Hierarchical-Hyperplane Kernels for Actively Learning Gaussian Process Models of Nonstationary Systems

Matthias Bitzer
Bosch Center for Artificial Intelligence, Renningen, Germany

Mona Meister
Bosch Center for Artificial Intelligence, Renningen, Germany

Christoph Zimmer
Bosch Center for Artificial Intelligence, Renningen, Germany

Abstract

Learning precise surrogate models of complex computer simulations and physical machines often require long-lasting or expensive experiments. Furthermore, the modeled physical dependencies exhibit nonlinear and nonstationary behavior. Machine learning methods that are used to produce the surrogate model should therefore address these problems by providing a scheme to keep the number of queries small, e.g. by using active learning and be able to capture the nonlinear and nonstationary properties of the system. One way of modeling the nonstationarity is to induce input-partitioning, a principle that has proven to be advantageous in active learning for Gaussian processes. However, these methods either assume a known partitioning, need to introduce complex sampling schemes or rely on very simple geometries. In this work, we present a simple, yet powerful kernel family that incorporates a partitioning that: i) is learnable via gradient-based methods, ii) uses a geometry that is more flexible than previous ones, while still being applicable in the low data regime. Thus, it provides a good prior for active learning procedures. We empirically demonstrate excellent performance on various active learning tasks.

1 INTRODUCTION

Active learning is a principled way to learn a model in a sequential data-efficient manner. It is especially useful when the collection of data is expensive. For classification, this is the case for the manual labeling procedure (Settles, 2009). Regression tasks in which active learning is used are, e.g. the learning of a surrogate model of complex physical processes like complex machines (Zimmer et al., 2018) or surrogate modeling of long-running computer simulations (Gramacy, 2020). Thereby, queries to the oracle are either very expensive/energy intensive or take a very long time, which makes it necessary to minimize the number of queries. Recent studies on active learning for regression tasks utilized Gaussian processes (Garnett et al., 2014; Schreiter et al., 2015; Marmin et al., 2018; Yue et al., 2021; Li et al., 2022), which have the great advantage of providing a principled notion of uncertainty, making them ideal candidates for active learning algorithms.

In practice, Gaussian process regression is often used with stationary kernels, such as the Squared Exponential Kernel. However, the stationarity of the kernel implicitly assumes that the correlation of the function values of the learned function is translation invariant. This assumption is often not met in practice and major performance gains were shown for passive learning using nonstationary Gaussian processes (Remes et al., 2017; Gramacy and Lee, 2007; Wilson et al., 2016). Furthermore, when active learning is employed, the stationarity assumption implies an almost uniform input design [see Marmin et al. (2018)]. Nonstationary kernels on the other hand induce a non-uniform sampling in the input space, which has been shown to be beneficial when the data actually exhibits nonstationarities (Gramacy et al., 2004; Marmin et al., 2018).

In particular, input-partitioning showed promising results when combined with active learning (Lee et al., 2021; Gramacy et al., 2004; Krause and Guestrin, 2007). However, existing methods either rely on a fixed, known partition (Lee et al., 2021; Krause and Guestrin, 2007) or use restricted geometries/ priors in function space combined with complex sampling schemes (Gramacy et al., 2004). Our goal is to provide a simple, yet powerful partitioning kernel that can be used as plug-and-play in most Gaussian process frameworks with the same (or even better) sampling behavior in active learning procedures. Concretely, our partitioning is constructed via a hierarchy of hyperplanes, build with sigmoidal gates to introduce differentiability of the kernel and smoothness of the function prior. Furthermore, the ge-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

ometry of the partitioning is more flexible compared to the input-aligned partitions of Gramacy et al. (2004), but still simple enough for the usage in the low data regime. In the next section, we give an overview over related work. In Section 3 we introduce our method/kernel and give some motivation for its usage in active learning settings. Finally, in the experimental section, we compare against different nonstationary Gaussian process based models on several real-world active learning tasks and show excellent performance.

2 BACKGROUND AND RELATED WORK

Gaussian Processes. Gaussian processes provide expressive priors over functions that can be used for surrogate modeling and regression tasks. Formally, for $\mathcal{X} \subset \mathbb{R}^d$, a Gaussian process (GP) is a probability distribution over functions $f : \mathcal{X} \to \mathbb{R}$ for which each finite selection of function values $[f(x_t)]_{t=1}^T$ at input points $x_1, \dots, x_T$ has a multivariate normal distribution. The GP is fully characterized by its mean function $\mu : \mathcal{X} \to \mathbb{R}$ and its covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ also called the kernel. The kernel incorporates the major properties of the resulting sample functions and is often parameterized with some parameters Θ. For regression, a dataset $\mathcal{D} := \{\mathbf{x}_T, \mathbf{y}_T\}$ with inputs $\mathbf{x}_T = [x_1, \dots, x_T]$ and ouputs $\mathbf{y}_T = [y_1, \dots, y_T]$ is considered, where the observations are perturbed with Gaussian noise $y_t = f(x_t) + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$. A major advantage of the Gaussian process is that the posterior $f|\mathcal{D}$ is again a Gaussian process with closed-form expressions for the mean and kernel function

$$\mu_T(x) = \mu(x) + \mathbf{k}_T(x)^\intercal (K_T + \sigma^2 I)^{-1}(\mathbf{y}_T - \mu(\mathbf{x}_T)),$$
$$k_T(x, y) = k(x, y) - \mathbf{k}_T(x)^\intercal (K_T + \sigma^2 I)^{-1}\mathbf{k}_T(y),$$

where $\mathbf{k}_T(x) = [k(x, x_1), \dots, k(x, x_T)]^\intercal$ and $K_T = [k(x_t, x_l)]_{t,l=1}^T$. Thus, the predictive distribution for a new point $x^* \in \mathcal{X}$ can also be written in closed form by

$$p(f^*|x^*, \mathcal{D}, \Theta) = \mathcal{N}(\mu_T(x^*), \sigma_T^2(x^*)),$$

where $f^* := f(x^*)$ and $\sigma_T^2(x^*) := k_T(x^*, x^*)$. We denote the dependence on Θ if necessary. For more details on GP regression, we refer the interested reader to Rasmussen and Williams (2006).

Nonstationary GP's and Input-Partitioning. In GP regression, the learned function f is assumed to be a sample from a GP with kernel k_Θ. Herein, the kernel provides the main a priori assumption on the learned function f. The most popular kernels such as the RBF and the Matérn kernel are *stationary*, that means $k_\Theta(x, y) = k_\Theta(x - y)$ for all $x, y \in \mathcal{X}$. The correlation between function values therefore is translation invariant, and the modeled function is assumed to behave similarly over the complete input region. Different kinds of nonstationary kernels have been proposed so far. For time-warped kernels (Marmin et al., 2018; Wilson et al., 2016; Snoek et al., 2014) the input is transformed with a nonlinear mapping and chained with a stationary kernel. Further methods render parameters of stationary kernels input-dependent like input-dependent kernel variance and input-dependent lengthscale (Heinonen et al., 2016; Remes et al., 2017; Herlands et al., 2016). For many proposed kernels [as in Heinonen et al. (2016) or Herlands et al. (2016)] the goal is to provide flexible priors and scalable inference, with priors not explicitly dedicated to the low-data regime. A further principle to induce nonstationarity is the partitioning of the input space, as for example done in Gramacy and Lee (2007), who provide a model named TreedGP or in Krause and Guestrin (2007) and Lloyd et al. (2014) who provide input-partitioning on one-dimensional datasets via change-points. We present a model that can be viewed as a multi-dimensional generalization of change-points, where we replace change-points with change-hyperplanes. The resulting inductive bias in d dimensions of our model is most similar to the TreedGP model. Technically, TreedGP is a Bayesian CART (Classification and Regression Tree) model with independent GP's in its leaves. However, rather than using a CART model, we employ input-dependent weighting (Herlands et al., 2016; Krause and Guestrin, 2007) to induce a non-axis aligned partitioning which at the same time has the advantage that the kernel is differentiable with respect to all its parameters and that the associated GP induces continuous sample paths.

