Estimation of simulation failure set with active learning based on Gaussian Process classifiers and random set theory

Morgane Menz, Miguel Munoz-Zuniga, Delphine Sinoquet

Estimation of simulation failure set with active learning based on Gaussian Process classifiers and random set theory

Morgane Menz[1*†], Miguel Munoz Zuniga[1†] and Delphine Sinoquet[1†]

[1]IFP Energies nouvelles, 1 et 4 avenue de Bois-Préau, Rueil-Malmaison, 92852, France.

*Corresponding author(s). E-mail(s): morgane.menz@ifpen.fr;
Contributing authors: miguel.munoz-zuniga@ifpen.fr;
delphine.sinoquet@ifpen.fr;
†These authors contributed equally to this work.

Abstract

A broad spectrum of applications require numerous simulations (such as optimization, calibration or reliability assessment for example) in various input sets. In that context, some simulation failures or instabilities can be observed, due for instance, to convergence issues of the numerical scheme of complex partial derivative equations. Most of the time, the set of inputs corresponding to failures is not known a priori and thus may be associated to a hidden constraint. Since the observation of a simulation failure regarding this unknown constraint may be as costly as a feasible expensive simulation, we seek to learn the feasible set of inputs and thus target areas without simulation failure before further analysis. In this classification context, we propose to learn the feasible domain with an adaptive Gaussian Process Classifier. The proposed methodology is a batch mode active learning classification strategy based on a Stepwise Uncertainty Reduction of random sets derived from the Gaussian Process Classifiers. The performance of this strategy will be presented on different hidden-constrained problems and in particular within a wind turbine simulator-based reliability analysis.

Keywords: Gaussian Process; Classification; Stepwise Uncertainty Reduction; Batch; Hidden Constraints; Failure; Simulator; Random set

1 Introduction

Nowadays, the design of engineering systems implies the use of numerical models, involving complex physical systems with non-linear behaviours and numerous design and uncertain variables. Taking into account uncertainties, for instance, in the context of uncertainty propagation (Sudret, 2007) or robust design (Moustapha et al, 2022) can require numerous simulations of these numerical models and become computationally very expensive.

Moreover, some combinations of values of the input variables $x \in \Omega$ can lead to simulation instabilities or failures (Digabel and Wild, 2015). Indeed, the numerical simulator f can for instance fail to converge due to instabilities in the numerical scheme of complex partial derivative equations (e.g. time step smaller than the critical step) or to an inadequate mesh (e.g. not fine enough in areas of high-stress concentration) and sometimes computed values can also tend to infinity (e.g. species concentration outputs in chemical reaction simulations). Such cases have been reported in the literature in different fields such as offshore wind turbine design (Poirette et al, 2017), styrene chemical compound production management (Audet et al, 2008) or the simulation of fuel-coolant interaction in severe nuclear accident (Hakimi et al, 2022). In that respect, we are interested in the identification of the subset of the parameter space where simulations succeed. This feasible set $\Gamma^* \subset \Omega$ can be formalized by

$$\Gamma^* = \{x \in \Omega \ / \ y(x) = 1\}, \tag{1}$$

where $y : \Omega \to \{0, 1\}$ and $y(x)$ takes the value 0 when the simulator fails to provide a converged output and the value 1 when the simulation succeeds.

However, it is difficult and most often impossible for the expert to define a priori the feasible set of inputs i.e. inputs corresponding to converged simulation and therefore the function y is a priori unknown. In the field of optimization, belonging to this feasible set is commonly referred to as verifying a hidden constraint (Digabel and Wild, 2015).

Since the observation of a simulation failure $y(x) = 0$ might be as costly as a feasible simulation, it is useful to assess the convergence domain likeliness.

Hence, we seek to learn the feasible set of inputs in order to target areas without simulation failure during processes such as optimization, active metamodel learning, reliability assessment or more basically to study code performances.

In order to learn the hidden set of input-values leading to failures, we consider in this study that only binary observations corresponding to failure or non-failure status are available. Indeed we do not consider partial failure/convergence of the simulator nor any smoothness or regularity hypothesis between the failure and feasible areas.

It is actually a binary classification problem and thus classification modeling techniques can be used to approximate the failure likelihood. Moreover, adaptive approaches, also known as active learning in the Machine Learning

(ML) community, are one promising way to reduce the number of expensive simulations for classifier construction.

Active learning for classification has been a prolific subject of interest in ML and the object of extensive surveys as in Settles (2009); Kumar and Gupta (2020). Among the large spectrum of available approaches, one-step-look-ahead methods based on the Expected Error Reduction (EER) have recently presented promising results in comparison with published strategies (Zhao et al, 2021b,a), especially when coupled with a Gaussian process classifier (GPC) Kapoor et al (2007); Freytag et al (2014); Zhao et al (2021a). Indeed, coupling both GPC and EER enables to take advantage of the flexibility of the non-parametric GP model and the goal-oriented EER formulation of the classification problem. However EER-based learning criteria might be numerically costly due to the repeated training of the predictive posterior. To overcome this issue, fast update of the posterior predictor without retraining the GP model has been proposed for instance in Zhao et al (2021a). Notice that, in the Uncertainty Quantification community, EER approaches have been independently introduced as Stepwise Uncertainty Reduction (SUR) which will be introduced in section 3. From another perspective, Adcock et al (2023) proposed an adaptive strategy to learn hidden constraints by sequentially sampling the domain with respect to a measure which support is updated according to Christoffel function estimations (Nevai, 1986). However, this method can present high rejection rates and therefore not suitable in our context of expensive simulations.

In this paper, we propose a novel one-step-look-ahead GPC active learning strategy based on random set theory to assess the feasible domain. Indeed, Gaussian process models have the advantage of providing a measure of uncertainty on the (class) prediction and thus allowing the definition of enrichment criteria naturally derived from this uncertainty measure. This latter makes easier the introduction of an exploration/exploitation tradeoff in the active learning criterion and therefore presents a clear advantage against methods using the SVC or other models which focuses mainly on exploitation (Tong and Koller, 2001; Roy and McCallum, 2001; Moskovitch et al, 2007). The proposed methodology is an adaptation of the Stepwise Uncertainty Reduction strategies introduced and used for excursion set estimation with Gaussian Process Regression as in Bect et al (2012); Chevalier (2013); Duhamel et al (2023). We propose to extend this latter to the classification setting with GPC models based on signs (Bachoc et al, 2020). In terms of improvement/learning criterion, our approach generalizes the Mean Objective Cost of Uncertainty (MOCU) method presented in Zhao et al (2021b) and will be compared to SMOCU an improved version of MOCU introduced in the same paper.

The article is organized as follows, first, we provide in Section 2 a presentation of the GPC model formulation. Then in Section 3, we recall the Stepwise Uncertainty Reduction (SUR) paradigm and propose a (batch mode) learning criterion, from the random set theory, for classification. Section 4 is dedicated to an algorithmic presentation of our contribution: an active learning strategy

to recover hidden sets by a SUR method with a fast-to-update GPC model. Finally, we present the performance of our methodology on three application examples in Section 5. The two first applications are analytical cases and the third one is an industrial case targeting the simulation-based estimation of the cumulated damage of a wind turbine subject to environmental conditions.

2 The Gaussian process classification model

In probabilistic binary classification, only binary values $y_i \in \{0, 1\}$ are observed. Given these observations, one would like to predict the membership probability of a new point to the class of interest, class "1" indicating feasibility (convergence of the simulation). GP based classification models propose to use a latent random process that describes the class membership.

The classical GPC model is described in Rasmussen and Williams (2006); Nickisch and Rasmussen (2008). In this model, the latent distribution posterior is non-gaussian and several approaches have been proposed to build a Gaussian approximation of the posterior (such as the Laplace approximation, Expectation-Propagation, the Kullback-Leibler method and variational method). Nickisch and Rasmussen (2008) give an overview of all these methods. The authors conclude that the Expectation-Propagation (EP) approximation, although numerically expensive, is the best one in terms of accuracy.

More recently, another GP based classification model has been proposed Bachoc et al (2020), that consists in conditioning the latent GP on the sign observations characterizing the belonging to a class. This model has the advantage to come with theoretical guarantees, unlike the first model that implies Gaussian approximation of the posterior. In the rest of the section, we will present the formulation of this model on which our active learning strategy (presented in Section 3) is based.

Let Z be a GP characterized by a constant mean function $\mu_Z \in \mathbb{R}$ and a stationary kernel $k_\theta^Z(\cdot, \cdot)$ on Ω^2 with hyperparameters $\theta \in \Theta \subset \mathbb{R}^d$. The posterior distribution of Z knowing the observations $\{\mathcal{X}_n = (x_1, \ldots, x_n), \mathbf{z_n} = (z_1, \ldots, z_n)\}$ is Gaussian with posterior mean $m_n(\cdot, \mathbf{z_n})$ and covariance $k_n(\cdot, \cdot)$ given for $x, x' \in \Omega^2$ by

$$m_n(x, \mathbf{z_n}) = \mu_Z + k_\theta^Z(x) K^{-1}(z_n - \mu_Z) \tag{2}$$

$$k_n(x, x') = k_\theta^Z(x, x') + k_\theta^Z(x) K^{-1} k_\theta^Z(x') \tag{3}$$

with $K = k_\theta^Z(\mathcal{X}_n, \mathcal{X}_n)$ the covariance matrix between the observations and $k_\theta^Z(x) = (k(x, x_1), \ldots, k(x, x_n))$. In addition, we denote by $\sigma_n^2(x) = k_n(x, x)$ the posterior variance of the GP.

