UAM
Universidad Autónoma de Madrid

Biblos-e Archivo
Repositorio Institucional UAM

Repositorio Institucional de la Universidad Autónoma de Madrid
https://repositorio.uam.es

Esta es la **versión de autor** del artículo publicado en:
This is an **author produced version** of a paper published in:

Neurocomputing 380 (2020): 20-35

DOI: https://doi.org/10.1016/j.neucom.2019.11.004

Copyright: © 2019. This manuscript version is made available under the CC-BY-NC-ND 4.0 licence http://creativecommons.org/licenses/by-nc-nd/4.0/

El acceso a la versión del editor puede requerir la suscripción del recurso
Access to the published version may require subscription

Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes

Eduardo C. Garrido-Merchán

Computer Science Department
Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11, Madrid, Spain

Daniel Hernández-Lobato

Computer Science Department
Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11, Madrid, Spain

Declarations of interest: None.

Email addresses: eduardo.garrido@uam.es (Eduardo C. Garrido-Merchán), daniel.hernandez@uam.es (Daniel Hernández-Lobato)

URL: http://arantxa.ii.uam.es/~egarrido/ (Eduardo C. Garrido-Merchán), https://dhnzl.org/ (Daniel Hernández-Lobato)

Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes

Eduardo C. Garrido-Merchán

Computer Science Department
Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11, Madrid, Spain

Daniel Hernández-Lobato

Computer Science Department
Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11, Madrid, Spain

Abstract

Some optimization problems are characterized by an objective that is very expensive, that lacks an analytical expression, and whose evaluations can be contaminated by noise. Bayesian Optimization (BO) methods can be used to solve these problems efficiently. BO relies on a probabilistic model of the objective, which is typically a Gaussian process (GP). This model is used to compute an acquisition function that estimates the expected utility (for solving the optimization problem) of evaluating the objective at each potential new point. A problem with GPs is, however, that they assume real-valued input variables and cannot easily deal with categorical or integer-valued values. Common methods to account for these variables, before evaluating the

Email addresses: eduardo.garrido@uam.es (Eduardo C. Garrido-Merchán), daniel.hernandez@uam.es (Daniel Hernández-Lobato)

URL: http://arantxa.ii.uam.es/~egarrido/ (Eduardo C. Garrido-Merchán), https://dhnzl.org/ (Daniel Hernández-Lobato)

objective, include assuming they are real and then using a one-hot encoding, for categorical variables, or rounding to the closest integer, for integer-valued variables. We show that this leads to suboptimal results and introduce a novel approach to tackle categorical or integer-valued input variables within the context of BO with GPs. Several synthetic and real-world experiments support our hypotheses and show that our approach outperforms the results of standard BO using GPs on problems with categorical or integer-valued input variables.

Keywords: Parameter tuning, Bayesian optimization, Gaussian processes, Integer-valued variables, Categorical variables.

1. Introduction

On a daily basis, a plethora of optimization problems emerge. For many of these problems, there is no access to the analytical expression of the objective that needs to be optimized. Consider as an example optimizing the parameters of the control system of a robot to maximize the robot's locomotion speed [18]. Given specific values for these parameters, there is no analytical expression that returns an estimate of the corresponding speed. A practical experiment with the robot or a computer simulation will have to be carried out for this purpose. This also means that when the objective is evaluated, a significant amount of time can be consumed. Therefore, in practice, one can only evaluate a small number of times the objective. Importantly, evaluations with the same input parameters may lead to different results, which we refer to as noisy evaluations. Noisy evaluations can arise due to non-exact measures taken from the robot's experiment as a consequence of, *e.g.*, environmental conditions. An objective function with the characteristics described before is

referred to as a black-box function $f(\mathbf{x})$ [28]. Other problems that involve the optimization of black-box functions include tuning the hyper-parameters of machine learning systems automatically [30], optimizing the parameters of meta-heuristics such as genetic algorithms [4] or finding the optimal location for weather sensor placement [6].

