Hierarchically-partitioned Gaussian Process Approximation

Byung-Jun Lee **Jongmin Lee** **Kee-Eung Kim**
School of Computing,
KAIST, Republic of Korea
dlqudwns2002@gmail.com, jmlee@ai.kaist.ac.kr, kekim@cs.kaist.ac.kr

Abstract

The Gaussian process (GP) is a simple yet powerful probabilistic framework for various machine learning tasks. However, exact algorithms for learning and prediction are prohibitive to be applied to large datasets due to inherent computational complexity. To overcome this main limitation, various techniques have been proposed, and in particular, local GP algorithms that scales "truly linearly" with respect to the dataset size. In this paper, we introduce a hierarchical model based on local GP for large-scale datasets, which stacks inducing points over inducing points in layers. By using different kernels in each layer, the overall model becomes multi-scale and is able to capture both long- and short-range dependencies. We demonstrate the effectiveness of our model by speed-accuracy performance on challenging real-world datasets.

1 Introduction

Gaussian Processes (GPs) are a flexible nonparametric approach for machine learning, which allows learning prediction functions using Bayesian framework. Their main limitation, however, lies in the heavy computational demand, i.e. $\mathcal{O}(N^3)$ in the number of training instances N, making the exact GP inference tractable only for at most a few thousand instances. To overcome this limitation, a large number of approximation methods have been suggested to tackle this computational demand to gain scalability while maintaining most of its performance.

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54. Copyright 2017 by the author(s).

Among various approximation methods, the most intuitive and therefore most commonly studied are *inducing-point* methods, which use a smaller pseudo-dataset of size $M \ll N$ that summarizes the original data to reduce the computation to $\mathcal{O}(NM^2)$ in usual cases (Lázaro-Gredilla et al., 2010; Quiñonero-Candela and Rasmussen, 2005; Seeger et al., 2003; Snelson and Ghahramani, 2005, 2007; Titsias, 2009). These algorithms have started from widely different motivations but are analyzed thoroughly by two papers (Bui and Turner, 2014; Quiñonero-Candela and Rasmussen, 2005), which emphasized that these algorithms have common processes and the only difference lies in what prior they are using and what dependencies they are maintaining.

Despite the improvement in complexity, *inducing-point* algorithms are limited in medium-sized data, mainly due to the fact that the representational capacity of the model is restricted by the size of the pseudo-dataset. Since each pseudo-data point can only represent a small amount of information of the approximate posterior, it is not possible to compactly represent the function with small amount of pseudo-data, especially for highly varying functions with significant local structures. To maintain the desired accuracy, it is mandatory to scale M as the function gets complex and thus inducing-point methods typically fall short of scalability in terms of function complexity.

Another class of algorithms that address the scalability are *local* GP methods. Starting with partitioning the input space into the blocks of M points each, a local GP method replaces a single large global GP with a collection of small local ones to gain scalability. The computation is performed in $\mathcal{O}(NM^2)$ with constant M irrespective of the number of blocks, but the independence assumption among blocks and the discontinuities at boundaries due to partitioning significantly degrade performance. Bui and Turner (2014), and Moore and Russell (2015) have overcome some of these issues by keeping dependencies between adjacent partitions and outperformed previous methods on large datasets.

In this paper, we are interested in an approximate GP method that exhibits multi-scale hierarchical representation. Multi-scale hierarchical representation is often desirable for modeling real-world data, since it is possible to facilitate long-range propagation of local cues (Kato et al., 1996) for adaptation to the topology of the data. With such spirit, we develop a new inducing-point approximation scheme that effectively deals with long-range dependencies with multi-scale while mitigating scaling issues related to function complexity, and thus being capable of modeling more challenging data. We follow the basic structure of local GPs so that a large number of inducing points are maintained while locally partitioned so that the computational cost does not scale with function complexity. Then we introduce inducing points over inducing points to stack a hierarchical structure of inducing points, rather than having direct dependencies only among adjacent blocks at the same level. This is also a generalization of local and global consideration as in PITC (Quiñonero-Candela and Rasmussen, 2005) and PIC (Snelson and Ghahramani, 2007) being able to learn long-range dependencies.

The paper is organized as follows: in the following section, some existing GP approximation methods will be briefly reviewed. We then introduce the hierarchical representation and the inference algorithm in section 3. The demonstration of the proposed algorithm on challenging data showing that our hierarchical multi-scale representation can handle complex functions and long-range dependencies without scaling the computational cost, is presented in section 4.

2 Background

2.1 Gaussian process regression

The Gaussian process (GP) (Rasmussen and Williams, 2006) is the distribution over functions with a real continuous domain, for which any finite samples has joint Gaussian distribution. It is parameterized by a mean function, which is usually assumed to be $\mu(\mathbf{x}) = 0$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. Given a GP-distributed random function $f(x)$ and D-dimensional input points $\{\mathbf{x}_n\}_{n=1}^N$, the vector of function values $\mathbf{f} = \{f(\mathbf{x}_n)\}_{n=1}^N$ is therefore a multivariate Gaussian, $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_{\mathbf{ff}})$, $(\boldsymbol{K}_{\mathbf{ff}})_{n,n'} = k(\mathbf{x}_n, \mathbf{x}_{n'})$. One of the popular choices is the squared exponential covariance, $k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2l^2}\right)$, where the hyperparameters σ^2 and l^2 specify the prior variance and correlation length-scale, respectively.