In probabilistic binary classification, we don't have access to the realizations $\mathbf{z_n}$ of the GP Z but only to binary observations $\mathcal{Y} = (y_1, \ldots, y_n) \in \{0, 1\}^n$ with $\forall i$, $y_i = \mathbb{1}_{x_i \in \Gamma^*}$.

The GPC modeling of the feasibility (or non-failure) probability proposed in Bachoc et al (2020) consists in defining a latent GP Z and predicting for any $x \in \Omega$, $Y(x) = \mathbb{1}_{Z(x)>0}$ given $\mathbf{Y_n} = (\mathbb{1}_{Z(x_1)>0}, \ldots, \mathbb{1}_{Z(x_n)>0})$: the 0-1 encoding of the sign of Z(x) knowing the sign of $\mathbf{Z_n} = (Z(x_1), \ldots, Z(x_n))$.

The feasibility probability is then defined as

$$p_n(x) = \mathbb{P}[Z(x) > 0 | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}]. \tag{4}$$

The feasibility probability can actually be expressed as follows

$$p_n(x) = \int_{\mathbb{R}^n} \bar{\Phi}\left(\frac{-m_n(x, \mathbf{z_n})}{\sigma_n(x)}\right) \phi_n^{\mathbf{Z_n}}(\mathbf{z_n})\, \mathrm{d}\mathbf{z_n}, \tag{5}$$

where $\phi_n^{\mathbf{Z_n}}$ is the density function truncated Gaussian conditioned probability of the $\mathbf{Z_n}$ restricted to respect $\mathbf{Y_n} = \mathcal{Y}$ and $\bar{\Phi}$ is defined as:

$$\bar{\Phi}(\frac{a}{b}) = \begin{cases} 1 - \Phi(\frac{a}{b}) & \text{if } b \neq 0 \\ \mathbb{1}_{-a>0} & \text{if } b = 0 \end{cases} \tag{6}$$

with Φ is the cumulative density function (c.d.f.) of the standard normal distribution.

The value of $p_n(x)$ can be approximated by a Monte Carlo method as

$$\widehat{p}_n(x) = \frac{1}{N} \sum_{i=1}^{N} \bar{\Phi}\left(\frac{-m_n(x, \mathbf{z_n^i})}{\sigma_n(x)}\right), \tag{7}$$

where $(\mathbf{z_n^i})_{i=[1,N]}$ are N realizations of the latent random vector $\mathbf{Z_n}$ following the truncated Gaussian distribution $\phi_n^{\mathbf{Z_n}}$. As the sampling of the realizations $(\mathbf{z_n^i})_{i=[1,N]}$ is independent of x, it can thus be generated only once with a rejection strategy or more advanced ones dedicated to sample from truncated multivariate Gaussian distributions (Botev, 2017; Pakman and Paninski, 2014).

The hyperparameters μ_Z and θ of the latent GP must be estimated to sample realizations at observation points and approximate the probability of failure at any point. They are estimated by maximizing the likelihood leading to the estimators

$$(\hat{\mu_Z}, \hat{\theta}) = \underset{\mu_Z, \theta}{\arg\max}\, \mathbb{P}_{\mu_Z, \theta}\left[\mathbf{Y_n} = \mathcal{Y}\right]. \tag{8}$$

This likelihood is a Gaussian orthant probability that can be estimated by Monte Carlo based methods (Genz and Bretz, 2009; Botev, 2017; Azzimonti and Ginsbourger, 2018).

Note that the GPC model does not provide an analytical posterior GP mean in the sense that the latent GP posterior mean $m_n(x, \mathbf{Z_n})$ actually depends on the random vector $\mathbf{Z_n}$ of the latent GP at the observation points $\mathcal{X}_n$.

Hence, we can not make use of existing learning functions in GP regression that depend directly on the posterior GP mean and covariance function. In the classification setting, we are then restricted to learning strategies that rely on the feasibility probability $p_n(x)$ and the Stepwise Uncertainty Reduction (Bect et al, 2012) paradigm is well suited for this type of problem.

3 Stepwise Uncertainty Reduction strategy for hidden failure learning

3.1 The Vorob'ev deviation uncertainty measure from random set theory

In this section, we provide some useful notions from random set theory (Molchanov, 2005; Vorobyev and Lukyanova, 2013) that allow the definition of a measure of uncertainty on a random set Γ. These notions have already been used to define SUR strategies for excursion set estimation (Chevalier, 2013; El Amri et al, 2020; El Amri et al, 2023). Here, our aim is to introduce these random set notions in the context of GPC modelling introduced in the previous section.
In our framework, we are only interested in random set fully characterized by a given stochastic process. Let us consider

$$\begin{aligned} \Gamma &:= \{x \in \Omega \ / \ Y(x) = 1\} \\ &= \{x \in \Omega \ / \ Z(x) > 0\}, \end{aligned} \tag{9}$$

with Z the latent GP introduced in section 2 and $Y(x)$ a Bernouilli random variable with mean $p_0(x) = \mathbb{P}(Z(x) > 0)$. Γ is a random set modelling a prior on the set Γ^*. We then consider the posterior GP conditioned on failure/non-failure observations modelled with the 0-1 Bernouilli variable Y associated with the sign of Z. In this context, we are interested in the posterior random set induced by conditioning on Y observations. The conditional feasibility function $p_n(x)$ introduced in (4-5), also known as coverage function, is defined with respect to the posterior latent GP model.
Let us then define Q_α the (posterior) α-percentiles of Γ, that are actually linked to the level sets of $p_n(x)$, as

$$Q_\alpha = \{x \in \Omega : p_n(x) \geq \alpha\}, \alpha \in]0, 1]. \tag{10}$$

The posterior Vorob'ev expectation (Vorobyev and Lukyanova, 2013) is defined as the α^*-percentile of Γ, where α^* is chosen so that the volume of the defined set, Q_{α^*}, equals the expected volume of Γ

$$\mathbb{E}[\mu(\Gamma)|\mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}] = \mu(Q_{\alpha^*}), \tag{11}$$

with μ the Lebesgue measure. In practice, α^* can be obtained by a dichotomy method (El Amri, 2019).

The Vorob'ev expectation is actually a global minimizer (see proof in Molchanov (2005)), among closed sets of volume equal to the mean volume of Γ, of the Vorob'ev deviation given by

$$Var_n(\Gamma) = \mathbb{E}[\mu(Q_\alpha \Delta \Gamma) | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}], \tag{12}$$

with $\Gamma \Delta Q_\alpha = (\Gamma \setminus Q_\alpha) \cup (Q_\alpha \setminus \Gamma)$ the symmetric difference between the two sets Γ and Q_α. In Chevalier (2013), mainly by interverting the expectation and the integral involved in the volume calculation, one can derive the following expression of the Vorob'ev deviation:

$$\begin{array}{l} Var_n(\Gamma) \\ = \int (1 - p_n(x)) \mathbb{1}_{p_n(x) \geq \alpha^*} \mu(dx) \\ + \int p_n(x) \mathbb{1}_{p_n(x) < \alpha^*} \mu(dx) \end{array} \tag{13}$$

with p_n defined by (4), α^* solution of (11).

3.2 Stepwise Uncertainty Reduction strategy

A Stepwise Uncertainty Reduction (SUR) strategy aims to sequentially choose a sequence of learning points in order to reduce the expected future uncertainty on a quantity of interest. In the classification problem, our quantity of interest is the feasible set Γ^* which uncertainty is modeled with the prior random set Γ and its posterior counterpart previously introduced.

Let $\mathcal{U}_n$ be a measure of uncertainty on the random set Γ knowing observations at points $\mathcal{X}_n$. A SUR strategy for this uncertainty measure consists in finding at each step the point x^*_{n+1} such that

$$x^*_{n+1} = \underset{x_{n+1} \in \Omega}{\arg\min}\, J_n(x_{n+1}), \tag{14}$$

with $J_n(x_{n+1}) = \mathbb{E}_n[\mathcal{U}_{n+1}]$ and $\mathbb{E}_n[\cdot]$ corresponds to a conditional expectation on the stochastic process used to model Γ. Therefore, in our setting, the criterion has the following dependence

$$J_n(x_{n+1}) = \mathbb{E}_n\left[\mathcal{U}_{n+1}(x_{n+1}, Y(x_{n+1}))\right], \tag{15}$$

where the expectation is with respect to the posterior $Y(x_{n+1}) | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}$ (characterized by the posterior latent GP) i.e.

$$\mathbb{E}_n[\cdot] = \mathbb{E}_{Y(x_{n+1})}[\cdot | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}]. \tag{16}$$

3.3 SUR active learning criterion with Bernouilli model

In the regression setting investigated in Chevalier (2013), the author also considered a SUR strategy based on random set theory with Vorob'ev deviation as uncertainty measure. Nevertheless, in the regression context, continuous

observations are available such that the GP is explicitly conditioned. In our GPC setting, where the GP is latent and conditioned on sign observations, we directly extend this approach by considering the Vorob'ev deviation defined in Eq. (12) as uncertainty measure and the more complex p_n given by (4), involving an additional integration with respect to the latent GP, such that for any $y_{n+1} \in \{0, 1\}$

$$\begin{array}{l} \mathcal{U}_{n+1}(x_{n+1}, y_{n+1}) \\ = Var_{n+1}(\Gamma) \\ = \int (1 - p_{n+1}(x)) \mathbb{1}_{p_{n+1}(x) \geq \alpha^*} \mu(dx) \\ + \int p_{n+1}(x) \mathbb{1}_{p_{n+1}(x) < \alpha^*} \mu(dx) \end{array} , \tag{17}$$

where $p_{n+1}(x) = \mathbb{P}[Z(x) > 0 | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}, Y(x_{n+1}) = y_{n+1}]$. In the regression setting p_{n+1} simply involves the Gaussian cumulative function but no integral. In our classification context, because of the latent nature of Z, p_{n+1} is an integral w.r.t. $\phi_{n+1}^{\mathbf{Z_{n+1}}}$ the density function of the truncated Gaussian conditioned probability $\mathbf{Z_{n+1}}$ restricted to respect $\mathbf{Y_n} = \mathcal{Y}$ and $Y(x_{n+1}) = y_{n+1}$.

