Bayesian Active Learning with Fully Bayesian Gaussian Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract

The bias-variance trade-off is a well-known problem in machine learning that only gets more pronounced the less available data there is. In active learning, where labeled data is scarce or difficult to obtain, neglecting this trade-off can cause inefficient and non-optimal querying, leading to unnecessary data labeling. In this paper, we focus on active learning with Gaussian Processes (GPs). We argue that for the GP, the bias-variance trade-off is made by optimization of the two hyperparameters: the length scale and noise-term. Considering that the optimal mode of the joint posterior of the hyperparameters is equivalent to the optimal bias-variance trade-off, we approximate this joint posterior and utilize it to design two new acquisition functions. The first one is a Bayesian variant of Query-by-Committee (B-QBC), and the second is an extension that explicitly minimizes the predictive variance through a Query by Mixture of Gaussian Processes (QB-MGP) formulation. Across six common simulators, we empirically show that B-QBC, on average, achieves the best marginal likelihood, whereas QB-MGP achieves the best predictive performance. We show that incorporating the bias-variance trade-off in the acquisition functions mitigates unnecessary and expensive data labeling.

1 Introduction

Gaussian Processes (GPs) are well-known for their ability to deal with small to medium-size data sets by balancing model complexity and regularization [Williams and Rasmussen, 2006]. Together with their inherent ability to model uncertainties, this has made GPs the go-to models to use for Bayesian optimization and metamodeling [Snoek et al., 2012, Gramacy, 2020, Riis et al., 2021]. For both cases, the data is often scarce, making the modeling task a balance between complexity and regularization, i.e. preventing severe overfitting while maintaining the ability to fit nonlinear functions. Likewise, the ability to quantify the uncertainty often guides the acquisition functions of Bayesian optimization and active learning schemes, which are inevitable required to build metamodels efficiently.

On the other hand, it is not a flawless procedure to use GPs in Bayesian optimization and active learning. In both cases, the same GP is used in an iterative process firstly to predict the mean and variance of a new data point and then secondly to use those estimates to guide the data acquisition [Gramacy, 2020]. Since this is an iterative process, poor predictions will lead to poor data acquisition, and vice versa. The problem is less pronounced for larger data sets as predictions become increasingly accurate as more data is available. However, in Bayesian optimization and active learning, where the data sets tend to be relatively small, wrong predictions can result in misguidance, thus hindering performance and efficiency. In this paper, we mitigate this problem by applying a fully Bayesian approach to the GPs and formulating two new acquisition functions for active learning. Where a single GP trained with a maximum likelihood estimate only represents one model hypothesis, a Fully Bayesian Gaussian Process (FBGP) represent multiple model hypotheses at once. We will utilize

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

this extra information to create an acquisition function that simultaneously seeks the best model hypothesis and minimize the prediction error of the GP.

The hyperparameters of a GP are typically fitted through evaluation of the marginal likelihood, which automatically incorporates a trade-off between complexity and regularization [Williams and Rasmussen, 2006], which is also known as the bias-variance trade-off [Bishop, 2006]. However, when the data is scarce, it is more challenging to choose the appropriate trade-off, and different configurations of the hyperparameters of the GP can give rise to distinct fits. We highlight this issue in Figure 1, where two seemingly reasonable fits describe the data very distinctly, which would results in different acquisition of new data.

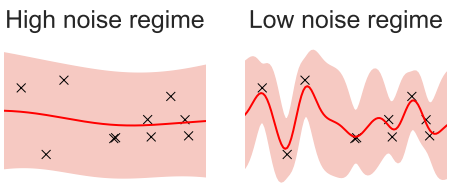

Figure 1: Two GPs with different hyperparameters. The left plot shows a GP with high noise and a long length scale, and the right plot shows a GP with low noise and a short length scale.

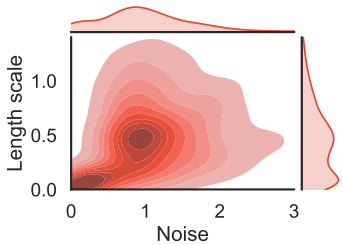

Figure 2: Joint posterior of the two hyperparameters length scale and noise. The posterior is multimodal with a low noise, high signal and a high noise, low signal mode.

The bias-variance trade-off for Gaussian Processes For a GP with a common stationary covariance function with a length scale and a noise-term, e.g., the radial-basis function or the Matérn class of functions with a Gaussian likelihood [Williams and Rasmussen, 2006], the challenge of choosing the best trade-off between bias-variance trade-off, can be formulated using the hyperparameters of the GP. The two central hyperparameters are the length scale ℓ and the variance of the noise σ_ε^2. The length scale describes how much the underlying model f fluctuates, i.e. if the length scale is short or long, the model varies quickly or slowly, respectively. In terms of tuning the hyperparameters, often a short length scale goes well with a small noise-term since the greater flexibility of a short length scale means that the noise level can be reduced. This results in a flexible model with high variance and low bias. conversely, a long length scale tends to give an increase in the noise level, resulting in a rigid model with low variance and high bias [Bishop, 2006]. In the extremes, the former corresponds to a white-noise process model, and the latter corresponds to a constant with white noise [Williams and Rasmussen, 2006]. Thus, there is a direct relation between the bias-variance trade-off and the values of the two hyperparameters.

How to choose the best bias-variance trade-off? Finding a good bias-variance trade-off is a non-trivial problem. In the case of small to medium-sized data sets, the joint posterior distribution of the two hyperparameters is likely to be characterized by two modes, as illustrated in Figure 2 for the data in Figure 1. The two modes illustrate two different bias-variance trade-offs, which both describe the data well. However, depending on the choice of trade-off, the data acquisition can be very different, and thus a wrong choice of mode will imply non-optimal guidance from the acquisition function.

Though the multimodal posterior have been studied for GPs before [Yao et al., 2020], the literature often search for the single best mode with clever initializations of the hyperparameters [Williams and Rasmussen, 1996] or by favoring small ℓ and σ_ε^2 by either always initializing hyperparameters in the low noise regime or by applying strong priors [Gramacy, 2020]. However, none of these approaches directly address the core problem: *which mode to choose*? When working with Bayesian optimization and active learning, this should ideally be answered with prior information about the problem, although typically that is not available, making these approaches less practical [Antunes et al., 2018, Gramacy, 2020, Riis et al., 2021].

Our contribution We follow a general approach to the problem and assume no prior knowledge about the functional form of the data, kernel, nor hyperparameters. If it is known that the data has a periodic trend or high noise, it is an advantage to incorporate that into the kernel and the hyperparameters by using a periodic kernel and a prior on the hyperparameters that favor high

noise, respectively. However, with no prior knowledge, we tend to use the general-purpose Radial-Basis function (RBF) kernel with non-informative priors on the hyperparameters. Further, the fitted hyperparameters are often given by point estimates (e.g. fitted with maximum likelihood estimate (MLE) or maximum a posteriori (MAP)), but we consider multiple model hypotheses by replacing the fitting procedure of the marginal likelihood with Markov Chain Monte Carlo (MCMC) sampling to get the joint posterior of the hyperparameters. The result is that we have multiple models (same kernel, but different hyperparameters), which represent different model hypotheses. We utilize the extra information from the hyperparameters' joint posterior to diminish the bias-variance trade-off by incorporating the extra information into two new acquisition functions.

Our main contribution is the proposal of two new acquisition functions for active learning that utilize the extra information from the hyperparamters' posterior estimated by MCMC to seek the most reasonable mode alongside minimizing the predictive variance. Through empirical results, we show that the two acquisition functions are more accurate and robust than other common functions across multiple benchmark simulators used in the related literature.

2 Related work

In this section, we review related work to the proposed acquisition functions. We cover active learning schemes for regression tasks, including Query-by-Committee and GP as Gaussian Mixture Models.

Active Learning The main idea of active learning (AL) is to *actively* choose a new data point to label and add to the current training data set, to iteratively improve the performance of the predictive model [Settles, 2009]. In the context of metamodeling or surrogate modeling of simulators, new data is often added sequentially, i.e., one data point at a time [Gramacy, 2020], but in other applications, it can be beneficial to query batches of data instead [Kirsch et al., 2019].

