https://doi.org/10.15388/vu.thesis.618
https://orcid.org/0000-0001-6530-0176

VILNIUS UNIVERSITY

Saulius Tautvaišas

Scalable Bayesian Global Optimization of Black-Box Functions

DOCTORAL DISSERTATION

Natural Sciences,
Informatics (N 009)

VILNIUS 2024

This dissertation was written between 2019 and 2023 at Vilnius University.

Academic supervisor:
Prof. Dr. Julius Žilinskas (Vilnius University, Natural Sciences, Informatics – N 009).

Defence Panel:
Chair – Prof. Habil. Dr. Gintautas Dzemyda (Vilnius University, Natural Sciences, Informatics – N 009).

Members:
Prof. Dr. Audrius Kabašinskas (Kaunas University of Technology, Natural Sciences, Informatics – N 009).
Assoc. Prof. Dr. Algirdas Lančinskas (Vilnius University, Natural Sciences, Informatics – N 009).
Prof. Dr. Dmitrij Šešok (Vilnius Gediminas Technical University, Natural Sciences, Informatics – N 009).
Prof. Habil. Dr. Anatoly Zhigljavsky (Cardiff University, Natural Sciences, Mathematics – N 001).

The dissertation shall be defended at a public meeting of the Dissertation Defence Panel at 12 p.m. on 26-th June 2024 in Room 203 at the Institute of Data Science and Digital Technologies of Vilnius University. Address: Akademijos g. 4, LT-04812, Vilnius, Lithuania.

The text of this dissertation can be accessed at the Library of Vilnius University and on the website of Vilnius University:
www.vu.lt/lt/naujienos/ivykiu-kalendorius.

https://doi.org/10.15388/vu.thesis.618
https://orcid.org/0000-0001-6530-0176

VILNIAUS UNIVERSITETAS

Saulius Tautvaišas

Bajeso metodai didelio masto juodosios dėžės globaliajam optimizavimui

DAKTARO DISERTACIJA

Gamtos mokslai,
Informatika (N 009)

VILNIUS 2024

Disertacija rengta 2019 – 2023 metais Vilniaus universitete.

Mokslinis vadovas:
prof. dr. Julius Žilinskas (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

Gynimo taryba:
Pirmininkas – prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

Nariai:
prof. dr. Audrius Kabašinskas (Kauno technologijos universitetas, gamtos mokslai, informatika -– N 009).
doc. dr. Algirdas Lančinskas (Vilniaus universitetas, gamtos mokslai, informatika — N 009).
prof. dr. Dmitrij Šešok (Vilniaus Gedimino technikos universitetas, gamtos mokslai, informatika — N 009).
prof. habil. dr. Anatoly Zhigljavsky (Kardifo universitetas, Jungtinė Karalystė, gamtos mokslai, matematika -– N 001).

Disertacija ginama viešame Gynimo tarybos posėdyje 2024 m. birželio 26 d. 12 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų institute Vilniuje, Akademijos g. 4, 203 auditorijoje.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir Vilniaus universiteto interneto svetainėje adresu:
www.vu.lt/lt/naujienos/ivykiu-kalendorius.

Table of Contents

Notation

f	An objective function.
x^*	Global maximizer of the optimization function.
$\mathcal{X}$	Optimization search space.
$\mathcal{D}_n$	Training dataset with n observations.
$\alpha\left(\cdot\vert\mathcal{D}_n\right)$	Acquisition function conditioned on the dataset.
$\boldsymbol{y}$	Vector of output labels, $\boldsymbol{y} = \{y_i\}_{i=1}^n$.
$f(x_i)$	Function value for the i-th input.
$\boldsymbol{f}$	Vector of function evaluations, $\boldsymbol{f} = [f_1, \ldots, f_n]^T$.
Cholesky(A)	Cholesky decomposition: L is a lower triangular matrix such that $\boldsymbol{L}\boldsymbol{L}^T = A$.
$k(x, x')$	The covariance (or kernel) function evaluated at x and x'.
$\mathbb{R}^D$	D-dimensional Euclidean space.
$k(x, x')$	Covariance function of the GP prior distribution.
$\boldsymbol{K}_{nn}$	An $n \times n$ covariance matrix, $[\boldsymbol{K}_{nn}]_{ij} = k(x_i, x_j)$.
$\mathbb{R}^n$	n-dimensional Euclidean space.
σ_ϵ^2	Noise variance.
$\mathcal{M}_i$	i-th GP expert model.
$\mathcal{D}_i$	Relevant dataset for the i-th GP expert.
$p_i(y \mid \mathcal{D}_i, x_*)$	Predictive distribution of i-th GP expert at test point x_*.
ϵ_i	The noise for observation y_i at point x_i ($1 \leq i \leq n$).
x, x_*	Training and test data points.
$\mu_{y_i}(x_*)$	Predictive mean of the i-th GP expert at test point x_*.
$\sigma_{y_i}^2(x_*)$	Predictive variance of the i-th GP expert at test point x_*.
y_*	Noisy test data.

$\sigma_i^2(x_*)$	Posterior predictive variance for the i-th expert at test point x_*.
$\sigma_{f_i}^2(x_*)$	Posterior predictive variance at the noise-free test point x_* for the i-th GP expert.
$\sigma_{\epsilon,i}^2$	Noise variance for the i-th expert.
$\sigma_{f_i^{**}}^2$	Variance of the prior distribution for the i-th GP expert.
$\mu_{f_{\mathcal{A}*}}, \sigma_{f_{\mathcal{A}*}}^2$	Aggregated GP predictive mean and variance for noise-free observation.
$\mu_{\mathcal{A}*}, \sigma_{\mathcal{A}*}^2$	Aggregated GP predictive mean and variance for noisy observation.

Summary

Bayesian optimization (BO) has recently become a popular approach for the global optimization of black-box functions. It has been demonstrated to outperform other state-of-the-art black-box optimization methods when function evaluations are expensive or if the number of allowed function evaluations is low. Many real-life optimization problems require a large number of observations in order to find the global optimum. However, existing BO approaches do not support a large number of observations or they require specialized hardware, which limits the usability of these methods for a regular user.

In this thesis, we propose to replace the standard Gaussian Process (GP) model in BO with the generalized product of experts (gPoE) model. Our proposed gPoEBO algorithm addresses the shortcomings of existing BO approaches based on standard GP model, allowing to scale BO to large-scale optimization problems on regular consumer hardware. We empirically show the efficiency and scalability of the gPoE-based BO on standard global optimization benchmark functions and real life problems. Additionally, we have shown that optimization accuracy can be improved by combining the gPoE model with search space reduction methods. The proposed gPoETRBO algorithm, which combines the trust region (TR) method with gPoE achieves the best performance compared to other GP experts based BO models and matches the perfomance of other state-of-the-art BO models with a significant speedup in computational time and using only moderate computing hardware.

Moreover, we show the benefits of using the gPoE model based BO for capturing the changing noise levels in the objective function. Many real-world optimization problems exhibit input-dependent (heterosce-

dastic) noise levels, which poses a challenge for the standard BO due to a homoscedastic noise assumtion. We proposed two heteroscedastic gPoE-based BO (GPOEBO) algorithms for the global optimization of functions with heteroscedastic noise. We modified two existing heteroscedastic acquisition functions to use individual noise levels from the GPOE model and penalize input space regions with high noise. Experiments on six global optimization functions and two real-world scientific datasets show that our proposed algorithms achieve the best results compared to other BO algorithms and are more robust to the magnitude of heteroscedastic noise.

Santrauka

Dėl efektyvaus geriausio optimizavimo uždavinio sprendinio radimo per mažiausią bandymų skaičių pastaruoju metu sparčiai pasaulyje išpopuliarėjo Bajeso optimizavimo (BO) algoritmai. Šie algoritmai ypač veiksmingi, kai optimizavimo tikslo funkcijos analizinė išraiška nėra žinoma (tokios funkcijos kitaip dar vadinamos juodosios dėžės funkcijomis), o funkcijos maksimumo (minimumo) taško radimas trunka ilgai ar brangiai kainuoja. Didėjant duomenų kiekiui, BO algoritmas tampa mažiau efektyvus ir reikalauja daug skaičiavimo išteklių. Šio algoritmo efektyvumo mažėjimą dažnai lemia Gauso proceso, kuris yra esminė BO algoritmo dalis, apskaičiavimo laiko sudėtingumas. Dėl šio trūkumo šiame darbe siūlome keisti standartinį Gauso proceso modelį į apibendrintą Gauso procesų ekspertų (gPoE) modelį. Šis modelis leidžia dalyti mokymo duomenų aibę į poaibius ir apmokyti skirtingus Gauso procesų modelius ant duomenų aibės poaibių, kurie dar vadinami ekspertų modeliais. Galiausiai kiekvieno eksperto modelio rezultatai agreguojami į galutinį modelį. Tai leidžia sumažinti BO algoritmo skaičiavimo laiko sudėtingumą, nes kiekvienas Gauso proceso ekspertas apmokomas su mažesniu duomenų kiekiu. Be to, ekspertų modelius galima apmokyti sykiu naudojant lygiagrečiuosius skaičiavimus. Gauti rezultatai parodė, kad šis modelis leidžia reikšmingai sumažinti standartinio BO algoritmo vykdymo laiką, neprarandant optimizavimo tikslumo.

Taip pat šiame darbe siūlome dar vieną Bajeso optimizavimo algoritmo modifikaciją – gPoETRBO optimizavimo algoritmą, kuris sujungia Gauso proceso ekspertų (gPoE) modelį su patikimos srities metodu (TR), leidžiančiu sumažinti paieškos erdvę. Atlikti tyrimai parodė, kad naudojant gPoETRBO algoritmą galima pasiekti tokį pat optimizavi-

mo tikslumą, kokį pasiekia esami pažangiausi BO algoritmai, tačiau per daug trumpesnį vykdymo laiką. Be to, papildomi eksperimentai parodė, kad gPoETRBO optimizavimo tikslumas yra geriausias, kai optimizavimo vykdymo laikas yra apribotas tam tikru laiko intervalu.

Papildomai šiame darbe parodėme, kad Gauso proceso ekspertų modeliu paremtas Bajeso optimizavimo algoritmas gali būti sėkmingai pritaikoma uždaviniams su kintamu triukšmo lygiu, priklausomu nuo taško padėties reikšmių srityje. Siūlome dvi įverčio funkcijas, kurios sujungia kiekvieno Gauso proceso eksperto modelio išmoktą triukšmo lygį į apibendrintą triukšmo lygio funkciją. Tai leidžia sumažinti sprendinių naudingumą reikšmių srityje, kurioje triukšmo lygis yra didelis. Gauti rezultatai parodė, kad mūsų pasiūlytos modifikacijos pasiekia geresnį optimizavimo funkcijų tikslumą už kitus Bajeso optimizavimo algoritmus.

List of Figures

List of Tables

List of Algorithms

Chapter 1

INTRODUCTION

1.1 Research Context and Motivation

Global optimization is concerned with the computation and characterization of global minima (or maxima) of nonlinear functions. Global optimization problems are widespread in the mathematical modelling of real-world systems for a very broad range of applications [27]. Bayesian optimization (BO) has become a popular approach for the global optimization of black-box functions [8, 18, 45]. It has been demonstrated to outperform other state-of-the-art black-box optimization methods when function evaluations are expensive or the number of allowed function evaluations is low [78]. The main efficiency of BO originates from the surrogate model which is used to approximate the original black-box function using the available observations. The most commonly used surrogate model is a Gaussian process (GP), which provides a principled and tractable way of modeling uncertainty and allows an informed exploration-exploitation trade-off during optimization.

BO typically works well for low-dimensional problems with a small number of observations. However, as the dimensionality increases, the number of observations required to accurately model the search space grows exponentially due to the curse of dimensionality [6, 47]. Training GP based BO requires an inversion of full covariance matrix. This process has a cubical computational time in the number of observations and becomes the major limiting factor for scaling BO to problems with a large number of observations. As a result, BO is typically limited to only a few thousands of evaluations [79]. However, with an increasing availability of distributed computing resources, a large number of function evaluations becomes possible if the underlying approach allows parallelisation and distributed computations. This motivates scientists to develop the algorithms that could be run in parallel and provide scalable uncertainty estimates to guide the search.

Another limiting factor in Bayesian optimization is the assumption that the noise level remains constant across the entire input space, which is considered homoscedastic, for the standard Gaussian Process. However, this assumption is often too restrictive in real-world applications as the noise levels can be input-dependent (i.e., heteroscedastic). Using the homoscedastic noise assumption in GP when the underlying objective function is corrupted with the heteroscedastic noise can lead to learning a model that will not be able to correctly capture the complexity of the objective function, which presents a challenge for Bayesian optimization.

1.2 Objectives and Tasks of the Thesis

The object of this thesis is Bayesian optimization algorithms. The main goal is to enhance the scalability and efficiency of existing Bayesian optimization algorithms ensuring their applicability for a broad range of optimization problems. The following specific objectives have been established:

1. Propose modifications for existing Bayesian optimization algorithms to improve their scalability and efficiency;
2. Demonstrate the generalizability of the proposed algorithms to problems characterized by heteroscedastic noise levels;
3. Compare the performance of the proposed algorithms to other related optimization methods in terms of efficiency and outcomes.

1.3 Scientific Novelty and Results

Existing BO approaches do not support a large number of observations or they require specialized hardware, which limits the usability of these methods for a regular user. Performing optimization using existing algorithms on regular hardware with only a moderate number of CPU cores increases their reported computational times significantly. In this thesis, we proposed two new algorithms gPoEBO and gPoETRBO based on the generalized product of experts (gPoE) model, which allowed to scale BO to the problems with a large number of observations without the need to have access to specialized hardware for optimization. We experimentally demonstrated the efficiency and scalability of these algorithms compared to the existing algorithms in terms of reduction in runtime. We also theoretically showed that our proposed modification that used search space reduction methods converges to the global maximum of the objective function.

Additionally, we show that our proposed gPoEBO algorithm can be extended to global optimization problems with heteroscedastic noise. We developed two new heteroscedastic gPoE based BO (GPOEBO) algorithms which use a novel combination of gPoE model with a heteroscedastic acquisition function that uses the individual noise levels learned from each GP expert to model the functions with varying noise levels. Our experiments showed the ability of our algorithms to outperform other state-of-the-art heteroscedastic and homoscedastic BO algorithms.

1.4 Statements Defended

The statements defended in this thesis are:

1. The algorithm based on the generalized product of experts model allows scaling Bayesian optimization to problems with a large number of observations without the need to have access to specialized hardware.

2. The gPoETRBO algorithm achieves the best accuracy compared to other expert based optimization algorithms and matches the performance of the state-of-the-art algorithm with a significant speedup in computational time while using only moderate computing hardware.
3. The proposed Bayesian optimization algorithm based on the generalized product of experts model is capable of handling heteroscedastic noise in optimization problems.

1.5 Approbation of the Thesis Results

The results of the dissertation were published in international research journals with a citation index in the Clarivate Analytics Web of Science (CA WoS) database:

1. Tautvaišas, S. and Žilinskas, J., 2024. Scalable Bayesian optimization with generalized product of experts. Journal of Global Optimization, 88(3), pp.777-802.
2. Tautvaišas, S. and Žilinskas, J., 2023. Heteroscedastic Bayesian optimization using generalized product of experts. Journal of Global Optimization, pp.1-21.

The results of this research were presented at the plenary sessions of the following conferences:

1. Tautvaišas S., Žilinskas J. , "Scalable Bayesian Optimization with Generalized Product of Experts", World Congress on Global Optimization 2021 (WCGO 2021), July 7-10, 2021, Athens, Greece.
2. Tautvaišas S., Žilinskas J. "Noisy Global Bayesian Optimization Using Generalized Product of Experts", HUGO 2022 - XV. Workshop on Global Optimization, September 6-8, Szeged, Hungary, 2022.

The results of this research were also presented at the following conference:

1. Tautvaišas S., Žilinskas J. , “Scalable Trust Region Bayesian Optimization with Product of Experts”, 12th International Workshop on Data Analysis Methods for Software Systems (DAMSS), December 2-4, 2021. Druskininkai, Lithuania.

1.6 Structure of the Dissertation

This dissertation is organized into four main chapters, followed by general conclusions and a bibliography. Chapter 1 offers an introduction to the research topic and outlines the structure of the thesis. Chapter 2 presents an overview of global Bayesian optimization with its main components and related global optimization algorithms. Chapter 3 gives the main findings and introduces the proposed algorithms. In Chapter 4, we conduct numerical experiments and offer additional analyses to evaluate the performance and efficacy of the proposed algorithms. Finally, the key findings and insights of the research are summarized in the general conclusion section. This thesis contains 133 pages including the summary in Lithuanian, which starts from page 115. It includes 19 figures, 16 tables and five algorithms.

Chapter 2

A REVIEW OF GLOBAL BAYESIAN OPTIMIZATION

Bayesian optimization (BO) has become a popular approach for global optimization of black-box functions. It has been demonstrated to outperform other state-of-the-art black-box optimization methods when function evaluations are expensive or the number of allowed function evaluations is low. The main efficiency of BO originates from the surrogate model which is used to approximate the original black-box function using the available observations. The most commonly used surrogate model is a Gaussian process (GP), which provides a principled and tractable way of modeling the uncertainty and allows informed exploration-exploitation trade-off during optimization.

2.1 Global Optimization

Global optimization is a branch of applied mathematics and numerical analysis that focuses on finding the global maximizer (minimizer) x^* of an unknown continuous function $f : \mathcal{X} \to \mathbb{R}$ defined on a compact subset $\mathcal{X} \subseteq \mathbb{R}^D$ [44, 52]. It has a wide range of applications in various fields, including engineering, economics, and operations research. Mathematically, the global optimization problem can be formulated as follows:

$$x^* = \arg\max_{x \in \mathcal{X}} f(x) \tag{2.1}$$

The function f is called an objective function and $\mathcal{X}$ is called a feasible set. Alternatively, $\mathcal{X}$ is referred to as the search space or domain [52]. The objective function f is called a black-box function if it does not have a closed-form expression and does not have easily available gradient information. We can only obtain black-box function f values by querying its function values at arbitrary $x \in \mathcal{X}$.

In general, global optimization problems can be challenging to solve. A variety of techniques have been developed to tackle these issues, ranging from deterministic approaches, such as branch-and-bound and interval arithmetic, to stochastic methods, like simulated annealing and genetic algorithms. Bayesian optimization is a particularly promising method for global optimization, as it leverages probabilistic models to guide the search for the global minimum or maximum.

2.2 Bayesian Optimization

Bayesian optimization is a methodology for performing global optimization of black-box functions that are noisy and expensive to evaluate[8, 18, 45, 58]. Given a small number of observed objective function inputs and corresponding outputs, Bayesian optimization iteratively develops a global statistical model of the objective function, which could provide an estimate of uncertainty about the objective function and can be used to balance trade-off between exploration and exploitation. The statistical model consists of a prior distribution that captures our assumptions about the behaviour of unknown objective function and data generation mechanism [60]. During each optimization iteration a posterior distribution is computed by conditioning on the previous evaluations of the objective function. This model is also called a probabilistic surrogate model because it approximates the original objective function and can be queried efficiently at lower computational cost.

In Bayesian optimization (BO) we specify a prior belief over the possible objective function f using the surrogate model and then se-

Algorithm 1 Bayesian Optimization

```
Require: objective function $f$, acquisition function $\alpha$, search space $\mathcal{X}$,
    model $\mathcal{M}$, initial design $\mathcal{D}$
repeat
    Fit the model $\mathcal{M}$ to the data $\mathcal{D}$
    Maximize the acquisition function: $\widehat{x} = \arg\max_{x \in \mathcal{X}} \alpha\left(x, \mathcal{M}\right)$
    Evaluate the function: $\widehat{y} = f\left(\widehat{x}\right)$
    Add the new data to the data set: $\mathcal{D} = \mathcal{D} \cup \left(\widehat{x}, \widehat{y}\right)$
until termination condition is met
Output: the recommendation $x^{*} = \arg\max_{x \in \mathcal{X}} \mathbb{E}_{\mathcal{M}}\left[f(x)\right]$
```

quentially at each iteration n the belief is updated conditioned on the current optimization history $\mathcal{D}_n$ [8]. BO uses an acquisition function $\alpha\left(\cdot|\mathcal{D}_n\right) : \mathcal{X} \rightarrow \mathbb{R}$ to measure how promising is each point in the search space $\mathcal{X}$ if it were to be evaluated next, based on the belief about f given $\mathcal{D}_n$. The main goal is to find the next candidate point x_{n+1} which maximizes the acquisition function given by $\mathrm{x}_{n+1} = \mathrm{argmax}_{\mathrm{x} \in \mathcal{X}} \alpha\left(\mathrm{x}|\mathcal{D}_n\right)$ and use it to evaluate the objective function f. The detailed steps are shown in Algorithm 1 and an illustration of the first three iterations is showed in Figure 2.1.

2.3 Gaussian Process

The Gaussian process (GP) is the most popular surrogate model used in Bayesian optimization for modelling the objective function f [60, 63]. It is defined as a collection of random variables, that any finite number of which has a joint Gaussian distribution[81]. The random variables in GP represent the value of the function $f(x)$ at location x. The illustration of GP function values is shown in Figure 2.2. The figure illustrates Gaussian Process prior functions alongside three slices at varying regions, marked as points $x1$ to $x3$. Scatter plots are shown for function values $f(x1)$ and $f(x2)$, as well as for $f(x2)$ and $f(x3)$. Points $x1$ and $x2$ are closer to each other, and their function values are also closely aligned, indicating a higher degree of correlation between function values, which is evident from the scatter plot. In contrast, point

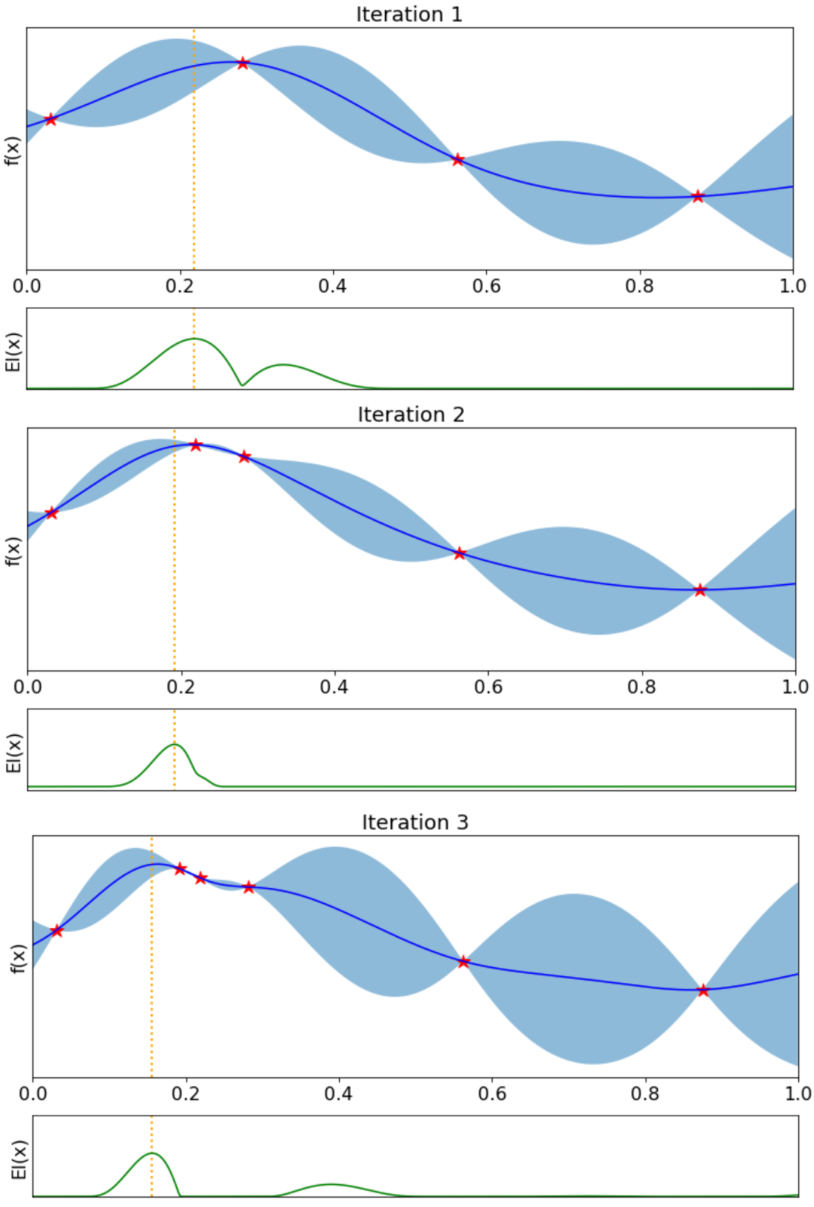


Figure 2.1: Illustration of the Bayesian optimisation algorithm.

$x3$ is further away, and its function values in relation to $x2$ are more dispersed, showing less correlation.

The Gaussian process is fully specified by the mean function and the covariance function (also known as the kernel function). We define the mean function $\mu(x)$ and the covariance function $k(x, x')$ of a real process $f(x)$ as:

$$\mu(x) = \mathbb{E}[f(x)], \tag{2.2}$$

$$k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))], \tag{2.3}$$

and express the Gaussian process as

$$f(x) \sim GP(\mu(x), k(x, x')). \tag{2.4}$$

The main component of GP is the covariance function, or kernel, which quantifies the relationship between data points and shapes the properties of the GP. Covariance functions play a pivotal role in determining the smoothness, continuity, and other characteristics of the modeled function.

Covariance functions, denoted as $k(x, x')$, quantify the correlation between function values $f(x)$ and $f(x')$ at points x and x' in the input space X. They are symmetric and positive definite functions that encode assumptions about how the function values $f(x)$ and $f(x')$ are related to their corresponding input values. The covariance function $k(x, x')$ maps two points $x, x' \in X$ in the input space points to a scalar value representing their covariance, which is then used to create a kernel matrix K with elements $K_{ij} = k(x_i, x_j)$. The choice of covariance function affects the predictions made by GP and determines the model's flexibility and generalization capabilities[81].

2.3.1 Gaussian Process Regression

In this thesis, we focus on supervised learning of regression problem. Given the training data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ with $\boldsymbol{X} = \{x_i\}_{i=1}^{n}, \boldsymbol{y} = \{y_i\}_{i=1}^{n}$, we consider a regression task $y_i = f(x_i) + \epsilon$, where $x \in \mathbb{R}^D$. We assume that the observed values y_i differ from the function values $f(x_i)$ by additive noise ϵ, which we assume follows an independent and identically distributed Gaussian distribution $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ with zero mean and variance σ_ϵ^2. We define $\boldsymbol{f} = [f_1, \ldots, f_n]^T$ as the evaluation of $f(\cdot)$ on the inputs $\boldsymbol{X}$. Then, the probability of observing the targets $\boldsymbol{y}$,

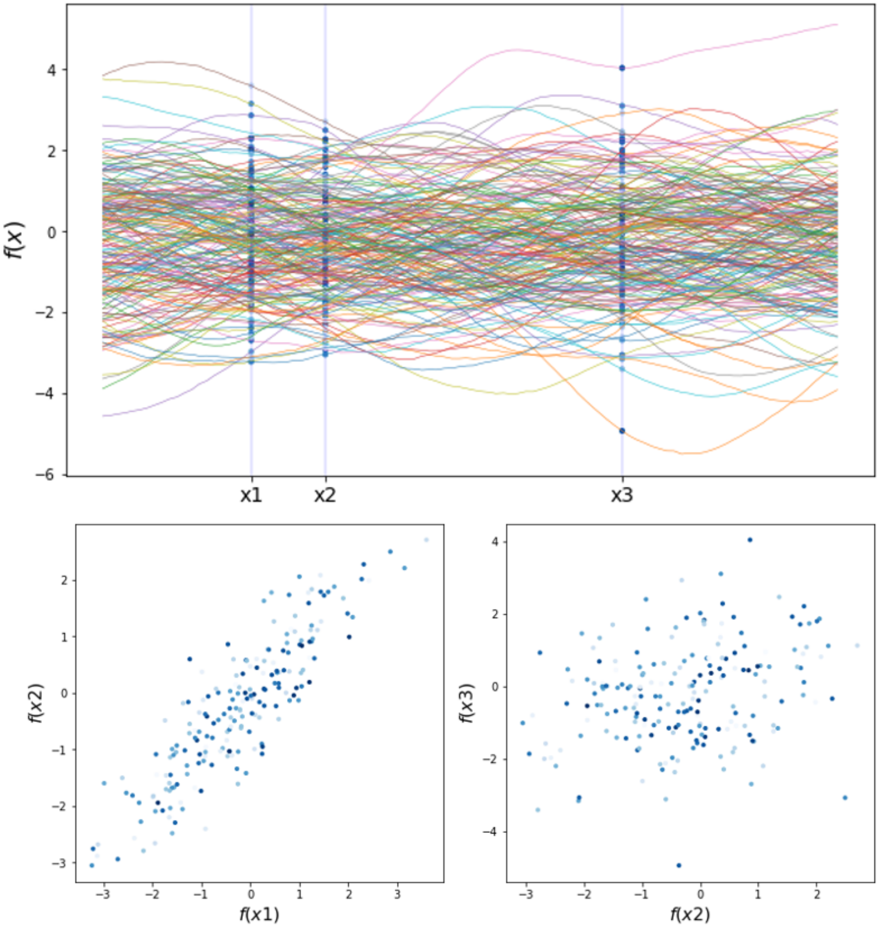


Figure 2.2: Illustration of Gaussian Process functions.

given the function $\boldsymbol{f}$ values is given by the Gaussian likelihood $p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{f}, \sigma_{\epsilon}^{2}\right)$. The objective is to infer the noiseless latent function f from a training data set of n noisy observations.

To achieve this the GP prior distribution is placed over latent function f with a zero mean $m(x) = 0$ and a positive-definite covariance $k(x, x')$ function. A GP prior for the function f values at the input points is given by $p(\boldsymbol{f}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{f}|0, \boldsymbol{K}_{nn})$ with the covariance matrix $[\boldsymbol{K}_{nn}]_{ij} = k(x_i, x_j)$ evaluated at all pairs of input vectors. The covariance matrix $\boldsymbol{K}_{nn}$ constructed from the data inputs captures our assumptions about the smoothness, periodicity, and other properties of the unknown function, before we observe the corresponding target values.

Given the prior distribution $p(\boldsymbol{f}|\boldsymbol{X})$ and the likelihood $p(\boldsymbol{y}|\boldsymbol{f})$, we can compute the GP posterior distribution $p(\boldsymbol{f}|\boldsymbol{y}, \boldsymbol{X})$ using Bayes' theorem as follows:

$$p(\boldsymbol{f}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{f}) \cdot p(\boldsymbol{f}|\boldsymbol{X})}{p(\boldsymbol{y}|\boldsymbol{X})}. \tag{2.5}$$

The marginal likelihood in the denominator is a measure of the probability of observing the targets $\boldsymbol{y}$ given the inputs $\boldsymbol{X}$, marginalizing over all possible function values $\boldsymbol{f}$. This can be computed using the likelihood $p(\boldsymbol{y}|\boldsymbol{f})$ and prior $p(\boldsymbol{f}|\boldsymbol{X})$ as follows:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{f})\, p(\boldsymbol{f}|\boldsymbol{X})\, d\boldsymbol{f}. \tag{2.6}$$

In the case of GP, due to the conjugacy between the Gaussian process prior $p(\boldsymbol{f}|\boldsymbol{X})$ and the Gaussian likelihood $p(\boldsymbol{y}|\boldsymbol{f})$, this integral is tractable and a closed-form solution can be derived. Thus, the marginal likelihood simplifies to:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{y}|0, \boldsymbol{K}_{\boldsymbol{nn}} + \sigma_\epsilon^2 \boldsymbol{I}_n\right), \tag{2.7}$$

where σ_ϵ^2 is the noise variance and $\boldsymbol{I}_n$ is the identity matrix of size n. This expression states that the marginal likelihood is a multivariate normal distribution with zero mean and a covariance matrix that takes into account both the kernel-induced correlation structure among the inputs and the noise in the data.