Active Learning with Gaussian Processes. In active learning, data is selected sequentially, often guided by the current model state, which can drastically reduce the number of evaluations. We focus on active learning for regression tasks which has applications in industry, e.g. surrogate modeling of combustion engines (Zimmer et al., 2018), shape control (Yue et al., 2021) and in the approximation of long-running computer simulations (Gramacy, 2020). Gaussian processes are a natural choice to pursue active learning for regression due to their principled uncertainty quantification [see Zimmer et al. (2018); Gramacy (2020); Yue et al. (2021); Krause et al. (2008); Marmin et al. (2018)]. In case the underlying ground-truth system can be better described via a nonstationary Gaussian process, using a nonstationary kernel also has profound impact on the sample selection as the samples are not selected homogenous over the input space [see Krause and Guestrin (2007); Marmin et al. (2018); Gramacy et al. (2004)]. In our experimental section, we will compare against other nonstationary GP priors that were proposed to be used in active-learning settings, such as the TreedGP model (Gramacy et al., 2004) or an input-warped GP (Marmin et al., 2018). Furthermore, Sauer et al. (2020) investigated the ap-

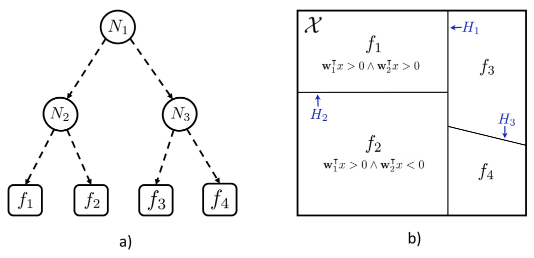
N_1
N_2
N_3
f_1
f_2
f_3
f_4
a)
$\mathcal{X}$
$\leftarrow H_1$
f_1
$\mathbf{w}_1^\intercal x > 0 \wedge \mathbf{w}_2^\intercal x > 0$
f_3
H_2
H_3
f_2
$\mathbf{w}_1^\intercal x > 0 \wedge \mathbf{w}_2^\intercal x < 0$
f_4
b)

Figure 1: Partitioning: a) Binary (symmetric) tree for partitioning with four leaves. b) Possible partition of the input space induced by that tree.

plication of DeepGPs in active-learning settings. We will stick to the natural way of doing active-learning via querying the point with the highest information gain between the observation and all uncertain variables in the model [see MacKay (1992); Houlsby et al. (2011)] to investigate which impact our prior has on the active-learning performance. In Section 4, we show superior performance in terms of RMSE curves compared to the main competitors on various tasks.

3 METHOD

We induce input-partitioning by defining the final GP f on $\mathcal{X} \subset \mathbb{R}^d$ as an input-dependent sum of J independent latent GPs $f_1, \dots, f_J$, i.e.

$$f(x) = \sum_{j=1}^{J} \lambda_j(x) f_j(x),$$

where $\lambda_j \colon \mathcal{X} \to [0,1]$ are weighting functions with $\sum_{j=1}^{J} \lambda_j(x) = 1$ for all $x \in \mathcal{X}$. The latent GPs are equipped with stationary kernels $k_j(x,y)$. The function f is therefore a GP itself with kernel

$$k(x,y) = \sum_{j=1}^{J} \lambda_j(x) \lambda_j(y) k_j(x,y).$$

We present a new kernel of this form that partitions the input hierarchically through λ_j. The weighting functions $\lambda_j, j = 1, \dots, J$, specify the regions in which the modeled function f is described by the corresponding latent functions f_j. If, for example, the weighting function $\lambda_j(x)$ is close to one for all x in some region $A \subset \mathcal{X}$, then the GP f is described in that region by the kernel $k_j(x,y)$. Therefore, the GP behaves like a stationary GP in region A. The proposed geometry of the partition via a hierarchy of hyperplanes is inspired by a mixture of linear experts model (Bishop and Svenskn, 2002). In this work, we use the partitioning logic to define a nonstationary GP.

3.1 Input-Partitioning

The partitioning is done along a binary tree $\mathcal{T}$ with $M := J-1$ nodes and J leaves. Each node $N_i, i = 1, \dots, M$, is associated with a vector $\mathbf{w}_i \in \mathbb{R}^{d+1}$ and an induced hyperplane $H_i = \{x \in \mathbb{R}^d | \mathbf{w}_i^\intercal \tilde{x} = 0\}$ where $\tilde{x} = (1, x)^\intercal \in \mathbb{R}^{d+1}$. Each leaf represents one latent GP f_j [see Figure 1a) for an example]. Every node N_i splits the input region with its hyperplane by placing an input-dependent weight $\sigma(\mathbf{w}_i^\intercal \tilde{x})$ to its left subtree and all associated GPs and $1 - \sigma(\mathbf{w}_i^\intercal \tilde{x})$ to its right subtree, where $\sigma(\cdot)$ is the standard logistic sigmoid function. The final weights $\lambda_j(x), j = 1, \dots, J$, are given by multiplying all weights along their respective path in the tree:

$$\lambda_j(x) = \prod_{i=1}^{M} \sigma(\mathbf{w}_i^\intercal \tilde{x})^{\xi_L(j,i)} (1 - \sigma(\mathbf{w}_i^\intercal \tilde{x}))^{\xi_R(j,i)}, \quad (1)$$

where $\xi_L, \xi_R \colon \{1, \dots, J\} \times \{1, \dots, M\} \to \{0,1\}$ encode the tree structure with

$$\xi_L(j,i) = \begin{cases} 1 & \text{if } f_j \text{ is in the left subtree of } N_i, \\ 0 & \text{else,} \end{cases}$$

and

$$\xi_R(j,i) = \begin{cases} 1 & \text{if } f_j \text{ is in the right subtree of } N_i, \\ 0 & \text{else.} \end{cases}$$

This weighting function leads to a hierarchical partitioning of the input space. This can be understood by observing that the multiplication of the sigmoid functions acts as a soft version of logicals AND's, where the weight $\lambda_j(x)$ is large whenever the input $x \in \mathcal{X}$ lies at the correct side of each hyperplane when traversing down the tree to f_j. Here, correct means either on one or the other side of the hyperplane, depending if the path in the tree progresses on the right or the left subtree. We call the resulting kernel Hierarchical-Hyperplane Kernel (HHK).

An illustrative example of the partitioning is given in Figure 1 a) and b). The weighting functions recursively divide the input space $\mathcal{X} =: A_1$ beginning with the hyperplane H_1 at node N_1 into sets $A_2 := \{x \in A_1 | \mathbf{w}_1^\intercal \tilde{x} > 0\}$ and $A_3 := \{x \in A_1 | \mathbf{w}_1^\intercal \tilde{x} < 0\}$. In the next layer, at node N_i, the associated set A_i is again divided into the sets $\{x \in A_i | \mathbf{w}_i^\intercal \tilde{x} > 0\}$ and $\{x \in A_i | \mathbf{w}_i^\intercal \tilde{x} < 0\}$. The actual weighting of the stationary kernels is soft due to the application of the sigmoid functions. The partitioning can be rendered finer by increasing the tree size.

3.2 Kernel Parameters and Inference

We provide a fully-Bayesian GP model, marginalized over its hyperparameters. This has the advantage of being more robust against model misspecification in the low-data regime [see. Snoek et al. (2012); Riis et al. (2022)].

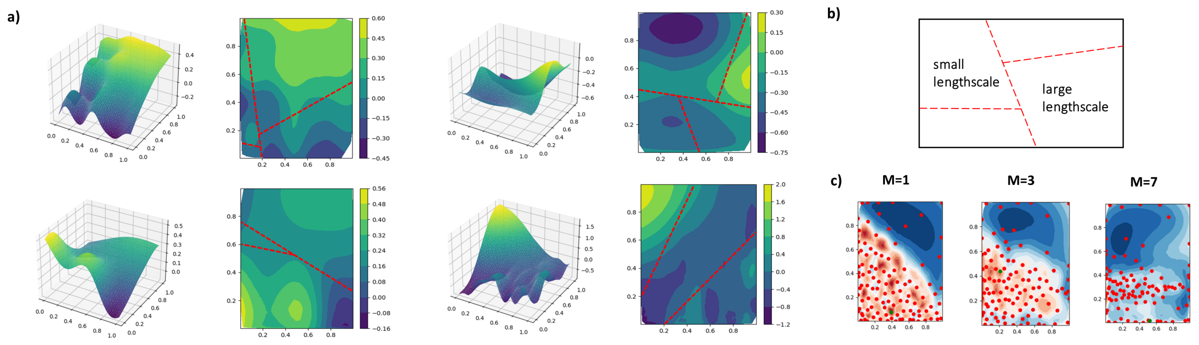

Figure 2: Prior Draws and Sampling Illustrations. a) Prior draws from a GP with HHK kernel with four latent GP's (three hyperplanes). The sampled functions (left) and the sampled hyperplanes/partitions (right) are shown - here background colors show function values. b) Conceptual illustration of the partitioning and the different kernel parameters in different regions. c) Impact of the number of hyperplanes on the actively collected samples. Here, active learning was executed on the Exponential 2D function (details in Section 4).

We assume the input to be a subset of the unit square $\mathcal{X} \subset [0,1]^d$ and treat the kernel parameters in a Bayesian way by placing priors on them. For the hyperplanes, we introduce relevance parameters that scale the hyperplanes with $\mathbf{w}_i = \alpha_i \tilde{\mathbf{w}}_i$, where $\alpha_i \in \mathbb{R}$ and $\tilde{\mathbf{w}}_i \in \mathbb{R}^{d+1}$. Concretely, we specify the following priors on α_i and $\tilde{\mathbf{w}}_i$:

$$\tilde{\mathbf{w}}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \alpha_i \sim \mathrm{Gamma}(\alpha, \beta).$$

Thus, $\mathbf{w}_i | \alpha_i \sim \mathcal{N}(0, \alpha_i^2 \mathbf{I})$. This prior on $\mathbf{w}_i$ makes only very weak assumptions about the position of the hyperplane. However, via the prior of the relevance parameter, the scaling of the hyperplane can be influenced, which affects the slope of the sigmoid functions in (1) and, thus, the overlap of the partitioning. The local kernels $k_j(x, y)$ come with their own parameters θ_j. Our default setting utilizes RBF kernels with lengthscales and variance as parameters for which we set broad Gamma priors (see Appendix B.1 for details). The parameters of the final model are denoted with $\Theta = \{\tilde{\mathbf{w}}_i, \alpha_i, \theta_j, \sigma | i = 1, \dots, M, j = 1, \dots, J\}$.

