Class GP: Gaussian Process Modeling for Heterogeneous Functions

Mohit Malu[1,3], Giulia Pedrielli[2], Gautam Dasarathy[1], and Andreas Spanias[1,3]

[1] School of ECEE, Arizona State University, Tempe AZ 85281, USA
[2] SCAI, Arizona State University, Tempe AZ 85281, USA
[3] SenSIP Center
{mmalu,giulia.pedrielli,gautamd,spanias}@asu.edu

Abstract. Gaussian Processes (GP) are a powerful framework for modeling expensive black-box functions and have thus been adopted for various challenging modeling and optimization problems. In GP-based modeling, we typically default to a stationary covariance kernel to model the underlying function over the input domain, but many real-world applications, such as controls and cyber-physical system safety, often require modeling and optimization of functions that are locally stationary and globally non-stationary across the domain; using standard GPs with a stationary kernel often yields poor modeling performance in such scenarios. In this paper, we propose a novel modeling technique called Class-GP (Class Gaussian Process) to model a class of heterogeneous functions, i.e., non-stationary functions which can be divided into locally stationary functions over the partitions of input space with one active stationary function in each partition. We provide theoretical insights into the modeling power of Class-GP and demonstrate its benefits over standard modeling techniques via extensive empirical evaluations.

Keywords: Gaussian process · Black-box modeling · Heterogeneous function · Non-stationary function modeling · Optimization

1 Introduction

Many modern day science and engineering applications, such as machine learning, hyperparameter optimization of neural networks, robotics, cyber-physical systems, etc., call for modeling techniques to model black-box functions. Gaussian Process (GP) modeling is a popular Bayesian non-parametric framework heavily employed to model expensive black-box functions for analysis such as prediction or optimization [20]. Traditionally, GP models assume stationarity of the underlying, unknown, function. As a result a unique covariance kernel (with constant hyperparameters) can be used over the entire domain. However, many real-world systems such as cyber-physical, natural, recommendation can only be characterized by locally stationary but globally non stationary functions. Breaking the assumption underlying the stationary kernel can deteriorate the quality of predictions generated by the GP.

Many studies in the literature tackle this problem. We can classify these studies in to three categories:
1. *Locally stationary and partition based approaches*: The work by Gramacy et al., [6] is one of the first ones to tackle the modeling of heterogeneous functions by partitioning

the input space with tree-based algorithms and using independent stationary GPs to model the underlying function. Kim et al., [10] and Pope et al., [19] propose Voronoi tessellations based partitioning of the input space. Candelieri et al.,[2] extends the work [6] by overcoming the modeling limitation of axis aligned partitions by using a support vector machine (SVM) based classifiers at each node of the tree. Fuentes et al., [4] uses an alternative kernel which is modeled as a convolution of fixed kernel with independent stationary kernel whose parameters vary over the sub regions of the space.
2. *GP's with non-stationary kernels*: The studies [18,17,5] use non-stationary kernels to model the heterogeneous function, [8] uses non stationary kernels with a input dependent kernel parameters and further models the parameter functions with a smooth stationary Gaussian process. However, the use of non stationary kernel makes these methods complex and computationally intractable.
3. *Warping or spatial deformation based methods*: Methods in [21,15] map the original input space to a new space where the process is assumed to be stationary, [3] uses composite GP to warp the input space.

For many engineering systems, the structure of the non-stationariety is known or can be evaluated. As an example, vehicle automatic transmission will exhibit switching behaviors, with a discrete and finite number of switches (gears changes). When a specific behavior (gear) is exercised, the system exhibits smooth state dynamics and the metrics associated to the system that we are interested in monitoring/predicting maintain such smoothness. The work [16], has considered the case of identifying unsafe system-level behaviors for Cyber-Physical Systems without exploiting any information about, for example, switching dynamics.

In this paper, we make a first step in the direction of improving analysis of systems that are characterized by a discrete and finite number of "classes" of behaviors. Notice that a single class may be represented by disconnected sets of the input space. In particular, given an input, we assume the class and the closest class that the system can switch to, can be both evaluated. Under this scenario, we extend the existing partition based methods to a family of heterogeneous functions adding the class information. We model the homogeneous behavior classes by imposing that the GPs learnt within the subregions of the input space that belong to the same class have same kernel hyperparameters. These functions are often encountered in many real world applications, where we can access the class information by learning a classifier using the features. To the best of our knowledge, we present a first tree-based algorithm with information sharing across non-contiguous partitions that belong to same homogeneous behaviour class to better learn the GP models. Our contributions include:

- A novel Class GP framework to model heterogeneous function with class information.
- Theoretical analysis - we compute uniform error bounds for our framework and compare it with the error bounds achieved by standard GP.
- Empirical analysis - we provide extensive empirical evaluations of the model and compare it with other modeling techniques.