In a simple 1-d regression problem, where we map from an input x to an output $f(x)$, we can first consider the *prior* distribution which represents our initial beliefs about the kinds of functions we might observe. This is illustrated in Figure 2.3 where multiple random functions are drawn from a GP prior. Observing certain data points, we can update our beliefs to form a *posterior* distribution, only considering functions that align with these data points. This combination of the prior and the data points leads to the posterior distribution over functions, reducing uncertainty near observed values.

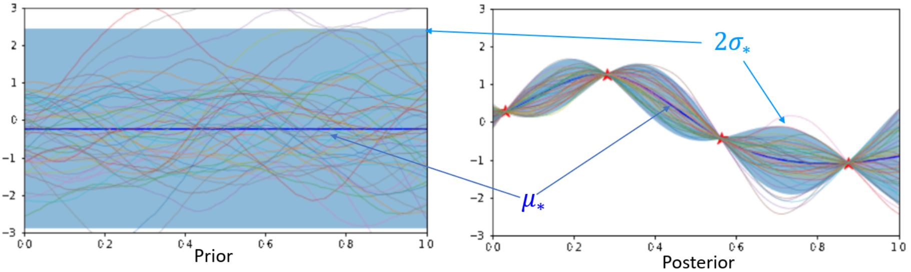


Figure 2.3: Illustration of the Gaussian Process prior and posterior function distributions. **(Left)** The panel shows functions drawn at random from a GP prior distribution. **(Right)** The panel shows the GP posterior after four datapoints have been observed. In both plots, the shaded region represents twice the standard deviation at each input value x.

The covariance function in the GP model is characterized by a set of parameters, denoted as $\boldsymbol{\lambda}$, which directly influence the behaviour of the function we aim to learn, including its smoothness, length scale, and periodicity. In addition to these parameters, the GP model includes a parameter σ_ϵ^2 to capture the variance of the inherent noise in the observations. Together, these parameters $\boldsymbol{\lambda}$ and σ_ϵ^2 form the set of hyperparameters of the GP model, which we denote by $\theta = \{\boldsymbol{\lambda}, \sigma_\epsilon^2\}$. A GP is typically trained by finding the hyperparameters θ that maximize the log-marginal likelihood:

$$\log p\left(\boldsymbol{y}|\,\boldsymbol{X},\ \theta\right) = \ -\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K}_{nn}+\sigma_\epsilon^2\boldsymbol{I})^{-1}\boldsymbol{y}-\frac{1}{2}log\left|\boldsymbol{K}_{nn}+\sigma_\epsilon^2\boldsymbol{I}\right|-\frac{n}{2}log2\pi\ . \tag{2.8}$$

2.3.2 Predictions

Conditioned on the training set $(\boldsymbol{X},\ \boldsymbol{y})$, hyperparameters θ and a test input $x_* \in \mathbb{R}^D$, the GP posterior predictive distribution $p\left(f_*|\ \boldsymbol{X},\ \boldsymbol{y}, \boldsymbol{\theta},\ x_*\right) = \mathcal{N}\left(\mu_*, \sigma_*^2\right)$ is Gaussian with the mean and variance given by

$$\mu_* = \boldsymbol{k}_{*n}{}^T\left(\boldsymbol{K}_{nn}+\sigma_\epsilon^2\boldsymbol{I}\right)^{-1}\boldsymbol{y}, \tag{2.9}$$

$$\sigma_*^2 = \boldsymbol{k}_{**} -\ \boldsymbol{k}_{*n}\left(\boldsymbol{K}_{nn}+\sigma_\epsilon^2\boldsymbol{I}\right)^{-1}\boldsymbol{k}_{*n}{}^T, \tag{2.10}$$

where $\boldsymbol{k}_{*n} = k\left(x_*, \boldsymbol{X}\right)$ and $\boldsymbol{k}_{**} = k\left(x_*, x_*\right)$. The main challenge of GP is that training requires the inversion and the determinant of matrix $\boldsymbol{K}_{nn} + \sigma_\epsilon^2 \boldsymbol{I}$, which is frequently realised via the Cholesky decomposition with the computational cost of $O\left(n^3\right)$. For this reason, training GP on large datasets is computationally intractable.

2.3.3 Learning Gaussian Process Hyperparameters

In many practical applications of the Gaussian Process (GP) regression, specifying all aspects of the kernel function may not be straightforward. While some properties such as stationarity of the covariance function can be determined from the context, it is more challenging to obtain information about other properties, such as the value of free hyperparameters. A mismatch between the hyperparameters and the data can lead to poor performance [6].

For the squared exponential kernel function, hyperparameters play the role of characteristic length-scales, which define how far the length-scale needs to move along a particular axis in input space for the function values to become uncorrelated. If the length-scales in the kernel function are set very large, GP prior may not capture the higher variations in the objective function. Conversely, if the length scales are set too small, GP might fail to generalize. The kernel hyperparameters can be learned from the data by maximizing the marginal likelihood of GP, which is given by:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{n}{2}\log(2\pi), \tag{2.11}$$

where $\mathbf{K}_y = \mathbf{K}_f + \sigma_n^2 \mathbf{I}$ is the covariance matrix for the noisy targets $\mathbf{y}$, and $\mathbf{K}_f$ is the covariance matrix for the noise-free latent function $\boldsymbol{f}$ values. In most cases the complete marginalization over all hyper-parameter variables is analytically intractable in a fully Bayesian approach. Thus, the use of approximation is needed. This is known as type 2 maximum likelihood or evidence approximation [6].

The simplest approach is to make a point estimate of $\boldsymbol{\theta}$ by maximizing the log-marginal likelihood function. The maximization of the log-marginal likelihood can be done using efficient gradient-based optimization algorithms such as conjugate gradients, which will also require to estimate partial derivatives information of the marginal likelihood with respect to the hyperparameters [6].

The complexity of computing the marginal likelihood is dominated by the need to invert the $\mathbf{K}_y^{-1}$ matrix, which requires $O(n^3)$ time for inversion of an $n \times n$ matrix. Once the inverted matrix is known, the computation of the derivatives with respect to hyperparameters requires only time $O(n^2)$ per hyperparameter [81].

2.3.4 Covariance Functions

Covariance functions can be classified into stationary, dot-product or non-stationary functions. The choice of covariance function depends on the specific problem and the properties of the modeled data. The main differences between stationary, dot-product, and non-stationary covariance functions are in how they model the correlation between the function values at input points and the assumptions they make about the underlying function.

2.3.4.1 Stationary Covariance Functions

Stationary kernels are functions whose value depends only on the relative distance between the input points, not their absolute position in the input space. In other words, they are invariant under translation. A stationary kernel can be expressed as:

$$k(x, x') = k(||x - x'||). \tag{2.12}$$

Stationary kernels are often used when the underlying function exhibits a similar behaviour throughout the input space. These kernels assume that the correlation between data points depends only on their distance from each other. The most popular and widely used are the Squared Exponential (SE) kernel and the Matérn kernel functions.

Squared Exponential (SE) Kernel Also known as the Radial Basis Function (RBF) kernel, the SE kernel is one of the most widely used covariance functions. It is a smooth and infinitely differentiable stationary kernel that assumes a higher correlation for points closer in the input space. The SE kernel is defined as:

$$k_{\text{SE}}(x, x') = \sigma^2 \exp\left(-\frac{||x - x'||^2}{2l^2}\right), \tag{2.13}$$

where σ^2 is the amplitude parameter, l is the length scale parameter, and $||x - x'||$ denotes the Euclidean distance between x and x'. SE kernel puts a strong smoothness assumption on the objective function, which is unrealistic for modelling many physical processes. For this reason, it is recommended to use the Matérn 5/2 kernel function which is only two times mean square differentiable.

Matérn Kernel The Matérn kernel is a more general and flexible covariance function, encompassing a family of kernels with varying degrees of smoothness. It is defined as:

$$k_{\text{Matérn}}(x, x') = \sigma^2 \frac{(\sqrt{2\nu}||x - x'||)^{\nu} K_{\nu}(\sqrt{2\nu}||x - x'||)}{2^{\nu-1}\Gamma(\nu)}, \tag{2.14}$$

where σ^2 is the amplitude parameter, ν is the smoothness parameter, l is the length scale parameter, $\Gamma(\cdot)$ is the gamma function, and $K_\nu(\cdot)$ is the modified Bessel function of the second kind of order ν. As ν increases, the Matérn kernel converges to the SE kernel.

2.3.4.2 Dot-Product Covariance Functions

Dot-product kernels depend on the inner product of the input points rather than their distance. These kernels are often used for modeling linear or polynomial relationships between the input variables. A dot-product kernel can be expressed as:

$$k(x, x') = k(\langle x, x' \rangle), \tag{2.15}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Dot-product kernels are less flexible than stationary kernels, but they can model different degrees of polynomial relationships between input variables. The Linear kernel and the Polynomial kernel are examples of dot-product covariance functions.

2.3.4.3 Non-Stationary Covariance Functions

Non-stationary kernels are functions whose value depends on the absolute position of the input points in the input space, and the correlation structure may change across the input space. These kernels are suitable for modeling functions that exhibit different behaviour in different regions of the input space. The Periodic kernel and the Neural Network kernel are examples of non-stationary covariance functions.

2.3.5 Computational Issues

We can see from Section 2.3.2 that the predictive mean of the Gaussian process can be computed as $\mu_* = \boldsymbol{k}_{*n} \left(\boldsymbol{K}_{nn} + \sigma_\epsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}$. Direct inversion of covariance matrix $\boldsymbol{K}_{nn} + \sigma_\epsilon^2 \boldsymbol{I}$ is not recommended, because it can result in numerical instability [47]. The main reason for this is that covariance matrix can be ill-conditioned, i.e., it can have a large condition number, which means that the matrix is close to being singular. Also, due to the limitations of floating-point arithmetic in computers, direct matrix inversion can accumulate errors that make the results unreliable or incorrect.

An alternative to direct inversion of the covariance matrix is the Cholesky decomposition, which can be expressed as $\boldsymbol{K}_{nn} + \sigma_\epsilon^2 \boldsymbol{I} = \boldsymbol{L}\boldsymbol{L}^T$. The Cholesky decomposition takes $O(n^3)$ time to compute, and $O(n^2)$ time is needed to solve for $\boldsymbol{\alpha} = \left(\boldsymbol{K}_{nn} + \sigma_\epsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} = \boldsymbol{L}^{-T}\boldsymbol{L}^{-1}\boldsymbol{y}$. Then, the mean can be computed using ${\boldsymbol{k}_{*n}}^T \boldsymbol{\alpha}$ in $O(N)$ time and the variance can be computed using $\boldsymbol{k}_{**} - {\boldsymbol{k}_{*n}}^T \boldsymbol{L}^{-T}\boldsymbol{L}^{-1}\boldsymbol{k}_{*n}$ in $O(n^2)$ time for each test case.

The pseudo-code presented in Algorithm 2 illustrates the computation of the predictive mean and variance, as well as the log marginal likelihood, using Cholesky decomposition, as described in the book by [81].

Algorithm 2 Gaussian Process regression algorithm

```
$\boldsymbol{L} = \text{cholesky}(\boldsymbol{K}_{nn} + \sigma_\epsilon^2 \boldsymbol{I})$
$\boldsymbol{\alpha} = \boldsymbol{L}^T \setminus (\boldsymbol{L} \setminus \boldsymbol{y})$
$\mu_* = \boldsymbol{k}_{*n}^T \boldsymbol{\alpha}$
$\boldsymbol{v} = \boldsymbol{L} \setminus \boldsymbol{k}_{*n}$
$\sigma_*^2 = k(\boldsymbol{x}_{*n}, \boldsymbol{x}_{*n}) - \boldsymbol{v}^T \boldsymbol{v}$
$\log p(\boldsymbol{y}|\boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{N}{2}\log(2\pi)$
```

To avoid cubic training cost of GP on large datasets several approaches have been proposed. These methods are mostly based on sparse approximation of the covariance matrix using the inducing point methods [53, 62, 71] or training distributed local experts on subsets of training data [12, 16, 25, 38, 72, 73]. Alternative approaches use large-scale computing infrastructure and incomplete Cholesky decompositions [76].

2.3.6 Relationships between GP and Other Models

The Gaussian process model has connections to several other models in machine learning and statistics, which share some similarities in their underlying concepts or mathematical foundations.

Linear Regression The Gaussian process can be viewed as a generalization of linear regression [81]. In linear regression, the goal is to learn a linear function that maps input features to output values. The model is parametric, with the parameters being the weights and biases of the linear function. In contrast, GP is a non-parametric models that defines a prior distribution over functions using a covariance function or kernel. If the kernel chosen for the GP is a linear kernel, the GP reduces to a Bayesian linear regression model [6]. The main difference between the two approaches is that GP provide full Bayesian treatment, which provides a predictive distribution over the output values and allows for the quantification of uncertainties in the predictions.

Kernel Methods Gaussian process is closely related to kernel methods, such as kernel ridge regression (KRR) [59]. Both GP and kernel methods use a kernel function to implicitly map the input data into a high-dimensional feature space, where linear methods can be applied to learn complex, non-linear relationships in the data. The choice of a kernel function plays a crucial role in determining the capacity of both models to capture these relationships.

The primary difference between GP and kernel methods lies in their treatment of uncertainty and their learning objectives. GP are probabilistic models that provide a distribution over function values, whereas kernel methods typically produce point estimates for the predictions. Additionally, GP learns the kernel hyperparameters by maximizing the marginal likelihood of the data, while kernel methods, such as SVM, employ a margin-based learning approach, and KRR relies on regularization [59].

Artificial Neural Networks The Gaussian process and artificial neural networks (ANN) share some connections, particularly in terms of function approximation and the limiting behaviour of infinitely wide neural networks [48, 80]. Both models aim to learn a mapping from input features to output values, with GP using a kernel function to define the prior belief about the function and ANN learning this mapping through a composition of layers, activation functions, and weights.

An interesting connection between GP and NN arises when considering the limit of an infinitely wide neural network. In this case, the distribution of the function values of the neural network converges to the Gaussian process, with a specific covariance function determined by the network architecture and activation functions [48, 80]. This result highlights the potential for GP to be used as a tractable and analytically convenient approximation of large neural networks.

Uncertainty quantification is a key difference between GP and standard NN, with GP providing a full Bayesian treatment and NN producing point estimates for the predictions. Recent developments in Bayesian neural networks aim to incorporate uncertainty estimates into NN by placing priors over the weights and performing Bayesian inference [7]. Scalability is another significant difference, with GP facing computational challenges for large datasets, while NN is more suitable for large-scale problems.

2.4 Acquisition Functions

In Bayesian optimization, an acquisition function plays a crucial role in guiding the search for the global optimum of an unknown objective function. These functions leverage the Gaussian process model's predictive uncertainty to find a balance between exploration and exploitation, resulting in efficient optimization [44]. The choice of acquisition function is nontrivial. Each works well for certain classes of functions, and it is often difficult or impossible to know which will perform best on an unknown function [8].

Probability of Improvement (PI) Introduced by [36], PI is the simplest acquisition function that quantifies the likelihood of an improvement over the current best observed value. Given a Gaussian process model with predictive mean $\mu(x)$ and standard deviation $\sigma(x)$, the PI at a point x is given by:

$$\alpha_{\text{PI}}(x) = \Phi\left(\frac{\mu(x) - f(x_{\text{best}}) - \xi}{\sigma(x)}\right), \tag{2.16}$$

where Φ is the cumulative distribution function of a standard normal distribution, $f(x_{\text{best}})$ is the current best observed function value, and ξ is an optional exploration parameter.

Expected Improvement (EI) Proposed by [45], EI measures the expected improvement over the current best observed value. EI has been widely used due to its desirable properties, such as being differentiable and having a closed-form expression [33, 44, 45]. The EI at a point x is given by:

$$\alpha_{\text{EI}}(x) = \sigma(x)\left[Z\Phi(Z) + \phi(Z)\right], \tag{2.17}$$

where $Z = \frac{\mu(x) - f(x_{\text{best}}) - \xi}{\sigma(x)}$, Φ is the cumulative distribution function of a standard normal distribution and ϕ is the probability density function of a standard normal distribution.

Upper Confidence Bound (UCB) UCB was introduced by [66] and is based on the optimism in the face of the uncertainty principle. It sets an upper bound on the true function value by taking into account the predictive mean and standard deviation. The UCB at a point x is given by:

$$\alpha_{\text{UCB}}(x) = \mu(x) + \kappa\sigma(x), \tag{2.18}$$

where κ is a tunable exploration parameter that controls the trade-off between exploration and exploitation.

Thompson Sampling (TS) TS is a sampling-based acquisition function that involves drawing a sample from the posterior distribution and selecting the point that maximizes the sampled function [70]. Recently it has attracted renewed interest in multi-armed bandits problems. In the bandit setting this strategy samples a reward function from the posterior and selects the arm with the highest simulated reward, while in GP context, this strategy corresponds to sampling the objective function from the GP posterior and then finding the maximum of that sample. TS can be formulated as an acquisition function:

$$\alpha_{TS}\left(x,\, \mathcal{D}_n\right) = f^{(n)}(x) \tag{2.19}$$

$$f^{(n)}(x) \;\sim\; GP(\mu, \kappa|\, \mathcal{D}_n), \tag{2.20}$$

Empirical evaluations show good performance which, however, seems to deteriorate in high dimensional problems, likely due to aggressive exploration [60].

Entropy search (ES) The goal of the ES acquisition function is to reduce the uncertainty in the location x^* by selecting the point which is expected to cause the largest reduction in entropy of the distribution $p_* \left(x \middle| \mathcal{D}_n\right)$ [22]. The acquisition function for ES can be expressed formally as:

$$\alpha_{ES}(x) = H\left(x^* \middle| \mathcal{D}_n\right) - \mathbb{E}_{y \mid \mathcal{D}_n,\, x}\, H\left(x^* \middle| \mathcal{D}_n \cup \{(x,\, y)\}\right), \tag{2.21}$$

where $H\left(x^* \middle| \mathcal{D}_n\right)$ denotes the differential entropy of the posterior distribution $p_* \left(x \middle| \mathcal{D}_n\right)$ and the expectation is over the distribution of the random variable y. This function is not tractable for continuous search spaces so approximations must be made. Recent work uses a discretization of the search space to obtain a smooth approximation $p_* \left(x \middle| \mathcal{D}_n\right)$ and its expected information gain [60].

Predictive Entropy Search (PES) The strategy of the PES acquisition function is to select the next point from the search space which maximizes the expected reduction in the negative differential entropy of $p_* \left(x \middle| \mathcal{D}_n\right)$

$$\alpha_{PES}(x) = H[p\left(x^* \middle| \mathcal{D}_n\right)] - \mathbb{E}_{y \mid \mathcal{D}_n,\, x}\left[H[p\left(x^* \middle| \mathcal{D}_n \cup (x, y)\right)]\right], \tag{2.22}$$

where $H\left[p(x)\right] = -\int p(x) \log p(x) dx$ represents the differential entropy of its argument and the expectation above is taken with respect to the posterior predictive distribution of y given x [23]. The exact evaluation of this equation is not feasible in practice. However, after making a few simplifying assumptions the expectation can be approximated via Monte Carlo with Thompson samples [18, 60].

2.4.1 Optimizing the Acquisition Function

Optimizing the acquisition function is a challenging task as it often involves optimizing a multimodal and potentially non-differentiable function. There are several optimization methods that can be applied to this problem, depending on the properties of the acquisition function and the complexity of the search space. Some common approaches include:

- **Grid search:** One straightforward approach is to perform a grid search over the input space, evaluating the acquisition function at each point and selecting the point with the highest value. However, this method can be computationally expensive, especially in high-dimensional spaces [60].

- **Random search:** Another simple approach is to use random search, where a set of candidate points are sampled randomly from the input space, and the acquisition function is evaluated at each point. The point with the highest acquisition value is then selected. While random search is less efficient than BO, it can still provide reasonable results, especially when combined with other optimization methods [3].

- **Gradient-based optimization:** If the acquisition function is differentiable, gradient-based optimization methods can be employed to find the maximum of the acquisition function. Techniques such as gradient descent, conjugate gradient, or quasi-Newton methods can be used for this purpose [82]. However, these methods may require multiple evaluations of the acquisition function and its gradients, which can be computationally expensive. Additionally, they are prone to getting trapped in local optima if the acquisition function is multimodal.

- **Global optimization algorithms:** In cases where the acquisition function is multimodal and non-differentiable, global optimization algorithms such as genetic algorithms, particle swarm optimization, or simulated annealing can be employed [27]. These methods are typically more robust to local optima but they can be computationally intensive. Moreover, the DIviding RECTangles (DIRECT) algorithm is another global optimization method suitable for optimizing the acquisition function, particularly when dealing with a black-box objective function with a potentially unknown Lipschitz constant [32].

- **Multi-start local optimization:** A popular approach for optimizing acquisition functions is to use a multi-start local optimization strategy [60]. This method involves initializing multiple local optimizers at different starting points in the input space and running them in parallel or sequentially. The best solution found across all local optimizers is then selected. This approach has been shown to be effective in practice.

When optimizing the acquisition function, it is important to balance the computational complexity of the optimization method with the desired accuracy and robustness. In practice, many Bayesian optimization algorithms employ a combination of techniques, such as using a multi-start local optimization strategy with a random search initialization or employing gradient-based optimization methods with occasional random restarts [60].

2.5 Limitations of Bayesian Optimization

Despite its numerous advantages, Bayesian optimization also has certain limitations that can affect its performance and applicability. Some of the well known limitations of Bayesian optimization are:

1. **High-dimensional spaces:** Bayesian optimization tends to be less efficient in high-dimensional spaces, as the complexity of modeling the objective function and optimizing the acquisition function increases with the number of dimensions [18, 60]. As a result, more iterations are required to find the global optimum, which increases both the time and computational resources. There have been efforts to address this limitation, such as the use of random embeddings [78] or additive models [34], but handling high-dimensional spaces remains a challenging task.
2. **Scalability:** Bayesian optimization can become computationally expensive as the number of observed data points increases. This is primarily due to the need to invert the covariance matrix, which has a computational complexity of $O(n^3)$, where n is the number

of data points [81]. Various approaches have been proposed to alleviate this limitation, such as sparse Gaussian processes [53] and local approximation methods [9], but the scalability issue remains a significant concern in large-scale problems.

3. **Noisy evaluations:** When the objective function evaluations are noisy and noise levels can be input-dependent (i.e., heteroscedastic), Bayesian optimization performance can be adversely affected [37, 51]. Although Gaussian process models can handle noise through the incorporation of a noise term in the covariance function, this can lead to a more complex optimization problem and increased computational cost [63]. Some research has focused on developing robust acquisition functions to handle noisy evaluations, but noisy evaluations continue to be a challenging issue in Bayesian optimization [21, 37, 67].

4. **Discrete or categorical variables:** Bayesian optimization is primarily designed for continuous optimization problems. Handling discrete or categorical variables can be challenging, as Gaussian process models are not well-suited for modeling discrete or categorical input spaces [57, 60]. Various approaches have been proposed to address this issue, such as using a continuous relaxation of the categorical variables [64], employing specialized kernels for mixed continuous-discrete input spaces [67] or using multi-armed bandits to select values for both categorical and continuous inputs [57] but the performance of these methods can be problem dependent.

2.6 Scalable Bayesian Optimization

BO typically works well for low-dimensional problems with a small number of observations. As the dimensionality increases, the number of observations required to generalize well over the whole search space, grows exponentially due to the curse of dimensionality. The training of the GP based BO requires the inversion of a full covariance matrix, which has a cubical computational time in the number of observations.

For this reason, BO is typically limited to only a few thousand of evaluations [79]. Thus, the lack of scalable uncertainty estimates to guide the search is a major limiting factor for large-scale BO. However, with an increasing availability of distributed computing resources, a large number of function evaluations becomes possible if the underlying approach allows parallelisation and distributed computations.

There has been a series of research trying to reduce the number of required observations for high-dimensional BO problems to overcome the curse of dimensionality. Some methods rely on structural assumptions, such as additive structure, where methods try to exploit additive structure in the objective function [19, 34, 79]. Other methods assume low effective dimensionality of the objective function and rely on transforming a high-dimensional space into a low-dimensional subspace [5, 78]. A novel subspace embedding algorithm HeSBO [46] has been proposed that overcomes the limitations of the high-dimensional projections and is based on a sound theoretical framework.

It was observed that in high dimensional problems, where the number of observations is large, BO over-explores the boundary of a search space [49]. To reduce the number of redundant observations, recent works have started to explore space partitioning algorithms and local modeling that uses multiple local models in promising regions, which showed very promising empirical results. The BOCK algorithm [49] applies a cylindrical transformation on Euclidean geometry of the search space to avoid over-exploring the boundary. Ensemble Bayesian optimization (EBO) [79] builds an ensemble of local additive GP models on partitioned search space to scale BO to high-dimensional problems with a large number of observations. A trust region method based BO algorithm TuRBO was proposed [17], which abandons a global GP model and uses a collection of independent local GP models, where each model represents a different trust region. To address computational issues in the standard GP model the authors used the Lanczos process approximation method. A novel meta-algorithm LA-MCTS was proposed [77] that learns how to partition the search space and find the most promising regions to avoid over-exploration. The authors empirically showed that by combining their algorithm with the TuRBO algorithm they were able to improve the state-of-the art results for many high dimensional optimization problems. Recently, the Vecchia approximation of the standard GP was used in the TuRBO algorithm to scale the BO to large number of observation [31]. The results showed that TuRBO with Vecchia approximation compared favorably to the other state-of-the-art algorithms and can be used to speed up BO when many evaluations of the objective function are necessary.

The major problem with the proposed algorithms is that to accelerate the computations they use a specialized hardware, which is expensive or not available for a regular user. To achieve scalability the EBO algorithm uses 240 CPU cores [79], while TuRBO and LA-MCTS use GPU hardware to accelerate the computations. Performing the optimization using these algorithms on regular hardware with only a moderate number of CPU cores significantly increases their computational times.

2.6.1 Gaussian Process Experts

Gaussian Process Expert (GPE) models have been introduced to address GP scalability issues by combining multiple GP models, each focusing on different aspects of the underlying function or different regions of the input space [74]. GPE models offer improved scalability compared to standalone GP models. The computational complexity of training a GP model scales cubically with the number of data points, which can become infeasible for large-scale problems. By dividing the data into smaller subsets and training a GP expert on each subset, GPE models can reduce the overall computational complexity. Additionally, each expert can be trained and updated independently, allowing for parallelization and more effective allocation of computational resources. Furthermore, GPE models can provide more accurate and expressive representations of the target function by combining the strengths of individual GP experts. This can lead to better generalization performance and more robust predictions compared to using a single GP model.

GPE models can be grouped into two main categories: mixture of experts and product of experts. The mixture of experts models (MoE) divides the input space into regions, and each expert is responsible for modeling the target function within its assigned region [30, 74, 84].

This approach can be useful for capturing local patterns in the data and can lead to more accurate predictions. However, mixture of experts models may suffer from overfitting or underfitting if the regions are not appropriately defined, and the combination of experts' predictions can be less straightforward.

Product of experts models (PoE) considers the contribution of all experts for every input point but weighs their predictions based on their uncertainty levels [26]. This approach can help to capture complex dependencies in the data and can be more robust to noise and overfitting. Moreover, the combination of experts' predictions in product of experts models is very simple computation as they only directly take the product of the predictive distributions, allowing for a more coherent integration of the individual experts' contributions.

Despite these useful properties not all PoE models offer consistent predictions which means that aggregated predictive distribution cannot converge to the true underlying predictive distribution when the training size n approaches infinity [38]. Furthermore, different model assumptions limit its efficiency and flexibility.

The best state-of-the-art models like Bayesian committee machine (BCM) [73] is derived from the conditional independence assumption and a common prior $p(f)$. The conditional independence assumption means that different subsets of training data have low correlation, while common prior means that each local expert trained on the subset of data needs to share the same kernel and kernel hyperparameters. Therefore, hyperparameters have to be learned jointly for all expert GP models. Also, common prior requirement limits the expressiveness of the objective function that can be learned using this model. The robust Bayesian committee machine (rBCM) has been shown empirically to outperform BCM [16], but because it inherits the theoretical basis of the BCM it is limited to all the restrictions of BCM [13]. Recently, the generalized robust Bayesian committee machine (grBCM) model [38] was proposed as an extension to the rBCM. This model introduces a global GP expert which communicates with child GP experts leading to consistent and more accurate predictive distribution compared to the BCM and rBCM. Compared to the BCM and rBCM model where aggregation is performed in f-space, this model aggregates predictive distribution in y-space. However, this model still inherits common prior restriction and because it has additional independence assumptions, it has higher computing complexity cost compared to rBCM. Another approach is generalised product of experts (gPoE) proposed by [12, 13]. This approach is strongly motivated by the log opinion pool [24]. This framework provides a sound and flexible theoretical basis for combining GP experts which does not require conditional independence assumption between different subsets of training data and expert GP models do not need to share common prior. For this reason, training data subsets can potentially overlap, and local GP experts can have different kernels [11]. Furthermore, this approach can also potentially model heteroscedasticity and non-stationarity, even though individual expert GPs use relatively simple stationary kernels [13].

2.6.1.1 Training

To train the GP on a large training set the product-of-expert models partitions the data into M subsets $\mathcal{D}^{(i)} = \left\{\boldsymbol{X}^{(i)},\ \boldsymbol{y}^{(i)}\right\}$, where $1 \leq i \leq M$, and train GP on $\mathcal{D}^{(i)}$ as an expert GP model. All M experts share hyperparameters. If we partition the training data into disjoint subsets and ignore the correlation between GP experts, then marginal likelihood can be factorized into

$$p\left(\boldsymbol{y}|\boldsymbol{X},\ \theta\right) \approx \prod_{i=1}^{M} p_i \left(\boldsymbol{y}^{(i)}|\boldsymbol{X}^{(i)}, \theta\right), \tag{2.23}$$

where $p_i\left(\boldsymbol{y}^{(i)}|\boldsymbol{X}^{(i)}, \theta\right) \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{K}_i + \sigma_\epsilon^2 \boldsymbol{I}_i\right)$ with $\boldsymbol{K}_i = k\left(\boldsymbol{X}^{(i)}, \boldsymbol{X}^{(i)}\right) \in \mathbb{R}^{n_i \times n_i}$ and n_i is the size of training data assigned to the *i*-th GP expert model and $n_i \ll n$. The factorization of the log-marginal likelihood degenerates the full covariance matrix $\boldsymbol{K}_{nn} = k(\boldsymbol{X}, \boldsymbol{X})$ into block-diagonal matrix and, thus full inverse covariance matrix can be approximated by $\boldsymbol{K}^{-1} \approx diag\left[\boldsymbol{K}_1^{-1},\ \ldots, \boldsymbol{K}_M^{-1}\right]$. This process is illustrated in Figure 2.4.

