Parallel Gaussian Process Regression with Low-Rank Covariance Matrix Approximations

Jie Chen[†]**, Nannan Cao**[†]**, Kian Hsiang Low**[†]**, Ruofei Ouyang**[†]**, Colin Keng-Yan Tan**[†]**, and Patrick Jaillet**[§]
Department of Computer Science, National University of Singapore, Republic of Singapore[†]
Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, USA[§]

Abstract

Gaussian processes (GP) are Bayesian non-parametric models that are widely used for probabilistic regression. Unfortunately, it cannot scale well with large data nor perform real-time predictions due to its cubic time cost in the data size. This paper presents two parallel GP regression methods that exploit low-rank covariance matrix approximations for distributing the computational load among parallel machines to achieve time efficiency and scalability. We theoretically guarantee the predictive performances of our proposed parallel GPs to be equivalent to that of some centralized approximate GP regression methods: The computation of their centralized counterparts can be distributed among parallel machines, hence achieving greater time efficiency and scalability. We analytically compare the properties of our parallel GPs such as time, space, and communication complexity. Empirical evaluation on two real-world datasets in a cluster of 20 computing nodes shows that our parallel GPs are significantly more time-efficient and scalable than their centralized counterparts and exact/full GP while achieving predictive performances comparable to full GP.

1 Introduction

Gaussian processes (GP) are Bayesian non-parametric models for performing nonlinear regression, which offer an important advantage of providing fully probabilistic predictive distributions with formal measures of the uncertainty of the predictions. The key limitation hindering the practical use of GP for large data is the high computational cost: It incurs cubic time and quadratic memory in the size of the data. To reduce the computational cost, two classes of approximate GP regression methods have been proposed: (a) Low-rank covariance matrix approximation methods (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005; Williams and Seeger, 2000) are especially suitable for modeling smoothly-varying func-

tions with high correlation (i.e., long length-scales) and they utilize all the data for predictions like the exact/full GP; and (b) localized regression methods (e.g., local GPs (Das and Srivastava, 2010; Choudhury *et al.*, 2002; Park *et al.*, 2011) and compactly supported covariance functions (Furrer *et al.*, 2006)) are capable of modeling highly-varying functions with low correlation (i.e., short length-scales) but they use only local data for predictions, hence predicting poorly in input regions with sparse data. Recent approximate GP regression methods of Snelson (2007) and Vanhatalo and Vehtari (2008) have attempted to combine the best of both worlds.

Despite these various efforts to scale up GP, it remains computationally impractical for performing real-time predictions necessary in many time-critical applications and decision support systems (e.g., ocean sensing (Cao *et al.*, 2013; Dolan *et al.*, 2009; Low *et al.*, 2007, 2011, 2012; Podnar *et al.*, 2010), traffic monitoring (Chen *et al.*, 2012; Yu *et al.*, 2012), geographical information systems) that need to process and analyze huge quantities of data collected over short time durations (e.g., in astronomy, internet traffic, meteorology, surveillance). To resolve this, the work in this paper considers exploiting clusters of parallel machines to achieve efficient and scalable predictions in real time. Such an idea of scaling up machine learning techniques (e.g., clustering, support vector machines, graphical models) has recently attracted widespread interest in the machine learning community (Bekkerman *et al.*, 2011). For the case of Gaussian process regression, the local GPs method (Das and Srivastava, 2010; Choudhury *et al.*, 2002) appears most straightforward to be "embarrassingly" parallelized but they suffer from discontinuities in predictions on the boundaries of different local GPs. The work of Park *et al.* (2011) rectifies this problem by imposing continuity constraints along the boundaries in a centralized manner. But, its use is restricted strictly to data with 1- and 2-dimensional input features.

This paper presents two parallel GP regression methods (Sections 3 and 4) that, in particular, exploit low-rank covariance matrix approximations for distributing the compu-

tational load among parallel machines to achieve time efficiency and scalability. Different from the above-mentioned parallel local GPs method, our proposed parallel GPs do not suffer from boundary effects, work with multi-dimensional input features, and exploit all the data for predictions but do not incur the cubic time cost of the full/exact GP. The specific contributions of our work include:

- Theoretically guaranteeing the predictive performances of our parallel GPs (i.e., *parallel partially independent conditional* (*p*PIC) and *parallel incomplete Cholesky factorization* (*p*ICF)-based approximations of GP regression model) to be equivalent to that of some centralized approaches to approximate GP regression (Sections 3 and 4). An important practical implication of these results is that the computation of their centralized counterparts can be distributed among a cluster of parallel machines, hence achieving greater time efficiency and scalability. Furthermore, our parallel GPs inherit an advantage of their centralized counterparts in providing a parameter (i.e., size of support set for *p*PIC and reduced rank for *p*ICF-based GP) to be adjusted in order to trade off between predictive performance and time efficiency;
- Analytically comparing the properties of our parallel GPs such as time, space, and communication complexity, capability of online learning, and practical implications of the structural assumptions (Section 5);
- Implementing our parallel GPs using the *message passing interface* (MPI) framework to run in a cluster of 20 computing nodes and empirically evaluating their predictive performances, time efficiency, scalability, and speedups on two real-world datasets (Section 6).

2 Gaussian Process Regression

The *Gaussian process* (GP) can be used to perform probabilistic regression as follows: Let $\mathcal{X}$ be a set representing the input domain such that each input $x \in \mathcal{X}$ denotes a d-dimensional feature vector and is associated with a realized output value y_x (random output variable Y_x) if it is observed (unobserved). Let $\{Y_x\}_{x \in \mathcal{X}}$ denote a GP, that is, every finite subset of $\{Y_x\}_{x \in \mathcal{X}}$ follows a multivariate Gaussian distribution (Rasmussen and Williams, 2006). Then, the GP is fully specified by its *prior* mean $\mu_x \triangleq \mathbb{E}[Y_x]$ and covariance $\sigma_{xx'} \triangleq \text{cov}[Y_x, Y_{x'}]$ for all $x, x' \in \mathcal{X}$.

Given that a column vector $y_\mathcal{D}$ of realized outputs is observed for some set $\mathcal{D} \subset \mathcal{X}$ of inputs, the GP can exploit this data $(\mathcal{D}, y_\mathcal{D})$ to provide predictions of the unobserved outputs for any set $\mathcal{U} \subseteq \mathcal{X} \setminus \mathcal{D}$ of inputs and their corresponding predictive uncertainties using the following Gaussian *posterior* mean vector and covariance matrix, respectively:

$$\mu_{\mathcal{U}|\mathcal{D}} \triangleq \mu_\mathcal{U} + \Sigma_{\mathcal{U}\mathcal{D}}\Sigma_{\mathcal{D}\mathcal{D}}^{-1}(y_\mathcal{D} - \mu_\mathcal{D}) \tag{1}$$

$$\Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}} \triangleq \Sigma_{\mathcal{U}\mathcal{U}} - \Sigma_{\mathcal{U}\mathcal{D}}\Sigma_{\mathcal{D}\mathcal{D}}^{-1}\Sigma_{\mathcal{D}\mathcal{U}} \tag{2}$$

where $\mu_\mathcal{U}$ ($\mu_\mathcal{D}$) is a column vector with mean components μ_x for all $x \in \mathcal{U}$ ($x \in \mathcal{D}$), $\Sigma_{\mathcal{U}\mathcal{D}}$ ($\Sigma_{\mathcal{D}\mathcal{D}}$) is a covariance matrix with covariance components $\sigma_{xx'}$ for all $x \in \mathcal{U}, x' \in \mathcal{D}$ ($x, x' \in \mathcal{D}$), and $\Sigma_{\mathcal{D}\mathcal{U}}$ is the transpose of $\Sigma_{\mathcal{U}\mathcal{D}}$. The uncertainty of predicting the unobserved outputs can be measured using the trace of $\Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}$ (2) (i.e., sum of posterior variances $\Sigma_{xx|\mathcal{D}}$ over all $x \in \mathcal{U}$), which is independent of the realized outputs $y_\mathcal{D}$.

3 Parallel Gaussian Process Regression using Support Set

The centralized approach to exact/full GP regression described in Section 2, which we call the *full Gaussian process* (FGP), unfortunately cannot scale well and be performed in real time due to its cubic time complexity in the size $|\mathcal{D}|$ of the data. In this section, we will present a class of parallel Gaussian processes (*p*PITC and *p*PIC) that distributes the computational load among parallel machines to achieve efficient and scalable approximate GP regression by exploiting the notion of a support set.

