UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO DE PAULA VASCONCELOS

CONTRIBUTIONS ON PORTFOLIO-BASED BAYESIAN OPTIMIZATION

FORTALEZA

2020

THIAGO DE PAULA VASCONCELOS

CONTRIBUTIONS ON PORTFOLIO-BASED BAYESIAN OPTIMIZATION

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. João Paulo Pordeus Gomes

Coorientador: Prof. Dr. César Lincoln Cavalcante Mattos

FORTALEZA

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V451c Vasconcelos, Thiago de Paula.
Contributions on portfolio-based Bayesian optimization / Thiago de Paula Vasconcelos. – 2020.
53 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2020.
Orientação: Prof. Dr. João Paulo Pordeus Gomes.
Coorientação: Prof. Dr. César Lincoln Cavalcante Mattos.

1. Bayesian Optimization. 2. Acquisition Functions. 3. Portfolio Allocation. 4. Thompson Sampling. I. Título.

CDD 005

THIAGO DE PAULA VASCONCELOS

CONTRIBUTIONS ON PORTFOLIO-BASED BAYESIAN OPTIMIZATION

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. João Paulo Pordeus Gomes (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. César Lincoln Cavalcante Mattos (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Universidade Federal de Pernambuco (UFPE)

Prof. Dr. Amauri Holanda de Souza Júnior
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

To my family and fiancee, for their guidelines, patience and comprehension in may absences. They gave me support and confidence to focus in my work, specially during the toughest time.

ACKNOWLEDGEMENTS

To Prof. Dr. João Paulo Pordeus Gomes and Prof. Dr. César Lincoln Cavalcante for all the orientation during the development of this dissertation.

To my laboratory mates for all the discussions and knowledge shared.

To my parents, fiancee and sister, who understood my absences during this time of study, and who gave unconditional support during difficult times.

I acknowledge my professors, for all of the lessons they taught me. They have provided the necessary tools to complete this work.

To Doctor of Electrical Engineering candidate, Ednardo Moreira Rodrigues and his assistant, the undergraduate student in Electrical Engeneering, Alan Batista de Oliveira, for the adequation of the template used in this dissertation so that it could be in accordance with the Universidade Federal do Ceará (UFC) library's norms.

Finally I would like to acknowledge the sponsors of this project. This dissertation was part of a project financed through the R&D program of the Brazilian Electric Energy Agency (ANEEL), with number PD-00063-3045 and funded by CPFL Energy and Companhia Energetica Rio das Antas (CERAN). I also acknowledge the support from Delfos Intelligent Maintenance and Fundação de Apoio a Serviços Técnicos, Ensino e Fomento a Pesquisas (ASTEF).

“I may not have gone where I intended to go, but I think I have ended up where I needed to be.”

(Douglas Adams, The Long Dark Tea-Time of the Soul)

RESUMO

A Otimização Bayesiana é um método para otimização de uma função caixa-preta, sendo especialmente adequada para funções de alto custo de avaliação. Uma das partes mais importantes do algoritmo de otimização Bayesiana, a função aquisição é de fundamental importância, uma vez que ela guia o algoritmo transformando a incerteza do modelo de regressão em uma medida de pontuação para cada ponto a ser avaliado. Considerando tal aspecto, a escolha e a definição de funções de aquisição são alguns dos tópicos de pesquisa mais populares da área. Como nenhuma função de aquisição foi provada ser melhor que todas as outras em todos os problemas, uma abordagem comum consiste em selecionar diferentes funções de aquisição ao longo das iterações da execução do método. Em tal abordagem o algoritmo GP-Hedge é uma opção amplamente usada dada sua simplicidade e desempenho. Apesar de seus bons resultados em diversas aplicações, o GP-Hedge apresenta propriedades indesejadas, como considerar o desempenho de todas as iterações passadas para cada função de aquisição ao selecionar a próxima função a ser usada. Nesse caso, valores muito bons ou muito ruins obtidos em uma iteração inicial pode impactar nas escolhas das funções de aquisição a serem usadas pelo resto do algoritmo. Isso pode fazer com que uma função de aquisição domine as outras, afetando o desempenho do método. Para superar tal limitação, este trabalho propõe uma variante do GP-Hedge, chamado de Normalized Portfolio Allocation Strategy BO (No-PASt-BO), que reduz a influência de avaliações passadas ao longo do tempo. Além disso, este método apresenta uma normalização que evita que as funções no portfólio tenham probabilidades iguais. Entretanto, tais melhorias foram alcançadas ao custo da adição de dois hiperparâmetros. Como evolução desse método, é proposto um segundo método que considera amostras da *posteriori* desses hyperparâmetros por meio de *Thompson sampling*. É possível atualizar as posteriores analiticamente a cada iteração desde que as *pioris* correspondentes sejam cuidadosamente escolhidas. Esta segunda abordagem, chamada de *Self-tunning Portfolio-based Bayesian Optimization* (SeTuP-BO), mantém as vantagens do método No-Past-BO enquanto remove a necessidade de se ajustar manualmente os hyperparâmetros. Ambos os métodos e seus competidores foram avaliados em diferentes tarefas tendo ambos os métodos alcançado resultados promissores, indicando que os métodos propostos são competitivos com alternativas viáveis.

Palavras-chave: Otimização Bayesiana. Funções de Aquisição. Portfólio. Thompson sampling.

ABSTRACT

Bayesian Optimization (BO) is a framework for black-box optimization that is especially suitable for expensive cost functions. Among the main parts of a BO algorithm, the acquisition function is of fundamental importance, since it guides the optimization algorithm by translating the uncertainty of the regression model in a utility measure for each point to be evaluated. Considering such aspect, selection and design of acquisition functions are one of the most popular research topics in BO. As no single acquisition function was proved to have better performance in all tasks, a well-established approach consists of selecting different acquisition functions along the iterations of a BO execution. In such approach, the GP-Hedge algorithm is a widely used option given its simplicity and good performance. Despite its success in various applications, GP-Hedge shows an undesirable characteristic of accounting on all past performance measures of each acquisition function to select the next function to be used. In this case, good or bad values obtained in an initial iteration may impact the choice of the acquisition function for the rest of the algorithm. This fact may induce a dominant behavior of an acquisition function and may impact the final performance of the method. To overcome such limitation, this work proposes a variant of GP-Hedge, named Normalized Portfolio Allocation Strategy BO (No-PASt-BO), that reduces the influence of far past evaluations. Moreover, this method presents a built-in normalization that avoids the functions in the portfolio to have similar probabilities, thus improving the exploration. However, such an improvement has been achieved at the cost of including two hyperparameters. To improve that method, it is proposed a second one which samples from the posterior of these portfolio hyperparameters during the optimization via Thompson sampling. We can update the posteriors analytically at each iteration by carefully choosing the corresponding priors. The later approach, named Self-Tunning Portfolio-based Bayesian Optimization (SeTuP-BO), maintains the advantages of the original No-PASt-BO method without needing manually tuning hyperparameters. We evaluated both methods and their competitors across several tasks achieving promising results, indicating that the proposed methods ares competitive with the available alternatives.

Keywords: Bayesian Optimization. Acquisition Functions. Portfolio Allocation. Thompson Sampling

LIST OF FIGURES

Figure 1 – Gaussian Process Motivation 17
Figure 2 – Effect of memory factor and normalization 24
Figure 3 – Probability of choosing the least probable function 27
Figure 4 – Probability of choosing the most probable function 27
Figure 5 – Probability of choosing each of the portfolio functions based on its score . . 29
Figure 6 – Effect of different memory factors in the No-PASt-BO 30
Figure 7 – No-PASt-BO with three acquisition functions results in synthetic benchmark 33
Figure 8 – No-PASt-BO with nine acquisition functions results in synthetic benchmark 34
Figure 9 – No-PASt-BO results in real world problems 36
Figure 10 – Beta(17,3) probability density function 39
Figure 11 – Comparison between Boltzmann distribution and the approximated exponential distribution 40
Figure 12 – Gamma(40,10) probability density function 41
Figure 13 – SeTuP-BO results in synthetic benchmark 44
Figure 14 – SeTuP-BO results in PROFET benchmark 46
Figure 15 – SeTuP-BO results in real world problem 48
Figure 16 – Predictions resulted from the excecution of SeTuP-BO 49

LIST OF TABLES

Table 1 – Meta surrogate functions dimensions. 45
Table 2 – Modelled inputs and outputs by component. 47
Table 3 – NARX model variables. 48

LIST OF ABBREVIATIONS AND ACRONYMS

BO	Bayesian Optmization
CDF	Continuous Distribution Function
EI	Expected of Improvement
FNET	Fully Connected Network
GP	Gaussian Process
GP-LCB	Gaussian Process-Lower Confidence Bound
HPO	Hyperparameter optimization
LHS	Latin Hypercube Sampling
MLP	Multilayer Perceptron
NARX	Nonlinear autorregressive exogenous inputs
NN	Neural Network
No-PASt-BO	Normalized Portfolio Allocation Strategy for Bayesian Optimization
PDF	Probability Density Function
PI	Probability of Improvement
PROFET	PRObabilistic data-eFficient Experimentation Tool
RMSE	Root Mean Square Error
RP	Random Portfolio
SDO	Sequential Design for Optimization
SeTuP-BO	Self-Tunning Portfolio-based Bayesian Optimization
SGD	Sthocastic Gradient Descendent
SVM	Support Vector Machine
SVR	Support Vector Regressor
TS	Thompson Sampling
WTG	Wind Turbine Generator
XGBoost	Extreme Gradient Boosting

CONTENTS

1 INTRODUCTION 13
1.1 Objective 15
1.2 Organization 15
1.3 Related Publications 16
2 THEORETICAL BACKGROUND 17
2.1 Gaussian Process Basics 17
2.2 The Bayesian Optimization Framework 18
2.3 Acquisition Functions 20
2.3.1 Probability of improvement 20
2.3.2 Expected improvement 20
2.3.3 GP - Lower confidence bound 20
2.4 GP-Hedge 21
3 NO-PAST-BO 23
3.1 GP-Hedge limitations 23
3.2 Memory Factor 23
3.3 Rewards Normalization 25
3.4 The No-PASt-BO Algorithm 27
3.5 Results 27
3.6 Memory Factor Sensibility 28
3.6.1 Synthetic Benchmark Functions 29
3.6.2 Real World Problems 35
4 SETUP-BO 37
4.1 No-PASt-BO limitations 37
4.2 Thompson Sampling 38
4.3 SeTuP-BO 38
4.4 Empirical evaluation 42
4.4.1 Initial experiments 42
4.4.2 HPO Meta-Surrogate Benchmarking 43
4.4.3 Real-world HPO task 47
5 CONCLUSION AND SUGGESTIONS FOR FURTHER WORK 50
BIBLIOGRAPHY 51

1 INTRODUCTION

The global optimization of unknown functions is a problem which appears in a wide number of tasks. One popular approach to solve it is the use of Bayesian Optmization (BO) (MOCKUS; MOCKUS, 1991), which can be used to optimize possibly noisy functions without known closed-form expression and gradient information. BO has been used to select the hyperparameters for machine learning algorithms (SNOEK *et al.*, 2012; KLEIN *et al.*, 2017; KOTTHOFF *et al.*, 2017; FALKNER *et al.*, 2018; FEURER; HUTTER, 2019), control policies in robotics (CALANDRA *et al.*, 2016; CHATZILYGEROUDIS *et al.*, 2017), automated circuit design (LYU *et al.*, 2018; TORUN *et al.*, 2018), etc.

The BO approach is especially useful in scenarios where the objective function is expensive to evaluate. The inherent uncertainty considered by the Bayesian methodology allows for a more efficient exploration of the optimization domain with respect to the number of queried points. This is critical in applications in which each individual evaluation usually involves substantial financial and/or computational effort. The uncertainty with respect to the function to be optimized usually is modeled using the Gaussian Process (GP) framework (RASMUSSEN; WILLIAMS, 2006).

One of the main ingredients of the BO methodology is the so-called acquisition function, which translates the uncertainty in the task domain to a simple evaluation function that quantifies the expected *utility* of each point. Such function is then optimized by selecting the next point to be evaluated, i.e., the next candidate solution. However, although several acquisition functions have been proposed in the literature, there is not a single choice that is always better for any task (HOFFMAN *et al.*, 2011).

Hoffman *et al.* (2011) tackled the aforementioned issue by proposing a hierarchical hedging approach for managing an adaptive portfolio of acquisition functions based on their past performances. Similar approaches have been pursued by other authors. Shahriari *et al.* (2014) expands the original hedge by proposing a choice criterion based on information theoretic considerations. Although in a multi-armed bandit learning context, Shen *et al.* (2015) also considered a portfolio-based strategy for balancing exploration and exploitation during the sequential decision procedure. Recently, Lyu *et al.* (2018) proposed an alternative strategy that considers the multi-objective optimization of multiple acquisition functions to obtain a Pareto front from where candidate points can be sampled in a batch fashion.

Despite the above compelling recent work on the acquisition function choice problem,

the resulting solutions stray from the simplicity and applicability of the original hedging strategy presented by Hoffman *et al.* (2011), named GP-Hedge. Furthermore, GP-Hedge presents some undesirable properties. For instance, since it accounts for the historical performance of the individual acquisition functions to select the next candidate solution, initial discrepant values of either good or bad performance can compromise the quality of the selection strategy.

This work aims at proposing a modified portfolio-based BO methodology that overcomes the GP-Hedge limitations while maintaining its ease of use. Our approach reduces the influence of far past evaluations to enable the recovery of initially bad performing acquisition functions and to avoid the dominance of initially good performing functions.

The first attempt to achieve such objective was named Normalized Portfolio Allocation Strategy for Bayesian Optimization (No-PASt-BO), which presents a built-in normalization mechanism to avoid the functions within the portfolio of having similar probabilities of being chosen, promoting exploration. The new No-PASt-BO approach is empirically evaluated in the task of optimizing synthetic benchmark functions. We also consider the task of optimizing the hyperparameters of machine learning models in real world applications. The obtained results indicate that No-PASt-BO presents competitive performance and always outperforms GP-Hedge.

The No-PASt-BO introduces however 2 hyperparameters which influence the algorithm performance. Although there is a recommended configuration, the choice of their values are task-specific. Because of this, we use Thompson Sampling (TS) to propose a variation of the No-PASt-BO method in which no additional manual hyperparameter tuning is required. More specifically, we investigate TS techniques to sample from the posterior of the portfolio hyperparameters. At each iteration such sample is used to select which acquisition function will guide the choice of the next queried point. As we will detail later on, this is equivalent to consider each acquisition function in a portfolio as an arm of a multi-armed bandit problem. Furthermore, the posterior of each portfolio hyperparameter is analytically updated after each optimization step thanks to careful conjugate prior choices.

