Gaussian Process-Based Real-Time Learning for Safety Critical Applications

Armin Lederer [1] **Alejandro José Ordóñez Conejo** [2] **Korbinian Maier** [1] **Wenxin Xiao** [3] **Jonas Umlauft** [1] **Sandra Hirche** [1]

Abstract

The safe operation of physical systems typically relies on high-quality models. Since a continuous stream of data is generated during run-time, such models are often obtained through the application of Gaussian process regression because it provides guarantees on the prediction error. Due to its high computational complexity, Gaussian process regression must be used offline on batches of data, which prevents applications, where a fast adaptation through online learning is necessary to ensure safety. In order to overcome this issue, we propose the LoG-GP. It achieves a logarithmic update and prediction complexity in the number of training points through the aggregation of locally active Gaussian process models. Under weak assumptions on the aggregation scheme, it inherits safety guarantees from exact Gaussian process regression. These theoretical advantages are exemplarily exploited in the design of a safe and data-efficient, online-learning control policy. The efficiency and performance of the proposed real-time learning approach is demonstrated in a comparison to state-of-the-art methods.

1. Introduction

Recent technological trends enable an increasing autonomy of physical systems, often operating in uncertain and dynamically changing environments. In order to ensure safety and high performance of these systems, they need the ability to quickly adapt to new situations by inferring mathematical models from observed data. Especially in control applications, predictions and model updates must typically be performed in real-time due to the fast evolution of many physical processes. These applications include the control of autonomous cars (Kendall et al., 2019), unmanned aerial vehicles (Andersson et al., 2017), robotic manipulators (Nguyen-Tuong & Peters, 2010), combustion engines (Lee et al., 2017), and many others, where update rates in the magnitude of 10^2 Hz to 10^4 Hz are required. In case of predictive control schemes, where possible future trajectories are inferred and evaluated, multiple predictions are made for a single control command, requiring prediction rates, which are orders of magnitudes higher (Kong et al., 2015).

A common supervised machine learning technique in safety critical applications are Gaussian processes (GPs), which provide a high expressive power, and guarantee probabilistically bounded prediction errors (Rasmussen & Williams, 2006). Since the computational complexity of updates and predictions grows strongly with the number of training points, many approximations have been developed to enable the employment in real-time applications. Deterministic training conditional approximations (Nguyen-Tuong & Peters, 2010; Schreiter et al., 2016) and inducing point methods (Huber, 2014; Bijl et al., 2017) can speed up predictions, while variational inference approaches for streaming data (Bui et al., 2017) allow fast model updates and exhibit a beneficial performance-complexity trade-off compared to stochastic variational inference (Hensman et al., 2013). However, error bounds from exact GP inference as derived by, e.g., Srinivas et al. (2012); Lederer et al. (2019a), do not extend to these methods, which prevents the usage in safety critical applications. Even though finite feature approximations of kernels (Gijsberts & Metta, 2013) are advantageous in this regard and yield constant update and prediction complexities, safety guarantees require an impractically high number of features (Mutný & Krause, 2018). Therefore, there is a clear lack of methods which allow updates and predictions in real-time for safety critical applications.

The main contribution of this paper is a novel, computationally efficient, GP-based method for real-time predictions and model updates in safety critical applications, called locally growing random tree of GPs (LoG-GP). Based on distributed Gaussian processes (Deisenroth & Ng, 2015), we propose an iterative random division of individual models, which results in a random tree as computation graph and guarantees logarithmic complexity of model updates. In order to reduce the complexity of predictions, the number of

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

Proceedings of the 38^{th} *International Conference on Machine Learning*, PMLR 139, 2021.

necessary individual GP evaluations is limited through the application of locally active models. We prove that uniform error bounds from exact GP inference directly carry over to the proposed method, such that it can be used in safety critical applications. This is demonstrated through the application of LoG-GPs in a data-efficient, online-learning control scheme, where we prove a bounded control error. In a comparison on real-world data sets and a control simulation, the superior computational efficiency is demonstrated while providing comparable regression performance to state-of-the-art methods.

The remainder of this paper is structured as follows: In Section 2, distributed GPs are briefly introduced and the considered problem is stated. Section 3 presents the proposed LoG-GP method, which is used in Section 4 to design a safe, real-time learning control policy. The proposed methods are compared to state-of-the-art techniques in Section 5.

2. Problem Set-up and Objective

We consider a real-time regression problem $y = f(\boldsymbol{x}) + \epsilon \in \mathbb{R}$, where $\boldsymbol{x} \in \mathbb{R}^d$ and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, $\sigma_n^2 \in \mathbb{R}_+$. The objective is to iterate between updating a model $\hat{f}(\cdot)$ of the unknown function $f(\cdot)$ based on sequentially arriving training pairs $(\boldsymbol{x}^{(i)}, y^{(i)})$, $i = 1, \dots, \infty$ and evaluating $\hat{f}(\cdot)$ at arbitrary test points $\boldsymbol{x}$. When no parametric structure of $f(\cdot)$ is known, Gaussian processes are a suitable choice for non-parametric probabilistic regression.

A Gaussian process $\mathcal{GP}$ is the generalization of a Gaussian distribution, and bases on the assumption that any finite collection of random variables $y^{(i)} \in \mathbb{R}$ follows a joint Gaussian distribution. This distribution is defined by the prior mean, which is commonly set to 0, and a covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (Rasmussen & Williams, 2006). We concatenate input training samples $\boldsymbol{x}^{(i)}$ and outputs $y^{(i)}$, $i = 1, \dots, N$, into a matrix[1] $\boldsymbol{X}$ and a vector $\boldsymbol{y}$, which represent the training data set $\mathbb{D}$. Furthermore, we define the elements of the kernel matrix $\boldsymbol{K}$ as $K_{ij} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ and define the elements of the kernel vector $\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x})$ accordingly. Based on these definitions, we can represent the GP model as

$$\boldsymbol{L} = \text{cholesky}(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}), \qquad \boldsymbol{\alpha} = \boldsymbol{L}^T \setminus (\boldsymbol{L} \setminus \boldsymbol{y}), \tag{1}$$

where "\" denotes the forward and backward substitution, respectively, such that $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ operations are required for the computation of $\boldsymbol{L}$ and $\boldsymbol{\alpha}$ (Rasmussen & Williams, 2006), respectively. Moreover, when samples are added online to the training set, i.e., $\boldsymbol{X}_{N+1} = [\boldsymbol{X}_N \ \boldsymbol{x}^{(N+1)}]$, $\boldsymbol{y}_{N+1} = [\boldsymbol{y}_N^T \ y^{(N+1)}]^T$, $\boldsymbol{L}_{N+1}$ can be directly obtained from $\boldsymbol{L}_N$ using rank-one updates in $\mathcal{O}(N^2)$ operations. The posterior GP distribution $p_{\mathcal{GP}}(f(\boldsymbol{x}) | \boldsymbol{x}, \mathbb{D}) = \mathcal{N}(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x}))$ at a test point $\boldsymbol{x}$ can finally be computed using

$$\mu(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{\alpha} \tag{2}$$

$$\sigma^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{v}^T\boldsymbol{v}, \qquad \boldsymbol{v} = \boldsymbol{L} \setminus \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}), \tag{3}$$

which requires $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ calculations for the posterior mean and variance, respectively.

Although GP inference allows incremental updates, the streaming data quickly accumulates to large data sets in real-time learning problems, slowing down the computation of updates and predictions significantly, such that the total number of training samples for straightforward inference is roughly limited to 10^4 training samples in practice on today's machines (Deisenroth & Ng, 2015). A common approach to deal with the problems arising from large data sets lies in dividing the data into several sets and training individual models $\mathcal{GP}_i$ with means $\mu_i(\cdot)$ and variances $\sigma_i^2(\cdot)$ defined through (2) and (3), respectively. For aggregating the individual predictions, several different methods exist, whose structure can be generalized to

$$\tilde{\mu}(\boldsymbol{x}) = \phi_\mu \left(\sum_{m \in \mathbb{M}} \omega_m \psi_\mu(\mu_m(\boldsymbol{x}), \sigma_m^2(\boldsymbol{x})) \right) \tag{4}$$

$$\tilde{\sigma}^2(\boldsymbol{x}) = \phi_\sigma \left(\sum_{m \in \mathbb{M}} \omega_m \psi_\sigma(\mu_m(\boldsymbol{x}), \sigma_m^2(\boldsymbol{x})) \right), \tag{5}$$

where ω_m are weighting factors, which should be chosen such that $\sum_{m \in \mathbb{M}} \omega_m = 1$, $\mathbb{M}$ denotes the index set of the individual models and $\phi_\mu, \phi_\sigma : \mathbb{R} \to \mathbb{R}$ and $\psi_\mu, \psi_\sigma : \mathbb{R}^2 \to \mathbb{R}$ are nonlinear functions. For example, a mixture of GP experts approach (Tresp, 2001) corresponds to

$$\tilde{\mu}(\boldsymbol{x}) = \sum_{m \in \mathbb{M}} \omega_m \mu_m(\boldsymbol{x}) \tag{6}$$

$$\tilde{\sigma}^2(\boldsymbol{x}) = \sum_{m \in \mathbb{M}} \omega_m (\sigma_m^2(\boldsymbol{x}) + \mu_m^2(\boldsymbol{x})) - \tilde{\mu}^2(\boldsymbol{x}), \tag{7}$$

which is often used in the form of a mixture of explicitly localized experts (Masoudnia & Ebrahimpour, 2014), see, e.g., Nguyen-Tuong et al. (2009b); Liu et al. (2016). Similarly, the generalized product of GP experts approach (Cao & Fleet, 2014) can be obtained by choosing

$$\tilde{\mu}(\boldsymbol{x}) = \sum_{m \in \mathbb{M}} \omega_m \frac{\tilde{\sigma}^2(\boldsymbol{x})}{\sigma_m^2(\boldsymbol{x})} \mu_m(\boldsymbol{x}) \tag{8}$$

$$\tilde{\sigma}^2(\boldsymbol{x}) = \frac{1}{\sum_{m \in \mathbb{M}}^{M} \omega_m \sigma_m^{-2}(\boldsymbol{x})}. \tag{9}$$