For training the model we seek to maximise the log-marginal likeli-

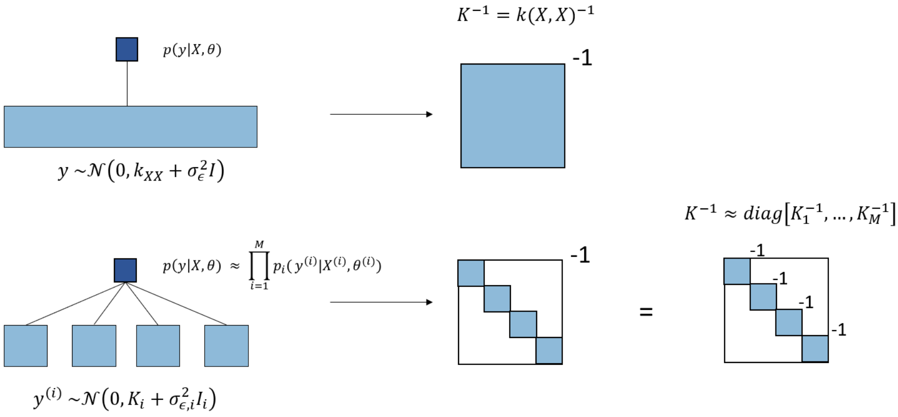

$K^{-1} = k(X,X)^{-1}$
-1
$p(y|X,\theta)$
$y \sim \mathcal{N}(0, k_{XX} + \sigma_\epsilon^2 I)$
$K^{-1} \approx diag[K_1^{-1}, \dots, K_M^{-1}]$
-1
$p(y|X,\theta) \approx \prod_{i=1}^{M} p_i(y^{(i)}|X^{(i)}, \theta^{(i)})$
=
-1
-1
-1
-1
$y^{(i)} \sim \mathcal{N}(0, K_i + \sigma_{\epsilon,i}^2 I_i)$

Figure 2.4: Block-diagonal approximation of full covariance matrix, which allows efficient training and prediction, because we only need to invert a covariance matrix for each expert with only a subset of data.

hood with respect to the (shared) kernel hyperparameters

$$\begin{aligned}\log p_i\left(\boldsymbol{y}^{(i)}|\boldsymbol{X}^{(i)}, \theta\right) = &-\frac{1}{2}\boldsymbol{y}^{(i)}\left(\boldsymbol{K}_i + \sigma_\epsilon^2 \boldsymbol{I}_i\right)^{-1}\boldsymbol{y}^{(i)} \\ &-\frac{1}{2}\log\left|\boldsymbol{K}_i + \sigma_\epsilon^2 \boldsymbol{I}_i\right| + const.\end{aligned} \tag{2.24}$$

Training can be distributed, which reduces the training complexity time to $O(M{n_i}^3)$, where M is the number of experts. If we run the training in parallel with M compute nodes the training time complexity is reduced to $O({n_i}^3)$, which is significantly lower than $O\left(n^3\right)$ thecomplexity of the full GP training.

2.6.1.2 Prediction aggregation

In this section we describe the process of computing the predictive GP distribution and introduce to several of product-of-GP-experts models. We will assume that a set of M GP experts has been trained according to Section 2.6.1.1. We want to predict a function value f_* at a corresponding test input x_*. An important feature of product-of-GP-experts models is that the predictive distribution $p\left(f_*|x_*\right)$ of function values after recombining predictions from trained GP experts is still a Gaussian distribution. Also, the prediction aggregation is performed in function space f, except specified otherwise, which means that we need to map the aggregated predictive GP distribution predictive distribution $p\left(f_*|x_*\right)$ through a likelihood function to predict the y_* labels.

Product of Experts (PoE) The Product of Experts model [26] aggregates predictions of M experts at test point x_* according to

$$p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right) = \prod_{i=1}^{M} p_i\left(f_*|x_*,\ \mathcal{D}^{(i)}\right), \tag{2.25}$$

where the predictive aggregated mean and precision are

$$\mu_{\mathcal{A}} = \sigma_{\mathcal{A}}^2\left(x_*\right)\ \sum_{i=1}^{M} \sigma_i^{-2}\left(x_*\right)\mu_i\left(x_*\right), \tag{2.26}$$

$$\sigma_{\mathcal{A}}^{-2}\left(x_*\right) = \sum_{i=1}^{M} \sigma_i^{-2}\left(x_*\right). \tag{2.27}$$

The experts in this model need to be jointly calibrated by training the entire model to avoid the risk of double counting the shared information [11]. Also, when we increase the number of experts the predictive variance vanishes, which leads to overconfident predictions, especially in regions without data [16].

Bayesian Committee Machine (BCM) Bayesian Committee Machine (BCM) has been introduced in [73] to aggregate predictions of GP experts. BCM makes the conditional independence assumption that $\mathcal{D}^{(i)}\mathcal{D}^{(j)} \mid f_*$ and explicitly incorporates the GP prior $p\left(f_*|x_*\right)$ when combining predictions. BCM posterior predictive distribution

$$p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right) = \frac{\prod_{i=1}^{M} p_i\left(y_*|x_*,\ \mathcal{D}^{(i)}\right)}{p^{M-1}\left(f_*|x_*\right)}, \tag{2.28}$$

here the predictive aggregated mean and precision are

$$\mu_{\mathcal{A}} = \sigma_{\mathcal{A}}^2\left(x_*\right)\ \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma_i^{-2}\left(x_*\right)\mu_i\left(x_*\right), \tag{2.29}$$

$$\sigma_{\mathcal{A}}^{-2}\left(x_*\right) = \sum_{i=1}^{M} \sigma_i^{-2}\left(x_*\right) + \left(1 - M\right)\sigma_{**}^{-2}, \tag{2.30}$$

where σ_{**}^{-2} is the prior precision of $p\left(f_*\right)$.

Because BCM is derived from the conditional independence assumption and a common GP prior, GP on each subset of training data $\mathcal{D}^{(i)}$ need to share the same kernel and kernel hyperparameters [11]. The main disadvantage of this model is that BCM exhibit problematic behaviour in regions transitioning from high to low-density data [16].

Robust Bayesian Committee Machine (rBCM) The rBCM has been introduced by [16] and mitigates some of the issues of the BCM in the case where there are only a few observations and allows for flexible weighting of GP experts, via $\alpha_i\left(x_*\right)$, which controls the contribution of expert i at x_*. The rBCM predictive distribution is

$$p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right) = \frac{\prod_{i=1}^{M} p_i\left(y_*|x_*,\ \mathcal{D}^{(i)}\right)}{p^{-1+\sum_{i=1}^{M}\alpha_i}\left(f_*|x_*\right)}, \tag{2.31}$$

where the predictive mean and precision are given as

$$\mu_{\mathcal{A}} = \sigma_{\mathcal{A}}^{2}\left(x_*\right)\ \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma_i^{-2}\left(x_*\right)\mu_i\left(x_*\right), \tag{2.32}$$

$$\sigma_{\mathcal{A}}^{-2}\left(x_*\right) = \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma_i^{-2}\left(x_*\right) + \left(1 - \sum_{i=1}^{M}\alpha_i\left(x_*\right)\right)\sigma_{**}^{-2}. \tag{2.33}$$

The rBCM has similar limitations of BCM, that subsets of training data for the different GP experts need to be disjoint and experts need to share the same kernel and hyperparameters. Although the rBCM mitigates some problematic issues of the BCM and allows for flexible weighting of GP experts, it still exhibits problematic behaviour in regions with changing data density [16].

Generalised product of experts (gPOE) Generalized Product of Experts (gPoE) has been introduced by [12]. The gPoE is strongly motivated by the log opinion pool framework [24]. The gPoE model combines each individual GP expert prediction into the final aggregate model

$$p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right) = \prod_{i=1}^{M} p_i^{\alpha_i(x_*)}\left(f_*|x_*,\ \mathcal{D}^{(i)}\right), \tag{2.34}$$

which is again Gaussian with mean and covariance given by

$$\mu_{\mathcal{A}} = \sigma_{\mathcal{A}}^2\left(x_*\right)\ \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma_i^{-2}\left(x_*\right)\mu_i\left(x_*\right), \tag{2.35}$$

$$\sigma_{\mathcal{A}}^{-2}\left(x_*\right) = \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma_i^{-2}\left(x_*\right). \tag{2.36}$$

The weight $\alpha_i\left(x_*\right)$ is a measure of reliability and controls the contribution of each expert *i* at test point x_*, where $\alpha_i\left(x_*\right) > 0$ and $\sum_{i=1}^{M} \alpha_i\left(x_*\right) = 1$. The gPoE considers $\alpha_i\left(x_*\right)$ to be proportional to the change between prior $p\left(f_*|\, x_*\right)$ and posterior $p\left(f_*|\, x_*, \mathcal{D}^{(i)}\right)$ entropy of the Gaussian distributions of the *i*-th expert at point x_* [13]. It can be represented as

$$\alpha_i\left(x_*\right) \propto H_i\left(x_*\right) = \frac{1}{2}\left(\log\sigma_{**}^2 - \log\sigma_i^2\left(x_*\right)\right), \tag{2.37}$$

where σ_{**}^2 is the prior and $\sigma_i^2\left(x_*\right)$ is the posterior variance at the test point x_*. When the change in entropy at point x_* is zero, it means the *i*-th expert provides no information about this point that comes from training observation and should not be used in combined predictions. In a case when the point x_* is significantly distant from the regions where the experts were trained, then $\alpha_i\left(x_*\right)$ becomes $\frac{1}{M}\forall i$ and the combined model falls back to the average of the priors of the experts [11].

Prediction aggregation in y-space Compared to other models the gPoE model does not require conditional independence assumption between subsets of training data and GP experts do not need to share a common prior. For this reason, training data subsets can potentially overlap and GP experts can have different kernels and hyperparameters.

As a results, GP experts can be trained independently without the need of joint training for all experts in y-space instead of f-space [11].

Predictive distribution of GP expert i conditioned on the related subset of the data $\mathcal{D}^{(i)}$ and test input $x_* \in \mathbb{R}^D$ in y-space is Gaussian $p_i\left(y_*|\mathcal{D}^{(i)}, x_*\right) \sim \mathcal{N}\left(\mu_i(x_*), \sigma_i^2(x_*)\right)$ with mean and covariance

$$\mu_i\left(x_*\right) = \boldsymbol{k}_{*i}\left(\boldsymbol{K}_i + \sigma_{\epsilon,i}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y_i}, \tag{2.38}$$

$$\sigma_i^2\left(x_*\right) = \boldsymbol{k}_{**} - \ \boldsymbol{k}_{*i}\left(\boldsymbol{K}_i + \sigma_{\epsilon,\ i}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_{*i}{}^T + \sigma_{\epsilon,i}^2, \tag{2.39}$$

where $\sigma_{\epsilon,i}$ is the noise variance of each GP expert. The gPoE model combines individual GP experts predictions into the final aggregate model

$$p_{\mathcal{A}}\left(y_*|x_*, \mathcal{D}\right) = \prod_{i=1}^{M} p_i^{\alpha_i(x_*)}\left(y_*|x_*,\ \mathcal{D}^{(i)}\right), \tag{2.40}$$

with mean and covariance given by (2.35) and (2.36), respectively.

Conservative prediction variance A well-known weakness of the gPoE is that it overestimates the prediction variance, which means that prediction variance becomes too conservative. The proof provided by [38] shows that when the number of data points increases together with an increasing number of experts, the gPoE yields a conservative prediction variance. The prediction variance at the test point x_* is higher than the true prediction variance and equal to the variance from the expert, which is farthest away from the test point and its variance is closest to σ_{**}^2 prior variance. This is consistent with the conservative fusion rule [2], also known as the covariance intersection algorithm, which states that when we combine multiple Gaussian distributions by raising to a power of weights, adding up to 1, we get a conservative distribution which is upper bound of true distribution.

2.6.2 Sparse Gaussian Process

Another approach to scale GP to large datasets is to use Sparse Gaussian Process (SPGP). SPGP provides an efficient approximation of the full Gaussian Process by utilizing a small set of representative points, called pseudo-inputs or inducing points, to capture the underlying structure of the data [62]. The main idea behind this method is to use a small set of representative points, called pseudo-inputs or inducing points, to approximate the full GP.

Given a dataset with N number of input-output pairs $(\mathbf{X}, \mathbf{y})$, the first step in the SPGP method is to select a subset of M inducing points. The sparsity in the model arises because we consider only a pseudo-data set $\overline{\mathcal{D}}$ of size $M < N$ with pseudo-inputs denoted as $\overline{\mathbf{X}} = \{\overline{\mathbf{x}}_m\}_{m=1}^{M}$ and pseudo targets as $\overline{\mathbf{f}} = \{\bar{f}_m\}_{m=1}^{M}$ [62]. Given the pseudo-inputs the target data are assumed to be i.i.d., which leads to the complete data likelihood as follows:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{X}, \overline{\mathbf{X}}, \overline{\mathbf{f}}) &= \prod_{n=1}^{N} p\left(y_n \mid \mathbf{x}_n, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) \\ &= \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{NM}\mathbf{K}_M^{-1}\overline{\mathbf{f}}, \mathbf{\Lambda} + \sigma^2\mathbf{I}\right) \end{aligned} \tag{2.41}$$

where $\mathbf{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda}), \lambda_n = K_{nn} - \mathbf{k}_n^{\top}\mathbf{K}_M^{-1}\mathbf{k}_n$ and $[\mathbf{K}_{NM}]_{nm} = K\left(\mathbf{x}_n, \overline{\mathbf{x}}_m\right)$.

A Gaussian prior is placed on the pseudo-targets $\overline{\mathbf{f}}$, which are the function values at the inducing points $\overline{\mathbf{X}}$:

$$p(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}) = \mathcal{N}\left(\overline{\mathbf{f}} \mid \mathbf{0}, \mathbf{K}_M\right). \tag{2.42}$$

The posterior distribution over the pseudo targets can be obtained using the Bayes rule on (2.41) and (2.42):

$$p(\overline{\mathbf{f}} \mid \mathcal{D}, \overline{\mathbf{X}}) = \mathcal{N}\left(\overline{\mathbf{f}} \mid \mathbf{K}_M\mathbf{Q}_M^{-1}\mathbf{K}_{MN}\left(\mathbf{\Lambda} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}, \mathbf{K}_M\mathbf{Q}_M^{-1}\mathbf{K}_M\right), \tag{2.43}$$

where $\mathbf{Q}_M = \mathbf{K}_M + \mathbf{K}_{MN}\left(\mathbf{\Lambda} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{K}_{NM}$.

The predictive distribution for a new input x_* is then obtained by integrating the likelihood (2.41) with the posterior (2.43):

$$\begin{aligned} p\left(y_* \mid \mathbf{x}_*, \mathcal{D}, \overline{\mathbf{X}}\right) &= \int p\left(y_* \mid \mathbf{x}_*, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) p(\overline{\mathbf{f}} \mid \mathcal{D}, \overline{\mathbf{X}}) \mathrm{d}\overline{\mathbf{f}} \\ &= \mathcal{N}\left(y_* \mid \mu_*, \sigma_*^2\right), \end{aligned} \tag{2.44}$$

where

$$\begin{aligned} \mu_* &= \mathbf{k}_*^\top \mathbf{Q}_M^{-1} \mathbf{K}_{MN} \left(\mathbf{\Lambda} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}, \\ \sigma_*^2 &= K_{**} - \mathbf{k}_*^\top \left(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1}\right) \mathbf{k}_* + \sigma^2. \end{aligned} \tag{2.45}$$

The computational cost is dominated by the matrix multiplication $\mathbf{K}_{MN}\left(\mathbf{\Lambda} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{K}_{NM}$ in the calculation of Q_M which is $\mathcal{O}\left(M^2 N\right)$. By using the pseudo-inputs, the SPGP method reduces the computational complexity of GP from $O(N^3)$ to $O(M^3)$. This allows for the application of GP to much larger datasets while maintaining a reasonable computational cost [53, 62].

The inducing points $\overline{\mathbf{X}}$ can be selected heuristically or optimized jointly with the kernel hyperparameters using gradient-based optimization techniques, such as conjugate gradients or L-BFGS, by maximizing the marginal likelihood of the sparse approximation.

The main limitation of SPGP is that the modeling performance is limited by the small set of global inducing points, which limits its ability to capture the quick-varying features, especially in high dimensions [39]. Despite the fact, that SPGP reduce the computational complexity of full GP from $O(N^3)$ to $O(M^3)$, the complexity still scales cubically with the number of inducing points where as Gaussian Process Expert models can scale linearly with the number of experts or regions, making them more suitable for larger datasets and higher-dimensional problems [16, 54]. Moreover, GPE models can incorporate various types of expert models, such as different kernel functions or even non-Gaussian models, to better fit the underlying data distribution, while the SPGP model can only use the same kernel function for the entire input space, which may not capture the full range of data variability [74].

2.7 Heteroscedastic Bayesian Optimization

In many optimization problems the evaluations of the objective function are only available via noisy observations. For the standard Gaussian Process (GP), the noise level is assumed to be constant across all the input space (i.e., homoscedastic). However, this assumption is too restrictive in real-world applications as the noise levels can be input-dependent (i.e., heteroscedastic). Using the homoscedastic noise assumption in GP when the underlying objective function is corrupted with the heteroscedastic noise can lead to learning a model that will not be able to correctly capture the complexity of the objective function, which presents a challenge for Bayesian optimization (BO).

Several approaches have been proposed to handle a heteroscedastic noise in BO. A treed GP model was proposed by [1] to handle the heteroscedasticity of the objective function. However, the authors used the standard expected improvement acquisition function, which does not take into account the noise level. The work by [10] used heteroscedastic Gaussian process from [42] to learn the noise distribution as a robustness metric, which was used as an additional objective within the multi-objective Bayesian optimization framework to estimate the Pareto front. Another work proposed by [21] used heteroscedastic Gaussian process and heteroscedastic acquisition functions for Bayesian optimization.

2.7.1 Heteroscedastic Gaussian Process

To define the heteroscedastic GP we proceed by placing a GP prior on f and assume that our observations have been generated according to $y_i = f\left(x_i\right) + \epsilon_i$ with independent Gaussian noise terms $\epsilon_i \sim \mathcal{N}\left(0, \sigma_i^2\right)$, where noise variances are given by $\sigma_i^2 = r\left(x_i\right)$. For standard (homoscedastic) GP we assume that noise variance is constant (i.e. $r\left(x_i\right) = \sigma_\epsilon^2$) across all the input space $\boldsymbol{x}$ and for this reason analytical inference is possible. In heteroscedastic setting the noise function $r\left(x_i\right)$ is non-constant function and it was observed that the input-dependent noise variance $\sigma_\epsilon^2\left(x_i\right)$ enables to describe the possible heteroscedasticity in the objective function[40]. An example of heteroscedastic noise distorted sin wave function is depicted in Figure 2.5. The noisy observations y_i are generated using noise free sin wave function $f(x_i) = \sin(x_i) + 0.2(x_i) + 3$ and noise variance function $r(x_i) = 0.5(x_i)$. We can observe that with larger x_i the sample values are spread further away from the noise free function $f(x_i)$.

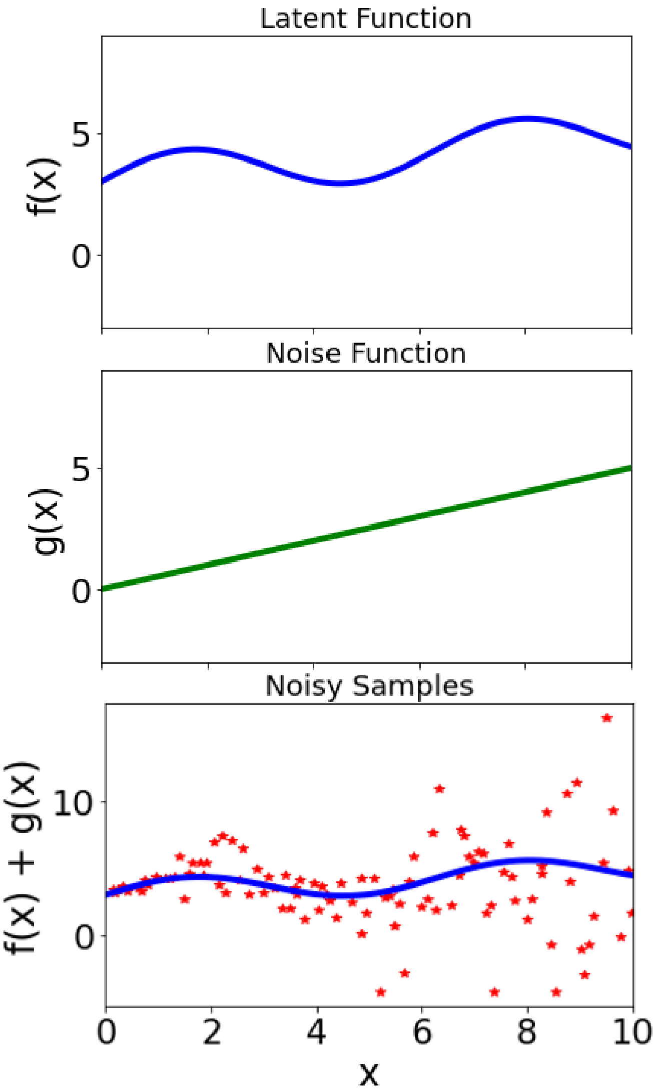


Figure 2.5: Illustrative sin wave function with heteroscedastic noise.

In order to learn the appropriate noise structure from the data and ensure that the model remains non-parametric, many authors does not specify the functional form for noise level, but place a Gaussian prior over it [20, 21, 35]. A secondary GP is placed on the latent function $g(x_i)$, which is used to model noise level directly from the data. To ensure the positivity for the noise function $r(x_i)$ it is parameterized as exponential form $r(x_i) = \exp g(x_i)$. The main problem with modeling heteroscedastic noise using secondary GP is that the marginal likelihood (evidence) $p\left(\boldsymbol{y}|\, \boldsymbol{X}, \theta\right)$ and posterior $p\left(f_*|\, \boldsymbol{X},\ \boldsymbol{y}, \boldsymbol{\theta},\ x_*\right)$ cannot be computed analytically due to the exponential relationship between the noise function and its GP, which breaks the traditional conjugacy property that allows analytical computations in homoscedastic GP models.

To circumvent this problem, the most likely heteroscedastic Gaussian process (MLHGP) was proposed in [35]. The parameters of both GP are learned using a modified version of the expectation–maximization (EM) algorithm. The predictive MLHGP distribution is similar to the homoscedastic GP described in Section 2.3.2, except that the homoscedastic noise in (2.9) is replaced with heteroscedastic noise variance, which was learned using secondary GP. By placing a GP prior on f and taking $r(x)$ as the assumed heteroscedastic noise function, the GP predictive distribution $p\left(y_*|\, x_*\right) = \mathcal{N}\left(\mu_*, \sigma_*^2\right)$ at the test input x_* is Gaussian with the mean and variance given by

$$\mu_* = \boldsymbol{k}_{*n}\left(\boldsymbol{K}_{nn} + \boldsymbol{R}_{nn}\right)^{-1}\boldsymbol{y}, \tag{2.46}$$

$$\sigma_*^2 = \boldsymbol{k}_{**} -\ \boldsymbol{k}_{*n}\left(\boldsymbol{K}_{nn} + \boldsymbol{R}_{nn}\right)^{-1}\boldsymbol{k}_{*n}{}^T + \boldsymbol{r}_{**}, \tag{2.47}$$

where $\boldsymbol{R}_{nn} = diag(\boldsymbol{r})$ with $\boldsymbol{r} = (r(x_1), \ldots, r(x_n))^T$ and $\boldsymbol{r}_{**} = r(x_*)$ represents posterior and predictive heteroscedastic noise variance. The covariance matrices $\mathbf{K}_{nn} = k(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{k}_{*n} = k\left(x_*, \boldsymbol{X}\right)$ and $\boldsymbol{k}_{**} = k\left(x_*, x_*\right)$ are the same as in homoscedastic GP. The main characteristic of this algorithm is that we learn the latent objective function using primary GP and then we use secondary GP to learn the noise variance function. The details of how to train MLHGP can be found in [35].

2.7.2 Heteroscedastic Acquisition Functions

The acquisition function is used to guide the search for finding the maximum of objective by trading off exploration and exploitation function in as few iterations as possible. At each iteration of BO, the acquisition function takes into account the GP predictive mean and variance to model the utility function, which is maximized to determine where to sample next.

Many acquisition functions have been proposed [18, 22, 33, 45], but the expected improvement (EI) acquisition function is the most popular and the most widely used acquisition function. In the noise-free setting, when we observe $f(x)$ without noise, we can find the largest observed objective function value $f^* = \max_{\mathbf{x}_i \in \mathbf{x}_{1:n}} f(\mathbf{x}_i)$, where n is the number of observations. Then the EI acquisition function can be written in a closed form as

$$\begin{aligned} \alpha_{\mathrm{EI}_n}(\mathbf{x}) &= \mathbb{E}[\max(f(\mathbf{x}) - f^*), 0] \\ &= \mathbb{E}[(f(\mathbf{x}) - f^*)^+] \\ &= \sigma_n(\mathbf{x}) \cdot (\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \phi(\gamma(\mathbf{x})), \end{aligned} \tag{2.48}$$

where $\gamma(\mathbf{x}) = \frac{\mu_n(\mathbf{x}) - f^*}{\sigma_n(\mathbf{x})}$, $\phi(\cdot)$ and $\Phi(\cdot)$ are the PDF and CDF of standard normal distribution, respectively. The expected improvement algorithm then evaluates at the point with the largest expected improvement

$$\mathbf{x}_{n+1} = \operatorname{argmax} \alpha_{\mathrm{EI}_n}(\mathbf{x}). \tag{2.49}$$

However, when we deal with noisy observations, we do not observe $f(\mathbf{x}_i)$, but rather $y_i = f\,(\mathbf{x}_i) + \epsilon_i$, where ϵ_i is the observation noise and for homoscedastic GP assumed to have fixed noise variance across all input space: $\epsilon_i \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$. To compute EI becomes challenging with noisy observations because we do not know the exact value of the current best observation f^*. One of the strategies proposed by [50] is to replace the current best observation f^* with the GP posterior mean estimate of the best function value $g^* = \max_{\mathbf{x}} \mu(\mathbf{x})$ referred to as a plug-in value. Using this strategy EI can be computed analytically in a similar way as with noise-free observations. However, one drawback using this approach with noisy observations is that it does not take into account the noise of the future observations.

The augmented expected improvement (AEI) proposed by [28] introduced a multiplicative penalty in order to penalize the points whose GP posterior variance σ_n^2 is small compared to the noise variance σ_ϵ^2, which can be computed as

$$\alpha_{\text{AEI}}(\mathbf{x}) \quad = \quad \alpha_{\text{EI}_n}(\mathbf{x}) \times \left(1 - \frac{\sigma_\epsilon}{\sqrt{\sigma_n^2 + \sigma_\epsilon^2}}\right). \tag{2.50}$$

When the noise level is $\sigma_\epsilon = 0$, AEI reduces to the original EI function. The heteroscedastic augmented expected improvement (HAEI) was proposed by [21], which extends the AEI acquisition function by exchanging the fixed noise level with input-dependent noise level:

$$\alpha_{\text{HAEI}}(\mathbf{x}) \quad = \quad \alpha_{\text{EI}_n}(\mathbf{x}) \times \left(1 - \frac{\sigma_\epsilon(\mathbf{x})}{\sqrt{\sigma_n^2 + \gamma^2\sigma_\epsilon^2(\mathbf{x})}}\right), \tag{2.51}$$

where $\sigma_\epsilon^2(\mathbf{x})$ is the predictive posterior noise variance at the input $\mathbf{x}$ and γ is a positive penalty parameter for regions with high heteroscedastic noise.

2.8 Conclusions

In this chapter, we introduced Bayesian optimization and its main components, including Gaussian process regression, acquisition functions, and the process of learning hyperparameters. We then discussed the primary challenges and limitations facing Bayesian optimization. Next, we presented various approaches for scaling Bayesian optimization to handle problems with a large number of observations. Moreover, we described optimization problems with heteroscedastic noise and why this type of noise presents a challenge for Bayesian optimization. Finally, we presented different approaches for handling heteroscedastic noise in Bayesian optimization, employing heteroscedastic Gaussian processes and heteroscedastic acquisition functions.

Chapter 3

EXTENDING GLOBAL BAYESIAN OPTIMIZATION

Bayesian optimization is challenging for problems with thousands of observations. One of the approaches to scale BO to large-scale optimization problems is to replace the standard Gaussian process with Gaussian process expert models. Product-of-expert belongs to this group of models which are effective with low computation cost and is easy to parallelize. Despite these useful properties not all PoE models offer consistent predictions which means that aggregated predictive distribution cannot converge to the true underlying predictive distribution when the training size approaches infinity [40]. Furthermore, different model assumptions limit its efficiency and flexibility.

In this chapter, we propose to replace the standard GP model in BO with generalized product of experts (gPoE) model in Section 3.1. Additionally, we propose a new algorithm gPoETRBO which combines trust region and gPoE models in Section 3.2. We finally show that generalized product of experts (gPoE) model can be applied to heteroscedastic BO in Section 3.3. The proposed algorithms and main findings were published in [68, 69].

3.1 Bayesian Optimization with Generalized Product of Experts

To scale BO we propose to replace the standard GP model with GP experts model. In this section we focus on the gPoE model, because it provides the most flexible framework and has the most desirable properties compared to other GP experts model [13]. We name our algorithm gPoEBO (generalized PoE based Bayesian Optimization) and present the pseudocode in Algorithm 3. We highlight the key features of our algorithm in this section below.

At each iteration t, we have a dataset $D_t = \{x_k,\ y_k\}_{k=1}^{t}$ from previous evaluations. We randomly partition this dataset into M disjoint subsets, each with t/M data points, and use them to train M local GP experts. Once the set of M GP experts has been trained, we compute their posterior mean and variance on a candidate set from q randomly generated samples using Sobol sequence [65]. We use the gPoE model to aggregate the final GP model by using a differential entropy weighting scheme [13]. Figure 3.1 illustrates the process of creating the final aggregated GP model from local GP experts. To find the next most promising candidate point for optimization problem we use the Upper Confidence Bound (UCB) acquisition function [66]. To optimize UCB we use the mean and variance measures already available from final the GP model. The objective function is evaluated at this candidate point and the new data point is added to the dataset. The iterations continue until the desired number of iterations is reached. The output of the algorithm is the best recommendation obtained through these iterations.