The rest of the paper is organized as follows: Section 2 gives a formal introduction to the problem and notations used in the paper followed by a brief overview of Gaussian process modeling and classification tree algorithm in section 3 and introducing the

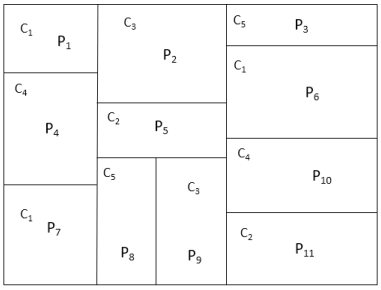


Fig. 1. Class-partition space with axis aligned partitions (P_j) and classes (C_i)

Class GP framework in Section 4. Section 5 provides theoretical insights for Class-GP algorithm followed by details of experimental setup and corresponding results in section 6. Section 7 gives conclusion over the performance of Class-GP as compared to other methods and insights on future work. Finally, paper ends with an appendix in Section 8.

2 Problem Setup and Notation

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a compact space with p axis aligned partitions $\{\mathcal{X}_j\}_{j=1}^{p}$ and each partition $j \in [p]$ is assigned a class label $i \in [m]$ i.e., $\mathbb{C}(j) = i$, we call this space as *class-partition* space. This paper, models a family of non-stationary functions f defined over class-partition space, $f : \mathcal{X} \to \mathbb{R}$, such that f boils down to a stationary functions g_j's over each partition $j \in [p]$ where each g_j's are sampled from a Gaussian process with a continuous stationary kernel κ_j i.e., $g_j \sim \mathcal{GP}(\mu_j(.), \kappa_j(.,.))$. For notational convenience, and $w.l.o.g$, we assume the prior mean $\mu_j = 0$. Further, partitions j_1, j_2 that belong to same class i have same covariance kernel i.e., $\kappa_{j_1} = \kappa_{j_2} = \kappa_i$. Let $\mathcal{C}_i$ denote all the partitions with class label i, i.e., $\mathcal{C}_i = \cup_{\{j:\mathbb{C}(j)=i\}} \mathcal{X}_j$, this can be visualized with the help of an example as shown in figure 1. The function f is formally given as follows:

$$f(\mathbf{x}) = \sum_{j=1}^{p} \mathbb{1}\{\mathbf{x} \in \mathcal{X}_j\} g_j(\mathbf{x}) \tag{1}$$

Note: For consistency we denote partitions with a subscript j and classes with subscript i, owing to this notation any variable with subscript i or j would refer to class or partition variable respectively.

2.1 Observation Model

Evaluating the function at any point $\mathbf{x}$ in the input space reveals the following information: function evaluation y, the class label z of the partition to which the point belongs and the tuple distance w = (distance from closest boundary, feature index). We denote that training data set $\mathcal{D} = \{\mathbf{x}_n, y_n, z_n, w_n\}_{n=1}^{N}$ where N is number of training data points.

Also, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T, \mathbf{y}, \mathbf{z}$ are the vector of corresponding evaluations, classes respectively and $\mathbf{W}$ is a list of tuples of distance and feature index along which the distance is measured.

3 Background

This section gives a brief overview of Gaussian Process modeling and the classification tree algorithm used in the Class-GP framework.

3.1 Gaussian Process Modeling

Gaussian process (GP) modeling is a popular statistical framework to model non-linear black box functions f due to its analytical tractability of posteriors. With in this framework the function, $f : \mathcal{X} \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$, being modeled is assumed the to be a distributed as per a Gaussian process prior, formally written as follows:

$$f \sim \mathcal{GP}(\mu(.), \kappa(.,.)),$$

GP is completely given by its mean $\mu(.)$ and covariance kernel $\kappa(.,.)$, where for convenience, and without loss of generality, the mean function $\mu(.)$ is set to 0. The choice of the covariance kernel depends on the degree of smoothness warranted by the function being modeled and is defaulted to stationary kernels such as squared exponential (SE) kernel or Matérn kernel. Functions modeled within this framework are typically assumed to be *stationary* i.e., function can be modeled using a same stationary covariance function over the entire input space.

Learning a GP model involves computing the posterior conditioned on the observed data and learning kernel hyperparameters. Let $\mathcal{D}_n : \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$ be the sampled training data of the objective function f. The posterior of the function f conditioned on the training data $\mathcal{D}_n$ is given by a Gaussian distribution i.e., $f(\mathbf{x})|\mathcal{D}_n \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x}))$, where the mean $\mu_n(\mathbf{x})$ and covariance $\sigma_n^2(\mathbf{x})$ are given as follows:

$$\mu_n(\mathbf{x}) = k^T K^{-1} \mathbf{y} \quad \text{and} \quad \sigma_n^2(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}) - k^T K^{-1} k \tag{2}$$

Here, $\mathbf{y}$ is the vector of noise free observations, k is a vector with $k_p = \kappa(\mathbf{x}, \mathbf{x}_p)$. The matrix K is such that $K_{p,q} = \kappa(\mathbf{x}_p, \mathbf{x}_q)$ $p, q \in \{1, \ldots, n\}$.