While these aggregation approaches allow to regress large data sets and exhibit many advantages compared to inducing point methods (Deisenroth & Ng, 2015), they do not deal with the specific difficulties of applying GPs to real-time

[1] We generally omit the indication of the number of samples in the notation, but when necessary for clarity, an index N is used.

learning problems. The complexity of computing predictions still grows linearly with the number of individual models, and the assignment of streaming data to individual models is often not investigated (Deisenroth & Ng, 2015), or becomes inefficient for large data sets and requires further approximations (Nguyen-Tuong et al., 2009b). Moreover, the existence of probabilistic error bounds for these methods is unclear, such that their usage in real-time learning for safety-critical applications remains a challenge.

3. Locally Growing Random Trees with Gaussian Process Models

In order to address these issues of GP aggregation methods, we develop an efficient alternative for an iterative data distribution to individual models such that predictions are computed based on local data, allowing both updates and predictions with logarithmic complexity. Starting from a single, global model, local models are iteratively generated by dividing existing models. This is efficiently performed by sampling the model, to which each training sample is assigned, from localizing random distributions. Thereby, we locally grow a random tree of GP models. We explain this iterative tree construction using random data assignment in detail in Section 3.1. In Section 3.2, we demonstrate how the LoG-GP approach naturally extends existing distributed GP approaches to real-time problems. We derive complexity guarantees for LoG-GPs in Section 3.3, and provide uniform error bounds for the LoG-GP regression in Section 3.4.

3.1. Iterative Random Tree Construction

Since we consider the problem of real-time regression, we have to deal with streaming data, i.e., sequentially arriving data samples. Therefore, we iteratively construct a model, starting with a single data set $\mathbb{D}_1 = \emptyset$. This data set constitutes the root node 1 of a rooted tree T, as depicted in Fig. 1. Incoming training data is added to the data set $\mathbb{D}_1$, and each new data point can be efficiently included into the single GP model (1) using rank one updates, which exhibit quadratic complexity (Nguyen-Tuong et al., 2009a). When the number of training samples reaches a prescribed threshold $\bar{N}$, we extend the tree T by growing leaf nodes $1, \dots, K$, $K \in \mathbb{N}_+$ with data sets $\mathbb{D}_2, \dots, \mathbb{D}_{K+1}$ as children of the root node 1, as shown in the center of Fig. 1. In order to distribute the data efficiently to the sets $\mathbb{D}_2, \dots, \mathbb{D}_{K+1}$, we define a function $\boldsymbol{p}^1 : \mathbb{R}^d \to [0,1]^K$, $\sum_{k=1}^K p_k^1(\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in \mathbb{R}^d$, which returns the probability of an assignment of a point $\boldsymbol{x} \in \mathbb{R}^d$ to the sets $\mathbb{D}_{k+1}$, $k = 1, \dots, K$, i.e., $P(\boldsymbol{x} \in \mathbb{D}_{k+1}) = p_k^1(\boldsymbol{x})$. We determine the probabilities $\boldsymbol{p}^1(\boldsymbol{x})$ for each data pair $(\boldsymbol{x}, y)$ in $\mathbb{D}_1$, and sample the child nodes n from the corresponding discrete probability distributions. After the data set division, we compute the local GP models (1) for all data sets, which generally has a complexity of $\mathcal{O}(\bar{N}^3)$ (Rasmussen & Williams, 2006).

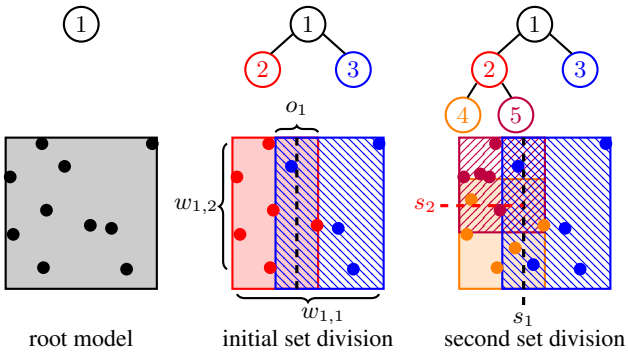

Figure 1. Iterative model tree construction and corresponding layout of the input space for $K = 2$: active regions and training samples belonging to the same node are depicted in the same color.

Algorithm 1 Updating of a LoG-GP

```
UPDATE(K-ary tree T, training input x, target y)
n ← T.ROOT()
while ¬T.ISLEAF(n) do
    n ← n.GETCHILD(RANDOMDRAW(p^n(x)))
end while
if |D_n| = N̄ then
    n.GENERATECHILDREN(K)
    for each (x', y') ∈ D_n do
        m ← n.GETCHILD(RANDOMDRAW(p^n(x')))
        m.ADDTODATASET(x', y')
        m.UPDATELOCALGP()
    end for
    n ← n.GETCHILD(RANDOMDRAW(p^n(x')))
end if
n.ADDTODATASET(x, y)
n.UPDATELOCALGP()
return T
```

After the initial data set division, we continue to assign the streaming data pairs $(\boldsymbol{x}, y)$ to the sets $\mathbb{D}_2, \dots, \mathbb{D}_{K+1}$ by sampling from the discrete distributions with parameters $\boldsymbol{p}^1(\boldsymbol{x})$. When either of the sets $\mathbb{D}_1, \dots, \mathbb{D}_{K+1}$ reaches its data capacity limit $\bar{N}$, we define a new function $\boldsymbol{p}^{k+1}(\cdot)$, $k = 1, \dots, K$, such that it induces the conditional probabilities given the parent node, e.g., $P(\boldsymbol{x} \in \mathbb{D}_{K+2} | \boldsymbol{x} \in \mathbb{D}_2) = p_1^2(\boldsymbol{x})$. Based on this conditional probability, we repeat the division process as explained for the root node. Therefore, we add another level to the random tree as shown on the right-hand side of Fig. 1. For further training data assignment, it is necessary to iteratively determine a branch of the tree using random transitions based on the discrete distributions with probability parameters $\boldsymbol{p}^n(\boldsymbol{x})$ until a leaf node is reached, as outlined in Algorithm 1.

Note that the nodes n, which are not leaves, contain neither data nor a local GP model after the data set division, but

instead encode the structure of the data distribution to individual data sets using the discrete distributions $\boldsymbol{p}^n(\cdot)$. Therefore, $\boldsymbol{p}^n(\cdot)$ are crucial design choices of the LoG-GP approach. Intuitively, they should be chosen such that the data is distributed equally to all children in order to generate a balanced tree, and this condition indeed guarantees a logarithmic growth in complexity for the random tree construction as shown in Section 3.3. A trivial example for a probability distribution satisfying this requirement is the discrete uniform distribution, i.e., $p_k^n(\boldsymbol{x}) = 1/K$, which can be seen as the sequential version of the commonly used random assignment in batch aggregation methods (Cao & Fleet, 2014; Deisenroth & Ng, 2015).

3.2. Predicting using Localizing Probability Functions

Although the random tree construction in Algorithm 1 reduces the complexity of updates, it barely affects the complexity of predictions, since the direct evaluation of (4) and (5) with $\mathbb{M}$ denoting the set of leaf nodes still exhibits a linear complexity in the number of training samples. In order to achieve a low complexity of predictions as well, we propose to enforce a small number of active models at each input using the remaining design parameters ω_m. Since a typical condition for these parameters requires their sum to equal one, a straightforward choice is to set them equal to the marginal probabilities $P(\boldsymbol{x} \in \mathbb{D}_m)$ of the leaf nodes, i.e., $\omega_m = P(\boldsymbol{x} \in \mathbb{D}_m)$. The marginal probabilities of a leaf m with depth h^m can be determined by multiplying the conditional probabilities on the branch $\mathbb{B}_m = \{(s_1^m, b_1^m), \dots, (s_{h^m}^m, b_{h^m}^m)\}$, where $s_1^m = 1$, $b_i^m = 1, \dots, K$ denotes the child index of the subsequent node and s_i^m denotes the sequence of nodes before reaching leaf m. Therefore, we can express the marginal probability of a leaf as

$$\omega_m = \prod_{i=1}^{h^m} p_{s_i^m}^{b_i^m}(\boldsymbol{x}). \tag{10}$$

It is straightforward to see that the computation of a marginal probability requires only local information of nodes along the branch, but is independent of other branches. This independence of the branches is the keystone for a reduction of the computational complexity of predictions: a conditional probability $p_{s_i^m}^{b_i^m}(\boldsymbol{x}) = 0$ allows to omit determining the following conditional probabilities $p_{s_j^m}^{b_j^m}(\boldsymbol{x})$, $j > i$, since $\omega_m = 0$ holds regardless of their values. Together with the structure (4), (5), this allows to spare the computation of individual GP predictions with a zero conditional probability on the branch, which can be efficiently exploited through recursive tree search algorithms as depicted in Algorithm 2.