The proposed variant of No-PASt-BO, the Self-Tunning Portfolio-based Bayesian Optimization (SeTuP-BO), has the following goals: (*i*) to preserve the simpliicity of the GP-Hedge portfolio allocation strategy; (*ii*) to maintain, or to improve, the performance of the original No-PASt-BO algorithm; (*iii*) to mitigate the need of manual intervention in the optimization procedure.

We thoroughly evaluate the proposed SeTuP-BO and compare it with several base-

lines. Within our experiments, we use the PRObabilistic data-eFficient Experimentation Tool (PROFET) framework (KLEIN *et al.*, 2019) to sample artificial Hyperparameter optimization (HPO) tasks for Neural Network (NN), Support Vector Machine (SVM) and Extreme Gradient Boosting (XGBoost). Additionally, we perform experiments with tuning NN-based Nonlinear autorregressive exogenous inputs (NARX) models in the task of fault detection in energy plants considering real-world data. The promising obtained results indicate the competitiveness of the proposed approach.

1.1 Objective

The main objective of this work is to propose a general purpose BO method which combines several acquisition functions and which performs better than the best performing acquisition function, while keeping its easy of use. Besides that we also target to get better results than the GP-Hedge algorithm, which is, to the best of our knowledge the first one to propose combining acquisition functions in the context of BO.

The specific objectives of this research are:

- To make a literature review on BO, portfolio allocation strategies and TS techniques for BO;
- To propose new methods which are able to outperform the GP-Hedge algorithm;
- To run computational experiments by following the methodology presented in the literature to evaluate how the new proposed methods compare to the baselines.

1.2 Organization

The remaining of the dissertation is organized as follows. Chapter 2 summarizes the required theoretical background; Chapter 3 illustrates the GP-Hedge limitations, details the proposed No-PASt-BO methodology and presents and discuss the performed computational experiments; Chapter 4 details the proposed SeTuP-BO methodology and presents and discuss the performed computational experiments; Chapter 5 concludes the thesis with recommendations for further investigations.

1.3 Related Publications

The contributions of this dissertation were previously published or submitted to publication, in the name of the co-authors.

Our first result is presented in (VASCONCELOS *et al.*, 2019). Vasconcelos *et al.* (2019) proposes a variant of the original GP-Hedge (HOFFMAN *et al.*, 2011) method that overcomes some of its main limitations. The proposed method, named No-PASt-BO, was able to outperform the original GP-Hedge and other single acquisition functions BO methods in many situations. A further development was submitted for publication to a highly reputed peer-reviewed journal but by the time of the submission of this dissertation, the paper is still under review. Our method, named SeTuP-BO, improves the No-PASt-BO method by eliminating all its hyperparamenters.

During the development of this thesis the author also had one paper (see (SOUZA *et al.*, 2019)) that was not a direct result of this thesis but an application of the methods presented in this document.

2 THEORETICAL BACKGROUND

This section summarizes the main theoretical aspects of the BO framework, including GP basics, the role of the acquisition function and the original GP-Hedge algorithm.

2.1 Gaussian Process Basics

As a motivation for using the GP,there is the fact that if an arbitrary number of points are given from an unknown function, it is possible to guess functions which are a match for the data, with some uncertainty. As more data points are given, the uncertainty reduces and it is possible to guess a better function to describe the data. Figure 1 shows an example of two guesses for the sample function, where the second guess knows more points than the first one of the real function.

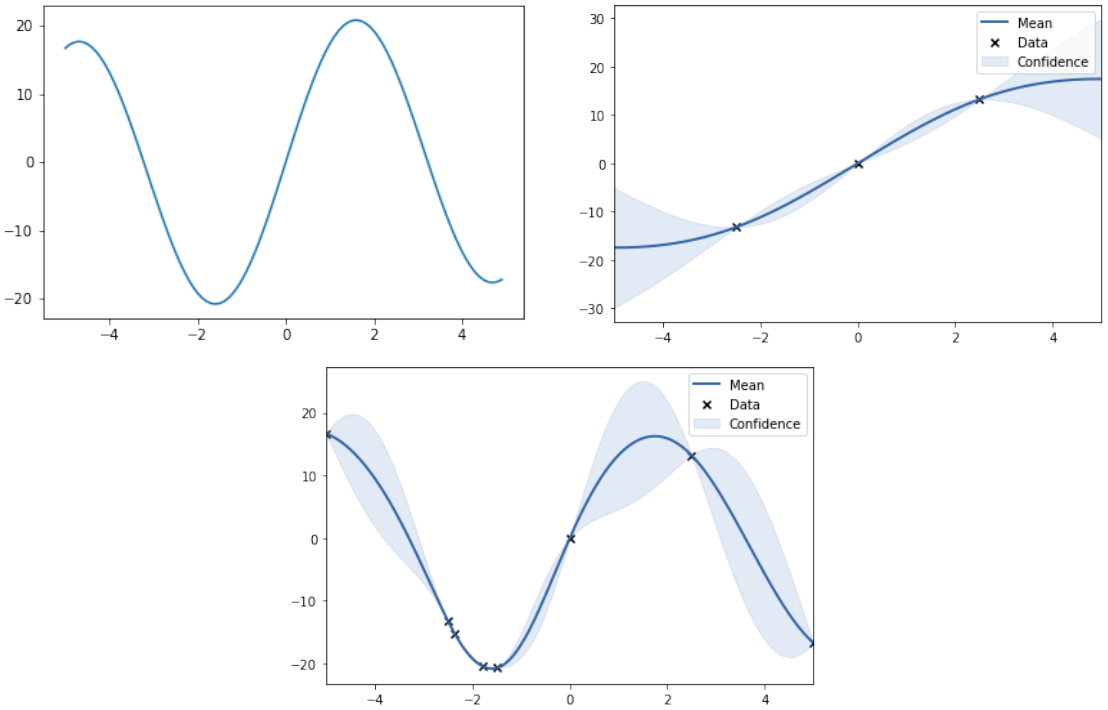

Figure 1 – The image on first line represents the original function from which some points are sampled. The image on the second line tries to reconstruct the function on the first line knowing three and eight data points.

In a standard single dimension output regression setting, the aim is to obtain a mapping $f : \mathbb{R}^D \to \mathbb{R}$ from a set of N inputs $\boldsymbol{x}_i \in \mathbb{R}^D$, organized in a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, to a set of N correspondent outputs $f_i \in \mathbb{R}$. However, we usually observe $\boldsymbol{y} \in \mathbb{R}^N$, only a noisy version

of the vector $\boldsymbol{f}$. Considering a Gaussian observation noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ we have:

$$y_i = f_i + \varepsilon, \text{ where } f_i = f(\boldsymbol{x}_i), \quad 1 \leq i \leq N. \tag{2.1}$$

In the GP probabilistic modeling framework it was chosen a multivariate Gaussian prior for the *latent* (non-observed) vector $\boldsymbol{f}$. All the available inputs are collected in the matrix $\boldsymbol{X} \in \mathbb{R}^{(N \times D)}$, while the observed outputs form the vector $\boldsymbol{y} \in \mathbb{R}^N$. If a zero mean prior is chosen, we get (RASMUSSEN; WILLIAMS, 2006):

$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{X}) &= \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \\ p(\boldsymbol{y}|\boldsymbol{f}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2 \boldsymbol{I}), \\ p(\boldsymbol{y}|\boldsymbol{X}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_f + \sigma^2 \boldsymbol{I}), \end{aligned} \tag{2.2}$$

where in Eq. (2.2) it was possible to analytically integrate out $\boldsymbol{f}$. The elements of the covariance matrix $\boldsymbol{K}_f \in \mathbb{R}^{N \times N}$ which are calculated by $[\boldsymbol{K}_f]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), \forall i, j \in \{1, \cdots, N\}$, where $k(\cdot, \cdot)$ is the so-called covariance (or *kernel*) function.

The kernel hyperparameters are usually optimized following the gradients of the log-marginal likelihood, i.e., the logarithm of Eq. (2.2), also called the *evidence* of the model.

Given a new input $\boldsymbol{x}_* \in \mathbb{R}^D$, the posterior predictive distribution of the related output $f_* \in \mathbb{R}$ is calculated analytically using standard Gaussian distribution conditioning properties:

$$\begin{aligned} p(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) &= \mathcal{N}\left(f_* \middle| \mu_*, \sigma_*^2\right), \\ \mu_* &= \boldsymbol{k}_{*f}(\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \\ \sigma_*^2 &= K_* - \boldsymbol{k}_{*f}(\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{f*}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{k}_{f*} = [k(\boldsymbol{x}_*, \boldsymbol{x}_1), \cdots, k(\boldsymbol{x}_*, \boldsymbol{x}_N)]^\top \in \mathbb{R}^N$, $\boldsymbol{k}_{*f} = \boldsymbol{k}_{f*}^\top$ and $K_* = k(\boldsymbol{x}_*, \boldsymbol{x}_*) \in \mathbb{R}$. It is important to mention that each prediction is a fully defined distribution, instead of a point estimate, which reflects the inherent uncertainty of the regression problem.

2.2 The Bayesian Optimization Framework

As a motivation for BO, if it is needed to make some oil well, in a completely unknown terrain, in which no previous study can be made. The only solution would be to make the well and then get the results. If an arbitrary number of points are given from an unknown function, it is possible to guess functions which are a match for the data, with some uncertainty.

As more data points are given the uncertainty is reduced and one would possibly guess a better function to describe the data. Image 1 shows an example of two guesses for the sample function.

BO is a general framework for black-box optimization. Mathematically, the problem consists in finding a global minimizer (or maximizer) of an unknown loss function (BROCHU *et al.*, 2010)

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathscr{X}} l(\boldsymbol{x}), \tag{2.4}$$

where $l(\cdot) : \mathscr{X} \to \mathbb{R}$ denotes the loss function (or objective, when maximizing), $\mathscr{X}$ is the search space, usually given by a subset of $\mathbb{R}^D$, and $\boldsymbol{x}^* \in \mathbb{R}^D$ denotes the optimal solution. The loss function is usually assumed to be either hard to compute or having no simple closed form, but it can be evaluated at an arbitrary query point $\boldsymbol{x}$. Similar to Eq. (2.1), the BO framework also considers cases in which one does not observe $l(\cdot)$ directly, but rather noisy observations.

BO solves the problem in Eq. (2.4) by sequentially querying the loss function as we keep the best-so-far candidate solution $\boldsymbol{x}^+$. In doing so, at iteration t, we select a new location $\boldsymbol{x}_{t+1}$ at which the loss function $l(\boldsymbol{x}_{t+1})$ is evaluated. As the method iterates, the querying points and its corresponding loss values $\mathscr{D}_t = \{\boldsymbol{x}_i, l(\boldsymbol{x}_i)\}_{i=1}^t$ are used for modeling the loss function. When a stopping criterion has been achieved, we return $\boldsymbol{x}^+$ as an approximation to the actual minimizer $\boldsymbol{x}^*$. A fundamental aspect is how to provide location guesses along iterations. For this reason a probabilistic model is necessary, since $l(\cdot)$ is unknown.

The most common probabilistic approach used in the BO framework is the GP model, summarized in Section 2.1. The GP model is able to quantify the uncertainty with respect the function $l(\cdot)$, especially at locations in which the function has not yet been evaluated and knowledge is scarce.

The uncertainty about the loss is used to select the most promising candidate point for evaluation. It is achieved by an easy-to-compute *acquisition function*. Typically, such acquisition functions are model-derived and are used to trade-off exploration and exploitation; in the sense that exploration means investigate non-explored areas (with high uncertainty), and exploitation refers to considering regions where the model prediction is high. The goal is to maximize the acquisition function, and it can be achieved because such functions are assumed to be cheap to evaluate, usually with gradient information available.

We refer the reader to the comprehensive survey by Shahriari *et al.* (2015) for more details and challenges in the general BO framework.

2.3 Acquisition Functions

Several acquisition functions have been proposed in the literature (HERNÁNDEZ-LOBATO *et al.*, 2014; HENNIG; SCHULER, 2012). Each proposal aims at measuring the quality of the candidates to be queried following a specific strategy. Next we detail three of the most used acquisition functions in practice.

2.3.1 Probability of improvement

The Probability of Improvement (PI) function, firstly proposed by Kushner (1964), focuses on choosing the domain point with the highest probability of being lower than $\mu^- = \min_i \mu(\boldsymbol{x}_i)$, where $\mu(\boldsymbol{x}_i)$ indicates the GP predicted mean at the input $\boldsymbol{x}_i$. This formulation focuses on exploitation (HOFFMAN *et al.*, 2011), which can be balanced with the trade-off hyperparameter $\xi \geq 0$ as follows:

$$\text{PI}(\boldsymbol{x}) = P(f(\boldsymbol{x}) \leq \mu^- - \xi) = \Phi\left(\frac{\mu^- - \xi - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right), \tag{2.5}$$

where Φ is the Continuous Distribution Function (CDF) of a standard Gaussian distribution.

2.3.2 Expected improvement

The Expected of Improvement (EI) function, introduced by Mockus *et al.* (1978), considers the probability of an evaluation being lower than the current best known evaluation, but it also takes into account the magnitude of the improvement. Let $\mu^- = \min_i \mu(\boldsymbol{x}_i)$, the EI function is zero if $\sigma(\boldsymbol{x}) = 0$, otherwise it is given by

$$\text{EI}(\boldsymbol{x}) = \tau(\boldsymbol{x})\Phi\left(\frac{\tau(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{\tau(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right) \tag{2.6}$$

where $\tau(\boldsymbol{x}) = \mu^- - \xi - \mu(\boldsymbol{x})$

In Eq. (2.6), Φ and ϕ are respectively the CDF and the Probability Density Function (PDF) of a standard Gaussian distribution.

2.3.3 GP - Lower confidence bound

In Cox e John (1997) it is introduced the Sequential Design for Optimization (SDO), which selects a point to evaluate based on the posterior mean and variance, minimizing $\mu(\boldsymbol{x}) - \kappa\sigma(\boldsymbol{x})$. In the original paper, κ is a hyperparameter, but no clues are given in how to select it. In

Srinivas *et al.* (2010), the SDO algorithm is revisited and distinct approaches to select a value for κ are discussed. Thus, the so-called Gaussian Process-Lower Confidence Bound (GP-LCB) formulation is presented below:

$$\text{GP-LCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) - \sqrt{\nu\beta_t}\sigma(\boldsymbol{x}), \tag{2.7}$$

where we have considered $\kappa = \sqrt{\nu\beta_t}$, $\beta_t = 2\log(t^{D/2+2}\pi^2/3\delta)$ varies with the sequential iteration t and $\nu, \delta > 0$ (BROCHU *et al.*, 2010).