The hyperparameters of the model are learnt by maximising the log marginal likelihood which is given as follows:

$$\log p(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^T K^{-1} \mathbf{y} - \frac{1}{2} \log |K| - \frac{n}{2} \log 2\pi \tag{3}$$

and $\theta* = \arg\max_\theta \log p(\mathbf{y}|\mathbf{X}, \theta)$, this optimization problem is solved using off the shelf non convex optimization packages such as Dividing Rectangles (DIRECT)[9], LBGFS[12], CMA-ES[7]. For a detailed treatment of Gaussian process modeling we refer readers to [20] and [22]

3.2 Classification Tree Algorithm

A classification tree / decision tree classifier is a binary tree algorithm which yield axis aligned partitions of the input space by recursively partitioning the input space on one dimension (feature) at a time. The tree is learnt from the training data and the predictions for any input $\mathbf{x}$ is given by traversing the learnt tree from root node to a leaf node. Notice that each leaf node corresponds to a partition of the input space. We use CART algorithm to grow/learn the tree. During training at each the goal is to select the best feature and splitting threshold that minimizes the weighted impurity metric of the children nodes. Most of the tree based algorithms typically use Gini index as the impurity metric to grow the tree, which is given as follows:

$$\textit{Gini index} = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{4}$$

where p_i is the probability of a given class at a given node. The recursion is continued until one of the stopping criterion's is met or no further improvement in the impurity index is achievable. Typical choice of stopping criterion's include maximum depth of the tree, minimum number of samples in a leaf, maximum number of leaf / partitions. For more detailed overview on classification tree please refer the work by [1] and [13].

4 Class-GP Framework

In this section, we introduce a framework to model the family of heterogeneous functions as defined in section 2. Within this framework we solve two sub-problems: 1. Learning partitions of the input space using closest boundary information $\mathbf{W}$ along with class information $\mathbf{z}$ and, 2. Training a Gaussian Process within each partition such that GP's of the partitions that share same class label learn the same set of hyperparameters. Current framework considers both noise less and noisy function evaluations. Further, this framework can also be extended to other partitioning methods that can use closest boundary information.

4.1 Learning Partitions

To learn the partitions of the input space we use decision tree algorithm tailored for the current framework. The algorithm learns the tree is 2 steps: While in both the steps we use recursive binary splitting to grow the tree, in the first step we use closest boundary information to find the best feature index and splitting threshold that maximize reduction of Gini index (or any other impurity metric) until all the closest boundary information $\mathbf{W}$ is exhausted, in the second step the best feature index and splitting threshold that maximize reduction of Impurity metric (Gini index) are selected from available training data at each node as in the CART algorithm 3.2. The nodes are recursively split until a stopping criterion is satisfied. In our proposed framework we default to Gini index as impurity metric and, minimum number of samples at the leaf node and max depth are used as stopping criterion's.

4.2 Gaussian Process in each partition

The partitions of the input space learnt from the decision tree algorithm 4.1 is used to divide the training data set $\mathcal{D}$ into partition based subsets $\mathcal{D}_j$ with n_j data points for all $j \in [p]$. For each partition $\mathcal{X}_j$ underlying stationary function g_j is modeled using a zero mean Gaussian Process prior with continuous stationary kernel $\kappa_j(.,.)$ and subset of training data $\mathcal{D}_j$ is used to learn/train the model. The function modeling and training in each partition is similar to that of a standard Gaussian process regression problem with one exception of learning the hyperparameters of the partition GPs. The posterior of partition GP conditioned on the partition training data is given by $y(\mathbf{x})|\mathbf{x}, j, \mathcal{D}_j \sim \mathcal{N}(\mu_{j,n_j}(\mathbf{x}), \sigma^2_{j,n_j}(\mathbf{x}))$ where mean $\mu_{j,n_j} = \mathbb{E}(y|\mathbf{x}, j, \mathcal{D}_j)$ and variance σ^2_{j,n_j} are given as follows:

$$\mu_{j,n_j}(\mathbf{x}) = \mathbf{k}_j^T K_j^{-1} \mathbf{y}_j \quad \text{and} \quad \sigma^2_{j,n_j}(\mathbf{x}) = \kappa_j(\mathbf{x}, \mathbf{x}) - \mathbf{k}_j^T K_j^{-1} \mathbf{k}_j$$

where $\mathbf{y}_j$ is the vector of observations in given partition j, $\mathbf{k}_j$ is the vector with $k_j^{(s)} = \kappa_j(\mathbf{x}, \mathbf{x}_s)$. The matrix K_j is such that $K_j^{(s,t)} = \kappa_j(\mathbf{x}_s, \mathbf{x}_t)$ where $s, t \in \{1, \ldots, n_j\}$. Note the superscripts here represent the components of the vector and matrix.

Learning hyperparameters in a standard GP typically involves finding the the set of hyperparameters that maximize the log marginal likelihood of the observations 3, where as in the current framework we are required to find the set of hyperparameters that maximizes the log marginal likelihood across all the partition within a class. We propose a novel method to learn of the hyperparameters. A new class likelihood is formed by summing the log marginal likelihoods of all partition GPs for a given class and class kernel hyperparameters are learnt by maximizing new likelihood. The formulation of the class-likelihood function assumes that the data from different partitions are independent of each other and this reduces the computational complexity of learning the hyperparameters significantly while still taking data from all the partitions in the class.