This paper will mainly consider non-linear Bayesian regression: in regression, the main objective is to predict the function value $\mathbf{f}^*$ at test point $\mathbf{x}^*$ given the noisy observation $\mathbf{y} = \mathbf{f} + \epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \boldsymbol{I})$ on the input points. Thus, the observations are Gaussian $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_{\mathbf{ff}} + \sigma_n^2 \boldsymbol{I})$, and prediction can be done by computing the predictive distribution $p(\mathbf{f}^*|\mathbf{y})$, which is also Gaussian and thus can be computed analytically. After a simple manipulation of Gaussians, the mean and covariance are given by

$$\boldsymbol{\mu}^* = \boldsymbol{k}_{*\mathbf{f}}(\boldsymbol{K}_{\mathbf{ff}} + \sigma_n^2 \boldsymbol{I})^{-1}\mathbf{y} \tag{1}$$

$$\boldsymbol{\Sigma}^* = k_{**} - \boldsymbol{k}_{*\mathbf{f}}(\boldsymbol{K}_{\mathbf{ff}} + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{k}_{\mathbf{f}*}, \tag{2}$$

where k_{**} and $\boldsymbol{k}_{*\mathbf{f}}$ are covariance function evaluated on test-test inputs and test-train inputs.

When performing regression using GPs, hyperparameters $\{\sigma^2, l^2\}$ and the noise variance σ_n^2 are often selected by maximizing a marginal likelihood, $p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \boldsymbol{K}_{\mathbf{ff}} + \sigma_n^2 \boldsymbol{I})$ as specified above. Although such an objective is generally non-convex, the maximization is usually performed by gradient-based methods. These calculations have a bottleneck in computing $(\boldsymbol{K}_{\mathbf{ff}} + \sigma_n^2 \boldsymbol{I})^{-1}$, requiring $\mathcal{O}(N^3)$ computations per iteration in optimization. The prediction can be made in $\mathcal{O}(N^2)$ per test point with pre-computed $(\boldsymbol{K}_{\mathbf{ff}} + \sigma_n^2 \boldsymbol{I})^{-1}$. Because of the cubic computation time bottleneck, GP cannot be straightforwardly applied to large datasets.

2.2 Inducing-point methods

To mitigate high computational demand in GPs, a number of approximation methods have been proposed. Here we review *inducing-point* methods, which represent the latent function with its values $\mathbf{u} = \{u_m\}_{m=1}^M$ at pseudo-input points $\{\mathbf{x}_m^\dagger\}_{m=1}^M$, comprising a pseudo-dataset. These approximation methods can be seen as replacing the full GP prior with the sparse ones with some of the dependencies being removed; referring to Bui and Turner (2014), the sparse models used by these methods can be explained in the KL minimization framework.

More specifically, consider the joint GP prior $p(\mathbf{f}, \mathbf{f}^*) = \mathcal{N}(\mathbf{0}, \boldsymbol{K}_{(\mathbf{f},\mathbf{f}^*)(\mathbf{f},\mathbf{f}^*)})$. Inducing-point methods introduce conditional independence $p(\mathbf{f}, \mathbf{f}^*) = \int p(\mathbf{f}|\mathbf{u})p(\mathbf{f}^*|\mathbf{u})p(\mathbf{u})d\mathbf{u}$ so that the information on training values $\mathbf{f}$ are passed only via the pseudo-input values $\mathbf{u}$. $p(\mathbf{f}|\mathbf{u})$ is further approximated to obtain a simpler model. The most simple yet popular method is the Fully Independent Training Conditional (FITC) approximation (Snelson and Ghahramani, 2005). FITC assumes conditional independence among every function values $\mathbf{f}$ given $\mathbf{u}$, which turns out to be equivalent to a factor analysis model.

The inducing points can be learned using variational inference (Titsias, 2009), as well as its stochastic variant for large data (Hensman et al., 2013). It can be also extended so that inducing points have a different covariance function (Figueiras-Vidal and Lázaro-gredilla, 2009).

Although FITC assumes full conditional independence, we could make approximation more accurate by retaining more dependencies. Bayesian Committee Machine (BCM) (Tresp, 2000), later renamed to Partially Independent Training Conditional (PITC) in (Quiñonero-Candela and Rasmussen, 2005), retains intra-block dependency to achieve better prediction. Partially Independent Conditional (PIC) (Snelson and Ghahramani, 2007) extends PITC to retain dependencies between training input values $\mathbf{f}$ and test input values $\mathbf{f}^*$ so that $\mathbf{f}^*$ can directly benefit from intra-block dependency while incurring a small amount of additional computation.

There are also a number of methods that do not follow this framework, e.g. Sparse Spectrum Gaussian Process (SSGP) (Lázaro-Gredilla et al., 2010) computes *spectral* points (instead of inducing points) to approximate stationary covariance function for time-series data.

2.3 Local GP methods

Local GP methods simply decompose the input domain into smaller regions and predict using training inputs only in the region where the test input belongs to. Local GP methods have attracted a lot of attention lately, since inducing-point methods typically need to grow the size of the pseudo-dataset in order to increase their representational power, and as such they do not perform well on large and complex datasets in practice[1]. However, it is also well known that local GP methods heavily suffers from discontinuities in the predictions at the boundaries of regions. As such, prior studies focused on mitigating this discontinuity by using the information from adjacent blocks, e.g. combining local regressors (Nguyen-Tuong et al., 2009; Park et al., 2011).