The acquisition functions can be divided into model-based and model-free functions, where the former utilize information from the model and the latter do not [O'Neill et al., 2017]. Both types of functions seek to minimize the expected predictive loss of the model. Another approach is to minimize the number of possible models. Houlsby et al. [2011] divide the active learning acquisition functions into being based on either decision or information theory. Decision-based functions seek to minimize the expected predictive loss of the model in the hope of maximizing the performance on the test set. Information-theoretic-based functions instead try to reduce the number of possible models, for example, through the KL-divergence or Shannon entropy [Houlsby et al., 2011].

It is not straightforward to use information-theoretic acquisition functions. However, if one has access to the posterior of the parameters, Houlsby et al. [2011] have derived the algorithm *Bayesian Active Learning by Disagreement* (BALD), which can be applied in general. Generally, BALD seeks the data point that maximizes the decrease in the expected posterior entropy of the parameters.

Query-by-Committee The Query-by-Committee (QBC) is a specific acquisition function that was originally proposed for classification tasks [Seung et al., 1992]. It aims to maximize the disagreement among the committee to get the highest information gain and minimize the version space, which is the set of model hypotheses aligned with the training data. The construction of the committee is the core component of QBC since it is the committee's ability to accurately and diversely represent the version space that gives rise to informative disagreement criteria [Settles, 2009].

Query-by-Committee can also be applied for regression problems. Krogh and Vedelsby [1995] construct the members of the committee by random initializations of the weights in the neural networks. RayChaudhuri and Hamey [1995] apply bagging and train the members on different subsets of the data set. In general, QBC constructed by bagging has been used as a benchmark with mixed results [Cai et al., 2013, Wu, 2018, Wu et al., 2019]. Burbidge et al. [2007] show that the less noise there is in the output, the better QBC is compared to random querying. They also highlight the fact that with a misspecified model, QBC might perform worse than random querying. None of these approaches explore the usage of MCMC samples of the posterior to construct a committee.

Gaussian Process as a Gaussian Mixture Model Mixture models have recently been applied in active learning for classification tasks. Iswanto [2021] proposes to use Gaussian Mixture Models (GMMs) with active learning, where they design a specific acquisition function that queries the data point that maximizes the expected likelihood of the model. Zhao et al. [2020] use a mixture of GPs in active learning, where each component is fitted to a subset of the training set. The combination of

GMMs and GPs have previously been explored for static data sets. Chen and Ren [2009] investigate regression tasks and apply bagging, where they repeatedly randomly sample data points from the training set to construct new subsets to get GPs fitted to different data.

3 Gaussian Processes

The Gaussian Processes (GPs) are the central models in this work. In this section, we give a brief overview of GPs before covering the Fully Bayesian GPs. For a thorough description of GPs, we refer to Williams and Rasmussen [2006].

A Gaussian Process (GP) is a stochastic function fully defined by a mean function $m(\cdot)$ and a covariance function (often called a kernel) $k(\cdot,\cdot)$. Given the data $\mathcal{D} = (X, \boldsymbol{y}) = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where y_i is the corrupted observations of some latent function values $\boldsymbol{f}$ with Gaussian noise ε, i.e., $y_i = f_i + \varepsilon_i,\ \varepsilon_i \in \mathcal{N}(0, \sigma_\varepsilon^2)$, a GP is typically denoted as $\mathcal{GP}(m_{\boldsymbol{f}}(\boldsymbol{x}), k_{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{x}'))$. It is common practice to set the mean function equal to the zero-value vector and thus, the GP is fully determined by the kernel $k_{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{x}')$. For short, we will denote the kernel $K_{\boldsymbol{\theta}}$, which explicitly states that the kernel is parameterized with some hyperparameters $\boldsymbol{\theta}$. The generative model of the GP can be found in Appendix A.1. Given the hyperparameters $\boldsymbol{\theta}$, the predictive posterior for unknown test inputs $X^\star$ is given by $p(\boldsymbol{f}^\star|\boldsymbol{\theta}, \boldsymbol{y}, X, X^\star) = \mathcal{N}(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star)$ with

$$\boldsymbol{\mu}^\star = K_{\boldsymbol{\theta}}^\star \left(K_{\boldsymbol{\theta}} + \sigma_\varepsilon^2 \mathbb{I}\right)^{-1} y \quad \text{and} \quad \boldsymbol{\Sigma}^\star = K_{\boldsymbol{\theta}}^{\star\star} - K_{\boldsymbol{\theta}}^\star \left(K_{\boldsymbol{\theta}} + \sigma_\varepsilon^2 \mathbb{I}\right)^{-1} K_{\boldsymbol{\theta}}^{\star\top} \tag{1}$$

where $K_{\boldsymbol{\theta}}^{\star\star}$ denotes the covariance matrix between the test inputs, and $K_{\boldsymbol{\theta}}^\star$ denotes the covariance matrix between the test inputs and training inputs.

We use the canonical kernel automatic relevance determination (ARD) Radial-basis function (RBF) given by $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-||\boldsymbol{x} - \boldsymbol{x}'||^2 / 2\boldsymbol{\ell}^2\right)$ where $\boldsymbol{\ell}$ is a vector of length scales $\ell_1, ..., \ell_d$, one for each input dimension. Often the kernel is scaled by an output variance but here we fix it to one and solely focus on the two other hyperparameters: length scale and noise-term. The noise-term σ_ε^2 is integrated into the kernel with an indicator variable by adding the term $\sigma_\varepsilon^2 \mathbb{I}_{\{\boldsymbol{x}=\boldsymbol{x}'\}}$ to the current kernel [Williams and Rasmussen, 2006, Bishop, 2006].

Fully Bayesian Gaussian Processes (FBGP) A FBGP extends a GP by putting a prior over the hyperparameters $p(\boldsymbol{\theta})$ and approximate their full posteriors. The joint posterior is then given by

$$p(\boldsymbol{f}, \boldsymbol{\theta}|\boldsymbol{y}, X) \propto p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|\boldsymbol{\theta}, X) p(\boldsymbol{\theta}) \tag{2}$$

and the predictive posterior for the test inputs $X^\star$ is

$$p(\boldsymbol{y}^\star|\boldsymbol{y}) = \iint p\left(\boldsymbol{y}^\star|\boldsymbol{f}^\star, \boldsymbol{\theta}\right) p(\boldsymbol{f}^\star|\boldsymbol{\theta}, \boldsymbol{y}) p(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{f}^\star d\boldsymbol{\theta} \tag{3}$$

where the conditioning on X and $X^\star$ have been omitted for brevity. The inner integral reduces to the predictive posterior given by a normal GP, whereas the outer integral remains intractable and is approximated with MCMC inference with M samples as

$$p\left(\boldsymbol{y}^\star|\boldsymbol{y}\right) = \int p\left(\boldsymbol{y}^\star|\boldsymbol{y}, \boldsymbol{\theta}\right) p(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{\theta} \quad \approx \quad \frac{1}{M} \sum_{j=1}^{M} p\left(\boldsymbol{y}^\star|\boldsymbol{y}, \boldsymbol{\theta}_j\right), \quad \boldsymbol{\theta}_j \sim p(\boldsymbol{\theta}|\boldsymbol{y}) \tag{4}$$

Adapting the hyperparameters of a FBGP is computationally expensive compared to the approach with GPs and maximum likelihood estimates. However, in Bayesian optimization and active learning, the computational burden for querying a new data point will often be of magnitudes higher.

4 Active Learning

In this section, we lay out the most common acquisition functions and then propose first a Bayesian variant of Query-by-Committee and second an extension motivated by Gaussian Mixture Models, which seek to minimize both the predictive variance and the number of model hypotheses. All the acquisition functions choose a data point $\boldsymbol{x}$ among the possible data points in the unlabeled pool U.

Many classic active learning acquisition functions are based on the model's uncertainty and entropy and can thus be denoted as *Bayesian* active learning acquisition functions [Settles, 2009, Gramacy,

2020]. The most common acquisition function is based on the predictive entropy and denoted *Active Learning MacKay* (ALM) [MacKay, 1992]. All the following objective functions query a new data point by maximizing the argument $\boldsymbol{x}$. In the following, we write a new test point $\boldsymbol{x}^\star$ as $\boldsymbol{x}$ for brevity.

Entropy (ALM) For a Gaussian distribution, the Shannon entropy $H[\cdot]$ is proportional to the predictive variance $\sigma^2(\boldsymbol{x})$ (derived in A.2), so ALM is given as

$$\text{ALM} = H[y|\boldsymbol{x}, \mathcal{D}] \propto \sigma^2(\boldsymbol{x}) \tag{5}$$

Intuitively, ALM queries the data point, where the uncertainty of the prediction is the highest.