Bayesian Optimization (BO) methods are very successful for optimizing black-box functions with the characteristics described above [19]. We now formally describe the general working principle of these methods. Consider a real-valued objective function $f(\mathbf{x})$ over some bounded domain $\mathcal{X}$. This function is assumed to lack an analytical expression. This means that we can not access its gradients. We also assume that the evaluation of the objective is very expensive and its evaluations may be contaminated with additive noise (*i.e.* we observe $y = f(\mathbf{x}) + \epsilon$, with ϵ some noise random variable, rather than directly observing $f(\mathbf{x})$). The aim of BO methods is to minimize the black-box function using the minimum number of evaluations possible. With this goal, they iteratively suggest, in an intelligent way, an input location at which the objective that is being optimized should be evaluated. For this, at each iteration $N = 1, 2, 3, \ldots$ of the BO method, a probabilistic model is fitted to the objective observations $\{y_i\}_{i=1}^{N}$. Typically, this model is a Gaussian Process (GP) [24]. Specifically, a GP outputs a predictive distribution about the potential values of the objective at each input location. This predictive distribution summarizes the uncertainty about the objective and is used to compute an acquisition function $\alpha(\cdot)$ on $\mathcal{X}$. The acquisition function measures, at every input location, the expected utility of evaluating the objective $f(\cdot)$ at that precise input location, with the goal of solving the optimization problem

in the smallest number of iterations. The next point $\mathbf{x}_{N+1}$ at which the objective $f(\cdot)$ is evaluated is simply chosen as the one that maximizes $\alpha(\cdot)$. After this new observation is collected, the sequence of operations described before is repeated iteratively. When the budget of total potential evaluations of the objective is reached, a solution of the optimization problem can be computed by optimizing the GP predictive distribution (typically the mean value), as a proxy of the actual objective.

The key feature responsible of the success of BO methods in solving the optimization problem is that evaluating the acquisition function $\alpha(\cdot)$ is inexpensive compared to evaluating the black-box objective $f(\cdot)$. This is a consequence of the one-to-one relation between the acquisition function and the predictive distribution of the GP for $f(\cdot)$. More precisely, the acquisition function only depends on the predictive distribution of the GP, and not on the actual objective $f(\cdot)$, which is assumed to be very expensive. This implies that $\alpha(\cdot)$ can be maximized cheaply. Therefore, BO methods mainly work by spending a bit of time thinking about where the black-box objective should be evaluated next with the goal of finding the optimum using as fewer evaluations as possible. If evaluating the actual objective $f(\cdot)$ is very expensive, this extra thinking time pays off and saves a large amount of computational time. BO hence is suited to scenarios where the cost of evaluating the black-box is very high. Furthermore, a GP can naturally consider additive noise in the target value to predict. Therefore, BO methods can easily address settings in which the objective evaluations are contaminated by noise.

An issue that arises when using GPs for BO is that these models assume that the input variables of the optimization problem are real-valued. In

problems in which the input variables take integer or categorical values, the BO methodology has to be modified by introducing extra approximations. The most common approach is to consider that these variables take real values. Then, in the case of integer-valued variables, one simply rounds to the closest integer the corresponding integer-valued variable after optimizing the acquisition function. In the case of categorical-valued variables, a popular approach is to use a one-hot encoding of the variable. This involves replacing the categorical variable with as many variables as categories. Then, after the acquisition function is optimized, the largest extra variable is set equal to one and all the others equal to zero. This is precisely the approach followed, *e.g.*, by the popular software for BO Spearmint (https://github.com/HIPS/Spearmint).

