
Supplementary Material: Gaussian Process-Based Real-Time Learning for Safety Critical Applications

Armin Lederer¹ Alejandro José Ordóñez Conejo² Korbinian Maier¹ Wenxin Xiao³ Jonas Umlauf¹
Sandra Hirche¹

A. Related Work

Although scalability is a major issue of exact Gaussian process regression, a wide variety of methods has been developed in recent years to overcome this problem. An extensive overview of these methods can be found in (Liu et al., 2020). Following the classification of methods introduced in this survey article, we distinguish between global and local approximations of GPs for online learning.

A.1. Global Gaussian Process Approximations

Global GP approximations comprise by far the largest group of online learning methods. Among the most common approaches are sparse approximations (Snelson & Ghahramani, 2007), which aim to reduce the computational complexity of the inverse of the kernel matrix. This can be achieved using prior approximations, posterior approximations and structured sparse approximations. The deterministic training conditional (DTC) approximation is a particularly widespread prior approximation for online learning, since it achieves a constant complexity of predictions by heuristically choosing an active subset from the training data set. For determining the active subset, various methods have been proposed with different complexities (Csató & Oppel, 2002; Nguyen-Tuong & Peters, 2010; Schreiter et al., 2016; Koppel, 2019). The constant complexity of predictions comes at the price of a linear complexity of updates. Moreover, the selection heuristics often work best when the data is passed through them several times. Therefore, these approaches are well-suited for real-time predictions, but typically become too slow for online updates, preventing applications such as event-triggered learning.

Fully independent training conditional (FITC) and partially independent training conditional (PITC) approximations follow a similar idea for approximating the prior, but use

arbitrary inducing points to compress the information of the original training data. The flexibility of choosing arbitrary inducing points can be used to construct a suitable covering of the input domain online, e.g., through a prior design (Huber, 2014), online clustering techniques (Bijl et al., 2015; 2017), or by selecting training inputs as inducing points (Le et al., 2017). While this can be advantageous regarding regression performance, it generally does not have a positive effect on the computational complexity. Therefore, these approximations are best suited for off-line training and on-line predictions, but cannot be applied when fast updates are necessary as in, e.g., event-triggered learning.

Subset of regressor approaches are a form of prior approximation, which is rather depreciated in big data problems, but has demonstrated to be very successful in online learning problems. The idea of these approaches lies in the approximation of the kernel, such that the complexity of the kernel inverse is reduced. This can be achieved directly via a compactification of covariance functions (Ranganathan & Yang, 2008; Ranganathan et al., 2011), such that sparsity in the cholesky factors is ensured. More popularly, finite feature maps are constructed, which allow to approximately express the kernel as a scalar product. This procedure results in constant update and prediction complexities, which only depend on the number of features. For determining the features different approaches exist. When prior data is available, meta-training can be used to fit features to the training data using neural networks (Harrison et al., 2018) or least squares (Camoriano et al., 2016). Conversely, without any offline data, random trigonometric features with strong theoretical guarantees can be easily determined using Bochner’s theorem (Rahimi & Recht, 2008), such that the method is often referred to as sparse spectrum GP (Lázaro-Gredilla et al., 2010; Gijsberts & Metta, 2013). In contrast to most other methods, sparse spectrum GPs directly inherit many theoretical properties from exact Gaussian process regression due to Bochner’s theorem (Lu et al., 2020). Moreover, when numerical integration is used for obtaining the feature maps instead of random sampling, uniform error bounds can be extended from exact GP inference (Munín & Krause, 2018; Angelis et al., 2020). However, these bounds often require a practically intractable number of features,

¹Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany ²Tecnológico de Costa Rica, Cartago, Costa Rica ³Department of Computer Science and Technology, Peking University, Beijing, China. Correspondence to: Armin Lederer <armin.lederer@tum.de>.

as discussed in Appendix B.4. Moreover, these methods are known to suffer from overfitting (Gal & Turner, 2015) and their posterior variances are overconfident (Liu et al., 2020). Furthermore, the posterior mean and variance will be periodic functions, such that the variance might collapse far from any training samples (van der Wilk, 2018) leading to overconfident predictions.

Posterior approximations of GPs do not approximate the prior distribution, but instead aim at minimizing the difference between the approximate and exact posterior GP distributions. The most common posterior approximation is the variational free energy (Titsias, 2009), which can be efficiently optimized using stochastic optimization methods (Hensman et al., 2013; Cheng & Boots, 2016). Although these approaches can be applied in online learning problems with streaming data in principle, they are usually unsuited for this task as discussed in (Bui et al., 2017). The reasons for this are manifold. First, the optimization methods have the underlying assumption that data is uniformly randomly subsampled into mini-batches. While streaming data can often be aggregated into mini-batches, the data is rarely drawn i.i.d. from the input distribution. Moreover, the data should typically be passed to the optimizer multiple times, which typically cannot be satisfied with streaming data due to computational constraints. Finally, typically only a single gradient step is performed for every mini-batch. Since data cannot be revisited, this causes a risk of forgetting old data. In order to overcome these issues, Bui et al. (2017) proposed a posterior approximation for streaming data, which allows online predictions and online updates in minibatches. While this algorithm achieves a good regression performance, the limitation to minibatches can be prohibitive in applications such as event-triggered learning, where the update must be performed after every new sample.

In contrast to prior and posterior approximations, structured sparse approximations do not change the involved distributions directly, but instead aim at exploiting fast matrix-vector multiplication methods for computing an approximate of the inverse kernel matrix. In (Wilson & Nickisch, 2015), inducing points on a grid together with linear interpolation are used for this purpose, such that a constant complexity of mean predictions can be achieved. Using Lanczos approximation, these ideas are extended to reduce the complexity of posterior variance computations to $\mathcal{O}(1)$ (Pleiss et al., 2018). Although these methods achieve impressive prediction rates, online updates have not been investigated. Therefore, these methods cannot be applied to problems with streaming data.

A.2. Local Gaussian Process Approximations

The number of local GP approximations for online learning is significantly lower than for global approximations, but they are frequently used in practical applications. Among

the most straightforward approaches are naive local models, which adapt the used data set to the input. This can be achieved, e.g., through a windowing approach (Meier & Schaal, 2016) or by choosing the data subset based on task-specific information theoretic metrics (Umlauf et al., 2020). Although these approaches achieve a constant update and prediction complexity and typically work well in applications where a local model is sufficient, they suffer from several issues. For example, the predictive mean function of these methods is usually discontinuous, which is in contrast to the smoothness assumptions posed by many commonly used kernel functions. Moreover, predictions are only valid locally, which prevents the usage in applications such as model predictive control or model-based reinforcement learning.

Mixture of experts approaches overcome this issue by composing a global model of multiple locally active Gaussian process experts. While mixture of experts have originally been proposed to address the challenge of multi-modal data (Tresp, 2001), explicitly localized models have led to great success in online learning (Nguyen-Tuong et al., 2009b;a; Liu et al., 2016). Since the number of local models and their respective region in the input domain are not known a priori, they are typically adapted to the streaming data online. The resulting prediction performance crucially depends on parameters controlling this domain clustering behavior. In order to avoid an excessive number of data points per local model, data points are typically added and removed according to an information criterion. When this happens too often, the regression performance can suffer. However, if too many local models are generated, the computation time increases due to a linear dependency of the update and prediction complexity on the number of models. The trade-off between computation time and prediction performance depends on the a few crucial parameters, which are hard to tune, particularly in online learning problems with streaming data. Therefore, the application of mixtures of explicitly localized experts in real-time learning problems is often challenging.

A.3. Tree-Structure in Gaussian Process Approximations

In order to overcome the issues of mixtures of explicitly localized GP experts, our approach employs trees for defining the computing architecture. This idea goes back to Cao & Fleet (2014), who used a generalized product of experts approach for aggregating individual GP models. In the original approach, the data is split into multiple subsets by constructing a ball-tree (Omohundro, 1989), which is an efficient method for representing models and allows fast queries of individual leaves of the tree. Although each node of the ball-tree contains a separate GP model and all models are generally evaluated for the prediction of a test point, only evaluating models along the branch assigned to a test point

has been investigated, too. Similar ideas have been used in (Ng & Deisenroth, 2014), where the tree is employed primarily as computation graph. In contrast to the ball-tree, a k-d tree is recursively constructed from a batch of data until a prescribed number of leaves containing all the individual GP models is reached. While these approaches exploit methods for the explicit localization of models, this idea is dropped in (Deisenroth & Ng, 2015). Instead, the Bayesian Committee Machine proposed in (Tresp, 2000) is adapted for aggregating the predictions of Gaussian process models, such that a higher importance is put on models with low posterior variance. The tree structure of the individual models serves in this method as an efficient way for distributing the data to several computing nodes. Since the previously mentioned data clustering and aggregation methods can be shown to be inconsistent, i.e., they do not converge to the true distribution asymptotically, Rulli  re et al. (2018) propose to consider covariances between the individual models. While the resulting model is consistent, the computational complexity of the aggregation increases significantly. Therefore, (Liu et al., 2018) introduce the generalized robust Bayesian committee machine, which augments existing tree architectures by maintaining an additional global data set, which contains data uniformly spread over the input domain. By communicating this data to all leaves of the computing tree, consistency is recovered. Although these approaches can scale GPs to millions of training samples, this is mostly achieved through parallelization, but the asymptotic complexity of predictions typically remains linear in the number of individual GP models. In online learning problems, this can become problematic, since the overhead of parallelization becomes significant when only a single prediction is computed. Moreover, an efficient online construction of the tree computing structure as well as error bounds for the predictions as required for safety-critical application has not been investigated.

In contrast to the existing GP approximations using trees as computation graph, LoG-GPs employ locally growing random trees, which provide multiple advantages for online learning. Using probability functions for the assignment of data to nodes allows the efficient construction of the computing tree with streaming data. In fact, the proposed method is very general and includes many existing methods as special cases, e.g., the k-d (Ng & Deisenroth, 2014) and ball-tree constructions (Cao & Fleet, 2014) can be seen as deterministic special cases for batch data. Although the idea of adapting the density and extension of local models to the data density has been inherent in aggregation schemes with localized models, most of the proposed approaches require the data in advance for clustering the data points into the leaves, such that they cannot handle streaming data. Moreover, the main motivation behind localized models in existing methods lies in an improvement of the regression

performance. All models, even those far from a test input, are typically evaluated for the overall prediction, which leads to a linear computational complexity in the number of models. LoG-GPs overcome this issue in a principled way by exploiting the tree structure and locality in each layer of the tree to limit the number of individual models which need to be evaluated at a test point. Each model can only be active for prediction in regions, in which it has also a positive probability of receiving training samples. This ensures a good prediction performance, while at the same time active models can be determined very efficiently using recursive tree search algorithms. Since the graph generated by LoG-GPs can be interpreted as random splitting tree, logarithmic complexity guarantees for predictions and updates can be straightforwardly obtained under weak assumptions. Moreover, the definition of the aggregation weights as probabilities directly guarantees that uniform error bounds from exact GP inference are inherited. Thereby, LoG-GPs are well-suited for online learning of streaming data in safety-critical applications.