Let us now consider the random set Γ in terms of the Bernoulli random process Y as defined in Eq. (9). We can then derive the posterior distribution of $Y_n(x)$ representing the conditional Bernoulli random variable $Y(x) | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}$ (Dai et al, 2013). Indeed straightforwardly, $Y_n(x)$ is also a Bernoulli random variable with mean $p_n(x)$ given in (4).
Then, the expression (15) of $J_n(x_{n+1})$ using the expectation with respect to the Bernoulli process $Y_n(x_{n+1})$ writes

$$\begin{array}{rl} J_n(x_{n+1}) = & (1 - p_n(x_{n+1}))\, \mathcal{U}_{n+1}(x_{n+1}, 0) \\ & + p_n(x_{n+1})\, \mathcal{U}_{n+1}(x_{n+1}, 1). \end{array} \tag{18}$$

Finally, our SUR strategy boils down to iteratively minimizing $J_n(x_{n+1})$ (18) with uncertainty measure (17).

Note that the calculation of $p_{n+1}(x)$ involves potentially costly conditional simulations. In the regression setting, with the use of GP update formulae, Chevalier (2013) proposed closed-form expressions for the coverage probability $p_{n+1}(x)$. Unfortunately, these latter are not applicable in our GPC context due to the form of the coverage probability that is expressed as a mean of standard normal c.d.f. Nevertheless, also using GP update formulae, we propose in section 4 an alternative formulation to mitigate the expectation computation cost.

3.4 Batch version of the active learning criterion

The criterion introduced in the previous section is here extended to a parallel version that allows evaluating several learning points simultaneously at each improvement iteration. Indeed, when the criterion optimization is perfectly solved and has a negligible computational cost compared to a simulation then this parallelisation enables accuracy enhancement. If one of the two previous

conditions is not met, a compromise needs to be found between the accuracy and the computational cost of the learning stage. This latter point will be discussed in section 4.

Let denote in the sequel $x^q = (x_{n+1}, \ldots, x_{n+q}) \in \Omega^q$ a batch of $q \geq 1$ candidate points at step n and let use the same notation $x^i = (x_{n+1}, \ldots, x_{n+i})$ for any $1 \leq i \leq q$. Denoting $Y_n(x)$ the conditional Bernouilli random variable $Y(x)|\mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}$, then the random vector $Y_n(x^q) = (Y_n(x_i))_{i=n+1,\ldots,n+q}$ follows a multivariate Bernoulli distribution with, for all i, $Y_n(x_i)$ a Bernoulli random variable with probability $p_n(x_i)$. In this setting, we define the uncertainty measure for a batch of points as

$$\begin{array}{l} \mathcal{U}_{n+q}(x^q, y^q) \\ = Var_{n+q}(\Gamma) \\ = \int (1 - p_{n+q}(x)) \mathbb{1}_{p_{n+q}(x) \geq \alpha^*} \mu(dx) \\ + \int p_{n+q}(x) \mathbb{1}_{p_{n+q}(x) < \alpha^*} \mu(dx), \end{array} \tag{19}$$

where $p_{n+q}(x)$ is the updated feasibility probability estimated with the updated GPC for the considered outcome $y^q = (y_{n+1}, \ldots, y_{n+q}) \in \{0,1\}^q$, i.e.

$$p_{n+q}(x) = \mathbb{P}[Z(x) > 0 | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}, x^q, Y(x^q) = y^q]. \tag{20}$$

Then, the expression of the parallel criterion boils down to

$$\begin{array}{rl} J_n(x^q) & = \mathbb{E}_{Y(x^q)} \left[\mathcal{U}_{n+q}(x^q, Y(x^q)) | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y} \right] \\ & = \displaystyle\sum_{y^q \in \{0,1\}^q} \mathcal{U}_{n+q}(x^q, y^q) p(y^q), \end{array} \tag{21}$$

where $y^q = (y_{n+1}, \ldots, y_{n+q}) \in \{0,1\}^q$ are the possible outcomes of $Y_n(x^q)$ and the probability of these outcomes is thanks to the Bayes formula

$$\begin{array}{rl} p(y^q) & = \mathbb{P}(Y(x^q) = y^q | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}) \\ & = \prod_{i=n+1}^{n+q} p_n(y_i | y^{i-1}), \end{array} \tag{22}$$

where for $i = n+1$ to $n+q$

$$\begin{array}{rl} p_n(y_i | y^{i-1}) & = \mathbb{P}[Y(x_i) = y_i | \mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}, x^{i-1}, Y(x^{i-1}) = y^{i-1}] \\ & = p_{i-1}(x_i)^{y_i} (1 - p_{i-1}(x_i))^{1-y_i}. \end{array} \tag{23}$$

Note that the number of terms of the sum in (21) increases exponentially with q. Accordingly, the computation time of the criterion (21), and even more its optimization (14), grows rapidly with the increase of q. For this reason, a greedy strategy will be discussed in the next section.

4 ARCHISSUR algorithm

For the classification problems that we wish to tackle, in the more general batch-setting, we propose an Active Recovery of a Constrained and Hidden

Set by SUR (ARCHISSUR) method with learning criterion Eq. (21) developed in Section 3.4. In this section, we first introduce some tricks to reduce the computational cost of the method followed by the overall algorithm.

The considered learning function (21) coupled with (19) and (22) depends on the future feasibility probability p_{n+q} that can be estimated, given the estimator of the conditional feasibility probability expression in Eq. (7), by

$$\widehat{p}_{n+q}(x) = \frac{1}{N}\sum_{i=1}^{N} \bar{\Phi}\left(\frac{-m_{n+q}(x, \mathbf{z^i_{n+q}})}{\sigma_{n+q}(x)}\right), \tag{24}$$

where $(\mathbf{z^i_{n+q}})_{i=[1,N]}$ are N realizations of the latent random vector $\mathbf{Z_{n+q}} = (Z(x_1), \ldots, Z(x_n), Z(x_{n+1}), \ldots, Z(x_{n+q}))$ sampled from the truncated distribution $\phi_n^{\mathbf{Z_{n+q}}}(\mathbf{z_{n+q}})$ and $m_{n+q}(x, \mathbf{z^i_{n+q}})$, $k_{n+q}(x, x')$ are respectively the updated mean and covariance function.

For each q-batch of points x^q, the evaluation of the criterion given by (21) (resp. (18) if $q = 1$) involves 2^q distinct updates of the GPC $Y_n(x)$ for the 2^q possible outcomes of $Y_n(x^q)$. The use of a crude GPC update with $((\mathcal{X}_n, x^q), (\mathcal{Y}, Y(x^q)))$ would require generating 2^q times realizations of a truncated (n+q)-dimensional normal distribution, leading to an untractable criterion. Hence, to allow fast computation of the criterion we propose to use the GP update formula provided in Chevalier (2013) so that we avoid sampling the whole random vector $\mathbf{Z_{n+q}}$.

The updated mean m_{n+q} and covariance function k_{n+q} of the GPC are then

$$\begin{aligned} m_{n+q}(x, \mathbf{z^i_{n+q}}) &= m_n(x, \mathbf{z^i_n}) \\ &\quad + \lambda_{new}(x)^T \left(\mathbf{z^{q,i}} - m_n(x^q, \mathbf{z^i_n})\right), \end{aligned} \tag{25}$$

$$k_{n+q}(x, x') = k_n(x, x') - k_n(x, x^q)^T K_{new}^{-1} k_n(x', x^q) \tag{26}$$

with $\lambda_{new}(x) = K_{new}^{-T} k_n(x, x^q) \in \mathbb{R}^q$, $K_{new} = k_n(x^q, x^q) \in \mathbb{R}^{q\times q}$ and $\mathbf{z^{q,i}} = (z^i_{n+1}, \ldots, z^i_{n+q})$.

Moreover, $\mathbf{z^{q,i}}$ follows the truncated multivariate normal distribution given $\mathbf{z^i_n}$

$$\mathbf{z^{q,i}} \sim \mathcal{N}\left(m_n(x^q, \mathbf{z^i_n}), K_{new}\right) \tag{27}$$

such that $(\mathbb{1}_{z^i_{n+1}>0}, \ldots, \mathbb{1}_{z^i_{n+q}>0}) = \mathbf{y^q}$.