2.4 GP-Hedge

As previously mentioned, there is not a single acquisition function which is the best choice for all possible optimization tasks. Therefore, a strategy which can choose among a set of acquisition functions may be a good direction to handle this issue. The GP-Hedge, introduced this approach in Hoffman *et al.* (2011), and other authors have used similar approaches (SHAHRIARI *et al.*, 2014).

In the GP-Hedge framework, a set of predefined acquisition functions is considered, comprising a *portfolio*. Each function nominates a candidate for the next point $\boldsymbol{x}$ of the domain to be evaluated. The candidates are then selected with a probability proportional to how good the posterior mean of the previous points the corresponding acquisition function has suggested before.

The aforementioned approach follows the *hedge* strategy. According to Auer *et al.* (1995), such method consists in choosing the action j among J options with probability $p_j \propto \exp(\eta G_j(t))$, where η is a hyperparameter, which determines the importance of the score in the selection, the bigger the η more determinant is the score, and $G_j(t) = \sum_{t'=1}^{t} \text{score}_j(\boldsymbol{x}_j(t'))$ is the total score of the action j up to the time t.

The original GP-Hedge algorithm proposed by Hoffman *et al.* (2011) was defined to solve a maximization problem, so the reward of each acquisition function is equal to the sum of the previous posterior means, i.e., $G_j(t) = \sum_{t'=1}^{t} \mu_j(\boldsymbol{x}_j(t'))$. Since we define our tasks as minimization problems, we multiply each score by -1 before adding it, i.e., $G_j(t) = -\sum_{t'=1}^{t} \mu_j(\boldsymbol{x}_j(t'))$. The full algorithm is detailed in Algorithm 1.

Algorithm 1: GP-Hedge

Select hyperparameter $\eta \in \mathbb{R}^+$
Set $G_j(0) = 0$ for $j = 1, 2, \ldots, J$
for t = 1,2,... **do**
  Nominate points from each acquisition function h_j:
  $\boldsymbol{x}_j(t) = \arg\max_{\boldsymbol{x}} h_j(\boldsymbol{x})$.
  Select a nominee $\boldsymbol{x}(t) = \boldsymbol{x}_j(t)$ with probability
  $p_j(t) = \frac{\exp(\eta G_j(t-1))}{\sum_{j'=1}^{J} \exp(\eta G_{j'}(t-1))}$.
  Compute $y(t)$ by evaluating the objective on point $\boldsymbol{x}(t)$.
  Augment the data $\mathscr{D}_t$ with the new pair $(\boldsymbol{x}(t), y(t))$.
  Update the surrogate GP model.
  Update the rewards $G_j(t) = G_j(t-1) - \mu(\boldsymbol{x}_j(t))$ from the updated GP posterior.
end for

3 NO-PAST-BO

3.1 GP-Hedge limitations

The original GP-Hedge algorithm relies on the cumulative performance of each acquisition function in the portfolio during previous iterations to favor the choice of a function over the others. However, for large enough horizons, the influence of the first iterations may not be relevant or even desirable. For this reason, it is necessary to reduce the importance of an iteration in the reward function as the former becomes more distant from the current iteration. It is important to mention that it is emphasized by Shahriari *et al.* (2014) that the reward function is critical for the GP-Hedge performance, which encourages our argument.

3.2 Memory Factor

In order to tackle the presented GP-Hedge issues, it is proposed to change the reward function update by including a memory factor. The main goal of the memory factor is to decrease the influence of previous iterations in the reward evaluation as new iterations are completed, while still considering past experiences. By doing that, we avoid that discrepant values in the first few iterations of being determinant in the acquisition function selection during all the later iterations. This approach also enables more probability for acquisition functions that are better in the recent past, according to the memory factor value, enabling "recovery" from far past mistakes.

In Figure 2 it is shown an example of how the scores of each acquisition function evolve with the iterations. We consider a 3-function portfolio comprised by the PI, EI and GP-LCB functions and the well known Hartmann 6 (HARTMAN, 1973) benchmark. It is also presented the corresponding probabilities of being chosen. It can be noted that for the GP-Hedge the GP-LCB function finishes with a large score lead over the other 2 functions. In comparison with the versions with a memory factor, one notes that this lead is smaller and can still be lost. Both score graphs for the versions with memory factors are very similar. However, the probability graphs are different, since in the normalized version it is more uncommon to obtain situations of equal probabilities, which would result in a completely random choice between the 3 available functions. Both the memory factor and normalization mechanisms will be detailed in the next sections.

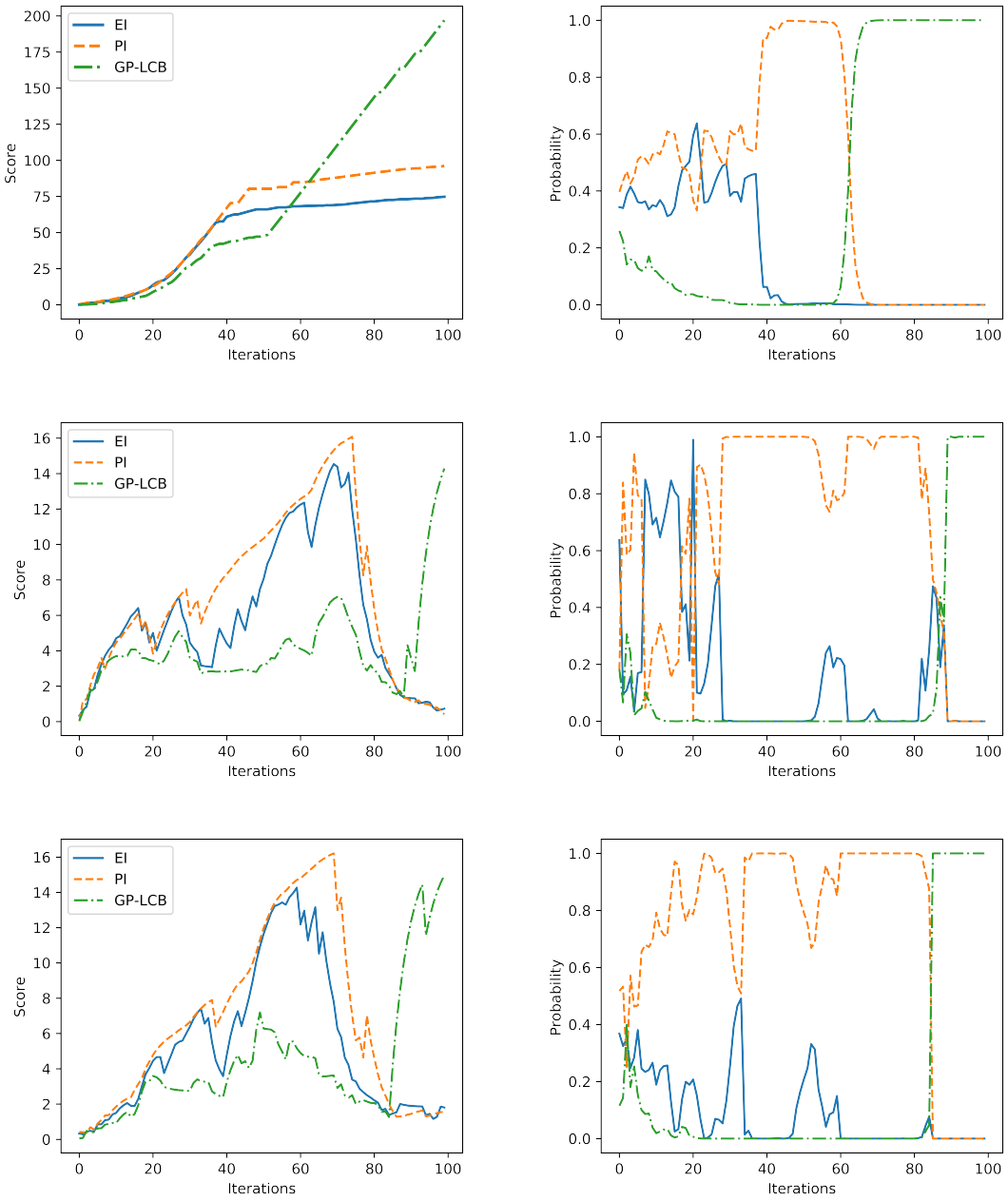

Figure 2 – In the first column, it is presented the scores of each acquisition function along the 100 iterations of a typical run for the Hartmann 6 benchmark function. The first row corresponds to the GP-Hedge, the second row corresponds to the GP-Hedge with a memory factor of 0.8, and, finally the third row corresponds to the memory factor of 0.8 using the normalization strategy, which consists in our approach. In the second column, the correspondent probabilities of each acquisition function chosen is shown.

In sum, we propose to change the reward update computation as follows:

$$G_j(t) = mG_j(t-1) - \mu(\boldsymbol{x}_j(t)), \tag{3.1}$$

where $0 \leq m \leq 1$ is the memory factor hyperparameter, $\mu(\boldsymbol{x}_j(t))$ is the GP posterior mean for the input $\boldsymbol{x}_j(t)$ suggested by acquisition function h_j at iteration t. Typical values for the memory factor hyperparameter are between 0.7 and 1, where the latter recovers the original GP-Hedge.

Eq. (3.1) imposes a decrease in relevance to past rewards. For $m < 1$ we achieve two behaviors that are difficult to observe in the original GP-Hedge: (*i*) initially bad acquisition functions may receive some probability in the later iterations if they improve along the optimization steps; (*ii*) acquisition functions that go very well in the beginning may lose the preference if they cannot keep the good performance. Those behaviors can be observed in Figure 2, where we can note, for instance, that after some initial iterations the EI function is able to recover some probability around the 50^{th} iteration, and then, as it is not doing well, returns to almost 0% chance of being chosen.

3.3 Rewards Normalization

As we have reduced the influence of the initial iterations evaluations with the previously presented memory factor, it is possible that the rewards of the portfolio to be very close at some iterations, as illustrated in Figure 2. In such scenario, all the acquisition functions have about the same probability of being chosen. In the extreme case where all the functions have equal probability we would get a completely undesired random portfolio behavior.

In order to solve this problem, we propose to normalize the reward functions values before computing the choice probabilities. By doing it, we are able to have some control in the range of possible values. The proposed normalization is presented as follows:

$$r_j(t) = \frac{G_j(t) - r_{\max}(t)}{r_{\max}(t) - r_{\min}(t)}, \tag{3.2}$$

where $r_{\min}(t) = \min_j(G_j(t))$,

$r_{\max}(t) = \max_j(G_j(t))$.

In the former expressions, the term $r_j(t)$ indicates the normalized reward for the acquisition function j after the iteration t. Those values, computed for each acquisition function $h_j|_{j=1}^{J}$, are

then used to obtain the probabilities of each acquisition function being chosen as follows:

$$p_j(t) = \frac{\exp(\eta r_j(t-1))}{\sum_{j'=1}^{J} \exp(\eta r_{j'}(t-1)))}. \tag{3.3}$$

Note that the normalized $r_j(t)$ values are considered only to compute Equation 3.3. The original values of $G_j(t)$ are not overwritten.

The proposed normalization step constrains the terms $r_j(t)$ to be between -1 and 0, with the highest value always being 0 and the lowest always being -1. To solve a possible division by 0, if the highest and lowest rewards are equal, all probabilities are set equally. Otherwise, because of the normalization, at least one of the functions will score 1, and at least one of the other functions will score $\exp(-\eta)$. This way, we have $r_{j_1} = 0$ and $r_{j_2} = -1$ and we can remove $exp(\eta r_{j_1}) = 1$ and $exp(\eta r_{j_2}) = exp(-\eta)$ from the summation in Equation 3.3, by adjusting it to:

$$p_j(t) = \frac{\exp(\eta r_j(t-1))}{1+\exp(-\eta)+\sum_{j'=1}^{J-\{j_1,j_2\}} \exp(\eta r_{j'}(t-1)))}. \tag{3.4}$$

The proposed η is strictly related with the probability of choosing an arbitrary function. For a given n number of acquisition functions, the minimum probability of a function being selected occurs when $n-1$ functions have the same score and the $i-th$ function have a score which is smaller than the others. The probability value is given by the expression in Equation 3.5.

$$p_j(t) = \frac{\exp(-\eta)}{n-1+\exp(-\eta)}. \tag{3.5}$$

Similarly, the maximum probability of a function being selected occurs when $n-1$ functions have the same score and the ith function has a score which is greater than the others. The probability value is given by the expression in Equation 3.6.

$$p_j(t) = \frac{1}{1+(n-1)\exp(-\eta)}. \tag{3.6}$$

Assuming the use of three acquisition functions ($n = 3$), which is the case of most of the experiments, the relation between the minimum probability of a function being chosen and the η value is presented in Figure 3

Still assuming the use of three acquisition functions, the relation between the maximum probability of a function being chosen and the η value is presented in Figure 4

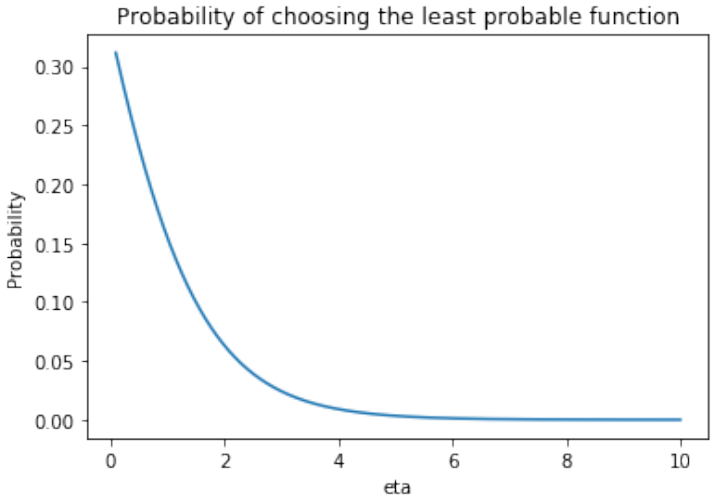

Figure 3 – Probability of choosing the least probable function

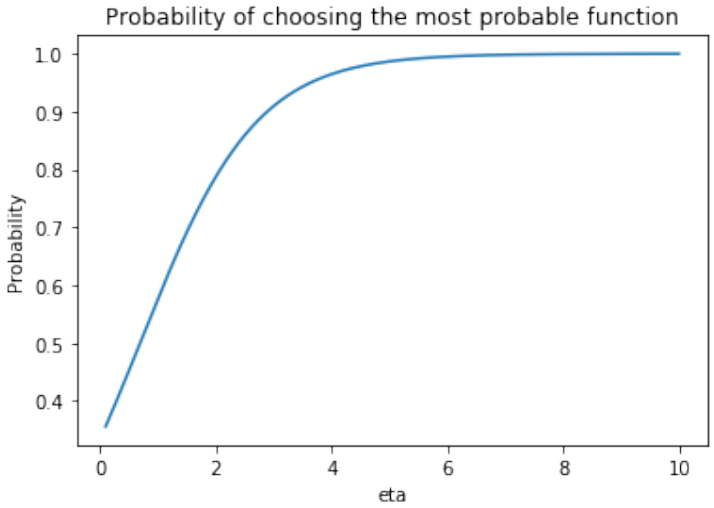

Figure 4 – Probability of choosing the least probable function

3.4 The No-PASt-BO Algorithm

The proposed changes in the rewards function of the base GP-Hedge approach result in the proposal of the No-PASt-BO algorithm, which is detailed in Algorithm 2. Note that the proposed method presents the same computational cost of the original GP-Hedge.