It should be noted that in addition to the usage as computation graph, trees have found various other applications in GP approximations. Using a k-d tree to query the closest data to a test point and only use this data for inference, Vasudevan et al. (2009) can achieve a constant prediction complexity. However, this comes at the cost of a discontinuous model. In structured sparse approximations, k-d trees can be used to quickly cluster training samples, such that linear interpolation can be used between cluster centers (Shen et al., 2006). Thereby, a linear prediction complexity can be achieved, too. In (Gramacy & Lee, 2008), partition trees are employed with GPs at the leaf nodes, in order to regress multi-modal data with stationary kernels. A prior is used to specify the probability of splitting a leaf and generating children, such that Markov chain Monte Carlo methods can be used to determine the posterior. Finally, a linear complexity in the number of inducing points is achieved by clustering them into blocks and imposing a tree structure on the blocks (Bui & Turner, 2014). While these approaches demonstrate the capability of trees for scaling GPs to large non-stationary data sets, they play no role in online learning problems.

B. Proofs and Other Theoretical Results

This section presents proofs of the main results and auxiliary theory, which is helpful in their practical application. In order to ease the comprehension, theoretical results from Section 3 are repeated before the proof.

B.1. Computational Complexity

For proving the complexity guarantees of predictions and updates with LoG-GPs, we rely on the theory of random

split trees as introduced in (Devroye, 1998). A random split tree T is defined through the parameters K , \bar{N} , s_0 , s_1 , \mathbf{p} and N . The parameter N describes the number of balls in the tree, while \bar{N} denotes the maximum number of balls in a node of the tree. The number of children of each node is given by K . Each internal (non-leaf) node has s_0 balls, while each leaf node has at least s_1 balls. The split probability is described by \mathbf{p} .

The distribution of balls is done iteratively. Starting at the tree, a ball is assigned to a child by drawing from the random distribution \mathbf{p} until a leaf node is reached. If this leaf has already reached its capacity, then the tree is extended and s_1 are assigned to each child. The remaining $\bar{N} + 1 - Ks_1 - s_0$ nodes are finally assigned according to the random distribution \mathbf{p} .

It can be clearly seen that the tree construction of LoG-GPs is identical to that of a random split tree with $s_0 = 0$, $s_1 = 0$ and input dependent $\mathbf{p}^n(\mathbf{x})$. Due to Assumption 3.1, this allows us to bound the height of the tree in LoG-GPs by the height of random split tree with $c_1 \leq p_i \leq 1 - Kc_1$ for all $i = 1, \dots, K$. This is exploited to bound the complexity of updates in LoG-GPs.

Theorem 3.1. *The update of a LoG-GP with conditional assignment probabilities $\mathbf{p}^n(\cdot)$ satisfying Assumption 3.1 requires $\mathcal{O}_p(\log(N))$ computations¹.*

Proof. The tree of LoG-GPs is a random split tree in the sense of (Devroye, 1998). Assumption 3.3 ensures that in any split of the tree, data is distributed approximately equally to both sides in the sense that no child gets all the data almost surely. Hence, it follows from (Devroye, 1998, Theorem 1) that the height of the tree, i.e., the maximum depth of any leaf, grows logarithmically in probability. Moreover, it is trivial to see that the updating complexity of LoG-GPs depends linearly on the height of the tree, which proves the result. \square

While the height of the tree is crucial for the update complexity of LoG-GPs, the number of active leaf nodes is important for the prediction complexity, too. If the active number of leaves grows logarithmic with the number of training samples as guaranteed by Assumption 3.2, we obtain the following result.

Theorem 3.2. *Mean and variance predictions of LoG-GPs with conditional assignment probabilities $\mathbf{p}^n(\cdot)$ satisfying Assumptions 3.1 and 3.2 require $\mathcal{O}_p(\log^2(N))$ computations.*

¹Due to the stochasticity of random split trees, deterministic statements about the asymptotic complexity are not possible. Therefore we describe the asymptotic behavior in probability using $\mathcal{O}_p(\cdot)$, e.g., $h^m \in \mathcal{O}_p(\log(N)) \Leftrightarrow \lim_{N \rightarrow \infty} P(h^m > c \log(N)) = 0$ for some finite $c \in \mathbb{R}$.

Proof. It is trivial to see that the prediction of a single branch requires $\mathcal{O}(h^m)$ operations, such that Theorem 3.1 guarantees a complexity of $\mathcal{O}_p(\log(N))$ for a single branch. Moreover, the number of leaves m in the tree of a LoG-GP depends linearly on the number of training samples N , i.e., $|\mathbb{M}| \in \mathcal{O}(N)$. Therefore, we have to show that only $\mathcal{O}_p(\log(N))$ leaves m have a positive marginal probability ω_m and must be evaluated. It immediately follows from Assumption 3.2 that no more than $K^{\log(c_2 h^m + c_3)}$ leaves can be active, which implies that the number of active leaves behaves as $\mathcal{O}(h^m)$. Therefore, a prediction requires $\mathcal{O}((h^m)^2)$, which concludes the proof using Theorem 3.1. \square

B.2. Uniform Error Bound using Bayesian Principles

Based on Assumptions 3.3 and 3.4, it is straightforward to extend the error bound in (Lederer et al., 2019a, Theorem 3.1) to LoG-GPs as shown in the following.

Theorem 3.3. *Consider a distributed GP approach satisfying Assumption 3.4 and defined through the continuous covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$ with Lipschitz constant L_k on the compact set $\mathbb{X} \subset \mathbb{R}^d$. Furthermore, consider a continuous unknown function $f : \mathbb{X} \rightarrow \mathbb{R}$ with Lipschitz constant L_f and $N \in \mathbb{N}$ observations $y^{(i)}$ satisfying Assumption 3.3. Pick $\delta \in (0, 1)$, $\tau \in \mathbb{R}_+$ and set*

$$\beta(\tau) = 2 \log \left(d^{\frac{d}{2}} \max_{\mathbf{x}, \mathbf{x}' \in \mathbb{X}} \|\mathbf{x} - \mathbf{x}'\|_\infty^d |\mathbb{M}| \right) - \log(\delta 2^d \tau^d) \quad (29)$$

$$\gamma(\tau) = \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) \left(L_{\mu_m} \tau + \sqrt{\beta(\tau)} L_{\sigma_m} \tau \right) + L_f \tau, \quad (30)$$

where L_{μ_m} and L_{σ_m} denote the Lipschitz constants of the GP mean and standard deviation, respectively. Then, it holds that

$$P(|f(\mathbf{x}) - \tilde{\mu}(\mathbf{x})| \leq \eta(\tau, \mathbf{x}), \forall \mathbf{x} \in \mathbb{X}) \geq 1 - \delta, \quad (31)$$

where

$$\eta(\tau, \mathbf{x}) = \sqrt{\beta(\tau)} \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) \sigma_m(\mathbf{x}) + \gamma(\tau). \quad (32)$$

Proof. Due to Assumption 3.3 and (Lederer et al., 2019a, Theorem 3.1), for each individual model it holds with probability of at least $1 - \tilde{\delta}$ that

$$|f(\mathbf{x}) - \mu_m(\mathbf{x})| \leq \sqrt{\tilde{\beta}(\tau)} \sigma_m(\mathbf{x}) + (L_{\mu_m} + L_f) \tau + \sqrt{\beta(\tau)} L_{\sigma_m} \tau, \quad (33)$$

where

$$\tilde{\beta}(\tau) = 2 \log \left(\frac{d^{\frac{d}{2}} \max_{\mathbf{x}, \mathbf{x}' \in \mathbb{X}} \|\mathbf{x} - \mathbf{x}'\|_\infty^d}{\tilde{\delta} 2^d \tau^d} \right) \quad (34)$$

and $\tau \in \mathbb{R}_+$ can be an arbitrary constant. Moreover, we have

$$|f(\mathbf{x}) - \tilde{\mu}(\mathbf{x})| = \left| f(\mathbf{x}) - \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) \mu_m(\mathbf{x}) \right| \quad (35)$$

$$= \left| \sum_{m \in \mathbb{M}} \mu_m(\mathbf{x}) (f(\mathbf{x}) - \mu_m(\mathbf{x})) \right| \quad (36)$$

$$\leq \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) |f(\mathbf{x}) - \mu_m(\mathbf{x})|, \quad (37)$$

where the second line follows from Assumption 3.4 and the third line follows from the triangle inequality. Applying the union bound and setting $\tilde{\delta} = \delta/|\mathbb{M}|$, (33) holds jointly for all $m \in \mathbb{M}$, which concludes the proof. \square

A major advantage of Theorem 3.3 is that all involved parameters can directly be computed. In order to bound the Lipschitz constant of the posterior mean of GP models, we need to define the Lipschitz constant L_k of the kernel $k(\cdot, \cdot)$. Following standard definitions of Lipschitz constants, we consider every value L_k satisfying

$$|k(\mathbf{x}, \mathbf{x}') - k(\tilde{\mathbf{x}}, \mathbf{x}')| \leq L_k \|\mathbf{x} - \tilde{\mathbf{x}}\| \quad (38)$$

for all $\mathbf{x}, \mathbf{x}', \tilde{\mathbf{x}}$ to be a valid Lipschitz constant of $k(\cdot, \cdot)$. Since most kernels are differentiable, this value can typically be obtained through the first derivative of the kernel. This allows to directly compute the Lipschitz constant L_μ of the GP mean using the following lemma.

Lemma B.1. *The Lipschitz constant L_μ of a GP posterior mean with N training samples is bounded by*

$$L_\mu \leq L_k \sqrt{N} \|\boldsymbol{\alpha}\| \quad (39)$$

where L_k denotes the Lipschitz constant of the kernel $k(\cdot, \cdot)$.