3.2 Trust region Bayesian optimization with Generalized Product of Experts

To improve the accuracy of gPoEBO algorithm, we propose a gPoETRBO algorithm which is inspired by trust region BO algorithms [17, 55]. The main idea of the trust region method is to use an approximate model for the objective function which can be trusted and is significantly easier

Algorithm 3 Generalized PoE based Bayesian Optimization (gPoEBO)

```
Input: Number of initializing points $N$, number of iterations $T$, number of points per expert $n_i$.
Output: The best recommendation $x^*_T$.
Randomly select and evaluate $N$ points in the search space $\mathcal{D}_0 = \{(x_i, f(x_i))\}_{i=1}^N$.
for $t = 1$ to $T$ do
    Randomly partition $\mathcal{D}_{t-1}$ into $M = |\mathcal{D}_{t-1}|/n_i$ subsets.
    Train $M$ local GP experts on $\{\mathcal{D}^i_{t-1}\}_{i=1}^M$ subsets.
    Generate $q$ candidate points $\mathbf{X}^c = \{x^c_1, \dots, x^c_q\}$ from the search space.
    Evaluate $i$ local GP expert posterior mean $\mu^i_t$ and variance $\sigma^i_t$ on $\mathbf{X}^c$ points.
    Aggregate $\mu^{\mathcal{A}}_t$ and $\sigma^{\mathcal{A}}_t$ using (2.35) and (2.36).
    Maximize UCB acquisition function $\hat{x} = \mathrm{argmax}_{x \in \mathbf{X}^c}\ \mu^{\mathcal{A}}_t(x) + \sqrt{\beta}\sigma^{\mathcal{A}}_t(x)$
    Evaluate the objective function $\hat{y} = f(\hat{x})$.
    Add new data point to the dataset $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\hat{x}, \hat{y}\}$
end for
```

to optimize in a neighbourhood of the current guess of the solution of the optimization problem. The neighborhood considered is called the trust region. The trust region is expanded or shrunk depending on the improvement of the objective function. If the comparison is good, we take the new step and increase the trust-region radius. If the comparison is bad, we reject the new step and decrease the trust-region radius [15, 83].

3.2.1 The gPoETRBO Algorithm

We present the pseudocode in Algorithm 4 and an illustrated workflow in Figure 3.2. Our algorithm uses generalized PoE as a surrogate model with UCB acquisition function. We define a trust region to be a rectangle around the current best solution. The size of the trust region is adjusted based on whether we find a better solution in that region. The region is increased if we find a better solution and decreased if we are not able to make progress.

We start optimization by initializing the base length size of the trust

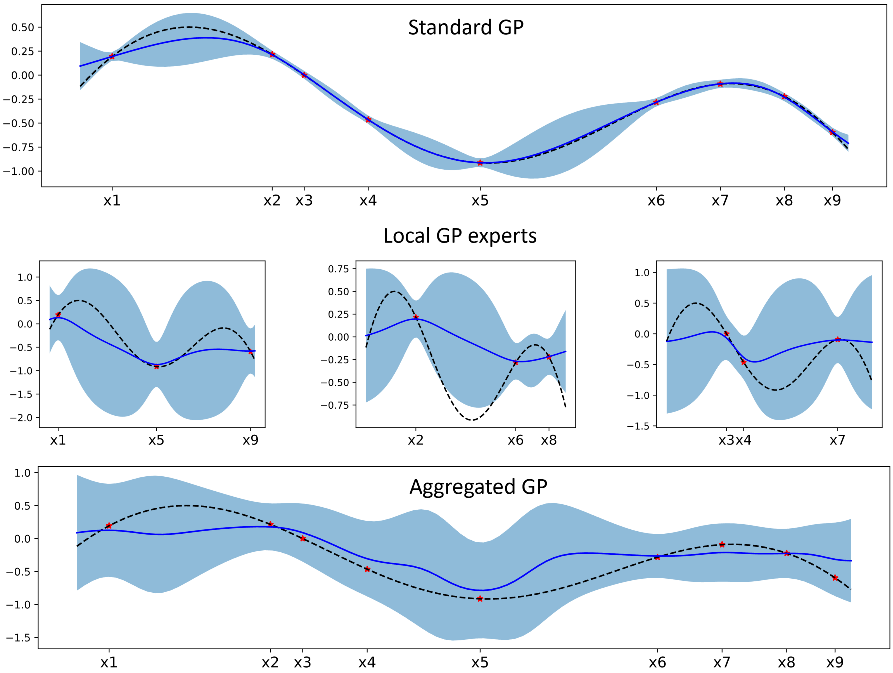


Figure 3.1: Illustration of building local GP experts model. Suppose we have a dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_9, y_9)\}$. In a standard approach we use all available data to build a GP. However, in the gPoE model, we randomly partition the dataset into $M = |\mathcal{D}|/n_i = 3$ subsets, where we choose the expert size of $n_i = 3$ and build M local GP experts $\{\mathcal{M}_i\}_{i=1}^{M}$ independently. We combine experts posterior predictions using (2.40) to get the aggregated GP model.

region $L = L_{init}$ and defining the minimum L_{min} and maximum L_{max} side length. At every iteration, as in the gPoEBO algorithm, we partition the data randomly into M disjoint sets and train the M GP experts. Then, we find the best point in our dataset D_t corresponding to the best objective function value. We draw a trust region rectangle around this point and generate q random points using the Sobol sequence in that region. For each GP expert, we compute posterior mean and variance on generated points and use the gPoE model to compute the final aggregated GP model. To find the next most promising candidate point we optimize the UCB acquisition function. We evaluate the objective function value on that candidate point and compare the objective function value to the current best solution. If we improve the

Algorithm 4 Generalized PoE based Trust Region Bayesian Optimization (gPoETRBO)

```
Input: Number of initializing points $N$, number of iterations $T$, number of points per expert $n_i$, initial TR parameters.
Output: The best recommendation $x_T^*$.
Randomly select and evaluate $N$ points in the search space $\mathcal{D}_0 = \{(x_i, f(x_i))\}_{i=1}^N$.
for $t = 1$ to $T$ do
    Randomly partition $\mathcal{D}_{t-1}$ into $M = |D_{t-1}|/n_i$ subsets.
    Train $M$ local GP experts on $\{\mathcal{D}_{t-1}^i\}_{i=1}^M$ subsets.
    Construct a hyper-rectangle TR of the length $L$ around the best point $x_t^* = \max_{1 \leq i \leq |\mathcal{D}_{t-1}|} f(x_i)$.
    Generate $q$ candidate points $\mathbf{X}^c = \{x_1^c, \ldots, x_q^c\}$ from $TR\,(x_t^*)$.
    Evaluate $i$ local GP expert posterior mean $\mu_t^i$ and variance $\sigma_t^i$ on $\mathbf{X}^c$ points.
    Aggregate $\mu_t^A$ and $\sigma_t^A$ using (2.35) and (2.36).
    Maximize UCB acquisition function $\hat{x} = \text{argmax}_{x \in \mathbf{X}^c} \mu_t^A(x) + \sqrt{\beta}\sigma_t^A(x)$
    Evaluate the objective function $\hat{y} = f(\hat{x})$.
    Add a new data point to the dataset $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\hat{x}, \hat{y}\}$
    Update the TR parameters and check whether to restart.
end for
```

current best solution, we increment the success counter and reset the failure counter to zero, otherwise we set the success counter to zero and increment the failure counter. We use expansion rate $\alpha_e = 2$ to increase the size of the trust region $L = min\,(L_{max}, \alpha_e \times L)$ after τ_{succ} number of successive improvements and shrinking rate of $\alpha_s = \frac{1}{\alpha_e} = \frac{1}{2}$ to reduce the size $L = L \times \alpha_s$ after τ_{fail} consecutive failures. We reset the counters after we change the size of the trust region. Additionally, we do not allow the side length of a trust region to become larger than L_{max}. When the trust region length size becomes less than L_{min}, we discard all the values and restart the optimization. In all experiments, we use the following hyperparameters suggested by [17]: $\tau_{succ} = 3$, $\tau_{fail} = d$, $L_{min} = 2^{-7}, L_{max} = 1.6, L_{init} = 0.8$, where d is the number of dimensions.

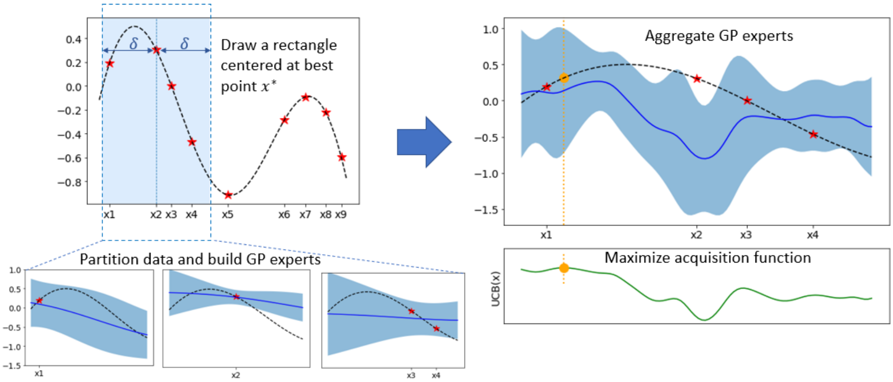


Figure 3.2: Workflow of the gPoETRBO algorithm.

3.2.2 Restart Strategy

The trust region is a local optimization method and is biased toward the starting points. To achieve global optimization, we use a restart strategy. It was shown numerically that restarting optimization from scratch achieves a better performance than allowing the algorithm to continue [17, 55, 75]. In our case, we restart the optimization when the trust region length size falls bellow L_{min} (line 12 in Algorithm 4). Moreover, the proofs of convergence to the global maximum with restart strategy were provided in [56, 61, 75].

We briefly describe the proof provided by [75] for categorical and mixed search space, but in our case we use it to prove that gPoETRBO converges to the global maximum in continuous search space. The proof is based on assumptions that (i) the objective function f is bounded in the search space $\mathcal{X}$ and (ii) the surrogate model accurately approximates f in a small enough region. In BO it is generally assumed that f is Lipschitz continuous and we can choose a small enough trust region with the side length L_{min} so that the GP surrogate model can accurately approximate any data point in that region, these assumptions are satisfied. Given these two assumptions, we can show that if after a restart, gPoETRBO terminates in a finite number of iterations, then it converges to the local maxima of f, or, if it does not terminate after a finite number of iterations, it converges to the global maximum. If gPoETRBO does not terminate after a finite number of iterations there must have been at least one successive improvement per $N_{min} = \tau_{fail} \times \lceil \log_{\alpha_e}(L_{init}/L_{min}) \rceil$ iterations, and thus, we have a strictly increasing series $\left\{f(z^k)\right\}_{k=1}^{\infty}$, where $f(z^k) = \max_{(k-1)\times N_{min}+1,\ldots,k\times N_{min}} \{f(z_i)\}$ and $f(z_i)$ is the function value at the iteration i. Using the monotone convergence theorem [4] we can show that this series converges to the global maximum of the objective function.

3.3 Heteroscedastic Bayesian Optimization using Generalized Product of Experts

In many real-world optimization problems observations are corrupted by a heteroscedastic noise, which depends on the input location. Bayesian optimization (BO) is an efficient approach for the global optimization of black-box functions, but the performance of using a standard GP model can degrade with changing levels of noise due to a homoscedastic noise assumption.

The gPoE model is capable of modeling heteroscedastic noise because each GP expert can have its own individual hyperparameters. After training the experts $\{\mathcal{M}_i\}_{i=1}^{M}$ on the relevant subsets $\{\mathcal{D}_i\}_{i=1}^{M}$, we obtain their predictive distributions $\{p_i\left(y \mid \mathcal{D}_i, x_*\right)\}_{i=1}^{M}$ at the test point x_*, where predictive mean $\mu_{y_i}(x_*)$ and variance $\sigma_{y_i}^2(x_*)$ for each expert is defined as in (2.38) and (2.39). Figure 3.3 illustrates the process of building the gPoE model to handle heteroscedastic noise.

When dealing with heteroscedastic noise we would like to avoid directly aggregating predictive distribution for noisy test data y_*, because its predictive variance includes the heteroscedastic noise variance. From (2.39) we see that for noisy test data y_* the posterior predictive variance for the i-th expert is equal to $\sigma_{y_i}^2(x_*) = \sigma_{f_i}^2(x_*) + \sigma_{\epsilon,i}^2$, where $\sigma_{f_i}^2(x_*)$ is a predictive variance for a noise free test data point x_* and $\sigma_{\epsilon,i}^2$ is the noise variance.

To have a meaningful aggregated prediction variance, where prediction variance approaches zero with an increasing number of observations [40], we aggregate GP experts in latent functions space f. Then predictive distributions are equal to $\{p_i\left(f_* \mid \mathcal{D}_i, x_*\right)\}_{i=1}^{M}$, which are similar to $\{p_i\left(y_* \mid \mathcal{D}_i, x_*\right)\}_{i=1}^{M}$, but without added noise level $\sigma_{\epsilon,i}^2$ to each GP expert variance $\sigma_{y_i}^2\left(x_*\right)$. Conditioned on the related dataset $\mathcal{D}^{(i)}$, individual GP expert predictive distribution $p_i\left(f_*|x_*, \mathcal{D}^{(i)}\right) = \mathcal{N}\left(\mu_{f_i}\left(x_*\right), \sigma_{f_i}^2\left(x_*\right)\right)$ has the posterior predictive mean and variance:

$$\mu_{f_i}\left(x_*\right) = \boldsymbol{k}_{*i}\left(\boldsymbol{K}_i + \sigma_{\epsilon,i}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y_i}, \tag{3.1}$$

$$\sigma_{f_i}^2\left(x_*\right) = \boldsymbol{k}_{**} - \ \boldsymbol{k}_{*i}\left(\boldsymbol{K}_i + \sigma_{\epsilon,\ i}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_{*i}{}^T. \tag{3.2}$$

We can aggregate the predictive distribution for f_* using the gPoE

$$p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right) = \prod_{i=1}^{M} p_i^{\alpha_i\left(x_*\right)}\left(f_*|x_*,\ \mathcal{D}^{(i)}\right), \tag{3.3}$$

where predictive mean and variance is equal to

$$\mu_{f_{\mathcal{A}*}} = \sigma_{f_{\mathcal{A}*}}^2\left(x_*\right) \sum_{i=1}^{M} \alpha_i\left(x_*\right) \sigma_{f_i}^{-2}\left(x_*\right) \mu_{f_i}\left(x_*\right), \tag{3.4}$$

$$\sigma_{f_{\mathcal{A}*}}^{-2} = \sum_{i=1}^{M} \alpha_i\left(x_*\right) \sigma_{f_i}^{-2}\left(x_*\right). \tag{3.5}$$

The weight $\alpha_i\left(x_*\right)$ for each expert *i* at the test point x_* is the difference in differential entropy between the prior $p\left(f_*|\, x_*\right)$ and posterior $p\left(f_*|\, x_*, \mathcal{D}^{(i)}\right)$, which can be computed as

$$\alpha_i\left(x_*\right) = \frac{1}{2}\left(\log \sigma_{f_i**}^2 - \log \sigma_{f_i}^2\left(x_*\right)\right), \tag{3.6}$$

where $\sigma_{f_i**}^2$ is the prior and $\sigma_{f_i}^2\left(x_*\right)$ is the posterior predictive variance at the test point x_* for the i-th GP expert.

To make predictions for noisy observation y_*, we need to map an aggregated predictive GP distribution $p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right)$ through a likelihood function $p\left(y_*|f_*, r_*\right)$. Here, r_* is a heteroscedastic noise variance at a test point x_*. Finally, the aggregated GP predictive distribution for noisy observations $p_{\mathcal{A}}\left(y_*|x_*, \mathcal{D}\right) = \mathcal{N}\left(\mu_{\mathcal{A}*}, \sigma_{\mathcal{A}*}^2\right)$ with posterior predictive mean and variance equal to

$$\mu_{\mathcal{A}*} = \mu_{f_{\mathcal{A}*}}, \sigma^2_{\mathcal{A}*} = \sigma^2_{f_{\mathcal{A}*}} + r_*. \tag{3.7}$$

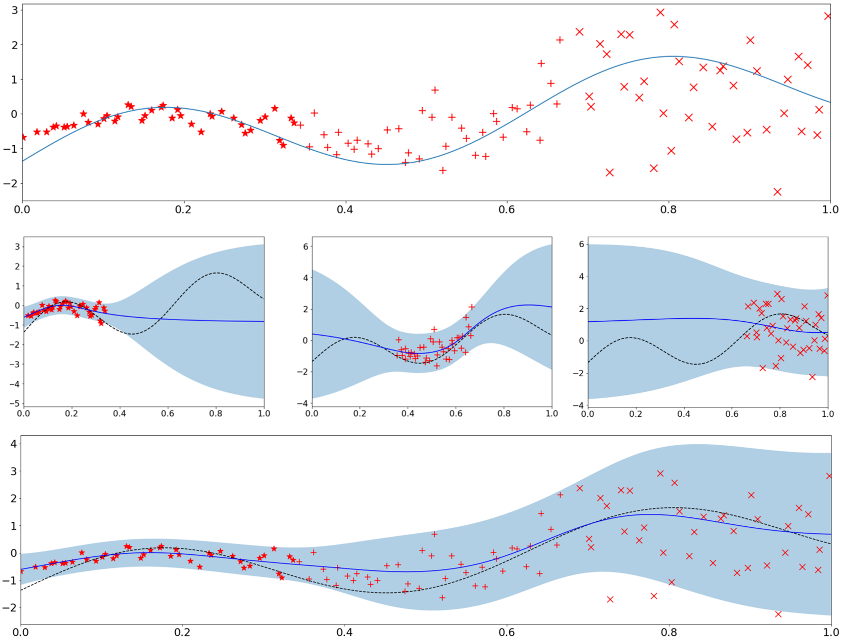


Figure 3.3: Illustration of building a gPoE model with heteroscedastic noise. **(Top)** Suppose we have n noisy observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ (denoted as markers) which are generated by adding heteroscedastic noise to the noise free objective function (smooth line). **(Middle)** We partition the noisy observations into $M = 3$ subsets using a clustering algorithm and build M local GP experts $\{\mathcal{M}_i\}_{i=1}^M$ independently. **(Bottom)** We combine individual GP experts posterior predictions using Equation 2.35 and 2.36 to get the aggregated gPoE model.

We propose to use the posterior noise variances from each GP expert $\left\{\sigma^2_{\epsilon,i}\right\}_{i=1}^M$ to find the r_*. We use the simple weighted average to aggregate the noise variances

$$r_* = r\left(x_*\right) = \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma^2_{\epsilon,i}, \tag{3.8}$$

where $\alpha_i\left(x_*\right)$ is a measure of reliability and quantifies the contribution of each GP expert i at the test point x_* by the prediction precision and is the same weight that is used to aggregate the mean and variance in (3.4) and (3.5).

To model the heteroscedastic noise in BO with the gPoE we propose a modified version of the HAEI acquisition function in combination with the individual noise variance levels learned from each individual GP expert. This acquisition function can be expressed as

$$\alpha_{\text{HAEI}}(x) \quad = \quad \alpha_{\text{EI}_n}(x) \times \left(1 - \frac{\gamma\sqrt{r\,(x)}}{\sqrt{\sigma^2_{f_\mathcal{A}} + \gamma^2 r\,(x)}}\right). \tag{3.9}$$

The aggregated noise variance function $r\,(x) = \sum_{i=1}^{M} \alpha_i\,(x)\,\sigma^2_{\epsilon,i}$ combines the noise levels $\sigma^2_{\epsilon,i}$, where $\alpha_i\,(x) > 0$ and $\sum_{i=1}^{M} \alpha_i\,(x) = 1$. This multiplicative penalty factor accounts for the diminishing return of additional replicates as the predictions become more accurate [28]. Similar to [21], we can show that the original EI acquisition function $\alpha_{\text{EI}_n}(x)$ can be recovered in the case that noise variance $r\,(x) = 0$ and in the case that $r\,(x) > 0$ the penalty factor operates as a rescaling of the EI acquisition function penalising the regions where the GP predictive variance is small relative to the noise variance level $r\,(x)$. Additionally, using the γ parameter we can control the penalty size for regions with high noise variance.

Proposition 1. *The HAEI acquisition function with the gPoE aggregated noise variance reduces to EI when the ratio of predictive posterior variance to predictive noise variance is much greater than γ^2.*

Proof. Let $k = \frac{\sigma^2_{f_\mathcal{A}}}{r(x)}$ denote the ratio of posterior predictive variance to noise variance at an arbitrary input location x. Dividing the numerator and the denominator of the second term in the second factor of (3.9) by $\sqrt{r(x)}$ yields

$$\alpha_{\text{HAEI}}(x) = \alpha_{\text{EI}_n}(x) \times \left(1 - \frac{\gamma}{\sqrt{k + \gamma^2}}\right). \tag{3.10}$$

Taking the limit analytically as k tends to infinity and assuming finite γ, we get

$$\lim_{k\to\infty} \alpha_{\mathrm{HAEI}}(x) = \lim_{k\to\infty} \alpha_{\mathrm{EI}_n}(x) \left(1 - \frac{\gamma}{\sqrt{k+\gamma^2}}\right) = \alpha_{\mathrm{EI}_n}(x), \qquad (3.11)$$

which recovers the expected improvement acquisition. □

In the Proposition 2 we show that the multiplicative factor penalizes the locations with small posterior prediction variance compared to the noise variance and therefore enhances exploration.

Proposition 2. *The HAEI acquisition function with the gPoE aggregated noise variance goes to zero as the ratio of posterior predictive variance to noise variance approaches zero.*

Proof. Taking the limit as k tends to zero in Equation 3.10 yields

$$\lim_{k\to 0} \alpha_{\mathrm{HAEI}}(x) = \lim_{k\to 0} \alpha_{\mathrm{EI}_n}(x) \left(1 - \frac{\gamma}{\sqrt{k+\gamma^2}}\right) = 0. \qquad (3.12)$$

□

We present the pseudocode for heteroscedastic the gPoE with the HAEI acquisition function in Algorithm 5.

Additionally, it can be shown [21] that in the case of large noise variance, when $\lim_{k\to 0}$ the multiplicative penalty $\left(1 - \frac{\gamma}{\sqrt{k+\gamma^2}}\right)$ can be approximated using Taylor expansion around $k = 0$. This approximation can be used when the ratio k is small relative to γ and could provide guidance in setting the γ parameter if prior knowledge about k is available.

Moreover, we propose a modified version of aleatoric noise-penalized expected improvement (ANPEI) [21] acquisition function, where we explicitly penalize the input space regions with large noise levels. The

Algorithm 5 Heteroscedastic GPOEBO with HAEI

```
Input: Number of initializing points $N$, number of iterations $T$, number of points per expert $n_i$.
Output: The best recommendation $x^*_T$.
Randomly select and evaluate $N$ points in the search space $\mathcal{D}_0 = \{(x_i, f(x_i))\}_{i=1}^{N}$.
for $t = 1$ to $T$ do
    Randomly partition $\mathcal{D}_{t-1}$ into $M = |\mathcal{D}_{t-1}|/n_i$ subsets.
    Train $M$ local GP experts on $\{\mathcal{D}^i_{t-1}\}_{i=1}^{M}$ subsets.
    For each $i$-th GP expert, compute noise variance $\sigma^2_{\epsilon,i}$.
    Generate $q$ candidate points $\mathbf{X}^c = \{x^c_1, \ldots, x^c_q\}$ from the search space.
    Evaluate $i$ local GP expert predictive mean $\mu_{f_i}(x^c_j)$ and variance $\sigma^2_{f_i}(x^c_j)$ on $\mathbf{X}^c$ points using (3.1) and (3.2).
    Aggregate $\mu_{f_{\mathcal{A}*}}$ and $\sigma^2_{f_{\mathcal{A}*}}$ using (3.4) and (3.5).
    Compute $r(x^c_j) = \sum_{i=1}^{M} \alpha_i(x^c_j)\sigma^2_{\epsilon,i}$ for all candidate points using (3.8).
    Compute the HAEI acquisition function $\alpha_{\text{HAEI}}(x^c_j)$ for all candidate points using (3.9).
    Select the candidate point with the maximum HAEI value: $\hat{x} = \text{argmax}_{x \in \mathbf{X}^c}\ \alpha_{\text{HAEI}}(x)$.
    Evaluate the objective function $\hat{y} = f(\hat{x})$.
    Add new data point to the dataset $\mathcal{D}_t = \mathcal{D}_{t-1} \cup\ \{\hat{x}, \hat{y}\}$.
end for
```

ANPEI acquisition function for POE models has the form

$$\alpha_{\text{ANPEI}}(x) \quad = \quad \beta \times \alpha_{\text{EI}_n}(x) - \big(1 - \beta\big) \times \sqrt{r\,(x)}, \tag{3.13}$$

where $r\,(x) = \sum_{i=1}^{M} \alpha_i\,(x)\,\sigma^2_{\epsilon,i}$ is an aggregated noise function from the individual GP experts and β is a constant, which controls the penalty trade-off between original EI and the noise variance level. The $\alpha_{\text{EI}_n}(x)$ term contributes positively to $\alpha_{\text{ANPEI}}(x)$ and the noise penalty term contributes negatively. When β is closer to 1, more weight is given to the original EI and less weight is given to the noise penalty. Conversely, when β is closer to 0, more weight is given to the noise penalty and less weight is given to the EI acquisition function. We show the advantages of our proposed modifications of acquisition functions in conjunction with the gPoE model compared to other algorithms in Section 4.2.

3.4 Conclusions

In this chapter, we propose a new approach how to scale Bayesian optimization to problems with a large number of observations. Our proposed algorithms replace the standard Gaussian process model with the generalized product of experts model in Bayesian optimization. This model trains multiple local Gaussian process experts on the subsets of training data and combines their weighted posterior distributions through a product operation. Given that the local Gaussian process experts in the generalized product of experts model can be trained independently, this allows the generalized product of experts based Bayesian optimization to achieve high levels of efficiency and scalability. Additionally, we have presented an approach for performing heteroscedastic Bayesian optimization using the generalized product of experts. We proposed two modified versions of excising heteroscedastic acquisition functions which use the individual noise levels form GP experts and can penalize the input space regions with high noise.

Chapter 4

NUMERICAL EXPERIMENTS

In this chapter, we empirically evaluate the efficiency and scalability of our proposed gPoEBO and gPoETRBO algorithms, and compare them with other baseline algorithms. Additionally, we perform ablation studies to assess the impact of different properties on optimization accuracy. Finally, we conduct an empirical comparison of our proposed heteroscedastic BO algorithms with other baseline algorithms. The results of this chapter have been published in scientific papers [68, 69].

4.1 Scalable Bayesian Optimization

We compare the performance and running times of our proposed gPoEBO and gPoETRBO algorithms with other GP experts based BO algorithms (PoE_BO, BCM_BO, rBCM_BO), sparse GP regression (SGPRBO) based BO, standard BO, TuRBO and random search baselines. Also, we compare gPoEBO with a modified gPoEBO version, where all GP experts share the same hyperparameters, which we call gPoEBO_f. We evaluate the algorithms on four global optimization benchmark functions with varying dimensions and continuous optimal control problems used in [17].

For global optimization benchmark functions we use Rosenbrock, Levy, Ackley and Rastrigin in domains $[-10, 10]^D$, $[-10, 10]^D$, $[-5, 10]^D$, $[-5.12, 5.12]^D$, respectively. Global optimization benchmark functions vary in optimization difficulty. Rosenbrock is unimodal function with the global minimum point found in a narrow, parabolic valley, which makes convergence to the minimum difficult. Levy, Ackley and Rastrigin functions have multiple local minima points, which makes it difficult to find the global minimum. In our experiments we transform the minimization problem to maximization problem by changing the sign of the objective function value for these benchmark functions. We evaluate the performance on these benchmark functions on 20 and 50 dimensions.

For continuous optimal control problems, we use a 12D Lunar Landing, 14D robot pushing problem, a 60D rover trajectory planning problem. The problems are multi-modal and challenging for many global optimization algorithms.

We use random and disjoint data partitioning to assign data points for GP experts models. Also, in all our models we use the Matérn-5/2 kernel with automatic relevance determination (ARD) for computing the covariance matrices. We evaluate all experiments using modest 8 core virtual CPU with 32 GB memory on Google Cloud platform, which is available on regular a laptop.

4.1.1 Results on 20D Benchmark Functions

For optimization on 20D benchmark functions we use a budget of 500 function evaluations with 50 initial points, 550 data points in total. We assign 50 data points per expert for all GP experts based BO algorithms. We choose to use 50 inducing points for the SPGPBO. All experiments are repeated 10 times.

The detailed results of optimization performance and running times are provided in Tables 4.1 and 4.2. The Figure 4.1 shows an optimization progress with respect to the number of evaluations. The improvement in optimization performance and running times for our proposed algorithms and TuRBO algorithm compared to the standard BO on 20D benchmark functions is provided in Table 4.3.

Based on the results, we can see that the best optimization results are achieved using the trust region based algorithms. The GP experts based BO algorithms show similar performance to the standard BO performance, but the runtimes of these algorithms are 2 times better. The gPoEBO algorithm showed the best runtimes on all functions and the best performance on Levy and Rastring functions compared to other GP experts based BO algorithms. Comparing gPoEBO with gPoEBO_f, we see that gPoEBO has better accuracy and runtimes on all functions, except the Ackley function, and faster runtimes on all functions.

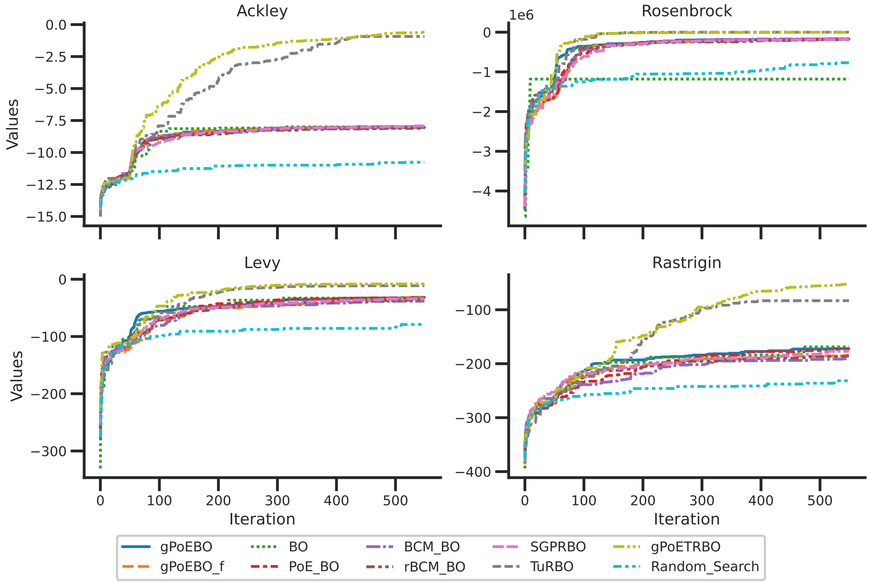


Figure 4.1: Optimization performance on 20D benchmark functions.

Similar results can be seen when comparing gPoETRBO to the TuRBO algorithm. The gPoETRBO algorithm outperformed the TuRBO algorithm on the Ackley and Rastrigin function with 2 times speedup in computational time.