Due to this property, the conditional probabilities $\boldsymbol{p}^n(\boldsymbol{x})$ effectively control the computational complexity of predictions: when only few children of every node can have a positive conditional probability $p_k^n(\boldsymbol{x}) > 0$, the recursion can often stop early and only few individual GP predictions have to be performed. Thus, the maximum number of active children with $p_k^n(\boldsymbol{x}) > 0$ should be kept small in each node n in order to achieve a low computational complexity. This in turn induces a notion of proximity of points $\boldsymbol{x}$, in which two points $\boldsymbol{x}, \boldsymbol{x}'$ can be considered close to each other if $p_k^n(\boldsymbol{x}) > 0$ and $p_k^n(\boldsymbol{x}') > 0$. Hence, the conditional probabilities can be considered as localizing probability functions.

Algorithm 2 Predicting with a LoG-GP

```
PREDICT(binary tree T, test input x, node n)
if T.ISLEAF(n) then
    return μ_n(x), σ_n^2(x), 1
else
    μ ← [ ], σ^2 ← [ ], ω ← [ ]
    for all j ∈ {i = 1, ..., K : p_i^n(x) > 0} do
        μ̂, σ̂^2, ω̂ ← PREDICT(T, x, n.GETCHILD(j))
        μ ← [μ  μ̂], σ^2 ← [σ^2  σ̂^2], ω ← [ω  p_j^n(x) ω̂]
    end for
    if ¬T.ISROOT(n) then
        return μ, σ^2, ω
    else
        return μ̃(x), σ̃^2(x) based on (4), (5)
    end if
end if
```

A simple class of conditional probabilities $\boldsymbol{p}^n(\boldsymbol{x})$ inducing spatial locality are saturating linear functions

$$\xi_k^n(\boldsymbol{x}) = \begin{cases} 0 & \text{if } x_{j_k^n} < s_k^n - \frac{o_k^n}{2} \\ \frac{x_{j_k^n} - s_k^n}{o_k^n} + \frac{1}{2} & \text{if } s_k^n - \frac{o_k^n}{2} \le x_{j_k^n} \le s_k^n + \frac{o_k^n}{2} \\ 1 & \text{if } s_k^n + \frac{o_k^n}{2} < x_{j_k^n}, \end{cases} \tag{11}$$

where j_k^n defines a splitting dimension, s_k^n denotes the position of the splitting hyperplane, and o_k^n is the overlapping ratio, which determines the size of the region in which two individual models have a non-zero probability. The interpretation of these parameters is illustrated in Fig. 1. Based on (11), the conditional probabilities can be defined as

$$p_k^n(\boldsymbol{x}) = \begin{cases} \xi_k^n(\boldsymbol{x}) \prod_{j=1, j\neq k}^{K-1} (1 - \xi_j^n(\boldsymbol{x})) & k < K \\ \prod_{j=1}^{K-1} (1 - \xi_j^n(\boldsymbol{x})) & k = K. \end{cases} \tag{12}$$

This parameterization allows straightforward heuristics for choosing the parameters j_k^n, s_k^n, o_k^n of the saturating linear functions $\xi_k^n(\cdot)$, such that the goal of equal division of existing training sets during the updating step as motivated in Section 3.1 can be approximately achieved, too. For example, one option to choose j_k^n is the maximum spread of the inputs $\boldsymbol{x}$ in the individual sets $\mathbb{D}_n$, a simple choice for s_k^n is the

mean in the dimensions j_k^n, and the overlapping ratio o_k^n can be designed as a constant fraction of the spread. Moreover, a PCA based definition of the parameters is straightforward as well (Terry & Choe, 2020). Therefore, it is easily possible to achieve the goal of a balanced tree and the desired limitation of the active number of children in each node.

Remark 3.1. *While the proposed approach can be used in combination with other regression techniques as a meta framework, the improvement in computational efficiency can be significantly smaller. However, locally growing random trees have the potential to improve the performance of many regression methods in non-stationary problems, where localization methods have been shown to be useful. Since this problem is not the focus of this work, we leave the combination of the proposed approach with other regression methods for future research.*

3.3. Complexity Guarantees

In this section, we formalize the intuitive conditions for achieving low computational complexities discussed in the previous sections. In order to define the meaning of an approximately equal splitting of data in nodes, we introduce the following assumption, which poses a condition on the relationship between the conditional probabilities $\boldsymbol{p}^n(\cdot)$ and the probability density $q(\boldsymbol{x})$ of the input training data.

Assumption 3.1. *There exists a constant $c_1 \in \mathbb{R}_+$, such that the conditional assignment probabilities $\boldsymbol{p}^n(\boldsymbol{x})$ satisfy*

$$c_1 \leq \int_{\mathbb{R}^d} q(\boldsymbol{x})\, p_{s_{h_m}^m}^{b_{h_m}^m}(\boldsymbol{x})\, \theta\left(\prod_{i=1}^{h^m-1} p_{s_i^m}^{b_i^m}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \qquad (13)$$

for all leafs $m \in \mathbb{M}$ with depths h^m and branches $\mathbb{B}_m$, where $\theta : \mathbb{R} \to \{0, 1\}$ denotes the unit step function.

The right handside of (13) corresponds to the marginal conditional probability that a training sample is assigned to leaf m, given the prior assignment to node $s_{h_m}^m$. Therefore, this assumption prevents nodes from never receiving training data. In practice, this can easily be achieved through a data-based design of the conditional probabilities as outlined in Section 3.2. Based on Assumption 3.1, it is straightforward to prove the following complexity guarantee for updates of LoG-GPs using the theory of random split trees (Devroye, 1998)[2].

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

In order to bound the complexity of predictions using LoG-GPs as well, an additional assumption on the maximum number of children with positive conditional probabilities along a branch is necessary. This is formalized as follows.

[2]Proofs for all theoretical results can be found in the appendix.

Assumption 3.2. *There exist constants $c_2, c_3 \in \mathbb{R}_+$, such that the conditional assignment probabilities $\boldsymbol{p}^n(\cdot)$ satisfy*

$$\sum_{i=1}^{h^m} \theta\left(1 - p_{s_i^m}^{b_i^m}(\boldsymbol{x})\right) \theta\left(p_{s_i^m}^{b_i^m}(\boldsymbol{x})\right) \leq \log(c_2 h^m + c_3) \qquad (14)$$

for all leaves $m \in \mathbb{M}$ with depths h^m and branches $\mathbb{B}_m$.

Since this condition can individually be checked for every branch during the generation of a new layer, it can directly be included into the specification of the conditional probabilities $\boldsymbol{p}^n(\cdot)$ during the generation of a new layer, e.g., through the adaptation of the overlapping ratio o_m in (11). Therefore, this assumption is not restrictive in practice. In combination with Assumption 3.1, it allows the following bound on the computational complexity of predictions.

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

Remark 3.2. *Although the maximum number of samples $\bar{N}$ has a strong impact on the computation time, it merely acts as a constant factor on the asymptotic complexities. Therefore, we drop it for clarity of presentation.*

3.4. Regression Error Bound

While a variety of approximations exists to reduce the complexity of GP updates and predictions, they typically cannot maintain the uniform error bounds of exact GP regression. We show that the LoG-GP approach exhibits the advantage of preserving uniform regression error bounds from exact GP inference, e.g., (Srinivas et al., 2012; Chowdhury & Gopalan, 2017; Lederer et al., 2019a; Maddalena et al., 2020). We focus here on the approach introduced in (Lederer et al., 2019a) for clarity of exposition, but other error bounds can be extended analogously as shown in the supplementary material. For this existing error bound to hold, the following assumption is introduced.

Assumption 3.3. *The unknown function $f(\cdot)$ is a sample from a Gaussian process $\mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ and observations $y = f(\boldsymbol{x}) + \epsilon$ are perturbed by zero mean i.i.d Gaussian noise ϵ with variance σ_n^2.*

This assumption defines a prior distribution over functions and implicitly assigns to each function a probability density. For example, the sample space of squared exponential kernels is the space of continuous functions on $\mathbb{X}$ (van der Vaart & van Zanten, 2011), and the hyperparameters of the kernel shape the distribution. A more detailed discussion of this assumption can be found in (Lederer et al., 2021).

In addition to the knowledge of a prior distribution, we require the following essential property of the aggregation scheme to inherit error bounds from exact GP regression.