For inference, we make use of the differentiability of the kernel and employ Hamiltonian Monte Carlo (HMC) (Duane et al., 1987). After receiving n posterior samples from $p(\Theta|\mathcal{D})$ using HMC, we employ a sample-based estimate of the marginal predictive distribution for prediction, i.e.

$$\begin{aligned} p(f^*|x^*, \mathcal{D}) &= \int p(f^*|x^*, \Theta, \mathcal{D}) p(\Theta|\mathcal{D}) d\Theta \\ &\approx \frac{1}{n} \sum_{\Theta_i \sim p(\Theta|\mathcal{D})} p(f^*|x^*, \Theta_i, \mathcal{D}). \end{aligned}$$

Thus, we perform a numerical sample-based approximation to the true marginalized predictive distribution.

3.3 Induced Prior in Function Space

In Figure 2 a) samples from our proposed prior are shown for $M = 3$ hyperplanes. The specified prior includes functions that can be described with GP's with i) different lengthscales in different regions, ii) with different variances in different regions. Furthermore, through the sigmoidal gates the transitions between partitions is soft, leading to continuous sample functions. The partitions themselves rely on simple geometries via hyperplanes that cut the space hierarchically.

3.4 Comparision to other Input-Partitioning Methods

Let us emphasize some characteristics of the partitioning that distinguish our model from other input-partitioning methods like the one dimensional change-point model or the TreedGP model.

Compared to TreedGP (Gramacy and Lee, 2007), the partitioning is not restricted to be axis aligned. Secondly, the partitioning is differentiable with respect to the hyperplane parameters and the inputs due to the sigmoid functions. This leads to continuous sample paths on one hand and the possibility to use gradient-based inference schemes for the hyperplanes on the other hand.

Furthermore, our partitioning can be viewed as a d dimensional generalization of change-points (Lloyd et al., 2014; Krause and Guestrin, 2007). We inherit the advantages of change-points of providing a model that can be used with gradient-based inference methods and that lead to continuous prior draws, whereas our partitioning method is applicable to d dimensions.

Concretely, our model can be formulated in a similar way to change-points [see Lloyd et al. (2014)] via introducing the change-hyperplane operator for two kernels k_1 and k_2 in $\mathbb{R}^d$

$$\begin{aligned} k(x,y) &= \mathcal{CH}(k_1,k_2)(x,y) \\ &= \sigma(\mathbf{w}^\intercal\tilde{x})\sigma(\mathbf{w}^\intercal\tilde{y})k_1(x,y) \\ &+ \bar{\sigma}(\mathbf{w}^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}^\intercal\tilde{y})k_2(x,y) \end{aligned}$$

with $\bar{\sigma}(\mathbf{w}^\intercal\tilde{x}) = 1 - \sigma(\mathbf{w}^\intercal\tilde{x})$. The proposed kernel can be written via recursive application of the change-hyperplane operator (see Appendix A.1). Importantly, we note that generalizing change-points to d dimensions via hyperplanes preserves a simple geometry that can be deduced with few datapoints - a necessary property for the application in active learning procedures. Additionally, the formulation via the change-hyperplane operator has the benefit that it could be utilized to dynamically search over a discrete set of trees/kernel structures, similar to what is done with the change-point operator [see (Lloyd et al., 2014; Duvenaud et al., 2013; Bitzer et al., 2022)].

3.5 Sample Selection

Our goal is to provide a GP prior that is tangled to active learning settings. Thus, for the sake of simplicity we stick to the most natural way of doing active learning via querying points with highest information gain between the observation and the uncertain variables in the model (MacKay, 1992). We start with an initial dataset $\mathcal{D}_0$ and sequentially query the oracle $f\colon \mathcal{X} \to \mathbb{R}$ at point x_t and receive the noisy observation $y_t = f(x_t) + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(0,\sigma^2)$. The next datasets are build up sequentially with $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{x_t, y_t\}$. For the acquisition function, we use the maximum information gain between the observation y and the uncertain variables which are the function f and the parameters Θ. Thus, the acquisition function is given by

$$\begin{aligned} a(x|\mathcal{D}_{t-1}) &= I(y; f, \Theta|\mathcal{D}_{t-1}, x) \\ &= H(y|\mathcal{D}_{t-1}, x) \\ &- \mathbb{E}_{p(\Theta|\mathcal{D}_{t-1})}\mathbb{E}_{p(f|\Theta,\mathcal{D}_{t-1},x)}[H(y|f,\mathcal{D}_{t-1},\Theta,x)] \\ &\propto H(y|\mathcal{D}_{t-1}, x), \end{aligned}$$

where the right-hand term of the difference is independent of x as $H(y|f,\mathcal{D}_{t-1},\Theta,x) = \log(\sigma\sqrt{2\pi e})$. This acquisition function accounts for the uncertainty in the kernel parameters Θ, including the uncertainty over the hyperplanes $\mathbf{w}_i, i = 1,\ldots,M$. In order to evaluate the acquisition function, the entropy of the marginal predictive distribution needs to be approximated, e.g. by quadrature (see Appendix B.6). The next query location is then obtained by

$$x_t = \mathrm{argmax}_{x\in\mathcal{X}} a(x|\mathcal{D}_{t-1}).$$

The optimization can be performed through grid-search, random shooting or evolutionary optimizers.

3.6 Induced Sampling Behavior

Partitioning models like the proposed model have the beneficial properties that they are able to distinguish regions with different levels of contained information. For example, as illustrated in Figure 2 b), a function might be described by local GP's, where one region has large and one has small lengthscale, e.g. a function that is flat or linear in one part of the space and highly fluctuating in another. The predictive intervals thus will be larger in the region of a small lengthscale, leading to an active sampling in that region, as for example shown in Figure 2 c). Finding the right partition with few datapoints is thus crucial, as it determines the sampling. The hierarchical partitioning provides a good trade-off between flexibility (it is not constrained to axis-aligned or orthogonal partitions) and the possibility to identify partitions with few datapoints. This is further enabled through gradient-based inference methods that lead to high quality partition samples, such as HMC.

In Appendix A.2 we also add a theorem which illustrates the sampling behavior for a simplified setting, with fixed, sharp partitions and extreme values for the lengthscales and variances of the stationary kernels.

4 EXPERIMENTS

In the following section, we will evaluate the active learning performance of our prior on three tasks which exhibit nonstationarities and compare it against the main competitors. We consider small/medium-sized tasks in terms of input-dimensionality, as this is the common application field of Gaussian processes. In the last part of this section, we also investigate the influence of the tree size and the inference scheme on the active learning performance. Furthermore, we provide code for our method.[1]

Model Setup. We equip our method with a symmetric tree with eight latent GPs. As stationary latent GPs, we employ RBF kernels with Gamma priors for the lengthscale and the variance. For the noise variance, we utilize an exponential prior. We chose prior parameters that induce broad priors such that many functions have sufficient support. The prior parameters can be found in Appendix B.1. For inference, we employ a burn-in phase of 500 iterations and continue with 5000 MCMC samples that are thinned to final 100 samples.

Active Learning Setup. All tasks are available as datasets with a large number of queries already executed. The inputs are transformed to lie in the unit cube and the outputs are normalized. The optimization of the acquisition function is done with random shooting by calculating

[1]HHK-Code `https://github.com/boschresearch/Hierarchical-Hyperplane-Kernels`

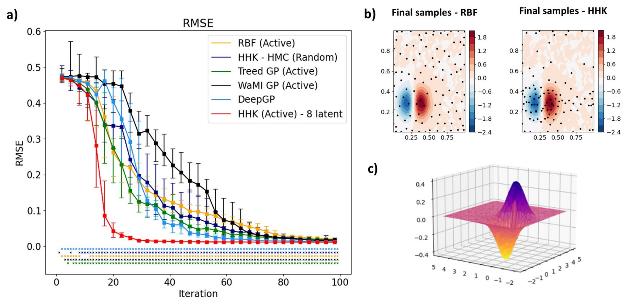

Figure 3: Exponential 2-D experiments. a) RMSE curves for the different methods. Shown are median and quartiles for each iteration over 30 runs with different initial datasets. Crosses at the bottom mark significant Wilcoxon tests in favor of our model. b) Final samples actively selected by the RBF model (left) and by our method (right) - axis are scaled to the unit cube. c) The exponential 2-D function.