We show in this paper that the methods described for dealing with integer-valued and categorical variables may lead to a failure of the BO method. More precisely, when they are used, the point of the input space at which the evaluation is performed is different from the one that maximizes the acquisition function (*e.g.*, as a consequence of the rounding to the closest integer). This can lead to a behavior of the BO method in which the objective is always evaluated at the same input location, at each iteration. We also show in this paper that these issues can be easily solved by performing the rounding to the closest integer value (in the case of integer-valued variables) or the one-hot encoding transformation (in the case of categorical variables) inside the wrapper that evaluates the black-box objective. When this is the case, the point of the input space at which the evaluation is performed is the same as the one that maximizes the acquisition function. This avoids the

aforementioned problem. Nevertheless, this makes the objective constant in particular regions of the input space, *i.e.*, those leading to the same value once the one-hot encoding transformation has been computed, in the case of categorical variables, and those that lead to the same integer value, in the case of integer-valued variables. The GP will ignore this constant behavior of the objective and hence will result in a poor model, providing sub-optimal optimization results, in consequence. To overcome this, we introduce in this paper a transformation of the input variables that leads to an alternative covariance function for the GP. With this covariance function, the GP will correctly describe the objective as constant in particular regions of the input space, leading to a better modeling of the objective and, in consequence, to better optimization results.

Some practical optimization problems involving a mix between integer-valued, real and categorical variables include finding the optimal hyper-parameters of a machine learning system [30]. Specifically, in a deep neural network some parameters of interest to be adjusted may include the learning rate, the activation function and the number of layers. These two last variables can only take integer and categorical values, respectively. However, the learning rate can take real values. Similarly, in a gradient boosting ensemble of regression trees [5] we might want to tune the tree maximum depth and the learning rate. While the first variable can take integer values only, the second variable can take real values.

We have carried out several experiments to evaluate the proposed approach to deal with integer-valued and categorical variables in BO methods based on GPs. These experiments show such an approach leads to improved

optimization results over standard techniques and other alternatives for BO from the literature. Furthermore, although not directly investigated in this paper, the proposed approach can be potentially used in other non-standard BO scenarios. For example, those considering the optimization of several objectives under some constraints [7]; those involving the optimization of the learning curves of machine learning algorithms [33]; or those involving solving several optimization tasks simultaneously [32].

Finally, we would like to point out that in some optimization problems the black-box objective is relatively cheap to evaluate. In this setting, instead of using BO methods one can make use of meta-heuristics [9] like the classic genetic algorithms [8], particle swarm optimization [15] or simulated annealing [35] or new meta-heuristics that try to accelerate the process of finding a solution in fewer iterations. Examples of the last ones include those based in variable variance Gaussian distribution sampling [21]; a meta-heuristic inspired by human society's intelligent contests [25]; or those based on the invasive weed algorithm of optimization by quantum computing [26]. These meta-heuristics can be used in a scenario where the evaluation cost of the objective is higher than in the case of genetic algorithms, but less expensive than in the case of BO, where we may just have a budget of 20 to 200 evaluations at most. In summary, BO differs from the previous methods by the fact that it uses a probabilistic model to make inference about the potential values of the objective function. This model is used to make intelligent decisions with the goal of solving the optimization problem with a small number of evaluations. When the number of potential evaluations is small, and the evaluations are expensive, one is expected to obtain much

better results by using BO methods instead of these techniques [10].

We organize the rest of the paper as follows: First, section 2 gives a short introduction to BO and Gaussian processes. Section 3 describes how the proposed approach deals with integer-valued and categorical variables within the context of GP. Section 4 reviews related methods from the literature. Section 5 describes real-world and synthetic experiments, showing the advantages of the proposed approach over standard methods for BO with GP and other related techniques for BO. We conclude with Section 6 that summarizes the conclusions of the paper.

2. Background on Gaussian Processes and Bayesian Optimization

The BO methodology relies on fitting a probabilistic model to observations of the black-box objective that is being optimized. The predictive distribution of that model specifies the potential values of the objective at each point of the input space. By taking into account this predictive distribution, BO methods guide the search focusing only on those regions of the input space that are expected to deliver the most information about the solution of the optimization problem. Typically, the probabilistic model used for BO is a Gaussian Process (GP) [24]. The reason for this is the ability of GPs to easily compute a predictive distribution of the objective. Other potential models for BO include Random Forests, encoded in the Auto-WEKA tool [34], Student's-T processes [27] or deep neural networks [31]. In this paper, we will focus on BO using GPs, but the proposed methods could also be used in the case of Student's-T processes.