Based on the results in Table 4.3, we can see that for the gPoEBO algorithm, the improvement in accuracies on 20D benchmark functions ranged from -2.87% to 6.45% compared to standard BO. However, the improvement in runtimes ranged from 106.10% to 209.58%, showing a significant improvement in running times while maintaining similar accuracy to the standard BO. For the gPoETRBO algorithm, the im-

Table 4.1: Optimization performance on 20D benchmark functions

	Ackley	Rosenbrock	Levy	Rastrigin
Algorithm				
BO	-7.935 (0.286)	-167163.444 (43356.440)	-33.792 (6.760)	-177.085 (10.278)
PoE_BO	-8.005 (0.269)	-186342.913 (43712.196)	-32.750 (3.270)	-173.878 (8.577)
BCM_BO	-7.990 (0.293)	-163501.899 (42181.016)	-38.033 (6.701)	-190.851 (8.018)
rBCM_BO	-8.101 (0.224)	-179548.000 (54906.183)	-35.413 (4.180)	-185.918 (15.210)
gPoEBO_f	-7.971 (0.373)	-173253.006 (44954.889)	-32.296 (4.525)	-184.812 (9.805)
gPoEBO	-8.043 (0.417)	-171959.197 (25469.116)	-31.614 (5.033)	-172.071 (15.096)
TuRBO	-0.922 (0.730)	**-271.574 (179.904)**	**-7.847 (7.424)**	-74.099 (39.399)
gPoETRBO	**-0.595 (0.067)**	-2549.391 (1455.148)	-8.240 (2.616)	**-52.219 (10.675)**
SGPRBO	-7.930 (0.299)	-182059.152 (32591.474)	-34.010 (6.417)	-176.999 (7.362)
Random Search	-10.511 (0.636)	-1033732.896 (292204.172)	-79.720 (21.641)	-234.806 (16.995)
Optimal value	0	0	0	0

Table 4.2: Optimization running times on 20D benchmark functions (seconds)

	Ackley	Rosenbrock	Levy	Rastrigin
Algorithm				
BO	364.633 (5.146)	566.642 (19.928)	503.049 (17.532)	365.657 (12.068)
PoE_BO	179.413 (4.956)	256.924 (8.953)	247.778 (10.600)	259.505 (8.616)
BCM_BO	178.048 (3.323)	258.223 (7.302)	250.388 (7.184)	262.411 (7.152)
rBCM_BO	179.389 (3.976)	268.037 (11.002)	244.259 (8.945)	260.350 (8.493)
gPoEBO_f	181.411 (4.555)	257.952 (8.118)	251.952 (10.063)	249.782 (5.401)
gPoEBO	**176.917 (6.153)**	**183.036 (7.698)**	**169.257 (2.886)**	**169.936 (2.201)**
TuRBO	380.452 (2.115)	434.922 (3.055)	440.946 (4.397)	442.488 (3.439)
gPoETRBO	202.628 (4.404)	224.513 (5.732)	226.182 (3.426)	216.930 (6.729)
SGPRBO	388.049 (6.269)	383.875 (3.950)	386.994 (1.996)	385.133 (3.967)
Random Search	0.091 (0.010)	0.059 (0.008)	0.122 (0.010)	0.067 (0.012)

provement in accuracies ranged from 70.51% to 98.47% compared to the standard BO on 20D benchmark functions. The improvement in runtimes ranged from 68.56% to 152.39%, indicating an improvement in optimization performance and efficiency over the standard BO. Comparing gPoETRBO to TuRBO, we can see that the gPoETRBO algorithm can match the accuracies of the TuRBO algorithm while demonstrating significant improvements in running times across all benchmark functions. For the Ackley function, gPoETRBO outperformed TuRBO in accuracy and showed a runtime improvement of 79.95% compared to decline of 4.16% for the TuRBO algorithm. Similarly, for the Rastrigin function, gPoETRBO surpassed TuRBO in accuracy and achieved a runtime improvement of 68.56%, while TuRBO showed a decrease of

17.36%.

Table 4.3: Improvement in optimization performance and running times compared to the standard BO on 20D benchmark functions

	Ackley	Rosenbrock	Levy	Rastrigin
gPoEBO				
Accuracy (%)	-1.36	-2.87	6.45	2.83
Runtime (%)	106.10	209.58	197.21	115.17
gPoETRBO				
Accuracy (%)	92.50	98.47	75.62	70.51
Runtime (%)	79.95	152.39	122.41	68.56
TuRBO				
Accuracy (%)	88.38	99.84	76.78	58.16
Runtime (%)	-4.16	30.29	14.08	-17.36

4.1.2 Results on 50D Benchmark Functions

The performance on 50D benchmark functions is evaluated using a budget of 2000 function evaluations with 100 initial data points. We chose the expert size to be 200 data points per experts for all GP expert based BO algorithms. We repeated the experiments 10 times and the optimization performance and computational time with the standard deviation of the results are presented in Table 4.4 and 4.5, while Figure 4.2 shows the optimization progress over the number of iterations. The improvement in optimization performance and running times for our proposed algorithms and TuRBO algorithm compared to the standard BO on 50D benchmark functions is provided in Table 4.6.

We see that the gPoEBO algorithm is not able to achieve the expected results on 50D benchmark functions. Its accuracy is worse than standard BO and random search on Levy and Rosenbrock functions. Other GP expert models based BO performed better than gPoEBO and performance closely matched the standard BO. Despite that, when comparing the efficiency of the algorithms, we see that the GP experts based BO algorithms are more efficient and their computational times are between 7 to 10 times shorter than standard BO depending on the optimization functions.

Comparing gPoEBO_f with gPoEBO we see that gPoEBO_f achieves better accuracy on all functions, even though it has slightly worse runtime than gPoEBO.

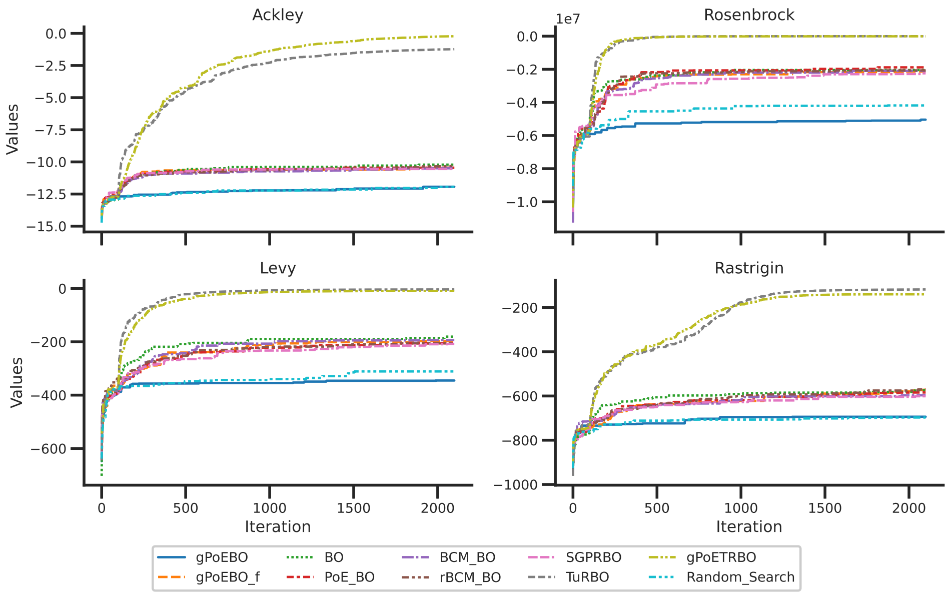


Figure 4.2: Optimization performance on 50D benchmark functions.

Table 4.4: Optimization performance on 50D benchmark functions

Algorithm	Ackley	Rosenbrock	Levy	Rastrigin
BO	-10.216 (0.242)	-2042973.383 (191889.884)	-180.611 (7.079)	-573.166 (17.653)
PoE_BO	-10.451 (0.104)	-1881639.575 (292858.153)	-205.602 (8.156)	-584.058 (20.305)
BCM_BO	-10.498 (0.226)	-2093670.300 (200765.134)	-193.730 (8.170)	-596.781 (11.721)
rBCM_BO	-10.321 (0.129)	-2062894.800 (152171.924)	-204.405 (5.692)	-569.846 (12.555)
gPoEBO_f	-10.470 (0.188)	-2132363.400 (252235.246)	-197.435 (15.438)	-571.605 (22.915)
gPoEBO	-11.935 (0.316)	-5039677.700 (391263.730)	-345.149 (19.835)	-683.752 (24.088)
TuRBO	-1.230 (0.166)	**-384.571 (121.836)**	**-3.610 (2.813)**	**-118.111 (38.689)**
gPoETRBO	**-0.220 (0.034)**	-7956.787 (3665.374)	-9.382 (6.797)	-140.077 (31.921)
SGPRBO	-10.553 (0.132)	-2241390.000 (199413.062)	-207.832 (6.421)	-601.082 (27.282)
Random_Search	-11.982 (0.241)	-4180252.092 (432158.966)	-301.336 (33.074)	-689.411 (19.190)
Optimal value	0	0	0	0

Trust region method based gPoETRBO and TuRBO achieve the best performance compared to other BO algorithms. We see that the gPoETRBO algorithm shows very good results on all 50D benchmark functions. It matches the performance of the state-of-the-art TuRBO algorithm and achieves a better accuracy on the Ackley function. Moreover, gPoETRBO is very efficient with a runtimes 11 to 12 times shorter than the TuRBO algorithm on all benchmark functions. For example, on the Ackley function gPoETRBO runtime was 2265.742 seconds compared 26864.023 seconds for TuRBO, which is 11.85 times shorter. For Levy function the runtime for gPoETRBO is 12.45 times shorter (2108.032 seconds vs 26240.322 seconds).

Table 4.5: Optimization running times on 50D benchmark functions (seconds)

	Ackley	Rosenbrock	Levy	Rastrigin
Algorithm				
BO	17388.273 (1858.737)	19893.231 (2011.091)	25690.148 (997.904)	21093.616 (540.624)
PoE_BO	2244.812 (66.189)	2924.190 (123.574)	3305.685 (139.850)	3886.263 (129.530)
BCM_BO	2269.010 (58.671)	3001.214 (100.393)	3324.535 (128.264)	3984.716 (58.673)
rBCM_BO	2235.832 (55.324)	2920.929 (109.558)	3332.704 (88.607)	3625.325 (201.311)
gPoEBO_f	2200.432 (97.215)	2947.697 (87.119)	3314.335 (125.566)	3717.829 (144.038)
gPoEBO	**2123.860 (19.559)**	2157.512 (17.734)	2109.236 (17.261)	2116.775 (25.977)
TuRBO	26864.023 (1921.912)	25075.302 (99.474)	26240.322 (289.881)	25365.491 (94.976)
gPoETRBO	2265.742 (56.076)	**2058.945 (29.689)**	**2108.032 (68.645)**	**2071.761 (33.439)**
SGPRBO	9260.272 (138.201)	9430.895 (367.998)	9154.619 (110.815)	9329.038 (185.747)
Random_Search	0.304 (0.018)	0.257 (0.054)	0.512 (0.069)	0.212 (0.032)

Table 4.6: Improvement in optimization performance and running times compared to the standard BO on 50D benchmark functions

	Ackley	Rosenbrock	Levy	Rastrigin
gPoEBO				
Accuracy (%)	-16.83	-146.68	-91.10	-19.29
Runtime (%)	718.71	822.04	1117.98	896.50
gPoETRBO				
Accuracy (%)	97.85	99.61	94.81	75.56
Runtime (%)	667.44	866.19	1118.68	918.15
TuRBO				
Accuracy (%)	87.96	99.98	98.00	79.39
Runtime (%)	-35.27	-20.67	-2.10	-16.84

Comparing the algorithms improvement over the standard BO in Table 4.6, we can see that for the gPoEBO algorithm accuracies declined compared to the standard BO on 50D benchmark functions, with reductions ranging from -146.68% to -16.83%. However, runtimes increased between 718.71% and 1117.98%, implying that despite not being able to improve the accuracy, the gPoEBO is still very efficient in handlling large number of observations compared to the standard BO. For the gPoETRBO algorithm, accuracies improved between 75.56% and 99.61% compared to the standard BO. Also, we see that runtime improvements are ranging from 667.44% to 1118.68%, indicating significant optimization performance and efficiency gains over the standard BO approach. For example, on the Levy benchmark function, the gPoETRBO algorithm demonstrated a 94.81% accuracy improvement compared to the standard BO. Comparing the runtimes, gPoETRBO consistently outperformed TuRBO across all 50D benchmark functions with improvements range between 667.44% and 1118.68%, while TuRBO showed declines ranging from -2.10% to -35.27%.

Our results show GP experts based BO suffers from the same over-exploration problem as in standard BO. Moreover, we see that GP expert models, which share the same hyperparameters perform slightly better compared to the gPoEBO algorithm, where GP experts have individual hyperparameters. The poor results of the gPoEBO algorithm on high dimensional functions can be explained by the conservative variance prediction problem discussed in Section 2.6.1.2.

4.1.3 12D Lunar Landing Reinforcement Learning

The goal for 12D Lunar Landing problem is to learn a controller for a lunar lander that minimizes fuel consumption and distance to a landing target, while also preventing crashes. We used the same heuristic policy from TuRBO [17] that has 12 parameters to optimize. The objective is to maximize the average final reward over 5 episodes because the simulation can be sensitive to small perturbations as was noticed by [17].

Table 4.7 shows the results for a total of 1000 function evaluations with 50 initial points for all algorithms with the runtimes provided in Table 4.8. The optimization progress is depicted in Figure 4.3. We choose to assign 50 data points per expert for all GP experts based BO algorithms with a maximum number of six experts. The results demonstrate that all algorithms except random search achieve better rewards than the handcrafted controller provided by OpenAI, whose value is around 280 [17]. We see that the gPoEBO_f algorithm reported the highest reward among other GP expert models, but was slightly worse than standard BO. However, when comparing the runtimes in

Table 4.8, we see that the GP experts based BO models are 6-10 times faster than standard BO. For example, the PoE_BO runtime was 6.17 times better compared with standard BO (1146.710 seconds compared to 7075.555 seconds), while the gPoEBO runtime is 10.64 times shorter (664.947 seconds compared to 7075.555 seconds).

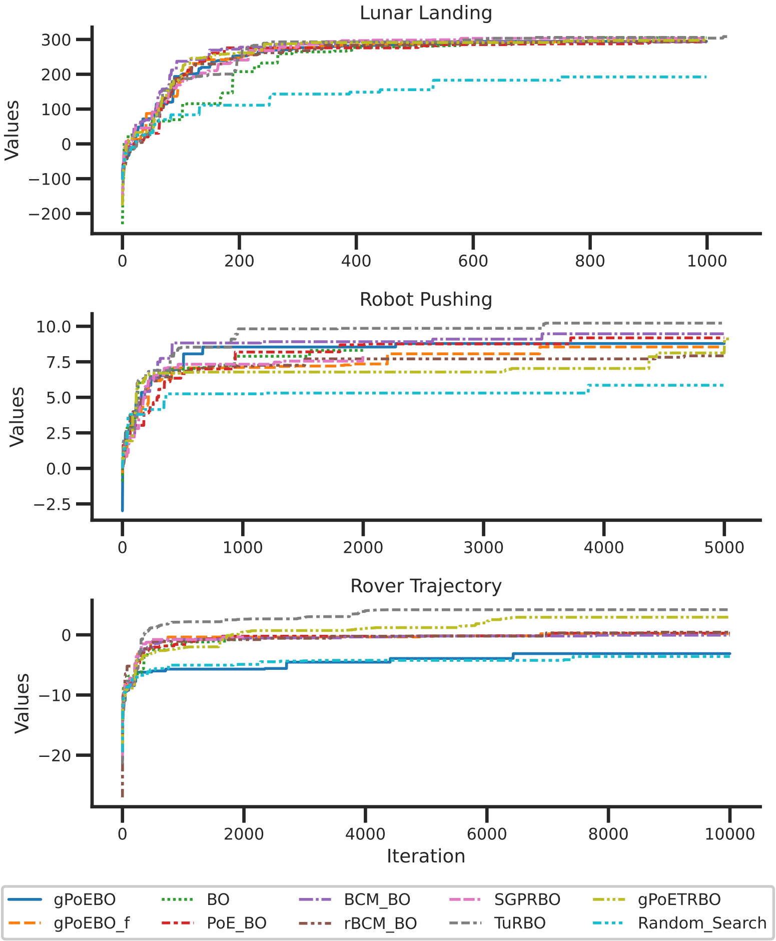


Figure 4.3: Optimization performance on optimal control problems.

Trust region-based algorithms gPoETRBO and TuRBO show similar accuracy and runtimes. It is evident from Table 4.8 that gPoETRBO runtime is 16.59 times shorter that compared to the standard BO (426.478 seconds compared to 7075.555 seconds), while TuRBO runtime is 14.16 times shorter (499.557 seconds compared to 7075.555 seconds). Comparing gPoETRBO and TuRBO to the other GP experts based BO algorithms

Table 4.7: Optimization performance on optimal control problems

Algorithm	12D Lunar Landing	14D Robot Pushing	60D Rover Trajectory
BO	305.028 (3.131)	8.308 (0.384)	-0.365 (0.172)
PoE_BO	292.794 (17.844)	9.185 (0.901)	0.278 (0.901)
BCM_BO	293.580 (5.083)	9.472 (0.525)	-0.059 (0.212)
rBCM_BO	293.738 (7.249)	7.919 (0.710)	0.438 (0.321)
gPoEBO_f	295.418 (11.217)	8.545 (0.505)	0.183 (0.170)
gPoEBO	294.274 (4.865)	8.787 (0.176)	-3.118 (0.297)
TuRBO	**305.910 (4.530)**	**10.223 (0.171)**	**4.186 (0.622)**
gPoETRBO	300.026 (8.352)	8.123 (1.394)	2.935 (0.652)
SGPRBO	303.285 (6.393)	8.003 (0.009)	-0.352 (0.396)
Random_Search	192.280 (52.448)	5.852 (0.147)	-3.603 (0.962)

we see that their runtimes are up to 1.5 times shorter. Moreover, we noticed that trust region based algorithms with restart strategy are restarted multiple times during the optimization for this problem, because they stuck in the local maximum and are not able to make any further progress. For this reason, models do not use all collected data points and GP is trained only on a small subset of newly collected data points, so training the GP is very fast and efficient.

Table 4.8: Optimization running times on optimal control problems (seconds)

Algorithm	12D Lunar Landing	14D Robot Pushing	60D Rover Trajectory
BO	7075.555 (1995.906)	8972.261 (659.187)	11083.001 (121.252)
PoE_BO	1146.710 (139.087)	3550.812 (21.950)	28004.079 (724.833)
BCM_BO	1004.450 (105.564)	3424.899 (39.532)	27947.945 (496.754)
rBCM_BO	987.079 (57.678)	3481.163 (192.793)	31972.164 (1299.382)
gPoEBO_f	962.113 (166.063)	3350.469 (119.729)	27262.038 (389.318)
gPoEBO	664.947 (82.385)	1895.881 (5.192)	5322.493 (54.741)
TuRBO	499.557 (135.357)	1288.586 (39.326)	31219.460 (755.911)
gPoETRBO	**426.478 (41.860)**	**1113.866 (18.665)**	**4981.538 (89.362)**
SGPRBO	970.373 (51.176)	8273.872 (54.583)	9346.468 (442.238)
Random_Search	43.447 (10.320)	24.744 (0.155)	11.893 (1.148)

Comparing the gPoEBO and gPoETRBO improvement over the standard BO in Table 4.9, we can see that accuracy for gPoEBO is 3.53 % lower, but improvement in running time is 964.08 % better, indicating a substantial improvement over the standard BO. Similarly, for the gPoETRBO the improvement in accuracy is 1.64% lower, but the improvement in runtime is very significant and is 1559.07%. We can see that only TuRBO is able to improve the accuracy by 4.2% with a runtime improvement over 1316.37%, which is significant, but lower than gPoETRBO.

Table 4.9: Improvement in optimization performance and running times compared to the standard BO on optimal control problems

	12D Lunar Landing	14D Robot Pushing	60D Rover Trajectory
gPoEBO			
Accuracy (%)	-3.53	5.77	-754.25
Runtime (%)	964.08	373.25	108.23
gPoETRBO			
Accuracy (%)	-1.64	-2.23	904.11
Runtime (%)	1559.07	705.51	122.48
TuRBO			
Accuracy (%)	4.20	7.93	1246.85
Runtime (%)	1316.37	596.29	-64.50

4.1.4 Robot Pushing

The robot pushing problem is a 14D control problem considered in [17, 79]. We run each method for a total of 5K evaluations with initial 100 data points, except for standard BO and SGPRBO where we run only 2K evaluations, because of the limitation of the algorithms. We assign 100 data points per experts for GP experts and limit the maximum number of GP experts to 6. For the SGPRBO we use 50 inducing points.

We observe from Table 4.7 and Figure 4.3 that all algorithms outperform the random search. The BCM_BO algorithm shows the best performance between the GP experts based BO algorithms. The gPoEBO shows slightly worse results than BCM_BO, but from Table 4.8, we can see that the runtime is almost 2 times faster. The gPoETRBO shows slightly worse performance than TuRBO, but has the shortest runtime compared to other BO algorithms. Note that [79] reported a median value of 8.3 for their EBO after 30K evaluations, while gPoEBO and gPoETRBO achieved mean and median rewards of around 8.8 and 8.1 only after 5K samples.

Analyzing the data in Table 4.9 for the 14D Robot Pushing problem, we observe that gPoEBO accuracy surpasses the standard BO by 5.77%. Furthermore, its running time shows an improvement of 373.25%, showing its efficiency. For the gPoETRBO algorithm, there is a slight decrease in accuracy by 2.23% relative to the standard BO. Nevertheless, it shows a significant runtime improvement of 705.51%. On the other hand, TuRBO shows an improvement of 7.93% in accuracy with an improvement in runnting time of 596.29%, which is lower than gPoETRBO runtime.

4.1.5 Rover Trajectory Planning

The Rover trajectory planning problem is 60D control problem considered in [17, 77, 79]. Here the goal is to optimize the locations of 30 points in the 2D-plane that determine the trajectory of a rover. We run each method for a total of 10K evaluations with initial 200 data points, except for standard BO and SGPRBO, which we run only for 2K evaluations. For GP experts based BO algorithms, we assign 100 data points per expert and limit the maximum number of experts to six experts. We use 100 inducing points for the SGPRBO algorithm.

The results in Table 4.7 demonstrate that after 10K evaluations the best results between GP expert based BO is achieved by rBCM_BO. The gPoEBO shows only a slightly better performance than the random search, but unperformed compared to other algorithms. We see that gPoEBO_f achieve higher reward than gPoEBO, despite having 5 times shorter computational time. We can see from Figure 4.2 that trust region based BO show the best results and TuRBO achieve the best overall performance. Authors in [79] reported a mean value of 1.5 for EBO after 35K evaluations, while TuRBO and gPoETRBO achieved a mean reward of about 4.1 and 2.9 after 10K evaluations, respectively. We see from Table 4.8 that gPoETRBO mean runtime is 4981.538 seconds (1.38 hours), which compared to TuRBO 31219.46 seconds (8.67 hours) provides 6.27 times speedup in computational time.

Comparing the gPoEBO and gPoETRBO improvement over the standard BO in Table 4.9 for the 60D Rover Trajectory, we find that the accuracy for gPoEBO has deteriorated significantly and was 754.25% lower. We compare the running times for gPoEBO, gPoETRBO and TuRBO for 10k evaluations with 2k evaluations of the standard BO, because it is not able to handle larger number of iterations. Comparing the running times for gPoEBO we still see an improvement of 108.23%. In the case of gPoETRBO, the accuracy shows a significant improvement by 904.11% with an improvement in runtime by 122.48%. We can see that TuRBO achieved the best improvement in accuracy of 1246.85%. However, the running time compared to the standard BO is lower -64.50%.

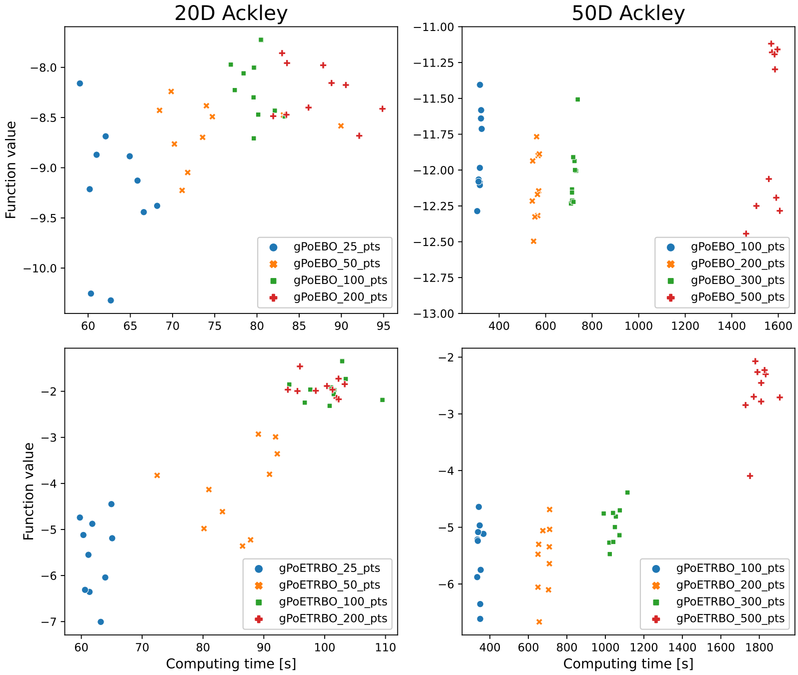
4.1.6 Ablation Studies

We investigate how the data size assigned to the expert in gPoEBO and gPoETRBO affects the optimization performance. We also compare the performance of gPoEBO, gPoETRBO, BO and TuRBO on 20D and 50D with time-restricted budget of 5 and 15 minutes. Finally, we compare two different point allocation strategies for GP experts in the gPoEBO algorithm.

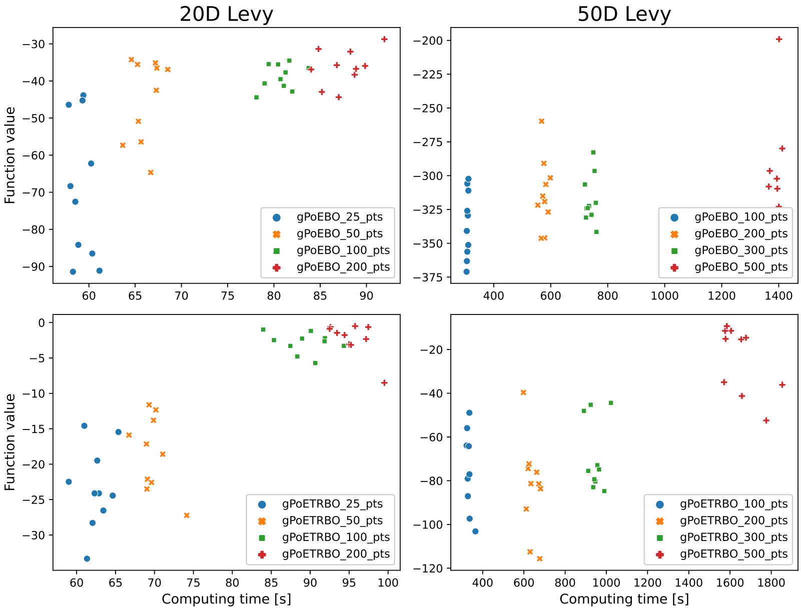
4.1.6.1 The impact of number of points per expert

We choose to evaluate the impact of the number of data points assigned to the GP expert size on optimization accuracy using four synthetic benchmark function in 20D and 50D. For 20D experiments, we initialize algorithms with 200 initial points and use 200 function evaluations. We evaluate the algorithms using 25, 50, 100 and 200 data points per expert. For 50D experiments, we use 1000 initial data points with 500 function evaluations and evaluate the performance using 100, 200, 300 and 500 data points per GP expert. Each experiment is repeated 10 times.

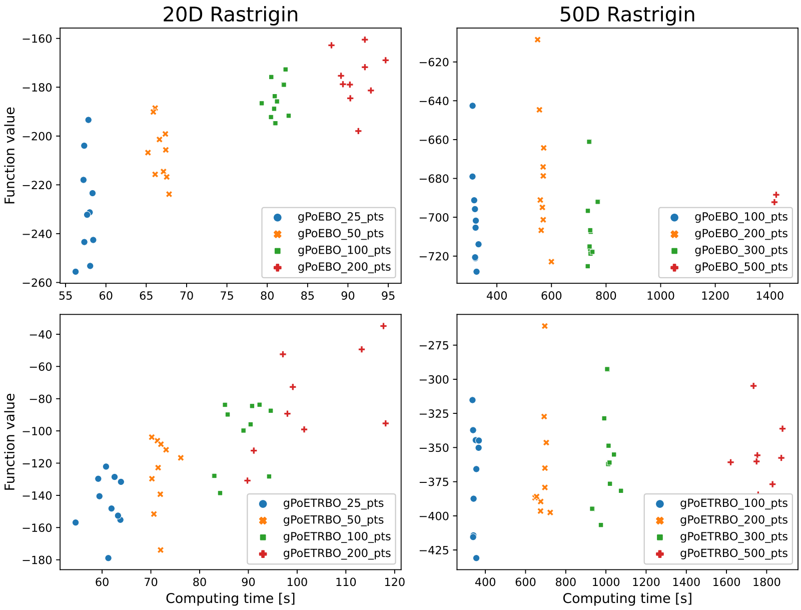
Figure 4.4 shows the optimization results for different 20D and 50D functions for both algorithms. The plot shows the trade-off between accuracy and runtime. As we assign more training points per expert the accuracy improves, but computing time increases. We see that the best balance between accuracy and computing time is achieved by gPoETRBO compared to gPoEBO on 20D and 50D functions. We also see that for higher dimensional functions on 50D increasing the GP expert size does not improve the accuracy for the gPoEBO algorithm. However, we see that for gPoETRBO increasing the size of data points per expert results in an improvement of accuracy, but also an increased computational time.



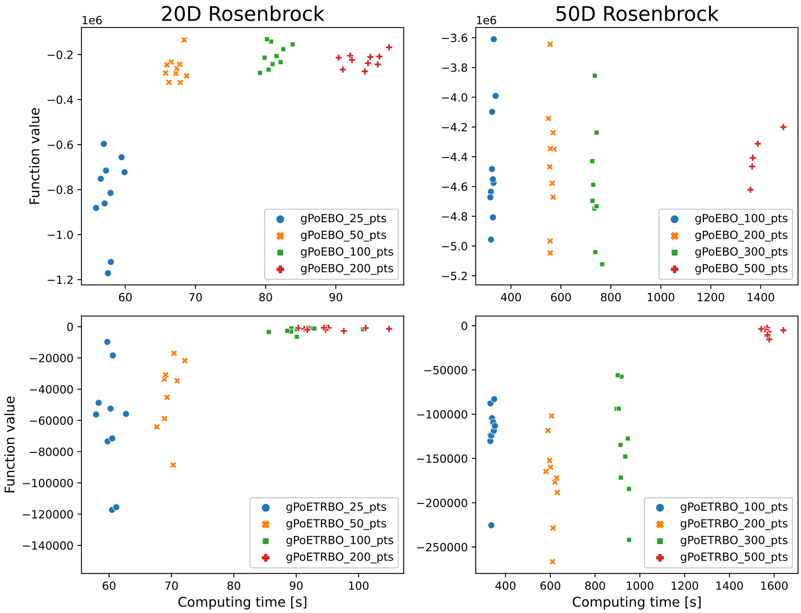
(a) Ackley function



(b) Levy function



(c) Rastrigin function



(d) Rosenbrock function

Figure 4.4: The effect of the number of data points per expert on optimization performance for gPoEBO and gPoETRBO algorithms

4.1.6.2 Time restricted optimization experiments

To evaluate the efficiency of the proposed algorithms we compare the performance of gPoEBO against BO and gPoETRBO against TuRBO with time restricted budget on 20D and 50D synthetic benchmark function. For 20D and 50D benchmark functions we run the experiments for 5 and 15 minutes respectively.