If we have access to the posterior of the model's hyperparameters, we can utilize acquisition functions with an extra Bayesian level. The posterior of the hyperparameters of an FBGP can be estimated with MCMC such that GPs with different kernel parameters can be drawn, e.g., the length scale and the noise-term $\boldsymbol{\ell}, \sigma_\varepsilon^2 \sim p(\boldsymbol{\theta}|\mathcal{D})$. The following four acquisition functions all utilize this information and are approximated using the samples from the MCMC, cf. (4).

Entropy (B-ALM) With the information from the posterior $p(\boldsymbol{\theta}|\mathcal{D})$, the criteria for the extra Bayesian variant of ALM (B-ALM) is then given as

$$\text{B-ALM} = H\left[\int p(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}\right] = H\left[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[p(y|\boldsymbol{x}, \boldsymbol{\theta})]\right] \propto \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})|\boldsymbol{\theta}] \tag{6}$$

Bayesian Active learning by Disagreement (BALD) Another common objective in Bayesian active learning, is to maximize the expected decrease in posterior entropy [Guestrin et al., 2005, Houlsby et al., 2012]. Houlsby et al. [2011] rewrite the objective from computing entropies in the parameter space to the output space by observing that it is equivalent to maximizing the conditional mutual information between the model's parameters $\hat{\boldsymbol{\theta}}$ and unknown output $I[\hat{\boldsymbol{\theta}}, y|\boldsymbol{x}, \mathcal{D}]$. The acquisition function is denoted Bayesian Active Learning by Disagreement (BALD) and the criteria is given by:

$$I[\hat{\boldsymbol{\theta}}, y|\boldsymbol{x}, \mathcal{D}] = H[y|\boldsymbol{x}, \mathcal{D}] - \mathbb{E}_{p(\hat{\boldsymbol{\theta}}|\mathcal{D})}[H[y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}]] \tag{7}$$

BALD was originally derived for non-parametric discriminative models but has recently been extended to batches and deep learning with great success [Kirsch et al., 2019]. In the context of non-parametric models, such as GPs, the models parameters now corresponds to the latent function $\boldsymbol{f}$. For a regular GP, BALD is equivalent to ALM (cf. A.3), although that is not the case for a FBGP. If we let the hyperparameters be the main parameters of interest and setting f to be a nuisance parameter, BALD can be written as (cf. A.3.1):

$$\text{BALD} = H\left[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[y|\boldsymbol{x}, \mathcal{D}, \boldsymbol{\theta}]\right] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[H[y|\boldsymbol{x}, \boldsymbol{\theta}]\right] \tag{8}$$

Bayesian Query-by-Committee (B-QBC) Motivated by finding the optimal bias-variance trade-off, we propose a Bayesian version of the Query-by-Committee, using the MCMC samples of the hyperparameters' joint posterior. We have previously argued that the optimal bias-variance trade-off is equivalent to the optimal mode of the multimodal posterior of the hyperparameters, which is exactly what we utilize here. We use the joint posterior of the hyperparameters obtained through MCMC to draw multiple models and then query a new data point where the mean predictions $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$ of these models disagree the most, i.e. querying the data point that maximizes the variance of $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$. Each mean predictor $\mu_{\boldsymbol{\theta}}(\cdot)$ drawn from the posterior is equivalent to a single model, and thus this criteria can be seen as a Bayesian variance of a Query-by-Committee, and thus denoted as Bayesian Query-by-Committee (B-QBC). Given that $\overline{\mu}(\boldsymbol{x})$ is the average mean function, B-QBC is given as

$$\text{B-QBC} = V_{p(\boldsymbol{\theta}|\mathcal{D})}[\mu_{\boldsymbol{\theta}}(\boldsymbol{x})|\boldsymbol{\theta}] = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}) - \overline{\mu}(\boldsymbol{x}))^2|\boldsymbol{\theta}] \tag{9}$$

Since the models are drawn from the hyperparameters' posterior, the collection of models is dominated by models near the posterior modes. High variance in $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$, thus corresponds to high disagreement between modes. Querying the data point that maximizes this disagreement, gives information about which mode is most likely to be the optimal one, and thus this can be seen as a mode-seeking Bayesian Query-by-Committee. To the best of our knowledge, we are the first to propose QBC based on model hypotheses drawn from the hyperparameters' joint posterior.

Query by Mixture of Gaussian Processes (QB-MGP) Bayesian Query-by-Committee (B-QBC) seek the optimal mode, but does not take the predictive performance of the model into account.

Table 1: Stochastic simulators used in the experiments.

Simulator	d	Noise σ_ε	Input space	Previously used in
Gramacy1d	1	0.1	$[0.5, 2.5]$	Gramacy and Lee [2012]
Higdon1d	1	0.1	$[0, 20]$	Gramacy and Lee [2009], Gramacy [2020]
Gramacy2d	2	0.05	$[-2, 6]^2$	Gramacy and Lee [2009, 2012], Sauer et al. [2021]
Branin	2	11.32	$[-5, 10] \times [0, 15]$	Keane et al. [2008], Picheny et al. [2013], Cole et al. [2022]
Ishigami	3	0.187	$[-\pi, \pi]^3$	Marrel et al. [2009], Cole et al. [2022]
Hartmann	6	0.0192	$[0, 1]^6$	Picheny et al. [2013], Cole et al. [2022]

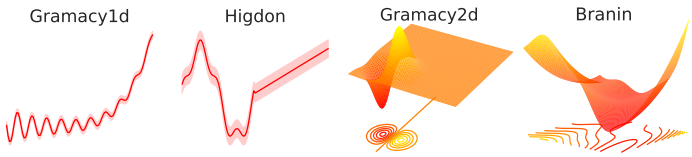

Figure 3: Visualization of the simulators (excl. Ishigami and Hartmann due to the dimensionality).

Since the predictive performance, and thereby the predictive uncertainty, is also important, we extend B-QBC to consider the predictive entropy as well. We denote the new acquisition function *Query by Mixture of Gaussian Processes* (QB-MGP), because of its relation to Gaussian Mixture Models (GMMs). Using the MCMC samples, each prediction of the FBGP can be seen as a GMM, yielding the predictive posterior given as in equation (4): $\frac{1}{M}\sum_{j=1}^{M} p\left(\boldsymbol{y}^\star|\boldsymbol{y}, \boldsymbol{\theta}_j\right)$. This hierarchical predictive posterior is a mixture of M Gaussians with mean μ_{GMM} and variance σ^2_{GMM} defined as (cf. A.4):

$$\mu_{GMM}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} \mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) \tag{10}$$

$$\sigma^2_{GMM}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} \sigma^2_{\boldsymbol{\theta}_j}(\boldsymbol{x}) + \frac{1}{M}\sum_{j=1}^{M} (\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) - \mu_{GMM}(\boldsymbol{x}))^2 \tag{11}$$

Finding the data point that maximizes the variance of the GMM is now equivalent to simultaneously considering the B-ALM and B-QBC:

$$\text{QB-MGP} = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})|\boldsymbol{\theta}] + \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mu_{GMM}(\boldsymbol{x}))^2|\boldsymbol{\theta}] \tag{12}$$

Instead of using bagging, we construct the multiple GPs by using the MCMC samples of the hyperparameters' joint posterior, and then obtain a natural weighting of the GPs: the GMM will consist of more GPs with hyperparameters close to the modes than hyperparameters far away. To the best of our knowledge, we are the first to combine GMMs and GPs in this manner for active learning.

5 Experiments

In this section, we benchmark the performance of the two proposed acquisition functions against the standard acquisition functions based on the entropy, i.e., ALM, B-ALM, and BALD, on various classic simulators used in recent literature on GPs and active learning. They are all listed in Table 1, and those with less than three inputs are shown in Figure 3.[1] The multimodal posteriors of the FBGPs fitted to the simulators can be found in the Appendix A.5.

Experimental settings In the experiments, we use a zero-mean GP with an ARD RBF kernel. In each iteration of the active learning loop, the inputs are rescaled to the unit cube $[0, 1]^d$, and the outputs are standardized to have zero mean and unit variance. Following Lalchand and Rasmussen [2020], we give all the hyperparameters relatively uninformative $\mathcal{N}(0, 3)$ priors in log space. The initial data sets consist of three data points chosen by maximin Latin Hypercube Sampling [Pedregosa et al., 2011], and in each iteration, one data point is queried. The unlabeled pool U consists of the input space discretized into 100 equidistant points along each dimension. If U contains more than $10,000$ data points, we randomly sample a subset of $10,000$ data points in each iteration and use that

[1] All of them can be found at `https://www.sfu.ca/~ssurjano/`.