Assumption 3.4. *The distributed GP mean can be expressed as* $\tilde{\mu}(\boldsymbol{x}) = \sum_{m\in\mathbb{M}} w_m(\boldsymbol{x})\mu_m(\boldsymbol{x})$*, where* $w_i : \mathbb{R}^d \to \mathbb{R}_{0,+}$ *is a weighting function satisfying* $\sum_{m\in\mathbb{M}} w_m(\boldsymbol{x}) = 1$, $\forall \boldsymbol{x} \in \mathbb{R}^d$.

It can be trivially checked that both the mixture of experts (6), (7) and the generalized product of experts approach (8), (9) in combination with weights ω_m following from the LoG-GP approach (10) satisfy the condition. Hence, this assumption does not severely restrict the class of possible distributed GP approaches, but allows to sum up the individual uniform error bounds for the mean functions $\mu_m(\cdot)$, which is the core idea in the following theorem.

Theorem 3.3. *Consider a distributed GP approach satisfying Assumption 3.4 and defined through the continuous covariance function* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ *with Lipschitz constant* L_k *on the compact set* $\mathbb{X} \subset \mathbb{R}^d$*. Furthermore, consider a continuous unknown function* $f : \mathbb{X} \to \mathbb{R}$ *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*

$$\beta(\tau) = 2\log\left(d^{\frac{d}{2}} \max_{\boldsymbol{x},\boldsymbol{x}'\in\mathbb{X}} \|\boldsymbol{x}-\boldsymbol{x}'\|_\infty^d |\mathbb{M}|\right) - \log\left(\delta 2^d \tau^d\right) \quad (15)$$

$$\gamma(\tau) = \sum_{m\in\mathbb{M}} w_m(\boldsymbol{x})\left(L_{\mu_m}\tau + \sqrt{\beta(\tau)} L_{\sigma_m}\tau\right) + L_f\tau, \quad (16)$$

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

$$P\left(|f(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})| \leq \eta(\tau,\boldsymbol{x}), \forall \boldsymbol{x} \in \mathbb{X}\right) \geq 1-\delta, \quad (17)$$

where

$$\eta(\tau,\boldsymbol{x}) = \sqrt{\beta(\tau)} \sum_{m\in\mathbb{M}} w_m(\boldsymbol{x})\sigma_m(\boldsymbol{x}) + \gamma(\tau). \quad (18)$$

Lipschitz continuity of the individual GP mean $\mu_m(\cdot)$ and standard deviation $\sigma_m(\cdot)$ immediately follows from (Lederer et al., 2019a, Theorem 3.1), such that bounded Lipschitz constants L_{μ_m}, L_{σ_m} exist. Moreover, the summand $\gamma(\tau)$ can be made arbitrarily small through a suitable choice of τ, such that posterior variance bounds as discussed in (Lederer et al., 2019b) guarantee arbitrarily small error bounds under weak conditions on the training data.

4. Safe Event-Triggered Learning Control

Since the structure of the uniform error bound in Theorem 3.3 is very similar to commonly used bounds in literature, the LoG-GP approach can directly be used to substitute exact GPs in many safety critical applications to enable online-learning. We demonstrate this capability by applying LoG-GPs to a learning-based control problem, where data of the system is gathered online during closed-loop control. The control task and policy are introduced in Section 4.1, while a data-efficient and safe online learning scheme using LoG-GPs is derived in Section 4.2.

4.1. Control task

Consider a nonlinear control affine dynamical system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = x_3, \qquad \dots \qquad \dot{x}_{d_x} = f(\boldsymbol{x}) + u, \quad (19)$$

with state $\boldsymbol{x} = [x_1 \ \cdots \ x_{d_x}]^T \in \mathbb{X} \subset \mathbb{R}^d$ and control input $u \in \mathbb{U} \subseteq \mathbb{R}$. We assume that the structure of the dynamics (19) is known, but the function $f(\cdot)$ itself is not. While we only consider single input systems for notational convenience, the approach can directly be extended to multiple inputs. The task is to track a desired trajectory $x_r(t)$ with the output x_1, aiming to achieve a small tracking error $\boldsymbol{e} = [e_1 \cdots e_d] = \boldsymbol{x} - \boldsymbol{x}_r$ with $\boldsymbol{x}_r = [x_r \ \dot{x}_r \ \cdots \frac{\mathrm{d}^d}{\mathrm{d}t^d} x_r]^T$.

We design a policy $\pi : \mathbb{X} \to \mathbb{U}$ which compensates the nonlinearity $f(\cdot)$ using the LoG-GP prediction $\tilde{\mu}(\cdot)$ and apply linear control principles to the approximately linearized system

$$u = \pi(\boldsymbol{x}) = -\tilde{\mu}(\boldsymbol{x}) + \nu, \quad (20)$$

with the linear control law $\nu = \frac{\mathrm{d}^d}{\mathrm{d}t^d} x_r - k_c[\boldsymbol{\lambda}^T \ 1]\boldsymbol{e}$, with gains $k_c \in \mathbb{R}_+$ and Hurwitz coefficients $\boldsymbol{\lambda} \in \mathbb{R}^{d-1}$. Using this policy, the dynamics of the tracking error are given by $\dot{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{e} + (f(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x}))[0 \ \ 0 \ \ \cdots \ \ 1]^T$, where

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\lambda_1 k_c & -\lambda_2 k_c & -\lambda_3 k_c & \cdots & -k_c \end{bmatrix}. \quad (21)$$

Since training data is often difficult to obtain safely in real-world systems, we assume that initially there is no data, but measurements can be taken as follows.

Assumption 4.1. *Noisy measurements* $y^{(i)} = f(\boldsymbol{x}^{(i)}) + \epsilon^{(i)}$ *can be taken at any instance of time.*

Thus, the controller does not only have to decide upon the control input u, but also on the time of new measurements.

4.2. Safety with Event-triggered Learning

In order to exploit this additional flexibility to safely achieve a high data-efficiency, we follow the idea in (Umlauft & Hirche, 2019) that measurements should not be taken periodically at a fixed sample time. Instead, the intuitive idea is to take a new measurement from the system whenever the model uncertainty becomes too high compared to the desired tracking performance. This is formally expressed using Lyapunov stability theory (Khalil, 2002), for which we

Algorithm 3 Safe event-triggered learning control

initialize LoG-GP T
while control task is not completed **do**
  while $2\|\boldsymbol{p}_d\|\eta_N(\boldsymbol{x},\tau) < \|\boldsymbol{e}\|$ OR $\|\boldsymbol{e}\| < 2\|\boldsymbol{p}_d\|\underline{\eta}(\tau)$ **do**
    Apply control law (20)
  end while
  while $2\|\boldsymbol{p}_d\|\eta_N(\boldsymbol{x},\tau) \geq \|\boldsymbol{e}\|$ **do**
    Take measurement $\boldsymbol{x}^{(N+1)}$, $y^{(N+1)}$
    T.UPDATE($\boldsymbol{x}^{(N+1)}$, $y^{(N+1)}$), $N \leftarrow N+1$
  end while
end while

define a quadratic Lyapunov function $V(\boldsymbol{e}) = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e}$ with a positive definite matrix $\boldsymbol{P} = [\boldsymbol{p}_1 \ \cdots \ \boldsymbol{p}_d] \in \mathbb{R}^{d\times d}$ such that $\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} = -\boldsymbol{I}_d$. The existence of $\boldsymbol{P}$ is guaranteed as $\boldsymbol{\lambda}$ is Hurwitz, and it ensures that the temporal derivative of the Lyapunov function can be bounded by

$$\dot{V}(\boldsymbol{e}) \leq -\|\boldsymbol{e}\|^2 + 2\|\boldsymbol{e}\|\|\boldsymbol{p}_d\|\eta_N(\boldsymbol{x},\tau) \tag{22}$$

using the uniform error bound in Theorem 3.3. Since a negative temporal derivative of the Lyapunov function guarantees convergence to the desired trajectory, an event for taking new measurements $\boldsymbol{x}^{(N+1)}$, $y^{(N+1)}$ should be triggered whenever $\|\boldsymbol{e}\| \leq 2\|\boldsymbol{p}_d\|\eta_N(\boldsymbol{x},\tau)$ because the additional data point reduces the posterior variance of an individual GP and thereby $\eta_{N+1}(\boldsymbol{x},\tau) < \eta_N(\boldsymbol{x},\tau)$. The computations corresponding to the update of the model must be performed online in real-time, since no decision about further data is possible until the updated uniform error bound $\eta_{N+1}(\cdot)$ has been computed. Moreover, high prediction rates, typically around $1\,\mathrm{kHz}$ for robotic applications, are necessary for continuously monitoring the triggering condition. These requirements emphasize the importance of the fast online updates and predictions provided by LoG-GPs.

Based on the triggering condition, we propose the online-learning control policy as outlined in Algorithm 3, which additionally suspends taking measurements during the satisfaction of a specified performance expressed via $\underline{\eta}(\tau) \in \mathbb{R}_+$ in order to avoid excessive triggering in close proximity to the desired trajectory. Since the posterior variance of an individual GP model is guaranteed to be smaller than $\sigma_n^2 k(0,0)/(k(0,0)+\sigma_n^2)$ at the position of a training input (Williams & Vivarelli, 2000), Algorithm 3 allows safe control without any data in advance.