Thus, only one realization of the random vector $\mathbf{z^{q,i}}$ knowing the $\mathbf{z^i_n}$ has to be sampled to compute $m_{n+q}(x, \mathbf{z^i_{n+q}})$. Moreover, if the batch size q is small, $\mathbf{z^{q,i}}$ can be simply sampled by rejection at low cost without requiring a dedicated truncated multivariate distribution improved sampling algorithm such as (Botev, 2017; Pakman and Paninski, 2014).

The ARCHISSUR learning function evaluation algorithm for a batch of points x^q is provided in Algorithm 1.

Despite the proposed tricks and formulae to decrease the numerical costs, the batch version numerical cost grows very fast with the increase of the

Algorithm 1 ARCHISSUR criterion computation

Require: n observations of the binary constraint, N samples $(\mathbf{z_n}^i)_{i=1,\dots,N}$ of the latent GP Z at $\mathcal{X}_n$, Z_n
Require: a batch of locations $\mathbf{x^q}$
for $y^q \in \{0,1\}^q$ **do**
  for $j = 2, \dots, q$ **do**
    for $i = 1, \dots, N$ **do**
      Generate a sample z^i_{n+j-1} by applying Eq. (27) (with $q \leftarrow 1$ and $n \leftarrow n+j-1$)
      Do the concatenation: $\mathbf{z^i_{n+j-1}} \leftarrow (\mathbf{z^i_{n+j-2}}, z^i_{n+j-1})$
    end for
    Compute $\sigma_{n+j-1}(x_{n+j})^2$ and $m_{n+j-1}(x_{n+j}, \mathbf{z^i_{n+j-1}})$ following Eq. (26)
    Compute $\hat{p}_{n+j-1}(x_{n+j})$ following Eq. (24)
    Compute $p_n(y_{n+j}|\mathbf{y^{n+j-1}})$ following Eq. (23)
  end for
  $p(\mathbf{y^q}) \leftarrow \prod_{j=1}^{q} p_n(y_{n+j}|\mathbf{y^{n+j-1}})$
  for $i = 1, \dots, N$ **do**
    Generate a sample z^i_{n+q} following Eq. (27)
    $\mathbf{z^i_{n+q}} \leftarrow (\mathbf{z^i_{n+q-1}}, z^i_{n+q})$
  end for
  Estimate $\mathcal{U}_{n+q}(\mathbf{x^q}, \mathbf{y^q})$ according to Eq. (19) with a Monte Carlo sampling of size M w.r.t. the distribution μ on Γ
end for
$J_n(\mathbf{x^q}) \leftarrow \sum_{\mathbf{y^q} \in \{0,1\}^q} \mathcal{U}_{n+q}(\mathbf{x^q}, \mathbf{y^q}) p(\mathbf{y^q})$

problem dimension and the number of batch points q. Indeed, it involves an NP-hard, non-linear, multivariate minimization of $J_n(x^q)$ that can be very difficult to tackle. In practice, as pointed out for instance in Krause et al (2008), a sub-optimal greedy version of the batch optimization problem is preferred, consisting in adding iteratively one point at a time. By using the latter strategy, the optimization problem that is solved at each iteration becomes numerically affordable. Another numerical advantage to mention concerns the sampling of $\mathbf{z^{q,i}}$ that is also done one point at a time in the greedy version.

Taking this latter into account and from our numerical tests, we recommend using the greedy version when batches of size greater than 3 or 4 are needed. In the numerical section, with the non-greedy strategy, we achieved good performances in terms of computational time and points selection with $q = 2$.

4.1 Link between ARCHISSUR and MOCU active learning criterion

Some other GPC based active learning methods have already been proposed. In particular, the Mean Objective Cost of Uncertainty, MOCU, method (Zhao et al, 2021b) proposes a learning function, given in Eq. (A1), which is defined as the increase of the classification error due to the model uncertainty. There is a relationship between the MOCU criterion and the one we proposed that, to our knowledge, has never been presented before. Indeed, in appendix A a few derivations enable the reinterpretation of the MOCU learning function expression Eq. (A1) as:

$$U^{MOCU}(x_{n+1}) = \mathbb{E}_n[Var_{n+1}(\Gamma)] - Var_n(\Gamma) \tag{28}$$

where $Var_n(\Gamma)$ is given by (12) with $\alpha = 1/2$.

In that respect, it can be interpreted as a particular case of the Vorob'ev based SUR strategy for $q = 1$ with a fixed level set α equals to $1/2$, i.e. with the Vorob'ev median instead of the value α^* solving (11).

Note that a smooth concave approximation of the MOCU function Soft-MOCU has also been proposed in Zhao et al (2021b) in order to improve efficiency in the long run, i.e. for the remaining iterations when the model has already been learned quite well, as the concavity enables the detection of small model changes. In fact, this function is obtained by using a smooth approximation of the maximum function (see the original expression of MOCU (A1)) by the nested log of the sum of exponentials, generally named LogSumExp (or RealSoftMax). Moreover, GPC updates formulas with EP-approximation (Nickisch and Rasmussen, 2008) are provided in Zhao et al (2021a). In the numerical applications, we will compare the results obtained with ARCHISSUR to the ones achieved using NR-SMOCU: the SMOCU learning function with no GPC retraining (no computation of the covariance matrix and its inverse for the $n + 1$ observations), using update formula.

5 Applications

5.1 Methodology settings and comparison measures

The GPC active learning method actually provides a random set whose uncertainty has been reduced as a result. Then, a deterministic classifier must be chosen to characterize the feasible set. This can be done by choosing a statistical moment of the random set. A natural choice is then the Vorob'ev expectation which minimizes the symmetric difference to the random set Γ.

The performances of the different learning strategies can be assessed by using different error measures on the built classifiers when we have access to the real feasible set (for analytic examples) by means of a grid of validation point

- relative error on the feasible set:

$$\begin{aligned} crit_F &= \frac{\mu(\Gamma^* \Delta Q_{\alpha^*})}{\mu(\Gamma^*)} \\ &= \frac{FN+FP}{TP+FN} \end{aligned} \tag{29}$$

- the true positives and negatives rates:

$$crit_P = \frac{TP}{TP+FN} \tag{30}$$

$$crit_N = \frac{TN}{TN+FP}, \tag{31}$$

where TP, TN hold respectively for the number of true positives and negatives, i.e. the number of correctly predicted validation points in each class, and FP, FN for the number of false positives and negatives, i.e. the number of mispredicted validation points in each class.

In practice, the stopping criterion proposed in El Amri et al (2020) for SUR strategies can be adopted for ARCHISSUR. Indeed, this criterion is based on the Vorob'ev deviation that is available at each step of the algorithm. This stopping criterion is given by

$$\forall \quad 0 \leq j \leq l, \quad \Delta^j \leq \epsilon \tag{32}$$

with $\Delta^j = |Var_{j-1}(\Gamma) - Var_j(\Gamma)|$ the absolute error on Vorob'ev deviations between two consecutive iterations, ϵ a tolerance value and l the number of steps during which Δ^j must be smaller than the ϵ.

The application of GP based active learning methods is usually done by updating GP hyperparameters by MLE at each iteration. However, we noticed that ARCHISSUR has better exploration properties when the hyperparameters are constant for a certain number of iterations. Moreover, the Vorob'ev deviation evolution is smoother when fixing hyperparameters, which allows the use of the stopping criterion presented above. Hence, we recommend fixing the hyperparameters for a certain number of iterations and updating them periodically by MLE considering the new observations.

We benchmark the proposed ARCHISSUR strategy with other active learning methods including

- the NR-Soft-MOCU (Zhao et al, 2021a) method using its implementation available at https://github.com/QianLab/NR_SMOCU_SGD_GPC,
- a naive approach for GPC that consists in enriching simultaneously the model with two points corresponding to the one maximising the variance of the latent process and the one which is the closest to a feasibility probability of 0.5,

- and with simple GPC built on Maximin Pronzato and Müller (2012) optimal design of experiments (DoE).

5.2 Analytic example in two dimensions

The performances of our proposed method and different methods are assessed on a two-dimensional classification problem based on the Branin function f_{branin} defined as follows

$$y(x) = \begin{cases} 1 \text{ if } f(x) \leq 10 \\ 0 \text{ else} \end{cases} \tag{33}$$

where

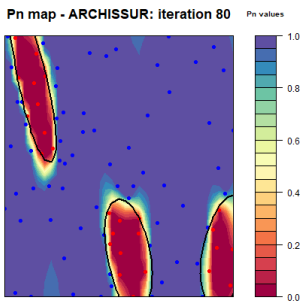
$$f_{branin}(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)cos(x_1) + s \tag{34}$$

with $a = 1$, $b = {}^{5.1}/_{4\pi^2}$, $c = {}^{5}/_{\pi}$, $r = 6$, $s = 10$ and $t = {}^{1}/_{(8\pi)}$.