More recent approaches propose probabilistic models for achieving this. The tree-structured GP approximation (Bui and Turner, 2014) removes most of the full inter-block dependencies of PIC while retaining dependencies among adjacent blocks. From the local GP perspective, it can be seen as introducing dependencies among inducing points in adjacent blocks. This algorithm imposes a chain structure in time series data and a tree structure in higher dimensional data and shown to outperform inducing point methods described in the previous section. The GP random field (Moore and Russell, 2015) constructs a pairwise Markov model between local blocks, which results in additional inter-block dependencies that mitigate the discontinuity in 2D data, but the exact inference becomes intractable. The paper introduces a new Bethe-type objective for approximate inference, which turns out to be a generalization of BCM in an unsupervised learning setting.

[1]Despite the local partitioning in PIC and PITC, they still lack scalability since the overall number of inducing points must increase with the global complexity of the data.

2.4 Multi-scale GP algorithms

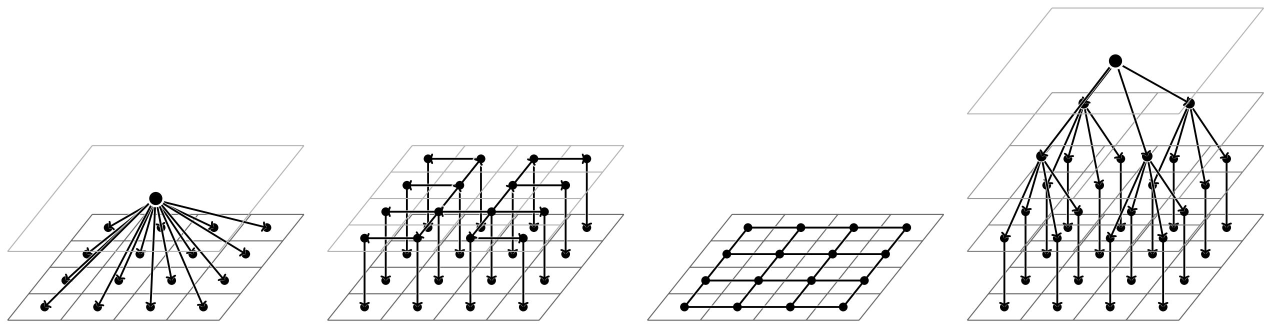
Real-world datasets typically exhibit multi-scale structure, having significant features at multiple scales of time and space. If we force GP models with the simple single-scale covariance function to represent such dataset, they only adapt to one of these scales, and eventually fails to represent the overall structure efficiently (see figure 2).

The multi-scale trait in a dataset can be generally modeled by introducing hierarchy, i.e. multiple levels of different scales, with the top level having the largest scale (Wainwright et al., 2001). This idea has been explored in hierarchical multi-scale GP models that impose a GP prior on the mean of GP, resulting in a hierarchy of GPs with different scales at each level, and the covariance function of the marginal GP becomes the sum of covariance functions at each level (Fox and Dunson, 2012; Park and Choi, 2010).

There are also multi-scale models that do not have a hierarchical representation. For example, Walder et al. (2008) extended FITC by allowing inducing points to have different length-scales. However, since this model is a variant of FITC, it still needs an increasing number of inducing points to adequately capture hierarchical structure in the data.

3 Hierarchically-partitioned Gaussian Process Approximation

We propose a GP approximation framework that combines the advantages of inducing-point methods, local GPs and multi-scale GPs. We start with the local FITC: N input points and M inducing points are both partitioned into K blocks with possibly different sizes denoted by $\{\mathbf{x}_{B_k}\}_{k=1}^K$ and $\{\mathbf{u}_{B_k}^{(1)}\}_{k=1}^K$ respectively (the superscript (1) explicitly denotes that the inducing points are at the lowest level). It is also naturally assumed that function values are divided into blocks $\{\mathbf{f}_{B_k}\}_{k=1}^K$. They are conditionally independent to function values in other blocks given the inducing points in the corresponding block. The tree-structured GP (Bui and Turner, 2014) then constructs a spanning



(a) PITC (b) Tree GP (c) GPRF (d) HPGPA

Figure 1: Graphical models of the different local-GP based fully probabilistic approximation methods.

tree over blocks that introduce dependencies among inducing points in adjacent blocks.

In contrast, our model introduces hierarchical dependencies among blocks to represent dependency among blocks that are far apart, i.e. a hierarchy of inducing-point blocks over inducing-point blocks. In the tree structure representing the hierarchy of inducing-point blocks, the input region corresponding to a block at a specific level is defined to be the union of the input regions corresponding to its children. Therefore, the root node corresponding to the inducing-point block covers the whole input region. The full joint distribution over $\{\{\mathbf{u}_{B_k}^{(h)}\}_{k=1}^{K_h}\}_{h=1}^{H}$ and $\{\mathbf{f}_{B_k}\}_{k=1}^{K}$ is factored, given by

$$\begin{aligned} q(\{\{\mathbf{u}_{B_k}^{(h)}\}_{k=1}^{K_h}\}_{h=1}^{H}, \{\mathbf{f}_{B_k}\}_{k=1}^{K}) \\ = q(\{\{\mathbf{u}_{B_k}^{(h)}\}_{k=1}^{K_h}\}_{h=1}^{H}) \prod_{k=1}^{K} q(\mathbf{f}_{B_k} | \mathbf{u}_{B_k}^{(1)}) \end{aligned} \tag{3}$$

$$\begin{aligned} q(\{\{\mathbf{u}_{B_k}^{(h)}\}_{k=1}^{K_h}\}_{h=1}^{H}) \\ = q(\mathbf{u}^{(H)}) \prod_{k=1}^{K_{H-1}} q(\mathbf{u}_{B_k}^{(H-1)} | \mathbf{u}^{(H)}) \cdot \\ \prod_{l \in children(B_k)} q(\mathbf{u}_l^{(H-2)} | \mathbf{u}_{B_k}^{(H-1)}) ... \end{aligned} \tag{4}$$