Table 2: The relative decrease in the area under the active learning curves compared to the active learning curve of the baseline acquisition function ALM.

	Gramacy1d	**Higdon**	**Gramacy2d**	**Branin**	**Ishigami**	**Hartmann**	**Overall performance**	
Dimensions	1	1	2	2	3	6	Mean	Median
NLML: relative decrease in AUC (%)								
ALM	0.0 ±0.6	0.0 ±1.2	0.0 ±6.4	0.0 ±1.2	0.0 ±1.1	0.0 ±22.2	0.0 ±0.0	0.0 ±0.0
B-ALM	-1.7 ±0.5	1.2 ±1.4	**43.2 ±3.1**	-2.4 ±1.6	7.9 ±1.3	48.2 ±10.4	16.1 ±21.3	4.6 ±21.3
BALD	-28.1 ±2.1	-13.6 ±2.0	7.8 ±5.0	-36.0 ±3.6	**39.1 ±1.1**	-17.3 ±22.0	-8.0 ±25.1	-15.5 ±25.1
B-QBC	**36.3 ±1.0**	**41.5 ±1.1**	23.9 ±4.6	**33.1 ±1.2**	37.5 ±0.8	75.5 ±5.1	**41.3 ±16.2**	**36.9 ±16.2**
QB-MGP	20.6 ±0.9	15.9 ±1.4	30.6 ±4.4	7.1 ±1.1	21.7 ±0.7	**80.1 ±3.9**	29.3 ±23.8	21.1 ±23.8
RMSE: relative decrease in AUC (%)							Mean	Median
ALM	0.0 ±0.2	0.0 ±0.7	0.0 ±1.5	0.0 ±0.6	0.0 ±0.8	0.0 ±1.1	0.0 ±0.0	0.0 ±0.0
B-ALM	2.4 ±0.2	5.0 ±0.9	-2.5 ±1.3	1.5 ±0.7	7.2 ±0.6	-2.3 ±1.0	1.9 ±3.5	2.0 ±3.5
BALD	-23.3 ±3.9	-133.0 ±8.8	**7.9 ±1.5**	-18.0 ±1.9	**21.2 ±0.6**	**2.6 ±0.9**	-23.7 ±51.1	-7.7 ±51.1
B-QBC	**20.8 ±0.9**	4.5 ±1.0	1.7 ±1.9	-6.6 ±1.0	9.4 ±0.5	-3.9 ±1.3	4.3 ±9.0	3.1 ±9.0
QB-MGP	19.6 ±0.8	**10.4 ±0.8**	-6.7 ±1.4	**1.7 ±0.7**	7.2 ±0.6	-1.2 ±0.9	**5.2 ±8.5**	**4.5 ±8.5**

as the new pool. The inference in FBGP is carried out using NUTS [Hoffman and Gelman, 2014] in Pyro [Bingham et al., 2019] with five chains and 500 samples, including a warm-up period with 200 samples. The remaining 1500 samples are all used for the acquisition functions. For all the predictions, we use the best mode of the hyperparameters' posterior, since the mean is of limited value when posterior is multimodal. The best mode is computed by using a kernel density estimation with a Gaussian kernel [Pedregosa et al., 2011]. The models are implemented in GPyTorch [Gardner et al., 2018]. All experiments are repeated ten times with different initial data sets. With seven simulators and five acquisition functions, this give 350 active learning runs, each with a running time on approximately one hour, using four CPU cores on a Threadripper 3960X. The code for reproducing the experiments is available in the supplementary material.

5.1 Experiments with 6 simulators

We benchmark the acquisition functions on the six classic simulators. We divide the simulators into subgroups and describe how the functions work in different complexities and with multiple inputs, effectively encompassing three distinct modeling scenarios. The following paragraphs describe the active learning curves in Figure 4.

Noise or signal The simulator *Gramacy1d* has previously been used to study the effect of the noise-term in GPs [Gramacy and Lee, 2012]. The simulator has a periodic signal that is hard to reveal if the data points are not queried cleverly. Both B-QBC and QB-MGP reach convergence simultaneously, while the other acquisition functions struggle in distinguishing noise from the signal.

Linear and non-linear output regions The two simulators *Higdon* and *Gramacy2d* have been used to illustrate cases where GPs struggle in modeling the data due to the output signal having both linear and non-linear regions [Gramacy and Lee, 2009]. Our experiments on these simulators show the performance of the acquisition functions when the GP is an inadequate choice of model. Querying data points in the linear and non-linear regions will yield a GP with a longer and shorter length scale, respectively. On Higdon, it is seen that both B-QBC and QB-MGP balance the sampling since the corresponding NLML and RMSE are low. On Gramacy2d, QB-MGP is achieving the highest RMSE due to favoring to capture the small region with a non-linear signal compared to the vast linear region. Overall, these results show that when the GP is inadequate to model the data, both B-QBC and QB-MGP perform better than the other acquisition functions.

Multiple inputs To evaluate the performance on higher dimensions, we consider the smooth 2d *Branin* simulator, the strongly non-linear 3d *Ishigami* simulator, and the 6d *Hartmann* simulator with six local minima. BALD underperforms on Branin, but the other acquisition functions have similar performance. For the higher dimensions, the active learning curves exhibit more variation. BALD achieves the best performance on Ishigami, whereas both B-QBC and QB-MGP reach the best NLML, and are most steady across the iterations on Hartmann.

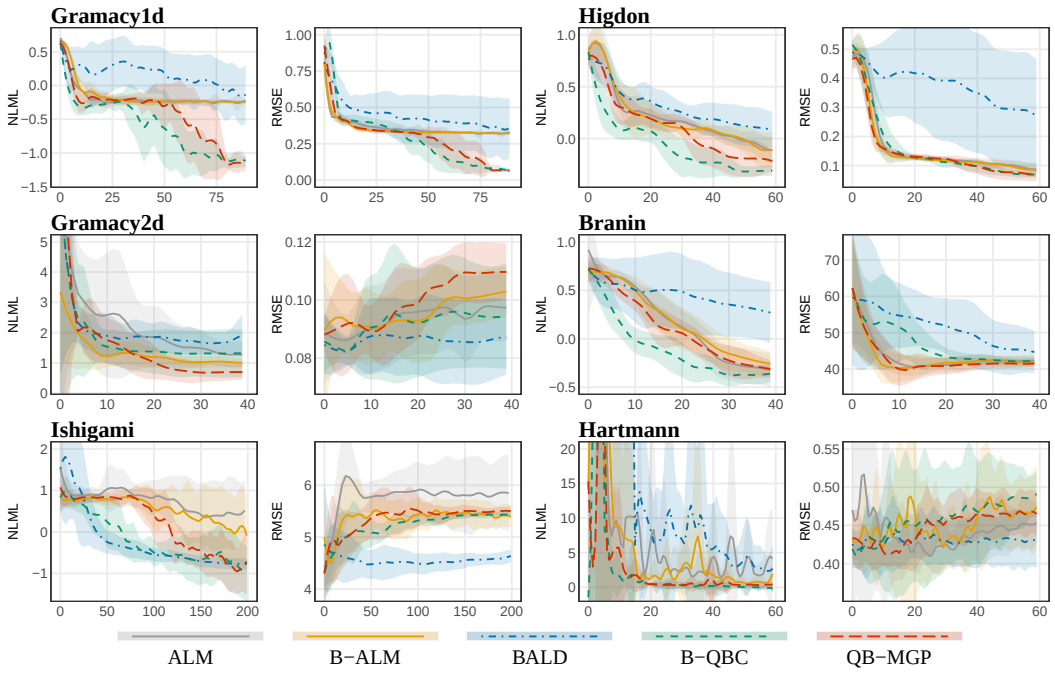

Figure 4: Performance across the 10 runs ± 1 standard deviation. The x-axis represents the number of iterations.

5.2 The Overall Performance

From the visual inspection of Figure 4, it is hard to measure how better an acquisition function is to another. To quantify the overall performance, we propose, similarly to Yang and Loog [2018], to use the relative decrease in Area Under the active learning Curve (AUC) to measure the performance. We compare the acquisition functions using ALM as the baseline, since it widely used and regarded as the standard within active learning with GPs.