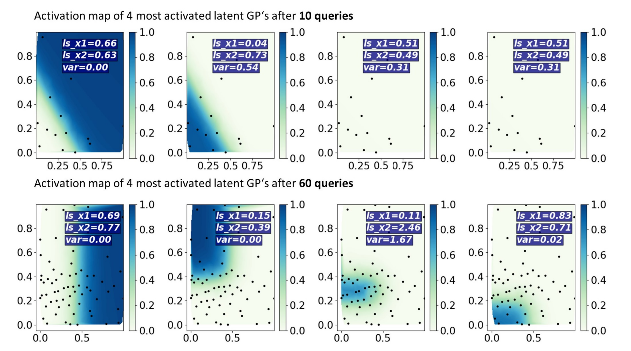

Figure 4: Activation map of the partition weights (for the MAP parameters) for the four most activated stationary kernels after 10 queries (top) and 60 queries (bottom) on the Exponential 2-D dataset.

the acquisition function on a subset of the possible queries. In practice, for an oracle with continuous input domain, one might employ Latin-Hypercube samples as grid points for optimization of the acquisition function. For evaluation, we consider the RMSE curve over the selected queries on a held out test set.

Compared Methods. We compare our method against the following methods:

a) **Random:** Here, instead of selecting the queries with maximum information gain, the queries are selected randomly.

b) **RBF:** In this case, an RBF kernel is used with the same prior on the kernel parameters as for the stationary kernels in our method, also learned with HMC and queries taken with maximum information gain.

c) **Warped Multi-Index GP:** This method is presented in Marmin et al. (2018) and uses a time-warped GP with type-2 maximum likelihood inference and maximum predictive variance as acquisition function.

d) **TreedGP:** The fully-Bayesian partitioning model presented in Gramacy and Lee (2007) and used for active learning of computer simulations in Gramacy et al. (2004). We use their *tgp* R-package and take queries via maximum information gain (the ALM criteria).

e) **DeepGP:** This approach is presented in Sauer et al. (2020) and uses DeepGPs to capture the nonstationarity. They utilize MCMC as inference and the ALC criteria Cohn et al. (1996) as acquisition function (see Appendix B.5 for further details).

4.1 Toy-function: Exponential 2-D

First, we investigate the performance of our method on the two-dimensional function

$$f(x_1, x_2) = x_1 \exp(-x_1^2 - x_2^2)$$

with $(x_1, x_2) \in [-2, 5] \times [-2, 5]$ [see Figure 3 c)]. This test function was also investigated in Gramacy et al. (2004). The function exhibits nonstationarities as it is almost flat in a large portion of the input space. Only in the lower left part of the input domain, fluctuations and larger values in the function values occur. As initial datasets we draw five data points uniformly. In comparison to the RBF model, which takes queries in a nearly space-filling manner, our method focuses the sampling on the complex region [see Figure 3 b)]. The RMSE curves are shown in Figure 3 a) for the different methods. Besides the RBF model, our method also shows significantly better results compared to the three nonstationary methods.

Impact of Partitions over the Iterations. We investigate the impact of the partitioning on the query selection. We show the activation maps of these weights for the four most activated kernels after 10 and 60 queries in Figure 4. Here, we examine the maximum a posteriori (MAP) hyperplanes $\hat{\mathbf{w}}_i$ and the resulting partition weights for the corresponding stationary kernels k_j, i.e.

$$\lambda_j^{MAP}(x) = \prod_{i=1}^{M} \sigma(\hat{\mathbf{w}}_i^{\intercal} \tilde{x})^{\xi_L(j,i)} (1 - \sigma(\hat{\mathbf{w}}_i^{\intercal} \tilde{x}))^{\xi_R(j,i)}.$$

The activation weights for all other latent GPs were nearly zero. At the beginning (after 10 queries), most of the stationary kernels are turned off and the data is explained with only two latent GPs separated by a hyperplane. After 60 queries, more kernels are active, leading to a finer partitioning. The region with large kernel variance is more focused

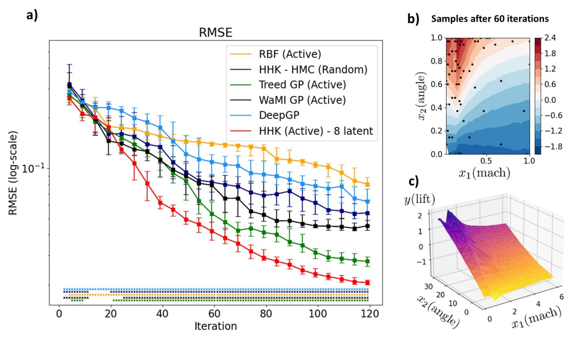

Figure 5: LGBB experiments. a) RMSE curves for the different methods. The curves show medians and quartiles for each iteration over 30 runs with different initial datasets. Colored crosses at the bottom signal significant Wilcoxon tests in favor of our model against the corresponding competitor. b) Dataset after 60 queries, selected by our method. c) LGBB response surface.

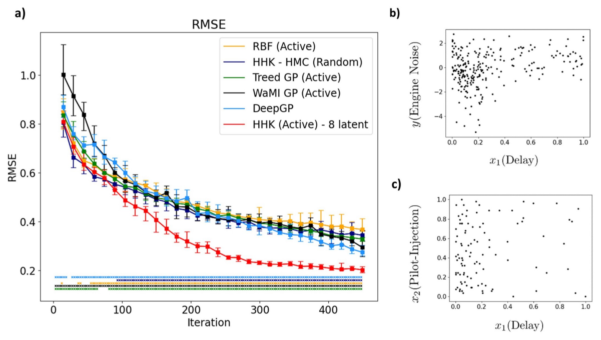

Figure 6: Combustion engine noise experiments. a) RMSE curve for the combustion engine noise dataset for the different models and querying schemes. b) Samples of x_1, y pairs to show the nonstationarity along the x_1 dimension. c) Dataset after 100 iterations actively selected by our method shown over the delay and pilot-injection dimension.

than before and almost all queries are selected in this region. We think that relying on only a few active partitions at the beginning helps to steer the samples in interesting directions.

4.2 Nasa - Langley-Glide-Back Booster

The second example consists of a computer simulation (computational fluid dynamics), employed by NASA, of a rocket reentering the atmosphere, called Langley-Glide-Back Booster (LGBB). The simulation was heavily investigated for the TreedGP model in Gramacy et al. (2004) and Gramacy and Lee (2007) as it exhibits nonstationarities and has long-lasting query times. The simulation receives as inputs the angle and speed of the rocket when entering the atmosphere, and outputs the lift that the rocket exhibits. Each query to the simulation lasts several hours. Furthermore, the simulation naturally falls into two regimes, one for a speed lower than mach one and one for higher entering velocities. Figure 5 c) shows the response surface of the simulation. The most complex region of the input space is at the upper left, where the lift value changes drastically at velocity mach one. The right part of the response surface on the other hand can be described almost by a linear function. We use the precomputed queries provided in Gramacy (2020) for our experiments. We employ five uniformly drawn initial data points and make 120 active queries afterwards. Furthermore, we add an artificial noise term onto the deterministic response surface. The noise is still learned for all considered models. Firstly, as can be seen in Figure 5 b), our method has the desired property that it focuses the samples to the complex region and only allocates very few samples to the simpler regions. This is also reflected in the predictive performance. The RMSE curves are shown in Figure 5 a). Our method leads to significantly lower RMSE values (tested with the Wilcoxon test) compared to the other methods from iteration 30 onwards.

4.3 Combustion Engine Noise

In the development process of combustion engines, a significant amount of time and cost comes from the calibration of the engine control unit (ECU) [see Tietze (2015)]. To fasten the calibration process, one possibility is to use a complete simulation of the engine for a large part of the calibration procedure. Therefore, the several engine functionalities need to be simulated, which can be done in an empirical way by learning surrogate models or by physics-based models Tietze (2015). One part of the engine calibration process is to adjust the engine noise. For that purpose, a surrogate model can be produced for the relationship between engine parameters and engine noise. The learning process bears two challenges. First, work bench time is costly and shared between several parties, and therefore the number of queries to the workbench should be kept small. Secondly, it is known that the engine noise admits major nonstationarities, in particular when varying the *delay of injections* parameter [see Tietze (2015)]. The input parameters are *delay of injections* (x_1), *volume of pilot injection* (x_2), *rail pressure* (x_3), *air mass* (x_4), *boost pressure* (x_5) and *controlstart (in ms)* (x_6) [see Tietze (2015)]. The output is the *engine noise* in dB. We start with an initial randomly drawn dataset of size five. The resulting RMSE curves can be seen in Figure 6 a). As with the other two experiments, our method provides more precise predictions with fewer selected data points. Also, as expected, the model queries more data points for lower values of x_1 as can be seen in Figure 6 c).