3.5 Results

A battery of experiments were made in synthetic benchmark function optimization in order to evaluate the proposed No-PASt-BO. It was also considered the real world application of BO in the task of optimizing the hyperparameters of machine learning models. Twenty-five runs were executed for each test and the mean logarithmic error is reported along with the

Algorithm 2: No-PASt-BO

Select hyperparameter $\eta \in \mathbb{R}^+$
Select hyperparameter $m \in [0,1]$
Set $G_j(0) = 0$ for $j = 1,2,\ldots,J$
for t = 1,2,... **do**
  Nominate points from each acquisition function h_j:
  $\boldsymbol{x}_j(t) = \arg\max_{\boldsymbol{x}} h_j(\boldsymbol{x})$
  Compute $r_{\min}(t-1) = \min_j(G_j(t-1))$
  Compute $r_{\max}(t-1) = \max_j(G_j(t-1))$
  Compute the normalized rewards:
  $r_j(t-1) = \frac{G_j(t-1) - r_{\max}(t-1)}{r_{\max}(t-1) - r_{\min}(t-1)}$
  Select a nominee $\boldsymbol{x}(t) = \boldsymbol{x}_j(t)$ with probability
  $p_j(t) = \frac{\exp(\eta r_j(t-1)}{\sum_{j'=1}^{J} \exp(\eta r_{j'}(t-1)))}$.
  Compute $y(t)$ by evaluating the objective on point $\boldsymbol{x}(t)$.
  Augment the data $\mathscr{D}_t$ with the new pair $(\boldsymbol{x}(t), y(t))$.
  Update the surrogate GP model.
  Update the rewards $G_j(t) = mG_j(t-1) - \mu(\boldsymbol{x}_j(t))$ from the updated GP posterior.
end for

correspondent confidence interval obtained, i.e., the standard deviation scaled by the square root of the number of runs.

As baselines, it was also evaluated a Random Portfolio (RP) approach, which chooses randomly among a set of predefined acquisition functions, the original GP-Hedge and standard BO with a single acquisition function. For all of the experiments, it was used the GpyOpt package, a general BO framework introduced by The GPyOpt authors (2016. Available at http://github.com/SheffieldML/GPyOpt. Accessed on September 19^{th} 2020). After preliminary experiments, the hyperparameter η was set to 4 for all of the No-PASt-BO experiments, while for the GP-Hedge the η was defined using the strategy suggested by Hoffman *et al.* (2011).

Applying Equation 3.5 and 3.6 assuming $n = 3$ and $\eta = 4$ the probability of a function being according to its score is presented in Figure 5.

3.6 Memory Factor Sensibility

Before comparing results, it was desirable exploring the impact of the memory factor m in the No-PASt-BO method.Thus, an initial experiment was performed with 7 different values for the memory factor: $\{0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1\}$. Note that when the memory factor is equal to 1, it means that previous executions do not lose importance. With the exception of the normalizing step (see Section 3.3), the latter is similar to the GP-Hedge. As the memory factor

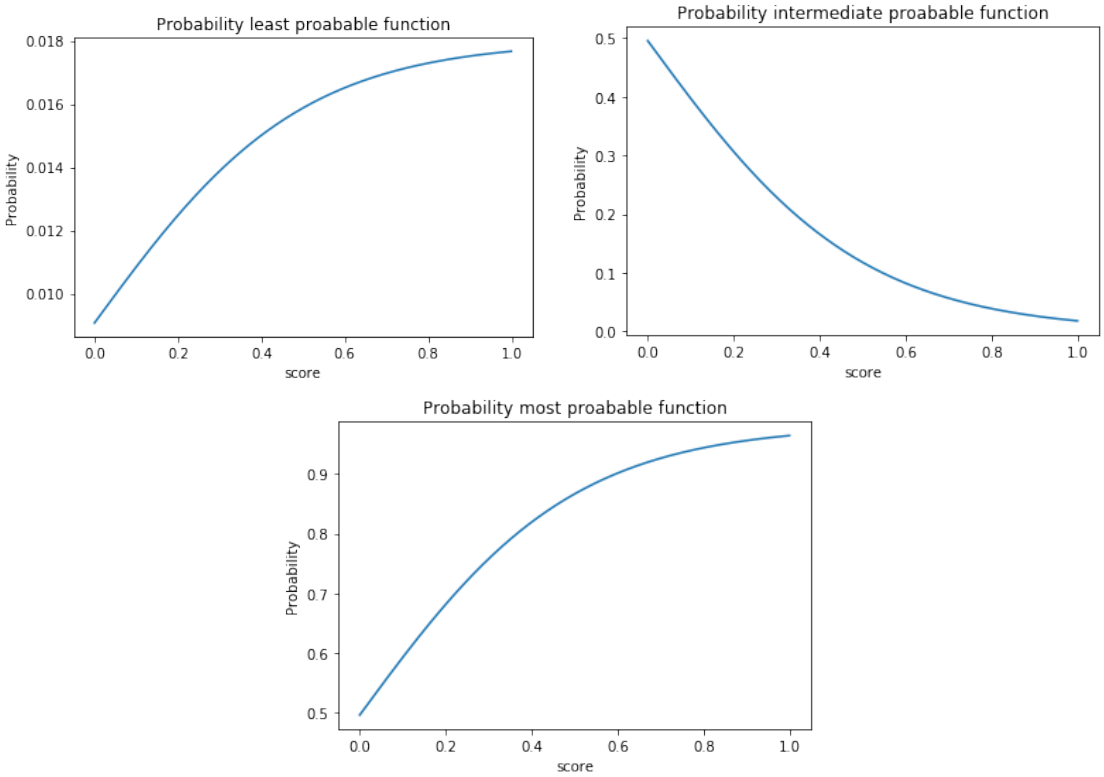

Figure 5 – Probability of choosing a function

decreases, the importance of previous evaluations also decreases.

The impact of different memory factor values are presented in Figure 6 for the Hartmann 6 benchmark function. It is possible to verify how the rewards change over the iterations, and how it affects the probability of the correspondent acquisition function being chosen. Higher values of a memory factor result in a more difficult to select an acquisition function as the most probable one. Moreover, lower values for the memory factor imply in more quickly varying probabilities, which may improve the diversity in the acquisition function selection step.

3.6.1 Synthetic Benchmark Functions

It was used 3 standard benchmark functions in this section: Branin (BRANIN, 1972), Hartmann 3 and Hartmann 6 (HARTMAN, 1973) [1]. Their domains are respectively 2, 3 and 6 dimensional. These functions were the same chosen in the original GP-Hedge paper (HOFFMAN *et al.*, 2011).

The Branin function consists in a 2 dimensional function $x_1 \in [-5, 10], x_2 \in [0, 15]$, with three global minimum at $[(-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)]$ with value 0.397887.

[1] Function definitions available at <https://www.sfu.ca/~ssurjano/optimization.html>.

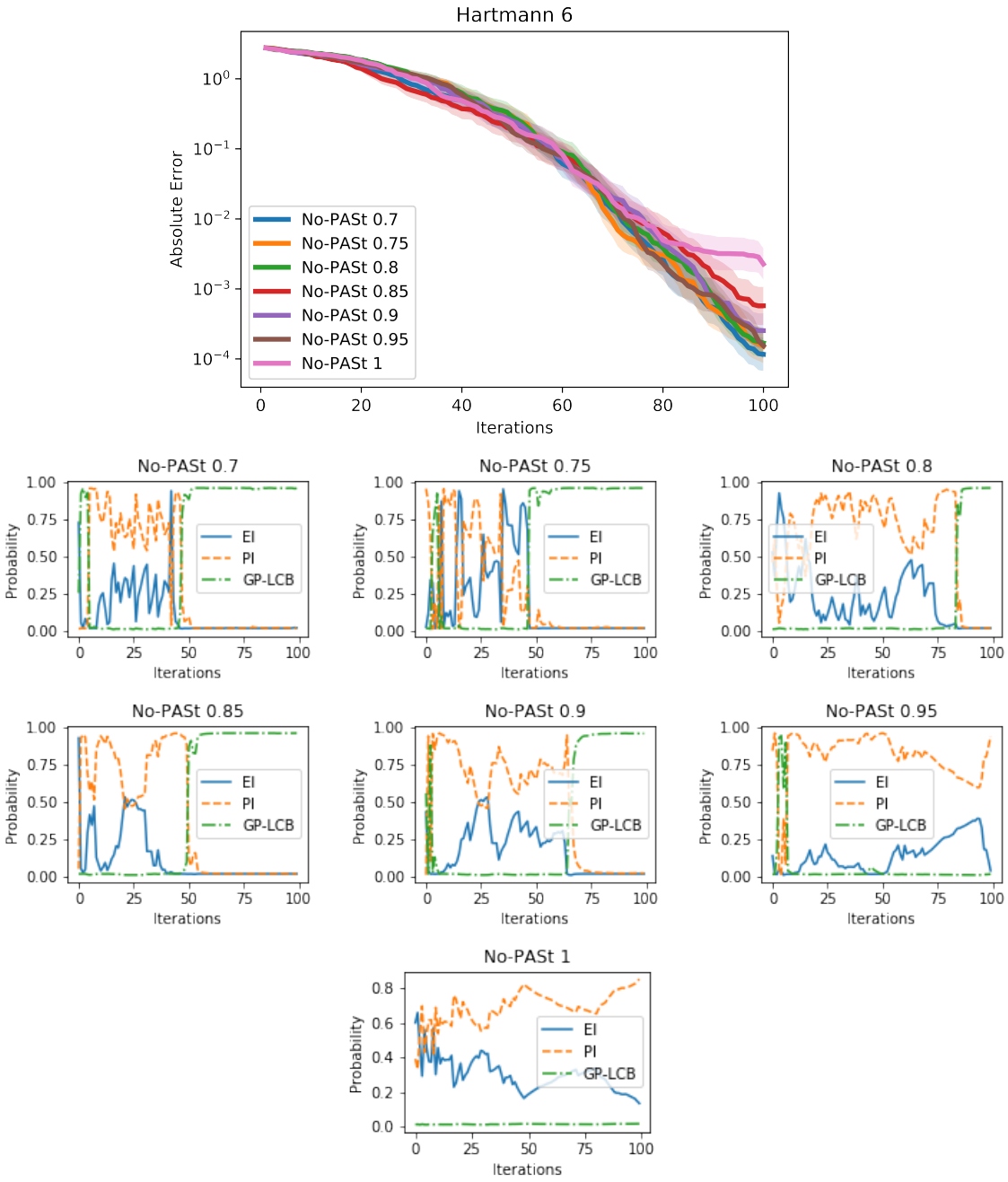

Figure 6 – The effect of different memory factors in the No-PASt-BO execution. As the memory factor increases, it takes more time to select an acquisition function as the one with higher probability.

The function is defined in Equation 3.7.

$$f(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)cos(x_1) + s \tag{3.7}$$

Where $a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, r = 6, s = 10$ and $t = \frac{1}{8\pi}$.

The Hartmann 3D function consists in a 3 dimensional function $x_1 \in [0,1], x_2 \in [0,1]$ and $x_3 \in [0,1]$, with one global minimum at $[(-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)]$ with value 0.397887, and four local minima. The function is defined in Equation 3.8.

$$f(x) = -\sum_{i=1}^{4} \alpha_i exp(-\sum_{j=1}^{3} A_{ij}(x_j - Pij)^2) \tag{3.8}$$

Where α, A, and P are given by Equation 3.9.

$$\begin{aligned} \alpha &= (1.0, 1.2, 3.0, 3.2)^T \\ A &= \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix} \\ P &= 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix} \end{aligned} \tag{3.9}$$

The Hartmann 6D function consists in a 6 dimensional function $x_1 \in [0,1], x_2 \in [0,1], x_3 \in [0,1], x_4 \in [0,1], x_5 \in [0,1]$ and $x_6 \in [0,1]$, with one global minimum at $(0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573)$ with value -3.32237, and six local minima. The function is defined in Equation 3.10.

$$f(x) = -\frac{1}{1.94}[2.58 + \sum_{i=1}^{4} \alpha_i exp(-\sum_{j=1}^{6} A_{ij}(x_j - Pij)^2)] \tag{3.10}$$

Where α, A, and P are given by Equation 3.11.

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T$$

$$A = \begin{pmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{pmatrix} \tag{3.11}$$

$$P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

In all experiments, the portfolio-based strategies used the same set of acquisition functions: PI, EI and GP-LCB. We have set $\xi = 0.01$ for PI and EI, and $\delta = 0.1$ and $\nu = 0.2$ for GP-LCB. Furthermore, it was evaluated the proposed No-PASt-BO with 7 different memory factors. For each of the experiments, 5 initial points were selected using the Latin Hypercube Sampling (LHS) approach (MCKAY *et al.*, 1979).

The results of these experiments can be found in Figure 7. For all of scenarios, a portfolio strategy obtained the best performance. A version of the No-PASt-BO achieved the best result in two of the three case, and was very close to the best version in the case it loses. Also, the No-PASt-BO with memory factor of 0.7 outperforms GP-Hedge in all of the executed experiments.

A second battery of tests was run for the same synthetic functions, but using 9 acquisition functions in the portfolio variants. These functions come from the same set of PI, EI and GP-LCB adding $\xi = 0.1$, $\xi = 1.0$, $\nu = 0.1$ and $\nu = 1$ to the already studied values. The results are shown in Figure 8. Overall, the RP performance was worse than the previous experiment with only 3 functions, with the remarkable exception of the Hartmann 3 experiment, due to the fact that as more functions were inserted in the portfolios, some of them were worse to the task at hand. In the same way, we can see that GP-Hedge got more competitive in general, but the No-PASt-BO still reaches lower optimized values faster.