For a metric that is lower bounded by zero, such as root mean square error (RMSE), the AUC will give the overall error of the acquisition function. We can directly calculate the relative decrease in error from the AUCs of the active learning acquisition functions and the baseline. If the metric has no lower bound, such as the negative log marginal likelihood (NLML), the interpretation is less intuitive. Therefore, we make a lower bound for the metric using the lowest NLML obtained across all the acquisition functions, such that the relative decrease in the AUC then can be interpreted in the same way as for the RMSE. We compare all 10 runs of each acquisition function with the 10 runs for the baseline to get a precise estimate of both the mean and standard deviation of the relative decreases. The results are shown in Table 2.

First of all, it is clear that there is not a single acquisition functions that is performing the best for all the simulators. In general, B-QBC and BALD achieves the largest decrease in AUC for either NLML or RMSE in four cases. QB-MGP is the next best, having the largest decrease three times. The performance across the different simulators shows, however, that BALD also sometimes performs significantly worse than the other two acquisition functions. This is reflected in the overall performance, where B-QBC and QB-MGP are the best performing model in terms of the marginal likelihood and root mean square error, respectively. Oppositely, BALD is the worst performing model overall, not even better than the baseline. In other words, given the overall good consistent performance of B-QBC and QB-MGP (each only worse than the baseline twice), we say that they are robust to different complexities in the simulators outputs.

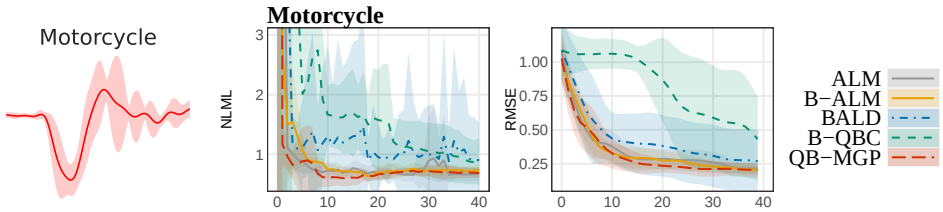

Figure 5: Left: Visualization of the Motorcycle simulator. Right: Performance across 10 runs ± 1 standard deviation. The x-axis represent the number of iterations.

5.3 Limitations

This paper is based on empirical results that are dependent on the specific simulators. The simulators represents diverse and distinct classic homoscedastic simulators and are representative of the engineering problems occurring in the real world [Gramacy, 2020, Sauer et al., 2021, Cole et al., 2022]. However, an interesting case that is less often investigated in the literature of active learning with simulators, is a simulator with heteroscedastic noise.

One of the more common heteroscedastic case studies, is the motorcycle data set [Silverman, 1985, Gramacy and Lee, 2008, Gramacy, 2020]. We create a corresponding simulator by fitting a variational GP [Hensman et al., 2015] to the motorcycle accident data. For reproducibility, the mean and standard deviation of the simulator is given in appendix (A.6). The experiments on the *Motorcycle* simulator explore how the active learning acquisition functions perform when the simulator has heteroscedastic noise, but we model it with a homoscedastic GP. The simulator and the results are seen in Figure 5. Most conspicuous is the poor performance of B-QBC, indicating that this is misled by the heteroscedastic noise. However, the combination of B-ALM and B-QBC in QB-MGP gives the best performance, which might be due to the fact that the B-ALM component of the QB-MGP encourages more exploration since it aims in reducing the overall predictive uncertainty.

Another aspect of the study, is the use of domain and expert knowledge. The incorporation of domain and expert guidance regarding the simulator under study can be a decisive factor in a successful active learning strategy. However, in many practical situations, such a priori domain expertise may not be readily accessible or even translatable into the functional structure of the model as useful modeling information. On these occasions, generic tools that are robust enough to handle a plethora of diverse simulation output behaviors are prudently advisable. If we had such information regarding the functional complexity, e.g., knowing that the signal is periodic, this study does not show which of the acquisition functions that is best. In future work, it would indeed be interesting to see if B-QBC and QB-MGP would perform equally well.

6 Conclusion

In this paper, we propose two active learning acquisition functions: Bayesian Query-by-Committee (B-QBC) and Query by a Mixture of Gaussian Processes (QB-MGP), both of which are suited for fully Bayesian GPs. They are designed to explicitly handle the well-known bias-variance trade-off by optimization of the GP's two hyperparameters, length scale and noise-term. We empirically show that they query new data points more efficiently than previously used acquisition functions. Across six classic simulators, which cover different complexities and number of inputs, we show that B-QBC and QB-MGP are the two functions that achieve the best marginal likelihood and root mean square error, respectively, with the fewest iterations. On average, across the simulators, B-QCB reduced the negative marginal log likelihood with 41%, and QB-MGP decreased the root mean square error with 5% compared to the baseline. To this end, we believe that the proposed acquisition functions are robust enough to handle a variety of diverse simulation output behaviors, while being entirely independent of any prior understanding of the underlying output distributions of the simulator.

References

F. Antunes, B. Ribeiro, F. C. Pereira, and R. Gomes. Efficient Transport Simulation With Restricted Batch-Mode Active Learning. *IEEE Transactions on Intelligent Transportation Systems*, 19(11):3642–3651, nov 2018.

ISSN 1524-9050. doi: 10.1109/TITS.2018.2842695. URL https://ieeexplore.ieee.org/document/8419064/.

E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 978-0387-31073-2. URL http://research.microsoft.com/en-us/um/people/cmbishop/prml/.

R. Burbidge, J. J. Rowland, and R. D. King. Active learning for regression based on query by committee. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4881 LNCS:209–218, 2007. ISSN 16113349. doi: 10.1007/978-3-540-77226-2_22.

W. Cai, Y. Zhang, and J. Zhou. Maximizing expected model change for active learning in regression. *Proceedings - IEEE International Conference on Data Mining, ICDM*, (December 2013):51–60, 2013. ISSN 15504786. doi: 10.1109/ICDM.2013.104.

T. Chen and J. Ren. Bagging for Gaussian process regression. *Neurocomputing*, 72(7-9):1605–1610, 2009. ISSN 09252312. doi: 10.1016/j.neucom.2008.09.002.

D. A. Cole, R. B. Gramacy, J. E. Warner, G. F. Bomarito, P. E. Leser, and W. P. Leser. Entropy-based adaptive design for contour finding and estimating reliability. *Journal of Quality Technology*, pages 1–18, 2022. URL https://doi.org/10.1080/00224065.2022.2053795.

J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in Neural Information Processing Systems*, 31: 7576–7586, 2018. ISSN 10495258.

R. B. Gramacy. *Surrogates*. Chapman and Hall/CRC, mar 2020. ISBN 9780367815493. doi: 10.1201/9780367815493. URL https://www.taylorfrancis.com/books/9781000766202.

R. B. Gramacy and H. K. Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103, 2008. ISSN 01621459. doi: 10.1198/016214508000000689.

R. B. Gramacy and H. K. Lee. Adaptive design and analysis of supercomputer experiments. *Technometrics*, 51 (2):130–145, 2009. ISSN 00401706. doi: 10.1198/TECH.2009.0015. URL https://doi.org/10.1198/TECH.2009.0015.

R. B. Gramacy and H. K. Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22, 2012. ISSN 09603174. doi: 10.1007/s11222-010-9224-x.

C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning - ICML '05*, volume 1, pages 265–272, New York, New York, USA, 2005. ACM Press. ISBN 1595931805. doi: 10.1145/1102351.1102385. URL http://portal.acm.org/citation.cfm?doid=1102351.1102385.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

M. D. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian Active Learning for Classification and Preference Learning. pages 1–17, 2011. URL http://arxiv.org/abs/1112.5745.

N. Houlsby, F. Huszar, Z. Ghahramani, and J. Hernández-lobato. Collaborative gaussian processes for preference learning. *Advances in Neural Information Processing Systems*, 25, 2012.

B. H. Iswanto. Active learning by increasing model likelihood for Gaussian mixture models based classifiers. *IOP Conference Series: Materials Science and Engineering*, 1098(3):032036, mar 2021. ISSN 1757-8981. doi: 10.1088/1757-899X/1098/3/032036. URL https://iopscience.iop.org/article/10.1088/1757-899X/1098/3/032036.