The *parallel partially independent training conditional* (*p*PITC) approximation of FGP model is adapted from our previous work on decentralized data fusion (Chen *et al.*, 2012) for sampling environmental phenomena with mobile sensors. But, the latter does not address the practical implementation issues of parallelization on a cluster of machines nor demonstrate scalability with large data. So, we present *p*PITC here under the setting of parallel machines and then show how its shortcomings can be overcome by extending it to *p*PIC. The key idea of *p*PITC is as follows: After distributing the data evenly among M machines (Step 1), each machine encapsulates its local data, based on a common prior support set $\mathcal{S} \subset X$ where $|\mathcal{S}| \ll |\mathcal{D}|$, into a local summary that is communicated to the master[1] (Step 2). The master assimilates the local summaries into a global summary (Step 3), which is then sent back to the M machines to be used for predictions distributed among them (Step 4). These steps are detailed below:

Step 1: Distribute data among M machines.

The data $(\mathcal{D}, y_\mathcal{D})$ is partitioned evenly into M blocks, each of which is assigned to a machine, as defined below:

Definition 1 (Local Data) *The local data of machine m is defined as a tuple $(\mathcal{D}_m, y_{\mathcal{D}_m})$ where $\mathcal{D}_m \subseteq \mathcal{D}$, $\mathcal{D}_m \bigcap \mathcal{D}_i = \emptyset$ and $|\mathcal{D}_m| = |\mathcal{D}_i| = |\mathcal{D}|/M$ for $i \neq m$.*

Step 2: Each machine constructs and sends local summary to master.

Definition 2 (Local Summary) *Given a common support set $\mathcal{S} \subset \mathcal{X}$ known to all M machines and the local data $(\mathcal{D}_m, y_{\mathcal{D}_m})$, the local summary of machine m is defined as a tuple $(\dot{y}_\mathcal{S}^m, \dot{\Sigma}_{\mathcal{S}\mathcal{S}}^m)$ where*

$$\dot{y}_\mathcal{B}^m \triangleq \Sigma_{\mathcal{B}\mathcal{D}_m}\Sigma_{\mathcal{D}_m\mathcal{D}_m|\mathcal{S}}^{-1}(y_{\mathcal{D}_m} - \mu_{\mathcal{D}_m}) \tag{3}$$

---

[1]One of the M machines can be assigned to be the master.

$$\dot{\Sigma}_{\mathcal{B}\mathcal{B}'}^m \triangleq \Sigma_{\mathcal{B}\mathcal{D}_m}\Sigma_{\mathcal{D}_m\mathcal{D}_m|\mathcal{S}}^{-1}\Sigma_{\mathcal{D}_m\mathcal{B}'} \tag{4}$$

such that $\Sigma_{\mathcal{D}_m\mathcal{D}_m|\mathcal{S}}$ is defined in a similar manner as (2) and $\mathcal{B}, \mathcal{B}' \subset \mathcal{X}$.

Remark. Since the local summary is independent of the outputs $y_{\mathcal{S}}$, they need not be observed. So, the support set $\mathcal{S}$ does not have to be a subset of $\mathcal{D}$ and can be selected prior to data collection. Predictive performances of pPITC and pPIC are sensitive to the selection of $\mathcal{S}$. An informative support set $\mathcal{S}$ can be selected from domain $\mathcal{X}$ using an iterative greedy active selection procedure (Krause *et al.*, 2008; Lawrence *et al.*, 2003; Seeger and Williams, 2003) prior to observing data. For example, the differential entropy score criterion (Lawrence *et al.*, 2003) can be used to greedily select an input $x \in \mathcal{X} \setminus \mathcal{S}$ with the largest posterior variance $\Sigma_{xx|\mathcal{S}}$ (2) to be included in $\mathcal{S}$ in each iteration.

STEP 3: MASTER CONSTRUCTS AND SENDS GLOBAL SUMMARY TO M MACHINES.

Definition 3 (Global Summary) *Given a common support set $\mathcal{S} \subset \mathcal{X}$ known to all M machines and the local summary $(\dot{y}_{\mathcal{S}}^m, \dot{\Sigma}_{\mathcal{S}\mathcal{S}}^m)$ of every machine $m = 1, \ldots, M$, the global summary is defined as a tuple $(\ddot{y}_{\mathcal{S}}, \ddot{\Sigma}_{\mathcal{S}\mathcal{S}})$ where*

$$\ddot{y}_{\mathcal{S}} \triangleq \sum_{m=1}^M \dot{y}_{\mathcal{S}}^m \tag{5}$$

$$\ddot{\Sigma}_{\mathcal{S}\mathcal{S}} \triangleq \Sigma_{\mathcal{S}\mathcal{S}} + \sum_{m=1}^M \dot{\Sigma}_{\mathcal{S}\mathcal{S}}^m . \tag{6}$$

STEP 4: DISTRIBUTE PREDICTIONS AMONG M MACHINES.

To predict the unobserved outputs for any set $\mathcal{U}$ of inputs, $\mathcal{U}$ is partitioned evenly into disjoint subsets $\mathcal{U}_1, \ldots, \mathcal{U}_M$ to be assigned to the respective machines $1, \ldots, M$. So, $|\mathcal{U}_m| = |\mathcal{U}|/M$ for $m = 1, \ldots, M$.

Definition 4 (pPITC) *Given a common support set $\mathcal{S} \subset \mathcal{X}$ known to all M machines and the global summary $(\ddot{y}_{\mathcal{S}}, \ddot{\Sigma}_{\mathcal{S}\mathcal{S}})$, each machine m computes a predictive Gaussian distribution $\mathcal{N}(\widehat{\mu}_{\mathcal{U}_m}, \widehat{\Sigma}_{\mathcal{U}_m\mathcal{U}_m})$ of the unobserved outputs for the set $\mathcal{U}_m$ of inputs where*

$$\widehat{\mu}_{\mathcal{U}_m} \triangleq \mu_{\mathcal{U}_m} + \Sigma_{\mathcal{U}_m\mathcal{S}}\ddot{\Sigma}_{\mathcal{S}\mathcal{S}}^{-1}\ddot{y}_{\mathcal{S}} \tag{7}$$

$$\widehat{\Sigma}_{\mathcal{U}_m\mathcal{U}_m} \triangleq \Sigma_{\mathcal{U}_m\mathcal{U}_m} - \Sigma_{\mathcal{U}_m\mathcal{S}}\left(\Sigma_{\mathcal{S}\mathcal{S}}^{-1} - \ddot{\Sigma}_{\mathcal{S}\mathcal{S}}^{-1}\right)\Sigma_{\mathcal{S}\mathcal{U}_m} . \tag{8}$$

Theorem 1 [Chen *et al.* (2012)] *Let a common support set $\mathcal{S} \subset \mathcal{X}$ be known to all M machines. Let $\mathcal{N}(\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}}, \Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}})$ be the predictive Gaussian distribution computed by the centralized partially independent training conditional (PITC) approximation of FGP model (Quiñonero-Candela and Rasmussen, 2005) where*

$$\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}} \triangleq \mu_{\mathcal{U}} + \Gamma_{\mathcal{U}\mathcal{D}}\left(\Gamma_{\mathcal{D}\mathcal{D}} + \Lambda\right)^{-1}\left(y_{\mathcal{D}} - \mu_{\mathcal{D}}\right) \tag{9}$$

$$\Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}} \triangleq \Sigma_{\mathcal{U}\mathcal{U}} - \Gamma_{\mathcal{U}\mathcal{D}}\left(\Gamma_{\mathcal{D}\mathcal{D}} + \Lambda\right)^{-1}\Gamma_{\mathcal{D}\mathcal{U}} \tag{10}$$

such that

$$\Gamma_{\mathcal{B}\mathcal{B}'} \triangleq \Sigma_{\mathcal{B}\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\Sigma_{\mathcal{S}\mathcal{B}'} \tag{11}$$

and Λ is a block-diagonal matrix constructed from the M diagonal blocks of $\Sigma_{\mathcal{D}\mathcal{D}|\mathcal{S}}$, each of which is a matrix $\Sigma_{\mathcal{D}_m\mathcal{D}_m|\mathcal{S}}$ for $m = 1, \ldots, M$ where $\mathcal{D} = \bigcup_{m=1}^M \mathcal{D}_m$. Then, $\widehat{\mu}_{\mathcal{U}} = \mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}}$ and $\widehat{\Sigma}_{\mathcal{U}\mathcal{U}} = \Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}}$.