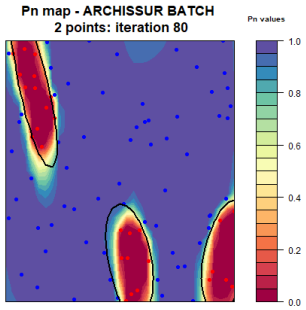
We have applied the naive strategy, the SMOCU method as well as the ARCHISSUR method with one point and with a batch of two points on the example (33). All methods were run for the same 80 initial DoEs of 12 samples, a budget of 80 enrichment points and 1000 integration points (used in the integration-based criterion estimation) for ARCHISSUR and SMOCU. Moreover, the GPC hyperparameters were optimized every 10 iterations for ARCHISSUR and every 5 iterations for ARCHISSUR with a batch of two points.

The final feasibility probability map and DoE resulting from a run of ARCHISSUR, ARCHISSUR BATCH 2 points and the NR-SMOCU method for the same initial DoE are illustrated on Figures 1. On these Figures, we can see that all methods manage to learn a good approximation of the real feasible set (in black lines on the Figures). It can also be observed that the classical GPC model with EP approximation that is used for the SMOCU algorithm is much smoother than the GPC model based on signs used in ARCHISSUR, as already highlighted in the article (Bachoc et al, 2020).

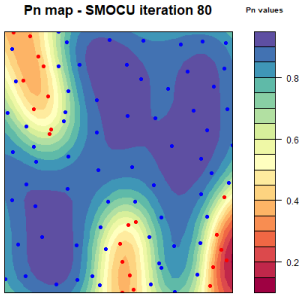
The results obtained, on average, by 80 runs on GPC model with a Maximin DoE of 92 points and the mean results of all active learning methods are presented in Tab. 2. Moreover, the evolution of the relative error through the run of each active learning method is presented on Figure 2. These results show that ARCHISSUR achieves a better estimation of the feasible set on this example. In fact, the significant difference in performance between ARCHISSUR and NR-SMOCU algorithms is mainly due to the type of GPC model chosen in the original NR-SMOCU algorithm implementation. Indeed, using the classical GPC model with Expected-Propagation approximation in ARCHISSUR implementation is greatly reducing its efficiency.



(a) ARCHISSUR



(b) ARCHISSUR BATCH



(c) NR-SMOCU with GPC model (Nickisch and Rasmussen, 2008)

Fig. 1: Probability of feasibility $p_n(x)$ maps obtained with ARCHISSUR, ARCHISSUR BATCH with 2 points and SMOCU active learning methods

Method	$\mathbb{E}[crit_F]$	$CoV(crit_F)$
Maximin with GPC (Bachoc et al, 2020)	4.97×10^{-2}	0.23
ARCHISSUR	$\mathbf{2.86 \times 10^{-2}}$	0.30
ARCHISSUR BATCH	3.08×10^{-2}	0.23
NR-SMOCU	1.32×10^{-1}	0.25
MIX	4.82×10^{-2}	0.28

Table 1: Branin-Hoo based analytical hidden constraint — mean and coefficient of variation (CoV) of the relative error $crit_F$ (29) for each method

In order to compare the criteria and their implementations in terms of numerical cost, the computational times of ARCHISSUR, ARCHISSUR for a batch of two points and NR-SMOCU criteria were assessed on this test case

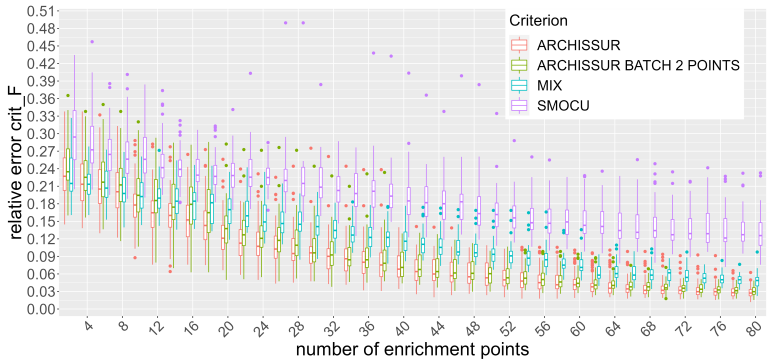


Fig. 2: Evolution of the relative error on the feasible set $crit_F$ as a function of the number of enrichment points for the Branin based example

and are given in Table 2. Note that the proposed ARCHISSUR formulation allows reducing on average the computational time by almost 4 in comparison to the usual definition adapted to classification.

Method	CPU time (s)
ARCHISSUR	**0.079**
ARCHISSUR BATCH (2 points)	0.412
NR-SMOCU	0.849

Table 2: Mean of system and user CPU times sum (in seconds) on 20 repetitions for each criterion evaluation with 2000 integration points x

The Figures 3 to 6 show the evolution of the true negative and positives rates $crit_N$ and $crit_P$ during the runs of each algorithm. From these Figures, it can be noted that the approximation obtained by NR-SMOCU is more conservative, in the sense that the unfeasible domain is overestimated (rates of true positive around 20%), compared to the other methods (rates of true positives around 98%). This is as well a consequence of the smoothness of the model.

5.3 Analytic example in ten dimensions

The performances of the different algorithms were tested in higher dimension on an analytical example with 10 inputs $x \in [0,1]^{10}$, given by:

$$y(x) = \begin{cases} 0 \text{ if } x \in \{x/f_1(x) \geq 0\} \cup \{x/f_2(x) \geq 0\} \\ 1 \text{ else} \end{cases} \tag{35}$$

where

$$\begin{aligned} f_1(x) &= x_2 + x_1 - 1.6 + 0.1x_3 \\ f_2(x) &= 0.15(0.1 + x_4) - (x_5^2 + (x_6 + 0.1)^2). \end{aligned} \tag{36}$$

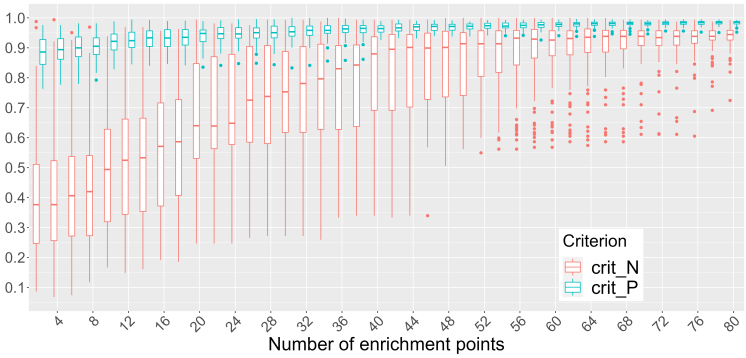


Fig. 3: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR on the Branin based example

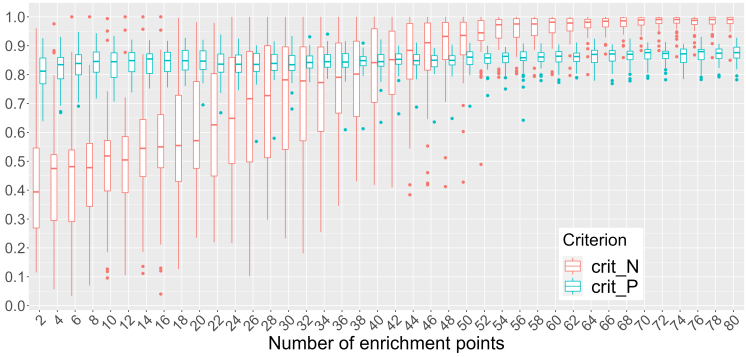


Fig. 4: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of SMOCU on the Branin based example

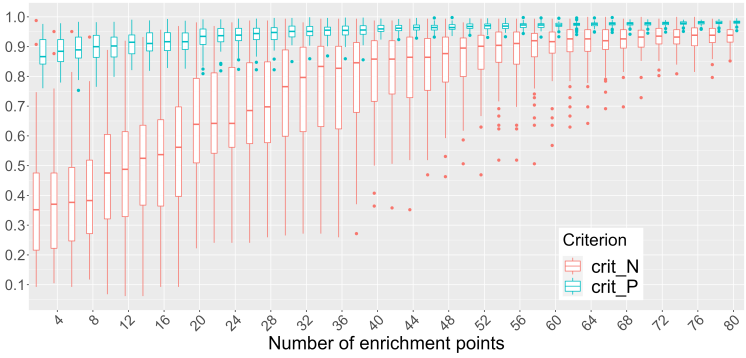


Fig. 5: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR with a batch of two points on the Branin based example

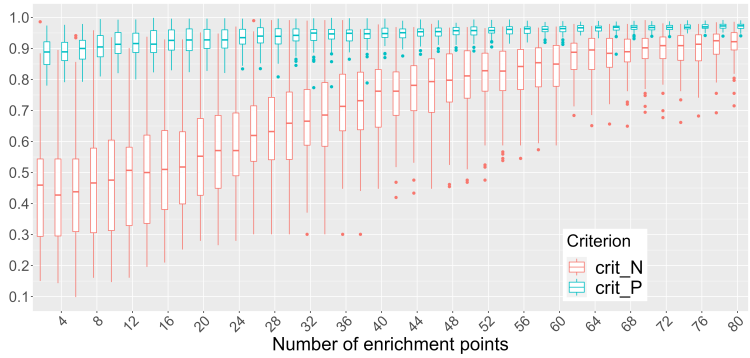


Fig. 6: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of the naive algorithm on the Branin based example

The feasible set represents 86% of the domain for this example and the unfeasible sets are located at borders of the domain with different shapes.