Similar to FITC and PIC, minimizing KL divergence $KL(p(\mathbf{f}, \mathbf{u}) | q(\mathbf{f}, \mathbf{u}))$ yields the following distributions:

$$q(\mathbf{u}^{(H)}) = p(\mathbf{u}^{(H)}) = \mathcal{N}(\mathbf{0}, \boldsymbol{K}_{\mathbf{u}^{(H)},\mathbf{u}^{(H)}}) \tag{5}$$

$$\begin{aligned} q(\mathbf{u}_k^{(h)} | \mathbf{u}_{l=par(k)}^{(h+1)}) = p(\mathbf{u}_k | \mathbf{u}_l) \\ = \mathcal{N}(\boldsymbol{K}_{\mathbf{u}_k,\mathbf{u}_l} \boldsymbol{K}_{\mathbf{u}_l,\mathbf{u}_l}^{-1} \mathbf{u}_l, \\ \boldsymbol{K}_{\mathbf{u}_k,\mathbf{u}_k} - \boldsymbol{K}_{\mathbf{u}_k,\mathbf{u}_l} \boldsymbol{K}_{\mathbf{u}_l,\mathbf{u}_l}^{-1} \boldsymbol{K}_{\mathbf{u}_l,\mathbf{u}_k}) \end{aligned} \tag{6}$$

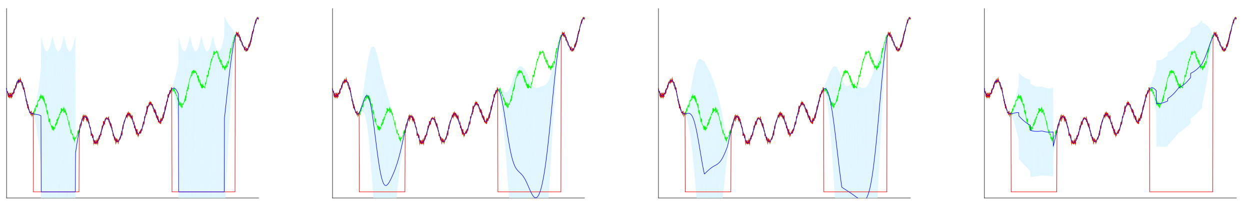
$$\begin{aligned} q(\mathbf{f}_k | \mathbf{u}_k^{(1)}) = p(\mathbf{f}_k | \mathbf{u}_k) \\ = \mathcal{N}(\boldsymbol{K}_{\mathbf{f}_k,\mathbf{u}_k} \boldsymbol{K}_{\mathbf{u}_k,\mathbf{u}_k}^{-1} \mathbf{u}_k, \\ \boldsymbol{K}_{\mathbf{f}_k,\mathbf{f}_k} - \boldsymbol{K}_{\mathbf{f}_k,\mathbf{u}_k} \boldsymbol{K}_{\mathbf{u}_k,\mathbf{u}_k}^{-1} \boldsymbol{K}_{\mathbf{u}_k,\mathbf{f}_k}) \end{aligned} \tag{7}$$

which turns out to be identical to the conditional distributions in the unapproximated model. Figure 1 compares the graphical model of HPGPA with relevant local GP models. The test input point is assigned to the closest block in the lowest level (e.g. B_k), and the prediction is done locally with the marginal posterior of local inducing points $p(\mathbf{u}_{B_k}^{(1)} | \mathbf{y})$ since the function values are conditionally independent to those in other blocks:

$$p(\mathbf{f}_{B_k}^* | \mathbf{y}) = \int p(\mathbf{f}_{B_k}^* | \mathbf{u}_{B_k}^{(1)}) p(\mathbf{u}_{B_k}^{(1)} | \mathbf{y}) d\mathbf{u}_{B_k}^{(1)}. \tag{8}$$

3.1 Inference and training

Inference, i.e. computing the marginal posterior of local inducing points $\{p(\mathbf{u}_{B_k}^{(1)} | \mathbf{y})\}_{k=1}^{K}$, is done by the upward-downward algorithm for Gaussian networks Desbouvries et al. (2006). In upward pass, we recursively calculate $p(\mathbf{u}_k | \mathbf{y}_{\in \mathbf{u}_k}, \mathbf{u}_{l=par(k)})$ with eq. 6 and eq. 7. Due to the assumption that $\mathbf{u}_k$ is conditionally independent to those of $\mathbf{y}$ that belong to other region given its parent $\mathbf{u}_l$, the distribution is equivalent to $p(\mathbf{u}_k | \mathbf{y}, \mathbf{u}_{l=par(k)})$. In the downward pass, we start by calculating $p(\mathbf{u}^{(H)} | \mathbf{y})$, and proceed to lower levels, recursively computing $p(\mathbf{u}_k | \mathbf{y}) = \int p(\mathbf{u}_{l=par(k)} | \mathbf{y}) p(\mathbf{u}_k | \mathbf{y}, \mathbf{u}_l) d\mathbf{u}_l$. Training, i.e. hyperparameter fitting, requires computing the marginal likelihood and its gradient. These can be easily computed with intermediate results obtained in the inference algorithm. The computational cost is $\mathcal{O}(L^3 K) \approx \mathcal{O}(NL^2)$ where K is the number of blocks at the lowest level and L is the average number of observations per block. Note that hierarchical structure does not increase asymptotic complexity since the total number of blocks does not exceed $2K$. The overall inference and training algorithm is similar to Bui and Turner (2014): see supplementary material for details.