The proof of Theorem 1 is previously reported in (Chen *et al.*, 2012) and reproduced in Appendix A of Chen *et al.* (2013) to reflect our notations.

Remark. Since PITC generalizes the Bayesian Committee Machine (BCM) of Schwaighofer and Tresp (2002), pPITC generalizes parallel BCM (Ingram and Cornford, 2010), the latter of which assumes the support set $\mathcal{S}$ to be $\mathcal{U}$ (Quiñonero-Candela and Rasmussen, 2005). As a result, parallel BCM does not scale well with large $\mathcal{U}$.

Though pPITC scales very well with large data (Table 1), it can predict poorly due to (a) loss of information caused by summarizing the realized outputs and correlation structure of the original data; and (b) sparse coverage of $\mathcal{U}$ by the support set. We propose a novel parallel Gaussian process regression method called pPIC that combines the best of both worlds, that is, the predictive power of FGP and time efficiency of pPITC. pPIC is based on the following intuition: A machine can exploit its local data to improve the predictions of the unobserved outputs that are highly correlated with its data. At the same time, pPIC can preserve the time efficiency of pPITC by exploiting its idea of encapsulating information into local and global summaries.

Definition 5 (pPIC) *Given a common support set $\mathcal{S} \subset \mathcal{X}$ known to all M machines, the global summary $(\ddot{y}_{\mathcal{S}}, \ddot{\Sigma}_{\mathcal{S}\mathcal{S}})$, the local summary $(\dot{y}_{\mathcal{S}}^m, \dot{\Sigma}_{\mathcal{S}\mathcal{S}}^m)$, and the local data $(\mathcal{D}_m, y_{\mathcal{D}_m})$, each machine m computes a predictive Gaussian distribution $\mathcal{N}(\widehat{\mu}_{\mathcal{U}_m}^+, \widehat{\Sigma}_{\mathcal{U}_m\mathcal{U}_m}^+)$ of the unobserved outputs for the set $\mathcal{U}_m$ of inputs where*

$$\widehat{\mu}_{\mathcal{U}_m}^+ \triangleq \mu_{\mathcal{U}_m} + \left(\Phi_{\mathcal{U}_m\mathcal{S}}^m\ddot{\Sigma}_{\mathcal{S}\mathcal{S}}^{-1}\ddot{y}_{\mathcal{S}} - \Sigma_{\mathcal{U}_m\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\dot{y}_{\mathcal{S}}^m\right) + \dot{y}_{\mathcal{U}_m}^m \tag{12}$$

$$\widehat{\Sigma}_{\mathcal{U}_m\mathcal{U}_m}^+ \triangleq \Sigma_{\mathcal{U}_m\mathcal{U}_m} - \left(\Phi_{\mathcal{U}_m\mathcal{S}}^m\ddot{\Sigma}_{\mathcal{S}\mathcal{S}}^{-1}\Sigma_{\mathcal{S}\mathcal{U}_m} - \Sigma_{\mathcal{U}_m\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\dot{\Sigma}_{\mathcal{S}\mathcal{U}_m}^m\right.$$
$$\left. - \Phi_{\mathcal{U}_m\mathcal{S}}^m\ddot{\Sigma}_{\mathcal{S}\mathcal{S}}^{-1}\Phi_{\mathcal{S}\mathcal{U}_m}^m\right) - \dot{\Sigma}_{\mathcal{U}_m\mathcal{U}_m}^m \tag{13}$$

such that

$$\Phi_{\mathcal{U}_m\mathcal{S}}^m \triangleq \Sigma_{\mathcal{U}_m\mathcal{S}} + \Sigma_{\mathcal{U}_m\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\dot{\Sigma}_{\mathcal{S}\mathcal{S}}^m - \dot{\Sigma}_{\mathcal{U}_m\mathcal{S}}^m \tag{14}$$

and $\Phi_{\mathcal{S}\mathcal{U}_m}^m$ is the transpose of $\Phi_{\mathcal{U}_m\mathcal{S}}^m$.

Remark 1. The predictive Gaussian mean $\widehat{\mu}_{\mathcal{U}_m}^+$ (12) and covariance $\widehat{\Sigma}_{\mathcal{U}_m\mathcal{U}_m}^+$ (13) of pPIC exploit both summary information (i.e., bracketed term) and local information (i.e., last term). In contrast, pPITC only exploits the global summary (see (7) and (8)).

Remark 2. To improve the predictive performance of pPIC, $\mathcal{D}$ and $\mathcal{U}$ should be partitioned into tuples of $(\mathcal{D}_1, \mathcal{U}_1), \ldots, (\mathcal{D}_M, \mathcal{U}_M)$ such that the outputs $y_{\mathcal{D}_m}$ and $Y_{\mathcal{U}_m}$ are as highly correlated as possible for $m = 1, \ldots, M$. To achieve this, we employ a simple parallelized clustering scheme in our experiments: Each machine m randomly selects a cluster center from its local data $\mathcal{D}_m$ and informs the other machines about its chosen cluster center. Then, each input in $\mathcal{D}_m$ and $\mathcal{U}_m$ is simply assigned to the "nearest" cluster center i and sent to the corresponding machine i while being subject to the constraints of the new D_i and U_i not exceeding $|\mathcal{D}|/M$ and $|\mathcal{U}|/M$, respectively. More sophisticated clustering schemes can be utilized at the expense of greater time and communication complexity.

Remark 3. Predictive performances of pPITC and pPIC are improved by increasing size of $\mathcal{S}$ at the expense of greater time, space, and communication complexity (Table 1).

Theorem 2 *Let a common support set $\mathcal{S} \subset \mathcal{X}$ be known to all M machines. Let $\mathcal{N}(\mu^{\mathrm{PIC}}_{\mathcal{U}|\mathcal{D}}, \Sigma^{\mathrm{PIC}}_{\mathcal{U}\mathcal{U}|\mathcal{D}})$ be the predictive Gaussian distribution computed by the centralized partially independent conditional (PIC) approximation of FGP model (Snelson, 2007) where*

$$\mu^{\mathrm{PIC}}_{\mathcal{U}|\mathcal{D}} \triangleq \mu_{\mathcal{U}} + \widetilde{\Gamma}_{\mathcal{U}\mathcal{D}} \left(\Gamma_{\mathcal{D}\mathcal{D}} + \Lambda\right)^{-1} \left(y_{\mathcal{D}} - \mu_{\mathcal{D}}\right) \quad (15)$$

$$\Sigma^{\mathrm{PIC}}_{\mathcal{U}\mathcal{U}|\mathcal{D}} \triangleq \Sigma_{\mathcal{U}\mathcal{U}} - \widetilde{\Gamma}_{\mathcal{U}\mathcal{D}} \left(\Gamma_{\mathcal{D}\mathcal{D}} + \Lambda\right)^{-1} \widetilde{\Gamma}_{\mathcal{D}\mathcal{U}} \quad (16)$$

and $\widetilde{\Gamma}_{\mathcal{D}\mathcal{U}}$ is the transpose of $\widetilde{\Gamma}_{\mathcal{U}\mathcal{D}}$ such that

$$\widetilde{\Gamma}_{\mathcal{U}\mathcal{D}} \triangleq \left(\widetilde{\Gamma}_{\mathcal{U}_i \mathcal{D}_m}\right)_{i,m=1,\ldots,M} \quad (17)$$

$$\widetilde{\Gamma}_{\mathcal{U}_i \mathcal{D}_m} \triangleq \begin{cases} \Sigma_{\mathcal{U}_i \mathcal{D}_m} & \text{if } i = m, \\ \Gamma_{\mathcal{U}_i \mathcal{D}_m} & \text{otherwise.} \end{cases} \quad (18)$$

Then, $\widehat{\mu}^+_{\mathcal{U}} = \mu^{\mathrm{PIC}}_{\mathcal{U}|\mathcal{D}}$ and $\widehat{\Sigma}^+_{\mathcal{U}\mathcal{U}} = \Sigma^{\mathrm{PIC}}_{\mathcal{U}\mathcal{U}|\mathcal{D}}$.

Its proof is given in Appendix B of Chen *et al.* (2013).