The extensive empirical results show that this new class likelihood reduces the modelling error and provides with the better estimates of the hyperparameters, intuitively this makes sense as we have more data points to estimate hyperparameters even though all the data points do not belong to the same partition. The new class likelihood function for a given class i is given as follows:

$$\mathcal{L}_i(\mathbf{y}_i|\mathbf{X}_i, \theta_i) = \sum_{\{j:\mathbb{C}(j)=i\}} \log p(\mathbf{y}_j|\mathbf{X}_j, \theta_i) = -\frac{1}{2}\mathbf{y}_i^T K_i^{-1} \mathbf{y}_i - \frac{1}{2}\log|K_i| - \frac{n_i}{2}\log 2\pi$$

where $\theta_i^* = \arg\max_{\theta_i} \mathcal{L}_i(\mathbf{y}_i|\mathbf{X}_i, \theta_i)$, K_i is the block diagonal matrix with blocks of K_j's for all $\{j : \mathbb{C}(j) = i\}$, $n_i = \sum_{\{j:\mathbb{C}(j)=i\}} n_j$, and $\mathbf{y}_i$'s is the vector formed by $\mathbf{y}_j$ for all $\{j : \mathbb{C}(j) = i\}$.

5 Class-GP Analysis

In this section, we provide formal statement for probabilistic uniform error bounds for Class GP framework, the results are the extension of the results from [11]. To state our theorem we first introduce the required assumptions on the unknown function $f = \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\} g_j$ over the input space with p partitions.

A0: *g_j's in each partition are continuous with Lipschitz constant L_{g_j}.*

A1: *g_j's in each partition are sampled from a zero mean Gaussian process with known continuous kernel function κ_j* on the compact set *$\mathcal{X}_j$.*

A2: *Kernels κ_j's are Lipschitz continuous with Lipschitz constant L_{κ_j}*

Theorem 1. *Consider an unknown function $f : \mathcal{X} \to \mathbb{R}$ which induces p partitions on the input space, and is given as $f = \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\} g_j$ obeying* **A0:**,**A1:** *and* **A2:**. *Given $n_j \in \mathbb{N}$ noisy observations $\mathbf{y}_j$ with i.i.d zero mean Gaussian noise in a given partition $j \in [p]$ the posterior mean (μ_{n_j}) and standard deviation (σ_{n_j}) of the Gaussian Process conditioned on $\mathcal{D}_j = \{\mathbf{X}_j, \mathbf{y}_j\}$ of the partition are continuous with Lipschitz constant $L_{\mu_{n_j}}$ and modulus of continuity $\omega_{\sigma_{n_j}}$ on $\mathcal{X}_j$ such that*

$$L_{\mu_{n_j}} \leq L_{\kappa_j} \sqrt{n_j} \|\widehat{\mathbf{K}}_j^{-1} \mathbf{y}_j\| \tag{5}$$

$$\omega_{\sigma_{n_j}}(r) \leq \sqrt{2 r L_{\kappa_j} \left(1 + n_j \|\widehat{\mathbf{K}}_j^{-1}\| \max_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}_j} \kappa_j(\mathbf{x}, \mathbf{x}')\right)} \tag{6}$$

where $\widehat{\mathbf{K}}_j = (\mathbf{K}_j + \eta^2 \mathbf{I}_{n_j})$.
Moreover, pick $\delta_j \in (0, 1), r \in \mathbb{R}_+$ and set

$$\beta_j(r) = 2 \log \left(\frac{M(r, \mathcal{X}_j)}{\delta_j}\right) \tag{7}$$

$$\gamma_j(r) = (L_{\mu_{n_j}} + L_{g_j}) r + \sqrt{\beta(r)} \omega_{\sigma_{n_j}} \tag{8}$$

then the following bound on each partition holds with probability $1 - \delta_j$

$$\left|g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x})\right| \leq \sqrt{\beta_j(r)} \sigma_{n_j}(\mathbf{x}) + \gamma_j(r), \forall\, \mathbf{x} \in \mathcal{X}_j \tag{9}$$

and the following bound on the entire input space holds with probability $1 - \delta$ where $\delta = \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\} \delta_j$ i.e.,

$$|f(\mathbf{x}) - \mu_n(\mathbf{x})| \leq \sqrt{\beta(r)} \sigma_n(\mathbf{x}) + \gamma(r), \forall\, \mathbf{x} \in \mathcal{X} \tag{10}$$

Corollary 1. *Given problem setup defined in the theorem 1 the following bound on L_1 norm holds with probability $1 - \delta$*

$$\|f - \mu_n\|_1 \leq \zeta r^d \sum_{j=1}^{p} M(r, \mathcal{X}_j) \left(\sqrt{\beta_j(r)} \sigma_{n_j}(\mathbf{x}) + \gamma_j(r)\right) \tag{11}$$

where ζ is a non negative constant, $\delta = \sum_{j=1}^{1} \delta_j$ and $\delta_j = 1 - M(r, \mathcal{X}_j) e^{-\beta_j(r)/2}$.

6 Numerical Results

In this section, we compare the performance of Class-GP framework with other baselines Partition-GP and Standard-GP over the extensive empirical evaluations on both noisy and noiseless simulated dataset. Performance of the models is evaluated using the mean square error (MSE) metric. Brief overview of the baseline models is given below:
Standard GP: In this framework, we use a single GP with continuous stationary kernel across the entire space to model the underlying function.
Partition GP: - This framework is similar to that of Treed Gaussian process framework [6] with additional class information. We learn the partitions of the input space using the class information followed by modeling the function in each partition individually, i.e., the hyperparameters in each partition are learnt independently of other partition data of the same class.