Proof. The proof can be found in (Lederer et al., 2019a). \square

Although the Lipschitz constant of the posterior variance can also be computed as outlined in (Lederer et al., 2019a), this approach suffers from an increasing Lipschitz constant with growing data set size. Therefore, we exploit the kernel pseudo-metric for deriving a Lipschitz constant, as suggested in (Curi et al., 2020).

Lemma B.2. *The posterior standard deviation $\sigma(\cdot)$ of a Gaussian process with stationary kernel $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ admits a Lipschitz constant*

$$L_\sigma = \sup_{\mathbf{x}, \mathbf{x}' \in \mathbb{X}} \left\| \sqrt{\frac{1}{s_f^2 - k(\mathbf{x} - \mathbf{x}')}} \frac{\partial k(\mathbf{r})}{\partial \mathbf{r}} \right|_{\mathbf{r}=\mathbf{x}-\mathbf{x}'} \right\|, \quad (40)$$

where $s_f^2 = k(\mathbf{x}, \mathbf{x})$.

Proof. Due to (Curi et al., 2020, Lemma 12), we can bound the difference between two GP standard deviations by

$$|\sigma(\mathbf{x}) - \sigma(\mathbf{x}')| \leq d_k(\mathbf{x}, \mathbf{x}'), \quad (41)$$

where the kernel pseudo-metric is defined as

$$d_k(\mathbf{x}, \mathbf{x}') = \sqrt{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}')}. \quad (42)$$

Due to stationarity of the kernel $k(\cdot, \cdot)$, we can simplify the kernel pseudo-metric to

$$d_k(\mathbf{x}, \mathbf{x}') = \sqrt{2s_f^2 - 2k(\mathbf{r})}, \quad (43)$$

where $s_f^2 = k(\mathbf{x}, \mathbf{x})$ and $\mathbf{r} = \mathbf{x} - \mathbf{x}'$. Hence, the Lipschitz constant of $\sigma(\cdot)$ is given by

$$L_\sigma = \sup_{\mathbf{x}, \mathbf{x}' \in \mathbb{X}} \left\| \sqrt{\frac{1}{s_f^2 - k(\mathbf{x} - \mathbf{x}')}} \frac{\partial k(\mathbf{r})}{\partial \mathbf{r}} \right|_{\mathbf{r}=\mathbf{x}-\mathbf{x}'} \right\|. \quad (44)$$

\square

The Lipschitz constant in (40) often reduces to a simple expression. For example, for the ARD squared exponential kernel

$$k(\mathbf{r}) = s_f^2 \exp \left(- \sum_{i=1}^d \frac{r_i^2}{2l_i^2} \right) \quad (45)$$

we have

$$\frac{\partial k(\mathbf{r})}{\partial \mathbf{r}} = k(\mathbf{r}) \begin{bmatrix} -\frac{r_1}{l_1^2} \\ \vdots \\ -\frac{r_d}{l_d^2} \end{bmatrix}. \quad (46)$$

Therefore, we obtain

$$L_\sigma = \sup_{\mathbf{x}, \mathbf{x}' \in \mathbb{X}} \left\| \sqrt{\frac{k^2(\mathbf{x} - \mathbf{x}')}{s_f^2 - k(\mathbf{x} - \mathbf{x}')}} \begin{bmatrix} \frac{x_1 - x'_1}{l_1^2} \\ \vdots \\ \frac{x_d - x'_d}{l_d^2} \end{bmatrix} \right\|. \quad (47)$$

It can be shown that this expression reaches its maximum for $\mathbf{x} \rightarrow \mathbf{x}'$, such that L'Hôpital's rule can be used to derive

$$L_\sigma = \sigma_f \left\| \begin{bmatrix} \frac{1}{l_1} \\ \vdots \\ \frac{1}{l_d} \end{bmatrix} \right\|. \quad (48)$$

B.3. Uniform Error Bound based on RKHS Theory

While we focus on uniform error bounds derived from Bayesian principles in the main article, these bounds can be analogously derived using the theory of reproducing kernel

Hilbert spaces (RKHS) (Srinivas et al., 2012; Chowdhury & Gopalan, 2017; Fiedler et al., 2021). We demonstrate this by extending the uniform error bound presented in (Fiedler et al., 2021) to LoG-GP predictions.

Every kernel $k(\cdot, \cdot)$ uniquely defines a RKHS $\mathcal{H}_k(\mathbb{X})$ on a compact set \mathbb{X} with an inner product $\langle \cdot, \cdot \rangle_k$ obeying the reproducing property $f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot) \rangle_k$ for all $f(\cdot) \in \mathcal{H}_k(\mathbb{X})$ (Schölkopf & Smola, 2002). The RKHS norm of a function $\|f\|_k = \sqrt{\langle f, f \rangle_k}$ is a measure of the functions complexity, which motivates the following commonly used assumption.

Assumption B.1. *The RKHS norm of the unknown function $f(\cdot)$ is upper bounded by B , i.e. $\|f\|_k \leq B$.*

While the Bayesian approach is restricted to Gaussian noise with known variance, RKHS based approaches are more flexible and admit the following assumption on the observation noise.

Assumption B.2. *The noise sequence ϵ_n is conditionally R -sub-Gaussian for a fixed constant $R \geq 0$, i.e.,*

$$\forall n \geq 0, s \in \mathbb{R} : \mathbb{E} \left[e^{s\epsilon_n | \mathcal{F}_{n-1}} \right] \leq \exp \left(\frac{s^2 R^2}{2} \right), \quad (49)$$

where \mathcal{F}_{n-1} is the σ -algebra generated by the random variables $\{\mathbf{x}^{(t)}, \epsilon_t\}_{t=1}^{n-1}$ and $\mathbf{x}^{(n)}$.

This assumption is not restrictive as it is satisfied by, e.g., bounded or Gaussian noise (Chowdhury & Gopalan, 2017).

Based on these assumptions, we can extend Theorem 1 in (Fiedler et al., 2021) to LoG-GPs as shown in the following.

Theorem B.1. *Consider a distributed GP approach satisfying Assumption 3.4 and defined through the covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$. Furthermore, consider an unknown function $f : \mathbb{X} \rightarrow \mathbb{R}$ and $N \in \mathbb{N}$ observations $y^{(i)}$ satisfying Assumptions B.1 and B.2. Pick $\delta \in (0, 1)$, define $\bar{\sigma}_n = \max\{\sigma_n, 1\}$, and set*

$$\beta_m^{\text{RKHS}}(\delta) = B + R \sqrt{\log(\det(\mathbf{K}_m + \bar{\sigma}_n^2 \mathbf{I}_{N_m})) - 2 \log(\delta)} \quad (50)$$

Then, it holds that

$$P(|f(\mathbf{x}) - \tilde{\mu}(\mathbf{x})| \leq \eta_{\text{RKHS}}(\mathbf{x}), \forall \mathbf{x} \in \mathbb{X}) \geq 1 - \delta, \quad (51)$$

where

$$\eta^{\text{RKHS}}(\mathbf{x}) = \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) \sqrt{\beta_m^{\text{RKHS}} \left(\frac{\delta}{M} \right) \sigma_m(\mathbf{x})}. \quad (52)$$

Proof. Due to (Fiedler et al., 2021), we have for each local GP model that

$$|f(\mathbf{x}) - \mu_m(\mathbf{x})| \leq \sqrt{\beta_m^{\text{RKHS}}(\tilde{\delta}) \sigma_m(\mathbf{x})} \quad (53)$$

with probability of at least $1 - \tilde{\delta}$. Moreover, we have

$$|f(\mathbf{x}) - \tilde{\mu}(\mathbf{x})| = \left| f(\mathbf{x}) - \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) \mu_m(\mathbf{x}) \right| \quad (54)$$

$$= \left| \sum_{m \in \mathbb{M}} \mu_m(\mathbf{x}) (f(\mathbf{x}) - \mu_m(\mathbf{x})) \right| \quad (55)$$

$$\leq \sum_{m \in \mathbb{M}} w_m(\mathbf{x}) |f(\mathbf{x}) - \mu_m(\mathbf{x})|, \quad (56)$$

where the second line follows from Assumption 3.4 and the third line follows from the triangle inequality. Applying the union bound and setting $\tilde{\delta} = \delta/|\mathbb{M}|$, (53) holds jointly for all $m \in \mathbb{M}$, which concludes the proof. \square

B.4. Comparison to Theoretical Guarantees for Spectral Approximations

Although finite feature approximations of stationary kernels also allow to extend uniform error bounds from exact GP inference (Mutný & Krause, 2018; Angelis et al., 2020), they often require an impractically high number of features to be useful. We demonstrate this in the following for the commonly used random Fourier features (Rahimi & Recht, 2008) and the more frequently introduced quadrature Fourier features (Mutný & Krause, 2018; Angelis et al., 2020).

Given a trigonometric feature map $\mathbf{z} : \mathbb{R}^d \rightarrow \mathbb{R}^{2D}$, $D \in \mathbb{R}_+$, the covariance function can be approximated by $k(\mathbf{x}, \mathbf{x}') = \mathbf{z}^T(\mathbf{x}) \mathbf{z}(\mathbf{x}')$, such that Gaussian process regression reduces to Bayesian linear regression, see, e.g., (Bishop, 2006). Error bounds from exact inference can straightforwardly be extended by uniformly bounding the approximation error caused in the posterior mean and posterior variance. For example, the difference of the posterior mean $\mu(\cdot)$ of the Gaussian process with kernel $k(\cdot, \cdot)$ and the posterior mean $\tilde{\mu}(\cdot)$ resulting from the Fourier features can be bounded by

$$|\mu(\mathbf{x}) - \tilde{\mu}(\mathbf{x})| \leq \epsilon \frac{(N+1)^2}{\sigma_n^2} \left(B + \sqrt{2 \log \left(\frac{1}{\delta} \right)} \right) \quad (57)$$

for all $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$ with probability of at least $1 - \delta$ due to (Mutný & Krause, 2018, Theorem 5), where $B \in \mathbb{R}_+$ is an upper bound on the characteristic spectral function of $f(\cdot)$ and ρ denotes the uniform approximation bound for the feature map, i.e., $|k(\mathbf{x}, \mathbf{x}') - \mathbf{z}^T(\mathbf{x}) \mathbf{z}(\mathbf{x}')| \leq \rho$ for all $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$. Due to the linear dependence on ρ , small values must be guaranteed for the uniform approximation bound. For random Fourier features, small values can be guaranteed with probability of at least $1 - \tilde{\delta}$ (Rahimi & Recht, 2008),