Remark 1. The equivalence results of Theorems 1 and 2 imply that the computational load of the centralized PITC and PIC approximations of FGP can be distributed among M parallel machines, hence improving the time efficiency and scalability of approximate GP regression (Table 1).

Remark 2. The equivalence results also shed some light on the underlying properties of pPITC and pPIC based on the structural assumptions of PITC and PIC, respectively: pPITC assumes that $Y_{\mathcal{D}_1}, \ldots, Y_{\mathcal{D}_M}, Y_{\mathcal{U}_1}, \ldots, Y_{\mathcal{U}_M}$ are conditionally independent given $Y_{\mathcal{S}}$. In contrast, pPIC can predict the unobserved outputs $Y_{\mathcal{U}}$ better since it imposes a less restrictive assumption of conditional independence between $Y_{\mathcal{D}_1 \bigcup \mathcal{U}_1}, \ldots, Y_{\mathcal{D}_M \bigcup \mathcal{U}_M}$ given $Y_{\mathcal{S}}$. This assumption further supports an earlier remark just before Theorem 2 on clustering inputs $\mathcal{D}_m$ and $\mathcal{U}_m$ whose corresponding outputs are highly correlated for improving predictive performance of pPIC. Experimental results on two real-world datasets

(Section 6) show that pPIC achieves predictive accuracy comparable to FGP and significantly better than pPITC, thus justifying the practicality of such an assumption.

4 Parallel Gaussian Process Regression using Incomplete Cholesky Factorization

In this section, we will present another parallel Gaussian process called pICF-based GP that distributes the computational load among parallel machines to achieve efficient and scalable approximate GP regression by exploiting incomplete Cholesky factorization (ICF). A fundamental step of pICF-based GP is to use ICF to approximate the covariance matrix $\Sigma_{\mathcal{D}\mathcal{D}}$ in (1) and (2) of FGP by a low-rank symmetric positive semidefinite matrix: $\Sigma_{\mathcal{D}\mathcal{D}} \approx F^\top F + \sigma_n^2 I$ where $F \in \mathbb{R}^{R \times |\mathcal{D}|}$ denotes the upper triangular incomplete Cholesky factor and $R \ll |\mathcal{D}|$ is the reduced rank. The steps of performing pICF-based GP are as follows:

STEP 1: DISTRIBUTE DATA AMONG M MACHINES.

This step is the same as that of pPITC and pPIC in Section 3.

STEP 2: RUN PARALLEL ICF TO PRODUCE INCOMPLETE CHOLESKY FACTOR AND DISTRIBUTE ITS STORAGE.

ICF can in fact be parallelized: Instead of using a column-based parallel ICF (Golub and Van Loan, 1996), our proposed pICF-based GP employs a row-based parallel ICF, the latter of which incurs lower time, space, and communication complexity. Interested readers are referred to (Chang *et al.*, 2007) for a detailed implementation of the row-based parallel ICF, which is beyond the scope of this paper. More importantly, it produces an upper triangular incomplete Cholesky factor $F \triangleq (F_1 \cdots F_M)$ and each submatrix $F_m \in \mathbb{R}^{R \times |\mathcal{D}_m|}$ is stored distributedly on machine m for $m = 1, \ldots, M$.

STEP 3: EACH MACHINE CONSTRUCTS AND SENDS LOCAL SUMMARY TO MASTER.

Definition 6 (Local Summary) *Given the local data $(\mathcal{D}_m, y_{\mathcal{D}_m})$ and incomplete Cholesky factor F_m, the local summary of machine m is defined as a tuple $(\dot{y}_m, \dot{\Sigma}_m, \Phi_m)$ where*

$$\dot{y}_m \triangleq F_m(y_{\mathcal{D}_m} - \mu_{\mathcal{D}_m}) \quad (19)$$

$$\dot{\Sigma}_m \triangleq F_m \Sigma_{\mathcal{D}_m \mathcal{U}} \quad (20)$$

$$\Phi_m \triangleq F_m F_m^\top . \quad (21)$$

STEP 4: MASTER CONSTRUCTS AND SENDS GLOBAL SUMMARY TO M MACHINES.

Definition 7 (Global Summary) *Given the local summary $(\dot{y}_m, \dot{\Sigma}_m, \Phi_m)$ of every machine $m = 1, \ldots, M$, the*

global summary is defined as a tuple $(\ddot{y}, \ddot{\Sigma})$ *where*

$$\ddot{y} \triangleq \Phi^{-1} \sum_{m=1}^{M} \dot{y}_m \qquad (22)$$

$$\ddot{\Sigma} \triangleq \Phi^{-1} \sum_{m=1}^{M} \dot{\Sigma}_m \qquad (23)$$

such that $\Phi \triangleq I + \sigma_n^{-2} \sum_{m=1}^{M} \Phi_m$.

Remark. If $|\mathcal{U}|$ is large, the computation of (23) can be parallelized by partitioning $\mathcal{U}$: Let $\dot{\Sigma}_m \triangleq (\dot{\Sigma}_m^1 \cdots \dot{\Sigma}_m^M)$ where $\dot{\Sigma}_m^i \triangleq F_m \Sigma_{\mathcal{D}_m \mathcal{U}_i}$ is defined in a similar way as (20) and $|\mathcal{U}|_i = |\mathcal{U}|/M$. So, in Step 3, instead of sending $\dot{\Sigma}_m$ to the master, each machine m sends $\dot{\Sigma}_m^i$ to machine i for $i = 1, \ldots, M$. Then, each machine i computes and sends $\ddot{\Sigma}_i \triangleq \Phi^{-1} \sum_{m=1}^{M} \dot{\Sigma}_m^i$ to every other machine to obtain $\ddot{\Sigma} = (\ddot{\Sigma}_1 \cdots \ddot{\Sigma}_M)$.

STEP 5: EACH MACHINE CONSTRUCTS AND SENDS PREDICTIVE COMPONENT TO MASTER.

Definition 8 (Predictive Component) *Given the local data* $(\mathcal{D}_m, y_{\mathcal{D}_m})$, *a component* $\dot{\Sigma}_m$ *of the local summary, and the global summary* $(\ddot{y}, \ddot{\Sigma})$, *the predictive component of machine* m *is defined as a tuple* $(\widetilde{\mu}_{\mathcal{U}}^m, \widetilde{\Sigma}_{\mathcal{U}\mathcal{U}}^m)$ *where*

$$\widetilde{\mu}_{\mathcal{U}}^m \triangleq \sigma_n^{-2} \Sigma_{\mathcal{U}\mathcal{D}_m}(y_{\mathcal{D}_m} - \mu_{\mathcal{D}_m}) - \sigma_n^{-4} \dot{\Sigma}_m^{\top} \ddot{y} \qquad (24)$$

$$\widetilde{\Sigma}_{\mathcal{U}\mathcal{U}}^m \triangleq \sigma_n^{-2} \Sigma_{\mathcal{U}\mathcal{D}_m} \Sigma_{\mathcal{D}_m \mathcal{U}} - \sigma_n^{-4} \dot{\Sigma}_m^{\top} \ddot{\Sigma}. \qquad (25)$$

STEP 6: MASTER PERFORMS PREDICTIONS.

Definition 9 (pICF-based GP) *Given the predictive component* $(\widetilde{\mu}_{\mathcal{U}}^m, \widetilde{\Sigma}_{\mathcal{U}\mathcal{U}}^m)$ *of every machine* $m = 1, \ldots, M$, *the master computes a predictive Gaussian distribution* $\mathcal{N}(\widetilde{\mu}_{\mathcal{U}}, \widetilde{\Sigma}_{\mathcal{U}\mathcal{U}})$ *of the unobserved outputs for any set* $\mathcal{U}$ *of inputs where*

$$\widetilde{\mu}_{\mathcal{U}} \triangleq \mu_{\mathcal{U}} + \sum_{m=1}^{M} \widetilde{\mu}_{\mathcal{U}}^m \qquad (26)$$

$$\widetilde{\Sigma}_{\mathcal{U}\mathcal{U}} \triangleq \Sigma_{\mathcal{U}\mathcal{U}} - \sum_{m=1}^{M} \widetilde{\Sigma}_{\mathcal{U}\mathcal{U}}^m. \qquad (27)$$

Remark. Predictive performance of pICF-based GP can be improved by increasing rank R at the expense of greater time, space, and communication complexity (Table 1).