Theorem 4.1. *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 $\boldsymbol{P} \in \mathbb{R}^{d\times d}$ the unique, positive definite solution to the continuous Lyapunov equation $\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} = -\boldsymbol{I}_d$ with $\boldsymbol{A}$ 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*

$$\underline{\eta}(\tau) = \sqrt{\beta(\tau)}\underline{\sigma} + \gamma(\tau) \tag{23}$$

$$\underline{\sigma}^2 > \sigma_n^2 k(0,0)/(k(0,0)+\sigma_n^2) \tag{24}$$

guarantees with probability $1-\delta$ that the tracking error $\boldsymbol{e}$ converges to

$$\mathbb{T} = \left\{\boldsymbol{x} \in \mathbb{X} \,\middle|\, \|\boldsymbol{e}\| \leq 2\underline{\eta}(\tau)\|\boldsymbol{p}_d\|\right\}. \tag{25}$$

5. Experimental Evaluation

In order to demonstrate the computational efficiency and the prediction performance of LoG-GPs[3], we compare them to several state-of-the-art GP approximations for online learning on real world regression problems in Section 5.1. Moreover, we illustrate the need for real-time regression methods with provable uniform error bounds on a control problem with event-triggered learning in Section 5.2.

5.1. Regression Performance Evaluation

We evaluate the performance of the LoG-GP approach on three real-world data sets. The SARCOS data set (Rasmussen & Williams, 2006) contains 44484 samples of the dynamics of a robotic manipulator ($d=21$), which is a widely used data set for comparison of GP approximations. Moreover, we employ the buzz in social media data set (Douzal-chouakria et al., 2013), which consists of 583250 samples with $d=77$ features, and the individual household electric power consumption data set (Dua & Graff, 2017) composed of 2048380 measurements with $d=11$. The data is preprocessed following (Wilson et al., 2016).

The LoG-GP is evaluated with $\bar{N} = 100$ and $K = 2$ using mixture of experts (MoE), generalized product of experts (gPoE) and robust Bayesian committee machine (rBCM) aggregations and conditional probabilities (12). We compare to incremental sparse spectrum Gaussian processes (ISSGP) with 200 random features (Gijsberts & Metta, 2013), local Gaussian processes with $\bar{N} = 100$ and model generation threshold 0.9 (Nguyen-Tuong et al., 2009b), streaming sparse GPs (SSGP) with 100 inducing points (Bui et al., 2017), and the robust Bayesian committee machine (Deisenroth & Ng, 2015) with $\bar{N} = 100$. All GPs use an ARD squared exponential kernel and the hyperparameters are fitted using the first 1000 training samples for all methods in order to ensure that poor hyperparameters are excluded throughout all simulations. The data used for hyperparameter optimization is added to the GP approximations, before they are evaluated in a sequential setting, in which we iterate between prediction for an input $\boldsymbol{x}^{(i)}$ and update of a model using $(\boldsymbol{x}^{(i)}, y^{(i)})$.

[3]Matlab code is online available at `https://gitlab.lrz.de/alederer/Log-GP`.

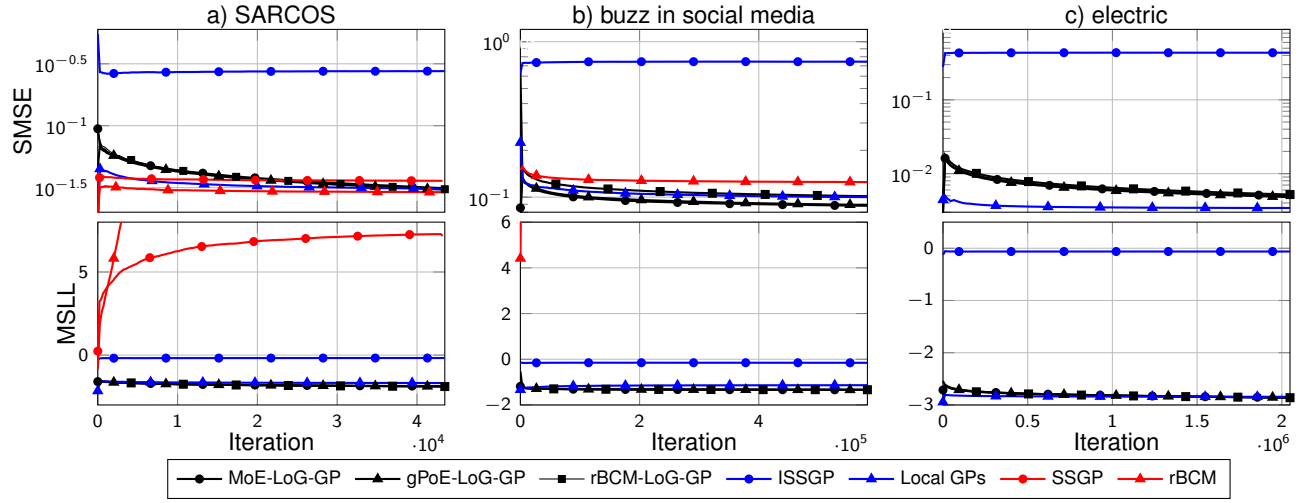

Figure 2. Plots of the SMSE (top) and MSLL (bottom) on a) SARCOS b) buzz in social media and c) electric data sets. Due the high computation times, the SSGP could only be applied to the SARCOS, while the rBCM could not be evaluated on the electric data set. LoG-GP approaches achieve at least a comparable performance to existing methods with slight advantages in high dimensional problems.

Table 1. Average update and prediction times in ms for the SARCOS, buzz in social media and electric data sets. LoG-methods outperform state-of-the art methods for streaming data regarding the computation time: updates are up to 10 times faster, and they achieve state-of-the-art prediction rates. SSGP and rBCM are greyed out as they allow only batch updates.

AVERAGE TIME (ms)	SARCOS		BUZZ		ELECTRIC	
	t_{pred}	t_{up}	t_{pred}	t_{up}	t_{pred}	t_{up}
MOE-LOG-GP	**0.12**	0.17	0.19	0.18	**0.12**	**0.15**
GPOE-LOG-GP	**0.12**	**0.16**	0.19	0.18	0.13	0.16
RBCM-LOG-GP	**0.12**	**0.16**	**0.17**	**0.17**	0.13	0.16
ISSGP	0.17	1.6	0.19	1.7	0.16	1.9
LOCAL GPS	0.94	1.1	1.5	1.9	1.1	0.91
SSGP	19	16	> 20	> 20	> 20	> 20
RBCM	2.5	4.3	17	10	> 20	> 10

Since SSGP and rBCM do not allow sequential updates, they are updated in batches[4]. As performance metric we use the average prediction and update times, as well as the standardized mean squared error (SMSE) and the mean standardized log loss (MSLL) (Rasmussen & Williams, 2006) in a sequential interpretation, e.g.,

$$\text{SMSE}_k = \frac{\sum_{i=1}^{k}(\tilde{\mu}_{k-1}(\boldsymbol{x}^{(k)}) - y^{(k)})^2}{k s_y^2}, \qquad (26)$$

where s_y^2 denotes the empirical variance of the targets $y^{(i)}$ and $\tilde{\mu}_{k-1}(\boldsymbol{x}^{(k)})$ the prediction after observing $k-1$ training samples.

The resulting computation times averaged over 20 runs are depicted in Table 1. It can be clearly seen that LoG-GP approaches achieve the lowest average computation times in these simulations, with significant advantages over existing methods regarding the model updates. While state-of-the-art real-time learning methods such as ISSGPs can yield a similar prediction rate, batch methods such as SSGPs or the rBCM exhibit a quickly growing complexity. This prevents their application in online learning, even though a small prediction error could be obtained, as illustrated in Fig. 2. While local GPs provide a poor accuracy for large data sets, LoG-GP methods and ISSGPs achieve a similar performance with minor advantages for ISSGPs on the low-dimensional electric data set and slightly better performance of LoG-GP methods on the high-dimensional buzz in social media set. Additionally, LoG-GP approaches result in the best MSLL values, merely marginally outperformed by ISSGPs for the first half of the electric data set. This emphasizes the high reliability of the epistemic uncertainty estimate provided by LoG-approaches, which is crucial together with the strong theoretical foundation and the low computation times for enabling real-time learning in safety critical applications.

5.2. Application to Event-Triggered Learning Control

For the numerical illustration of the event-triggered learning control, we consider a modification of a pendulum system with

$$f(\boldsymbol{x}) = 1 - \sin(x_1) + \frac{0.5}{1 + \exp(-x_2/10)}. \qquad (27)$$

As reference trajectory, we set an outwards spiral

$$x_r(t) = \left(1 + \frac{9}{1+\exp(-0.1(t-100))}\right)\sin(0.5t). \qquad (28)$$

For real-time learning, we employ a MoE-LoG-GP with $\bar{N} = 100$ and an ARD squared exponential kernel and com-

[4] More details on the simulation setup and additional results can be found in the appendix.