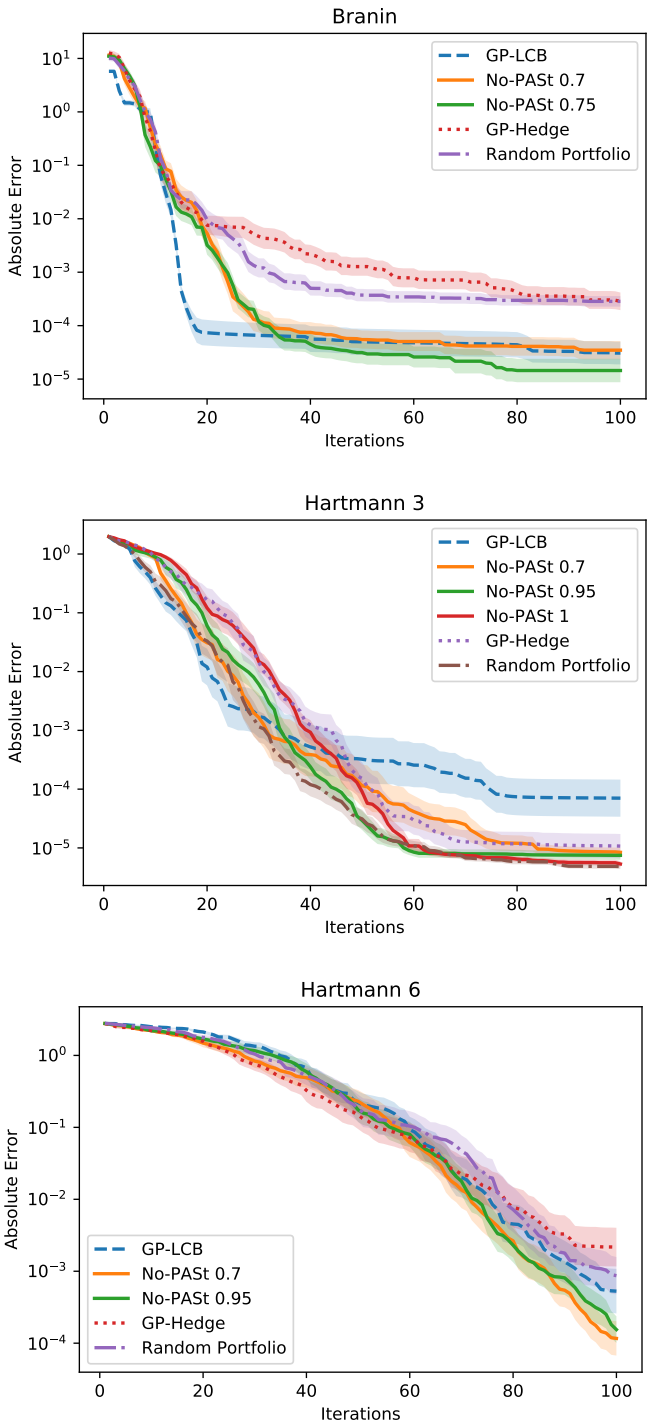

Figure 7 – Results of the minimization of the synthetic benchmark using 3 acquisition functions. Note that only the 2 best choices for the No-PASt-BO memory factor are shown. Also, only the best non-portfolio acquisition function is presented.

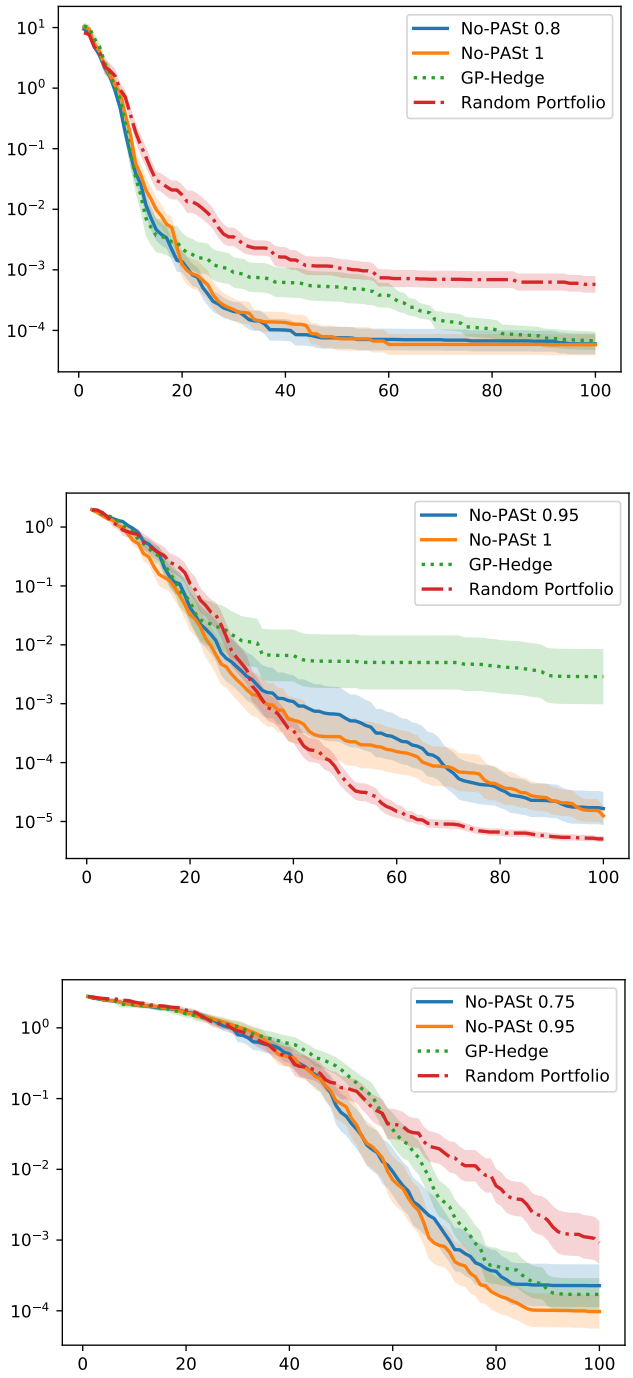

Figure 8 – Results of the minimization of the synthetic benchmark using 9 acquisition functions. Note that only the 2 best choices for the No-PASt-BO memory factor are shown. Also, only the best non-portfolio acquisition function is presented.

3.6.2 Real World Problems

To evaluate the No-PASt-BO and compare it with the available baselines in a real world problem, it was first considered a regression setting with the standard Boston Housing dataset[2]. The Support Vector Regressor (SVR) model (DRUCKER *et al.*, 1997) is used to predict house prices from the available attributes. The BO strategies have the task of optimizing the 3 SVR hyperparameters (gamma, C and epsilon), as implemented in the scikit learn toolbox (PEDREGOSA *et al.*, 2011).

In this experiment it was used a 10-fold cross-validation procedure, where the objective of each BO step is to minimize the average Root Mean Square Error (RMSE). For each experiment 100 optimization iterations were performed and each experiment was repeated 25 times. The obtained average RMSE values and their confidence intervals are shown in Figure 9.One can see that the No-PASt-BO variants achieved the best results both in terms of best final solution and faster lower values. Moreover, the GP-Hedge performed comparable to the the random portfolio strategy.

It was tackled a second real world problem to compare the evaluated BO methods in a regression setting which consists in predicting the average gearbox high-speed shaft temperature in a Wind Turbine Generator (WTG) from a given set of measures. Following the same methodology presented by Bangalore *et al.* (2017. Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2102 Accessed on September 19^{th} 2020), Souza *et al.* (2019), the input variables were: average active power, average rotor speed, average nacelle (the turbine outer casing) temperature and average outdoor temperature. A limit filter was applied to the available measured data, removing values that are physically absurd. After that, a clustering filter was applied to remove some outliers from the dataset and, finally, it was applied a continuity filter to remove isolated points. Details of these preprocessing steps can be seen in Bangalore *et al.* (2017. Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2102. Accessed on September 19^{th} 2020), Souza *et al.* (2019).

It was considered a nonlinear model comprised of a Multilayer Perceptron (MLP) network with a single hidden layer and hyperbolic tangent hidden activations. The model was trained via stochastic gradient descent optimizer. The task of the BO was to minimize the RMSE of the predicted temperature in a hold-out validation set, varying over the scikit-learn MLPRegressor hyperparameters (PEDREGOSA *et al.*, 2011): neurons in the hidden layer,

[2] Available at <https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>.

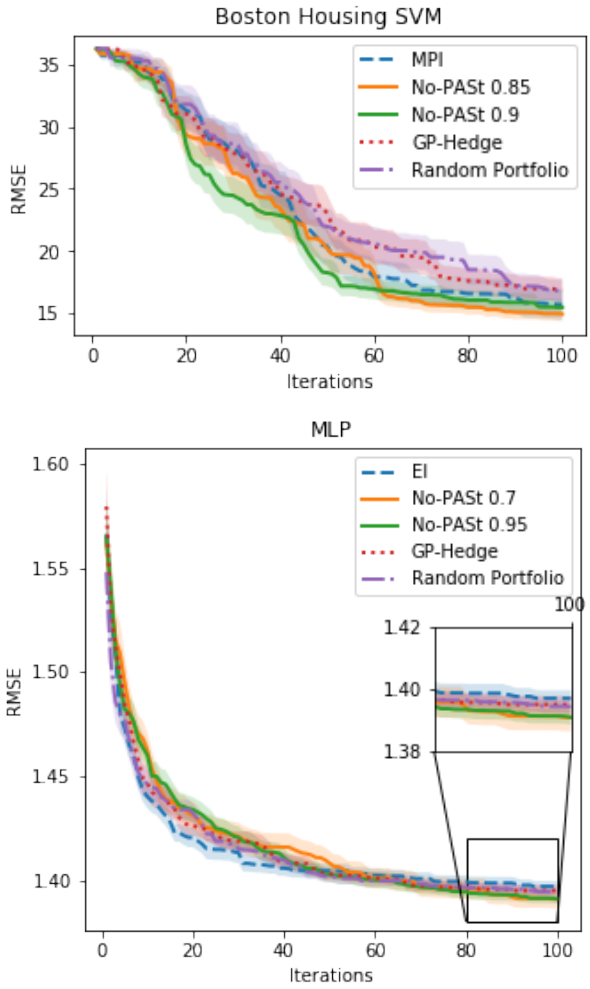

Figure 9 – Results for automatic tuning of SVM and MLP hyperparameters. In both cases a No-PASt-BO version has achieved the best performance.

learning rate, alpha (weight decay hyperparameter) and momentum. The data used in this experiment was obtained from a WTG located in Brazil.

Results are presented in Figure 9. For the Boston Housing dataset, the proposed No-PASt-BO achieves the best results. For the problem of predicting the average gearbox high-speed shaft temperature in a WTG, although all the methods obtained somehow comparable results, a detailed inspection indicates that No-PASt-BO variants obtained the best RMSE values at the end.

4 SETUP-BO

4.1 No-PASt-BO limitations

The presented No-PASt-BO introduced 2 hyperparameters which influence the algorithm performance, and, although a recommended configuration is presented by Vasconcelos *et al.* (2019), their values choices are task-specific. This way, tuning these hyperparameters may be an expensive challenge, above all considering that BO is usually applied in problems where the objective function is costly to evaluate. This way it seems important to use a strategy that can choose the best hyperparameter for each case.

To solve the hyperparameter tuning problem, without including more evaluations to the objective function, we propose applying TS (THOMPSON, 1933). TS is a well established probabilistic technique, and it has been successfully used in general bandit problems more recently (CHAPELLE; LI, 2011; AGRAWAL; GOYAL, 2012; RUSSO *et al.*, 2018). In summary, when using TS we randomly sample from the posterior of a reward function and select the candidate (e.g. an arm in a multi-armed bandit set up) with the highest sampled reward. In the context of BO, TS techniques have been used within an entropy-based acquisition function (HERNÁNDEZ-LOBATO *et al.*, 2014), to sample from the posterior of the surrogate model parameters (PALMA *et al.*, 2019), to bypass the need of an acquisition function by directly sampling from the maximum distribution (BIJL *et al.*, 2016) and to enable batch parallel optimization (KANDASAMY *et al.*, 2018).

More specifically, we investigate TS techniques to sample from the posterior of the portfolio hyperparameters. At each iteration such sample is used to select which acquisition function will guide the choice of the next queried point. As we will detail later on, this is equivalent to consider each acquisition function in a portfolio as an arm of a multi-armed bandit problem. Furthermore, the posterior of each portfolio hyperparameter is analytically updated after each optimization step thanks to careful conjugate prior choices.

Our new method, named SeTuP-BO, has the following goals: (*i*) to preserve the practicality of the GP-Hedge portfolio allocation strategy; (*ii*) to maintain, or to improve, the performance of the original No-PASt-BO algorithm; (*iii*) to mitigate the need of manual intervention in the optimization procedure.

4.2 Thompson Sampling

According to Russo *et al.* (2018), TS is a method for online decision problems, where at each iteration one must take an action trying to balance between exploitation and acquiring new information.

Russo *et al.* (2018) define the TS general problem as, given a system and an agent, the agent takes a sequence of actions $x_1, x_2, x_3 \ldots$ selecting each from a set χ, which can be finite or infinite. After each action x_i, the agent observes an output y_i which is generated based on the conditional probability $q(\cdot|x_i)$, with an associated reward $r(y_i)$. The initial probability distribution is usually unknown, and because of that, it is represented by a prior.

In the proposed method, the TS procedure is used to update the value of the hyperparameters introduced by the No-PASt-BO by Vasconcelos *et al.* (2019), decreasing the impact of the initial selected value, and making the algorithm adapt itself according to the problem. More specifically, we use priors with large variances to represent the initial uncertainty and update them at each iteration in a way that at the advanced steps of the experiment the posteriors better represent the uncertainty about the task.

In some sense, a multi-armed bandit problem, a common TS application scenario, is similar to how BO handle an acquisition function. In both cases we have some candidate points to evaluate and consider the uncertainty in the task to balance between exploration and exploitation to choose the next point to query. TS faces the multi-armed bandit problem as a portfolio of one-armed bandit problems and try to make decisions considering all of the options that will lead to the best output. We can view the portfolio of acquisition functions as a multi-armed bandit problem, in which we have to select a function that will lead to the best evaluation of the black-box function. Thereby, applying TS to help selecting which acquisition function to choose from seems a promising approach.

4.3 SeTuP-BO

The two hyperparameters introduced by Vasconcelos *et al.* (2019) which will be handled by our TS-based approach are the memory factor and the η.