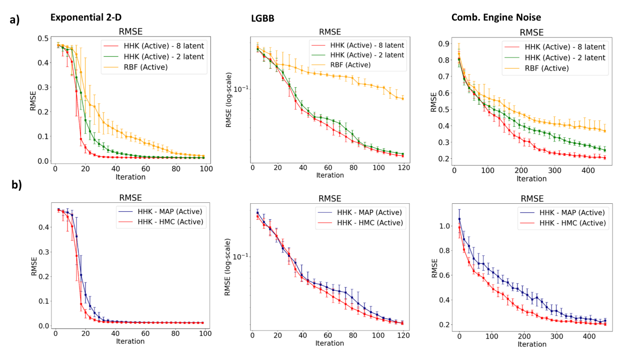

Figure 7: a) RMSE curves for different tree sizes (1 (stat. RBF), 2 and 8 latent GPs) over the three datasets. b) RMSE curves for HMC vs MAP inference for a HHK with 8 latent GPs.

4.4 Influence of the Tree Size

We examine the impact of the tree size on the performance of the active learning procedure. Figure 7 a) shows the RMSE curves for the trivial tree (stationary RBF kernel), a HHK model with two and with eight latent GPs on all considered datasets. We see that for the Exponential 2-D data and the combustion engine noise dataset, the active learning procedure benefits from larger trees. For the LGBB dataset, we can see that even adding one hyperplane seems to give a large performance gain. This is plausible since the CFD simulation has two regimes that can be distinguished with a hyperplane. For the combustion engine noise data, previous expert experience [see Tietze (2015)] has shown that nonstationarity occurs mainly within one dimension. However, we find that we gain performance by using finer partitioning with more hyperplanes [see Figure 7].

4.5 Influence of the Inference Scheme

In Figure 7 b), we investigate the influence of the kernel parameter inference on the active learning performance. To this end, we compare HMC inference with optimization of the GP hyperparameters using maximum-a-posteriori (MAP). We observe that using a Bayesian approach via HMC leads to a faster convergence compared to MAP on the higher-dimensional ClosePI task. On the other two tasks we observe a slight advantage of HMC. This also supports the recent finding of Riis et al. (2022) that marginalizing over kernel parameters is beneficial for active learning. However, we note that on LGBB and Exponential 2-D the difference between HMC and MAP ist not very large, and MAP might be a reasonable choice in cases where, for example, inference time is important (see Appendix B.6 for details on inference time).

5 CONCLUSION

In this work, we presented a new input-partitioning kernel for GP regression and investigated its active learning performance on a variety of tasks. Our method induces partitioning via a hierarchy of hyperplanes, having the advantage of preserving useful properties such as differentiability of the kernel and smooth sample functions, while keeping a simple geometry. We show that the induced active sampling focuses on the complicated region of the input space, and that our method significantly outperforms previous methods, including previous partitioning models, on real-world active learning tasks.

References

C. M. Bishop and M. Svenskn. Bayesian hierarchical mixtures of experts. UAI'03, pages 57–64, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

M. Bitzer, M. Meister, and C. Zimmer. Structural kernel search via bayesian optimization and symbolical optimal transport. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=-bLLVk-WRPy`.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1): 129145, Mar. 1996. ISSN 1076-9757.

S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195:216–222, 1987.

D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1166–1174, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

R. Garnett, M. A. Osborne, and P. Hennig. Active learning of linear embeddings for gaussian processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 230–239, Arlington, Virginia, USA, 2014. AUAI Press.

R. B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020.

R. B. Gramacy and H. Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103, 2007.

R. B. Gramacy, H. K. H. Lee, and W. G. Macready. Parameter space exploration with gaussian process trees.

In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, New York, NY, USA, 2004. Association for Computing Machinery.

M. Heinonen, H. Mannerstrm, J. Rousu, S. Kaski, and H. Lhdesmki. Non-stationary gaussian process regression with hamiltonian monte carlo. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 732–740, Cadiz, Spain, 2016. PMLR.

W. Herlands, A. Wilson, H. Nickisch, S. Flaxman, D. Neill, W. V. Panhuis, and E. Xing. Scalable gaussian processes for characterizing multidimensional change surfaces. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1013–1021. PMLR, 2016.

N. Houlsby, F. Huszr, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning, 2011. URL https://arxiv.org/abs/1112.5745.

A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: An exploration-exploitation approach. ICML '07, pages 449–456, New York, NY, USA, 2007. Association for Computing Machinery.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

C. Lee, K. Wang, J. Wu, W. Cai, and X. Yue. Partitioned active learning for heterogeneous systems, 2021. URL https://arxiv.org/abs/2105.08547.

C.-Y. Li, B. Rakitsch, and C. Zimmer. Safe active learning for multi-output gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 4512–4551. PMLR, 2022.

J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 12421250. AAAI Press, 2014.

D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4:590–604, 1992.

S. Marmin. *Warping and sampling approaches to non-stationary gaussian process modelling.* Theses, Ecole Centrale Marseille ; Université de Berne. Collegium generale, Dec. 2017.

S. Marmin, D. Ginsbourger, J. Baccou, and J. Liandrat. Warped gaussian processes and derivative-based sequential designs for functions with heterogeneous variations. *SIAM/ASA Journal on Uncertainty Quantification*, 6: 991–1018, 2018.

Rasmussen and Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, Jan. 2006.

S. Remes, M. Heinonen, and S. Kaski. Non-stationary spectral kernels. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

C. Riis, F. N. Antunes, F. B. Httel, C. L. Azevedo, and F. C. Pereira. Bayesian active learning with fully bayesian gaussian processes, 2022. URL https://arxiv.org/abs/2205.10186.

A. Sauer, R. B. Gramacy, and D. Higdon. Active learning for deep gaussian process surrogates. *arXiv preprint arXiv:2012.08015*, 2020.

J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe exploration for active learning with gaussian processes. In *Proceedings of the 2015th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, ECMLPKDD'15, pages 133–149. Springer, 2015.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

J. Snoek, K. Swersky, R. Zemel, and R. Adams. Input warping for bayesian optimization of non-stationary functions. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1674–1682, Bejing, China, 2014. PMLR.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. ICML'10, pages 1015–1022, Madison, WI, USA, 2010. Omnipress.

N. Tietze. *Model-based Calibration of Engine Control Units Using Gaussian Process Regression.* PhD thesis, Technische Universität, Darmstadt, February 2015.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 2016. PMLR.

X. Yue, Y. Wen, J. H. Hunt, and J. Shi. Active learning for gaussian process considering uncertainties with application to shape control of composite fuselage. *IEEE Transactions on Automation Science and Engineering*, 18:36–46, 2021.

C. Zimmer, M. Meister, and D. Nguyen-Tuong. Safe active learning for time-series modeling with gaussian processes. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

A METHODOLOGICAL CONSIDERATIONS

A.1 Formulation via Change-Hyperplanes

In Lloyd et al. (2014) the change-point operator is introduced for two kernels $k_1(x, y)$ and $k_2(x, y)$ and a point $s \in \mathbb{R}$

$$\mathcal{CP}_s(k_1, k_2)(x, y) = \sigma(x - s)\sigma(y - s)k_1(x, y) + \bar{\sigma}(x - s)\bar{\sigma}(y - s)k_2(x, y)$$

with $\bar{\sigma}(x-s) = 1-\sigma(x-s)$ and $x, y \in \mathbb{R}$. Multiple change-points can be applied via iterative application of this operator. For example, for three change-points $s_1, s_2, s_3 \in \mathbb{R}$ the final kernel can be defined as

$$k(x, y) = \mathcal{CP}_{s_1}(\mathcal{CP}_{s_2}(k_1, k_2), \mathcal{CP}_{s_3}(k_3, k_4)),$$

In case $k_1, \dots, k_4$ are stationary, the final kernel would induce functions that behave stationary in-between the change-points, but are less correlated/behave differently between two separate intervals.