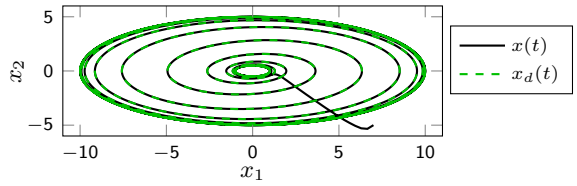

Figure 3. The system (red) properly tracks the desired outwards spiral trajectory (green) after an initial transient phase.

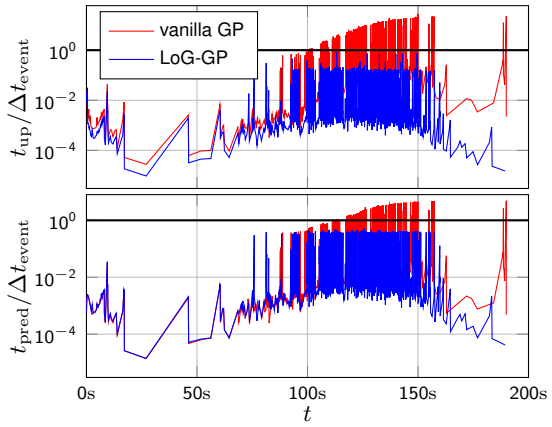

Figure 4. The ratio between training/prediction and inter-event time keeps growing for the vanilla GP (red) and eventually exceeds 1, while it stagnates after some time for the LoG-GP (blue) despite a slight growth in the added training samples (1310 vs. 1741).

pare it to a vanilla GP as used in (Umlauft & Hirche, 2019).

The resulting trajectory for $t \in [0\text{s}, 200\text{s}]$ is illustrated in Fig. 3, and it can be seen that it closely follows the reference trajectory. While the vanilla GP only requires 1310 events, the event for learning is triggered 1741 times for the MoE-LoG-GP. However, this slight reduction in data efficiency is necessary in order to meet the real-time constraints as depicted in Fig. 4. In contrast to the vanilla GP, where the prediction and update times t_{pred} and t_{up} start to exceed the inter-event time Δt_{event} after $\approx t = 120\text{s}$, LoG-GPs remain fast enough to satisfy this condition. Therefore, LoG-GPs can enable the application of Gaussian processes in safety critical learning problems with real-time constraints.

6. Conclusion

This paper presents a novel method for real-time learning based on Gaussian process regression. By iteratively dividing individual models according to a localizing random distribution, the computation graph corresponds to a random tree providing logarithmic complexity guarantees for predictions and model updates. In order to allow the usage in safety-critical applications, uniform error bounds from exact Gaussian process regression are extended, which is exploited in the design of a safe, online-learning control policy.

Acknowledgements

This work was supported by the European Research Council (ERC) Consolidator Grant "Safe data-driven control for human-centric systems (CO-MAN)" under grant agreement number 864686. Armin Lederer gratefully acknowledges financial support from the German Academic Scholarship Foundation.

References

Andersson, O., Wzorek, M., and Doherty, P. Deep Learning Quadcopter Control via Risk-Aware Active Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 3812–3818, 2017.

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.

Douzal-chouakria, A., Gaussier, E., Dimert, E., Douzal-chouakria, A., Gaussier, E., and Pr, E. D. Prédictions d'activité dans les réseaux sociaux en ligne. In *4ième conférence sur les modèles et l'analyse des réseaux : Approches mathématiques et informatiques*, pp. 16, 2013.

Dua, D. and Graff, C. UCI Machine Learning Repository, 2017. URL http://archive.ics.uci.edu/ml.

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.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J. M., Lam, V. D., Bewley, A., and Shah, A. Learning to Drive in a Day. In *Proceedings of the International Conference on Robotics and Automation*, pp. 8248–8254, 2019.

Khalil, H. K. *Nonlinear Systems*. Prentice-Hall, Upper Saddle River, NJ, third edition, 2002.

Kong, J., Pfeiffer, M., Schildbach, G., and Borrelli, F. Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 1094–1099, 2015.

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., Quiñonero-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., Umlauft, 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., Umlauft, J., and Hirche, S. Posterior Variance Analysis of Gaussian Processes with Application to Average Learning Curves. 2019b. URL http://arxiv.org/abs/1906.01404.

Lederer, A., Umlauft, J., and Hirche, S. Uniform Error and Posterior Variance Bounds for Gaussian Process Regression with Application to Safe Control. 2021. URL http://arxiv.org/abs/2101.05328.

Lee, S., Choi, H., and Min, K. Reduction of Engine Emissions via a Real-Time Engine Combustion Control with an EGR Rate Estimation Model. *International Journal of Automotive Technology*, 18(4):571–578, 2017.

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 Online Interactive Learning with Scalability. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1910–1920, 2020.

Maddalena, E. T., Scharnhorst, P., and Jones, C. N. Deterministic Error Bounds for Kernel-Based Learning Techniques under Bounded Noise. pp. 1–9, 2020. URL https://arxiv.org/pdf/2008.04005.pdf.

Masoudnia, S. and Ebrahimpour, R. Mixture of Experts: A Literature Survey. *Artificial Intelligence Review*, 42(2): 275–293, 2014.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, 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.

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

Rullière, 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.

Schölkopf, 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.

Terry, N. and Choe, Y. Splitting Gaussian Process Regression for Streaming Data. 2020. URL http://arxiv.org/abs/2010.02424.

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.

Umlauft, J. and Hirche, S. Feedback Linearization based on Gaussian Processes with event-triggered Online Learning. *IEEE Transactions on Automatic Control*, 65(10):4154–4169, 2019.

Umlauft, 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 Vaart, A. and van Zanten, H. Information Rates of Nonparametric Gaussian Process Methods. *Journal of Machine Learning Research*, 12:2095–2119, 2011.

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.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep Kernel Learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, volume 51, pp. 370–378, 2016.

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 & Opper, 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 (Hu-

ber, 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 online 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 (Mutný & Krause, 2018; Angelis et al., 2020). However, these bounds often require a practically intractable number of features, 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 (Umlauft 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, Rullière 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 proofing 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, $\boldsymbol{p}$ 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 $\boldsymbol{p}$.

The distribution of balls is done iteratively. Starting at the tree, a ball is assigned to a child by drawing from the random distribution $\boldsymbol{p}$ 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 $\boldsymbol{p}$.

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 $\boldsymbol{p}^n(\boldsymbol{x})$. 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 $\boldsymbol{p}^n(\cdot)$ satisfying Assumption 3.1 requires $\mathcal{O}_p(\log(N))$ computations*[5].

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. □

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 $\boldsymbol{p}^n(\cdot)$ satisfying Assumptions 3.1 and 3.2require $\mathcal{O}_p(\log^2(N))$ computations.*

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. □

[5] Due 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., $h^m \in \mathcal{O}_p(\log(N)) \Leftrightarrow \lim_{N\to\infty} P(h^m > c\log(N)) = 0$ for some finite $c \in \mathbb{R}$.

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 $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ with Lipschitz constant L_k on the compact set $\mathbb{X} \subset \mathbb{R}^d$. Furthermore, consider a continuous unknown function $f : \mathbb{X} \to \mathbb{R}$ 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*

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

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

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

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

where

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

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

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

where

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

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

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

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

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

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. □

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

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

for all $\boldsymbol{x}, \boldsymbol{x}', \tilde{\boldsymbol{x}}$ 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). □

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 $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}')$ admits a Lipschitz constant*

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

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

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

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

where the kernel pseudo-metric is defined as

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

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

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

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

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

□

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

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

we have

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

Therefore, we obtain

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

It can be shown that this expression reaches its maximum for $\boldsymbol{x} \to \boldsymbol{x}'$, 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 $f(\boldsymbol{x}) = \langle f, k(\boldsymbol{x}, \cdot) \rangle_k$ 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 $\{\boldsymbol{x}^{(t)}, \epsilon_t\}_{t=1}^{n-1}$ and $\boldsymbol{x}^{(n)}$.