6.1 Synthetic Data and Experimental Setup

Synthetic data for all the experiments is uniformly sampled from input space $\mathcal{X} = [-10, 10]^d$ where, d is the dimension of the input space. Partitions p are generated by equally dividing the input space and each partition $j \in [p]$ is assigned a class label $i \in [m]$ to forms a checkered pattern. We use Gini index as the node splitting criterion to learn the tree and, squared exponential (SE) covariance kernel for all GPs to model the underlying function in each learnt partition. Each model is evaluated and compared across different functions as given below:

1. Harmonic function:

$$f(\mathbf{x}) = \sum_{j=1}^{p} \mathbb{1}\{\mathbf{x} \in \mathcal{X}_j\}(\cos \mathbf{x}^T \omega_{\mathbb{C}(j)} + b_j)$$

2. Modified Levy function (only for d = 2):

$$f(\mathbf{x}) = \sum_{j=1}^{p} \mathbb{1}\{\mathbf{x} \in \mathcal{X}_j\} \left(\sin^2(\pi v_1) + \sum_{k=1}^{d-1} (v_k - 1)^2 \left(1 + 10 \sin^2(\pi v_k + 1)\right) \right.$$
$$\left. + (v_d - 1)^2 (1 + \sin^2(2\pi v_d)) + b_j \right), \quad \text{where} \quad v_k = \left(1 + \frac{x_k - 1}{4}\right) \omega_{\mathbb{C}(j),k}$$

3. Modified Qing function (only for d = 2):

$$f(\mathbf{x}) = \sum_{j=1}^{p} \mathbb{1}\{\mathbf{x} \in \mathcal{X}_j\} \left(\sum_{k=1}^{d} \left((x_k \omega_{\mathbb{C}(j),k})^2 - i \right)^2 + b_j \right)$$

4. Modified Rosenbrock function (only for d = 2):

$$f(\mathbf{x}) = \sum_{j=1}^{p} \mathbb{1}\{\mathbf{x} \in \mathcal{X}_j\} \left(\sum_{k=1}^{d-1} [100 \left(v_{k+1} - v_k^2 \right)^2 + (1 - v_k)^2] + b_j \right)$$

here $v_k = x_k \omega_{\mathbb{C}(j),k}$.

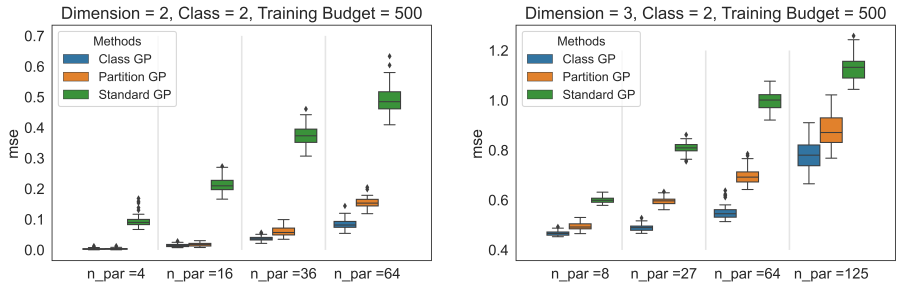
where the frequency vector $\omega_i = i * \omega$ depends on the class i, $\omega_{\mathbb{C}(j),k}$ is the k^{th} component. Further, vector ω is sampled from a normal distribution, constant b_j (intercepts) depends on the partition j and each b_j is sampled from a normal distribution.

Following parameters are initialized for each simulation: dimension d, training budget N, number of partitions p, number of classes m, frequency vector ω, Constant (intercept) vector $\mathbf{b}$, maximum depth of the tree, minimum samples per node, initial kernel hyperparameters θ. For a fixed set of initialized parameters, 50 independent simulation runs for each baseline are performed to compute a confidence interval over the model performance, training data is re-sampled for each run.

To analyze the effects and compare the model performance with baselines each parameter i.e., number of partitions (p), number of classes (m), training budget (N) and, dimension (d), is varied while keeping the others constant. Various initialization of the parameters for the simulations are shown in the table 1. The performance measure metric (MSE) for each model across all the simulations is evaluated on uniformly sampled test data set of fixed size i.e., 5000 data points.

Table 1. Parameter initialization across simulations

Parameters	**Values**
Number of Classes (c)	2, 4, 6, 8
Number of Partitions (p)	4, 16, 36, 64
Training Budget (N)	50, 500, 1000
Dimension (d)	2, 3



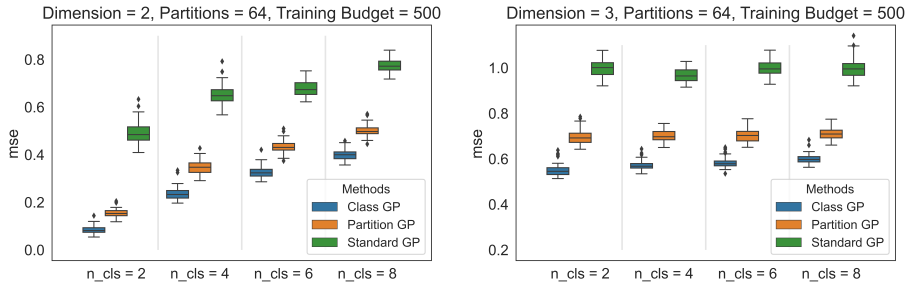
(a) Effect of varying partitions for noiseless 2D harmonic function evaluation.