The change-point operator is defined on one-dimensional inputs, e.g. $s, x, y \in \mathbb{R}$. Our kernel can be viewed as a d dimensional generalization of change-points. We illustrate this for the example in Figure 1 (in the main paper) with a symmetric tree with three hyperplanes. We denote $\tilde{x} = (1, x)^\intercal \in \mathbb{R}^{d+1}$ and define the change-hyperplane operator as

$$\mathcal{CH}_{\mathbf{w}}(k_1, k_2)(x, y) = \sigma(\mathbf{w}^\intercal\tilde{x})\sigma(\mathbf{w}^\intercal\tilde{y})k_1(x, y) + \bar{\sigma}(\mathbf{w}^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}^\intercal\tilde{y})k_2(x, y), \quad x, y \in \mathbb{R}^d,$$

where $\bar{\sigma}(\mathbf{w}^\intercal\tilde{x}) = 1 - \sigma(\mathbf{w}^\intercal\tilde{x})$. For the HHK with three hyperplanes [Figure 1 a)], the HHK weights $\lambda_j(x)$ result to:

$$\begin{aligned}
\lambda_1(x) &= \prod_{i=1}^{3} \sigma(\mathbf{w}_i^\intercal\tilde{x})^{\xi_L(1,i)}(1 - \sigma(\mathbf{w}_i^\intercal\tilde{x}))^{\xi_R(1,i)} = \sigma(\mathbf{w}_1^\intercal\tilde{x})\sigma(\mathbf{w}_2^\intercal\tilde{x}),\\
\lambda_2(x) &= \prod_{i=1}^{3} \sigma(\mathbf{w}_i^\intercal\tilde{x})^{\xi_L(2,i)}(1 - \sigma(\mathbf{w}_i^\intercal\tilde{x}))^{\xi_R(2,i)} = \sigma(\mathbf{w}_1^\intercal\tilde{x})(1 - \sigma(\mathbf{w}_2^\intercal\tilde{x})) = \sigma(\mathbf{w}_1^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_2^\intercal\tilde{x}),\\
\lambda_3(x) &= \prod_{i=1}^{3} \sigma(\mathbf{w}_i^\intercal\tilde{x})^{\xi_L(3,i)}(1 - \sigma(\mathbf{w}_i^\intercal\tilde{x}))^{\xi_R(3,i)} = (1 - \sigma(\mathbf{w}_1^\intercal\tilde{x}))\sigma(\mathbf{w}_3^\intercal\tilde{x}) = \bar{\sigma}(\mathbf{w}_1^\intercal\tilde{x})\sigma(\mathbf{w}_3^\intercal\tilde{x}),\\
\lambda_4(x) &= \prod_{i=1}^{3} \sigma(\mathbf{w}_i^\intercal\tilde{x})^{\xi_L(4,i)}(1 - \sigma(\mathbf{w}_i^\intercal\tilde{x}))^{\xi_R(4,i)} = (1 - \sigma(\mathbf{w}_1^\intercal\tilde{x}))(1 - \sigma(\mathbf{w}_3^\intercal\tilde{x})) = \bar{\sigma}(\mathbf{w}_1^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_3^\intercal\tilde{x}),
\end{aligned}$$

Thus, the kernel can be written via the change-hyperplane operator as

$$\begin{aligned}
k(x, y) &= \sum_{j=1}^{4} \lambda_j(x)\lambda_j(y)k_j(x, y)\\
&= \sigma(\mathbf{w}_1^\intercal\tilde{x})\sigma(\mathbf{w}_2^\intercal\tilde{x})\sigma(\mathbf{w}_1^\intercal\tilde{y})\sigma(\mathbf{w}_2^\intercal\tilde{y})k_1(x, y) + \sigma(\mathbf{w}_1^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_2^\intercal\tilde{x})\sigma(\mathbf{w}_1^\intercal\tilde{y})\bar{\sigma}(\mathbf{w}_2^\intercal\tilde{y})k_2(x, y)\\
&+ \bar{\sigma}(\mathbf{w}_1^\intercal\tilde{x})\sigma(\mathbf{w}_3^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_1^\intercal\tilde{y})\sigma(\mathbf{w}_3^\intercal\tilde{y})k_3(x, y) + \bar{\sigma}(\mathbf{w}_1^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_3^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_1^\intercal\tilde{y})\bar{\sigma}(\mathbf{w}_3^\intercal\tilde{y})k_4(x, y)\\
&= \sigma(\mathbf{w}_1^\intercal\tilde{x})\sigma(\mathbf{w}_1^\intercal\tilde{y})\Big(\sigma(\mathbf{w}_2^\intercal\tilde{x})\sigma(\mathbf{w}_2^\intercal\tilde{y})k_1(x, y) + \bar{\sigma}(\mathbf{w}_2^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_2^\intercal\tilde{y})k_2(x, y)\Big)\\
&+ \bar{\sigma}(\mathbf{w}_1^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_1^\intercal\tilde{y})\Big(\sigma(\mathbf{w}_3^\intercal\tilde{x})\sigma(\mathbf{w}_3^\intercal\tilde{y})k_3(x, y) + \bar{\sigma}(\mathbf{w}_3^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_3^\intercal\tilde{y})k_4(x, y)\Big)\\
&= \sigma(\mathbf{w}_1^\intercal\tilde{x})\sigma(\mathbf{w}_1^\intercal\tilde{y})\mathcal{CH}_{\mathbf{w_2}}(k_1, k_2)(x, y) + \bar{\sigma}(\mathbf{w}_1^\intercal\tilde{x})\bar{\sigma}(\mathbf{w}_1^\intercal\tilde{y})\mathcal{CH}_{\mathbf{w_3}}(k_3, k_4)(x, y)\\
&= \mathcal{CH}_{\mathbf{w_1}}(\mathcal{CH}_{\mathbf{w_2}}(k_1, k_2), \mathcal{CH}_{\mathbf{w_3}}(k_3, k_4))(x, y)
\end{aligned}$$

Thus, one might interpret the HHK as iterative application of the change-hyperplane operator along the nodes of its tree.

A.2 Theory: Sampling Behavior of Partitioning Models

In the following subsection, f is drawn from a Gaussian process with kernel k, $A = \{x_1, \dots, x_T\} \subset D$ is a set of input locations with $D \subset \mathcal{X}$ compact, $f_A = [f(x)]_{x\in A}$, and $y_A = f_A + \epsilon_A$ is the resulting observation with noise

$\epsilon_A \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$. The Shannon information $I(\cdot)$ between observations and latent function can be calculated in closed form with $I(y_A, f|k) = I(y_A, f_A|k) = \frac{1}{2}\log|\mathbf{I} + \sigma^{-2} K_A|$, where $K_A = [k(x,y)]_{x,y\in A}$ is the Gram matrix at locations A.

We want to gain theoretical insights into the sampling behavior that is induced by partitioning kernels. For that, we take a detailed look at the sampling behavior in some extreme cases of the kernel parameters. We consider the general partitioning kernel $k(x,y) = \sum_{j=1}^{J} \mathbf{1}_{\{x,y\in D_j\}} k_j^{l_j,\sigma_j^2}(x,y)$ for some partition $D_j, j = 1,\dots,J$, of the input space. Here, $k_j^{l_j,\sigma_j^2}, j = 1,\dots,J$ are stationary kernels with lengthscales l_j and variances σ_j^2, for example $k_j^{l_j,\sigma_j^2} = \sigma_j^2 \exp\left(-\frac{\|x-y\|_2^2}{l_j^2}\right)$. We consider a simplified setting in the sense that the partitions are sharp (defined via indicator functions) and furthermore the kernel parameters, including the partitions, are considered *fixed*. Our kernel becomes this form in the limit of the smoothness parameter in the sigmoid function (see Lemma 1). Furthermore, the MAP estimate of the TreedGP model would be a special case of this kernel family (that has axis-aligned partitions D_j). We note that in case of fixed kernel parameters the maximum information gain acquisition function simplifies to

$$a(x|\mathcal{D}_{t-1}) = I(y; f|\mathcal{D}_{t-1}, x, \Theta) \propto \sigma_{t-1}^2(x|\Theta) \tag{2}$$

where $\sigma_{t-1}^2(x|\Theta)$ is the predictive variance of the posterior GP $f|\mathcal{D}_{t-1}$ with parameters Θ. The acquisition function (2) is the greedy step to maximize the complete information $I(y_A, f_A|k)$ [see Srinivas et al. (2010)]. We will therefore analyze the samples A that maximize $I(y_A, f_A|k)$ with $|A| = T$ and $T \in \mathbb{N}$. In Theorem 1 we take a detailed look at the optimally selected samples, when the kernel exhibits the extreme cases, where either the variance of one local kernel is zero or its lengthscale goes to infinity.

The theorem states that, when the partition is sharp, the sampling will ignore the region, where the kernel variance is zero and will only need a tiny fraction of the region, where the kernel lengthscale is very large. A look into the proof reveals that, in the regions that exhibit large lengthscales, the function can be considered almost as constant, such that datapoints from a tiny portion of this area are sufficient to deduce the function values in the remaining part. In the regions exhibiting zero variance, there is no information left, such that no datapoints need to be gathered here. Instead, the samples can be allocated to regions where more information is present. In practice, the variance will not be zero and the lengthscale might have very large, but finite values. In the experimental section of the main paper we see examples for input regions that exhibit very small variance or large lengthscales, and we observe the indicated sampling behavior.

It is important to note that, in practice, the correct identification of the different partitions is crucial for the sampling behavior to behave as described. This is where the properties of our kernel play an important role, as the hierarchical hyperplanes allow for flexible partitions that can be learned efficiently due to the differentiability of the kernel.