(a) Local GP
(smse: 11.289)

(b) Tree-GP
(smse: 4.976)

(c) HPGPA-single
(smse: 7.161)

(d) HPGPA-multi
(smse: 0.315)

Figure 2: Various local GP based algorithms interpolating synthetic data. The green, red, and blue lines show original signal, training data, and prediction of each algorithm respectively. The local GP method can easily fail on an interpolating task with missing intervals much larger than the local block, especially when there exist various trends. In contrast, multi-scale HPGPA can handle such cases, high-level inducing points effectively capturing long-term trends (using flexible kernels like spectral mixture would easily interpolate the problem; nevertheless, simple SE kernel is used in this case to pedagogically demonstrate the limitation).

3.2 Multi-scale regression

The model and algorithm explained above will not work well under single-scale assumption, i.e. enforcing the same covariance function in every layer. With K blocks in the lowest level, inducing points of the root block will be K times more sparse than in the leaf blocks. In such a case, the length-scale of the single covariance function will be fit to the value close to that at the lowest level, which is too short for higher level inducing points and render them useless. In some sense, a single-scale HPGPA will degenerate to a naive local GP as the size of data increases.

Here we use the technique by Melkumyan and Ramos (2011) that deals with constructing multi-task covariance functions out of arbitrary stationary covariance functions. In case of a stationary covariance function given by a convolution of basis functions: $k(\boldsymbol{\tau} = \mathbf{x} - \mathbf{x}') = \int g(\boldsymbol{\tau} - \mathbf{u})g(\mathbf{u})d\mathbf{u}$, if we define cross-covariance function as convolving two basis functions each from different covariance functions, it is proved that it guarantees the nonnegative definiteness of overall covariance function. Making use of the convolution theorem, the cross-covariance function of arbitrary stationary kernels k_1 and k_2 can be obtained from Fourier and inverse Fourier transform,

$$k_{1,2}(\boldsymbol{\tau}) = (2\pi)^{-\frac{D}{2}} \int \mathcal{F}^{-1}\left[\sqrt{\mathcal{F}[k_1(\boldsymbol{\tau} - \mathbf{u})]}\right] \cdot \mathcal{F}^{-1}\left[\sqrt{\mathcal{F}[k_2(\mathbf{u})]}\right] d\mathbf{u}. \tag{9}$$

For example, in the case of SE kernel with two different hyper-parameters, we obtain the following cross-covariance,

$$k_1(\boldsymbol{\tau}) = \sigma_1 \exp\left[-\frac{1}{2}\boldsymbol{\tau}^\top \boldsymbol{P}_1^{-1}\boldsymbol{\tau}\right] \tag{10}$$

$$k_2(\boldsymbol{\tau}) = \sigma_2 \exp\left[-\frac{1}{2}\boldsymbol{\tau}^\top \boldsymbol{P}_2^{-1}\boldsymbol{\tau}\right] \tag{11}$$

$$k_{1,2}(\boldsymbol{\tau}) = \sigma_1\sigma_2 \frac{|\boldsymbol{P}_1|^{\frac{1}{4}}|\boldsymbol{P}_2|^{\frac{1}{4}}}{|(\boldsymbol{P}_1 + \boldsymbol{P}_2)/2|^{\frac{1}{2}}} \cdot \exp\left[-\boldsymbol{\tau}^\top(\boldsymbol{P}_1 + \boldsymbol{P}_2)^{-1}\boldsymbol{\tau}\right]. \tag{12}$$

The prior variance σ's except the lowest-level data kernel vanish during inference, so there is no need of optimizing σ's at each level. As for optimizing length-scales, there are two choices: (1) optimize length-scales independently at each level, or (2) tie the length-scales with the average distance among inducing points at the corresponding level. The first option turned out to over-fit, so we chose the second option, e.g. doubling at each level in the case of a 1D binary tree.

The power of multi-scale HPGPA is demonstrated in figure 2, compared to a number of local GP methods on a synthetic dataset that contains long-term as well as short-term trends.

3.3 Discontinuities at boundaries

In tree-GP and single-scale HPGPA, the continuity at a boundary of blocks can be guaranteed by placing inducing points at the boundaries. If two points each in adjacent blocks in tree-GP, or two points each in adjacent blocks and one point in their parent block in HPGPA are placed at the same site, $p(u_k|u_{par(k)})$ of corresponding points from eq. 6 is guaranteed to give the same mean with zero variance. This tying constraint is kept throughout the inference process, resulting in a continuous prediction over adjacent blocks.

