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).

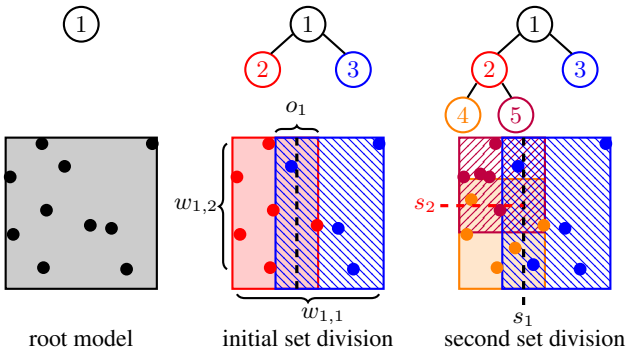

Figure 1. Iterative model tree construction and corresponding layout of the input space for $K = 2$: active regions and training samples belonging to the same node are depicted in the same color.

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.

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).

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

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

Algorithm 3 Safe event-triggered learning control

```
initialize LoG-GP T
while control task is not completed do
    while 2‖p_d‖η_N(x,τ) < ‖e‖ OR ‖e‖ < 2‖p_d‖η̲(τ) do
        Apply control law (20)
    end while
    while 2‖p_d‖η_N(x,τ) ≥ ‖e‖ do
        Take measurement x^(N+1), y^(N+1)
        T.UPDATE(x^(N+1), y^(N+1)), N ← N + 1
    end while
end while
```

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 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

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

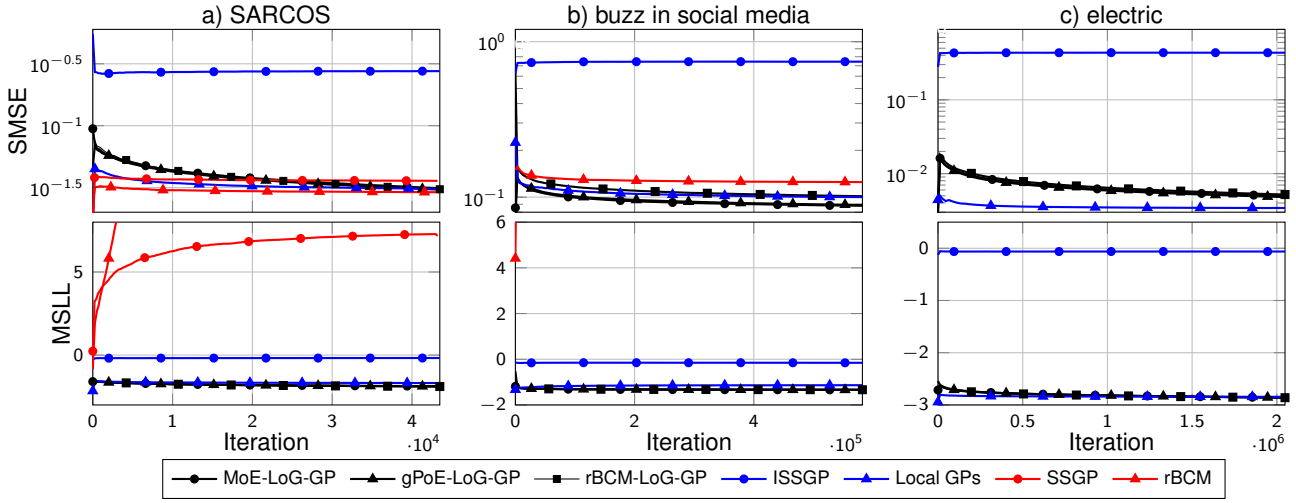

Figure 2. Plots of the SMSE (top) and MSLL (bottom) on a) SARCOS b) buzz in social media and c) electric data sets. Due the high computation times, the SSGP could only be applied to the SARCOS, while the rBCM could not be evaluated on the electric data set. LoG-GP approaches achieve at least a comparable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

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 $(\boldsymbol{x}^{(i)}, y^{(i)})$. 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

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

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

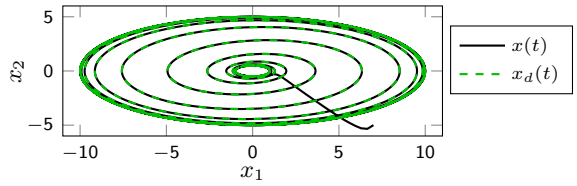

Figure 3. The system (red) properly tracks the desired outwards spiral trajectory (green) after an initial transient phase.

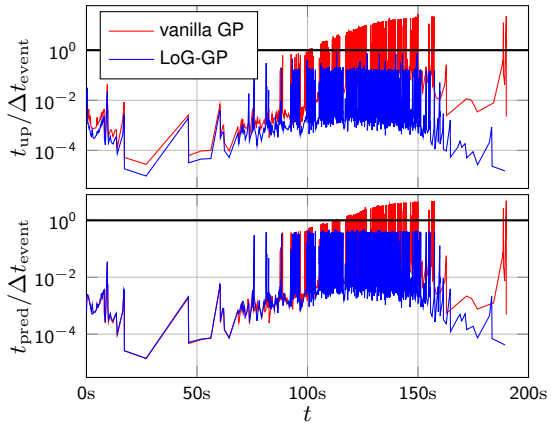

Figure 4. The ratio between training/prediction and inter-event time keeps growing for the vanilla GP (red) and eventually exceeds 1, while it stagnates after some time for the LoG-GP (blue) despite a slight growth in the added training samples (1310 vs. 1741).

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

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.

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.

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.

Chowdhury, S. R. and Gopalan, A. On Kernelized Multi-armed Bandits. In *Proceedings of the International Conference on Machine Learning*, pp. 844–853, 2017.

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.

Gijsberts, A. and Metta, G. Real-Time Model Learning using Incremental Sparse Spectrum Gaussian Process Regression. *Neural Networks*, 41:59–69, 2013.

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.

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.

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.

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.

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.

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.

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

Schreiter, J., Nguyen-Tuong, D., and Toussaint, M. Efficient Sparsification for Gaussian Process Regression. *Neurocomputing*, 192:29–37, 2016.

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.

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.

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

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., 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.