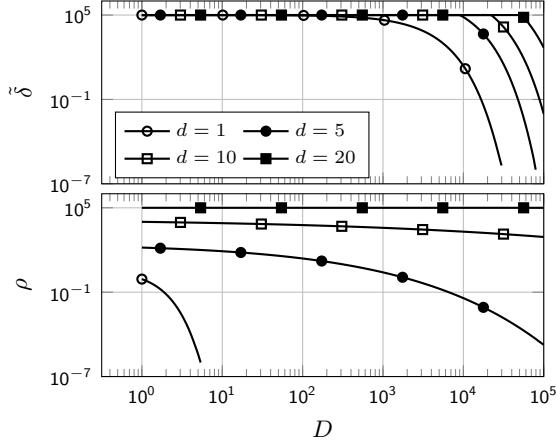


Figure 5. A high number of random Fourier features is generally required to guarantee at least a decent uniform approximation bound for the squared exponential kernel. While fewer quadrature Fourier features are sufficient for very low dimensional input spaces, medium values of $d \geq 5$ require an impractical high number D .

which depends on the number of features D through

$$\tilde{\delta} = 2^8 \left(\frac{\sigma_p \max_{\mathbf{x}, \mathbf{x}' \in \mathbb{X}} \|\mathbf{x} - \mathbf{x}'\|^d}{\rho} \right)^2 \exp \left(-\frac{D\rho^2}{4d+8} \right), \quad (58)$$

where σ_p^2 is the trace of the Hessian at 0, e.g., for the ARD squared exponential we have $\sigma_p^2 = 2 \sum_{i=1}^d \frac{1}{l_i^2}$. For quadrature Fourier features, the deterministic kernel approximation bound

$$\rho = d2^{d-1} \sqrt{\frac{\pi}{2}} \frac{1}{\bar{D}^{\bar{D}}} \left(\frac{e}{4 \min_{i=1, \dots, d} l_i^2} \right)^{\bar{D}} \quad (59)$$

can be derived with $D = \frac{(2\bar{D})^d}{2}$ as shown in (Mutný & Krause, 2018). It can be seen that both expressions strongly grow with the input dimensionality d , such that the bounds can only be exploited for low-dimensional inputs, but become impractical for large d . This is illustrated in Fig. 5, which shows the dependency of $\tilde{\delta}$ and ρ on D for different values of d . The value of ρ is set to 0.1, all other parameters are set to 1 for simplicity. It clearly demonstrates that bounds such as (57) are limited to small input dimensions and hence, these approaches for extending uniform error bounds from exact GP regression are not suited to many problems in practice.

B.5. Designing Probability Functions for Kernels with Structure

While the application of LoG-GPs is rather simple for stationary kernels, the definition of the localizing distributions $p^n(\mathbf{x})$ is a challenging problem in general. Therefore, we

provide some examples how these functions can be defined for kernels reflecting a priori known function structures in order to give some insights on the design of $p^n(\mathbf{x})$ for more general kernels.

Symmetric Kernels: If we know that an unknown function satisfies $f(\mathbf{x}) = f(-\mathbf{x})$, we can easily reflect this symmetry in the kernel by considering covariance functions of the form

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') + k(-\mathbf{x}, \mathbf{x}'), \quad (60)$$

where $k(\cdot, \cdot)$ is an arbitrary kernel (Duvenaud, 2014). In order to keep the advantages of the kernel $\tilde{k}(\cdot, \cdot)$, the symmetric structure should also be considered in the localizing probability functions $p^n(\cdot)$. This can be straightforwardly achieved by defining them based on (12), but employing embodying symmetry in the saturating linear functions, e.g.,

$$\xi_k^n(\mathbf{x}) = \begin{cases} 0 & \text{if } |x_{j_k}^n| < s_k^n - \frac{o_k^n}{2} \\ \frac{|x_{j_k}^n| - s_k^n}{o_k^n} + \frac{1}{2} & \text{if } s_k^n - \frac{o_k^n}{2} \leq |x_{j_k}^n| \leq s_k^n + \frac{o_k^n}{2} \\ 1 & \text{if } s_k^n + \frac{o_k^n}{2} < |x_{j_k}^n|. \end{cases} \quad (61)$$

Periodic Kernels: Similarly to symmetry, periodicity of unknown functions can be straightforwardly encoded in kernels. Given an isotropic covariance function $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$, the periodic analog is defined as

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = k \left(\sin^2 \left(\frac{\|\mathbf{x} - \mathbf{x}'\| \pi}{p} \right) \right), \quad (62)$$

where p is the period. By considering the periodicity in the saturating linear functions, this information can be also exploited in LoG-GPs. This is achieved by defining

$$\xi_k^n(\mathbf{x}) = \begin{cases} 0 & \text{if } x_{j_k}^n \% p < s_k^n - \frac{o_k^n}{2} \\ \frac{x_{j_k}^n \% p - s_k^n}{o_k^n} + \frac{1}{2} & \text{if } s_k^n - \frac{o_k^n}{2} \leq x_{j_k}^n \% p \leq s_k^n + \frac{o_k^n}{2} \\ 1 & \text{if } s_k^n + \frac{o_k^n}{2} < x_{j_k}^n \% p, \end{cases} \quad (63)$$

where $\%$ denotes the modulo division.

Multi-Dimensional Products of Stationary and Non-Stationary Kernels: In particular in control applications, prior knowledge of state dependencies is often available, e.g., if systems are control-affine. This can be exploited by multi-dimensional product kernels of the form

$$\tilde{k}([\mathbf{x}_1^T \ \mathbf{x}_2^T]^T, [\mathbf{x}_1'^T \ \mathbf{x}_2'^T]^T) = k_1(\mathbf{x}_1, \mathbf{x}_1') k_2(\mathbf{x}_2, \mathbf{x}_2'), \quad (64)$$

where $k_1(\cdot, \cdot)$ and $k_2(\cdot, \cdot)$ are covariance functions. As long as either $k_1(\cdot, \cdot)$ or $k_2(\cdot, \cdot)$ is stationary, we can employ the proposed standard approach for defining the probability functions $p^n(\cdot)$, merely restricting the indexes j_k of the saturating linear functions to the inputs of the stationary kernel.

B.6. Safe Event-Triggered Learning Control

Due to the event-triggered learning, the continuous-time control becomes affected by discrete time events. This leads to the fact that the closed-loop system becomes a switching system (Liberzon, 2003). Therefore, we use the concept of a common Lyapunov function to derive an ultimate bound in the following theorem.

Theorem 3.4. *Consider a control affine system (19), where $f(\cdot)$ satisfies Assumption 3.3 and admits a Lipschitz constant L_f on $\mathbb{X} \subset \mathbb{R}^d$, and measurements are available according to Assumption 4.1. Let $\mathbf{P} \in \mathbb{R}^{d \times d}$ the unique, positive definite solution to the algebraic Riccati equation $\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A} = -\mathbf{I}_d$ with \mathbf{A} defined in (21). Then, the feedback linearizing controller (20) with $\tilde{\mu}(\cdot)$ based on a stationary kernel and event-triggering mechanism given in Algorithm 3 with $\underline{\eta}(\tau) = \sqrt{\beta(\tau)\underline{\sigma} + \gamma(\tau)}$ and $\underline{\sigma}^2 > \sigma_n^2 k(0, 0)/(k(0, 0) + \sigma_n^2)$ guarantees with probability $1 - \delta$ that the tracking error e converges to $\mathbb{T} = \{\mathbf{x} \in \mathbb{X} \mid \|e\| \leq 2\underline{\eta}(\tau)\|\mathbf{p}_d\|\}$.*

Proof. Since the filter vector λ is assumed to be Hurwitz, there exists a unique and positive definite solution $\mathbf{P} \in \mathbb{R}^{d \times d}$ to the algebraic matrix Riccati equation

$$\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A} = -\mathbf{I}_d.$$

Based on this matrix, consider the common Lyapunov function $V(\mathbf{x}) = \mathbf{e}^T \mathbf{P} \mathbf{e}$ and the models $\tilde{\mu}_k(\cdot)$, where $k \in \mathbb{N}_0$ denotes the number of added data points, then

$$\begin{aligned} \dot{V}(\mathbf{e}) &= \frac{\partial V}{\partial \mathbf{e}} \dot{\mathbf{e}} \\ &= \mathbf{e}^T \mathbf{A}^T \mathbf{P} \mathbf{e} + \mathbf{e}^T \mathbf{P} \mathbf{A} \mathbf{e} + 2\mathbf{e}^T \mathbf{p}_d (f(\mathbf{x}) - \nu_N(\mathbf{x})) \\ &\leq -\|\mathbf{e}\|^2 + 2\|\mathbf{e}\|\|\mathbf{p}_d\|(f(\mathbf{x}) - \nu_N(\mathbf{x})) \\ &\leq 0 \quad \forall \frac{\|\mathbf{e}\|}{2\|\mathbf{p}_d\|} > |f(\mathbf{x}) - \tilde{\mu}_k(\mathbf{x})|. \end{aligned}$$

To guarantee convergence for arbitrary switching times, the Lyapunov function must be decreasing $\forall k$, i.e. the controller must ensure that the model error $|f(\mathbf{x}) - \tilde{\mu}_k(\mathbf{x})|$ does not exceed $\|\mathbf{e}\|/(2\|\mathbf{p}_d\|)$ for all k . With the model error bound in Theorem 3.3, one can conclude

$$P(\dot{V}(\mathbf{x}) < 0) \geq 1 - \delta \quad \forall \mathbf{x} \text{ if } \eta_k(\mathbf{x}, \tau) < \frac{\|\mathbf{e}\|}{2\|\mathbf{p}_d\|}, \quad (65)$$

where the latter condition is ensured by the triggering mechanism (first part line 3, Algorithm 3) for all $\mathbf{x} \in \mathbb{X} \setminus \mathbb{T}$ (second part line 3, Algorithm 3). In order to see this, we consider three cases.

First, we consider the case of the first measurement $y^{(1)}$ at state $\mathbf{x}^{(1)}$. It directly follows from (Williams & Vivarelli, 2000; Lederer et al., 2019b) that the variance satisfies

$$\tilde{\sigma}^2(\mathbf{x}^{(k)}) \leq \frac{\sigma_n^2}{1 + \frac{\sigma_n^2}{k(0,0)}}. \quad (66)$$

Hence, regardless of the error e , either (65) or the desired error bound described by $\underline{\eta}(\tau)$ is satisfied, such that the *while loop* in line 6 in Algorithm 3 terminates after a single iteration.