(b) Effect of varying partitions for noiseless 3D harmonic function evaluation.

Fig. 2. Effect of the number of partitions.

Effects of each parameter on the model performance is analyzed below:

1. Effect of number of partitions (p): For a fixed set of initial parameters as the number of partitions (p) is increased the performance of all the models deteriorates as seen



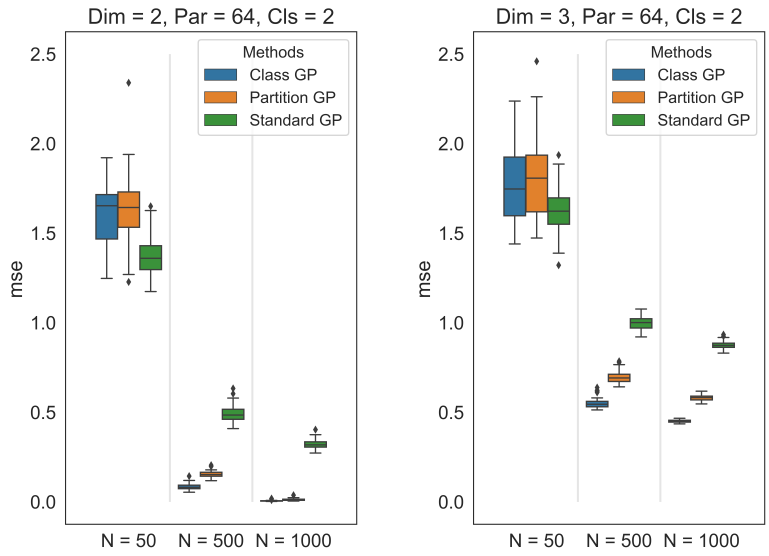
(a) Effect of varying classes for noiseless 2D harmonic function evaluation.

(b) Effect of varying classes for noiseless 3D harmonic function evaluation.

Fig. 3. Effect of number of classes

in Fig. 2(a) and 2(b). Notice that, the performance of Class-GP is superior or at least as good as Partition-GP owing to the fact that, while keeping the training budget constant and increasing the number of partitions leads to decrease in number of points per partition available to learn the hyperparameters of each GP independently in each partition for Partition-GP, whereas, since the number of classes (m) remain constant, number of points available to learn hyperparameters of GPs remain same for Class-GP because of the new likelihood function. The only information we lose on is the correlation information which leads to the small deterioration in the model performance. Also, when the training budget (N) is small or close to number of partitions (p), Standard-GP outperforms both class-GP and Partition-GP. This due to the insufficient data to learn all the partitions of the input space leading to sharp rise in MSE of Class-GP and Partition-GP.

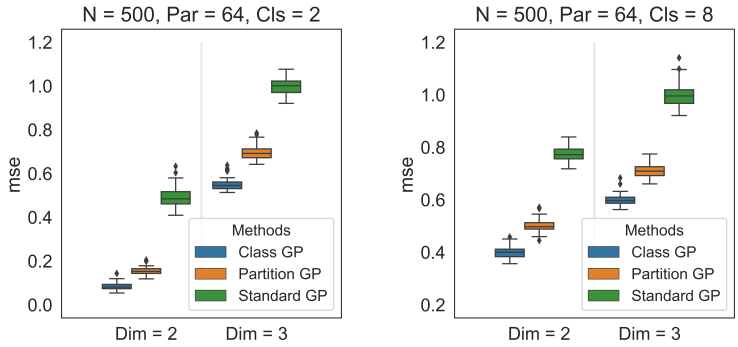
2. Effect of number of classes (m): Increasing the number of classes (m) while keeping other parameters fixed does not effect the performance of the models when the training budget (N) is significantly high because of the large training data available to learn the underlying model, whereas when the training budget is moderate the reduction in the performance of all the models is observed, as seen in the Fig.3(a) and 3(b), with the increasing classes, while keeping to the trend of performance between modeling methods. This is observed due to following reasons: For Class-GP with the increasing classes number of data points to learn the hyperparameters decreases resulting in reduction in the performance, where as for Partition-BO even though the number of data points per partition remain same we observe reduction in performance due to the fact that modeling of high frequency functions (which increase as the number of classes increase) require larger data points and, whereas for the Standard-GP the reduction in performance is because more functions are being modeled with a single GP.
3. Effect of Training Budget (N): Increasing the training budget N has an obvious effect of improvement in the performance of models as seen in Fig. 4(a) and 4(b) owing to the fact that GP's learns the model better with more training data points, but the drawback of increasing the data points is the computational complexity of



(a) Effect of varying training budget for noiseless 2D harmonic function.

(b) Effect of varying training budget for noiseless 3D harmonic function.

Fig. 4. Effect of training budget



(a) Effect of varying dimension for noiseless harmonic function with 2 classes.

(b) Effect of varying dimension for noiseless harmonic function with 2 classes.