We can see from the results in Fig. 4.5 that for 20D benchmark functions gPoEBO performance was better than BO on all functions, except for the Rosenbrock function. Comparing the performance of gPoETRBO with TuRBO, we can see that gPoETRBO achieved similar or better performance as TuRBO on all functions except on the Rastring function.

Figure 4.6 shows that the standard BO outperformed gPoEBO on all 50D benchmark functions. The poor performance is the result of the

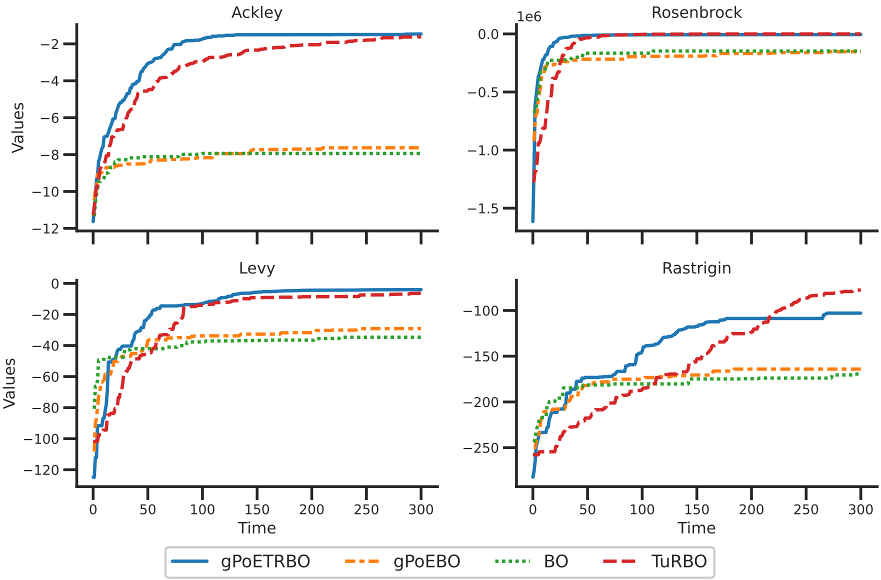


Figure 4.5: Optimization performance on 20D benchmark functions with time (in seconds) restricted budget.

conservative variance prediction problem in gPoEBO which becomes even worse in high dimensions. However, gPoETRBO outperformed TuRBO all 50D benchmark functions with a time-restricted budget.

Based on the results above we see that gPoEBO is not able to achieve good results on high dimensional functions. However, we see that gPoETRBO algorithm is very effective on high dimensional functions when we have limited amount of time to run the optimization and using only modest computational resources.

4.1.6.3 Point allocation strategies for GP exports

We compared the performance of gPoEBO with two different point allocation strategies for GP experts: 1) the points are randomly partitioned into disjoint subsets (gPoEBO_RANDOM) and 2) the points are partitioned into disjoint subsets using k-means clustering technique (gPoEBO_KMEANS). We use time-restricted budget of 5 and 15 minutes for 20D and 50D synthetic benchmark functions respectively.

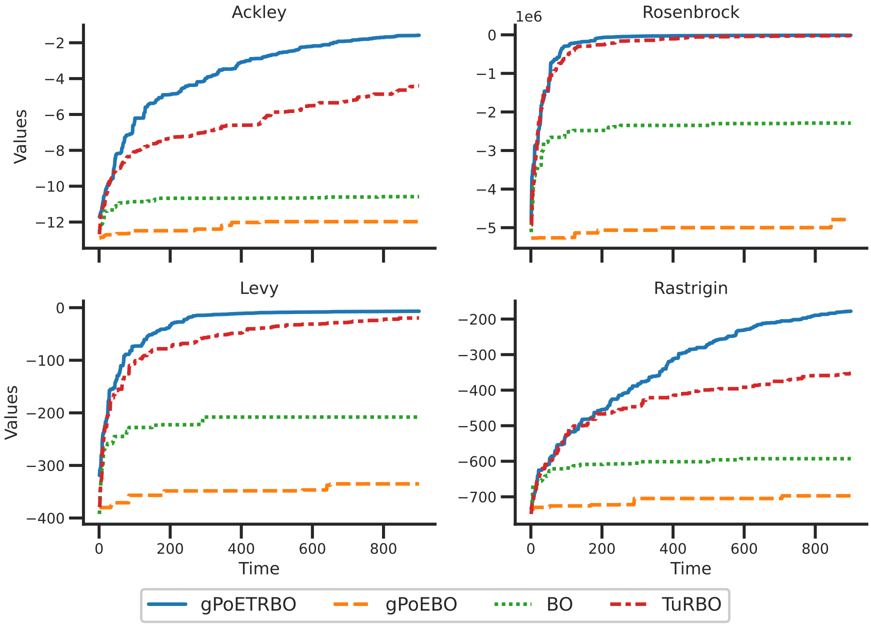


Figure 4.6: Optimization performance on 50D benchmark functions with time (in seconds) restricted budget.

We can see from the 20D experiment results in Figure 4.7 that performance for both strategies is very similar, but gPoEBO_RANDOM achieves slightly better objective function values than gPoEBO_KMEANS.

Comparing the results for 50D benchmark functions in Figure 4.8 we see that the results are mixed and gPoEBO_RANDOM shows better results on Ackley and Rosenbrock functions, while gPoEBO_KMEANS shows better results on Levy and Rastring functions.

Based on the results, we do not see a significant difference between these two point allocation strategies for GP experts. Therefore, we would recommend using a point allocation strategy where points are randomly partitioned into disjoint subsets, because this strategy is simpler and easier to implement.

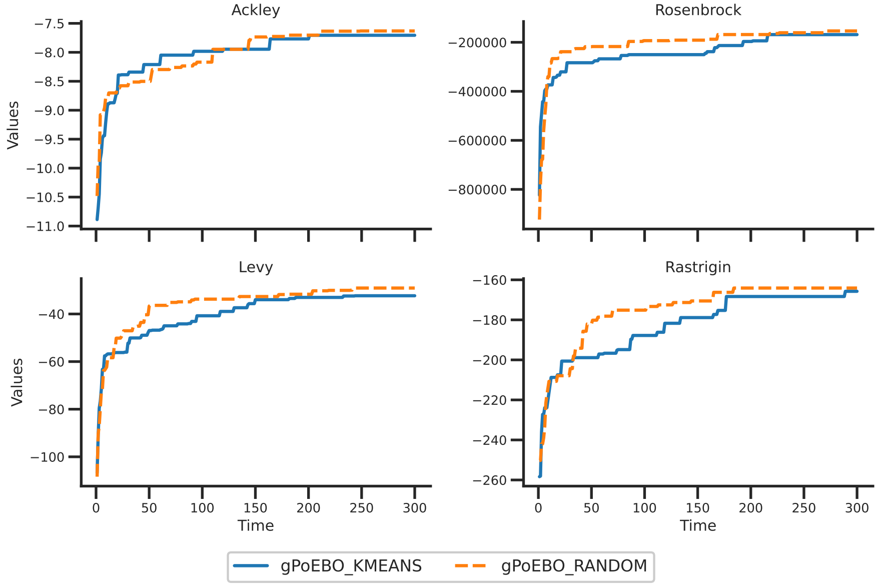


Figure 4.7: Optimization performance on 20D benchmark functions with different point allocation strategies and restricted-time budget (in seconds).

4.2 Heteroscedastic Bayesian Optimization

In this section we will empirically evaluate the algorithms on six synthetic global optimization functions and two real-world scientific datasets.

4.2.1 Synthetic Benchmark Functions Optimization

The performance of our proposed algorithms is evaluated on six widely used synthetic global optimization functions as in [51], which cover important problem properties, such as uni- and multi-modality. We use functions with 2D (Branin, GoldsteinPrice), 4D (Hartman, Rosenbrock) and 6D (Hartman, Sphere) search space dimensions. The search spaces of the original functions have been rescaled to $[0, 1]^D$, with their mean equal to zero and variance equal to one. To simulate heteroscedastic noise, we model the noise variance function by the Sphere function, which is scaled to a range of [0.0, 1.0].

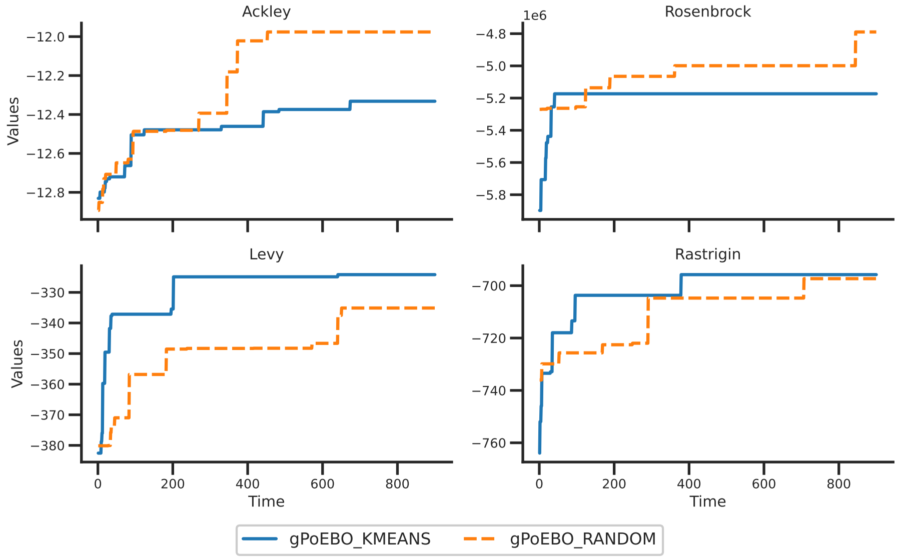


Figure 4.8: Optimization performance on 50D benchmark functions with different point allocation strategies and restricted-time budget (in seconds).

We compare the performance of PoE based BO (POEBO) and gPoE based BO (GPOEBO) using our proposed modifications of heteroscedastic acquisition functions (HAEI, ANPEI) against heteroscedastic BO algorithm version which replaces the standard GP model with MLHGP model (MLHGPBO)[21]. Also, we include the results of homoscedastic BO with EI (BO_EI) and AEI(BO_AEI) acquisition functions to demonstrate the benefits of considering heteroscedastic noise in optimization problems. Moreover, we compare our results with random search as it is known to be competitive with BO in noisy settings [21].

We train GP expert models on the subsets of data partitioned by using a random and disjoint point allocation strategy. This strategy performs similarly to using the clustering based allocation strategy using k-means algorithm, see the results in Section 4.2.4.1. This is in line with a work of [14] which found that simple random data partitioning worked similarly or better compared to the clustering algorithms.

In our experiments we set γ and β to equal values of 0.1 for 2D, 0.5 for 4D and 6D optimization functions in HAEI and ANPEI acquisition functions, respectively. We use the code and parameters for MLHGP based BO from [21].

All experiments are initialized with $10 \times D$ data points depending on the search space dimensionality. We set the maximum number of function evaluations to $5 \times D$ and use $4 \times D$ number of points per experts for PoE and gPoE models.

The performance measure is the absolute difference between the function value at the best $f(x^*)$ and the actual maximum of the noise free function f^* rescaled by the benchmark function standard deviation. Since our synthetic functions are scaled to have zero mean and a variance of one, the performance metric is the absolute error. Each experiment is repeated 50 times.

The results in Table 4.10 show the mean absolute error and standard deviation of absolute error (in brackets) between the function value at the best-found point and the actual function maximum across different synthetic benchmark functions for repeated runs. We can see that our proposed modifications of the acquisition function combined with the gPoE model show the best performance on all functions, except the Sphere function. The GPOEBO_HAEI algorithm had the lowest error in GoldsteinPrice, Rosenbrock, and Hartmann6D benchmark functions, with values of 0.8265, 0.0039, and 0.2932, respectively. The GPOEBO_-ANPEI algorithm achieved the best performance in the Branin function with an error value of 0.0332, which was the lowest among all the algorithms. Additionally, it showed the best result for the Hartmann4D function with an error of 0.1630. However,it is worth noting that the POEBO_ANPEI algorithm outperformed other algorithms for the Sphere function with an error of 0.0040, while other algorithms performed worse than random search. Also, we can see that on average the gPoE based BO performs better than PoE based BO on most benchmark functions. Moreover, comparing the performance of the GPOEBO based algorithms (GPOEBO_HAEI and GPOEBO_ANPE) and MLHGPBO algorithms (MLHGPBO_HAEI and MLHGPBO_ANPEI), we see that our proposed algorithms outperform MLHGPBO algorithms on all synthetic benchmark functions.

We compare the improvement in optimization accuracy for our pro-

Table 4.10: The optimization performance on heteroscedastic synthetic benchmark functions.

Function	Branin	GoldsteinPrice	Hartmann4D	Rosenbrock	Hartmann6D	Sphere
BO_EI	0.0422 (0.0370)	0.8930 (0.5840)	0.1839 (0.1927)	0.0048 (0.0055)	0.3473 (0.2477)	0.0236 (0.0099)
BO_AEI	0.0422 (0.0322)	0.8966 (0.5946)	0.1734 (0.1858)	0.0055 (0.0064)	0.3056 (0.1999)	0.0235 (0.0098)
GPOEBO_HAEI	0.0333 (0.0273)	**0.8265 (0.5558)**	0.1759 (0.1517)	**0.0039 (0.0034)**	**0.2932 (0.2144)**	0.0156 (0.0063)
GPOEBO_ANPEI	**0.0332 (0.0262)**	0.8371 (0.5237)	**0.1630 (0.1464)**	0.0057 (0.0071)	0.3109 (0.2035)	0.0171 (0.0066)
POEBO_HAEI	0.0406 (0.0330)	0.9671 (0.5655)	0.8603 (0.6499)	0.0834 (0.0620)	1.3205 (0.1804)	0.0043 (0.0027)
POEBO_ANPEI	0.0371 (0.0294)	0.9509 (0.5859)	1.0092 (0.5935)	0.0945 (0.0566)	1.3242 (0.1771)	**0.0040 (0.0025)**
MLHGPBO_HAEI	0.0443 (0.0319)	1.0693 (0.5463)	0.2851 (0.2197)	0.0214 (0.0159)	0.5252 (0.2984)	0.0204 (0.0088)
MLHGPBO_ANPEI	0.0465 (0.0444)	1.0004 (0.6133)	0.3079 (0.2473)	0.0245 (0.0127)	0.5231 (0.3439)	0.0174 (0.0042)
RANDOM_SEARCH	0.0802 (0.0748)	1.5819 (0.5698)	1.0465 (0.4233)	0.0234 (0.0241)	0.9077 (0.2488)	0.0082 (0.0040)

Table 4.11: Improvement in optimization accuracy compared to the standard BO and MLHGPBO on heteroscedastic synthetic benchmark functions

	Branin	GoldsteinPrice	Hartmann4D	Rosenbrock	Hartmann6D	Sphere
Improvement over BO_EI (%)						
GPOEBO_HAEI	21.09	7.45	4.35	18.75	15.58	33.90
GPOEBO_ANPEI	21.33	6.26	11.36	-18.75	10.48	27.54
Improvement over MLHGPBO (%)						
GPOEBO_HAEI	24.83	22.71	38.30	81.78	44.17	23.53
GPOEBO_ANPEI	28.60	16.32	47.06	76.73	40.57	1.72

posed algorithms GPOEBO_HAEI and GPOEBO_ANPEI compared to the standard BO on synthetic benchmark functions. The results are presented in Table 4.11. We can see that for the GPOEBO_HAEI algorithm, the improvement in accuracies on synthetic benchmark functions ranged from 4.35% to 33.90% compared to standard BO. For the GPOEBO_ANPEI algorithm, the improvement in accuracies on synthetic benchmark functions ranged from 6.26% to 27.54% compared to the standard BO, except for the Rosenbrock function, where we see a decline of -18.75%.

Comparing the GPOEBO_HAEI to MLHGPBO_HAEI we can see that the improvement in accuracy for GPOEBO_HAEI compared to MLHGPBO_HAEI ranged from 22.71% to 81.78%. We can see similar results when comparing the GPOEBO_ANPEI and MLHGPBO_ANPEI, the improvement for GPOEBO_ANPEI ranged from 1.72% to 76.73%.

Based on the results, we see that both our proposed algorithms consistently exhibit superior performance across most of the synthetic benchmark functions compared to the other algorithms. Their respective results suggest that they are capable of achieving more accurate approximations of the actual function maxima, thereby demonstrating a better capability to handle heteroscedastic noise.

We also evaluate the performance sensitivity of the PoE and gPoE based BO algorithms using the synthetic benchmark functions to a different number of points per expert in Section 4.2.4.

4.2.2 Soil Phosphorus Fraction Optimisation

In this real-world experiment we consider the optimization of the phosphorus fraction of soil, which is an essential nutrient for plant growth. We use the data provided by [21] to study the relationship between bulk soil density and the phosphorus fraction with the goal of minimising the phosphorus content of soil. Since we do not have access to a real objective function, we use a subset of the data for algorithm initialization and the query points are selected by mapping to the closest point in the heldout data. This is a 1-D problem and the algorithms are initialized with 12 data points, while the remaining 102 are used for heldout data. Each GP expert in the GPOEBO algorithm is initialized with 6 data points. We set the γ and β values to 0.1 for HAEI and ANPEI acquisition functions. We choose not to include POEBO in the experiment and use GPOEBO as a benchmark algorithm to compare to other algorithms. The performances of different algorithms are compared in Figure 4.9 and Table 4.12.

We can see that the GPOEBO_ANPEI achieves the best performance compared to other heteroscedastic and homoscedastic BO algorithms, showing the lowest function value by the fifth iteration at 5.53. Both GPOEBO_ANPEI and GPOEBO_HAEI consistently outperform the MLHGPBO algorithms (MLHGPBO_ANPEI and MLHGPBO_HAEI). The MLHGPBO_ANPEI and MLHGPBO_HAEI have higher function values across all iterations, ending with 7.10 and 7.24 respectively at the fifth iteration. Despite the fact, that MLHGPBO algorithms do show improvements over iterations, indicating their learning capability, they do not match the efficiency and robustness of GPOEBO algorithms in minimizing phosphorus content.

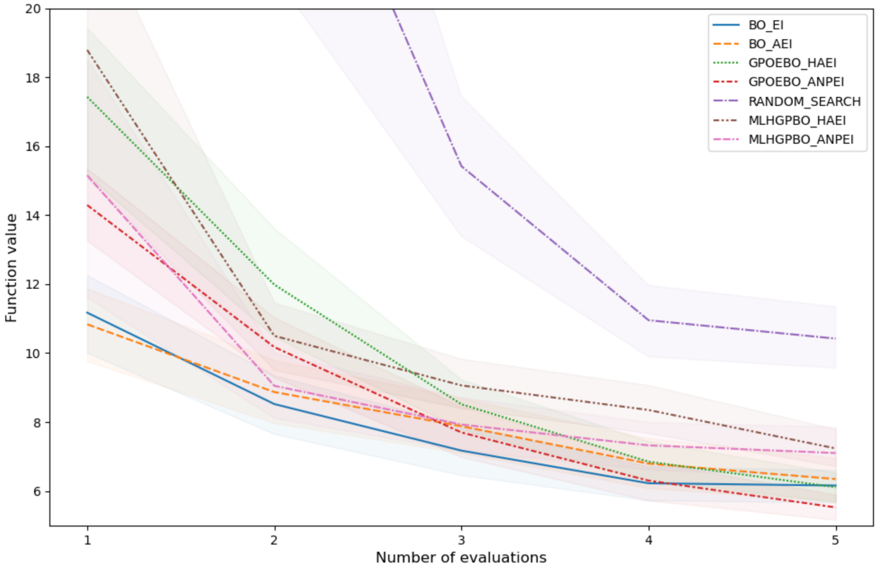


Figure 4.9: The performance results on the soil phosphorus fraction optimization problem.

Moreover, the performance of the standard BO_EI proves to be highly competitive compared to other algorithms. BO_EI consistently outperforms BO_AEI in all iterations. It also significantly outperforms both MLHGPBO algorithms and shows results which are very close to GPOEBO_ANPEI and GPOEBO_HAEI. By the fifth iteration, its function value is 6.16, just slightly higher than the GPOEBO algorithms. It indicates that standard BO_EI can provide robust results, outperforming all other algorithms with the exception of the GPOEBO algorithms. This can also be due to the reason that the regions of low phosphorus fraction coincide with the regions of small heteroscedastic noise as explained by the authors in [21]. In contrast, the RANDOM_SEARCH algorithm, while improving over time, consistently performs worse than the other algorithms in all iterations. Its function value at the fifth iteration is higher than any of the other algorithms.

The improvement in optimization accuracy for GPOEBO_HAEI and GPOEBO_ANPEI algorithms compared to the standard BO on the soil phosphorus fraction optimization problem is provided in Table 4.13. Based on the results, we can see that for the GPOEBO_HAEI, the im-

Table 4.12: Comparative analysis of average optimization performance with corresponding standard deviations (in brackets) over multiple runs for various algorithms aimed at minimizing phosphorus content in soil under heteroscedastic noise conditions.

	BO_AEI	BO_EI	GPOEBO_ANPEI	GPOEBO_HAEI	MLHGPBO_ANPEI	MLHGPBO_HAEI	RANDOM_SEARCH
Iteration							
1	10.83 (7.55)	11.17 (7.94)	14.29 (7.54)	17.43 (15.23)	15.15 (24.76)	18.79 (27.79)	66.42 (94.27)
2	8.87 (6.36)	8.52 (6.30)	10.17 (6.07)	11.99 (10.70)	9.05 (6.52)	10.50 (6.78)	27.91 (44.58)
3	7.88 (5.71)	7.17 (5.13)	7.70 (4.89)	8.52 (5.34)	7.93 (5.50)	9.06 (5.39)	15.42 (14.77)
4	6.80 (4.85)	6.23 (3.56)	6.30 (3.85)	6.85 (4.04)	7.33 (4.93)	8.35 (5.08)	10.95 (7.37)
5	6.35 (4.55)	6.16 (3.53)	**5.53 (2.54)**	6.11 (3.13)	7.10 (4.93)	7.24 (3.91)	10.42 (6.32)

provement in accuracy is 3.78% and for GPOEBO_ANPEI is 12.91% compared to standard BO.

Comparing the GPOEBO_HAEI to MLHGPBO_HAEI we can see that the improvement in accuracy for GPOEBO_HAEI compared to MLHGPBO_HAEI is 15.61%. Similary, comparing the GPOEBO_ANPEI and MLHGPBO_ANPEI, the improvement for GPOEBO_ANPEI is 22.11%.

Table 4.13: Improvement in optimization accuracy compared to BO_EI and MLHGPBO on real-world scientific benchmarks

	Soil Phosphorus Fraction	Molecular Hydration Free Energy
Improvement over BO_EI (%)		
GPOEBO_HAEI	3.78%	10.14%
GPOEBO_ANPEI	12.91%	7.93%
Improvement over MLHGPBO (%)		
GPOEBO_HAEI	15.61%	22.96%
GPOEBO_ANPEI	22.11%	46.17%

In conclusion, it is apparent that GPOEBO algorithms showed the best performance compared to the other algorithms underlining their efficiency and robustness in dealing with heteroscedastic noise.

4.2.3 Molecular Hydration Free Energy Optimization

In the second real-world experiment we used Freesolv dataset and evaluation method proposed by [21]. The goal of this experiment is to perform a retrospective virtual screening experiment to identify the molecules with favorable hydration free energy, which is an important property in determining the binding affinity of a drug candidate. This is a minimization problem with 642 molecules from the FreeSolv dataset.

The lower the hydration free energy of a molecule, the more likely it is to dissolve in water and bind with its target. It is because when a molecule has a lower hydration-free-energy, it interacts more favourably with water molecules, making it easier to dissolve in water [29, 43].

The algorithms are initialized with 64 molecules, while the remaining 578 molecules are used as a heldout set. The dimensionality of chemical fragment features is reduced from 85 to 14 dimensions using principal component analysis while retaining more than 90 % of the variance on average across random trials. The results are shown in Figure 4.10 and Table 4.14.

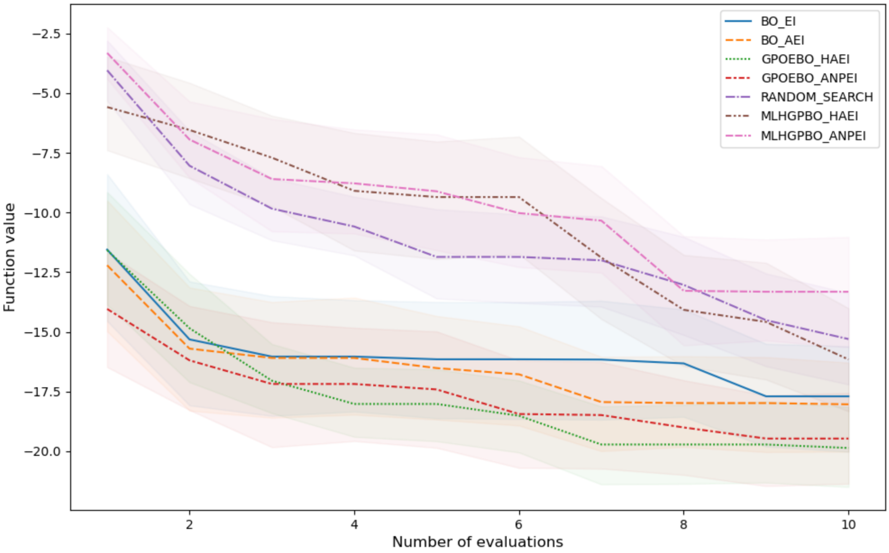


Figure 4.10: The performance results on the FreeSolv hydration free energy optimization problem.

We can see that our proposed GPOEBO_HAEI shows the best performance, achieving the lowest phosphorus value of -19.87. The GPOEBO_-ANPEI achieve the second-best value of -19.47. Both GPOEBO algorithms significantly outperformed other methods, indicating that the GPOEBO algorithms are more effective in minimizing phosphorus content in these specific soil conditions.

Comparatively, the standard BO algorithms, BO_AEI and BO_EI, exhibit similar performance throughout all the iterations, with BO_AEI

Table 4.14: Comparative analysis of average optimization performance with corresponding standard deviations (in brackets) over multiple runs for various algorithms aimed at minimizing hydration free energy of a molecule.

	BO_AEI	BO_EI	GPOEBO_ANPEI	GPOEBO_HAEI	MLHGPBO_ANPEI	MLHGPBO_HAEI	RANDOM_SEARCH
Iteration							
1	-12.21 (9.32)	-11.55 (10.00)	-14.04 (7.96)	-11.58 (7.57)	-3.31 (3.47)	-5.58 (6.50)	-4.04 (4.00)
2	-15.70 (8.34)	-15.32 (8.85)	-16.19 (7.31)	-14.85 (7.58)	-6.94 (5.19)	-6.54 (6.81)	-8.04 (4.98)
3	-16.09 (7.88)	-16.04 (8.08)	-17.18 (8.20)	-17.05 (4.75)	-8.60 (7.84)	-7.70 (5.86)	-9.84 (4.35)
4	-16.09 (7.88)	-16.04 (8.08)	-17.18 (8.20)	-18.02 (4.88)	-8.78 (7.75)	-9.09 (7.97)	-10.59 (4.20)
5	-16.52 (7.47)	-16.15 (7.97)	-17.41 (8.13)	-18.02 (4.88)	-9.11 (7.85)	-9.35 (7.86)	-11.86 (6.32)
6	-16.78 (7.11)	-16.15 (7.97)	-18.44 (7.21)	-18.52 (5.38)	-10.02 (7.76)	-9.35 (7.86)	-11.86 (6.32)
7	-17.94 (6.57)	-16.16 (7.96)	-18.49 (7.15)	-19.72 (5.47)	-10.34 (7.59)	-11.89 (7.95)	-12.00 (6.15)
8	-17.98 (6.52)	-16.32 (7.83)	-19.01 (6.55)	-19.72 (5.47)	-13.28 (7.44)	-14.08 (7.60)	-13.03 (6.66)
9	-17.98 (6.52)	-17.70 (7.10)	-19.47 (6.33)	-19.72 (5.47)	-13.32 (7.39)	-14.58 (7.86)	-14.51 (6.50)
10	-18.04 (6.44)	-17.70 (7.10)	-19.47 (6.33)	**-19.87 (5.36)**	-13.32 (7.39)	-16.16 (7.24)	-15.30 (6.23)

slightly outperforming BO_EI by the 10th iteration (-18.04 vs. -17.70). However, both of these algorithms are outperformed by the GPOEBO based algorithms, indicating the GPOEBO approach provides better optimization in this context.

The improvement in optimization accuracy for our proposed GPOEBO_HAEI and GPOEBO_ANPEI algorithms compared to the standard BO on the FreeSolv hydration free energy optimization problem is provided in Table 4.13. Based on the results, we can see that for the GPOEBO_HAEI algorithm, the improvement in accuracy is 10.14% and for GPOEBO_ANPEI is 7.93% compared to the standard BO.

Comparing GPOEBO_HAEI to MLHGPBO_HAEI we can see that the improvement in mean absolute error is 22.96%. Similary, comparing GPOEBO_ANPEI and MLHGPBO_ANPEI, the improvement for GPOEBO_ANPEI is 46.17%.

In conclusion, we can observe that the standard BO algorithms outperformed the heteroscedastic MLHGPBO models. We see that the MLHGPBO models underperform compared to other algorithms. The main reason for this is that the noise levels in this problem is lower relative to the magnitude of the hydration free energy, which leads to only very marginal gains in obtaining low noise solutions[21].

4.2.4 Performance Sensitivity Analysis

4.2.4.1 Data Partitioning Strategies for GP Experts

The effectiveness of GPOEBO algorithms can be influenced by various factors, among which the strategy for partitioning the data is particularly critical. In this section we investigate the performance of GPOEBO algorithms with two distinct partitioning strategies: a random allocation strategy and a k-means-based allocation strategy. In the random partitioning strategy, we partition the data $\mathcal{D}_n$ into M subsets, where each expert is allocated a random subset of n_i data points without replacement. This guarantees that each expert receives a unique set of data points, ensuring diversity across experts. The k-means point allocation strategy aims to group data points with similar characteristics, allowing experts to specialize in distinct data patterns. We use k-means algorithm to identify M cluster centers, which equals the number of experts. Then, for each cluster center, we query the BallTree [41] to identify its n_i nearest data points. These points are then assigned to the corresponding i-th expert.

We evaluate these strategies based on the mean and standard deviation of absolute error between the function value at the best found point and the actual function maximum across several benchmark functions and multiple runs.

The results presented in Table 4.15 show the effectiveness of different partitioning strategies. In particular, the random allocation strategy consistently outperforms the k-means-based strategy across various test functions. For instance, in the case of the Branin, GoldsteinPrice, and Rosenbrock functions, the strategies GPOEBO_ANPEI_RANDOM and GPOEBO_HAEI_RANDOM achieve the smallest mean errors, which are 0.0332, 0.8265, and 0.0039 respectively.

In contrast, the data allocation strategy using the k-means allocation (GPOEBO_HAEI_KMEANS and GPOEBO_ANPEI_KMEANS) shows larger mean absolute error. This is more visible in the GoldsteinPrice and Branin functions, where the mean errors for the k-means strategy are noticeably larger than those for the random allocation strategies.