Theorem 3 *Let* $\mathcal{N}(\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}}, \Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}})$ *be the predictive Gaussian distribution computed by the centralized ICF approximation of FGP model where*

$$\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}} \triangleq \mu_{\mathcal{U}} + \Sigma_{\mathcal{U}\mathcal{D}}(F^{\top}F + \sigma_n^2 I)^{-1}(y_{\mathcal{D}} - \mu_{\mathcal{D}}) \qquad (28)$$

$$\Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}} \triangleq \Sigma_{\mathcal{U}\mathcal{U}} - \Sigma_{\mathcal{U}\mathcal{D}}(F^{\top}F + \sigma_n^2 I)^{-1}\Sigma_{\mathcal{D}\mathcal{U}}. \qquad (29)$$

Then, $\widetilde{\mu}_{\mathcal{U}} = \mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}}$ *and* $\widetilde{\Sigma}_{\mathcal{U}\mathcal{U}} = \Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}}$.

Its proof is given in Appendix C of Chen *et al.* (2013).

Remark 1. The equivalence result of Theorem 3 implies that the computational load of the centralized ICF approximation of FGP can be distributed among the M parallel machines, hence improving the time efficiency and scalability of approximate GP regression (Table 1).

Remark 2. By approximating the covariance matrix $\Sigma_{\mathcal{D}\mathcal{D}}$ in (1) and (2) of FGP with $F^{\top}F + \sigma_n^2 I$, $\widetilde{\Sigma}_{\mathcal{U}\mathcal{U}} = \Sigma_{\mathcal{U}\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}}$ is not guaranteed to be positive semidefinite, hence rendering such a measure of predictive uncertainty not very useful. However, it is observed in our experiments (Section 6) that this problem can be alleviated by choosing a sufficiently large rank R.

5 Analytical Comparison

This section compares and contrasts the properties of the proposed parallel GPs analytically.

5.1 Time, Space, and Communication Complexity

Table 1 analytically compares the time, space, and communication complexity between pPITC, pPIC, pICF-based GP, PITC, PIC, ICF-based GP, and FGP based on the following assumptions: (a) These respective methods compute the predictive means (i.e., $\widehat{\mu}_{\mathcal{U}}$ (7), $\widehat{\mu}_{\mathcal{U}}^+$ (12), $\widetilde{\mu}_{\mathcal{U}}$ (26), $\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{PITC}}$ (9), $\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{PIC}}$ (15), $\mu_{\mathcal{U}|\mathcal{D}}^{\mathrm{ICF}}$ (28), and $\mu_{\mathcal{U}|\mathcal{D}}$ (1)) and their corresponding predictive variances (i.e., $\widehat{\Sigma}_{xx}$ (8), $\widehat{\Sigma}_{xx}^+$ (13), $\widetilde{\Sigma}_{xx}$ (27), $\Sigma_{xx|\mathcal{D}}^{\mathrm{PITC}}$ (10), $\Sigma_{xx|\mathcal{D}}^{\mathrm{PIC}}$ (16), $\Sigma_{xx|\mathcal{D}}^{\mathrm{ICF}}$ (29), and $\Sigma_{xx|\mathcal{D}}$ (2) for all $x \in \mathcal{U}$); (b) $|\mathcal{U}| < |\mathcal{D}|$ and recall $|\mathcal{S}|, R \ll |\mathcal{D}|$; (c) the data is already distributed among M parallel machines for pPITC, pPIC, and pICF-based GP; and (d) for MPI, a broadcast operation in the communication network of M machines incurs $\mathcal{O}(\log M)$ messages (Pjesivac-Grbovic *et al.*, 2007). The observations are as follows:

(a) Our pPITC, pPIC, and pICF-based GP improve the scalability of their centralized counterparts (respectively, PITC, PIC, and ICF-based GP) in the size $|\mathcal{D}|$ of data by distributing their computational loads among the M parallel machines.

(b) The speedups of pPITC, pPIC, and pICF-based GP over their centralized counterparts deviate further from ideal speedup with increasing number M of machines due to their additional $\mathcal{O}(|\mathcal{S}|^2 M)$ or $\mathcal{O}(R^2 M)$ time.

(c) The speedups of pPITC and pPIC grow with increasing size $|\mathcal{D}|$ of data because, unlike the additional $\mathcal{O}(|\mathcal{S}|^2|\mathcal{D}|)$ time of PITC and PIC that increase with more data, they do not have corresponding $\mathcal{O}(|\mathcal{S}|^2|\mathcal{D}|/M)$ terms.

(d) Our pPIC incurs additional $\mathcal{O}(|\mathcal{D}|)$ time and $\mathcal{O}((|\mathcal{D}|/M)\log M)$-sized messages over pPITC

Table 1: Comparison of time, space, and communication complexity between pPITC, pPIC, pICF-based GP, PITC, PIC, ICF-based GP, and FGP. Note that PITC, PIC, and ICF-based GP are, respectively, the centralized counterparts of pPITC, pPIC, and pICF-based GP, as proven in Theorems 1, 2, and 3.

GP	Time complexity	Space complexity	Communication complexity
pPITC	$\mathcal{O}\left(\|\mathcal{S}\|^2 \left(\|\mathcal{S}\| + M + \frac{\|\mathcal{U}\|}{M} \right) + \left(\frac{\|\mathcal{D}\|}{M} \right)^3 \right)$	$\mathcal{O}\left(\|\mathcal{S}\|^2 + \left(\frac{\|\mathcal{D}\|}{M} \right)^2 \right)$	$\mathcal{O}\left(\|\mathcal{S}\|^2 \log M \right)$
pPIC	$\mathcal{O}\left(\|\mathcal{S}\|^2 \left(\|\mathcal{S}\| + M + \frac{\|\mathcal{U}\|}{M} \right) + \left(\frac{\|\mathcal{D}\|}{M} \right)^3 + \|\mathcal{D}\| \right)$	$\mathcal{O}\left(\|\mathcal{S}\|^2 + \left(\frac{\|\mathcal{D}\|}{M} \right)^2 \right)$	$\mathcal{O}\left(\left(\|\mathcal{S}\|^2 + \frac{\|\mathcal{D}\|}{M} \right) \log M \right)$
pICF-based	$\mathcal{O}\left(R^2 \left(R + M + \frac{\|\mathcal{D}\|}{M} \right) + R\|\mathcal{U}\| \left(M + \frac{\|\mathcal{D}\|}{M} \right) \right)$	$\mathcal{O}\left(R^2 + R\frac{\|\mathcal{D}\|}{M} \right)$	$\mathcal{O}\left((R^2 + R\|\mathcal{U}\|) \log M \right)$
PITC	$\mathcal{O}\left(\|\mathcal{S}\|^2\|\mathcal{D}\| + \|\mathcal{D}\| \left(\frac{\|\mathcal{D}\|}{M} \right)^2 \right)$	$\mathcal{O}\left(\|\mathcal{S}\|^2 + \left(\frac{\|\mathcal{D}\|}{M} \right)^2 \right)$	–
PIC	$\mathcal{O}\left(\|\mathcal{S}\|^2\|\mathcal{D}\| + \|\mathcal{D}\| \left(\frac{\|\mathcal{D}\|}{M} \right)^2 + M\|\mathcal{D}\| \right)$	$\mathcal{O}\left(\|\mathcal{S}\|^2 + \left(\frac{\|\mathcal{D}\|}{M} \right)^2 \right)$	–
ICF-based	$\mathcal{O}(R^2\|\mathcal{D}\| + R\|\mathcal{U}\|\|\mathcal{D}\|)$	$\mathcal{O}(R\|\mathcal{D}\|)$	–
FGP	$\mathcal{O}\left(\|\mathcal{D}\|^3 \right)$	$\mathcal{O}\left(\|\mathcal{D}\|^2 \right)$	–

due to its parallelized clustering (see Remark 2 after Definition 5).

(e) Keeping the other variables fixed, an increasing number M of machines reduces the time and space complexity of pPITC and pPIC at a faster rate than pICF-based GP while increasing size $|\mathcal{D}|$ of data raises the time and space complexity of pICF-based GP at a slower rate than pPITC and pPIC.

(f) Our pICF-based GP distributes the memory requirement of ICF-based GP among the M parallel machines.

(g) The communication complexity of pICF-based GP depends on the number $|\mathcal{U}|$ of predictions whereas that of pPITC and pPIC are independent of it.