For the memory factor parameter m, a Beta probability distribution was chosen as the prior, since it covers the range of possible values for m in the interval $[0, 1]$. We also choose a Bernoulli likelihood function, where a success (a value equal one) is defined by a result better

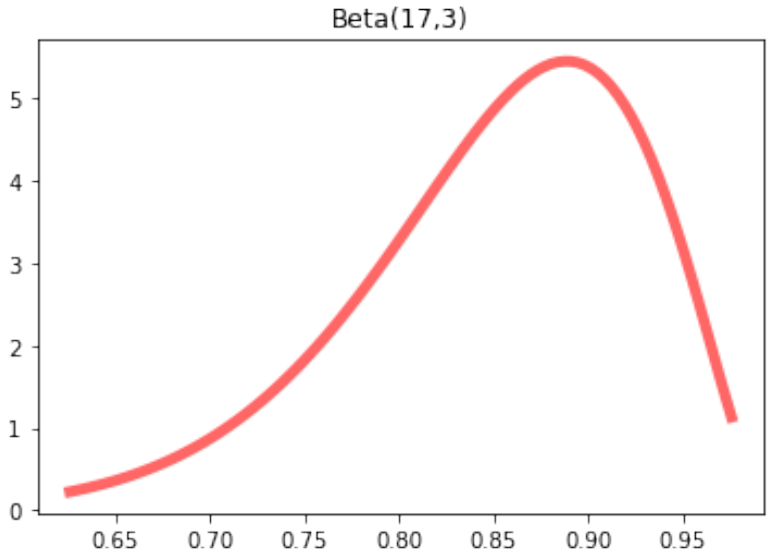

Figure 10 – Beta(17,3) probability density function.

than the best result so far. Otherwise, we consider it a failure (a zero value). Thus, the posterior can easily be achieved, since Beta and Bernoulli are conjugate distributions. The beta probability distribution was empirically defined: $\text{Beta}(17,3)$, and its probability density function is shown in Figure 10.

For the η value, a Gamma probability distribution was chosen as the prior due to its positive domain, $\eta \in \mathbb{R}^+$. Another reason for choosing the Gamma distribution is because it is conjugate with the exponential distribution, as will be discussed later in this section. Examples of application of Gamma distribution are used by Mesquita *et al.* (2019), Mesquita *et al.* (2017. Available at http://www.sciencedirect.com/science/article/pii/S0925231217304320. Accessed on September 19^{th} 2020), where the Gamma is used to model squared random variables. The likelihood function has the form of the Boltzmann distribution, which represents the probability of a function being in a certain state (in our case, selecting certain acquisition function), given the energy of the estate (in our case, the score).

The Boltzmann likelihood is not conjugate with the Gamma distribution prior. This means that there is not a closed form for the correspondent posterior, which demands approximate inference. In order to achieve conjugate posterior updates, we will make some additional considerations.

First, it is a fact that, if all of the rewards are not equal, there will be at least one $r_{\text{max}} = 0$ and one $r_{\text{min}} = -1$, due to the normalization step. Since our experiments consider 3 acquisition functions, the probability of choosing an acquisition function j is given by Eq. (3.4).

To make use of conjugate properties, we approximate Eq. (3.4) with the following

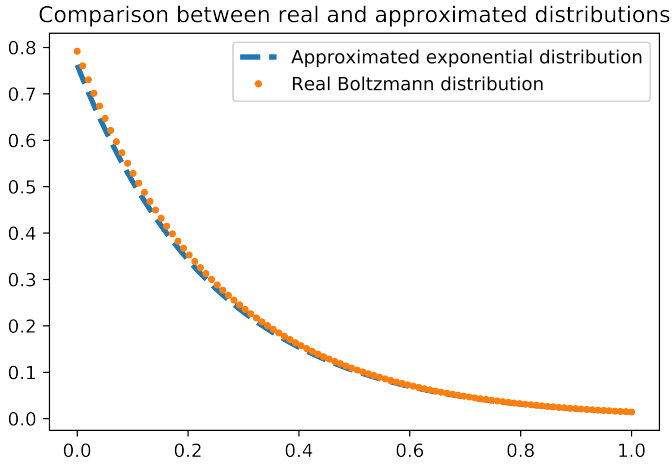

Figure 11 – Comparison between the real Boltzmann distribution and the approximated exponential distribution used as likelihood functions. For the value of $r_{\text{intermediary}}$ in Eq. (3.4) we take 100 samples uniformly distributed in the interval $[-1,0]$. The Boltzmann distribution curve is the average of the 100 obtained values. The exponential distribution curve considers the constant C in Eq. (4.1) equal to the inverse of the mean value of the denominator in Eq. (3.4). For both distributions, it is assumed $\eta = 4$.

exponential:

$$p_j(t) = C\exp(\eta r_j(t-1)). \tag{4.1}$$

Although such an approximation is not completely accurate, it respects the fact that η should assume higher values for small $r_j(t-1)$ and low values otherwise. Importantly, the problem now consists of a Gamma-exponential conjugate pair, which has a closed-form expression for the posterior distribution. The comparison between the real and the approximated likelihood is shown in Figure 11. The value $r_{\text{intermediary}}$ was sampled 100 times uniformly distributed between -1 and 0. The reported Boltzmann distribution is the mean of the 100 obtained distributions. The reported exponential distributions considers the constant C as the inverse of the mean value of the denominator in Eq. (3.4), considering all sampled values for $r_{\text{intermediary}}$. For both plots, it is assumed $\eta = 4$

By construction, $r_j(t-1)$ is always a number in the interval $[-1,0]$, which means that the sampled value for the argument in the exponential will always be in the interval $[0,1]$. Thus, an initial $\eta = 4$ is a reasonable choice, because it makes the 98^{th} percentile to be below 1 for the exponential distribution. Because of this, the two parameters of the Gamma distribution, i.e., the prior for η, are chosen in way to have mean 4. This way, the Gamma prior becomes Gamma$(40,10)$, which is illustrated in Figure 12.

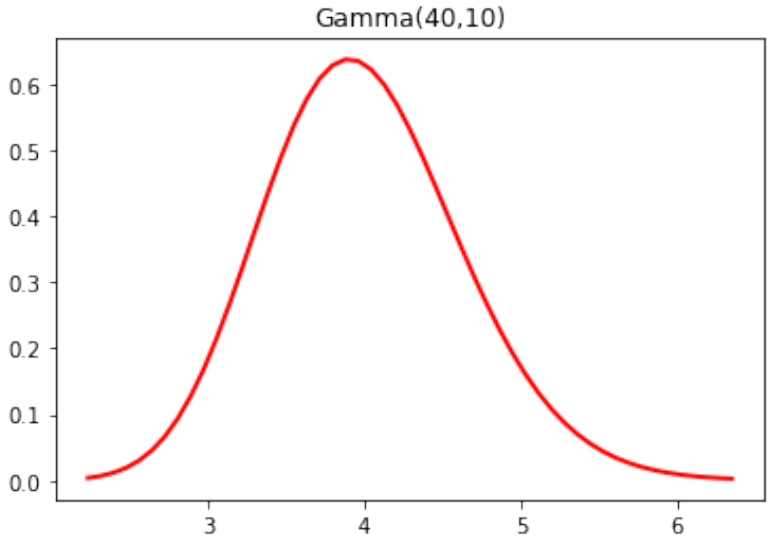

Figure 12 – Gamma(40,10) probability density function.

From the aforementioned two conjugate pairs, it is possible to avoid choosing the two hyperparameters introduced by Vasconcelos *et al.* (2019) for the No-Past-BO, while also enabling them to adapt for each scenario during the optimization steps. To do this, at each iteration a value is sampled for η from its Gamma distribution. Then, a reward is sampled from the exponential distribution given by Eq. (4.1). Those steps are formalized as follows:

$$\begin{aligned} \eta &\sim \text{Gamma}(\alpha,\beta), \\ p_j(r_j(t)|\eta) &= \text{exponential}(\eta). \end{aligned} \tag{4.2}$$

After sampling the reward, the posterior distribution is updated as follows:

$$\begin{aligned} \alpha &\leftarrow \alpha + 1, \\ \beta &\leftarrow \beta + r_j(t). \end{aligned} \tag{4.3}$$

The aforementioned steps can be interpreted as the following rule: we tend to choose the the highest score, so the most probable value for $r_j(t)$ is 0, and as we keep updating α and β the mean value of the Gamma distribution tends to increase, what means we will choose the most probable with a higher probability.

A similar process is applied to the variable m. First, a value is sampled from the distribution $\text{Beta}(a,b)$. After that, we consider a success if the sampled point leads to the best solution found so far, and a failure otherwise. The conjugate pair then becomes

$$\begin{aligned} m &\sim \text{Beta}(a,b), \\ p(z = \mathbb{I}(y_t < y_{\min})|m) &= \text{Bernoulli}(m), \end{aligned} \tag{4.4}$$

where $y_{\min}$ is the minimum value found by the algorithm up to the current iteration and $\mathbb{I}(\cdot)$ is the indicator function, which is equal 1 for a true argument and 0 otherwise.

The posterior update as follows:

$$\begin{aligned} a &\leftarrow a+1, \quad \text{if } y_t < y_{min}, \\ b &\leftarrow b+1, \quad \text{otherwise.} \end{aligned} \tag{4.5}$$

The aforementioned steps can be interpreted as the following rule: if the results are improving, than keep the previous evaluations, otherwise, decrease the past evaluations' importance. The final algorithm is presented in Algorithm 3.

Algorithm 3: SETUP-BO

Set $G_j(0) = 0$ for $j = 1,2,\ldots,J$
for t = 1,2,... **do**
  Sample hyperparameter $\eta \sim \text{Gamma}(\alpha,\beta)$
  Sample hyperparameter $m \sim \text{Beta}(a,b)$
  Nominate points from each acquisition function h_j:
  $\boldsymbol{x}_j(t) = \arg\max_{\boldsymbol{x}} h_j(\boldsymbol{x})$
  Compute $r_{\min}(t-1) = \min_j(G_j(t-1))$
  Compute $r_{\max}(t-1) = \max_j(G_j(t-1))$
  Compute the normalized rewards:
  $r_j(t-1) = \frac{G_j(t-1)-r_{\max}(t-1)}{r_{\max}(t-1)-r_{\min}(t-1)}$
  Select a nominee $\boldsymbol{x}(t) = \boldsymbol{x}_j(t)$ with probability
  $p_j(t) = \frac{\exp(\eta r_j(t-1)}{\sum_{j'=1}^{J}\exp(\eta r_{j'}(t-1)))}$.
  Compute $y(t)$ by evaluating the objective on point $\boldsymbol{x}(t)$.
  Augment the data $\mathscr{D}_t$ with the new pair $(\boldsymbol{x}(t), y(t))$.
  Update the surrogate GP model.
  Update the rewards $G_j(t) = mG_j(t-1) - \mu(\boldsymbol{x}_j(t))$ from the updated GP posterior.
  Update a = a + 1, if $y(t)$ is the best point evaluate so far, otherwise update b = b + 1.
  Update $\alpha \leftarrow \alpha + 1$
  Update $\beta \leftarrow \beta + x(t)$
end for

4.4 Empirical evaluation

4.4.1 Initial experiments

To compare the results with the ones obtained by Vasconcelos *et al.* (2019), the same 3 standard benchmark functions were used: Branin, Hartmann 3 and Hartmann 6[1]. Also the same

[1] Definitions available at <https://www.sfu.ca/~ssurjano/optimization.html>.

BO framework, *GPy* and *GPyOpt* ((GPy, since 2012. Availabe at http://github.com/SheffieldML/GPy. Accessed on September 19^{th} 2020; The GPyOpt authors, 2016. Available at http://github.com/SheffieldML/GP Accessed on September 19^{th} 2020)), were used.

Branin is a two-dimensional function, while Hartmann 3 is a three-dimensional function and Hartmann 6 is a six-dimensional function, which makes optimizing the Branin the easiest task and the Hartmann 6 the hardest.

For all experiments in this subsection, three acquisition functions were used as baselines: PI and EI, both with $\xi = 0.01$, and GP-LCB, with $\delta = 0.1$ and $\nu = 0.2$.

We also consider four portfolio strategies, each of them using the three base acquisition function with the given configuration. Three of the portfolio strategies were used as a baseline. First the random portfolio, which consists of randomly choosing one of the available acquisition functions at each iteration. Second the GP-Hedge algorithm (HOFFMAN *et al.*, 2011). Third the No-PASt-BO (VASCONCELOS *et al.*, 2019), using the recommended configuration $\eta = 4$ and $m = 0.7$. Finally, the fourth method used is the proposed SeTuP-BO.

For each of the experiments, 5 initial points obtained with the LHS approach (MCKAY *et al.*, 1979)) were used. Each experiment was run 25 times for 100 iterations. The mean and standard deviation are reported in Figure 13.

The proposed SeTuP-BO approach obtained the best results for Branin and Hartmann 6. Although the best performance in the Hartmann 3 function was obtained by the random portfolio, the proposed method still outperformed the GP-Hedge and is competitive when compared to the No-PASt-BO strategies.

4.4.2 HPO Meta-Surrogate Benchmarking

Although SeTuP-BO has performed well in the three experiments so far, more extensive tests are necessary. To facilitate the execution of HPO tasks, Klein *et al.* (2019) proposed PROFET, a framework for evaluating the performance of HPO algorithms. PROFET contains a set of 1000 surrogate benchmarks for SVM, Fully Connected Network (FNET) and XGBoost, which are cheap to evaluate, making it possible to run several tests without tremendous computational resources.

The SVM tasks optimize two dimensions, while the FNET tasks have six dimensions and the XGBoost tasks have eight. The optimized variables and the corresponding search space is shown in Table 1.