Table 4.15: Optimization performance on heteroscedastic synthetic benchmark functions with different point allocation strategies.

Function	Branin	GoldsteinPrice	Hartmann4D	Rosenbrock	Hartmann6D	Sphere
GPOEBO_HAEI_RANDOM	0.0333 (0.0273)	**0.8265 (0.5558)**	0.1759 (0.1517)	**0.0039 (0.0034)**	0.2932 (0.2144)	0.0156 (0.0063)
GPOEBO_ANPEI_RANDOM	**0.0332 (0.0262)**	0.8371 (0.5237)	**0.1630 (0.1464)**	0.0057 (0.0071)	0.3109 (0.2035)	0.0171 (0.0066)
GPOEBO_HAEI_KMEANS	0.0453 (0.0367)	1.0991 (0.5513)	0.1841 (0.1479)	0.0135 (0.0146)	**0.2573 (0.1114)**	0.0179 (0.0067)
GPOEBO_ANPEI_KMEANS	0.0455 (0.0343)	1.0365 (0.5481)	0.2071 (0.1851)	0.0091 (0.0092)	0.2847 (0.1576)	**0.0151 (0.0077)**

Nonetheless, the random allocation strategy does not always shows the best performance. For Hartmann6D and Sphere functions the k-means strategy show the smallest absolute mean errors. Despite this, the overall performance of the random allocation strategy across all test functions suggests its promise as a robust and effective approach for partitioning data for GPOEBO algorithms.

4.2.4.2 Number of data points per GP expert

The optimization performance of the GP expert models tends to vary depending on the number of points assigned per expert. We evaluate the performance of POEBO and GPOEBO algorithms using $4 \times D$, $3 \times D$, $2 \times D$ and $1 \times D$ number of points per experts. The effect of the number of data points per expert on optimization performance are shown in Table 4.16 and Figure 4.11.

We can see that performance tends to vary between different functions, but the overall performance improves (absolute error gets closer to zero) as the number of points per expert increases. The GPOEBO methods consistently achieve a lower average mean error across most functions, highlighting a better optimization performance over POEBO methods.

When analysing the results per individual optimization function, we can see that for the functions like Branin, Hartmann4D, Rosenbrock, and Hartmann6D, as the number of data points per expert increased from $1 \times D$ to $4 \times D$, both GPOEBO_HAEI and GPOEBO_ANPEI showed a reduction in average mean absolute error. The GPOEBO_HAEI algorithm showed a large decrease in mean absolute error for Hartmann4D function from 0.4155 at $1 \times D$ to 0.1759 at $4 \times D$.

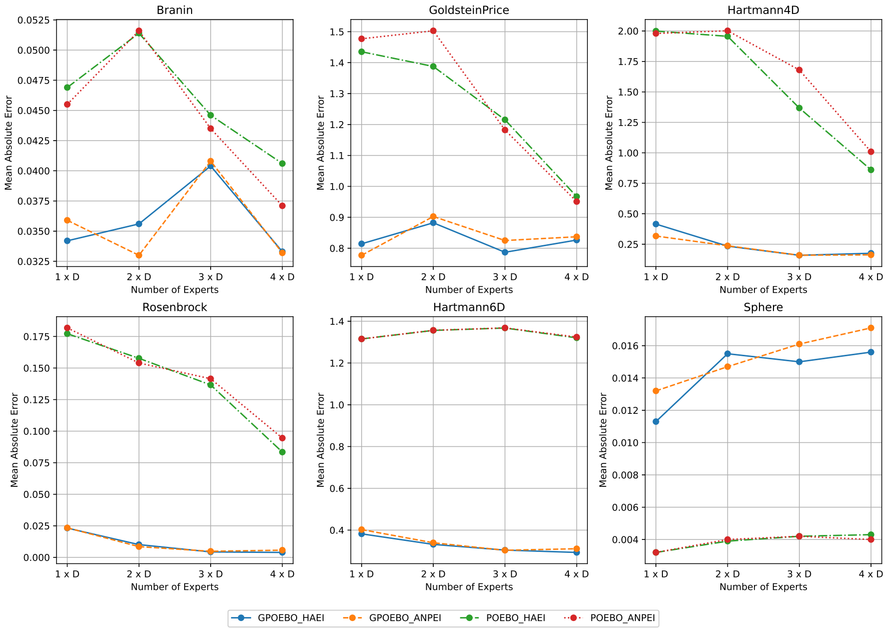


Figure 4.11: The effect of the number of data points per expert on the optimization performance of heteroscedastic synthetic benchmark functions.

However, the results for the GoldsteinPrice and Sphere functions showed different results. As the number of data points per expert increased, both GPOEBO algorithms showcased a decrease in their performance. For the GoldsteinPrice function, the GPOEBO_ANPEI mean absolute error increased from 0.7772 at $1 \times D$ to 0.8371 at $4 \times D$.

Based on the results, we can see that GPOEBO algorithms, regardless of whether they use the HAEI or ANPEI acquisition function, generally outperform POEBO algorithms for most heteroscedastic synthetic benchmark functions, especially when more data points are allocated to each GP expert. The only exception is the Sphere function, where POEBO algorithms show marginally lower mean absolute errors than GPOEBO algorithms.

Table 4.16: Optimization performance for heteroscedastic synthetic benchmark functions with varying numbers of points allocated to each GP expert.

Number of Points	**Method**	**Functions**					
		Branin	**GoldsteinPrice**	**Hartmann4D**	**Rosenbrock**	**Hartmann6D**	**Sphere**
4 x D	GPOEBO_HAEI	0.0333	0.8265	0.1759	0.0039	0.2932	0.0156
	GPOEBO_ANPEI	0.0332	0.8371	0.1630	0.0057	0.3109	0.0171
	POEBO_HAEI	0.0406	0.9671	0.8603	0.0834	1.3205	0.0043
	POEBO_ANPEI	0.0371	0.9509	1.0092	0.0945	1.3242	0.0040
3 x D	GPOEBO_HAEI	0.0404	0.7870	0.1593	0.0043	0.3044	0.0150
	GPOEBO_ANPEI	0.0408	0.8250	0.1597	0.0048	0.3035	0.0161
	POEBO_HAEI	0.0446	1.2153	1.3678	0.1366	1.3675	0.0042
	POEBO_ANPEI	0.0435	1.1823	1.6800	0.1415	1.3681	0.0042
2 x D	GPOEBO_HAEI	0.0356	0.8823	0.2343	0.0101	0.3317	0.0155
	GPOEBO_ANPEI	0.0330	0.9027	0.2374	0.0085	0.3395	0.0147
	POEBO_HAEI	0.0514	1.3877	1.9561	0.1577	1.3561	0.0039
	POEBO_ANPEI	0.0516	1.5031	2.0017	0.1540	1.3566	0.0040
1 x D	GPOEBO_HAEI	0.0342	0.8144	0.4155	0.0232	0.3822	0.0113
	GPOEBO_ANPEI	0.0359	0.7772	0.3173	0.0234	0.4026	0.0132
	POEBO_HAEI	0.0469	1.4352	1.9990	0.1772	1.3153	0.0032
	POEBO_ANPEI	0.0455	1.4771	1.9795	0.1818	1.3152	0.0032

4.2.5 Conclusions

In this chapter, we empirically showed the efficiency and scalability of the generalized product of experts based Bayesian optimization on standard global optimization benchmark functions and real-life problems ranging from 550 to 10K observations. Moreover, we saw that the accuracy could be improved by combining the generalized product of experts model with search space reduction methods. The proposed gPoETRBO algorithm, which combines the trust region method with the generalized product of experts model achieves the best performance compared to the other Gaussian process experts based Bayesian optimization algorithms and matches the performance of the state-of-the-art TuRBO algorithm with a significant speedup in computational time and using only moderate computing hardware.

We compared the performance of gPoEBO with two different point allocation strategies for Gaussian process experts. Based on our results, we found that there is no significant difference between these two point allocation strategies for Gaussian process experts. Additional experiments show that gPoETRBO outperforms other algorithms using limited time budget for optimization. Moreover, our analysis shows that there is a trade-off between the number of points assigned per Gaussian process expert and computing time.

Finally, our experiments on synthetic and real-world optimization problems show that, on average, the proposed heteroscedastic generalized product of experts based Bayesian optimization algorithms are competitive on many problems compared to other heteroscedastic and homoscedastic Bayesian optimization algorithms. Moreover, we can see that our proposed algorithms are more robust to the magnitude of heteroscedastic noise compared to MLHGPBO algorithms. However, our proposed algorithms have some limitations as their performance is sensitive to many points allocated to each Gaussian process expert, and their performance can degrade significantly if the number is set incorrectly.

GENERAL CONCLUSIONS

1. To address the scalability issues of standard Bayesian optimization, we proposed two new algorithms gPoEBO and gPoETRBO for global Bayesian optimization with large number of observations:

   1.1 For the gPoEBO algorithm, the improvement in accuracy ranged from -2.87% to 6.45% compared to standard Bayesian optimization on 20D benchmark functions. The improvement in runtimes ranged from 106.10% to 209.58%, showing the significant improvement in runtime while maintaining competitive accuracy.

   1.2 For the gPoETRBO algorithm, the improvement in accuracies on 20D ranged from 70.51% to 98.47% and from 75.56% to 99.61% on 50D benchmark functions compared to standard Bayesian optimization. The improvement in runtimes ranged from 68.56% to 152.39% for 20D benchmark functions and from 667.44% to 1118.68% for 50D benchmark functions, indicating substantial improvement in optimization accuracy and efficiency over the standard Bayesian optimization.

   1.3 The trust region method based gPoETRBO matches or achieves better optimization accuracy than the state-of-the-art TuRBO algorithm. Runtimes for the gPoETRBO compared to the TuRBO are between 11 to 12 times shorter on 50D benchmark functions.

   1.4 Experiments on real-life problems with our proposed the gPoETRBO algorithm, achieved similar accuracy on the 12D Lunar Landing problem with 1K observations and the 14D Robot Pushing problem with 2K observations compared to the standard Bayesian optimization with 8 and 16 times shorter runtimes. Additionally, we were able to optimize the 60D Rover Trajectory problem with 10K observations using gPoETRBO and aciehve 904% improvement in accuracy compared to standard Bayesian optimization. Moreover, gPoETRBO matched the performance of TuRBO with up to 6 times improvement in runtime on the 60D Rover Trajectory problem.

1.5 Our ablation study shows that there is a trade-off between the number of points assigned per Gaussian process expert and computing time for gPoETRBO on 20D and 50D benchmark functions and increasing the size of data points per expert results in an improvement of accuracy, but also an increased computational time.

1.6 The experiments with time restricted budget on 20D and 50D synthetic benchmark functions with time budgets of 5 and 15 minutes, showed that gPoETRBO achieved similar or better performance on 20D functions and outperformed other algorithms on all 50D benchmark functions.

2. We have presented an approach for performing heteroscedastic Bayesian optimization using the generalized product of experts with excising heteroscedastic acquisition functions:

2.1 The results showed that GPOEBO_HAEI compared to standard Bayesian optimization algorithm had from 4.35% to 33.90% lower mean absolute error on all synthetic benchmark functions.

2.2 The GPOEBO_HAEI compared to the state-of-the-art MLHGPBO_HAEI had from 22.71% to 81.78% lower mean absolute error on all synthetic benchmark functions.

2.3 The results on real-life scientific problems showed that our proposed GPOEBO_HAEI and GPOEBO_ANPEI algorithms on average achieved between 15.61% to 46.17% lower mean absolute error compared to MLHGPBO algorithm.

2.4 The sensitivity analysis shows that random allocation strategy for partitioning the data for Gaussian process experts, consistently outperforms the k-means based strategy across all synthetic benchmark functions.

2.5 Finally, we showed that optimization performance for our proposed algorithms is sensitive to the number of points allocated to each Gaussian process experts and its performance can degrade significantly if the number is set incorrectly. Our experiments showed that setting $4 \times D$ number of points per expert for each problem was the optimal number.

REFERENCES

[1] J.-A. M. Assael, Z. Wang, B. Shahriari, and N. de Freitas. Heteroscedastic treed Bayesian optimisation. *arXiv preprint arXiv:1410.7172*, 2014.

[2] T. Bailey, S. Julier, and G. Agamennoni. On conservative fusion of information with unknown non-Gaussian dependence. *15th International Conference on Information Fusion, FUSION 2012*, pages 1876–1883, 2012.

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2), 2012.

[4] J. Bibby. Axiomatisations of the average and a further generalisation of monotonic sequences. *Glasgow Mathematical Journal*, 15(1):63–65, 1974.

[5] M. Binois, D. Ginsbourger, and O. Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization*, 76(1):69–90, Jan 2020.

[6] C. M. Bishop and N. M. Nasrabadi. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[8] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[9] T. D. Bui, J. Yan, and R. E. Turner. A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *The Journal of Machine Learning Research*, 18 (1):3649–3720, 2017.

[10] R. Calandra. *Bayesian Modeling for Optimization and Control in Robotics*. PhD thesis, Technische Universität Darmstadt, 2017.

[11] Y. Cao. *Scaling Gaussian Processes*. PhD thesis, University of Toronto (Canada), 2018.

[12] Y. Cao and D. J. Fleet. Generalized product of experts for automatic and principled fusion of Gaussian process predictions. *Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS.*

arXiv:1410.7827, 2014.

[13] Y. Cao and D. J. Fleet. Transductive log opinion pool of Gaussian process experts. *Workshop on Nonparametric Methods for Large Scale Representation Learning at NIPS. arXiv:1511.07551*, 2015.

[14] K. Chalupka, C. K. Williams, and I. Murray. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14:333–350, 2013.

[15] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, USA, 2009.

[16] M. P. Deisenroth and J. W. Ng. Distributed Gaussian processes. In *32nd International Conference on Machine Learning, ICML 2015*, volume 2, 2015.

[17] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, pages 5496–5507, 2019.

[18] P. I. Frazier. Bayesian Optimization. In *Recent Advances in Optimization and Modeling of Contemporary Problems*, pages 255–278. 2018.

[19] J. R. Gardner, C. Guo, K. Q. Weinberger, R. Garnett, and R. Grosse. Discovering and exploiting additive structure for Bayesian optimization. volume 54 of *Proceedings of Machine Learning Research*, pages 1311–1319, 2017.

[20] P. Goldberg, C. Williams, and C. Bishop. Regression with input-dependent noise: A Gaussian process treatment. *Advances in Neural Information Processing Systems*, 10, 1997.

[21] R.-R. Griffiths, A. A. Aldrick, M. Garcia-Ortegon, V. Lalchand, et al. Achieving robustness to aleatoric uncertainty with heteroscedastic Bayesian optimisation. *Machine Learning: Science and Technology*, 3 (1):015004, 2021.

[22] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[23] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems*, 27, 2014.

[24] T. Heskes. Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems*, volume 10, page 266–272, 1998.

[25] G. E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999.

[26] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8), 2002.

[27] R. Horst and P. M. Pardalos. *Handbook of Global Optimization*, volume 2. Springer Science & Business Media, 2013.

[28] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, 2006.

[29] G. Hummer, L. R. Pratt, and A. E. Garcia. Hydration free energy of water. *The Journal of Physical Chemistry*, 99(38):14188–14194, 1995.

[30] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[31] F. Jimenez and M. Katzfuss. Scalable Bayesian optimization using vecchia approximations of Gaussian processes. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 1492–1512. PMLR, 25–27 Apr 2023.

[32] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79:157–181, 1993.

[33] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[34] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional Bayesian Optimisation and bandits via additive models. In *32nd International Conference on Machine Learning*, volume 37, page 295–304, 2015.

[35] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th International Conference on Machine Learning*, pages 393–400, 2007.

[36] H. J. Kushner. A new method of locating the maximum point

of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 03 1964. ISSN 0021-9223. doi: 10.1115/1.3653121.

[37] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

[38] H. Liu, J. Cai, Y. Wang, and Y. S. Ong. Generalized robust Bayesian committee machine for large-scale Gaussian process regression. In *35th International Conference on Machine Learning, ICML 2018*, volume 80, pages 3131–3140, 2018.

[39] H. Liu, J. Cai, Y.-S. Ong, and Y. Wang. Understanding and comparing scalable Gaussian process regression for big data. *Knowledge-Based Systems*, 164:324–335, 2019.

[40] H. Liu, Y.-S. Ong, and J. Cai. Large-scale heteroscedastic regression via Gaussian process. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):708–721, 2020.

[41] T. Liu, A. W. Moore, A. Gray, and C. Cardie. New algorithms for efficient high-dimensional nonparametric classification. *Journal of Machine Learning Research*, 7(6), 2006.

[42] M. Lázaro-Gredilla and M. Titsias. Variational heteroscedastic Gaussian process regression. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 841–848, 01 2011.

[43] D. L. Mobley and J. P. Guthrie. Freesolv: a database of experimental and calculated hydration free energies, with input files. *Journal of Computer-Aided Molecular Design*, 28:711–720, 2014.

[44] J. Mockus. *Bayesian Approach to Global Optimization: Theory and Applications*. Mathematics and its Applications. Springer Netherlands, 2011. ISBN 9789401068987.

[45] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. In *Towards Global Optimisation*, volume 2, pages 117–129, 1978.

[46] A. Munteanu, A. Nayebi, and M. Poloczek. A framework for Bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4752–4761, 2019.

[47] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, USA, 2012.

[48] R. M. Neal and R. M. Neal. Priors for infinite networks. *Bayesian Learning for Neural Networks*, pages 29–53, 1996.

[49] C. Y. Oh, E. Gavves, and M. Welling. BOCK: Bayesian Optimization with Cylindrical Kernels. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3868–3877, 2018.

[50] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55(1):2–13, 2013.

[51] V. Picheny, T. Wagner, and D. Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.

[52] J. D. Pintér. *Global Optimization: Scientific and Engineering Case Studies*, volume 85. Springer Science & Business Media, 2006.

[53] J. Quinonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[54] C. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*, 14, 2001.

[55] R. G. Regis. Trust regions in kriging-based optimization with expected improvement. *Engineering Optimization*, 48(6):1037–1059, 2016.

[56] R. G. Regis and C. A. Shoemaker. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37(1):113–135, 2007.

[57] B. Ru, A. Alvi, V. Nguyen, M. A. Osborne, and S. Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, pages 8276–8285, 2020.

[58] V. Šaltenis. On a method of multi-extremal optimization. *Automatics and Computers (Avtomatika i Vychislitelnayya Tekchnika)*, 3:33–38, 1971.

[59] B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[60] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[61] O. V. Shylo, T. Middelkoop, and P. M. Pardalos. Restart strategies in optimization: parallel and serial cases. *Parallel Computing*, 37(1): 60–68, 2011.

[62] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18, 2006.

[63] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[64] J. Snoek, K. Swersky, R. Zemel, and R. Adams. Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pages 1674–1682. PMLR, 2014.

[65] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. ISSN 0041-5553.

[66] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 1015–1022, 2010.

[67] K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. *Advances in Neural Information Processing Systems*, 26, 2013.

[68] S. Tautvaišas and J. Žilinskas. Heteroscedastic bayesian optimization using generalized product of experts. *Journal of Global Optimization*, pages 1–21, 2023.

[69] S. Tautvaišas and J. Žilinskas. Scalable bayesian optimization with generalized product of experts. *Journal of Global Optimization*, 88(3): 777–802, 2024.

[70] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[71] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, USA, 16–18 Apr 2009. PMLR.

[72] M. Trapp, R. Peharz, F. Pernkopf, and C. E. Rasmussen. Deep

structured mixtures of gaussian processes. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2251–2261, 26–28 Aug 2020.

[73] V. Tresp. A Bayesian committee machine. *Neural Computation*, 12 (11):2719–2741, 2000.

[74] V. Tresp. Mixtures of Gaussian processes. *Advances in Neural Information Processing Systems*, 13, 2000.

[75] X. Wan, V. Nguyen, H. Ha, B. Ru, C. Lu, and M. A. Osborne. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. In *38th International Conference on Machine Learning (ICML 2021)*, pages 10663–10674, 2021.

[76] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact Gaussian processes on a million data points. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[77] L. Wang, R. Fonseca, and Y. Tian. Learning search space partition for black-box optimization using Monte Carlo Tree Search. In *Advances in Neural Information Processing Systems*, volume 33, pages 19511–19522, 2020.

[78] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

[79] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, volume 84, pages 745–754, 2018.

[80] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

[81] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, USA, 2006.

[82] S. Wright, J. Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[83] Y.-x. Yuan. A review of trust region algorithms for optimization. In *ICIAM99: Proceedings of the Fourth International Congress on Industrial*

and Applied Mathematics Edinburgh, volume 99, pages 271–282, 2000.

[84] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012.

SUMMARY IN LITHUANIAN

Šiame darbe nagrinėjamas globalusis optimizavimas, kuris naudojamas tikslo funkcijos maksimumo (minimumo) taškui visoje funkcijos apibrėžimo srityje rasti. Šiuo metu vienas iš plačiai paplitusių algoritmų tokio tipo uždaviniams spręsti yra Bajeso optimizavimo (toliau – BO) algoritmas. Jis naudojamas, kai tikslo funkcijos analizinė išraiška nėra žinoma (tokios funkcijos kitaip dar vadinamos juodosios dėžės funkcijomis), o funkcijos maksimumo (minimumo) taško radimas trunka ilgai ar brangiai kainuoja.

Statistinis modelis (dažniausiai Gauso procesas), kuriuo pagrįstas BO algoritmas, aproksimuoja tikslo funkciją, remdamasis žinomomis šios funkcijos reikšmėmis. Šis modelis leidžia prognozuoti tikslo funkcijos reikšmes dar neištirtuose apibrėžimo srities taškuose bei įvertinti šių prognozių patikimumo matą. Naudojant šį modelį, BO algoritmas įvertina tikimybę aptikti didesnę (mažesnę) tikslo funkcijos reikšmę lyginant su turimais duomenimis. Tai leidžia BO algoritmui rasti maksimalų (minimalų) tikslo funkcijos tašką per mažiausią funkcijos įverčių skaičių. Dėl šių savybių BO dažnai naudojamas įvairiose srityse, pavyzdžiui, investavimo, inžinerijos ar moksliniuose tyrimuose.

Tyrimo sritis ir problemos aktualumas

BO algoritmas veikia ypač gerai, kai turimų duomenų kiekis yra mažas ar optimizavimui reikalingas tik nedidelis tikslo funkcijos įverčių skaičius.

Didėjant duomenų kiekiui, BO algoritmas tampa mažiau efektyvus ir reikalauja daug skaičiavimo išteklių. Šio algoritmo efektyvumo mažėjimą dažnai lemia Gauso proceso, kuris yra esminė BO algoritmo dalis, apskaičiavimo laiko sudėtingumas.

Net nedidelis duomenų kiekio ar reikalingų tikslo funkcijos įverčių skaičiaus padidėjimas gali reikšmingai pailginti BO algoritmo skaičiavimo laiką, nes Gauso procesas naudoja kovariacijos matricą, kuri aprašo turimų duomenų sąryšį. Prognozėms, kurios vėliau naudojamos atnaujinti modelį, apskaičiuoti reikia rasti šios matricos atvirkštinę matricą. Kadangi prognozės apskaičiuojamos iteracijų būdu, atvirkštinės matricos apskaičiavimo greitis išauga labai greitai. Dėl šios priežasties BO algoritmas paprastai apsiriboja tik iki 2000 funkcijos įverčių [79]. Technologijoms tobulėjant ir didėjant lygiagrečių skaičiavimo išteklių prieinamumui, galima atlikti vis didesnį tikslo funkcijos įverčių skaičių. Tai skatina tobulinti ir kurti algoritmus, kurie galėtų būti vykdomi lygiagrečiai.

Dar vienas BO algoritmo trūkumas yra tas, kad standartiniame BO algoritme laikomasi prielaidos, jog triukšmas visoje tikslo funkcijos apibrėžimo srityje yra pastovus (toks triukšmas kitaip dar vadinamas homoskedastišku triukšmu). Praktiškai ši prielaida ne visada tenkinama. Gali būti, kad triukšmas keičiasi priklausomai nuo kintamojo padėties tikslo funkcijos apibrėžimo srityje (toks triukšmas kitaip dar vadinamas heteroskedastišku triukšmu). Dėl šios priežasties BO algoritmo rezultatai gali būti klaidingi. Todėl reikalingos BO algoritmo modifikacijos, kurios galėtų būti pritaikomos optimizavimo problemoms su heteroskedastišku triukšmu.

Tyrimo objektas

Šios disertacijos darbo objektas yra Bajeso optimizavimo algoritmai, skirti didelio masto globaliojo optimizavimo problemoms spręsti.

Tyrimo tikslas ir uždaviniai

Šio darbo objektas yra Bajeso optimizavimo algoritmai. Pagrindinis tikslas – sukurti efektyvias Bajeso optimizavimo algoritmo modifikacijas, kurios būtų pritaikomos skirtingiems optimizavimo uždaviniams spręsti. Šiam tikslui pasiekti numatyti konkretūs uždaviniai:

1. Sukurti naujus arba modifikuoti esamus Bajeso optimizavimo algoritmus, siekiant pagerinti jų pritaikomumą didelio masto optimizavimo uždaviniams ir pagerinti jų efektyvumą;
2. Parodyti siūlomų algoritmų pritaikomumą optimizavimo uždaviniams su kintamu triukšmo lygiu;
3. Palyginti siūlomų algoritmų rezultatus su kitais optimizavimo algoritmais.

Mokslinis naujumas ir rezultatai

Dauguma iki šiol sukurtų BO algoritmo modifikacijų yra nepritaikomos didelio masto problemoms, kai pradinių duomenų kiekis yra didelis arba tikslo funkcijos optimizavimui yra reikalingas didelis įverčių skaičius. Kai duomenų kiekis yra didesnis už 2000, BO algoritmo veikimas gali tapti sudėtingas, o vykdymas užtrukti labai ilgai. Norint apdoroti tokį duomenų kiekį, reikalingi specialūs skaičiavimus spartinantys įrenginiai, kurie yra brangūs ir prieinami ne visiems. Tai riboja BO algoritmų pritaikomumą ir platesnį naudojimą.

Šioje disertacijoje siūlomos dvi naujos BO algoritmo modifikacijos, paremtos apibendrintu ekspertų sandaugos (toliau – gPoE) modeliu. Šis modelis leidžia pritaikyti BO algoritmą problemoms, kurioms reikalingas didelis funkcijos įverčių skaičius, be poreikio turėti prieigą prie specializuotų skaičiavimo įrenginių. Eksperimentiškai parodėme šių algoritmų veiksmingumą ir pritaikomumą didelio masto problemoms lygindami su esamais algoritmais. Taip pat teoriškai įrodėme, kad mūsų siūloma BO algoritmo modifikacija, naudojanti patikimos srities metodą, konverguoja į funkcijos globalų maksimumą.

Be to, šiame darbe parodome, kad mūsų siūlomas gPoEBO gali būti sėkmingai pritaikytas globaliojo optimizavimo problemoms su heteroskedastišku triukšmu. Sukūrėme du naujus heteroskedastiškus gPoE pagrįstus BO (toliau – GPOEBO) algoritmus. Šie algoritmai pagrįsti gPoE modelio ir heteroskedastiškos įverčio (angl. acquisition) funkcijos deriniu, naudojant kiekvieno atskiro GP eksperto išmoktus individualius triukšmus modeliuoti funkcijoms su heteroskedastišku triukšmu. Atlikti eksperimentai parodė, kad mūsų sukurti algoritmai pranoko kitus heteroskedastiškus ir homoskedastiškus BO algoritmus.

Ginamieji teiginiai

Šios disertacijos ginamieji teiginiai:

1. Naujai siūlomas gPoEBO algoritmas yra efektyvesnis už standartinį BO algoritmą skaičiavimo laiko atžvilgiu.
2. Siūlomas gPoETRBO algoritmas pasiekia geriausią tikslumą lyginant su kitais Gauso proceso ekspertų modelio pagrindu veikiančiais BO algoritmais ir atitinka pažangiausio BO algoritmo optimizavimo tikslumą, bei reikšmingai sutrumpiną optimizavimo vykdymo laiką, naudojant standartinę kompiuterinę skaičiavimo įrangą.
3. Sukurti du heteroskedastiški Bajeso optimizavimo algoritmai parodė geriausią optimizavimo tikslumą lyginant su kitais heteroskedastiškais ir homoskedastiškais BO algoritmais.

Darbo rezultatų aprobavimas

Disertacijos rezultatai paskelbti tarptautiniuose mokslo žurnaluose, turinčiuose citavimo indeksą „Clarivate Analytics Web of Science“ (CA WoS) duomenų bazėje:

1. Tautvaišas, S. and Žilinskas, J., 2024. Scalable Bayesian optimization with generalized product of experts. Journal of Global Optimization, 88(3), pp.777-802.
2. Tautvaišas, S. and Žilinskas, J., 2023. Heteroscedastic Bayesian optimization using generalized product of experts. Journal of Global Optimization, pp.1-21.

Disertacijos struktūra ir apimtis

Ši disertacija suskirstyta į keturis pagrindinius skyrius, o po jų pristatomos apibendrintos išvados ir literatūros sąrašas. Pirmajame skyriuje pateikiamas įvadas į tyrimo tematiką ir apibrėžiama disertacijos struktūra. Antrajame skyriuje pateikiama globaliojo Bajeso optimizavimo apžvalga ir supažindinama su pagrindiniais Bajeso optimizavimo komponentais. Trečiajame skyriuje pristatomos siūlomos Bajeso algoritmų modifikacijos. Ketvirtajame skyriuje aprašomi empiriniai eksperimentai, gauti rezultatai ir rezultatų analizė. Pagrindiniai eksperimentinio tyrimo rezultatai ir įžvalgos pateikiamos apibendrintų išvadų skyriuje. Disertacija susideda iš 133 puslapių, kuriuose yra 19 paveikslėlių, 16 lentelių ir penki algoritmai.

S1 BAJESO OPTIMIZAVIMO ALGORITMŲ APŽVALGA

Šioje dalyje pristatomas Bajeso optimizavimo algoritmas, jo pagrindiniai komponentai, privalumai ir trūkumai.

S1.1 Bajeso optimizavimo algoritmas

BO algoritmas yra statistiniu modeliu pagrįstas optimizavimo algoritmas, kuris naudojamas funkcijų optimizavimui, kurių reikšmių apskaičiavimas yra brangus ar užtrunkantis ilgą laiką [8, 18, 45]. Šis algoritmas naudoja statistinį modelį aproksimuoti tikslo funkcijos reikšmėms, pasitelkiant jau turimus funkcijos taškus. Statistinis modelis leidžia prognozuoti funkcijos reikšmes dar neištirtuose apibrėžimo srities taškuose bei parodo prognozės patikimumo matą.

Remdamasis statistiniu modeliu ir įverčio (angl. acquisition) funkcija, BO algoritmas kiekvienam apibrėžimo srities taškui priskiria kiekybinį įvertį, kuris parodo tinkamumą jame skaičiuoti tikslinės funkcijos reikšmę. Norint surasti naują tašką, kuris turėtų didžiausią potencialą būti tikslo funkcijos maksimumas, ieškomas didžiausią kiekybinį įverti turintis taškas. Kiekvieną kartą apskaičiavus naują funkcijos reikšmę, statistinis modelis atnaujinamas, kad atspindėtų naujausią turimą informaciją apie tikslo funkciją. Procesas kartojamas, kol pasiekiamas nustatytas maksimalus funkcijos įverčių skaičius.