5.2 Online/Incremental Learning

Supposing new data $(\mathcal{D}', y_{\mathcal{D}'})$ becomes available, pPITC and pPIC do not have to run Steps 1 to 4 (Section 3) on the entire data $(\mathcal{D} \bigcup \mathcal{D}', y_{\mathcal{D} \bigcup \mathcal{D}'})$. The local and global summaries of the old data $(\mathcal{D}, y_{\mathcal{D}})$ can in fact be reused and assimilated with that of the new data, thus saving the need of recomputing the computationally expensive matrix inverses in (3) and (4) for the old data. The exact mathematical details are omitted due to lack of space. As a result, the time complexity of pPITC and pPIC can be greatly reduced in situations where new data is expected to stream in at regular intervals. In contrast, pICF-based GP does not seem to share this advantage.

5.3 Structural Assumptions

The above advantage of online learning for pPITC and pPIC results from their assumptions of conditional inde-

pendence (see Remark 2 after Theorem 2) given the support set. With fewer machines, such an assumption is violated less, thus potentially improving their predictive performances. In contrast, the predictive performance of pICF-based GP is not affected by varying the number of machines. However, it suffers from a different problem: Utilizing a reduced-rank matrix approximation of $\Sigma_{\mathcal{DD}}$, its resulting predictive covariance matrix $\widetilde{\Sigma}_{\mathcal{UU}}$ is not guaranteed to be positive semidefinite (see Remark 2 after Theorem 3), thus rendering such a measure of predictive uncertainty not very useful. It is observed in our experiments (Section 6) that this problem can be alleviated by choosing a sufficiently large rank R.

6 Experiments and Discussion

This section empirically evaluates the predictive performances, time efficiency, scalability, and speedups of our proposed parallel GPs against their centralized counterparts and FGP on two real-world datasets: (a) The AIMPEAK dataset of size $|\mathcal{D}| = 41850$ contains traffic speeds (km/h) along 775 road segments of an urban road network (including highways, arterials, slip roads, etc.) during the morning peak hours (6-10:30 a.m.) on April 20, 2011. The traffic speeds are the outputs. The mean speed is 49.5 km/h and the standard deviation is 21.7 km/h. Each input (i.e., road segment) is specified by a 5-dimensional vector of features: length, number of lanes, speed limit, direction, and time. The time dimension comprises 54 five-minute time slots. This spatiotemporal traffic phenomenon is modeled using a relational GP (previously developed in (Chen *et al.*, 2012)) whose correlation structure can exploit both the road segment features and road network topology information; (b) The SARCOS dataset (Vijayakumar *et al.*, 2005) of size $|\mathcal{D}| = 48933$ pertains to an inverse dynamics problem for a

seven degrees-of-freedom SARCOS robot arm. Each input denotes a 21-dimensional vector of features: 7 joint positions, 7 joint velocities, and 7 joint accelerations. Only one of the 7 joint torques is used as the output. The mean torque is 13.7 and the standard deviation is 20.5.

Both datasets are modeled using GPs whose prior covariance $\sigma_{xx'}$ is defined by the squared exponential covariance function[2]:

$$\sigma_{xx'} \triangleq \sigma_s^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i - x_i'}{\ell_i}\right)^2\right) + \sigma_n^2 \delta_{xx'}$$

where $x_i\,(x_i')$ is the i-th component of the input feature vector $x\,(x')$, the hyperparameters $\sigma_s^2, \sigma_n^2, \ell_1, \ldots, \ell_d$ are, respectively, signal variance, noise variance, and lengthscales; and $\delta_{xx'}$ is a Kronecker delta that is 1 if $x = x'$ and 0 otherwise. The hyperparameters are learned using randomly selected data of size 10000 via maximum likelihood estimation (Rasmussen and Williams, 2006).

For each dataset, 10% of the data is randomly selected as test data for predictions (i.e., as $\mathcal{U}$). From the remaining data, training data of varying sizes $|\mathcal{D}| = 8000, 16000, 24000$, and 32000 are randomly selected. The training data are distributed among M machines based on the simple parallelized clustering scheme in Remark 2 after Definition 5. Our pPITC and pPIC are evaluated using support sets of varying sizes $|\mathcal{S}| = 256, 512, 1024$, and 2048 that are selected using differential entropy score criterion (see remark just after Definition 2). Our pICF-based GP is evaluated using varying reduced ranks R of the same values as $|\mathcal{S}|$ in the AIMPEAK domain and twice the values of $|\mathcal{S}|$ in the SARCOS domain.

Our experimental platform is a cluster of 20 computing nodes connected via gigabit links: Each node runs a Linux system with Intel® Xeon® CPU E5520 at 2.27 GHz and 20 GB memory. Our parallel GPs are tested with different number $M = 4, 8, 12, 16$, and 20 of computing nodes.

6.1 Performance Metrics

The tested GP regression methods are evaluated with four different performance metrics: (a) Root mean square error (RMSE) $\sqrt{|\mathcal{U}|^{-1}\sum_{x\in\mathcal{U}}\left(y_x - \mu_{x|\mathcal{D}}\right)^2}$; (b) mean negative log probability (MNLP) $0.5|\mathcal{U}|^{-1}\sum_{x\in\mathcal{U}}\left((y_x - \mu_{x|\mathcal{D}})^2/\Sigma_{xx|\mathcal{D}} + \log(2\pi\Sigma_{xx|\mathcal{D}})\right)$ (Rasmussen and Williams, 2006); (c) incurred time; and (d) speedup is defined as the incurred time of a sequential/centralized algorithm divided by that of its corresponding parallel algorithm. For the first two metrics, the tested methods have to plug their predictive mean and variance into $\mu_{u|\mathcal{D}}$ and $\Sigma_{uu|\mathcal{D}}$, respectively.

---

[2]For the AIMPEAK dataset, the domain of road segments is embedded into the Euclidean space using multi-dimensional scaling (Chen *et al.*, 2012) so that a squared exponential covariance function can then be applied.

6.2 Results and Analysis

In this section, we analyze the results that are obtained by averaging over 5 random instances.

6.2.1 Varying size $|\mathcal{D}|$ of data

Figs. 1a-b and 1e-f show that the predictive performances of our parallel GPs improve with more data and are comparable to that of FGP, hence justifying the practicality of their inherent structural assumptions.

From Figs. 1e-f, it can be observed that the predictive performance of pICF-based GP is very close to that of FGP when $|\mathcal{D}|$ is relatively small (i.e., $|\mathcal{D}| = 8000, 16000$). But, its performance approaches that of pPIC as $|\mathcal{D}|$ increases further because the reduced rank $R = 4096$ of pICF-based GP is not large enough (relative to $|\mathcal{D}|$) to maintain its close performance to FGP. In addition, pPIC achieves better predictive performance than pPITC since the former can exploit local information (see Remark 1 after Definition 5).

Figs. 1c and 1g indicate that our parallel GPs are significantly more time-efficient and scalable than FGP (i.e., 1-2 orders of magnitude faster) while achieving comparable predictive performance. Among the three parallel GPs, pPITC and pPIC are more time-efficient and thus more capable of meeting the real-time prediction requirement of a time-critical application/system.

Figs. 1d and 1h show that the speedups of our parallel GPs over their centralized counterparts increase with more data, which agree with observation c in Section 5.1. pPITC and pPIC achieve better speedups than pICF-based GP.

6.2.2 Varying number M of machines

Figs. 2a-b and 2e-f show that pPIC and pICF-based GP achieve predictive performance comparable to that of FGP with different number M of machines. pPIC achieves better predictive performance than pPITC due to its use of local information (see Remark 1 after Definition 5).

From Figs. 2e-f, it can be observed that as the number M of machines increases, the predictive performance of pPIC drops slightly due to smaller size of local data $\mathcal{D}_m$ assigned to each machine. In contrast, the predictive performance of pPITC improves: If the number M of machines is small as compared to the actual number of clusters in the data, then the clustering scheme (see Remark 2 after Definition 5) may assign data from different clusters to the same machine or data from the same cluster to multiple machines. Consequently, the conditional independence assumption is violated. Such an issue is mitigated by increasing the number M of machines to achieve better clustering, hence resulting in better predictive performance.