Next, we consider the case that no node is divided during the assignment of the measurement $y^{(k)}$ with state $\mathbf{x}^{(k)}$. Then, it follows from a slight adaptation of (Vivarelli, 1998) that for positive noise variance σ_n^2 , the posterior variance of an individual Gaussian process is strictly decreasing at the state $\mathbf{x}^{(k)}$ which is added to the training data, i.e.,

$$\sigma_{i,k}^2(\mathbf{x}^{(k)}) < \sigma_{i,k-1}^2(\mathbf{x}^{(k)}). \quad (67)$$

Hence, (65) is satisfied for all $\mathbf{x} \in \mathbb{T} \setminus \mathbb{B}$, and the *while loop* in line 6 in Algorithm 3 terminates again after a single iteration.

Finally, we consider the case that for the state $\mathbf{x}^{(k)}$, at which the measurement $y^{(k)}$ is recorded, several individual models have a positive probability $\omega_i(\mathbf{x})$ and a model division occurs. Due to the reduction of the training set size, a slight increase in the posterior variance of both new leaves cannot be excluded. However, the *while loop* in line 6 in Algorithm 3 ensures that additional data is added at the same position. Since all individual models contributing the aggregated prediction have a positive assignment probability, each of them will eventually be assigned a training sample with probability one. Due to the former two cases, this implies that (65) is satisfied for all $\mathbf{x} \in \mathbb{X} \setminus \mathbb{T}$.

Since (65) has been shown to hold for all $\mathbf{x} \in \mathbb{X} \setminus \mathbb{T}$, it follows that error converges to the ultimately bounded set \mathbb{T} , where Algorithm 3 does not trigger events. This concludes the proof. \square

Note that case 3 is a worst case consideration that does not occur in every node division. A slight increase of the posterior variance after a node division can only happen if the added point is in a region, where both new child nodes have a positive probability (since one of them does not get the new training sample), or if many training samples close to the newly added point get assigned to the other child during the division. Both cases can be easily addressed in practice by choosing conditional assignment probabilities that divide the state space sufficiently far away from the newly added training sample. Therefore, this case is not an issue in practice.

In fact, even without considering this in the design of the conditional assignment probabilities, an increase in the posterior variance is very unlikely to occur, since most of the time a single model is used for prediction anyways, as we want to employ as few as possible local models for computational efficiency.

C. Numerical Evaluation

In this section, we provide details on the numerical evaluation in Section 5, a discussion of the results and additional simulation results.

C.1. Detailed Information on Simulation Setup

All simulations are executed on a cluster computer with Intel(R) Core(TM) i9-9900X CPU and 128GB DDR4 RAM. The code is run using MATLAB R2019a when not stated differently.

Since the SARCOS data set² contains 7-dimensional targets, we only take the first column of the targets. The order of the rows in the data matrix is randomized in each of the 20 repetitions and inputs and targets are centralized. The training inputs of the buzz in social media and the household electric data sets³ are centralized. For determining the training targets, we take the last and sixth column of the data sets, respectively. Moreover, we take the logarithm of these values, but additionally add 1 for the buzz in social media data. Finally, these values are centralized.

For evaluating the performance of the LoG-GPs, ISSGP, local GPs, SSGP and rBCM, we use the following parameters:

- LoG-GP: each node n has $K = 2$ children. The probabilities $p^n(\cdot)$ are defined through (11), (12), where j_k^n is the dimension of the maximum spread of the local data set, s_k^n is the mean of the data in this dimension and

$$o_k = \frac{\max_{\mathbf{x}, \mathbf{x}' \in \mathbb{D}_k} \|\mathbf{x} - \mathbf{x}'\|}{100(\frac{h^n}{10} + 1)} \quad (68)$$

with h^n being the height of node n . Moreover, each local model can contain a maximum of $\bar{N} = 100$ training samples. For the MoE and gPoE aggregation schemes, we use (10), while we define

$$\omega_m = \frac{\log(k(\mathbf{x}, \mathbf{x})) - \log(\sigma_m^2(\mathbf{x}))}{2} \prod_{i=1}^{h^m} p_{s_i^m}^{b_i^m}(\mathbf{x}). \quad (69)$$

for the rBCM aggregation, where the first factor is the differential entropy between the prior and the posterior distribution as proposed in (Deisenroth & Ng, 2015).

- ISSGP: as suggested in (Gijsberts & Metta, 2013), we use $D = 200$ random Fourier features. Hence, the kernel matrix has size 400×400 .

²The data set is available at <http://www.gaussianprocess.org/gpml/data/>.

³The data and pre-processing are available at https://drive.google.com/file/d/0BxWe_IuTnMFcYXhxdUNwRHBKtLU/view.

- local GPs⁴: the maximum number of training samples of a local model is set to $\bar{N} = 100$. Moreover, a threshold of 0.9 is used to determine if a new model is generated.
- SSGP⁵: the method is used with 100 inducing points, but no online optimization of hyperparameters and inducing points to reduce computational complexity. Since the method does not allow iterative updates, we update it using mini-batches of size 300 as proposed in the original publication (Bui et al., 2017). This means that we determine the prediction error on 300 training samples before we update the model using these data samples. This method is implemented in Python.
- rBCM⁶: the maximum number of samples in a local model is set to $\bar{N} = 100$. The data is randomly distributed to the local models. Since the method does not allow iterative updates, we recompute the model each time after observing 1000 new samples using the data observed up to this time.

The hyperparameters for all methods are obtained through log-likelihood maximization based on 1000 training samples. This is done using the well-documented MATLAB internal routine whenever possible. For the Python implementation of SSGPs, hyperparameter optimization is based on the GPflow toolbox⁷ (Matthews et al., 2017). The computations for all methods are performed on a single computation unit. Note that the evaluation of the local models of LoG-GPs can be parallelized similarly as proposed by Deisenroth & Ng (2015) by distributing the active models to multiple computation units.

C.2. Additional Simulation Results and Discussion

C.2.1. REGRESSION PERFORMANCE

Tables 2 and 3 display the standardized mean square error and mean standardized log loss averaged over 20 simulation runs together with the corresponding standard deviations, whose evolution over the number of training samples is illustrated in Fig. 2. Table 2 clearly shows that LoG-GP approaches outperforms state-of-the-art online learning methods regarding the overall prediction error for data sets with medium and high dimensional input domains such as the SARCOS and buzz in social media data. For low di-

⁴The code is available at <https://www.ias.informatik.tu-darmstadt.de/Miscellaneous/Miscellaneous>.

⁵The code is available at https://github.com/thangbui/streaming_sparse_gp.

⁶The code is available at <https://github.com/LiuHaiTao01/GRBCM>.

⁷The code is available at <https://github.com/GPflow/GPflow>.

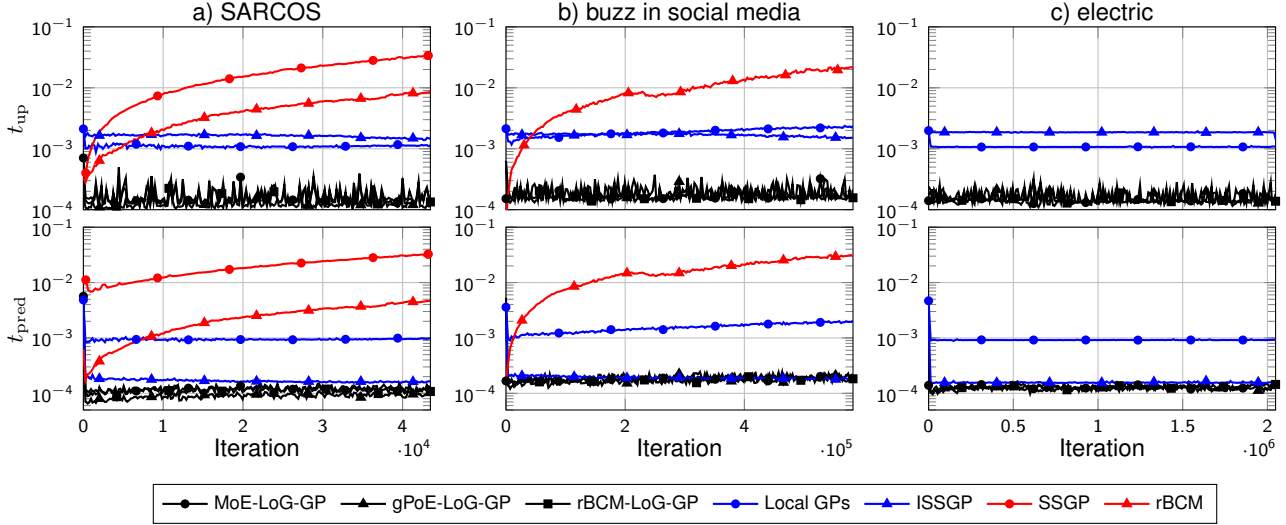


Figure 6. Plots of average update time t_{up} (top) and the average prediction time t_{pred} (bottom) on a) SARCOS b) buzz in social media and c) electric data sets. Due to the high computation times, the SSGP could only be applied to the SARCOS, while the rBCM could not be evaluated on the electric data set. Computation times of LoG-GP approaches are more noisy than those of existing methods due to the strongly varying size of local models. However, they are generally smaller, in particular for computing model updates.

Table 2. Average standardized mean squared error with the corresponding deviation in brackets for the SARCOS, buzz in social media and electric data sets. LoG-GP approaches show advantages in problems with medium and high dimensional input domains, while ISSGPs exhibit advantageous performance on low dimensional problems.

SMSE ($\cdot 10^{-3}$)	SARCOS	BUZZ	ELECTRIC
MoE-LoG-GP	31.3 (0.85)	88.0 (12.9)	5.0 (0.56)
gPoE-LoG-GP	30.3 (0.75)	89.2 (10.4)	4.8 (0.60)
rBCM-LoG-GP	30.7 (0.85)	101.4 (27.0)	5.2 (0.44)
ISSGP	30.9 (1.4)	100.1 (16.0)	3.4 (0.24)
LOCAL GPs	276.0 (64.5)	744.0 (35.9)	450.9 (87.9)
SSGP	35.9 (2.4)	—	—
rBCM	29.0 (1.3)	124.9 (42.4)	—

Table 3. Average mean standardized log loss with the corresponding standard deviation in brackets for the SARCOS, buzz in social media and electric data sets. LoG-GP approaches outperform existing approaches for online learning on all data sets.