Fig. 5. Effect of dimension

the model increases. Also, Notice that the gain in performance of Standard-GP is not as significant as Class-GP or Partition-GP because single GP is used to model the heterogeneous function.

4. Effect of Dimension (d): An increase in the problem's dimensionality increases the number of data points required to model the underlying function. This is also observed in the performance of the models as shown in Fig.5(a), and 5(b) i.e., with the increase in the number of dimensions, model performance decreases, while the other parameters are fixed.

We also evaluate the model's performance over noisy data sets for various parameter initialization, but due to the constraint of space, we only display a subset of the results in the tabular format. Table 2 shows each model's average MSE (not normalized) over 50 independent runs. It can be observed that when the number of partitions is low, the performance of Class GP is as good as Partition GP, whereas when the partitions increase, Class GP outperforms other methods. The full code used to perform simulations can be found at following Github repository.

Table 2. Model performance comparison in presence of noise

Parameters				**Average MSE over 50 runs**		
Functions	Training Budget	Classes	Partitions	Class GP	Patition GP	Standard GP
Harmonic	500	6	4	1.35	1.37	1.51
			64	1.68	1.9	1.85
		8	4	1.35	1.37	1.53
			64	1.7	1.95	1.95
Levy	500	6	4	2.97 e2	2.97 e2	5.72 e2
			64	1.15 e3	1.44 e3	1.56 e3
		8	4	2.97 e2	2.97 e2	5.83 e2
			64	3.36 e3	3.38 e3	4.02 e3
Qing	500	6	4	1.58 e3	1.58 e3	8.45 e7
			64	1.79 e9	2.74 e9	9.45 e10
		8	4	1.58 e3	1.58 e3	8.45 e7
			64	3.25 e10	5.24 e10	8.87 e11
Rosenbrock	500	6	4	1.51 e5	1.52 e5	1.85 e10
			64	9.28 e11	9.42 e11	2.24 e13
		8	4	1.51 e5	1.52 e5	1.85 e10
			64	6.13 e12	1.45 e13	1.97 e14

7 Conclusion and Future Work

This paper presents a novel tree based Class GP framework which extends the existing partition based methods to a family of heterogeneous functions with access to class information. We introduced a new likelihood function to exploit the homogeneous behaviour of the partitions that belong to same class, leading to enhanced the performance of GP models allowing to learn the hyperparameters across the entire class instead of individual partitions. Furthermore, we establish a tailored tree algorithm suitable for current framework that uses the closest boundary information to learn the initial tree.

We also provide some theoretical results in terms of the probabilistic uniform error bounds and bounds on L_1 norm. Finally, we conclude with extensive empirical analysis and clearly show the superior performance of Class GP as compared other baselines. Extension of the Class GP modeling framework to optimization, scaling to higher dimensions [14] and extensive theoretical analysis of the algorithm with practical error bounds are some promising venue to be explored in the future work.

Acknowledgments. This work is supported in part by National Science Foundation (NSF) under the awards 2200161, 2048223, 2003111, 2046588, 2134256, 1815361, 2031799, 2205080, 1901243, 1540040, 2003111, 2048223, by DARPA ARCOS program under contract FA8750-20-C-0507, Lockheed Martin funded contract FA8750-22-9-0001, and the SenSIP Center.

8 Appendix

Proof sketch for the theorem 1 follows along the lines of the proof of Theorem 3.1 in [11]. We get probabilistic uniform error bounds for GPs in each partitions $j \in [p]$ from [11] and we use per partition based bounds to bound the over all function and to derive bound on L_1 norm. The proof for the theorem and corollary given as follows:

Proof. 1

Following bounds on each partition holds with probability $1 - \delta_j$

$$\left|g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x})\right| \leq \sqrt{\beta_j(r)}\sigma_{n_j}(\mathbf{x}) + \gamma_j(r), \forall\, \mathbf{x} \in \mathcal{X}_j \tag{12}$$

where $\beta_j(r)$ and $\gamma_j(r)$ are given as follows

$$\beta_j(r) = 2\log\left(\frac{M(r, \mathcal{X}_j)}{\delta_j}\right) \tag{13}$$

$$\gamma_j(r) = (L_{\mu_{n_j}} + L_{g_j})r + \sqrt{\beta(r)}\omega_{\sigma_{n_j}} \tag{14}$$

Now to bound the entire function lets look at the difference $|f(\mathbf{x}) - \mu_n(\mathbf{x})|$.

$$|f(\mathbf{x}) - \mu_n(\mathbf{x})| = \left|\sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}(g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x}))\right| \tag{15}$$

$$= \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}\left|g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x}))\right| \tag{16}$$

$$\leq \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}\left(\sqrt{\beta_j(r)}\sigma_{n_j}(\mathbf{x}) + \gamma_j(r)\right), \forall\, \mathbf{x} \in \mathcal{X}_j \tag{17}$$

The last inequality (17) follows from (12) and holds with probability $1 - \delta$, where $\delta = \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}\delta_j$.