Gauso procesas, (toliau – GP) populiariausias statistinis modelis, yra naudojamas BO algoritme tikslo funkcijai f modeliuoti [81]. Jis apibrėžiamas kaip atsitiktinių kartu pagal Gauso skirstinį pasiskirsčiusių dydžių rinkinys. Statistinis modelis gali būti interpretuojamas kaip funkcijos f aproksimacija visoje apibrėžimo srityje remiantis jau žinomomis tikslo funkcijos reikšmėmis. Gauso procesas visiškai aprašomas vidurkio ir kovariacijos funkcijomis (pastaroji kitaip dar vadinama branduolio funkcija). Gauso proceso $f(x)$ vidurkio $\mu(x)$ ir kovariacijos funkcijos $k(x, x')$ apibrėžiamos taip:

$$\mu(x) = \mathbb{E}[f(x)],$$

$$k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))],$$

o Gauso procesas išreiškiamas kaip

$$f(x) \sim GP(\mu(x), k(x, x')).$$

Pagrindinė GP sudedamoji dalis – kovariacijos funkcija, nustatanti ryšį tarp duomenų ir apibrėžianti GP savybes. Kovariacijos funkcijos atlieka lemiamą vaidmenį nustatant modeliuojamos tikslo funkcijos savybes.

Kita svarbi BO algoritmo dalis yra įverčio funkcija. Ji priskiria kiekybinį įvertį kiekvienam apibrėžimo srities taškui x, kuris parodo tinkamumą skaičiuoti funkcijos reikšmę. Renkantis naują tašką tikslo funkcijos reikšmei apskaičiuoti, siekiama rasti didžiausią įvertį turintį tašką apibrėžimo srityje [44].

Populiariausia įverčio funkcija yra maksimalaus tikėtino pagerinimo funkcija (angl. expected improvement (EI)) [33, 44, 45]. Ši funkcija matuoja tikėtiną pagerėjimą lyginant su jau apskaičiuotomis tikslo funkcijos reikšmėmis. Naudojantis šia funkcija, galime apskaičiuoti taško x įvertį pagal toliau pateiktą formulę:

$$\alpha_{\text{EI}}(x) = \sigma(x)\left[Z\Phi(Z) + \phi(Z)\right], \tag{S4.1}$$

kur $Z = \frac{\mu(x) - f(x_{\text{geriausias}}) - \xi}{\sigma(x)}$, Φ yra standartinio normaliojo dydžio pasiskirstymo funkcija, o ϕ yra standartinio normaliojo skirstinio tankio funkcija.

S1.2 Bajeso optimizavimo algoritmo trūkumai

Nepaisant savo privalumų, BO algoritmas susiduria su keletu iššūkių ir ribotumų, kurie gali paveikti algoritmo efektyvumą ir pritaikomumą. Vieni iš pagrindinių trūkumų yra:

Skaičiavimo sudėtingumas. Didėjant duomenų kiekiui ar funkcijos įverčių skaičiui, BO algoritmas reikalauja vis daugiau skaičiavimo išteklių. Viena iš to priežasčių – Gauso proceso modelis, kuriuo remiasi BO algoritmas. Gauso proceso modelis naudoja kovariacijos matricą, kuri aprašo sąryšį tarp skirtingų funkcijos reikšmių. Norint atlikti prognozes ir atnaujinti modelį, reikia rasti šios matricos atvirkštinę matricą. Atvirkštinės matricos skaičiavimo sudėtingumas yra $O(n^3)$, kur n yra duomenų kiekis [81]. Pasiūlyti įvairūs algoritmai, leidžiantys supaprastinti šį apskaičiavimą – standartinio Gauso proceso modelio pakeitimas retu Gauso procesu (angl. sparse Gaussian Process) [53] ar duomenų kiekio sumažinimas aproksimuojant duomenis lokaliais duomenų taškais [9]. Nepaisant to, šie algoritmai turi trūkumų ir didėjant duomenų kiekiui, jų tikslumas labai suprastėja.

Heteroskedastiškas duomenų triukšmas. Standartiniame Gauso proceso modelyje laikomasi prielaidos, kad duomenų triukšmas yra homoskedastiškas. Tačiau kai duomenų triukšmas yra kintamas ir priklauso nuo padėties apibrėžimo srityje (t. y. heteroskedastiškas), BO algoritmo rezultatai gali būti klaidingi [37, 51, 63]. Siekiant sukurti BO algoritmus, kurie galėtų palaikyti kintamą triukšmą duomenyse, pasiūlytos naujos įverčio funkcijų modifikacijos [21, 37, 67]. Šios modifikacijos padidina BO algoritmo skaičiavimo sudėtingumą ir didėjant duomenų kiekiui, šie algoritmai tampa nepritaikomi praktiškai.

S1.3 Didelio masto Bajeso optimizavimo algoritmai

BO algoritmas veikia gerai, kai duomenų kiekis yra nedidelis ar optimizavimui reikalingas nedidelis tikslo funkcijos įverčių skaičius. Didėjant duomenų kiekiui, šio algoritmo vykdymo laikas pradeda sparčiai ilgėti. Taip atsitinka todėl, kad Gauso procesu paremtam BO algoritmui reikia apskaičiuoti atvirkštinę kovariacijos matricą su visais turimais duomenimis. Dėl šios priežasties BO paprastai apsiriboja tik keliais tūkstančiais funkcijos įverčių [79]. Atsižvelgiant į didėjantį lygiagrečių skaičiavimo įrenginių prieinamumą, verta pažymėti, kad optimizavimo vykdymo laiką galima sumažinti, jei optimizavimo algoritmas galėtų būti pritaikomas lygiagretiems skaičiavimams.

Pagrindinė sukurtų algoritmų problema yra ta, kad skaičiavimams pagreitinti jie naudoja specialią skaičiavimo įrangą, kuri yra brangi arba neprieinama paprastam vartotojui. Siekiant apdoroti didelį duomenų kiekį, pasiūlyti tokie algoritmai kaip EBO, naudojantis 240 procesorių branduolius [79], ir pažangiausi TuRBO ir LA-MCTS, naudojantys vaizdo apdorojimo procesorius skaičiavimams paspartinti. Šių algoritmų, naudojamų optimizavimo problemoms spręsti su paprasta kompiuterine įranga ar asmeniniu kompiuteriu, turinčiu tik ribotą procesoriaus branduolių skaičių, vykdymo laikas reikšmingai pailgėja ir gali trukti mėnesius ar net metus. Todėl šie algoritmai nėra plačiai taikomi praktiškai.

S1.4 Gauso proceso ekspertų modeliai

Apibendrintas sandaugos ekspertų modelis (angl. generalised Product of Experts (gPoE) – vienas iš Gauso proceso ekspertų modelių tipų, leidžiantis lanksčiai ir efektyviai modeliuoti tikslo funkcijas su dideliu duomenų kiekiu [12, 13]. Šis sandaugos ekspertų modelis yra sudarytas iš daugelio mažesnių Gauso proceso modelių, kurie vadinami eksperto modeliais. Kiekvieno eksperto modelis gali turėti skirtingus parametrus, nepriklausomus nuo kito eksperto modelio parametrų, o tai leidžia panaudoti lygiagrečiuosius skaičiavimus ekspertų mokymui [11].

Apibendrintas sandaugos ekspertų modelis sujungia kiekvieno atskiro GP eksperto prognozę į galutinį bendrą modelį [11]

$$p_{\mathcal{A}}\left(f_*|x_*, \mathcal{D}\right) = \prod_{i=1}^{M} p_i^{\alpha_i(x_*)} \left(f_*|x_*,\ \mathcal{D}^{(i)}\right), \tag{S4.2}$$

kurio prognozės rezultatas – atsitiktinis dydis, pasiskirstęs pagal Gauso skirstinį. Jo vidurkis ir dispersija taške x_* apskaičiuojami kaip

$$\mu_{\mathcal{A}} = \sigma_{\mathcal{A}}^2\left(x_*\right) \sum_{i=1}^{M} \alpha_i\left(x_*\right) \sigma_i^{-2}\left(x_*\right) \mu_i\left(x_*\right), \tag{S4.3}$$

$$\sigma_{\mathcal{A}}^{-2}\left(x_*\right) = \sum_{i=1}^{M} \alpha_i\left(x_*\right) \sigma_i^{-2}\left(x_*\right). \tag{S4.4}$$

Svoris $\alpha_i\left(x_*\right)$ yra patikimumo matas, kontroliuojantis kiekvieno eksperto *i* svorį testavimo taške x_*, kur $\alpha_i\left(x_*\right)$ > 0 ir $\sum_{i=1}^{M} \alpha_i\left(x_*\right) = 1$

Apibendrintame sandaugos ekspertų modelyje kiekvienas ekspertas gali būti apmokytas atskirai naudojant lygiagrečiuosius skaičiavimus. Jei mokymas vykdomas su *M* procesoriaus branduoliais, mokymo laiko sudėtingumas sumažėja lyginant su standartiniu GP nuo $O\left(n^3\right)$ iki $O({n_i}^3)$, kur n_i – duomenų aibės dydis, o n_i – ekspertui mokyti naudojamas duomenų poaibio dydis.

S1.5 Heteroskedastiškas Bajeso optimizavimo algoritmas

Daugelyje optimizavimo uždavinių tikslo funkcijos įverčiai gali būti netikslūs ar net klaidingi dėl triukšmo įtakos duomenims. Triukšmas gali iškraipyti arba sumažinti duomenų kokybę ir tikslumą. Taip nutinka dėl įvairių priežasčių: klaidingų matavimų, signalo perdavimo trikdžių ar netinkamo duomenų tvarkymo. Standartiniame Gauso proceso modelyje laikomasi prielaidos, kad triukšmo lygis yra pastovus (t. y. homoskedastiškas) visoje apibrėžimo srityje. Ši prielaida yra pernelyg ribota ir netinkama, nes dažnai triukšmo lygis praktiniuose optimizavimo uždaviniuose yra nepastovus ir priklauso nuo taško padėties (t. y. heteroskedastiški) apibrėžimo srityje. Klaidingai laikantis homoskedastiško triukšmo prielaidos, BO algoritmo rezultatai gali būti klaidingi.

Dėl šios priežasties pasiūlytos kelios BO algoritmo modifikacijos, kurios galėtų būti naudojamas tikslo funkcijai optimizuoti su heteroskedastišku triukšmu. Dažniausiai autoriai siūlo pakeisti standartinį Gauso procesą kitomis jo modifikacijomis, kurios palaikytų heteroskedastišką triukšmą [1, 10]. Be to, pasiūlytos BO algoritmo modifikacijos, naudojančios papildomą Gauso proceso modelį triukšmui prognozuoti priklausomai nuo taško padėties apibrėžimo srityje ir jį naudoti įverčio funkcijoje [20, 21, 35].

S2 BAJESO OPTIMIZAVIMO ALGORITMO MODIFIKACIJOS

Šiame skyriuje aprašomos mūsų pasiūlytos BO algoritmo modifikacijos. Siūlome keisti standartinį Gauso proceso modelį BO algoritme į apibendrintą sandaugos ekspertų (gPoE) modelį. Šis naujas gPoEBO algoritmas pristatomas S2.1 skyriuje. Be to, siūlome naują gPoETRBO algoritmą, sujungiantį patikimumo srities ir gPoE modelį, kaip nurodyta S2.2 skyriuje. Papildomai parodome, kad pasiūlytas gPoEBO algoritmas gali būti pritaikomas optimizavimo problemoms su heteroskedastišku triukšmu. Šis algoritmas aprašomas skyriuje S2.3. Pasiūlyti algoritmai publikuoti moksliniuose straipsniuose [68, 69].

S2.1 Bajeso optimizavimo algoritmas su apibendrintu sandaugos ekspertų modeliu

Siekdami išplėsti BO algoritmo taikymo galimybes didelio masto optimizavimo problemoms, siūlome pakeisti standartinį GP modelį į apibendrintą sandaugos ekspertų (gPoE) modelį. Šį algoritmą pavadinome BO algoritmu su apibendrintu sandaugos ekspertų modeliu (gPoEBO).

Šis algoritmas veikia kiekvienoje iteracijoje t, dalydamas turimą duomenų aibę $D_t = \{x_k,\, y_k\}_{k=1}^{t}$ į M nepersidengiančius poaibius, gautus iš ankstesnių funkcijos įverčių. Kiekviename poaibyje duomenų kiekis yra t/M, kurį panaudojame mokyti M GP ekspertų modelių. Kai M GP ekspertų modeliai apmokomi, jų aposteriorinis vidurkis ir dispersija apskaičiuojami naudojantis atsitiktinai atrinktais q taškais iš apibrėžimo srities. Taškams atrinkti naudojamas Sobolio sekos metodas [65]. Kiekvieno GP eksperto modelio prognozuojamos vidurkio ir dispersijos reikšmės šiuose taškuose yra agreguojamos naudojant gPoE modelį, kur kiekvieno eksperto įtaka galutiniam rezultatui apskaičiuojama naudojant diferencialinės entropijos metodą [13].

Norėdami rasti didžiausią kiekybinį įvertį turintį tašką iš atrinktų q taškų, kiekvienam taškui priskiriame kiekybinį įvertį, naudodami viršutinio patikimumo ribos (angl. Upper Confidence Bound) įverčio funkciją [66]. Ši įverčio funkcija skaičiavimui naudoja turimas vidurkio ir dispersijos reikšmes iš galutinio GP ekspertų modelio. Suradus geriausią įvertį turintį tašką, tikslo funkcija šiame taške įvertinama, o naujas taškas yra pridedamas prie turimų duomenų. Optimizavimo procesas tęsiamas, kol pasiekiamas maksimalus nustatytas funkcijos įverčių skaičius. BO algoritmo rezultatas – didžiausią funkcijos reikšmę turintis taškas, surastas per optimizavcijos procesą.

S2.2 Bajeso optimizavimo algoritmas su patikimos srities ir apibendrintu sandaugos ekspertų modeliu

Siūlome gPoETRBO algoritmą pagerinti gPoEBO algoritmo tikslumui. gPoETRBO algoritmas sujungia patikimumo srities metodą (angl. trust region) su gPoE modeliu. Patikimumo srities metodas remiasi supaprastintu tikslo funkcijos modeliu, kuris leidžia efektyviau ieškoti optimalaus sprendimo sudėtingose optimizavimo problemose. Patikimumo sritis dažniausiai yra hiper-stačiakampis, brėžiamas aplink geriausią jau žinomą tikslo funkcijos reikšmę [17, 55]. Priklausomai nuo to, ar nauja tikslo funkcijos reikšmė yra geresnė už jau žinomą, patikimumo sritis yra koreguojama: išplečiama, jei randamas taškas su geresne funkcijos reikšme, arba susiaurinama, jei funkcijos reikšmės nepavyksta pagerinti. Teigiamas rezultatas veda prie naujo žingsnio priėmimo ir patikimumo srities išplėtimo, o neigiamas rezultatas reiškia naujo žingsnio atmetimą ir patikimumo srities susiaurinimą [15, 83].

Naudojantis šiuo algoritmu, optimizavimą pradedame nustatydami patikimumo srities hiper-stačiakampio pradinį $L = L_{init}$, minimalų L_{min} ir maksimalų L_{max} kraštinės ilgį. Kiekvienoje iteracijoje, kaip ir gPoEBO algoritme, atsitiktinai dalijame duomenų aibę į M nepersidengiančius poaibius ir mokome M GP ekspertus. Paskui turimoje duomenų aibėje randame tašką su didžiausia tikslo funkcijos reikšme. Aplink šį tašką nustatome patikimumo srities hiper-stačiakampį ir sugeneruojame q atsitiktinius taškus šioje srityje naudodami Sobolio sekos metodą. Šiuose sugeneruotose taškuose apskaičiuojame kiekvieno GP eksperto aposteriorinį vidurkį ir dispersiją. Šias reikšmes naudojame rasti agreguotam GP modeliui, naudojantis gPoE modeliu. Norėdami rasti kitą didžiausią potencialą turintį tašką iš sugeneruotų taškų, randame tašką, turintį maksimalią UCB įverčio funkcijos vertę. Radus šį tašką, randame tikslo funkcijos reikšmę šiame taške ir lyginame šią vertę su geriausia jau žinoma tikslo funkcijos reikšme. Jei pagerinome funkcijos vertę, didiname sėkmės skaitiklį ir nustatome nesėkmės skaitiklį į nulį, priešingu atveju nustatome sėkmės skaitiklį į nulį ir didiname nesėkmės skaitiklį. Kai patikimumo srities hiper-stačiakampio ilgis tampa mažesnis už L_{min}, optimizavimo procesą pradedame iš naujo.

S2.3 Heteroskedastiškas Bajeso optimizavimo algoritmas su apibendrintu sandaugos ekspertų modeliu

BO algoritmas su apibendrintu sandaugos ekspertų modeliu gali būti pritaikytas modeliuoti heteroskedastišką duomenų triukšmą, nes kiekvienas GP ekspertas gali turėti individualius parametrus su skirtingu duomenų triukšmu. Apmokius ekspertus $\{\mathcal{M}_i\}_{i=1}^{M}$ su jiems priskirtais duomenų aibės poaibiais $\{\mathcal{D}_i\}_{i=1}^{M}$, randame jų tikimybinius skirstinius $\{p_i\left(y \mid \mathcal{D}_i, x_*\right)\}_{i=1}^{M}$, kur taškas x_* – duomenų aibės elementas, $\mu_{y_i}\left(x_*\right)$ ir $\sigma_{y_i}^2\left(x_*\right)$ – kiekvieno i-tojo eksperto vidurkis ir dispersija.

Siūlome sujungti kiekvieno GP eksperto išmoktus triukšmo lygius jų duomenų poaibiuose $\left\{\sigma_{\epsilon,i}^2\right\}_{i=1}^{M}$, siekiant rasti galutinį triukšmą r_* duomenų aibės taške x_*. Naudojame paprastą aritmetinį vidurkį apskaičiuoti galutiniam triukšmui

$$r_* = r\left(x_*\right) = \sum_{i=1}^{M} \alpha_i\left(x_*\right)\sigma_{\epsilon,i}^2, \tag{S4.5}$$

kur $\alpha_i\left(x_*\right)$ – patikimumo matas, nurodantis kiekvieno *i*-tojo GP eksperto svorį taške x_*, pagal prognozės tikslumą.

Be to, siūlome dvi įverčio funkcijų modifikacijas, kurios naudoja šį triukšmo lygį siekiant rasti didžiausią potencialą turintį tašką apibrėžimo srityje. Pirmoji įverčio funkcija HAEI apskaičiuojama

$$\alpha_{\text{HAEI}}(x) \quad = \quad \alpha_{\text{EI}_n}(x) \times \left(1 - \frac{\gamma\sqrt{r\left(x\right)}}{\sqrt{\sigma_{f_{\mathcal{A}}}^2 + \gamma^2 r\left(x\right)}}\right). \tag{S4.6}$$

Antroji pasiūlyta įverčio funkcija ANPEI tiesiogiai sumažina apibrėžimo srities taško įvertį, priklausomai nuo triukšmo lygio dydžio. Ji apskaičiuojama

$$\alpha_{\text{ANPEI}}(x) \quad = \quad \beta \times \alpha_{\text{EI}_n}(x) - \big(1 - \beta\big) \times \sqrt{r\left(x\right)}, \tag{S4.7}$$

čia β – konstanta, kurios dydis yra tarp 0 ir 1. Ši konstanta parodo pusiausvyrą tarp tikėtino patikimumo įverčio funkcijos EI ir galutinio triukšmo lygio.

S3 EMPIRINIS TYRIMAS

Šiame skyrelyje pristatysime disertacijoje atliktus eksperimentinius tyrimus ir pagrindinius gautus rezultatus. Gauti rezultatai publikuoti moksliniuose straipsniuose [68, 69].

S3.1 Didelio masto Bajeso optimizavimo algoritmų rezultatai

Lyginame savo siūlomų gPoEBO ir gPoETRBO algoritmų efektyvumą ir vykdymo trukmę su kitais GP ekspertų paremtais BO algoritmais (PoE_BO, BCM_BO, rBCM_BO), retosios GP regresijos (SGPRBO) pagrindu veikiančiu BO, standartiniu BO, TuRBO bei atsitiktinės paieškos algoritmu. Šiuos algoritmus vertiname naudodami keturias skirtingų 20 ir 50 dimensijų globaliojo optimizavimo testines funkcijas bei optimalios kontrolės valdymo problemas. Eksperimentai atlikti specialiai mūsų tyrimo problemoms sukurtoje „Google Cloud Platform" (GCP) virtualioje aplinkoje, turinčioje 8 branduolių procesorių.

Eksperimentų rezultatai su 20 ir 50 dimensijų globaliojo optimizavimo testinėmis funkcijomis pateikti 4.1, 4.2, 4.4 ir 4.5 lentelėse, o praktinių problemų optimizavimo rezultatai – 4.7 ir 4.8. Matome, kad geriausi optimizavimo rezultatai gauti naudojant patikimos srities metodo pagrindu veikiančius algoritmus (gPoETRBO, TuRBO). Šie algoritmai pasiekė geriausią optimizuojamos funkcijos tikslumą. Mūsų pasiūlyto gPoETRBO algoritmo tikslumas buvo panašus į TuRBO algoritmo, tačiau vykdymo laikas buvo trumpesnis. Siūlomo gPoEBO algoritmo rezultatai parodė panašų optimizavimo tikslumą, kaip ir standartinio BO, tačiau optimizavimo vykdymo laikas buvo 2 kartus trumpesnis.

S3.2 Heteroskedastiškų Bajeso optimizavimų algoritmų rezultatai

Eksperimentiškai įvertinome gPoE pagrįsto BO (GPOEBO) algoritmo optimizavimo rezultatą su mūsų pasiūlytomis heteroskedastiškomis įverčio funkcijų modifikacijomis (HAEI, ANPEI), lygindami su pažangiausiu heteroskedastišku BO algoritmu, naudojančiu MLHGP modelį

(MLHGPBO). Taip pat palyginome rezultatus su standartiniu BO algoritmu, naudojančiu dvi standartines įverčio funkcijas (BO_EI) ir AEI (BO_AEI), bei su atsitiktinės paieškos algoritmu, kuris dažnai naudojamas triukšmingų funkcijų optimizavime. [21].

Mūsų pasiūlytų algoritmų optimizavimo tikslumas buvo vertinamas naudojant šešias plačiai naudojamas globaliojo optimizavimo testines funkcijas, kaip nurodyta [51]. Eksperimentuose buvo naudojamos įvairių dimensijų funkcijos: 2D (Branin, GoldsteinPrice), 4D (Hartman, Rosenbrock) ir 6D (Hartman, Sphere). Be to, remiantis [21] pasiūlyta metodologija, eksperimentams panaudotos dvi cheminių junginių optimizavimo funkcijos.

Eksperimentinių tyrimų rezultatai buvo vertinami skaičiuojant absoliutinę paklaidą tarp žinomos testinės funkcijos maksimalios reikšmės ir optimizavimo algoritmo rastos geriausios funkcijos reikšmės. Eksperimentai su testinėmis funkcijomis pakartoti 50 kartų. Kaip matome iš 4.10, 4.12 ir 4.14 lentelių, mūsų pasiūlytų algoritmų vidutinė absoliuti paklaida buvo mažiausia lyginant su kitais optimizavimo algoritmais naudojant heteroskedastiškas testines ir praktines optimizavimo funkcijas. Remiantis gautais rezultatais, galime teigti, kad mūsų pasiūlyti algoritmai yra patikimesni spendžiant optimizavimo problemas su heteroskedastišku triukšmu.

IŠVADOS

1. Norėdami pagerinti Bajeso optimizavimo algoritmo taikymą didelio masto optimizavimo problemoms, siūlome du naujus gPoEBO ir gPoETRBO algoritmus. Iš atliktų eksperimentinių tyrimų galime daryti tokias išvadas:

   1.1 Iš gautų rezultatų nustatėme, kad lyginant su standartiniu BO algoritmu, gPoEBO algoritmo tikslumas gerėja nuo -2,87% iki 6,45%, naudojant 20 dimensijų globaliojo optimizavimo testines funkcijas. Optimizavimo vykdymo trukmės pagerėjimas siekė nuo 106,10% iki 209,58%. Šie rezultatai rodo, kad mūsų algoritmas labai sutrumpina optimizavimo vykdymo laiką, tuo pačiu išlaikydamas aukštą optimizavimo tikslumo lygį.

1.2 Pasiūlyto gPoETRBO algoritmo optimizavimo tikslumo pagerėjimas, naudojant 20 ir 50 dimensijų globaliojo optimizavimo testines funkcijas, atitinkamai svyravo nuo 70,51% iki 98,47%, ir nuo 75,56% iki 99,61%, lyginant su standartiniu BO algoritmu. Optimizavimo vykdymo laikas, lyginant su standartiniu BO, sutrumpėjo nuo 68,56% iki 152,39% naudojant 20 dimensijų testines funkcijas, ir nuo 667,44% iki 1118,68%, naudojant 50 dimensijų testines funkcijas. Gauti rezultatai rodo, kad patikimos srities metodas su gPoEBO algoritmu reikšmingai pagerina optimizavimo tikslumą ir sutrumpina vykdymo trukmę.

1.3 Patikimos srities metodu pagrįsto gPoETRBO optimizavimo tikslumas, lyginant su pažangiausiu TuRBO algoritmu, yra toks pats arba geresnis. Tačiau mūsų pasiūlyto gPoETRBO algoritmo optimizavimo trukmė yra nuo 11 iki 12 kartų trumpesnė nei TuRBO algoritmo, naudojant 50 dimensijų testines funkcijas.

1.4 Atlikę eksperimentus su praktinėmis optimizavimo problemomis, nustatėme, kad mūsų siūlomas gPoETRBO algoritmas pasiekė panašų tikslumą, kaip ir standartinis BO, tačiau optimizavimo vykdymas užtruko daug trumpiau. Be to, naudodami gPoETRBO, galėjome optimizuoti 60 dimensijų funkciją su 10000 duomenų aibės elementų standartine kompiuterine įranga, ir pasiekėme 904% optimizavimo tikslumo pagerėjimą, lyginant su standartiniu BO. Lyginant gPoETRBO su TuRBO algoritmu, naudojant 60 dimensijų funkciją, pasiektas panašus tikslumas, tačiau vykdymo trukmė buvo iki 6 kartų trumpesnė.

1.5 Nustatėme, kad gPoETRBO algoritmas, lyginant su kitais algoritmais, veikia geriausiai, kai optimizavimo trukmė yra ribota ir naudojant 20 ir 50 dimensijų globaliojo optimizavimo testines funkcijas su 5 ir 15 minučių optimizavimo vykdymo laiko apribojimu.

1.6 Papildomi eksperimentiniai tyrimai parodė, kad didinant GP ekspertų modeliams priskiriamo duomenų poaibio dydį, gPoEBO ir gPoETRBO optimizavimo tikslumas gerėja, tačiau vykdymo laikas pradeda ilgėti.

2. Šiame darbe siūlome GPOEBO_HAEI ir GPOEBO_ANPEI heteroskedastiškus Bajeso optimizavimo algoritmus, kurie leidžia sėkmingai pritaikyti gPoEBO algoritmą globaliojo optimizavimo problemoms su heteroskedastišku triukšmu:

    2.1 Atlikti eksperimentiniai tyrimai parodė, kad GPOEBO_HAEI algoritmo vidutinė absoliuti paklaida buvo nuo 4,35% iki 33,90% mažesnė visose globaliojo optimizavimo testinėse funkcijose su heteroskedastišku triukšmu, lyginant su standartiniu BO algoritmu.

    2.2 Lyginant GPOEBO_HAEI su pažangiausiu MLHGPBO_HAEI algoritmu, nustatėme, kad mūsų pasiūlyto algoritmo absoliuti vidutinė paklaida buvo nuo 22,71% iki 81,78% mažesnė naudojant testines funkcijas su heteroskedastišku triukšmu.

    2.3 Siūlomų GPOEBO_HAEI ir GPOEBO_ANPEI algoritmų rezultatai parodė nuo 15,61% iki 46,17% mažesnę vidutinę absoliučią paklaidą, lyginant su MLHGPBO_HAEI ir GPOEBO_ANPEI algoritmais, sprendžiant praktines optimizavimo problemas.

    2.4 Atlikus duomenų aibės elementų paskirstymo GP ekspertų modeliui jautrumo analizę, parodėme, kad BO optimizavimo rezultatai su atsitiktine duomenų aibės elementų paskirstymo strategija GP ekspertų modeliui yra tikslesni už K-vidurkių pagrįstą strategiją.

    2.5 Atlikti papildomi eksperimentiniai tyrimai parodė, kad GPOEBO_HAEI ir GPOEBO_ANPEI algoritmų optimizavimo tikslumas priklauso nuo skiriamų duomenų kiekio kiekvienam GP ekspertui. Jei šis skaičius nustatomas neteisingai, šių algoritmų tikslumas gali ženkliai pablogėti.

LIST OF AUTHOR PUBLICATIONS

The results of the dissertation were published in international research journals with a citation index in the Clarivate Analytics Web of Science (CA WoS) database:

- Tautvaišas, S. and Žilinskas, J., 2024. Scalable Bayesian optimization with generalized product of experts. Journal of Global Optimization, 88(3), pp.777-802.
- Tautvaišas, S. and Žilinskas, J., 2023. Heteroscedastic Bayesian optimization using generalized product of experts. Journal of Global Optimization, pp.1-21.

Conference proceedings and abstracts:

- Tautvaišas, S. and Žilinskas, J., 2022. Noisy global Bayesian optimization using generalized product of experts. In Proceedings of the Hungarian global optimization workshop HUGO 2022 (pp. 185-188). University of Szeged.
- Tautvaišas, S. and Žilinskas, J., 2021. Scalable trust region Bayesian optimization with product of experts. 12th International workshop on Data Analysis Methods for Software Systems (DAMSS) (pp. 74), December, Druskininkai, Lithuania.

ABOUT THE AUTHOR

Saulius Tautvaišas was born in Skaistgirys, Joniškis district, Lithuania, in 1988. He graduated from Skaistgirys High School in 2007. Saulius received a Bachelor's degree in Economics from Kaunas University of Technology in 2011 and a Master's degree in Informatics from Vilnius University in 2017. From 2019 to 2023, he was a Ph.D. student at Vilnius University. His industrial experience includes working as an application support analyst at Barclays Technology Centre Lithuania from 2013 to 2015 in Vilnius, Lithuania. He also held various software engineering positions at JP Morgan, UBS, and CME Group between 2015 and 2019 in London, United Kingdom. Since 2021, he has been working as a Data Scientist at Danske Bank in Vilnius, Lithuania.

NOTES

Saulius Tautvaišas

Scalable Bayesian Global
Optimization of Black-Box Functions

Doctoral Dissertation
Natural Sciences
Informatics (N 009)
Thesis Editor: Zuzana Šiušaitė

Saulius Tautvaišas

Bajeso metodai didelio masto juodosios dėžės
globaliajam optimizavimui

Daktaro disertacija
Gamtos mokslai
Informatika (N 009)
Santraukos redaktorė: Jorūnė Rimeisytė-Nekrašienė

Vilnius University Press
9 Saulėtekio al., LT-10222 Vilnius
Email: info@leidykla.vu.lt, www.leidykla.vu.lt
Print run of 20 copies