MSLL	SARCOS	BUZZ	ELECTRIC
MoE-LoG-GP	-1.87 (0.02)	-1.34 (0.02)	-2.86 (0.02)
gPoE-LoG-GP	-1.88 (0.02)	-1.34 (0.02)	-2.86 (0.03)
rBCM-LoG-GP	-1.89 (0.02)	-1.33 (0.04)	-2.86 (0.03)
ISSGP	-1.68 (0.05)	-1.14 (0.11)	-2.84 (0.03)
LOCAL GPs	-0.18 (0.07)	-0.15 (0.05)	-0.03 (0.15)
SSGP	7.17 (2.61)	—	—
rBCM	78.2 (10.8)	375 (11.7)	—

mensional input domains as in the house electric data set, ISSGPs provide a slightly better performance. The weaker performance of ISSGPs for high dimensional input domains is a direct consequence of the strong dependence of the kernel approximation error on the input dimension as discussed in Appendix B.4.

The poor performance of local GPs, which is significantly

worse than originally presented by Nguyen-Tuong et al. (2009b), is a result of not tuning the threshold for generating new models for each data set. While tuning this parameter could improve the performance, this would conflict with the principle of online learning: for tuning the parameter, a significant amount of training data is necessary, while the online learning paradigm assumes little or even no data in advance. Moreover, tuning must be done by hand, which is time-consuming. Therefore, we chose the value 0.9 for the threshold empirically such that many local models are generated, yet not too many to keep the computation time tractable.

Table 3 shows that LoG-GPs provide better predictive distributions than state-of-the-art methods. Even though the difference to some other methods is small, it should be noted that the quality of the predictive distributions is crucial in safety-critical applications due to the dependence of uniform error bounds on the posterior standard deviations. Therefore, even a small improvement over existing methods is highly beneficial in practice.

While it might be surprising that rBCMs provide poor predictive distributions as indicated by the high MSLL values, this effect has already been observed in (Liu et al., 2018), where it is shown that the rBCM asymptotically becomes overconfident. Similarly, the rather weak performance of the SSGP can be explained by the slight differences in the simulation setup we used. While the inducing points and hyperparameters are updated in every mini-batch in the original publication (Bui et al., 2017), we refrain from doing so in order to keep the computation time tractable.

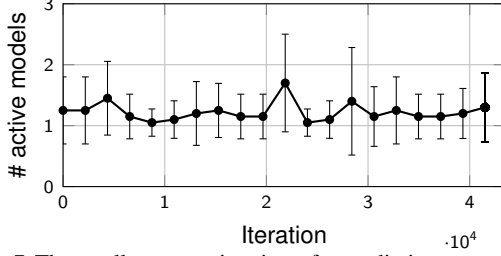


Figure 7. The small computation times for predictions are achieved by a low number of active models, which is on average less than 2 for the MoE-LoG-GP on the SARCOS data set. The comparatively high standard deviation of the number of active models illustrated by the error bars contributes to noisy prediction times of LoG-GPs. The observed maximum number of active models is 12 on the SARCOS data set.

Table 4. Standard deviation of update and prediction times in μs for the SARCOS, buzz in social media and electric data sets.

STD. DEVIATION (μs)	SARCOS		BUZZ		ELECTRIC	
	t_{pred}	t_{up}	t_{pred}	t_{up}	t_{pred}	t_{up}
MoE-LoG-GP	87	214	93	202	49	150
GPoE-LoG-GP	83	207	95	202	53	157
RBCM-LoG-GP	82	196	78	147	46	148
ISSGP	34	266	30	275	15	65
LOCAL GPs	113	182	657	672	30	34
SSGP	7481	9688	—	—	—	—
RBCM	1380	2550	9126	6619	—	—

In addition to the slight advantages in the regression performance, LoG-GP approaches exhibit significantly lower computation times as shown in Fig. 6. The total standard deviations of the update and prediction times are depicted in Table 4. These simulation results show that computation times of LoG-GPs are more strongly varying compared to existing methods, which is a consequence of the continuously changing size of the local models. Moreover, the logarithmic growth of the computation time is not visible, since the overlapping ratio α_k was chosen small, such that the average number of active models is almost constant as illustrated in Fig. 7 for the MoE-LoG-GP trained on the SARCOS data set. Since the depth of the tree is practically not relevant, almost constant computation times can be achieved with LoG-GPs, which is a strong advantage over the rBCM and the SSGP. Note that the difference in the used programming languages has an effect on the absolute computation time, but not on the behavior. While the computation time for updates of LoG-GPs in a Python implementation increased to approximately 5 ms on the SARCOS data set, it did practically not change with a growing number of training samples. Therefore, LoG-GPs remain advantageous compared to SSGP regardless of the employed programming language.

While other methods such as the ISSGP and the SSGP compress the information obtained from training data, the full data set is used for predictions with LoG-GP methods. Therefore, they exhibit a significantly higher memory com-

Table 5. Parameters for the learning control illustration

k_c	λ	\mathbf{x}_0	σ_n^2	σ_f	\mathbf{l}	τ	δ
10	1	$\begin{bmatrix} 7 \\ -5 \end{bmatrix}$	10^{-6}	5	$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	10^{-10}	0.05

plexity, which grows linearly with the number of training samples. However, in practice this does not cause significant problems with modern computers. For example, for a LoG-GP with 2^{16} nodes, which is sufficient for the buzz in social media data set, merely 1.05 GB RAM are necessary.

C.2.2. EVENT-TRIGGERED ONLINE LEARNING CONTROL

Since real-world continuous-time control systems are run on computers with finite processing power, controllers must be applied in a sampled-data sense in practice. We reflect this in our simulation by running the control loop with 1kHz rate and use zero order hold digital to analog conversion. The condition for triggering an updating event is sampled with the same rate. The control parameters and hyperparameters for the MoE-LoG-GP are depicted in Table 5. The desired upper bound on the tracking error is set to $\underline{\eta}(\tau) = 5\sqrt{\beta(\tau)\sigma_n} + \gamma(\tau)$. However, due to the implementation as sampled-data system, the learning event might be triggered delayed, such that adding a single data point might not sufficiently reduce the posterior variance, even for the exact GP. Therefore, we trigger the event early using the value $\tilde{\eta}(\tau) = 2\sqrt{\beta(\tau)\sigma_n} + \gamma(\tau)$, such that the original bound $\underline{\eta}(\tau)$ is met after adding a training sample.

The resulting tracking error together with the time instances of the triggered learning events is displayed in Fig. 8. It can be seen that the learning event is triggered frequently at the beginning until $\approx 20\text{s}$, when the system state has converged to the reference trajectory and sufficient data of the inner circular motion has been collected. After this time, barely any new data is sampled until the radius of the reference starts to increase at $\approx 60\text{s}$. Then, the model is again frequently updated until the learning has finished after 160s. Most importantly, it becomes clear that the stationary tracking error is clearly upper bounded, where most of time the early triggering condition $\tilde{\eta}(\tau)$ defines the bound. Only in a few instances, this bound is significantly exceeded. However, the error bound defined through $\underline{\eta}(\tau)$ permanently holds after the initial transition period.

C.3. Influence of Hyperparameters on Performance

When aiming for the highest possible prediction and update rates, any omissible computations must be spared. Therefore, we do not consider an online hyperparameter optimization in LoG-GPs and other methods. Nevertheless, hyperparameters can have a strong impact on the prediction

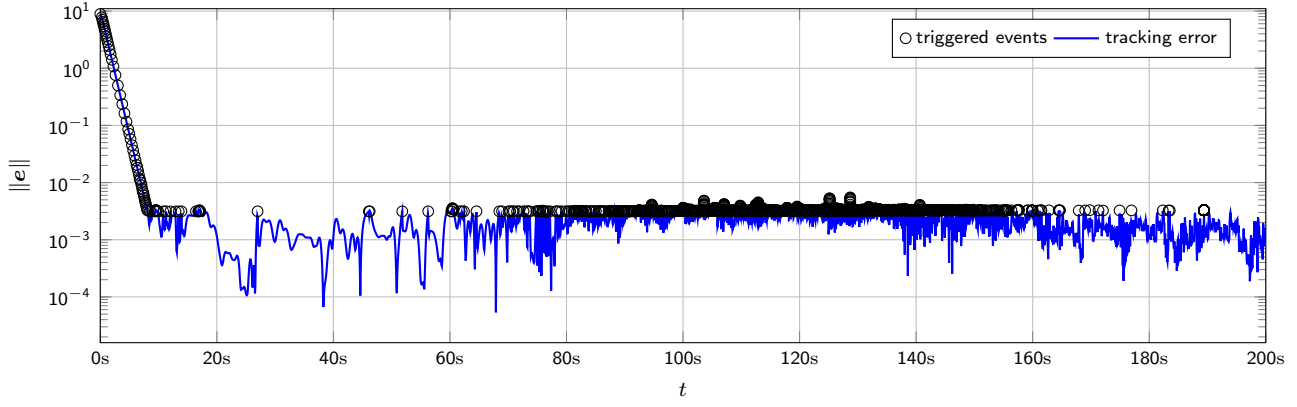


Figure 8. An event for learning can be triggered when the error exceeds the threshold prescribed by $\eta(\tau)$, which happens frequently in the transition period at the beginning and when changing the radius of the reference trajectory. Once a stationary behavior has been reached, very few events are triggered.

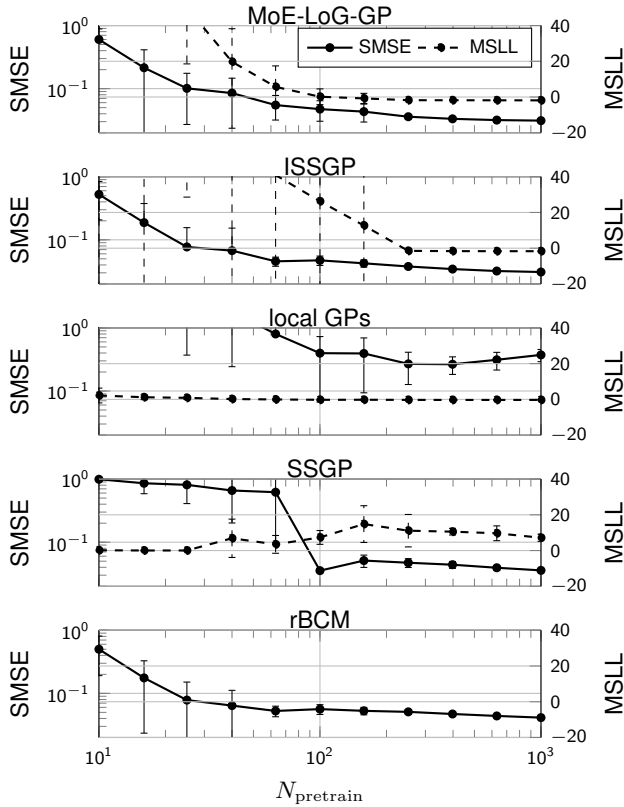


Figure 9. Comparison of the methods regarding their dependency on the quality of hyperparameters resulting from hyperparameter optimization with N_{pretrain} samples. The standard deviations are depicted via the error bars. The SMSE of all methods suffers from poor hyperparameters similarly. The MSL exhibits strongly different behavior for the methods: local GPs show almost constant MSL values, while the curves are decreasing for ISSGPs and LoG-GPs with slight advantages for the latter. The values for SSGPs and rBCMs are generally large, but keep decreasing for large values of N_{pretrain} .