Theorem 1 *Let $D \subset \mathbb{R}^d$ be compact and $D_j \subset D, j = 1,\dots,J$, with non-empty interior and such that $D_i \cap D_j = \emptyset$ and $\bigcup_{j=1}^{J} D_j = D$. Let $k(x,y) = \sum_{j=1}^{J} \mathbf{1}_{\{x,y\in D_j\}} k_j^{l_j,\sigma_j^2}(x,y)$, where $k_j^{l_j,\sigma_j^2}, j = 1,\dots,J$ are stationary kernels with lengthscales l_j and variances σ_j^2, e.g. $k_j^{l_j,\sigma_j^2} = \sigma_j^2 \exp\left(-\frac{\|x-y\|_2^2}{l_j^2}\right)$. Consider the two cases that for some $i \in \{1,\dots,J\}$ either 1) the variance is $\sigma_i^2 = 0$ or 2) the lengthscale $l_i \to \infty$. Denote in both cases the resulting kernel on the full input space with $k^*(x,y)$ (which is the p.w. limit for case 2). Then it holds:*

For case 1):

$$\begin{aligned} &\exists X^* = \{x_1,\dots,x_T\} \subset D \setminus D_i : \\ &X^* \in \underset{A\subset D, |A|=T}{\operatorname{argmax}}\ I(y_A, f_A|k^*). \end{aligned}$$

For case 2): Let ξ be an arbitrary interior point in D_i. Then for some $a > 0$ and any $0 < \epsilon < a$:

$$\begin{aligned} &\exists X^* = \{x_1,\dots,x_T\} \subset (D \setminus D_i) \cup B_\epsilon(\xi) : \\ &X^* \in \underset{A\subset D, |A|=T}{\operatorname{argmax}}\ I(y_A, f_A|k^*), \end{aligned}$$

where $B_\epsilon(\xi)$ is the open ball around ξ with radius ϵ.

Proof. Let $A \subset D$ with $|A| = T$. Now, let $A_j := A \cap D_j$. Then $\bigcup_{j=1}^{J} A_j = A$ and $A_j \cap A_{j'} = \emptyset$ for all $j \neq j'$. As

$k(x, y) = 0$ for $x \in D_j$ and $y \in D_{j'}$ with $j \neq j'$, it holds that $y_{A_j}, j = 1, \ldots, J$ are independent from each other, hence,

$$H(y_A) = \sum_{j=1}^{J} H(y_{A_j}).$$

Additionally, $y_{A_j}|f_{A_j}, j = 1, \ldots, J$ are independent (as the noise term is i.i.d.) and, thus,

$$H(y_A|f_A) = \sum_{j=1}^{J} H(y_{A_j}|f_{A_j}).$$

We denote $K_A^j := [k_j(x, y)]_{x,y\in A}$ and note that $K_{A_j} = K_{A_j}^j$ for all $j = 1, \ldots, J$ as $A_j \subset D_j$. Therefore, it holds,

$$\begin{aligned} I(y_A, f_A|k) &= H(y_A) - H(y_A|f_A) = \sum_{j=1}^{J} H(y_{A_j}) - \sum_{j=1}^{J} H(y_{A_j}|f_{A_j}) \\ &= \sum_{j=1}^{J} I(y_{A_j}, f_{A_j}|k) = \sum_{j=1}^{J} \frac{1}{2}\log|\mathbf{I} + \sigma^{-2}K_{A_j}| = \sum_{j=1}^{J} \frac{1}{2}\log|\mathbf{I} + \sigma^{-2}K_{A_j}^j| \\ &= \sum_{j=1}^{J} I(y_{A_j}, f_{A_j}|k_j). \end{aligned}$$

Case 1): For this case, where $\sigma_i^2 = 0$ for some $i \in \{1, \ldots, J\}$, it additionally holds,

$$I(y_A, f_A|k^*) = \sum_{j=1}^{J} I(y_{A_j}, f_{A_j}|k_j) = \sum_{j\neq i} I(y_{A_j}, f_{A_j}|k_j),$$

because $k_i(x, y) = 0$ is the trivial kernel, since $\sigma_i^2 = 0$, and it holds

$$I(y_{A_i}, f_{A_i}|k_i) = \frac{1}{2}\log|\mathbf{I} + \sigma^{-2}K_{A_i}^i| = \frac{1}{2}\log|\mathbf{I}| = \frac{1}{2}\log|1| = 0.$$

This yields

$$\begin{aligned} \max_{A\subset D,|A|=T} I(y_A, f_A|k^*) &= \max_{A\subset D,|A|=T} \sum_{j=1}^{J} I(y_{A\cap D_j}, f_{A\cap D_j}|k_j) = \max_{A\subset D,|A|=T} \sum_{j\neq i} I(y_{A\cap D_j}, f_{A\cap D_j}|k_j) \\ &= \max_{A\subset D\setminus D_i,|A|=T} \sum_{j\neq i} I(y_{A\cap D_j}, f_{A\cap D_j}|k_j) = \max_{A\subset D\setminus D_i,|A|=T} I(y_A, f_A|k^*). \end{aligned}$$

Case 2): For the second case, where $l_i \to \infty$ for some $i \in \{1, \ldots, J\}$, we denote the pointwise (p.w.) limit of $k_i(x, y)$ with $k_i^*(x, y)$ and it holds that $k_i^*(x, y) = c$ with some constant $c > 0$ (actually it holds that $c = \sigma_i^2$). We denote $K_A^{*,i} := [k_i^*(x, y)]_{x,y\in A}$. Let ξ be an interior point in D_i and $a > 0$ such that $B_a(\xi) \subset D_i$. Then, for any $\tilde{A}_i \subset B_\epsilon(\xi)$ with $|\tilde{A}_i| = |A_i|$ and $0 < \epsilon < a$ it holds that $K_{A_i}^{*,i} = K_c = K_{\tilde{A}_i}^{*,i}$ with $K_c := [c]_{x,y\in A_i}$ and thus

$$I(y_{A_i}, f_{A_i}|k_i^*) = \frac{1}{2}\log|\mathbf{I} + \sigma^{-2}K_c| = I(y_{\tilde{A}_i}, f_{\tilde{A}_i}|k_i^*).$$

Thus, we obtain

$$\begin{aligned} \max_{A\subset D,|A|=T} I(y_A, f_A|k^*) &= \max_{A\subset D,|A|=T} \left(I(y_{A\cap D_i}, f_{A\cap D_i}|k_i^*) + \sum_{j\neq i} I(y_{A\cap D_j}, f_{A\cap D_j}|k_j) \right) \\ &= \max_{A\subset (D\setminus D_i)\cup B_\epsilon(\xi),|A|=T} \left(I(y_{A\cap B_\epsilon(\xi)}, f_{A\cap B_\epsilon(\xi)}|k_i^*) + \sum_{j\neq i} I(y_{A\cap D_j}, f_{A\cap D_j}|k_j) \right) \\ &= \max_{A\subset (D\setminus D_i)\cup B_\epsilon(\xi),|A|=T} I(y_A, f_A|k^*). \quad \blacksquare \end{aligned}$$

The following lemma shows that the Hierarchical-Hyperplane Kernel reaches the considered form $k(x,y) = \sum_{j=1}^{J} \mathbf{1}_{\{x,y\in D_j\}} k_j^{l_j,\sigma_j^2}(x,y)$ for some partition $D_j, j = 1,\dots,J$ in the limit of the smoothness parameter. Furthermore, the proof reveals the exact form of the sharp partitions D_j.

Lemma 1 *Let* $k(x,y|[\mathbf{w}_i]) = \sum_{j=1}^{J} \lambda_j(x,[\mathbf{w}_i])\lambda_j(y,[\mathbf{w}_i])k_j(x,y)$ *be the Hierarchical Hyperplane Kernel (HHK), thus* $\lambda_j(x,[\mathbf{w}_i]) = \prod_{i=1}^{M} \sigma(\mathbf{w}_i^\intercal \tilde{x})^{\xi_L(j,i)}(1-\sigma(\mathbf{w}_i^\intercal \tilde{x}))^{\xi_R(j,i)}$ *with* $\tilde{x} = (1,x)^\intercal$ *and* k_j *be kernels on* $\mathbb{R}^d$*. Then there exists sequences* $(\mathbf{w}_i^{(n)})_{n\geq 0}$ *with* $\|\mathbf{w}_i^{(n)}\| \to \infty$*, for all* $i = 1,\dots,M$*, and* $D_j \subset \mathbb{R}^d, j = 1,\dots,J$ *such that*

$$k(x,y|[\mathbf{w}_i^{(n)}]) \to \sum_{j=1}^{J} \mathbf{1}_{\{x,y\in D_j\}} k_j(x,y)$$

pointwise.