A. Keane, A. Forrester, and A. Sobester. *Engineering Design via Surrogate Modelling: A Practical Guide*. 2008. doi: 10.2514/4.479557.

A. Kirsch, J. van Amersfoort, and Y. Gal. BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. *Advances in Neural Information Processing Systems*, 32, 2019. ISSN 10495258.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, volume 7, pages 231–238, 1995.

V. Lalchand and C. E. Rasmussen. Approximate inference for fully bayesian gaussian process regression. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12. PMLR, 2020.

D. J. C. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4): 590–604, 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.4.590.

A. Marrel, B. Iooss, B. Laurent, and O. Roustant. Calculations of sobol indices for the gaussian process metamodel. *Reliability Engineering and System Safety*, 94, 2009. ISSN 09518320. doi: 10.1016/j.ress.2008.07.008.

J. O'Neill, S. Jane Delany, and B. MacNamee. Model-Free and Model-Based Active Learning for Regression. In *Advances in computational intelligence systems*, pages 375–386. Springer, 2017. ISBN 9783319465623. doi: 10.1007/978-3-319-46562-3_24. URL `http://link.springer.com/10.1007/978-3-319-46562-3{_}24`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

V. Picheny, T. Wagner, and D. Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48, 2013. ISSN 16151488. doi: 10.1007/s00158-013-0919-4.

T. RayChaudhuri and L. G. Hamey. Minimization of data collection by active learning. *IEEE International Conference on Neural Networks - Conference Proceedings*, 3:1338–1341, 1995. doi: 10.1109/icnn.1995.487351.

C. Riis, F. Antunes, G. Gurtner, F. C. Pereira, L. Delgado, and C. M. Lima Azevedo. Active learning metamodels for atm simulation modeling. In *Proceedings of the 11th SESAR Innovation Days, 2021*, 2021. URL `https://www.sesarju.eu/SIDs2021`.

A. Sauer, R. B. Gramacy, and D. Higdon. Active learning for deep gaussian process surrogates. *Technometrics*, (just-accepted):1–39, 2021.

B. Settles. Active learning literature survey. 2009.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, pages 287–294, New York, New York, USA, 1992. ACM Press. ISBN 089791497X. doi: 10.1145/130385.130417. URL `http://portal.acm.org/citation.cfm?doid=130385.130417`.

B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47, 1985. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1985.tb01327.x.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. volume 4, 2012.

C. K. Williams and C. E. Rasmussen. Gaussian processes for regression. 1996.

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

D. Wu. Pool-Based Sequential Active Learning for Regression. *TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 30(5):1348–1359, 2018. ISSN 23318422.

D. Wu, C. T. Lin, and J. Huang. Active learning for regression using greedy sampling. *Information Sciences*, 474:90–105, 2019. ISSN 00200255. doi: 10.1016/j.ins.2018.09.060. URL `https://doi.org/10.1016/j.ins.2018.09.060`.

Y. Yang and M. Loog. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 83:401–415, 2018.

Y. Yao, A. Vehtari, and A. Gelman. Stacking for Non-mixing Bayesian Computations: The Curse and Blessing of Multimodal Posteriors. (June), 2020. ISSN 2331-8422. URL `http://arxiv.org/abs/2006.12335`.

J. Zhao, S. Sun, H. Wang, and Z. Cao. Promoting active learning with mixtures of Gaussian processes. *Knowledge-Based Systems*, 188:105044, 2020. ISSN 09507051. doi: 10.1016/j.knosys.2019.105044. URL `https://doi.org/10.1016/j.knosys.2019.105044`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] All the claims in the abstract and introduction is reflected in the discussion of the results, cf. section 5.
   (b) Did you describe the limitations of your work? [Yes] See subsection 5.3
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] No, we create a generic active learning acquisition function, so there is no direct negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] The work presents a generic algorithm that is test on simulators and does not have any ethic related issues.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We give no theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] We give no theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See section 5.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See paragraph "Experimental settings" in section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See both error bars in Figure 4 and Table 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See paragraph "Experimental settings" in section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite GPyTorch [Gardner et al., 2018] and Pyro [Bingham et al., 2019] as well as all the used simulators. See Table 1.
   (b) Did you mention the license of the assets? [No] All of it is open-source software.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We do not provide any new assets, only the code for reproducing the experiments.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] All the simulators are previously used in the literature and are available at `https://www.sfu.ca/~ssurjano/`.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Gaussian Process

The generative model of a GP is given as

$$\boldsymbol{f}|X \sim \mathcal{N}(0, K_{\boldsymbol{\theta}}) \quad \text{and} \quad \boldsymbol{y}|\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}, \sigma_\varepsilon^2 \mathbb{I}) \tag{13}$$

which defines the prior over the latent functions and the likelihood of the data, respectively. The generative model implies that the joint posterior over the latent parameters given as

$$p(\boldsymbol{f}|\boldsymbol{y}, X) = \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|X)}{p(\boldsymbol{y}|X)}. \tag{14}$$

Using the marginalization property for Gaussian distributions, the marginal distribution of $\boldsymbol{y}$ is given by

$$p(\boldsymbol{y}|X) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|X)d\boldsymbol{f} = \mathcal{N}(\boldsymbol{y}|0, K_{\boldsymbol{\theta}}). \tag{15}$$

Thus, the free hyperparameters $\boldsymbol{\theta}$ of the kernel can be optimized by the maxmimum marginal likelihood - or if we add priors on the $\boldsymbol{\theta}$ - by maximum a posteriori:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left(\log \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \boldsymbol{\theta} + \sigma_\varepsilon^2 \mathbb{I}) \right). \tag{16}$$

The log marginal likelihood is given as [Williams and Rasmussen, 2006]:

$$\log p(\boldsymbol{y}|X) = -\frac{1}{2}\boldsymbol{y}^\top (K_{\boldsymbol{\theta}} + \sigma_n^2 \mathbb{I})^{-1} \boldsymbol{y} - \frac{1}{2} \log |K_{\boldsymbol{\theta}} + \sigma_n^2 \mathbb{I}| - \frac{n}{2} \log 2\pi \tag{17}$$

A.2 Derivations of the entropy for a Gaussian distribution

The following derivation shows that the entropy of a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is proportional to the variance σ^2.

$$\begin{aligned}
H[X] &= \mathbb{E}\left[-\ln p(X)\right] && (18)\\
&= \mathbb{E}\left[\frac{1}{2}\ln 2\pi\sigma^2 + \frac{(X-\mu)^2}{2\sigma^2}\right] && (19)\\
&= \frac{1}{2}\ln 2\pi\sigma^2 + \frac{1}{2\sigma^2}\mathbb{E}\left[(X-\mu)^2\right] && (20)\\
&= \frac{1}{2}\ln 2\pi\sigma^2 + \frac{\sigma^2}{2\sigma^2} && (21)\\
&= \frac{1}{2}\ln 2\pi e\sigma^2, \quad \text{since } \frac{1}{2} = \frac{1}{2}\ln e && (22)\\
&\propto \sigma^2 && (23)
\end{aligned}$$

A.3 Derivations of BALD

BALD is given as [Houlsby et al., 2011]:

$$\begin{aligned}
I[\hat{\boldsymbol{\theta}}, y|\boldsymbol{x}, \mathcal{D}] &= H[\hat{\boldsymbol{\theta}}|\mathcal{D}] - \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}[H[\hat{\boldsymbol{\theta}}|y, \boldsymbol{x}, \mathcal{D}]] && (24)\\
&= H[y|\boldsymbol{x}, \mathcal{D}] - \mathbb{E}_{p(\hat{\boldsymbol{\theta}}|\mathcal{D})}[H[y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}]] && (25)
\end{aligned}$$

For a regular GP with a homoscedastic Gaussian likelihood ($\mathcal{N}(0, \sigma_\varepsilon^2)$), BALD is equivalent to ALM

$$\begin{aligned}
I[\hat{\boldsymbol{\theta}}, y|\boldsymbol{x}, \mathcal{D}] &= H[y|\boldsymbol{x}, \mathcal{D}] - \mathbb{E}_{p(\hat{\boldsymbol{\theta}}|\mathcal{D})}[H[y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}]] && (26)\\
&= H[y|\boldsymbol{x}, \mathcal{D}] - \mathbb{E}_{p(\boldsymbol{f}|\mathcal{D})}[H[y|\boldsymbol{x}, \boldsymbol{f}]] && (27)\\
&= \frac{1}{2}\ln(2\pi\sigma^2(\boldsymbol{x})) - \sigma_\varepsilon^2 && (28)\\
&\propto \sigma^2(\boldsymbol{x}) - const. && (29)
\end{aligned}$$

A.3.1 Fully Bayesian GP

For a Fully Bayesian GP (FBGP) with a homoscedastic Gaussian likelihood ($\mathcal{N}(0, \sigma_\varepsilon^2|\boldsymbol{\theta})$), there are three different ways to handle the hyperparameters Houlsby et al. [2011]:

1. The parameter of interest is $\boldsymbol{f}$, and the hyperparameters $\boldsymbol{\theta}$ are nuisance parameters,
2. The parameters of interest are both $\boldsymbol{f}$ and the hyperparameters $\boldsymbol{\theta}$,
3. The parameters of interest are the hyperparameters $\boldsymbol{\theta}$, and the parameter $\boldsymbol{f}$ is a nuisance parameter.