In this example, only the results obtained with the proposed algorithm ARCHISSUR and the naive said Mix method, are presented on Figure 7. Indeed, integration points are sampled using an importance sampling strategy with a truncated Gaussian auxiliary density in NR-SMOCU code (see (Zhao et al, 2021a)). The implemented sampling of this truncated density consists in a rejection method that does not scale in higher dimensions and fails to provide integration points in this test case.

The naive strategy and the ARCHISSUR method with one point were applied with initial DoEs of 60 points, a budget of 500 enrichment points and 10000 integration points for ARCHISSUR. Moreover, the GPC hyperparameters were optimized every 10 iterations for the first 100th iterations and on an interval of 20 iterations then.

The evolution of the relative error $crit_F$ for the two learning criteria is given on Figure 7. It can be observed that the relative error is decreasing a little faster for the naive method during the 250th first iterations but with notable outliers that reach higher error values in comparison to the evolution for ARCHISSUR which is more regular. After the 250th iteration, the relative error continues to decrease and achieves lower values with ARCHISSUR while the error interval for the naive method is wider with higher values. Table 4 gives the mean and coefficient of variation of $crit_F$ obtained at the end of each algorithm as well as the values for a GPC built on a Maximin DoE with the equivalent number of points as for the algorithms, i.e. 560 points. On the basis of these results, we can note that ARCHISSUR continues to perform well in higher dimensions with a final mean relative error 2.1 times lower than a naive method with far more regular results and 4.2 times lower than a crude Maximin DoE.

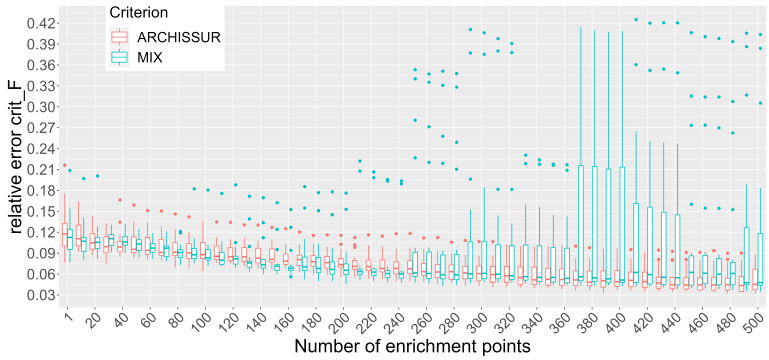


Fig. 7: Evolution of the relative error on the feasible set $crit_F$ as a function of the number of enrichment points for the ten-dimensional example

Method	$\mathbb{E}[crit_F]$	$CoV(crit_F)$
Maximin with GPC (Bachoc et al, 2020)	2.24×10^{-1}	0.86
ARCHISSUR	$\mathbf{5.3 \times 10^{-2}}$	0.38
MIX	1.12×10^{-1}	1.11

Table 3: Ten inputs example — mean and coefficient of variation (CoV) of the relative error $crit_F$ (29) for each method applied on the ten-dimensional example

The Figures 8 and 9 show the evolution of the true negative and positives rates $crit_N$ and $crit_P$ during the runs of both naive and ARCHISSUR algorithms. From these Figures, it can be noted that ARCHISSUR achieves better and more regular predictions of the feasible set.

Independently from the results, the computational times of the different criteria were assessed on this test case and are given in Table 4.

Method	CPU time (s)
ARCHISSUR	**0.363**
ARCHISSUR BATCH (2 points)	4.529
NR-SMOCU	7.58

Table 4: Mean System and User CPU time sum (in seconds) on 10 repetitions for each criterion evaluation with 10000 integration points x on the ten-dimensional test case

We can note that the numerical cost of ARCHISSUR with a batch of two points goes up significantly in comparison to the times measured in two dimensions. Combined with the cost of optimization in ten dimensions, the use of a batch strategy becomes quickly intractable.

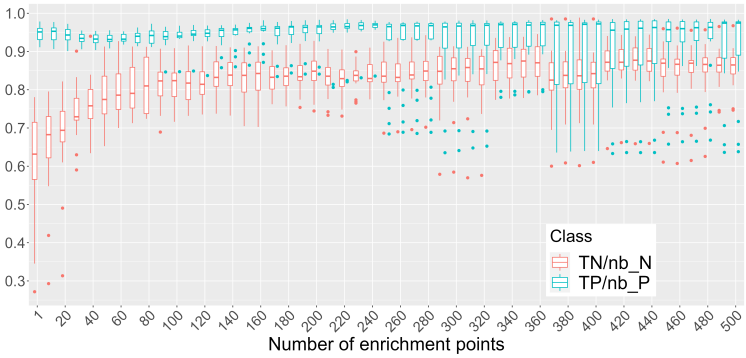


Fig. 8: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of the naive algorithm on the ten-dimensional example

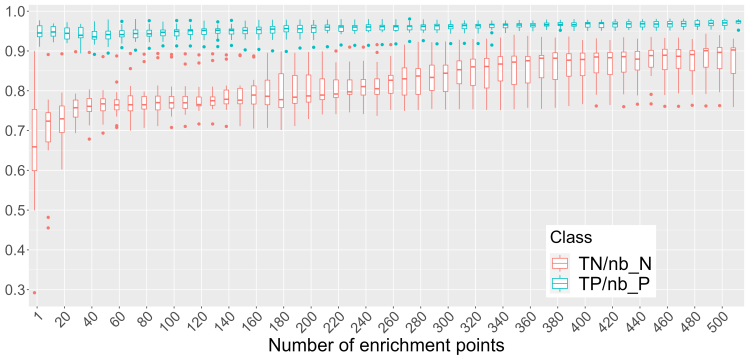


Fig. 9: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR algorithm on the ten-dimensional example

5.4 Hidden failure study of a wind turbine damage computation code

5.4.1 Description of the problem

In this section, the damage prediction of an onshore wind turbine NREL 5MW is studied. As illustrated on Figure 10, the wind turbine is subject to several wind loads that cause damage at the base of the tower. The computation of the damage involves the use of the open-source multi-physics simulator FAST (Jonkman and Buhl Jr., 2005) taking as inputs four environmental variables that concerns the wind loads. The simulator provides an estimation of one hour damage for 36 regularly spaced impact points that represent the damages on the whole wind turbine base of the tower circumference.

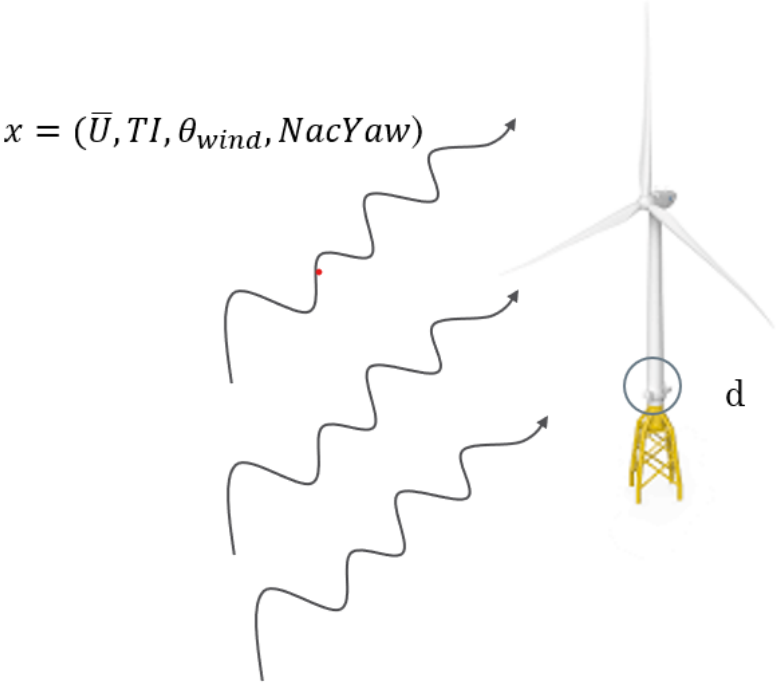


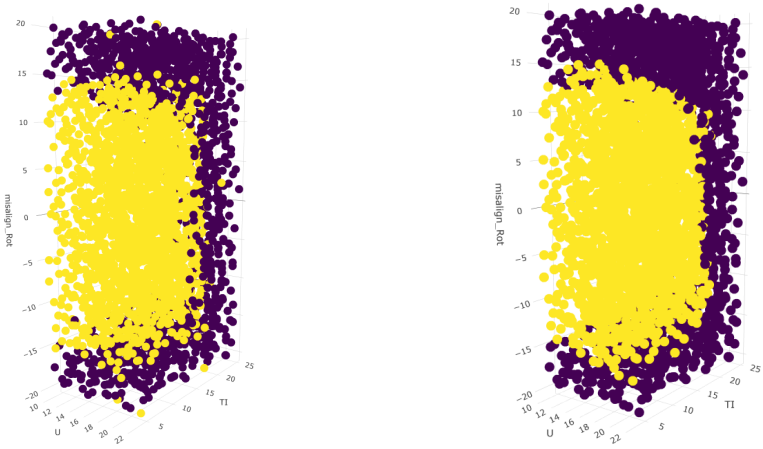
Fig. 10: Diagram of an onshore wind turbine subject to wind loads described by 4 parameters: wind speed $\bar{U}$, wind direction θ_{wind}, turbulence intensity TI and misalignment angle $NacYaw$.