Proof. Let $\alpha_n = n^2$, $\beta_n = \frac{1}{n}$ and

$$\mathbf{w}_i^{(n)} := \alpha_n \left(\tilde{\mathbf{w}}_i + \beta_n \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right)$$

for some fixed $\tilde{\mathbf{w}}_i, i = 1,\dots,M$. Then, for all $i = 1,\dots,M$, it holds

$$\sigma(\mathbf{w}_i^{(n)\intercal}\tilde{x}) = \frac{1}{1+\exp(-\alpha_n(\tilde{\mathbf{w}}_i^\intercal \tilde{x} + \beta_n))} \to \begin{cases} 1 & \text{for} \quad \tilde{\mathbf{w}}_i^\intercal \tilde{x} \geq 0, \\ 0 & \text{for} \quad \tilde{\mathbf{w}}_i^\intercal \tilde{x} < 0. \end{cases}$$

Therefore, we obtain

$$\begin{aligned} \lambda_j(x,[\mathbf{w}_i^{(n)}]) &= \prod_{i=1}^{M} \sigma(\mathbf{w}_i^{(n)\intercal}\tilde{x})^{\xi_L(j,i)}(1-\sigma(\mathbf{w}_i^{(n)\intercal}\tilde{x}))^{\xi_R(j,i)} \\ &\to \begin{cases} 1 & \text{if for all } \ i \ \text{ with } \ \xi_L(j,i) = 1 : \tilde{\mathbf{w}}_i^\intercal \tilde{x} \geq 0 \ \text{ and all } \ i \ \text{ with } \ \xi_R(j,i) = 1 : \tilde{\mathbf{w}}_i^\intercal \tilde{x} < 0, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

We set

$$D_j := \{x \in \mathbb{R}^d : \lambda_j(x,[\mathbf{w}_i^{(n)}]) \to 1\}$$

so that

$$\lambda_j(x,[\mathbf{w}_i^{(n)}])\lambda_j(y,[\mathbf{w}_i^{(n)}]) \to \mathbf{1}\{x,y \in D_j\}$$

pointwise. ■

A.3 Reasoning that HHK is Nonstationary

In order for active learning to focus on different parts of the input space, the kernel needs to be nonstationary. The HHK is in general not stationary (thus, nonstationary). This can easily be seen in case of a completely sharp partition as introduced in Section A.2. We recall that for a kernel on $\mathbb{R}^d$ to be stationary, it must hold that $k(x,y) = k(x+a,y+a)$ for all $x,y \in \mathbb{R}^d$ and all $a \in \mathbb{R}$. In case of the HHK with sharp partitions, denoted with k, we consider $x,y \in D_j$ and $a \in \mathbb{R}$ such that $x+a, y+a \in D_{j'}$, with $j \neq j'$. If we place different stationary kernels k_j and $k_{j'}$ into the regions with $k_j(x',y') \neq k_{j'}(x',y')$ for all $x',y' \in \mathbb{R}^d$ the characterizing equality for stationarity does not hold as

$$k(x,y) = k_j(x,y) \neq k_{j'}(x,y) = k_{j'}(x+a,y+a) = k(x+a,y+a).$$

Thus, k is not stationary. The property $k_j(x',y') \neq k_{j'}(x',y')$ for all $x',y' \in \mathbb{R}^d$ holds, for example, when the kernel variances of two SE kernels differ.

B EXPERIMENTAL DETAILS

B.1 Prior Parameters for HHK

Throughout the experiments, we use the following prior parameters for the Hierarchical Hyperplane Kernel, where the local kernels $k_j(x, y)$ are Squared-Exponential Kernels on $\mathbb{R}^d$ with lengthscales $l_{i,j}, i = 1, \dots, d$ and variance σ_j^2:

Table 1: Parameters of the Prior

Variable	Prior	Parameters
$l_{i,j}$	$\text{Gamma}(\alpha, \beta)$	$\alpha = 2, \beta = 2$
σ_j^2	$\text{Gamma}(\alpha, \beta)$	$\alpha = 2, \beta = 3$
α_j	$\text{Gamma}(\alpha, \beta)$	$\alpha = 6, \beta = 2$
$\tilde{\mathbf{w}}_i$	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	
σ^2	$\text{Exp}(\lambda)$	$\lambda = 10$

All prior parameters were chosen such that many functions $f : [0, 1]^d \to \mathbb{R}$ have sufficient support in the resulting prior in function space.

B.2 Dataset Preparation

All three datasets/tasks contained already executed queries. The input variables were transformed to the unit interval and the output was normalized. This was mainly done for the reason that the GP priors with the described prior parameters have support over the dataset, for the HHK model, but also for the TreedGP and RBF model (all models with priors on the kernel parameters). We don't see that as a restrictive assumption for real-world settings as often upper and lower bounds for input and output values are given and one might either rescale input and output variables or rescale the prior parameters.

B.3 Inference Time Comparision

In Figure 8, we show the inference times for the different kernels (HHK + RBF) and for the different inference schemes (HMC and MAP) on the LGBB dataset for different number of datapoints. For MAP inference, we do an optimization with ten restarts to avoid local maxima. We note that in practice this could be parallelized in order to decrease inference times. As the number of parameters in the model increases with the number of hyperplanes, it is clear that also the inference time increases with more hyperplanes. This holds for MAP as well as for HMC as can be seen in Figure 8. However, we note that the impact of the inference time on the active learning cycle is only relevant if it makes up a significant amount of the oracle time. To illustrate that we show in the right plot of Figure 8 a comparision of the inference times to an artificial oracle time of 1h. In this example the inference time would not have a big impact on the active learning performance.

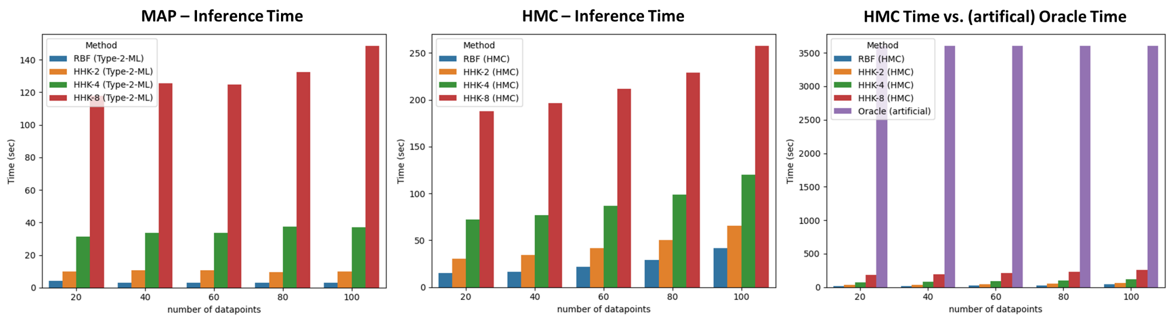

Figure 8: We show the computation time for model inference for the different HHK configurations and the RBF kernel on the LGBB dataset for different number of datapoints. In the left plot, MAP inference is shown, where the maximization is repeated ten times for inference. In the middle, the inference time for HMC is shown and in the right plot the same bars are shown with an additional bar that indicates an artificial oracle time of 1h in the active learning cycle.

B.4 Parameterization of Warped Multi-Index Kernel

We use the parameterization of the Warped Multi Index Gaussian process (WaMI) model described in the PhD thesis of Sébastien Marmin (Marmin (2017), p.45-47). The parameterization given in Marmin et al. (2018) can only be used for two-dimensional problems. Inside the described model in Marmin (2017) they use the Beta distribution as warping function. For computational reasons we used the almost identical Kumaraswamy distribution as a replacement, which has a simpler parameterization that can be easily differentiated with automatic differentiation software (this is a commonly used alternative to the Beta distribution also used for example in the *Spearmint* package for Bayesian Optimization).

B.5 Details on DeepGP experiments

We used the R-package *deepgp* associated with the method in Sauer et al. (2020) which implements DeepGPs with Slice-Sampling inference and the calculation of the ALC acquisition function. We chose a two-layer DeepGP for the experiments as it showed stable and good results in Sauer et al. (2020). The ALC criteria needs a reference input-set $\mathcal{X}_{ref}$. All three datasets/tasks contained already executed queries (see B.2) with input locations that were either uniform random (Exponential2D, ClosePI) or a grid of input locations (LGBB). We used the complete set of input locations as reference input-set $\mathcal{X}_{ref}$.

B.6 Approximation of the Predictive Entropy

The estimate of the predictive distribution $p(y|x)$ given by HMC is a mixture of Gaussians:

$$p(y|x) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{N}(y; \mu_i(x), \sigma_i^2(x))$$

As there is no analytical formulation for the Entropy $H(y|x) = \int p(y|x)\log(p(y|x))dy$ of a mixture of Gaussians we need to approximate this quantity/integral. First, the Gaussian mixture is a density in 1D which already simplifies computations. Still, we evaluated different approximations: sampling-based approximations, first- and second Taylor approximations and quadrature. We found quadrature to have the best cost-to-precision ratio and used the SciPy implementation for 1D integrals with integration bounds $\min_{i=1,\ldots,n}\{\mu_i - 2\sigma_i\}$ and $\max_{i=1,\ldots,N}\{\mu_i + 2\sigma_i\}$ (the two-sigma quantiles of the left and right-most MCMC samples) in order to concentrate the quadrature to the regions with higher density. We also think a reason for the quadrature to work accurately in this case is that the predictive distribution has an almost Gaussian-like shape, at least if a sufficient number of MCMC samples is used. Lastly, we note that doing quadrature still can have a computational overhead if it is done in sequence over a set of evaluation points. Thus, in case of shorter oracle times, it is recommended to parallelize this computation over several CPU cores.