In the first and the second case, BALD is equivalent to B-ALM (here derived for case 1):

$$I[\boldsymbol{f}, \boldsymbol{\theta}, y|\boldsymbol{x}, \mathcal{D}] = H\left[\mathbb{E}_{p(\boldsymbol{f},\boldsymbol{\theta}|D)}[y|\boldsymbol{x}, \mathcal{D}, \boldsymbol{f}, \boldsymbol{\theta}]\right] - \mathbb{E}_{p(\boldsymbol{f}|\mathcal{D})}\left[H[\mathbb{E}_{p(\boldsymbol{\theta}|f,\mathcal{D})}[y|\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{\theta}]]\right] \tag{30}$$

$$= H\left[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[y|\boldsymbol{x}, \boldsymbol{\theta}]\right] - \mathbb{E}_{p(\boldsymbol{f}|\mathcal{D})}\left[H[\mathbb{E}_{p(\boldsymbol{\theta}|\boldsymbol{f},\mathcal{D})}[y|\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{\theta}]]\right] \tag{31}$$

For the first term in the equation above, $\boldsymbol{f}$ is integrated out such that the predictive posterior $p(y|\boldsymbol{x})$ now corresponds to the regular GP predictive posterior. Since the $\boldsymbol{f}$ is given by the outer expectation and y is dependent on $\boldsymbol{f}$ in the inner expectation, the second term is independent of $\boldsymbol{x}$. Hence, BALD is then given as:

$$I[\boldsymbol{f}, \boldsymbol{\theta}, y|\boldsymbol{x}, \mathcal{D}] \propto \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\sigma_{\boldsymbol{\theta}}^2(\boldsymbol{x})|\boldsymbol{\theta}] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\sigma_\varepsilon^2|\boldsymbol{\theta}]. \tag{32}$$

$$\propto \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\sigma_{\boldsymbol{\theta}}^2(\boldsymbol{x})|\boldsymbol{\theta}] - const. \tag{33}$$

In the third case, where the hyperparameters are the parameters of interest, BALD is given as

$$I[\boldsymbol{f}, \boldsymbol{\theta}, y|\boldsymbol{x}, \mathcal{D}] = H\left[\mathbb{E}_{p(\boldsymbol{f},\boldsymbol{\theta}|\mathcal{D})}[y|\boldsymbol{x}, \mathcal{D}, \boldsymbol{f}, \boldsymbol{\theta}]\right] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[H[\mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{\theta},\mathcal{D})}[y|\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{\theta}]]\right] \tag{34}$$

The expectation in the second term is now equal to the standard GP predictive posterior, and thus BALD is given as:

$$I[\boldsymbol{f}, \boldsymbol{\theta}, y|\boldsymbol{x}, \mathcal{D}] = H\left[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[y|\boldsymbol{x}, \mathcal{D}, \boldsymbol{\theta}]\right] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[H[y|\boldsymbol{x}, \boldsymbol{\theta}]\right] \tag{35}$$

A.4 Proof of moments for Gaussian Mixture Model

Using the MCMC samples, each prediction of the FBGP can be seen as a GMM, yielding the predictive posterior given as

$$p\left(y|\boldsymbol{x}, \mathcal{D}\right) = \int p\left(y|\boldsymbol{x}, \mathcal{D}, \boldsymbol{\theta}\right) p(\boldsymbol{\theta}|y) d\boldsymbol{\theta} \quad \simeq \quad \frac{1}{M}\sum_{j=1}^{M} p\left(y|\boldsymbol{x}, \mathcal{D}, \boldsymbol{\theta}_j\right), \quad \boldsymbol{\theta}_j \sim p(\boldsymbol{\theta}|\mathcal{D}) \tag{36}$$

This hierarchical predictive posterior is a mixture of M Gaussians with mean μ_{GMM} and variance matrix σ_{GMM}^2. The mean μ_{GMM} derived as:

$$\mu_{GMM}(y) = \mathbb{E}[y|\boldsymbol{x}] \tag{37}$$

$$= \mathbb{E}\left[\frac{1}{M}\sum_{j=1}^{M} p\left(y|\boldsymbol{x}, \boldsymbol{\theta}_j\right)\right] \tag{38}$$

$$= \mathbb{E}\left[\frac{1}{M}\sum_{j=1}^{M} \mathcal{N}(y|\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}), \sigma_{\boldsymbol{\theta}_j}^2(\boldsymbol{x}))\right] \tag{39}$$

$$= \frac{1}{M}\sum_{j=1}^{M} \mathbb{E}\left[\mathcal{N}(y|\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}), \sigma_{\boldsymbol{\theta}_j}^2(\boldsymbol{x}))\right] \tag{40}$$

$$= \frac{1}{M}\sum_{j=1}^{M} \mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) \tag{41}$$

The variance σ_{GMM}^2 is derived as:

$$\sigma_{GMM}^2(y) = \mathbb{E}\left[\,||y - \mathbb{E}[y|\boldsymbol{x}]||^2 \,|\, \boldsymbol{x}\right] \tag{42}$$

For short, we set $\bar{y} := \mu_{GMM}(y) = \mathbb{E}[y|\boldsymbol{x}]$

$$= \mathbb{E}\left[\, ||y - \bar{y}||^2 \,|\, \boldsymbol{x}\right] \tag{43}$$

$$= \int ||y - \bar{y}||^2 p(y|\boldsymbol{x})\, dy \tag{44}$$

$$= \int ||y - \bar{y}||^2 \frac{1}{M} \sum_{j=1}^{M} p\left(y|\boldsymbol{x}, \boldsymbol{\theta}_j\right)\, dy \tag{45}$$

$$= \frac{1}{M} \sum_{j=1}^{M} \int ||y - \bar{y}||^2 p\left(y|\boldsymbol{x}, \boldsymbol{\theta}_j\right)\, dy \tag{46}$$

$$= \frac{1}{M} \sum_{j=1}^{M} \int (y^2 + \bar{y}^2 - 2y\bar{y})\, \mathcal{N}\big(y|\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}), \sigma_{\boldsymbol{\theta}_j}(\boldsymbol{x})\big)\, dy \tag{47}$$

$$= \frac{1}{M} \sum_{j=1}^{M} \mathbb{E}_{\mathcal{N}\left(y|\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}), \sigma_{\boldsymbol{\theta}_j}(\boldsymbol{x})\right)}\big[y^2 + \bar{y}^2 - 2y\bar{y}\big] \tag{48}$$

If we set $\mathcal{N}_{\boldsymbol{\theta}} := \mathcal{N}\big(y|\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}), \sigma_{\boldsymbol{\theta}_j}(\boldsymbol{x})\big)$, then we can re-write it as:

$$= \frac{1}{M} \sum_{j=1}^{M} \mathbb{E}_{\mathcal{N}_{\boldsymbol{\theta}}}\big[y^2\big] + \bar{y}^2 + 2\bar{y}\, \mathbb{E}_{\mathcal{N}_{\boldsymbol{\theta}}}[y] \tag{49}$$

Using that $V[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, we get

$$= \frac{1}{M} \sum_{j=1}^{M} \Big(V_{\mathcal{N}_{\boldsymbol{\theta}}}[y] + \mathbb{E}_{\mathcal{N}_{\boldsymbol{\theta}}}[y]^2 + \bar{y}^2 - 2\bar{y}\, \mathbb{E}_{\mathcal{N}_{\boldsymbol{\theta}}}[y]\Big) \tag{50}$$

$$= \frac{1}{M} \sum_{j=1}^{M} \big(\sigma_{\boldsymbol{\theta}_j}(\boldsymbol{x}) + \mu_{\boldsymbol{\theta}_j}(\boldsymbol{x})^2 + \mu_{GMM}(\boldsymbol{x})^2 - 2\mu_{GMM}(\boldsymbol{x})\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x})\big) \tag{51}$$

$$= \frac{1}{M} \sum_{j=1}^{M} \big(\sigma_{\boldsymbol{\theta}_j}(\boldsymbol{x}) + ||\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) - \mu_{GMM}(\boldsymbol{x})||^2\big) \tag{52}$$

A.5 Posterior of the hyperparameters

For the one dimensional simulators, the resulting FBGP has two hyperparameters: the length scale and noise-term. For the d-dimensional simulators, a FBGP with an ARD RBF kernel has $d + 1$ hyperparameters, and thus we cannot plot the joint posterior for simulators with more than one input dimension. Instead we plot the joint posterior of the hyperparameter of a FBGP with an RBF kernel. The following posteriors are all averaged across the 10 experiments for a particular data size. In Figure 6, the five posteriors that gave a multimodal posterior using the RBF kernel are shown.