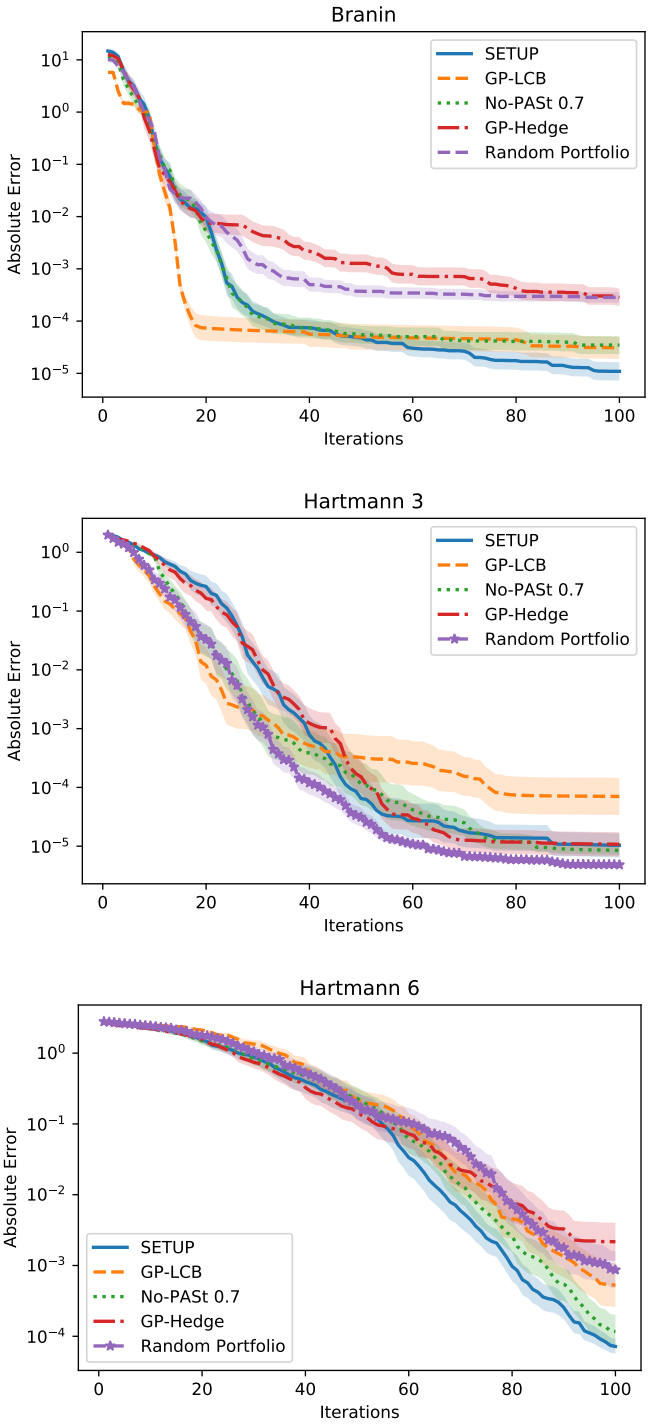

Figure 13 – Minimization of the synthetic benchmarks. Only the best non-portfolio acquisition function is presented.

Table 1 – Meta surrogate functions dimensions.

Method	Dimension	Space
SVM	C	[-10, 10]
	gamma	[-10,10]
FCNET	learning rate	[1e-6, 1e-1]
	batch size	[8, 128]
	neurons in first hidden	[16, 512]
	neurons in second hidden	[16, 512]
	first hidden dropout rate	[0, 0.99]
	second hidden dropout rate	[0, 0.99]
XGBoost	learning rate	[1e-6, 1e-1]
	gamma	[0, 2]
	alpha	[1e-5, 1e3]
	lambda	[1e-4, 1e3]
	number of estimators	[10, 500]
	subsample	[1e-1, 1]
	max depth	[1, 15]
	minimum child weight	[0, 20]

The experiments were executed for the SVM, FNET and XGBoost in the first 30 multiples of 5 tasks. For each of the tasks, the same methods of Subsection 4.4.1 (with the exception of the No-PASt-BO) were run 25 times for 100 iterations. For all the experiments the central point of the search space is considered as the single initial point. The output evaluations for SVM, FNET were multiplied by 100 to keep them in the $[0, 100]$ interval.

The PROFET framework also provides tools for comparing different tasks and executions. Figure 14 shows the ranks at each iteration for the all of the performed tasks and the cumulative distribution probability of finding in each task a solution which is at least 2% worse than the best solution found for that task, grouped by problem solved.

We can see that the proposed SeTuP-BO gets the best results for the FNET, while being the second-best for the other two problems. It is also visible that the difference between it and the GP-Hedge is minimum in the XGBoost (the only case where GP-Hedge achieves the best performance), with SeTuP-BO achieving the best performance until iteration 80. These two are problems with more dimensions (FNET has 6 dimensions and XGBoost has 8), this fact makes them more difficult than the SVM problem, and still, for both of them, the proposed SeTuP-BO has competitive results.

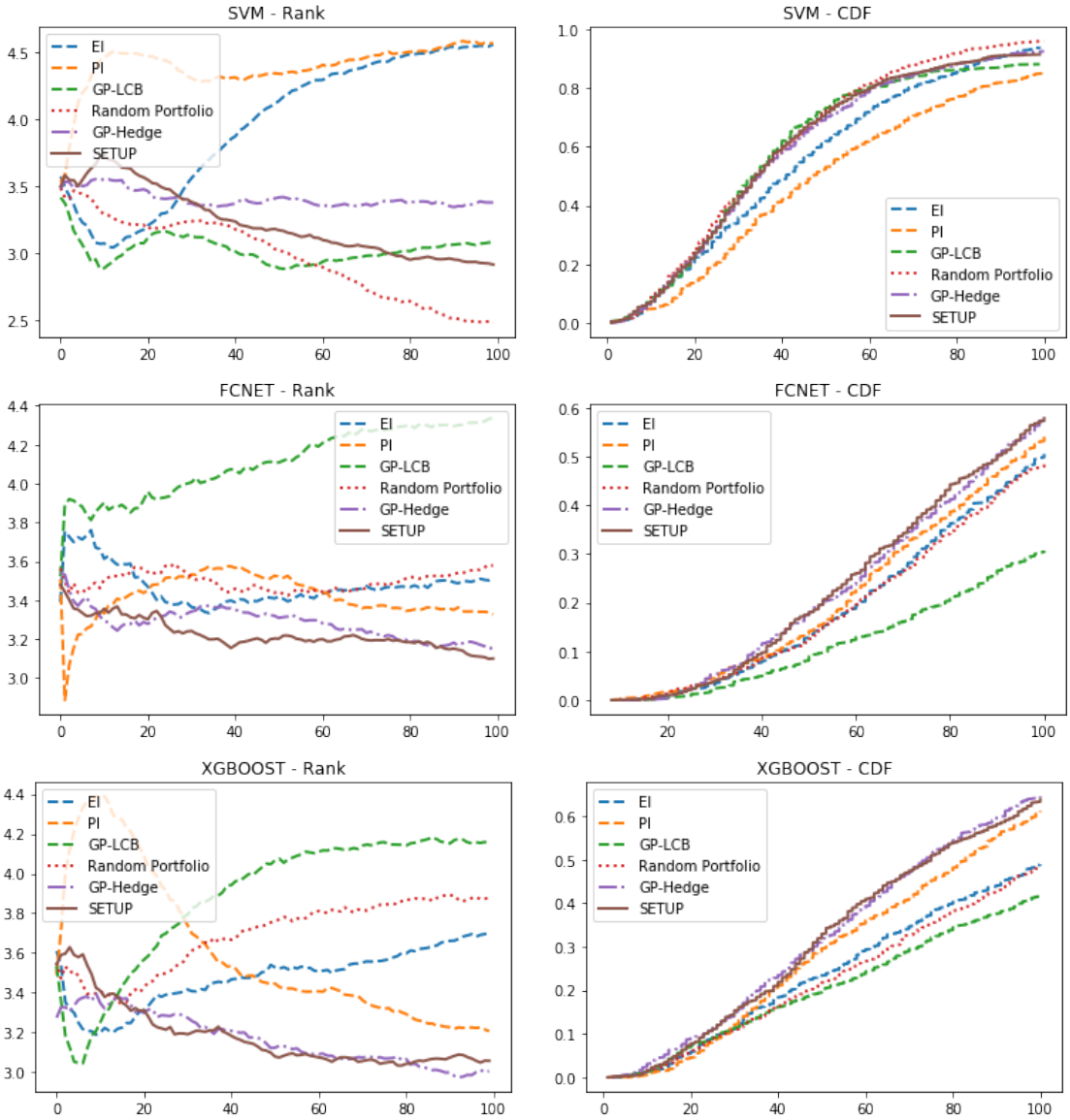

Figure 14 – In the first column, it is presented the rank for each BO along 100 iterations averaged for 25 executions and 30 tasks. The first row corresponds to the SVM, the second row corresponds to the FNET, and the third row corresponds to the XGBoost. In the second row, it is presented the cumulative distribution curves to find a value that is at most 2% higher than the best value found among all executions within each scenario.

4.4.3 Real-world HPO task

The final experiment consists in applying BO to a real-world scenario. The problem consists in modeling the normal behavior of a generator unit of a small hydroelectric power plant. This task is relevant because once the normal behavior is modeled it is possible to identify when the the real operation is diverging from the predicted one, and this behavior can lead to failures. The normal behavior modeling is widely used in failures predictions. For instance, Souza *et al.* (2019) apply such a technique to wind turbine generators.

The used data was collected from a small hydroelectric plant with nominal power of 30 MW with a sample frequency of 10 minutes between October 2018 and January 2019. The subsystem studied is the generator stator's heat exchanger, which is responsible to dissipate the heat generated in the electric generation process. The process ha an inherent seasonal dependency, as weather conditions affect directly its dynamics. The data available was not enough to represent all the seasons of the year, however it is not expected that this could strictly interfere in the final results. Finally the variables available were separated in outputs when they were inside the stator's heat exchanger closed environment and input when they were outside. The full list of inputs and outputs are presented in Table 2.

Table 2 – Modelled inputs and outputs by component.

Component	Sensor	Number of Sensors	Type
Cold Water	Temperature (°C)	1	Input
Warm Air	Temperature (°C)	6	Output
Cold Air	Temperature (°C)	6	Output
Stator and Rotor	Stator Winding Temperature (°C)	12	Output
	Stator Core Temperature (°C)	6	Output
	Active Power (kW)	1	Input
Warm Water	Temperature (°C)	1	Output

We use BO to optimize a NARX model comprised of a MLP with two hidden layers, trained with the Sthocastic Gradient Descendent (SGD) algorithm, using the provided algorithm from the *scikit-learn* toolbox (PEDREGOSA *et al.*, 2011). The BO has only one initial point, given by the middle point for each considered dimension. The optimized variables are presented in Table 3. For the discrete dimensions we followed the approach proposed by Garrido-Merchán e Hernández-Lobato (2020), which replaces the GP kernel function evaluation $k(x_i, x_j)$ by $k(T(x_i), T(x_j))$, where $T(\cdot)$ is a transformation which rounds the integer-domain

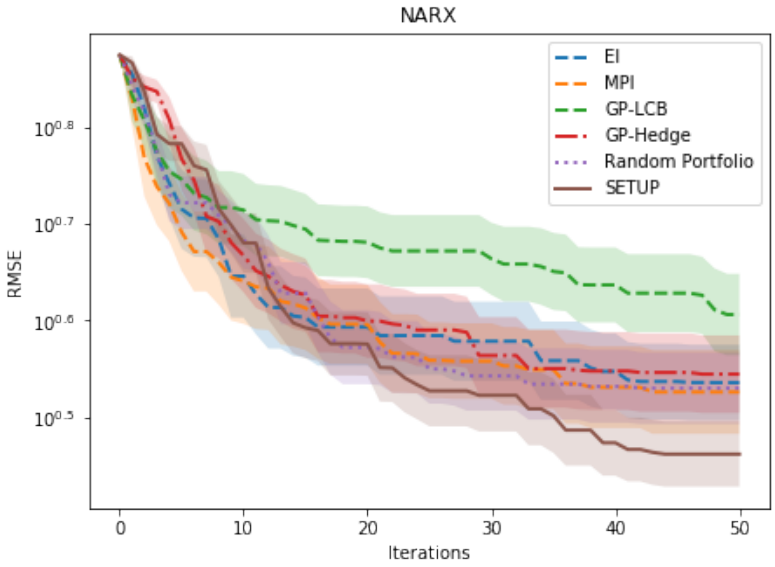

Figure 15 – RMSE across the 25 repetitions of the hyperparameter tuning of the NARX model for the stators' heat exchanger temperatures.

variables without changing the others.

Each run was executed for 50 iterations and the experiment was reproduced 25 times. The obtained means and standard deviations are reported in Figure 15. The proposed SeTuP-BO method outperformed the others, being the only one achieving results below $10^{0.5}$.

Table 3 – NARX model variables.

Variable	Space	Type
learning rate	[1e-6, 1e-1]	Continuous
momentum	[0.7, 0.95]	Continuous
alpha	[1e-7, 1e-1]	Continuous
batch size	(8, 16, 32, 64, 128)	Discrete
neurons in first hidden	(16,32,64,126,256,512)	Discrete
neurons in first hidden	(16,32,64,126,256,512)	Discrete
temperature lag order (in 10 minutes units)	(1,2,3,4,5,6)	Discrete
temperature lag period (in 10 minutes units)	(1,2,3,4,5,6)	Discrete
electric lag order (in 10 minutes units)	(1,2,3,4,5,6)	Discrete
electric lag period (in 10 minutes units)	(1,2,3,4,5,6)	Discrete

To illustrate the prediction performance, the best predicted values obtained by the SeTuP-BO are reported in Figure 16. The model is able to predict accurately the variables: Stator Winding Temperature, Stator Core Temperature, Stator Hot Air Temperature. However, it has some problems in predicting the values for Stator Cold Air Temperature and Stator Hot Water Temperature. Possible reasons for that are that these variables are more sensitive to external factors. Besides, the poor performance for the Hot Water may be due the fact that it only has a

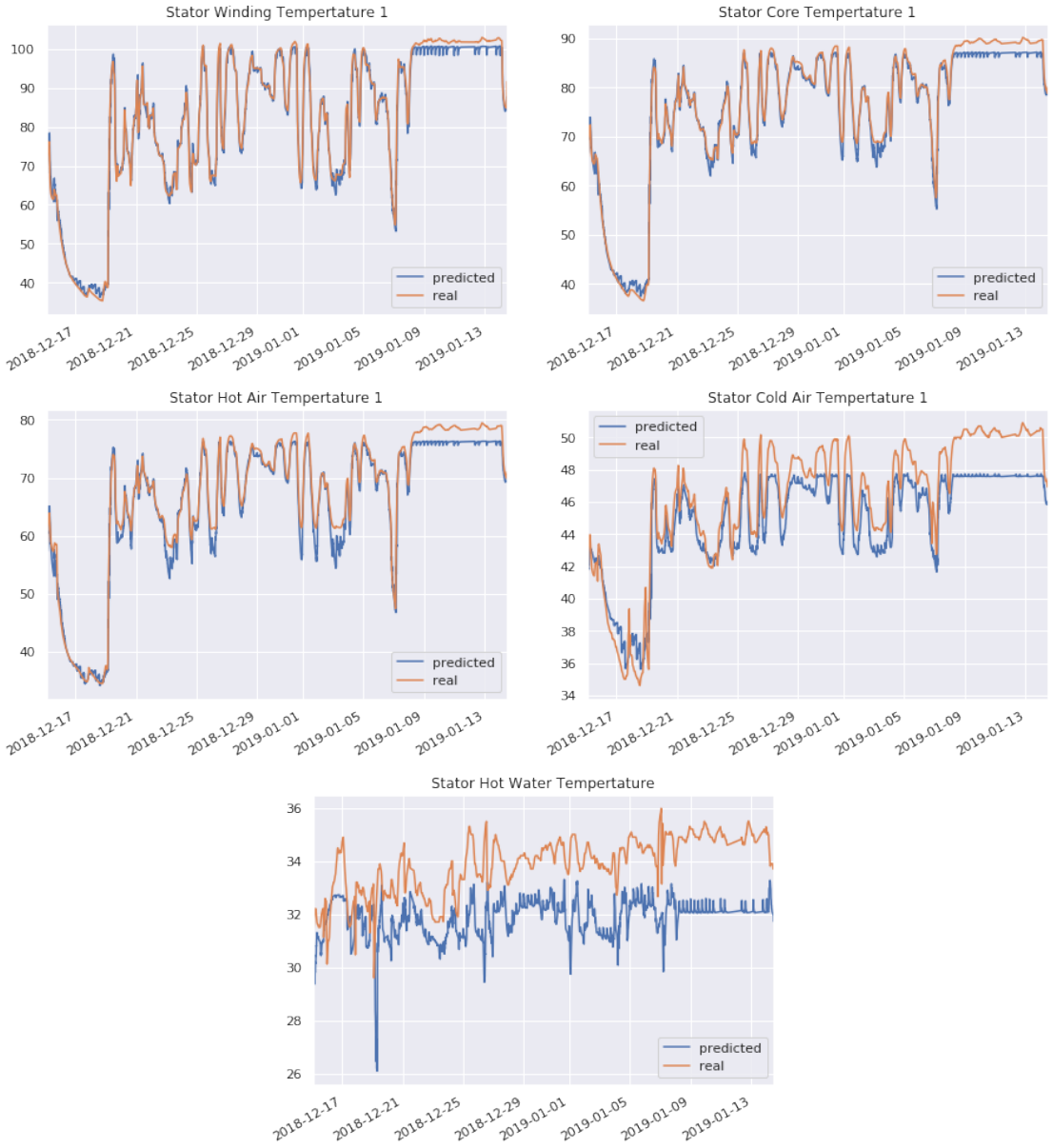

Figure 16 – Predictions for the stator's heat exchange outputs.

single sensor, so the model has less information about that output in the training step.

