of the testing location, while the local points are selected as nearest neighbors to a given testing location in the training data. The training complexity of our framework is $\mathcal{O}(g^3)$ where g is the number of global points used, and for t testing locations, the testing complexity is $\mathcal{O}(tg^2l)$ where l is the number of local points used. Our experiments show that the predictive performance with `TwinGP` is on par or better than state-of-the-art GP modeling frameworks aimed at large datasets, moreover, at a fraction of their computational cost.

Acknowledgments

This research is supported by U.S. National Science Foundation grants CMMI-1921646 and DMREF-1921873.

References

Ba, S. and Joseph, V. R. (2012). Composite gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, pages 1838–1860.

Basak, S., Petit, S., Bect, J., and Vazquez, E. (2022). Numerical issues in maximum likelihood parameter estimation for gaussian process interpolation. In *Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part II*, pages 116–131. Springer.

Chalupka, K., Williams, C. K. I., and Murray, I. (2013). A framework for evaluating approximation methods for gaussian process regression. *Journal of Machine Learning Research*, 14(1):333–350.

Chen, T. and Ren, J. (2009). Bagging for gaussian process regression. *Neurocomputing*, 72(7-9):1605–1610.

Dancik, G. M. and Dorman, K. S. (2008). mlegp: Statistical analysis for computer models of biological systems using R. *Bioinformatics*, 24(17):1966.

Das, K. and Srivastava, A. N. (2010). Block-gp: Scalable gaussian process regression for multimodal data. *2010 IEEE International Conference on Data Mining*, pages 791–796.

Deisenroth, M. and Ng, J. W. (2015). Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490. PMLR.

Dette, H. and Pepelyshev, A. (2010). Generalized latin hypercube design for computer experiments. *Technometrics*, 52(4):421–429.

Emery, X. (2009). The kriging update equations and their application to the selection of neighboring data. *Computational Geosciences*, 13(3):269.

Fasshauer, G. E. (2007). *Meshfree approximation methods with MATLAB*, volume 6. World Scientific.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. (2018). GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*.

Gneiting, T. (2002). Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508.

Gramacy, R. B. (2016). laGP: Large-scale spatial modeling via local approximate gaussian processes in R. *Journal of Statistical Software*, 72(1):1–46.

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. CRC press.

Gramacy, R. B. and Apley, D. W. (2015). Local gaussian process approximation for large computer

experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578.

Gramacy, R. B. and Lee, H. K. (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22:713–722.

Gramacy, R. B. and Lee, H. K. H. (2008). Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130.

Harari, O. and Steinberg, D. M. (2014). Convex combination of gaussian processes for bayesian analysis of deterministic computer experiments. *Technometrics*, 56(4):443–454.

Joseph, V. R. and Vakayil, A. (2022). Split: An optimal method for data splitting. *Technometrics*, 64(2):166–176.

Kaufman, C. G., Bingham, D., Habib, S., Heitmann, K., and Frieman, J. A. (2011). Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics*, 5(4):2470 – 2492.

Keerthi, S. and Chu, W. (2005). A matching pursuit approach to sparse gaussian process regression. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Kenett, R. S. and Zacks, S. (2021). *Modern industrial statistics: With applications in R, MINITAB, and JMP*. John Wiley & Sons.

Lawrence, N., Seeger, M., and Herbrich, R. (2002). Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Mak, S. and Joseph, V. R. (2018). Support points. *The Annals of Statistics*, 46:2562–2592.

Morris, M. D., Mitchell, T. J., and Ylvisaker, D. (1993). Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255.

Park, C. and Apley, D. (2018). Patchwork kriging for large-scale gaussian process regression. *The Journal of Machine Learning Research*, 19(1):269–311.

Park, C. and Huang, J. Z. (2016). Efficient computation of gaussian process regression for large spatial data sets by patching local gaussian processes. *Journal of Machine Learning Research*, 17(174):1–29.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959.

Rasmussen, C. and Ghahramani, Z. (2001). Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical science*, 4(4):409–423.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. Springer series in statistics. Springer-Verlag, New York.

Snelson, E. and Ghahramani, Z. (2005). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Snelson, E. and Ghahramani, Z. (2007). Local and global sparse gaussian process approximations. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 524–531, San Juan, Puerto Rico. PMLR.

Székely, G. J. and Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In van Dyk, D. and Welling, M., editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.

Tresp, V. (2000). A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741.

Vakayil, A. and Joseph, V. R. (2022). Data twinning. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(5):598–610.

Vanhatalo, J. and Vehtari, A. (2008). Modelling local and global phenomena with sparse gaussian processes. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, page 571–578, Arlington, Virginia, USA. AUAI Press.

Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the seventeenth international conference on machine learning (ICML 2000)*, volume 1, pages 288–293. Morgan Kaufmann.

Wendland, H. (2004). *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.