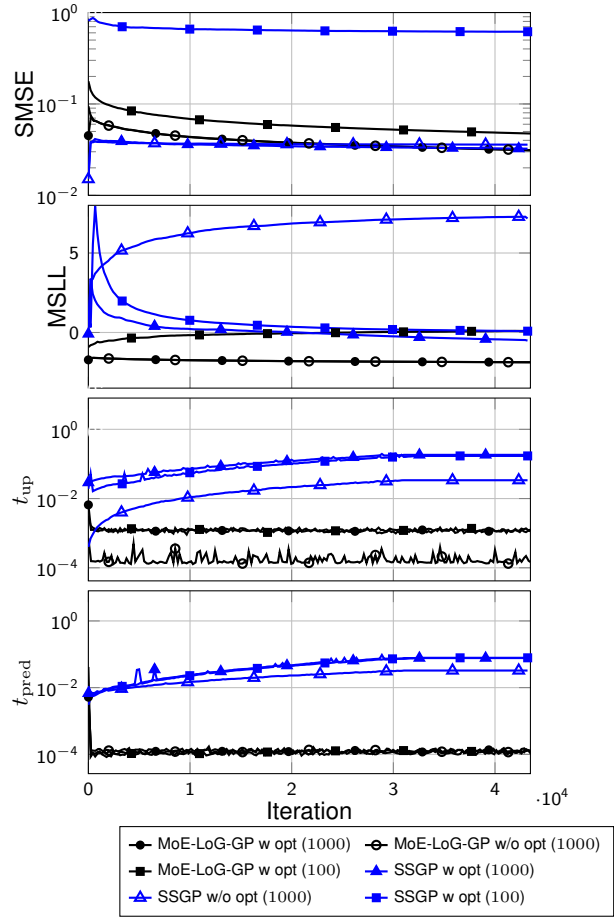


Figure 10. Online hyperparameter optimization has a weak impact on the SMSE. The MSL of the SSGP improves through the hyperparameter optimization, but it remains significantly worse than the MSL of LoG-GP approaches. When the number of pretraining samples is reduced to 100, both approaches suffer from a deteriorated prediction performance. While the prediction time barely changes for both methods, the update time increases approximately by a factor of 10. The computation time of LoG-GPs remains similar to state of the art methods, while SSGPs become more than 50 times slower than existing methods.

performance, particularly in low data regimes. Therefore, we compare the impact of hyperparameters on the different methods in this section. For this comparison, we determine the average SMSE and MSLL values using a varying number of samples N_{pretrain} for hyperparameter optimization. The results of this procedure are depicted in Fig. 9. A clear correlation between the quality of hyperparameters indicated by N_{pretrain} and the SMSE as well as MSLL values can be observed for most methods, even though some approaches such as LoG-GPs are more robust to poor hyperparameters.

This is a consequence of the local activity of GP models. The division of the input domain has a similar effect as reducing the length scales when more data becomes available. This approach has been proposed in (Berkenkamp et al., 2019) in order to achieve no regret in Bayesian optimization with unknown hyperparameters. While an improved robustness against poor hyperparameters can also be ensured for other methods such as ISSGPs (Lu et al., 2020), this comes at the price of additional computational complexity.

When slower prediction and update rates are sufficient in an application, an online hyperparameter optimization can be straightforwardly added to any LoG-GP approach. For example, we can perform a single gradient step for log-likelihood maximization of an individual model after a data point is added. The performance resulting from this online hyperparameter optimization for MoE-LoG-GPs on the SARCOS training set with $N_{\text{pretrain}} = 1000$ and $N_{\text{pretrain}} = 100$ is illustrated and compared to SSGPs with the online hyperparameter optimization proposed by Bui et al. (2017) in Fig. 10. It can be clearly seen that for $N_{\text{pretrain}} = 1000$ the MSLL of SSGPs benefits strongly from the online hyperparameter optimization, while the SMSE is barely affected. The regression accuracy and the quality of the predictive distributions of LoG-GPs remains almost unchanged. Moreover, both methods exhibit an inferior regression performance when the number of pre-training samples is reduced to $N_{\text{pretrain}} = 100$ despite the online hyperparameter training. Overall, LoG-GPs still yield lower SMSE and MSLL values with online hyperparameter optimization, but the gap between SSGPs and LoG-GPs becomes smaller.

While the hyperparameter optimization has a beneficial effect on the quality of the predictive distributions of SSGPs, it causes a significant increase in average update times for both methods (103.7 ms for SSGP, 1.2 ms for LoG-GP). While the overall computation time for the LoG-GP remains comparable to other methods without online hyperparameter optimization such as ISSGPs, SSGPs are more than 50 times slower. This underlines that disabling the hyperparameter learning can be crucial to realize model updates at high rates.

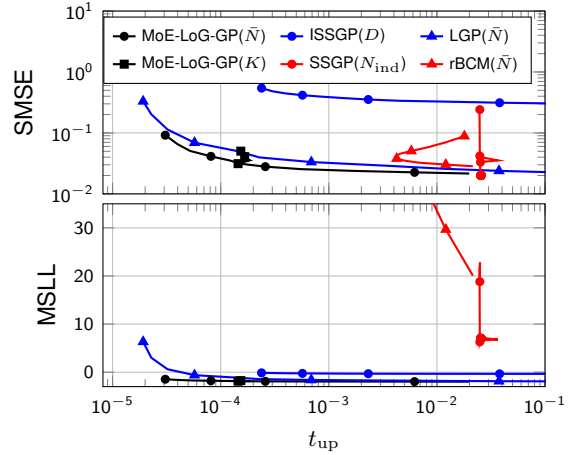


Figure 11. Given a desired update time, LoG-GPs exhibit the lowest SMSE and MSLL values on the SARCOS data set, indicating a beneficial performance-computation time trade-off. The performance and update time dependency of LoG-GPs on the number of children per node K is rather small.

Remark C.1. Since LoG-GPs have the same principled structure as distributed GPs proposed by Deisenroth & Ng (2015), the hyperparameter optimization approach can be employed to optimize the hyperparameters of LoG-GPs. If a separate process is spawned to perform the required optimization after batches of data while data is continuously added to the LoG-GP, existing hyperparameter optimization approaches can be executed online without crucial impact on the computational complexity. This approach allows a batch adaptation of hyperparameters similar to SSGPs.

C.4. Complexity-Performance Trade-Off

LoG-GP approaches do not only provide a beneficial performance with a certain parameterization, they seem to provide a generally advantageous complexity-performance trade-off. In order to demonstrate this, we compare gPoE-LoG-GPs, local GPs and rBCMs with $\tilde{N} \in [10, 1000]$, ISSGPs with $D \in [10, 1000]$, SSGPs⁸ with $N_{\text{ind}} \in [10, 1000]$. Additionally, we investigate the impact of the number of child nodes K on gPoE-LoG-GPs. Since \tilde{N} , N_{ind} , K and D are crucial for the computation time of the learning algorithms, this allows us to plot the regression performance against the computation time, as illustrated in Figs. 11 and 12. When comparing the SMSE and MSLL values for similar update times, it becomes clear that LoG-GPs exhibit a superior performance over the whole displayed range of computation times. This holds similarly for the prediction times, even though ISSGPs yield a slightly better SMSE for computation times smaller than 10^{-4} s. Therefore, LoG-GP approaches offer a beneficial performance-complexity trade-

⁸While 20 random data permutations are generally used for computing the curves for comparison, this takes too much computation time with SSGPs. Hence, the SSGP curves have been determined using 5 random permutations.

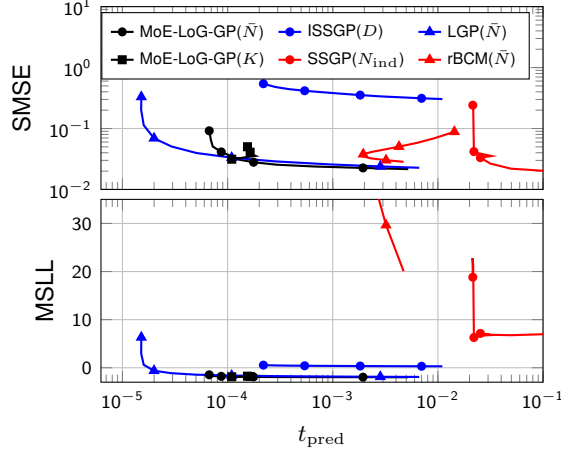


Figure 12. Given a desired prediction time, the LoG-GPs yield better MSL values on the SARCOS data set than the other methods. This also holds for the SMSE for prediction times larger than 10^{-4} s. The prediction time dependency of LoG-GPs on K is small.

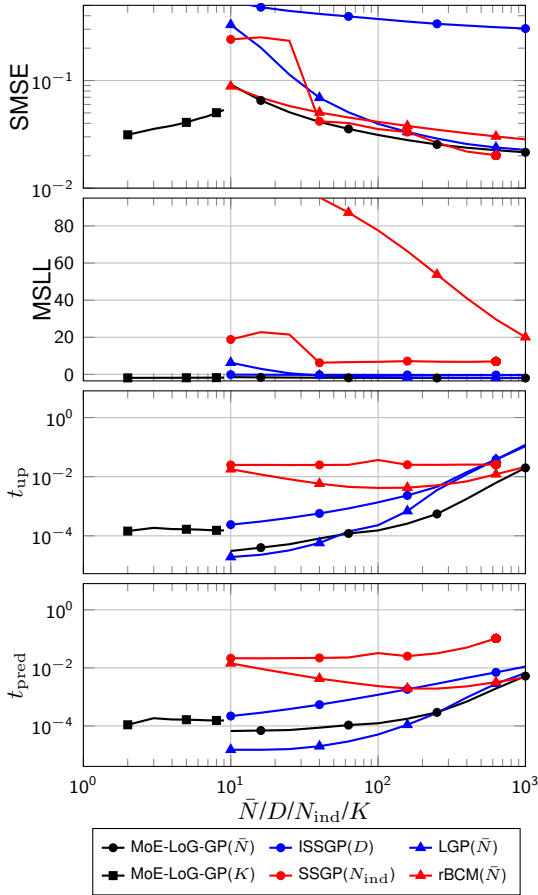


Figure 13. Comparison of the parameter dependency of the different learning methods. The SMSE values of LoG-GPs improve when more data points \bar{N} are allowed in each model, but the computation times grow. Other methods exhibit a similar behavior.

off compared to the other methods on the SARCOS data set.