In multi-scale HPGPA, however, this simple trick does

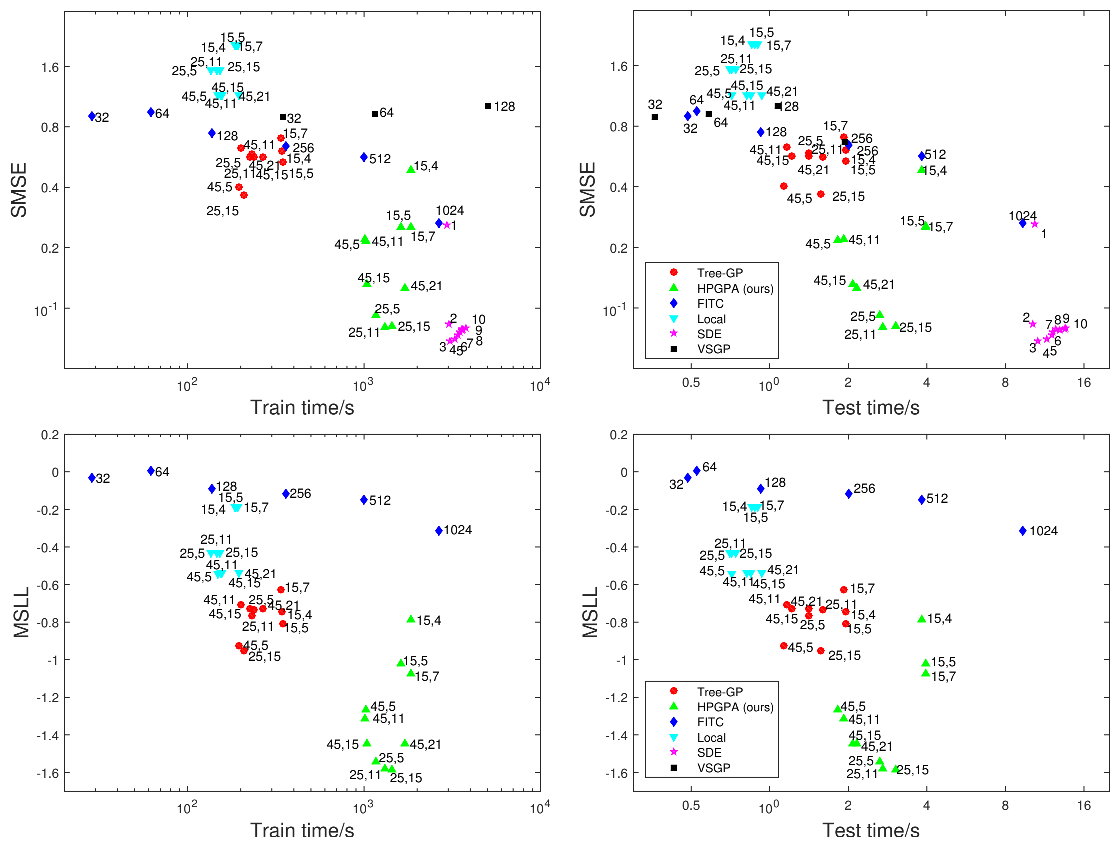


Figure 3: Interpolation errors versus train/test computation times of different approximation algorithms on the power consumption dataset. The numbers next to FITC and VSGP plots are the number of the global inducing points used, and the numbers next to SDE plots are the order of approximation. The pairs of numbers next to other plots are the sizes of blocks at the lowest level and inducing points per block. The MSLL results of VSGP and SDE lie outside the plot region(> 0.2).

not work. Considering the variance of $p(u_k|u_{par(k)})$ for instance, we have $k_1 - k_{1,2}^\top k_2^{-1} k_{2,1}$. If we have inducing points at the same point, exponential terms become 1 and results in variance given by

$$var = \sigma_1^2 - \sigma_1^2 \frac{|\boldsymbol{P}_1|^{\frac{1}{4}} |\boldsymbol{P}_2|^{\frac{1}{4}}}{|(\boldsymbol{P}_1 + \boldsymbol{P}_2)/2|^{\frac{1}{2}}}. \tag{13}$$

Since this is the geometric mean over arithmetic mean, the only condition when the variance vanishes is when two kernels have the same length-scale. It is therefore not possible to enforce a continuous prediction in HPGPA with the same trick used in tree-GP or single-scale HPGPA. Nonetheless, small prediction jumps at boundaries were insignificant compared their performance on a large dataset in practice. This issue can be resolved by introducing additional dependencies with approximate inference algorithms (Wainwright et al., 2001), and will be considered in future work.

4 Experiments

We consider two challenging large-scale real-world prediction tasks to demonstrate the performance of the proposed method, comparing speed-accuracy tradeoffs as in previous literature (Bui and Turner, 2014; Chalupka et al., 2013; Snelson and Ghahramani, 2007). In the experiments, the squared exponential kernel, $k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2l^2}\right)$ is used, and the hyperparameters are found by BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm until convergence or maximum 50 iterations. Here we use two widely used metrics, SMSE (standardized mean squared error) and MSLL (mean standardized log loss) (Rasmussen and Williams, 2006). Our method is implemented in MATLAB using the GPML package and extending the tree-GP code[2] and executed on workstations with two Intel Xeon E5-2660 v3 @ 2.60GHz CPUs.

[2][GPML] http://www.gaussianprocess.org/gpml/code/matlab/doc [Tree-GP] https://github.com/thangbui/tsgp [HPGPA] https://github.com/dlqudwns/HPGPA