Now, redefining $\sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}\left(\sqrt{\beta_j(r)}\sigma_{n_j}(\mathbf{x})\right) = \sqrt{\beta(r)}\sigma_n(\mathbf{x})$ and $\sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}\gamma_j(r) = \gamma_(r)$, we have the result. □

The proof for the corollary 1 uses the high confidence bound 10 and is given as follows:

Proof. We know that L_1 norm is given by

$$\|f(\mathbf{x}) - \mu_n(\mathbf{x})\|_1 = \mathrm{E}[|f(\mathbf{x}) - \mu_n(\mathbf{x})|] \tag{18}$$

$$= \int |f(\mathbf{x}) - \mu_n(\mathbf{x})| \, d\mu \tag{19}$$

$$= \int \left| \sum_{j=1}^{p} \mathbb{1}\{x \in \mathcal{X}_j\}(g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x})) \right| d\mu \tag{20}$$

$$= \sum_{j=1}^{p} \int \mathbb{1}\{x \in \mathcal{X}_j\} \left|(g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x}))\right| d\mu \tag{21}$$

$$= \sum_{j=1}^{p} \int_{\mathcal{X}_j} \left|(g_j(\mathbf{x}) - \mu_{n_j}(\mathbf{x}))\right| d\mu \tag{22}$$

$$\leq \zeta r^d \sum_{j=1}^{p} M(r, \mathcal{X}_j) \left(\sqrt{\beta_j(r)} \sigma_{n_j}(\mathbf{x}) + \gamma_j(r) \right) \quad \text{holds w.p } 1 - \delta \tag{23}$$

where $\delta = \sum_{j=1}^{p} \delta_j$ and $\delta_j = 1 - M(r, \mathcal{X}_j) \exp(-\beta_j(r)/2)$. □

References

1. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Routledge (2017)
2. Candelieri, A., Pedrielli, G.: Treed-gaussian processes with support vector machines as nodes for nonstationary bayesian optimization. In: 2021 Winter Simulation Conference (WSC). pp. 1–12. IEEE (2021)
3. Davis, C.B., Hans, C.M., Santner, T.J.: Prediction of non-stationary response functions using a bayesian composite gaussian process. Computational Statistics & Data Analysis **154**, 107083 (2021)
4. Fuentes, M., Smith, R.L.: A new class of nonstationary spatial models. Tech. rep., North Carolina State University. Dept. of Statistics (2001)
5. Gibbs, M.N.: Bayesian Gaussian processes for regression and classification. Ph.D. thesis, Citeseer (1998)
6. Gramacy, R.B., Lee, H.K.H.: Bayesian treed gaussian process models with an application to computer modeling. Journal of the American Statistical Association **103**(483), 1119–1130 (2008)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**(2), 159–195 (2001)
8. Heinonen, M., Mannerström, H., Rousu, J., Kaski, S., Lähdesmäki, H.: Non-stationary gaussian process regression with hamiltonian monte carlo. In: Gretton, A., Robert, C.C. (eds.) Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 51, pp. 732–740. PMLR, Cadiz, Spain (09–11 May 2016)

9. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the lipschitz constant. Journal of optimization Theory and Applications **79**(1), 157–181 (1993)
10. Kim, H.M., Mallick, B.K., Holmes, C.C.: Analyzing nonstationary spatial data using piecewise gaussian processes. Journal of the American Statistical Association **100**(470), 653–668 (2005)
11. Lederer, A., Umlauft, J., Hirche, S.: Uniform error bounds for gaussian process regression with application to safe control. Advances in Neural Information Processing Systems **32** (2019)
12. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical programming **45**(1), 503–528 (1989)
13. Loh, W.Y.: Classification and regression trees. Wiley interdisciplinary reviews: data mining and knowledge discovery **1**(1), 14–23 (2011)
14. Malu, M., Dasarathy, G., Spanias, A.: Bayesian optimization in high-dimensional spaces: A brief survey. In: 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA). pp. 1–8. IEEE (2021)
15. Marmin, S., Ginsbourger, D., Baccou, J., Liandrat, J.: Warped gaussian processes and derivative-based sequential designs for functions with heterogeneous variations. SIAM/ASA Journal on Uncertainty Quantification **6**(3), 991–1018 (2018)
16. Mathesen, L., Yaghoubi, S., Pedrielli, G., Fainekos, G.: Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart. In: 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). pp. 991–997. IEEE (2019)
17. Paciorek, C.J., Schervish, M.J.: Spatial modelling using a new class of nonstationary covariance functions. Environmetrics: The official journal of the International Environmetrics Society **17**(5), 483–506 (2006)
18. Paciorek, C.J.: Nonstationary Gaussian processes for regression and spatial modelling. Ph.D. thesis, Carnegie Mellon University (2003)
19. Pope, C.A., Gosling, J.P., Barber, S., Johnson, J.S., Yamaguchi, T., Feingold, G., Blackwell, P.G.: Gaussian process modeling of heterogeneity and discontinuities using voronoi tessellations. Technometrics **63**(1), 53–63 (2021)
20. Rasmussen, C.E.: Gaussian processes in machine learning. In: Summer school on machine learning. pp. 63–71. Springer (2003)
21. Schmidt, A.M., O'Hagan, A.: Bayesian inference for non-stationary spatial covariance structure via spatial deformations. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **65**(3), 743–758 (2003)
22. Schulz, E., Speekenbrink, M., Krause, A.: A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. Journal of Mathematical Psychology **85**, 1–16 (2018)