For the 3d simulator Ishigami and 6d simulator Hartmann, the RBF kernel gave only one mode using this visualization, see Figure 7. Due to the dimensionality, it is likely that with an ARD RBF kernel that the posterior will have more than one mode.

A.6 Motorcycle simulator

The motorcycle simulator is created by fitting a variational GP [Hensman et al., 2015] to the motorcycle accident data [Silverman, 1985], where the mean and standard deviation then is extracted. The simulator is fully specified by the following mean standard deviation.

```
mean = [0.5107, 0.5032, 0.4939, 0.4843, 0.4765, 0.4718, 0.4713,
        0.4748, 0.4805, 0.4856, 0.4870, 0.4824, 0.4724, 0.4604,
        0.4523, 0.4543, 0.4695, 0.4953, 0.5216, 0.5319, 0.5067,
```

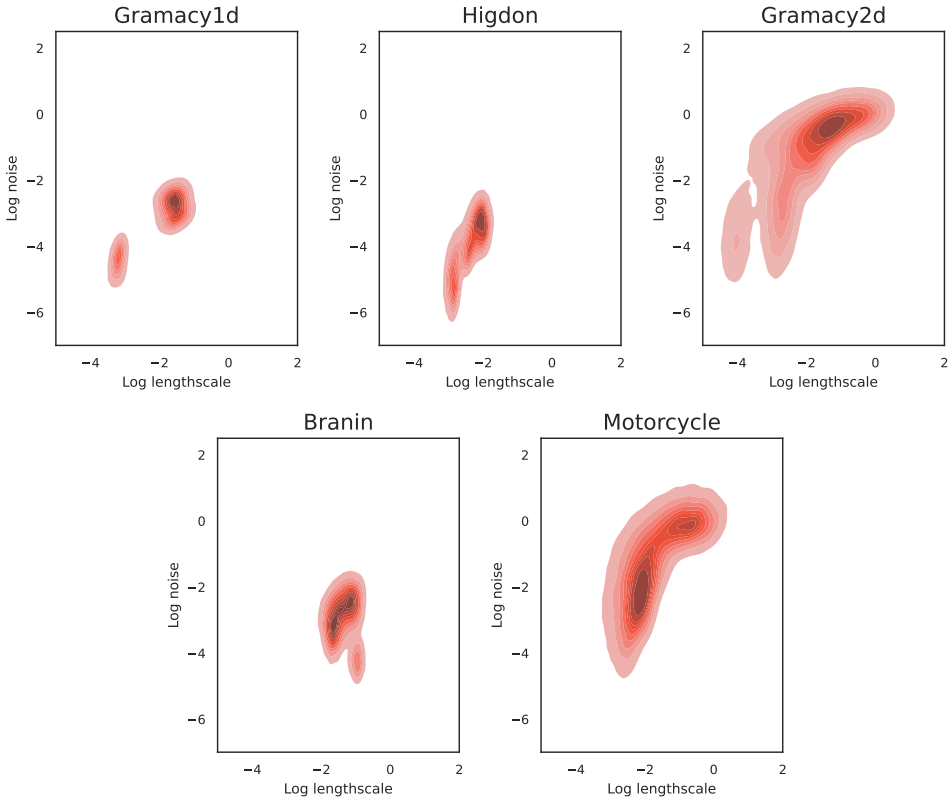

Figure 6: The joint posterior of the hyperparameters of a RBF GP fitted to data from the simulators. The data sizes are for Gramacy1d: 70, Higdon: 40, Gramacy2d: 40, Branin: 40, and Motorcycle: 10.

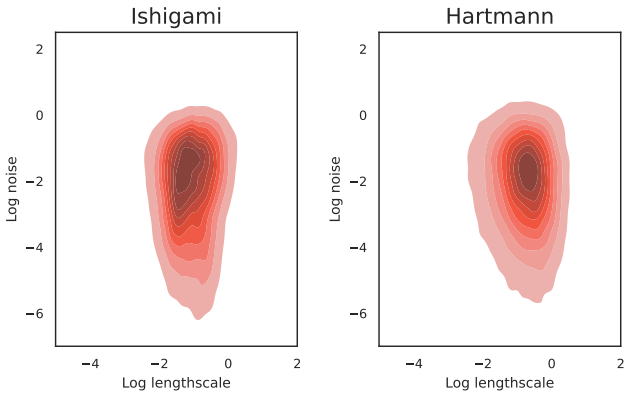

Figure 7: The joint posterior of the hyperparameters of a RBF GP fitted to 40 data points from the simulators.

```
0.4284, 0.2868, 0.0825,-0.1722,-0.4559,-0.7433,-1.0112,
-1.2435,-1.4339,-1.5851,-1.7049,-1.8015,-1.8788,-1.9333,
-1.9549,-1.9298,-1.8453,-1.6947,-1.4804,-1.2140,-0.9140,
-0.6013,-0.2950,-0.0080, 0.2536, 0.4900, 0.7046, 0.9006,
1.0778, 1.2317, 1.3542, 1.4362, 1.4710, 1.4573, 1.4012,
1.3147, 1.2139, 1.1143, 1.0277, 0.9589, 0.9056, 0.8605,
0.8146, 0.7605, 0.6959, 0.6241, 0.5533, 0.4935, 0.4538,
0.4395, 0.4501, 0.4795, 0.5173, 0.5507, 0.5678, 0.5600,
0.5241, 0.4632, 0.3869, 0.3092, 0.2456, 0.2098, 0.2103,
```

```
          0.2482, 0.3164, 0.4015, 0.4868, 0.5568, 0.6006, 0.6147,
          0.6033, 0.5769, 0.5487, 0.5311, 0.5324, 0.5549, 0.5950,
          0.6441, 0.6919, 0.7285]

stddev = [0.0477, 0.0400, 0.0463, 0.0514, 0.0529, 0.0502, 0.0438,
            0.0366, 0.0330, 0.0328, 0.0327, 0.0318, 0.0315, 0.0331,
            0.0363, 0.0433, 0.0560, 0.0702, 0.0780, 0.0733, 0.0615,
            0.0799, 0.1468, 0.2393, 0.3408, 0.4382, 0.5196, 0.5747,
            0.5959, 0.5800, 0.5295, 0.4540, 0.3712, 0.3067, 0.2822,
            0.2915, 0.3074, 0.3135, 0.3156, 0.3360, 0.3899, 0.4658,
            0.5385, 0.5873, 0.6045, 0.5975, 0.5860, 0.5901, 0.6133,
            0.6377, 0.6380, 0.5951, 0.5022, 0.3669, 0.2227, 0.1980,
            0.3483, 0.5319, 0.6956, 0.8186, 0.8886, 0.8987, 0.8470,
            0.7373, 0.5789, 0.3876, 0.1898, 0.1067, 0.2472, 0.3799,
            0.4620, 0.4847, 0.4492, 0.3643, 0.2463, 0.1253, 0.1068,
            0.2006, 0.2817, 0.3230, 0.3195, 0.2769, 0.2092, 0.1377,
            0.0911, 0.0856, 0.0919, 0.0940, 0.1116, 0.1568, 0.2117,
            0.2548, 0.2730, 0.2623, 0.2282, 0.1857, 0.1545, 0.1445,
            0.1445, 0.1437, 0.1542]
```