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 $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$. Furthermore, consider an unknown function $f : \mathbb{X} \to \mathbb{R}$ 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*

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

Then, it holds that

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

where

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

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

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

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

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

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

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

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. □

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 $\boldsymbol{z} : \mathbb{R}^d \to \mathbb{R}^{2D}$, $D \in \mathbb{R}_+$, the covariance function can be approximated by $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{z}^T(\boldsymbol{x})\boldsymbol{z}(\boldsymbol{x}')$, 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

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

for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$ 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., $|k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{z}^T(\boldsymbol{x})\boldsymbol{z}(\boldsymbol{x}')| \leq \rho$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$. 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), which depends on the number of features D through

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

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

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

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

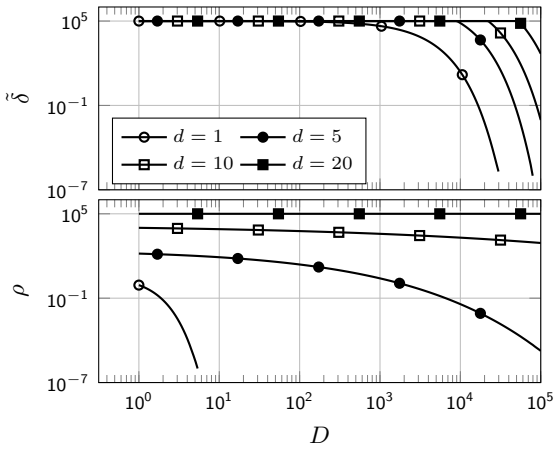

Figure 5. A high number of random Fourier features is generally required to guarantee at least a decent uniform approximation bound for the squared exponential kernel. While fewer quadrature Fourier features are sufficient for very low dimensional input spaces, medium values of $d \geq 5$ require an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

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

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

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

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 $\boldsymbol{p}^n(\cdot)$. This can be straightforwardly achieved by defining them based on (12), but employing

embodying symmetry in the saturating linear functions, e.g.,

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

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

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

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

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

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

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

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 $\boldsymbol{p}^n(\cdot)$, 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 $\boldsymbol{P} \in \mathbb{R}^{d \times d}$ the unique, positive definite solution to the algebraic Riccati equation $\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} = -\boldsymbol{I}_d$ with $\boldsymbol{A}$ 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 $\boldsymbol{e}$ converges to $\mathbb{T} = \left\{\boldsymbol{x} \in \mathbb{X} \middle| \|\boldsymbol{e}\| \leq 2\underline{\eta}(\tau)\|\boldsymbol{p}_d\|\right\}$.*

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

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

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

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

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 $|f(\boldsymbol{x}) - \tilde{\mu}_k(\boldsymbol{x})|$ does not exceed $\|\boldsymbol{e}\|/(2\|\boldsymbol{p}_d\|)$ for all k. With the model error bound in Theorem 3.3, one can conclude

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

where the latter condition is ensured by the triggering mechanism (first part line 3, Algorithm 3) for all $\boldsymbol{x} \in \mathbb{X} \setminus \mathbb{T}$ (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 $\boldsymbol{x}^{(1)}$. It directly follows from (Williams & Vivarelli, 2000; Lederer et al., 2019b) that the variance satisfies

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

Hence, regardless of the error $\boldsymbol{e}$, 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 $\boldsymbol{x}^{(k)}$. 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 $\boldsymbol{x}^{(k)}$ which is added to the training data, i.e.,

$$\sigma^2_{i,k}(\boldsymbol{x}^{(k)}) < \sigma^2_{i,k-1}(\boldsymbol{x}^{(k)}). \tag{67}$$

Hence, (65) is satisfied for all $\boldsymbol{x} \in \mathbb{T} \setminus \mathbb{B}$, 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 $\boldsymbol{x}^{(k)}$, at which the measurement $y^{(k)}$ is recorded, several individual models have a positive probability $\omega_i(\boldsymbol{x})$ 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 $\boldsymbol{x} \in \mathbb{X} \setminus \mathbb{T}$.

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

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 set[6] 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 sets[7] 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 $\boldsymbol{p}^n(\cdot)$ are defined through (11), (12) , where j^n_k is the dimension of the maximum spread of the local data set, s^n_k is the mean of the data in this dimension and

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

  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

  $$\omega_m = \frac{\log(k(\boldsymbol{x},\boldsymbol{x})) - \log(\sigma^2_m(\boldsymbol{x}))}{2} \prod_{i=1}^{h^m} p^{b^m_i}_{s^m_i}(\boldsymbol{x}). \tag{69}$$

  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.

- local GPs[8]: 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.

- SSGP[9]: the method is used with 100 inducing points, but no online optimization of hyperparameters and

[6]The data set is available at http://www.gaussianprocess.org/gpml/data/.

[7]The data and pre-processing are available at https://drive.google.com/file/d/0BxWe_IuTnMFcYXhxdUNwRHBKTlU/view.

[8]The code is available at https://www.ias.informatik.tu-darmstadt.de/Miscellaneous/Miscellaneous.

[9]The code is available at https://github.com/thangbui/streaming_sparse_gp.

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)	—

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.

- rBCM[10]: 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 toolbox[11] (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 dimensional 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.

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)	—

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.

In addition to the slight advantages in the regression performance, LoG-GP approaches exhibit significantly lower

[10]The code is available at https://github.com/LiuHaiTao01/GRBCM.

[11]The code is available at https://github.com/GPflow/GPflow.

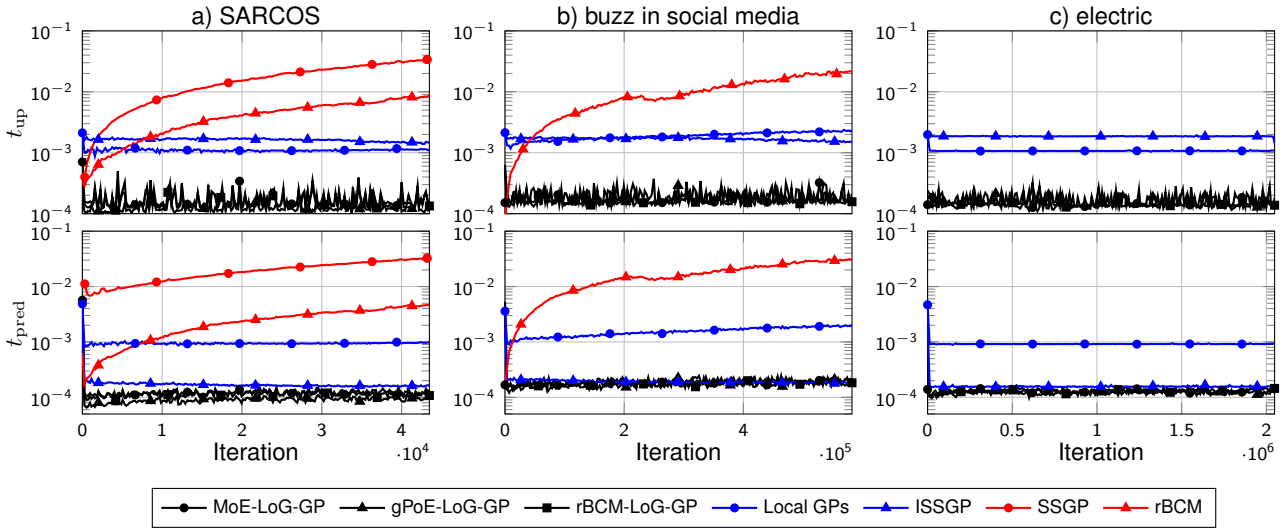

Figure 6. Plots of average update time t_{up} (top) and the average prediction time t_{pred} (bottom) on a) SARCOS b) buzz in social media and c) electric data sets. Due the high computation times, the SSGP could only be applied to the SARCOS, while the rBCM could not be evaluated on the electric data 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.

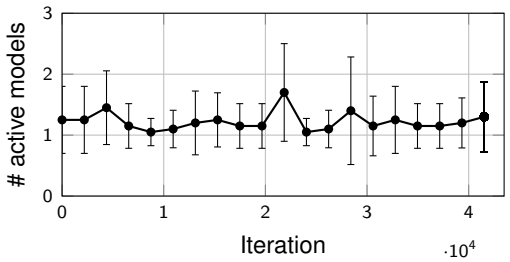

Figure 7. The small computation times for predictions are achieved by a low number of active models, which is on average less than 2 for the MoE-LoG-GP on the SARCOS data set. The comparatively high standard deviation of the number of active models illustrated by the error bars contributes to noisy prediction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

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 o_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 complexity, 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

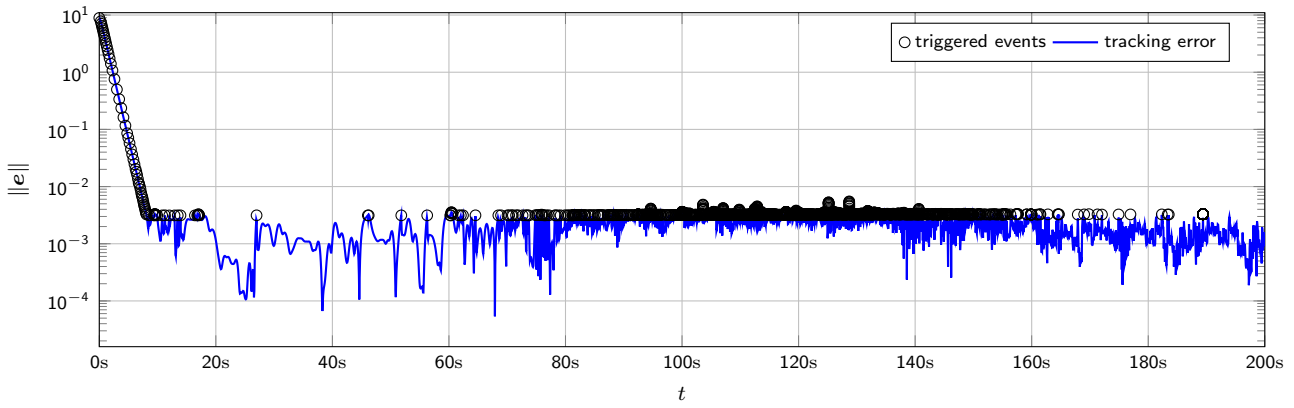

Figure 8. An event for learning can be triggered when the error exceeds the threshold prescribed by $\underline{\eta}(\tau)$, which happens frequently in the transition period at the beginning and when changing the radius of the reference trajectory. Once a stationary behavior has been reached, very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

Table 5. Parameters for the learning control illustration

k_c	λ	$\boldsymbol{x}_0$	σ_n^2	σ_f	$\boldsymbol{l}$	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

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 $\tilde{\underline{\eta}}(\tau) = 2\sqrt{\beta(\tau)}\sigma_n + \gamma(\tau)$, 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 $\approx$ 20s, 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 $\approx$ 60s. 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 $\tilde{\underline{\eta}}(\tau)$ 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.