Figs. 2c and 2g show that pPIC and pICF-based GP are significantly more time-efficient than FGP (i.e., 1-2 orders
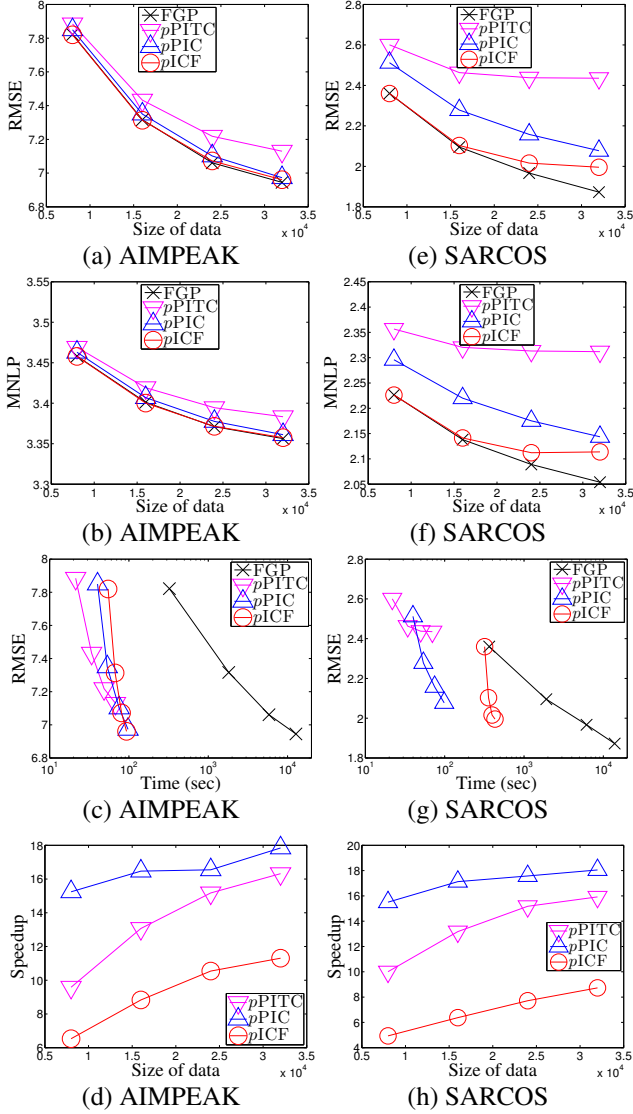
Figure 1: Performance of parallel GPs with varying data sizes $|\mathcal{D}| = 8000, 16000, 24000,$ and 32000, number $M = 20$ of machines, support set size $|\mathcal{S}| = 2048$, and reduced rank $R = 2048$ (4096) in the AIMPEAK (SARCOS) domain.
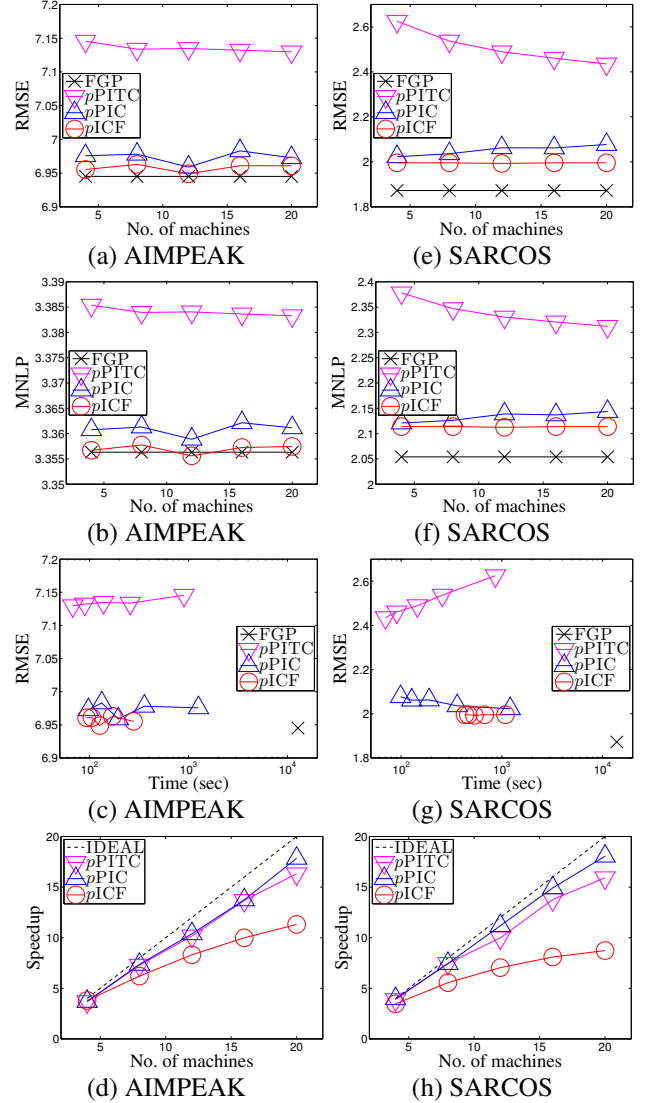

Figure 2: Performance of parallel GPs with varying number $M = 4, 8, 12, 16, 20$ of machines, data size $|\mathcal{D}| = 32000$, support set size $\mathcal{S} = 2048$, and reduced rank $R = 2048$ (4096) in the AIMPEAK (SARCOS) domain. The *ideal* speedup of a parallel algorithm is defined to be the number M of machines running it.

of magnitude faster) while achieving comparable predictive performance. This is previously explained in the analysis of their time complexity (Table 1).

Figs. 2c and 2g also reveal that as the number M of machines increases, the incurred time of pPITC and pPIC decreases at a faster rate than that of pICF-based GP, which agree with observation e in Section 5.1. Hence, we expect pPITC and pPIC to be more time-efficient than pICF-based GP when the number M of machines increases beyond 20.

Figs. 2d and 2h show that the speedups of our parallel GPs over their centralized counterparts deviate further from the ideal speedup with a greater number M of machines, which

agree with observation b in Section 5.1. The speedups of pPITC and pPIC are closer to the ideal speedup than that of pICF-based GP.

6.2.3 Varying support set size $|\mathcal{S}|$ and reduced rank R

Figs. 3a and 3e show that the predictive performance of pICF-based GP is extremely poor when the reduced rank R is not large enough (relative to $|\mathcal{D}|$), thus resulting in a poor ICF approximation of the covariance matrix $\Sigma_{\mathcal{D}\mathcal{D}}$. In addition, it can be observed that the reduced rank R of pICF-based GP needs to be much larger than the support set size $|\mathcal{S}|$ of pPITC and pPIC in order to achieve comparable

predictive performance. These results also indicate that the heuristic $R = \sqrt{|\mathcal{D}|}$, which is used by Chang *et al.* (2007) to determine the reduced rank R, fails to work well in both our datasets (e.g., $R = 1024 > \sqrt{32000} \approx 179$).

From Figs. 3b and 3f, it can be observed that pICF-based GP incurs negative MNLP for $R \leq 1024$ ($R \leq 2048$) in the AIMPEAK (SARCOS) domain. This is because pICF-based GP cannot guarantee positivity of predictive variance, as explained in Remark 2 after Theorem 3. But, it appears that when R is sufficiently large (i.e., $R = 2048$ ($R = 4096$) in the AIMPEAK (SARCOS) domain), this problem can be alleviated.

It can be observed in Figs. 3c and 3g that pPITC and pPIC are significantly more time-efficient than FGP (i.e., 2-4 orders of magnitude faster) while achieving comparable predictive performance. To ensure high predictive performance, pICF-based GP has to select a large enough rank $R = 2048$ ($R = 4096$) in the AIMPEAK (SARCOS) domain, thus making it less time-efficient than pPITC and pPIC. But, it can still incur 1-2 orders of magnitude less time than FGP. These results indicate that pPITC and pPIC are more capable than pICF-based GP of meeting the real-time prediction requirement of a time-critical application/system.

Figs. 3d and 3h show that pPITC and pPIC achieve better speedups than pICF-based GP.