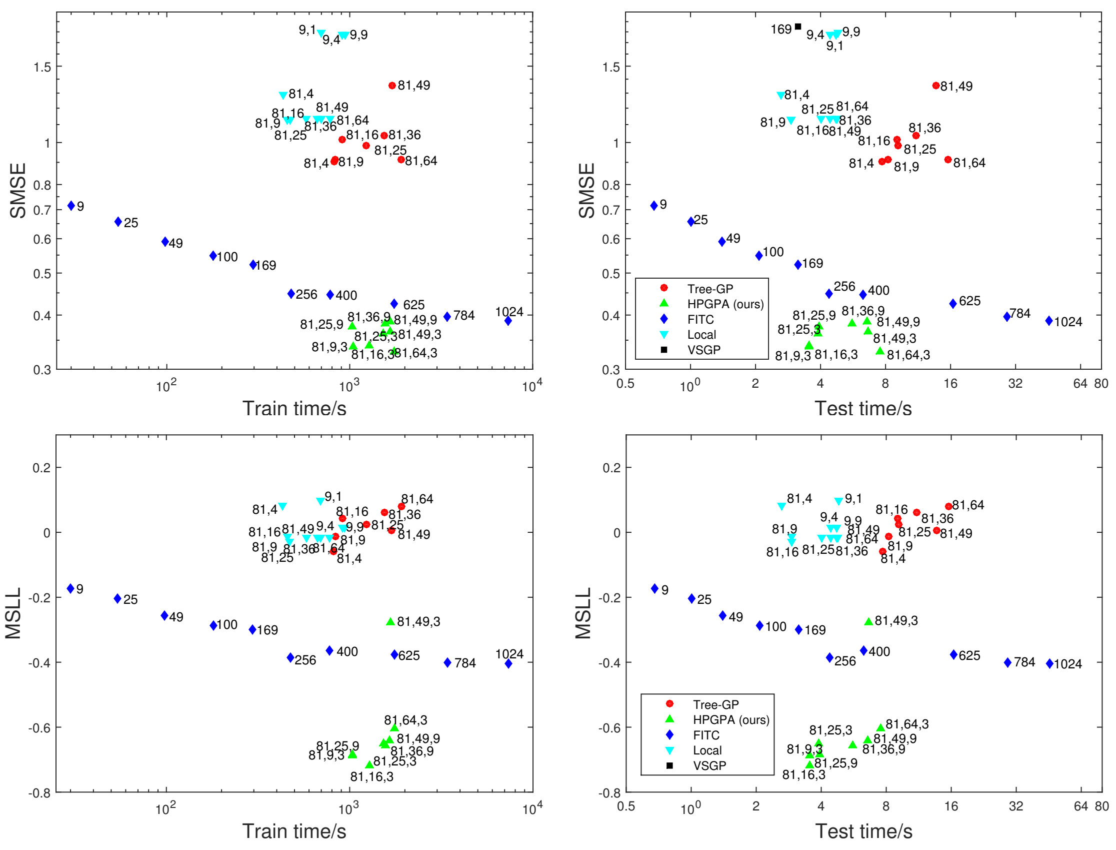


Figure 4: Interpolation errors versus train/test computation times of different approximation algorithms on the terrain dataset. The numbers next to the plots follow the description of figure 3, while the third numbers of HPGPA plots indicate the fanout (3×3 or 9×9) of the tree. The MSLL results of VSGP lie outside the plot region(> 0.3).

4.1 Power consumption data

The first experiment concerns with interpolating missing intervals in a large-scale time-series data. The dataset, household electric power consumption, was obtained from the UCI machine learning repository (Lichman, 2013)[3]. This dataset is a recording of active power consumption at every minute, which is smoothed with a moving average filter of length 60 (one hour) in order to remove certain noises in the data. In order to make the experiments with all the algorithms feasible, we used a portion of data of length 184320, which corresponds to 128 days. We stress that using more data is not an issue for our method since it scales very well. 50 intervals of length 64 are removed in training, and they are interpolated with all the methods and compared with the original data. The length of the missing interval is an important issue because local GP based methods start to collapse when the size of missing interval is longer than the size of a single local block. Our method constructs a binary tree over the local blocks, and the length-scale of the kernels are tied to be doubled at each level.

We compared the hierarchically-partitioned GP approximation model with FITC, local GP, tree-GP, VSGP (Walder et al., 2008) and SDE (Sarkka et al., 2013). The VSGP algorithm is a multi-scale FITC variant introduced before, while SDE is a powerful algorithm specialized in dealing with time-series data that uses linear Gaussian state space model. In case of the local GP algorithm used here, we used local inducing points to represent the corresponding block of data in order to impose the same condition with other algorithms (tree-GP and HPGPA). Although there are many other algorithms (PIC, VFE, SSGP, etc...) to be compared with, their trends were very similar to that of FITC, thus their results are omitted in this paper.

Figure 3 shows the data recovery performance versus computation time under different parameter settings. Although FITC gradually improves with the number of inducing points, it is outperformed by other GP

[3]https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

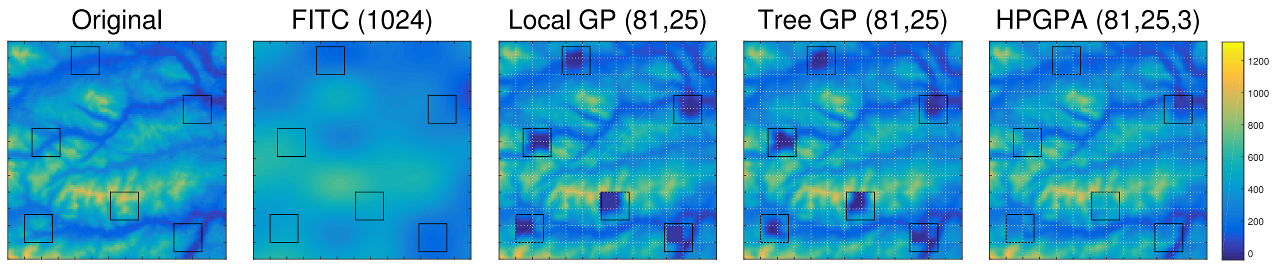


Figure 5: The visualization of interpolations made by GP methods on the terrain dataset. The white dotted lines represent the block boundaries at the data level, and the black squares are the missing sites.

methods since it requires a lot of inducing points to capture the short-term as well as long-term trends in the data. VSGP followed a similar trend to FITC. On the other hand, local GP performs worst in terms of accuracy since the length of missing intervals is larger than those of local blocks. Tree-GP performs better than local GP since it is able to harness the information from adjacent blocks, but surpassed by HPGPA with only twice the amount of computation. This is because the hierarchical multi-scale aspect of HPGPA was able to accurately capture the long-term trend over the missing intervals. SDE, which is one of the special GP implementations on time series domain, shows similar smse to that of HPGPA; however, HPGPA shows better MSLL and can be applied to other domains.