In addition to the comparison of the performance complexity trade-off, we analyze the impact of the previously described parameters on performance and computation times individually. This comparison is illustrated in Fig. 13, demonstrating that all methods depend on the considered parameters in a similar way. Larger values lead to lower prediction errors and better predictive distributions, but come along with higher computation times. The number of children K is an exception to this observation. The prediction error slightly increases with growing K , while the other criteria are barely affected.

References

- Angelis, E., Wenk, P., Schölkopf, B., Bauer, S., and Krause, A. SLEIPNIR: Deterministic and Provably Accurate Feature Expansion for Gaussian Process Regression with Derivatives. 2020. URL <https://arxiv.org/pdf/2003.02658.pdf>.
- Berkenkamp, F., Schoellig, A. P., and Krause, A. No-Regret Bayesian Optimization with Unknown Hyperparameters. *Journal of Machine Learning Research*, 20:1–24, 2019.
- Bijl, H., van Wingerden, J. W., B. Schön, T., and Verhaegen, M. Online Sparse Gaussian Process Regression using FITC and PITC Approximations. *IFAC-PapersOnLine*, 48(28):703–708, 2015.
- Bijl, H., Schön, T. B., van Wingerden, J.-W., and Verhaegen, M. System Identification through Online Sparse Gaussian Process Regression with Input Noise. *IFAC Journal of Systems and Control*, 2:1–11, 2017.
- Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, New York, NY, 2006.
- Bui, T. and Turner, R. Tree-Structured Gaussian Process Approximations. In *Advances in Neural Information Processing Systems*, pp. 2213–2221, 2014.
- Bui, T. D., Nguyen, C. V., and Turner, R. E. Streaming Sparse Gaussian Process Approximations. In *Advances in Neural Information Processing Systems*, pp. 3300–3308, 2017.
- Camoriano, R., Traversaro, S., Rosasco, L., Metta, G., and Nori, F. Incremental Semiparametric Inverse Dynamics Learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 544–550, 2016.
- Cao, Y. and Fleet, D. J. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process

- Predictions, 2014. URL <http://arxiv.org/abs/1410.7827>.
- Cheng, C. A. and Boots, B. Incremental Variational Sparse Gaussian Process Regression. In *Advances in Neural Information Processing Systems*, pp. 4410–4418, 2016.
- Chowdhury, S. R. and Gopalan, A. On Kernelized Multi-armed Bandits. In *Proceedings of the International Conference on Machine Learning*, pp. 844–853, 2017.
- Csató, L. and Opper, M. Sparse On-line Gaussian Processes. *Neural Computation*, 14(3):641–668, 2002.
- Curi, S., Berkenkamp, F., and Krause, A. Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning. In *Advances in Neural Information Processing Systems*, 2020.
- Deisenroth, M. P. and Ng, J. W. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, pp. 1481–1490, 2015.
- Devroye, L. Universal Limit Laws for Depths in Random Trees. *SIAM Journal on Computing*, 28(2):409–432, 1998.
- Duvenaud, D. K. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.
- Fiedler, C., Scherer, C. W., and Trimpe, S. Practical and Rigorous Uncertainty Bounds for Gaussian Process Regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.
- Gal, Y. and Turner, R. Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs. In *Proceedings of the International Conference on Machine Learning*, pp. 655–664, 2015.
- Gijsberts, A. and Metta, G. Real-Time Model Learning using Incremental Sparse Spectrum Gaussian Process Regression. *Neural Networks*, 41:59–69, 2013.
- Gramacy, R. B. and Lee, H. K. H. Bayesian Treed Gaussian Process Models with an Application to Computer Modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.
- Harrison, J., Sharma, A., and Pavone, M. Meta-Learning Priors for Efficient Online Bayesian Regression. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2018.
- Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian Processes for Big Data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.
- Huber, M. F. Recursive Gaussian Process: On-line Regression and Learning. *Pattern Recognition Letters*, 45(1): 85–91, 2014.
- Koppel, A. Consistent Online Gaussian Process Regression Without the Sample Complexity Bottleneck. In *Proceedings of the American Control Conference*, pp. 3512–3518, 2019.
- Lázaro-Gredilla, M., Quiñero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.
- Le, T., Nguyen, K., Nguyen, V., Nguyen, T. D., and Phung, D. GoGP: Fast Online Regression with Gaussian Processes. In *Proceedings of the IEEE International Conference on Data Mining*, pp. 257–266, 2017.
- Lederer, A., Umlauf, J., and Hirche, S. Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control. In *Advances in Neural Information Processing Systems*, pp. 659–669, 2019a.
- Lederer, A., Umlauf, J., and Hirche, S. Posterior Variance Analysis of Gaussian Processes with Application to Average Learning Curves. 2019b. URL <http://arxiv.org/abs/1906.01404>.
- Liberzon, D. *Switching in Systems and Control*. Springer Science Business Media, 2003.
- Liu, H., Cai, J., Wang, Y., and Ong, Y. S. Generalized Robust Bayesian Committee Machine for Large-Scale Gaussian Process Regression. In *Proceedings of the International Conference on Machine Learning*, 2018.
- Liu, H., Ong, Y. S., Shen, X., and Cai, J. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4405–4423, 2020.
- Liu, Z., Zhou, L., Leung, H., and Shum, H. P. Kinect Posture Reconstruction Based on a Local Mixture of Gaussian Process Models. *IEEE Transactions on Visualization and Computer Graphics*, 22(11):2437–2450, 2016.
- Lu, Q., Karanikolas, G., Shen, Y., and Giannakis, G. B. Ensemble Gaussian Processes with Spectral Features for On-line Interactive Learning with Scalability. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1910–1920, 2020.
- Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrà, P., Ghahramani, Z., and Hensman, J. GPflow: A Gaussian Process Library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

- Meier, F. and Schaal, S. Drifting Gaussian Processes with Varying Neighborhood Sizes for Online Model Learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 264–269. IEEE, 2016.
- Mutný, M. and Krause, A. Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features. In *Advances in Neural Information Processing Systems*, pp. 9005–9016, 2018.
- Ng, J. W. and Deisenroth, M. P. Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression. 2014. URL <http://arxiv.org/abs/1412.3078>.
- Nguyen-Tuong, D. and Peters, J. Incremental Sparsification for Real-Time Online Model Learning. *Journal of Machine Learning Research*, 9:557–564, 2010.
- Nguyen-Tuong, D., Seeger, M., and Peters, J. Model Learning with Local Gaussian Process Regression. *Advanced Robotics*, 23(15):2015–2034, 2009a.
- Nguyen-Tuong, D., Seeger, M., and Peters, J. Local Gaussian Process Regression for Real Time Online Model Learning and Control. In *Advances in Neural Information Processing Systems*, pp. 1193–1200, 2009b.
- Omohundro, S. M. Five Balltree Construction Algorithms. *Science*, 51(1):1–22, 1989.
- Pleiss, G., Gardner, J. R., Weinberger, K. Q., and Wilson, A. G. Constant-Time Predictive Distributions for Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, pp. 6575–6584, 2018.
- Rahimi, A. and Recht, B. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems*, pp. 1–8, 2008.
- Ranganathan, A. and Yang, M.-h. Online Sparse Matrix Gaussian Process Regression and Vision Applications. In *Proceedings of the European Conference on Computer Vision*, pp. 468–482, 2008.
- Ranganathan, A., Yang, M. H., and Ho, J. Online Sparse Gaussian Process Regression and its Applications. *IEEE Transactions on Image Processing*, 20(2):391–404, 2011.
- Rulli  re, D., Durrande, N., Bachoc, F., and Chevalier, C. Nested Kriging Predictions for Datasets with a Large Number of Observations. *Statistics and Computing*, 28(4):849–867, 2018.
- Sch  lkopf, B. and Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Cambridge, Massachusetts, 2002.
- Schreiter, J., Nguyen-Tuong, D., and Toussaint, M. Efficient Sparsification for Gaussian Process Regression. *Neurocomputing*, 192:29–37, 2016.
- Shen, Y., Ng, A. Y., and Seeger, M. Fast Gaussian Process Regression using KD-Trees. In *Advances in Neural Information Processing Systems*, 2006.
- Snelson, E. and Ghahramani, Z. Local and Global Sparse Gaussian Process Approximations. *Journal of Machine Learning Research*, 2:524–531, 2007.
- Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.
- Titsias, M. K. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 567–574, 2009.
- Tresp, V. A Bayesian Committee Machine. *Neural Computation*, 12:2719–2741, 2000.
- Tresp, V. Mixtures of Gaussian Processes. In *Advances in Neural Information Processing Systems*, 2001.
- Umlauf, J., Beckers, T., Capone, A., Lederer, A., and Hirche, S. Smart Forgetting for Safe Online Learning with Gaussian Processes. In *Learning for Dynamics & Control*, pp. 1–10, 2020.
- van der Wilk, M. *Sparse Gaussian Process Approximations and Applications*. PhD thesis, University of Cambridge, 2018. URL <https://markvdw.github.io/vanderwilk-thesis.pdf>.
- Vasudevan, S., Ramos, F., Nettleton, E., Durrant-Whyte, H., and Blair, A. Gaussian Process Modeling of Large Scale Terrain. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1047–1053, 2009.
- Vivarelli, F. *Studies on the Generalisation of Gaussian Processes and Bayesian Neural Networks*. PhD thesis, Aston University, 1998.
- Williams, C. K. I. and Vivarelli, F. Upper and Lower Bounds on the Learning Curve for Gaussian Processes. *Machine Learning*, 40:77–102, 2000.
- Wilson, A. G. and Nickisch, H. Kernel interpolation for Scalable Structured Gaussian Processes. *Proceedings of the International Conference on Machine Learning*, 3: 1775–1784, 2015.