6.2.4 Summary of results

pPIC and pICF-based GP are significantly more time-efficient and scalable than FGP (i.e., 1-4 orders of magnitude faster) while achieving comparable predictive performance, hence justifying the practicality of their structural assumptions. pPITC and pPIC are expected to be more time-efficient than pICF-based GP with an increasing number M of machines because their incurred time decreases at a faster rate than that of pICF-based GP. Since the predictive performances of pPITC and pPIC drop slightly (i.e., more stable) with smaller $|\mathcal{S}|$ as compared to that of pICF-based GP dropping rapidly with smaller R, pPITC and pPIC are more capable than pICF-based GP of meeting the real-time prediction requirement of time-critical applications. The speedups of our parallel GPs over their centralized counterparts improve with more data but deviate further from ideal speedup with larger number of machines.

7 Conclusion

This paper describes parallel GP regression methods called pPIC and pICF-based GP that, respectively, distribute the computational load of the centralized PIC and ICF-based GP among parallel machines to achieve greater time efficiency and scalability. Analytical and empirical results have demonstrated that our parallel GPs are significantly more time-efficient and scalable than their centralized counterparts and FGP while achieving predictive per-
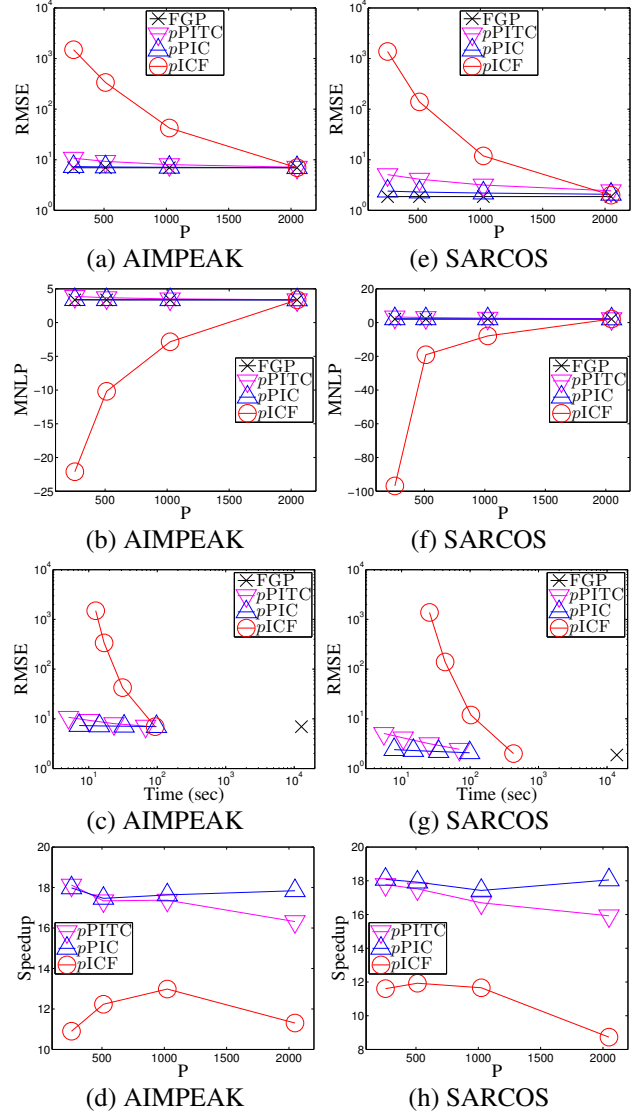

Figure 3: Performance of parallel GPs with data size $|\mathcal{D}| = 32000$, number $M = 20$ of machines, and varying parameter $P = 256, 512, 1024, 2048$ where $P = |\mathcal{S}| = R$ ($P = |\mathcal{S}| = R/2$) in the AIMPEAK (SARCOS) domain.

formance comparable to FGP. As a result, by exploiting large clusters of machines, our parallel GPs become substantially more capable of performing real-time predictions necessary in many time-critical applications/systems. We have also implemented pPITC and pPIC in the MapReduce framework for running in a Linux server with 2 Intel® Xeon® CPU E5-2670 at 2.60 GHz and 96 GB memory (i.e., 16 cores); due to shared memory, they incur slightly longer time than that in a cluster of 16 computing nodes. We plan to release the source code at http://code.google.com/p/pgpr/.

References

Bekkerman, R., Bilenko, M., and Langford, J. (2011). *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge Univ. Press, NY.

Cao, N., Low, K. H., and Dolan, J. M. (2013). Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proc. AAMAS*, pages 7–14.

Chang, E. Y., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z., and Cui, H. (2007). Parallelizing support vector machines on distributed computers. In *Proc. NIPS*.

Chen, J., Low, K. H., Tan, C. K.-Y., Oran, A., Jaillet, P., Dolan, J. M., and Sukhatme, G. S. (2012). Decentralized data fusion and active sensing with mobile sensors for modeling and predicting spatiotemporal traffic phenomena. In *Proc. UAI*, pages 163–173.

Chen, J., Cao, N., Low, K. H., Ouyang, R., Tan, C. K.-Y., and Jaillet, P. (2013). Parallel Gaussian process regression with low-rank covariance matrix approximations. arXiv:1305.5826.

Choudhury, A., Nair, P. B., and Keane, A. J. (2002). A data parallel approach for large-scale Gaussian process modeling. In *Proc. SDM*, pages 95–111.

Das, K. and Srivastava, A. N. (2010). Block-GP: Scalable Gaussian process regression for multimodal data. In *Proc. ICDM*, pages 791–796.

Dolan, J. M., Podnar, G., Stancliff, S., Low, K. H., Elfes, A., Higinbotham, J., Hosler, J. C., Moisan, T. A., and Moisan, J. (2009). Cooperative aquatic sensing using the telesupervised adaptive ocean sensor fleet. In *Proc. SPIE Conference on Remote Sensing of the Ocean, Sea Ice, and Large Water Regions*, volume 7473.

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance tapering for interpolation of large spatial datasets. *JCGS*, **15**(3), 502–523.

Golub, G. H. and Van Loan, C.-F. (1996). *Matrix Computations*. Johns Hopkins Univ. Press, third edition.

Ingram, B. and Cornford, D. (2010). Parallel geostatistics for sparse and dense datasets. In P. M. Atkinson and C. D. Lloyd, editors, *Proc. geoENV VII*, pages 371–381. Quantitative Geology and Geostatistics Volume 16, Springer, Netherlands.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, **9**, 235–284.

Lawrence, N. D., Seeger, M., and Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press.

Low, K. H., Gordon, G. J., Dolan, J. M., and Khosla, P. (2007). Adaptive sampling for multi-robot wide-area exploration. In *Proc. IEEE ICRA*, pages 755–760.

Low, K. H., Dolan, J. M., and Khosla, P. (2011). Active Markov information-theoretic path planning for robotic environmental sensing. In *Proc. AAMAS*, pages 753–760.

Low, K. H., Chen, J., Dolan, J. M., Chien, S., and Thompson, D. R. (2012). Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proc. AAMAS*, pages 105–112.

Park, C., Huang, J. Z., and Ding, Y. (2011). Domain decomposition approach for fast Gaussian process regression of large spatial data sets. *JMLR*, **12**, 1697–1728.

Pjesivac-Grbovic, J., Angskun, T., Bosilca, G., Fagg, G. E., Gabriel, E., and Dongarra, J. (2007). Performance analysis of MPI collective operations. *Cluster Computing*, **10**(2), 127–143.

Podnar, G., Dolan, J. M., Low, K. H., and Elfes, A. (2010). Telesupervised remote surface water quality sensing. In *Proc. IEEE Aerospace Conference*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *JMLR*, **6**, 1939–1959.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

Schwaighofer, A. and Tresp, V. (2002). Transductive and inductive methods for approximate Gaussian process regression. In *Proc. NIPS*, pages 953–960.

Seeger, M. and Williams, C. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Proc. AISTATS*.

Snelson, E. (2007). Local and global sparse Gaussian process approximations. In *Proc. AISTATS*.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Proc. NIPS*.

Vanhatalo, J. and Vehtari, A. (2008). Modeling local and global phenomena with sparse Gaussian processes. In *Proc. UAI*, pages 571–578.

Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Comput.*, **17**(12), 2602–2634.

Williams, C. K. I. and Seeger, M. (2000). Using the Nyström method to speed up kernel machines. In *Proc. NIPS*, pages 682–688.

Yu, J., Low, K. H., Oran, A., and Jaillet, P. (2012). Hierarchical Bayesian nonparametric approach to modeling and learning the wisdom of crowds of urban traffic route planning agents. In *Proc. IAT*, pages 478–485.