4.2 Terrain data

In the second experiment, we compare the performances of algorithms in a 2D terrain dataset where the task is to predict the altitude of corresponding location[4]. We used 729×729 (531k) sized data, which was down-sampled from the original 3645×3645 data (corresponds to 183km$\times$183km region). 80 randomly picked sites of size 15×15 (3.75km by 3.75km) are removed in training, interpolated by all the algorithms, and compared with the original data. The locations to be missed out from the data were randomly selected with probability proportional to the variance of altitudes in order to make the interpolation challenging (predicting the altitudes of a site in the ocean would be not interesting!). The tree in HPGPA algorithm was constructed in the way that each parent node has 3×3 (9) or 9×9 (81) children, and the length-scales of the kernels were tied to increase/decrease at each level proportional to the area of the region covered by the block.

Figure 4 shows the quality of recovered intervals versus the computation time. The results are very similar to the power consumption dataset experiment in the previous section: FITC does not scale well, local/tree GP makes inaccurate predictions although quite fast while HPGPA clearly makes better predictions than other methods.

This situation can be directly observed by visualization, shown in figure 5: FITC creates blurry images due to the deficiency of inducing points. Local GP and tree-GP are not able to interpolate well with a single-scale kernel, creating "sink holes". However, HPGPA successfully recovers the image close to the original image by learning complex trends with the hierarchical multi-scale modeling.

5 Conclusions and Future Work

We presented HPGPA, a scalable multi-scale approximation method for GPs using hierarchical representation. HPGPA harnesses the advantages from the scalability of local GP methods with local blocks and the expressibility of hierarchical GP methods with multi-scale kernels, thus successfully captures various trends at different scales present in large complex datasets. HPGPA leverages cross-covariance kernels originally proposed for multi-task GPs in order to deal with different kernels at each level of the hierarchy. Inference and training are done efficiently via message passing algorithm, which results in the same asymptotic computational complexity as that of local GP. We demonstrated the effectiveness of HPGPA through experiments on challenging regression tasks on real-world large datasets.

The first and foremost remaining work to be done as future work is extending the model to mitigate the small prediction discontinuities remaining in the current model. It would be interesting to investigate other types of hierarchical structures that still lend themselves to efficient inference algorithms. Other promising directions for future work would be developing a richer representation of the relationship among inducing points at adjacent levels, adopting some of the insights behind the deep GP model (Damianou and Lawrence, 2013).

[4]http://data.gov.uk/dataset/os-terrain-50-dtm.

Acknowledgements

This research was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.B0101-16-0307, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)) and the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-R2720-16-0007) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

Thang D Bui and Richard E Turner. Tree-structured Gaussian process approximations. In *Advances in Neural Information Processing Systems*, pages 2213–2221, 2014.

Krzysztof Chalupka, Christopher KI Williams, and Iain Murray. A framework for evaluating approximation methods for Gaussian process regression. *The Journal of Machine Learning Research*, 14(1):333–350, 2013.

Andreas C Damianou and Neil D Lawrence. Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 207–215, 2013.

François Desbouvries, Jean Lecomte, and Wojciech Pieczynski. Kalman filtering in pairwise markov trees. *Signal processing*, 86(5):1049–1054, 2006.

Aníbal R Figueiras-Vidal and Miguel Lázaro-gredilla. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, pages 1087–1095, 2009.

Emily Fox and David B Dunson. Multiresolution Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 737–745, 2012.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

Zoltan Kato, Marc Berthod, and Josiane Zerubia. A hierarchical markov random field model and multitemperature annealing for parallel image classification. *Graphical models and image processing*, 58(1): 18–37, 1996.

Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Arman Melkumyan and Fabio Ramos. Multi-kernel Gaussian processes. In *International Joint Conference on Artificial Intelligence*, volume 22, 2011.

David Moore and Stuart J Russell. Gaussian process random fields. In *Advances in Neural Information Processing Systems*, pages 3339–3347, 2015.

Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local Gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

Chiwoo Park, Jianhua Z Huang, and Yu Ding. Domain decomposition approach for fast Gaussian process regression of large spatial data sets. *The Journal of Machine Learning Research*, 12:1697–1728, 2011.

Sunho Park and Seungjin Choi. Hierarchical Gaussian process regression. In *Asian Conference on Machine Learning*, pages 95–110, 2010.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. 2006.

Simo Sarkka, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *International Conference on Artificial Intelligence and Statistics*, 2003.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2005.

Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In *International Conference on Artificial Intelligence and Statistics*, pages 524–531, 2007.

Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

Volker Tresp. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

Martin J Wainwright, Erik B Sudderth, and Alan S Willsky. Tree-based modeling and estimation of

Gaussian processes on graphs with cycles. In *Advances in Neural Information Processing Systems*, pages 661–667, 2001.

Christian Walder, Kwang In Kim, and Bernhard Schölkopf. Sparse multiscale Gaussian process regression. In *International conference on Machine learning*, pages 1112–1119. ACM, 2008.