The wind direction θ_{wind} influences on the damage is considered negligible and this parameter is fixed to 30°. Hence, the damage d computation depends on the three following parameters: the mean wind speed in the 10-minute interval: $\bar{U}$, the turbulence intensity: TI and the misalignment angle of the wind turbine blades: $NacYaw$. These parameters are supposed to be uncertain and are modeled by independent random variables following probability distributions given in Table 5.

Input	$\bar{U}$	TI	$NacYaw$
Unit	m/s	%	°
Probability law	Uniform	Uniform	Uniform
minimum	10	2.5	-20
maximum	22	25	20

Table 5: Probability distributions of the wind turbine problem parameters.

Moreover, the wind is modeled by a stochastic process and for each set of inputs $(\bar{U}, TI, NacYaw)$ realizations of the wind are simulated using the open



(a) Wind turbine damage simulated points classification: purple dots account for failed simulations and yellow dots for converged points

(b) Example of the classification obtained at the end of a run of ARCHISSUR with noisy GPC on the wind turbine test case

Fig. 11: Wind turbine damage validation points

source software Turbsim (Jonkman and Buhl Jr., 2006). In the following, we chose arbitrarily to fix the number of wind realizations to 18 per input set.

However, FAST and/or the Turbsim simulators encounter simulation crashes due to poor convergence for some values of the inputs. The feasible domain can thus be estimated in this test case in order to avoid simulation failures when predicting damage values. We consider that a simulation fails for an input set when a failure happens for at least one wind realization.

This test case is interesting as it allows to test the algorithms on a real physical problem on one side and is numerically affordable enough to simulate test points to have a picture of the feasible domain on the other. Indeed, short-term wind simulation and a rather small number of uncertain variables are considered, leading to a few minutes-long simulation with parallelization use. Thus, we have simulated 3000 test points whose classification in regards to simulation failure is plotted on Figure 11a. On this Figure, it can be observed that the feasible and unfeasible domains can be rather well differentiated. But we can also note that the frontiers are blurred and detect the presence of some outliers. Moreover, the feasible domain coverage, estimated from these test points, is about 58.8% of the domain.

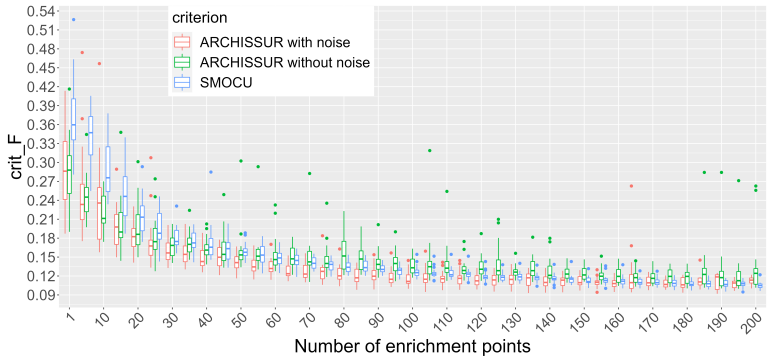


Fig. 12: Evolution of the relative error on the feasible set $crit_F$ as a function of the number of enrichment points for the wind turbine test case

5.4.2 Results

Contrary to the previous example, the hidden constraint is non-deterministic due to the stochasticity of the wind. As indicated in the previous section, the learning algorithm might encounter outliers and the GPC model is challenged by a blurred limit.

Thus, in order to get a more realistic modelling of the problem we consider a noisy latent GP. Considering independent homoscedastic noise σ_n, the prior $k_\theta^Z(\mathcal{X}_n, \mathcal{X}_n)$ becomes:

$$k_{\theta,\sigma_n}^Z(\mathcal{X}_n, \mathcal{X}_n) = k_\theta^Z(\mathcal{X}_n, \mathcal{X}_n) + \sigma_n I_n \tag{37}$$

In this model, the noise is an additional hyperparameter to be optimized by maximisation of the likelihood (MLE) as presented in Section 2.

The ARCHISSUR method with and without noise and the NR-SMOCU algorithms have been applied on this example. All methods were run for the same 20 initial DoEs of 20 samples, a budget of 200 enrichment points and 5000 integration points. The error indicators were computed on the basis of the same 3000 test points.

The evolution of the relative error through the run of each active learning method is presented on Figure 12. This Figure shows that the use of a noisy model greatly improves the results of ARCHISSUR. Moreover, it can be observed that the relative error decreases faster using ARCHISSUR with noisy GPC than NR-SMOCU. Both methods achieve an almost constant relative error value between 0.10 and 0.12, which is mainly due to the non-regularity of the test points at the frontiers of the feasible set.

The Figures 13 and 14 show the evolution of $crit_N$ and $crit_P$ during the runs of respectively ARCHISSUR with noisy GPC and NR-SMOCU methods. We can note that both methods assess a true positive rate of 0.95 and 0.925 respectively, which confirms the observation made with the relative error $crit_F$.

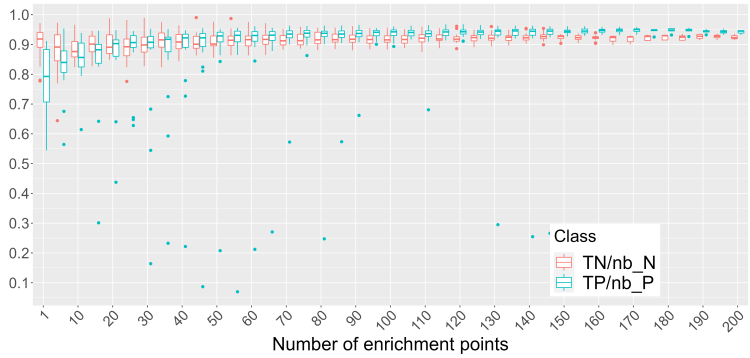


Fig. 13: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of ARCHISSUR with noisy GPC on the wind turbine test case

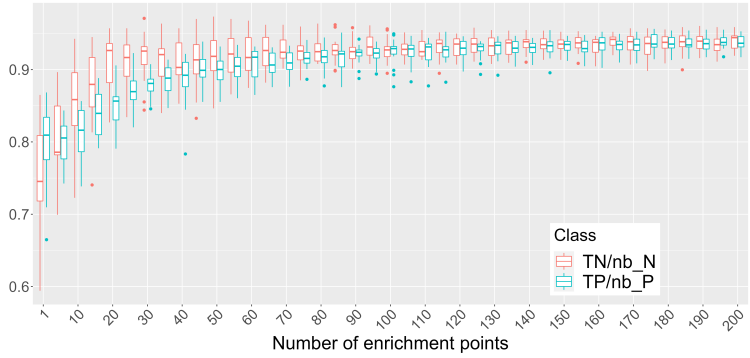


Fig. 14: Evolution of the true positives $crit_P$ and true negatives $crit_N$ rates as a function of the number of enrichment points throughout a run of SMOCU on the wind turbine test case

Finally, we provide on Figure 11b an example of the classification of the test points (see Figure 11a to observe the true class of each point) obtained at the end of a run of ARCHISSUR with a noisy GPC. This Figure allows to verify the conclusion drawn from the observation of the error reduction limit, i.e. the classification of the points at the border is more regular than the reality and the model obviously does not predict outliers.

5.5 Discussion and conclusions

In this paper, we proposed a GPC active learning method based on Stepwise Uncertainty Reduction strategies to assess hidden constraints prediction. We provided a formulation of the enrichment criteria suited for classification that is less numerically expensive compared to existing criterion implementations Zhao et al (2021a) and that allows the selection of a batch of multiple points at a time in order to improve the classification model.

The proposed algorithm was benchmarked with other methods on three different applications: two analytical examples and an industrial case. In the different applications, we have highlighted the importance of the choice of the GPC model for the performances of the algorithm. Indeed, we have noticed that the GPC model (Bachoc et al, 2020) shows improved performances on completely deterministic hidden constraints. In the non-deterministic case with significant outliers or blurred frontiers between feasible and unfeasible domains, a smoother model performs better. Hence, more accurate predictions were achieved by using a noisy GPC model in this context.

Moreover, we have observed that the use of the batch strategy becomes quickly untractable when the problem dimension increases. Future work should involve an improvement of the learning function to extend the applicability of the batch strategy.

Another perspective of this work is its use in constrained optimization. Indeed, hidden crash constraints are a well-known problem in design optimization. However, the learning of crash constraints is usually done using the points proposed by the optimizer. Hence, we are currently studying how to combine this method with the learning of crash constraints.

A final opening for further work could be to consider the problem directly with the conditional Bernoulli model framework introduced in Dai et al (2013) and draw a few links with our current GPC latent model.

Acknowledgments. This work was supported by the French National Research Agency (ANR) through the SAMOURAI project under grant ANR20-CE46-0013.