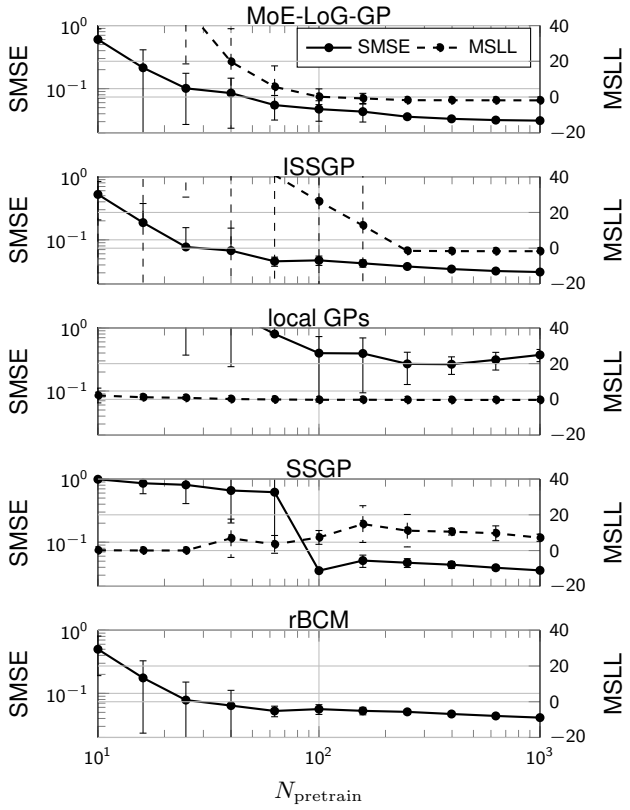

Figure 9. Comparison of the methods regarding their dependency on the quality of hyperparameters resulting from hyperarameter optimization with N_{pretrain} samples. The standard deviations are depicted via the error bars. The SMSE of all methods suffers from poor hyperparameters similarly. The MSLL exhibits strongly different behavior for the methods: local GPs show almost constant MSLL 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}.

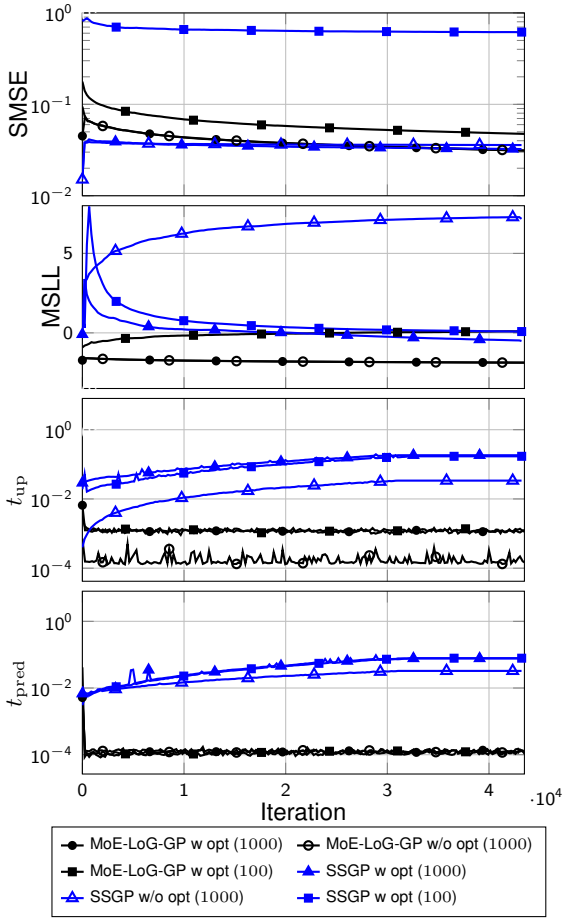

Figure 10. Online hyperparameter optimization has a weak impact on the SMSE. The MSLL of the SSGP improves through the hyperparameter optimization, but it remains significantly worse than the MSLL of LoG-GP approaches. When the number of pretraining samples is reduced to 100, both approaches suffer from 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.

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 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 be 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 ($103.7\,\text{ms}$ for SSGP, $1.2\,\text{ms}$ 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.

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*

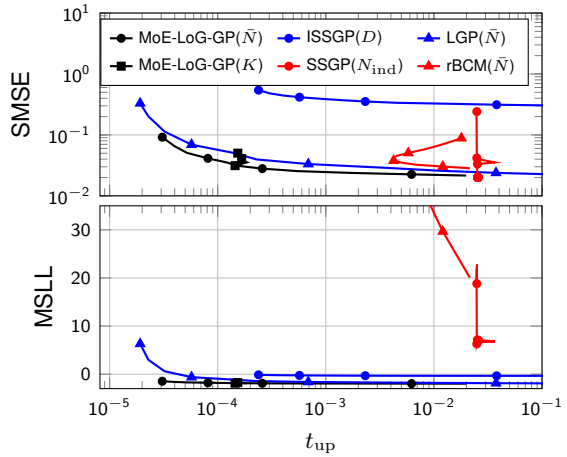

Figure 11. Given a desired update time, LoG-GPs exhibit the lowest SMSE and MSLL values on the SARCOS data set, indicating a beneficial performance-computation time trade-off. The performance and update time dependency of LoG-GPs on the number of children per node K is rather small.

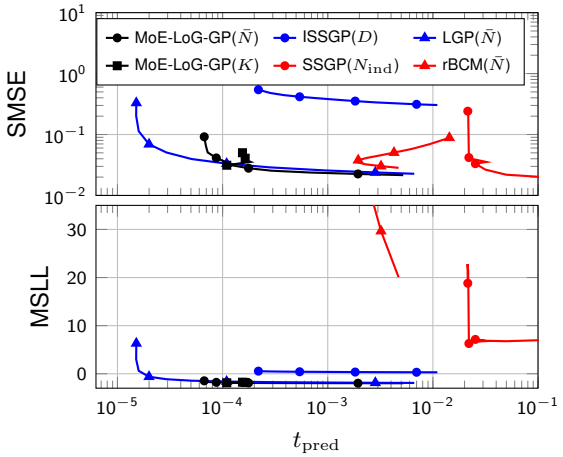

Figure 12. Given a desired prediction time, the LoG-GPs yield better MSLL values on the SARCOS data set than the other methods. This also holds for the SMSE for prediction times larger than 10^{-4}s. The prediction time dependency of LoG-GPs on K is small.

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 $\bar{N} \in [10, 1000]$, ISSGPs with $D \in [10, 1000]$, SSGPs[12] with $N_{\text{ind}} \in [10, 1000]$. Additionally, we investigate the impact of the number of child nodes K on gPoE-LoG-GPs. Since $\bar{N}$, 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-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 er-

[12]While 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.

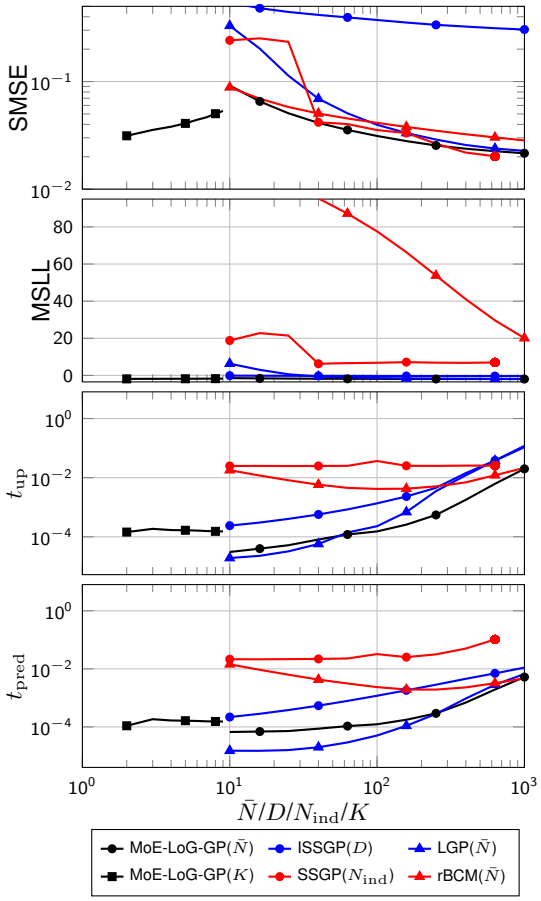

Figure 13. Comparison of the parameter dependency of the different learning methods. The SMSE values of LoG-GPs improve when more data points $\bar{N}$ are allowed in each model, but the computation times grow. Other methods exhibit a similar behavior.

rors 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.