5 CONCLUSION AND SUGGESTIONS FOR FURTHER WORK

This work has proposed two new methods for black-box Bayesian optimization of unknown functions. No-PASt-BO and SeTuP-BO are based on the popular GP-Hedge method (HOFFMAN *et al.*, 2011) that creates an adaptive portfolio of acquisition functions.

The No-PASt-BO aimed to tackle some of the GP-Hedge's limitations, such as the sometimes undesirable high influence of far past evaluations, by incorporating a limited memory and a normalization mechanism. The SeTuP-BO model focused on removing the hyperparameters introduced by No-PASt-BO, while keeping its improvement over GP-Hedge. Both methods were evaluated on synthetic and real-world optimization tasks, in which they obtained competitive performance with respect to the other evaluated strategies.

Further investigations may verify how to incorporate non-myopic concepts to the portfolio BO methodology, such as the ones explored by González *et al.* (2016). That would enable more clever decision making when there is a known limited budget in terms of number of objective evaluations.

Another interesting subject for investigation is the task of function optimization with constraints that are known *a priori*. Strategies such as the ones presented by Gardner *et al.* (2014) may turn the portfolio framework even more applicable in real world problems.

One can also work on improving the applicability of SeTuP-BO. A first study needs to made on the the TS reward process of SeTuP-BO, once all of the evaluated functions were noiseless, and the used methodology may fail when same points are evaluated differently. Another point of attention is the approximations made in order to improve performance. Although they are useful for the scenario of three acquisition functions, they may not be valid when using more acquisition functions, like the scenario with 9 acquisition functions studied by both GP-Hedge (HOFFMAN *et al.*, 2011) and No-PASt-BO (VASCONCELOS *et al.*, 2019).

It is important to mention that both the No-PASt-BO and the SeTuP-BO outperformed GP-Hedge in most of the evaluated applications, while maintaining its simplicity and general applicability. The latter features may enable No-PASt-BO to become the default off-the-shelf method for portfolio-based BO, while the SeTuP-BO may also fulfill this role after surpassing the aforementioned limitations.

BIBLIOGRAPHY

AGRAWAL, S.; GOYAL, N. Analysis of Thompson sampling for the multi-armed bandit problem. In: **Conference on learning theory**. Edinburgh, Scotland: JMLR, INC. and Microtone Publishin, 2012. p. 39–1.

AUER, P.; CESA-BIANCHI, N.; FREUND, Y.; SCHAPIRE, R. E. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In: IEEE. **Proceedings of IEEE 36th Annual Foundations of Computer Science**. Milwaukee, WI, USA, 1995. p. 322–331.

BANGALORE, P.; LETZGUS, S.; KARLSSON, D.; PATRIKSSON, M. An artificial neural network-based condition monitoring method for wind turbines, with application to the monitoring of the gearbox. **Wind Energy**, v. 20, n. 8, p. 1421–1438, 2017. Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2102. Accessed on September 19^{th} 2020.

BIJL, H.; SCHÖN, T. B.; WINGERDEN, J.-W. van; VERHAEGEN, M. A sequential Monte Carlo approach to Thompson sampling for Bayesian optimization. **arXiv preprint arXiv:1604.00169**, 2016.

BRANIN, F. H. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. **IBM Journal of Research and Development**, IBM, v. 16, n. 5, p. 504–522, 1972.

BROCHU, E.; CORA, V. M.; FREITAS, N. D. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. **arXiv preprint arXiv:1012.2599**, 2010.

CALANDRA, R.; SEYFARTH, A.; PETERS, J.; DEISENROTH, M. P. Bayesian optimization for learning gaits under uncertainty. **Annals of Mathematics and Artificial Intelligence**, Springer, v. 76, n. 1-2, p. 5–23, 2016.

CHAPELLE, O.; LI, L. An empirical evaluation of Thompson sampling. In: **Advances in neural information processing systems**. Guilin, China: Springer, 2011. p. 2249–2257.

CHATZILYGEROUDIS, K.; RAMA, R.; KAUSHIK, R.; GOEPP, D.; VASSILIADES, V.; MOURET, J.-B. Black-box data-efficient policy search for robotics. In: IEEE. **Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on**. Vancouver, BC, Canada, 2017. p. 51–58.

COX, D.; JOHN, S. Multidisciplinary design optimization: State of the art, chapter sdo: A statistical method for global optimization. **Society for Industrial and Applied Mathematics, Philadelphie**, p. 315–329, 1997.

DRUCKER, H.; BURGES, C. J.; KAUFMAN, L.; SMOLA, A. J.; VAPNIK, V. Support vector regression machines. In: **Advances in neural information processing systems**. Denver, CO, USA: MIT Press, 1997. p. 155–161.

FALKNER, S.; KLEIN, A.; HUTTER, F. Bohb: Robust and efficient hyperparameter optimization at scale. In: PMLR. **International Conference on Machine Learning**. Stockholm, Sweden, 2018. p. 1436–1445.

FEURER, M.; HUTTER, F. Hyperparameter optimization. In: **Automated Machine Learning**. [*S.l.*]: Springer, Cham, 2019. p. 3–33.

GARDNER, J. R.; KUSNER, M. J.; XU, Z. E.; WEINBERGER, K. Q.; CUNNINGHAM, J. P. Bayesian optimization with inequality constraints. In: PMLR. **ICML**. Beijing, China, 2014. p. 937–945.

GARRIDO-MERCHÁN, E. C.; HERNÁNDEZ-LOBATO, D. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. **Neurocomputing**, Elsevier, v. 380, p. 20–35, 2020.

GONZÁLEZ, J.; OSBORNE, M.; LAWRENCE, N. Glasses: Relieving the myopia of bayesian optimisation. In: PMLR. **Artificial Intelligence and Statistics**. [*S.l.*], 2016. p. 790–799.

GPy. **GPy: A Gaussian process framework in python**. since 2012. Availabe at http://github.com/SheffieldML/GPy. Accessed on September 19^{th} 2020.

HARTMAN, J. K. Some experiments in global optimization. **Naval Research Logistics Quarterly**, Wiley Online Library, v. 20, n. 3, p. 569–576, 1973.

HENNIG, P.; SCHULER, C. J. Entropy search for information-efficient global optimization. **The Journal of Machine Learning Research**, JMLR. org, v. 13, n. 1, p. 1809–1837, 2012.

HERNÁNDEZ-LOBATO, J. M.; HOFFMAN, M. W.; GHAHRAMANI, Z. Predictive entropy search for efficient global optimization of black-box functions. In: **Advances in neural information processing systems**. Montréal, Montréal, Canada: MIT Press, 2014. p. 918–926.

HOFFMAN, M. D.; BROCHU, E.; FREITAS, N. de. Portfolio allocation for bayesian optimization. In: **UAI**. Barcelona, Spain: AUAI Press, 2011. p. 327–336.

KANDASAMY, K.; KRISHNAMURTHY, A.; SCHNEIDER, J.; PÓCZOS, B. Parallelised Bayesian optimisation via Thompson sampling. In: PMLR. **International Conference on Artificial Intelligence and Statistics**. Lanzarote, Spain, 2018. p. 133–142.

KLEIN, A.; DAI, Z.; HUTTER, F.; LAWRENCE, N.; GONZALEZ, J. Meta-surrogate benchmarking for hyperparameter optimization. In: **Advances in Neural Information Processing Systems**. Vancouver, Canada: [*s.n.*], 2019. p. 6267–6277.

KLEIN, A.; FALKNER, S.; BARTELS, S.; HENNIG, P.; HUTTER, F. Fast bayesian optimization of machine learning hyperparameters on large datasets. In: PMLR. **Artificial Intelligence and Statistics**. Fort Lauderdale, FL, USA, 2017. p. 528–536.

KOTTHOFF, L.; THORNTON, C.; HOOS, H. H.; HUTTER, F.; LEYTON-BROWN, K. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. **The Journal of Machine Learning Research**, JMLR. org, v. 18, n. 1, p. 826–830, 2017.

KUSHNER, H. J. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. **Journal of Basic Engineering**, American Society of Mechanical Engineers, v. 86, n. 1, p. 97–106, 1964.

LYU, W.; YANG, F.; YAN, C.; ZHOU, D.; ZENG, X. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In: PMLR. **International Conference on Machine Learning**. Stockholm, Sweden, 2018. p. 3312–3320.

MCKAY, M. D.; BECKMAN, R. J.; CONOVER, W. J. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. **Technometrics**, Taylor & Francis, v. 21, n. 2, p. 239–245, 1979.

MESQUITA, D. P.; GOMES, J. P.; CORONA, F.; JUNIOR, A. H. S.; NOBRE, J. S. Gaussian kernels for incomplete data. **Applied Soft Computing**, Elsevier, v. 77, p. 356–365, 2019.

MESQUITA, D. P.; GOMES, J. P.; JUNIOR], A. H. S.; NOBRE, J. S. Euclidean distance estimation in incomplete datasets. **Neurocomputing**, v. 248, p. 11 – 18, 2017. Available at http://www.sciencedirect.com/science/article/pii/S0925231217304320. Accessed on September 19^{th} 2020. ISSN 0925-2312. Neural Networks : Learning Algorithms and Classification Systems.

MOCKUS, J.; MOCKUS, L. Bayesian approach to global optimization and application to multiobjective and constrained problems. **Journal of Optimization Theory and Applications**, Springer, v. 70, n. 1, p. 157–172, 1991.

MOCKUS, J.; TIESIS, V.; ZILINSKAS, A. The application of bayesian methods for seeking the extremum. **Towards global optimization**, v. 2, n. 117-129, p. 2, 1978.

PALMA, A. D.; MENDLER-DÜNNER, C.; PARNELL, T.; ANGHEL, A.; POZIDIS, H. Sampling acquisition functions for batch Bayesian optimization. **arXiv preprint arXiv:1903.09434**, 2019.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

RASMUSSEN, C.; WILLIAMS, C. **Gaussian Processes for Machine Learning**. 1. ed. Cambridge, MA, USA: MIT Press, 2006.

RUSSO, D. J.; ROY, B. V.; KAZEROUNI, A.; OSBAND, I.; WEN, Z. *et al.* A tutorial on Thompson sampling. **Foundations and Trends® in Machine Learning**, Now Publishers, Inc., v. 11, n. 1, p. 1–96, 2018.

SHAHRIARI, B.; SWERSKY, K.; WANG, Z.; ADAMS, R. P.; FREITAS, N. D. Taking the human out of the loop: A review of bayesian optimization. **Proceedings of the IEEE**, IEEE, v. 104, n. 1, p. 148–175, 2015.

SHAHRIARI, B.; WANG, Z.; HOFFMAN, M. W.; BOUCHARD-CÔTÉ, A.; FREITAS, N. de. An entropy search portfolio for bayesian optimization. **Proc. NIPS Workshop Bayesian Optim.**, 2014.

SHEN, W.; WANG, J.; JIANG, Y.-G.; ZHA, H. Portfolio choices with orthogonal bandit learning. In: **Twenty-Fourth International Joint Conference on Artificial Intelligence**. Buenos Aires, Argentina: AAAI Press, 2015.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical Bayesian optimization of machine learning algorithms. In: **Advances in Neural Information Processing Systems 25 (NIPS)**. Lake Tahoe, Nevada, USA: NIPS Foundation, 2012. p. 2951–2959.

SOUZA, D.; VASCONCELOS, T.; MATTOS, C.; GOMES, J. Evaluation of data based normal behavior models for fault detection in wind turbines. In: IEEE. **2019 8th Brazilian Conference on Intelligent Systems (BRACIS)**. Salvador, Brazil, 2019. p. 878–883.

SRINIVAS, N.; KRAUSE, A.; KAKADE, S.; SEEGER, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In: OMNIPRESS. **Proceedings of the 27th International Conference on Machine Learning**. Haifa, Israel, 2010.

The GPyOpt authors. **GPyOpt: A Bayesian Optimization framework in python**. 2016. Available at http://github.com/SheffieldML/GPyOpt. Accessed on September 19^{th} 2020.

THOMPSON, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. **Biometrika**, JSTOR, v. 25, n. 3/4, p. 285–294, 1933.

TORUN, H. M.; SWAMINATHAN, M.; DAVIS, A. K.; BELLAREDJ, M. L. F. A global bayesian optimization algorithm and its application to integrated system design. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 26, n. 4, p. 792–802, 2018.

VASCONCELOS, T. d. P.; SOUZA, D. A. R. M. A. de; MATTOS, C. L. C.; GOMES, J. P. P. No-PASt-BO: Normalized portfolio allocation strategy for Bayesian optimization. In: IEEE. **International Conference on Tools with Artificial Intelligence (ICTAI)**. Portland, OR, USA, 2019. p. 561–568.