Appendix A Mean Objective Cost of Uncertainty

The Mean Objective Cost of Uncertainty (MOCU) was first defined in a general parametric setting in Yoon et al (2013) and for instance, adapted to a parametric classification context in Zhao et al (2021b). In Zhao et al (2021a) the approach was extended to a non-parametric classification setting with a GPC model and a one-step-look-ahead active learning strategy. In this latter, the active learning criterion is written, with our notations, as follows

$$\begin{aligned} & U^{MOCU}(x_{n+1}) \\ & = \int_\Omega \left[\mathbb{E}_n \left[1 - \max(p_{n+1}(x), 1 - p_{n+1}(X))\right]\right] \mu(dx) \\ & - \int_\Omega \left[1 - \max\left(p_n(x), 1 - p_n(x)\right)\right] \mu(dx) \end{aligned} \tag{A1}$$

where $\mathbb{E}_n[.] = \mathbb{E}_{Y(x_{n+1})}[\cdot|\mathcal{X}_n, \mathbf{Y_n} = \mathcal{Y}]$. The second term in the previous formulation can be developped as:

$$\begin{aligned} & \int_\Omega 1 - \max\left(p_n(x), 1 - p_n(x)\right) \mu(dx) \\ & = \int_\Omega (1 - p_n(x)) \mathbb{1}_{p_n(x) \geq 1/2} dx + \int_\Omega p_n(x) \mathbb{1}_{p_n(x) < 1/2} \mu(dx) \\ & = \int_{Q_{1/2}} p_n(x) dx + \int_{Q_{1/2}^C} 1 - p_n(x) \mu(dx) \\ & = Var_n(\Gamma) \end{aligned} \tag{A2}$$

with

$$Q_{1/2} = \{x \in \Omega : p_n(x) \geq 1/2\}. \tag{A3}$$

Similarly, we have

$$\int_\Omega \left[1 - \max\left(p_{n+1}(x), 1 - p_{n+1}(x)\right)\right] \mu(dx) = Var_{n+1}(\Gamma). \tag{A4}$$

Hence, the MOCU boils down to:

$$U^{MOCU}(x_{n+1}) = \mathbb{E}_n[Var_{n+1}(\Gamma)] - Var_n(\Gamma) \tag{A5}$$

where $Var_n(\Gamma)$ is given by (12) with $\alpha = 1/2$.

References

Adcock B, Cardenas JM, Dexter N (2023) An adaptive sampling and domain learning strategy for multivariate function approximation on unknown domains. SIAM Journal on Scientific Computing 45(1):A200–A225

Audet C, Béchard V, Le Digable S (2008) Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search. Journal of Global Optimization 41(2):299–318. https://doi.org/10.1007/s10898-007-9234-1, URL https://dx.doi.org/Doi:10.1007/s10898-007-9234-1

Azzimonti D, Ginsbourger D (2018) Estimating orthant probabilities of high-dimensional gaussian vectors with an application to set estimation. Journal of Computational and Graphical Statistics 27(2):255–267. https://doi.org/10.1080/10618600.2017.1360781, URL https://doi.org/10.1080/10618600.2017.1360781, https://arxiv.org/abs/https://doi.org/10.1080/10618600.2017.1360781

Bachoc F, Helbert C, Picheny V (2020) Gaussian process optimization with failures: classification and convergence proof. Journal of Global Optimization

78(3):483–506. https://doi.org/10.1007/s10898-020-00920-0, URL https://link.springer.com/10.1007/s10898-020-00920-0

Bect J, Ginsbourger D, Li L, et al (2012) Sequential design of computer experiments for the estimation of a probability of failure. Statistics and Computing 22(3):773–793. https://doi.org/10.1007/s11222-011-9241-4, URL https://link.springer.com/article/10.1007/s11222-011-9241-4

Botev ZI (2017) The normal law under linear restrictions: simulation and estimation via minimax tilting. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 79(1):125–148

Chevalier C (2013) Fast uncertainty reduction strategies relying on Gaussian process models. PhD Thesis

Dai B, Ding S, Wahba G (2013) Multivariate Bernoulli distribution. Bernoulli 19(4):1465 – 1483. https://doi.org/10.3150/12-BEJSP10, URL https://doi.org/10.3150/12-BEJSP10

Digabel SL, Wild SM (2015) A taxonomy of constraints in simulation-based optimization. arXiv preprint arXiv:150507881

Duhamel C, Helbert C, Munoz Zuniga M, et al (2023) A sur version of the bichon criterion for excursion set estimation. Statistics and Computing 33(2):41

El Amri MR (2019) Analyse d'incertitudes et de robustesse pour les modèles à entrées et sorties fonctionnelles. Theses, Université Grenoble Alpes, URL https://theses.hal.science/tel-02433324

El Amri MR, Helbert C, Lepreux O, et al (2020) Data-driven stochastic inversion via functional quantization. Statistics and Computing 30(3):525–541

El Amri MR, Helbert C, Munoz Zuniga M, et al (2023) Feasible set estimation under functional uncertainty by gaussian process modelling. Physica D: Nonlinear Phenomena 455:133,893. https://doi.org/https://doi.org/10.1016/j.physd.2023.133893, URL https://www.sciencedirect.com/science/article/pii/S0167278923002476

Freytag A, Rodner E, Denzler J (2014) Selecting Influential Examples: Active Learning with Expected Model Output Changes. In: Fleet D, Pajdla T, Schiele B, et al (eds) Computer Vision – ECCV 2014. Springer International Publishing, Cham, Lecture Notes in Computer Science, pp 562–577, https://doi.org/10.1007/978-3-319-10593-2_37

Genz A, Bretz F (2009) Computation of multivariate normal and t probabilities, Lecture Notes in Statistics, vol 195. Springer Science & Business

Media

Hakimi F, Brayer C, Marrel A, et al (2022) Statistical methods for the study of computer experiments failures: Application to a fuel-coolant interaction simulation code, URL https://hal.science/hal-03791882, working paper or preprint

Jonkman JM, Buhl Jr. ML (2005) "FAST User's Guide" NREL/EL-500-29798. National Renewable Energy Laboratory, Golden, CO., URL https://www.nrel.gov/wind/nwtc/openfast.html

Jonkman JM, Buhl Jr. ML (2006) "TurbSim User's Guide " NREL/TP-500-39797. National Renewable Energy Laboratory, Golden, CO., URL https://www.nrel.gov/wind/nwtc/turbsim.html

Kapoor A, Horvitz E, Basu S (2007) Selective supervision: Guiding supervised learning with decision-theoretic active learning. In: IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence, pp 877–882

Krause A, Singh A, Guestrin C (2008) Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. J Mach Learn Res 9:235–284

Kumar P, Gupta A (2020) Active learning query strategies for classification, regression, and clustering: A survey. Journal of Computer Science and Technology 35(4):913–945. https://doi.org/10.1007/s11390-020-9487-4, URL https://doi.org/10.1007/s11390-020-9487-4

Molchanov I (2005) Theory of random sets, vol 19. Springer

Moskovitch R, Nissim N, Stopel D, et al (2007) Improving the detection of unknown computer worms activity using active learning. In: KI 2007: Advances in Artificial Intelligence: 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007. Proceedings 30, Springer, pp 489–493

Moustapha M, Marelli S, Sudret B (2022) Active learning for structural reliability: Survey, general framework and benchmark. Structural Safety 96:102,174

Nevai P (1986) Géza freud, orthogonal polynomials and christoffel functions. a case study. Journal of approximation theory 48(1):3–167

Nickisch H, Rasmussen CE (2008) Approximations for binary gaussian process classification. Journal of Machine Learning Research 9(67):2035–2078. URL http://jmlr.org/papers/v9/nickisch08a.html

Pakman A, Paninski L (2014) Exact hamiltonian monte carlo for truncated multivariate gaussians. Journal of Computational and Graphical Statistics 23(2):518–542. URL http://www.jstor.org/stable/43305741

Poirette Y, Guiton M, Huwart G, et al (2017) An optimization method for the configuration of inter array cables for floating offshore wind farm. In: International Conference on Offshore Mechanics and Arctic Engineering, American Society of Mechanical Engineers, p V010T09A072

Pronzato L, Müller WG (2012) Design of computer experiments: space filling and beyond. Statistics and Computing 22(3):681–701

Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. Adaptive computation and machine learning, MIT Press, Cambridge, Mass

Roy N, McCallum A (2001) Toward optimal active learning through monte carlo estimation of error reduction. ICML, Williamstown 2:441–448

Settles B (2009) Active learning literature survey

Sudret B (2007) Uncertainty propagation and sensitivity analysis in mechanical models–Contributions to structural reliability and stochastic spectral methods. Habilitationa diriger des recherches, Université Blaise Pascal, Clermont-Ferrand, France 147

Tong S, Koller D (2001) Support vector machine active learning with applications to text classification. Journal of machine learning research 2(Nov):45–66

Vorobyev OY, Lukyanova NA (2013) A Mean Probability Event for a Set of Events. Journal of Siberian Federal University Mathematics and Physics p 128–136

Yoon BJ, Qian X, Dougherty ER (2013) Quantifying the objective cost of uncertainty in complex dynamical systems. IEEE Transactions on Signal Processing 61(9):2256–2266

Zhao G, Dougherty E, Yoon BJ, et al (2021a) Efficient active learning for gaussian process classification by error reduction. Advances in Neural Information Processing Systems 34:9734–9746

Zhao G, Dougherty E, Yoon BJ, et al (2021b) Bayesian active learning by soft mean objective cost of uncertainty. In: International Conference on Artificial Intelligence and Statistics, PMLR, pp 3970–3978