A GP is defined as a prior distribution over functions. When using a

GP as the underlying model, the assumption made is that the black-box objective function $f(\cdot)$ that is being optimized has been randomly sampled from the GP. That is, $f(\cdot) \sim \mathcal{GP}(0, k(\cdot,\cdot))$. This distribution is fully specified in terms of a covariance function $k(\mathbf{x}, \mathbf{x}')$ and a zero mean. The intrinsic features of the objective, $f(\cdot)$, such as smoothness, level of additive noise, amplitude, etc., are specified by the covariance function $k(\mathbf{x}, \mathbf{x}')$. The output of this function is simply the covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$. Namely, $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')]$. A common covariance function used in the context of BO is the Matérn function, in which the ν parameter is set equal to $5/2$ [30]. This covariance function is:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \sqrt{5}\frac{r}{\ell} + \frac{5}{3}\frac{r^2}{\ell^2}\right) \exp\left(-\sqrt{5}\frac{r}{\ell}\right) , \tag{1}$$

where we define r as the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$. Namely, $|\mathbf{x} - \mathbf{x}'|$. Note that $k(\cdot,\cdot)$ only depends on r, the Euclidean distance, assuming that the hyper-parameters σ^2 and ℓ are fixed. The covariance functions that share this property are called *radial basis functions* (RBFs). ℓ is simply the length-scale hyper-parameter, which specifies the level of smoothness of the functions generated from the GP. Most of the times a different length scale ℓ_j is used for each dimension j. On the other hand, σ^2 is the amplitude parameter, that specifies the range of variability (variance) of the GP samples. Finally, ν is a hyper-parameter related to the number of times that the GP samples can be differentiated. The larger the value of ν, the smoother the GP samples look. In the squared exponential covariance function case, $k(\cdot,\cdot)$ is defined by:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{r^2}{2\ell^2}\right) . \tag{2}$$

This covariance function is a limiting case of the Matérn covariance function when $\nu \to \infty$ [24]. Therefore, the squared exponential covariance function leads to a GP prior that favors smoother functions.

Consider that the black-box objective has been evaluated at N input locations so far. We define the corresponding observed data as $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$, with ϵ_i some additive Gaussian noise with variance σ_0^2. The GP model generates a predictive distribution for the potential values of $f(\cdot)$ at each input space location. This distribution is Gaussian and is characterized by a mean $\mu(\mathbf{x})$ and a variance $\sigma^2(\mathbf{x})$. Namely, $p(f(\mathbf{x}^\star)|\mathbf{y}) = \mathcal{N}(f(\mathbf{x}^\star)|m(\mathbf{x}^\star), \sigma^2(\mathbf{x}^\star))$, where the mean and variance are respectively given by:

$$\mu(\mathbf{x}) = \mathbf{k}_*^T (\mathbf{K} + \sigma_0^2 \mathbf{I})^{-1} \mathbf{y}\,, \tag{3}$$

$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_*^T (\mathbf{K} + \sigma_0^2 \mathbf{I})^{-1} \mathbf{k}_*\,. \tag{4}$$

In the previous expression $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ are the objective evaluations obtained so far; $\mathbf{k}_*$ are the prior covariances between $f(\mathbf{x})$ and each y_i; σ_0^2 is the variance of the additive Gaussian noise; $\mathbf{K}$ is a matrix with the prior covariances among each $f(\mathbf{x}_i)$, for $i = 1, \ldots, N$; finally, $k(\mathbf{x}, \mathbf{x})$ is the prior variance of $f(\cdot)$ at the candidate location $\mathbf{x}$. All these quantities are obtained by evaluating the covariance function $k(\cdot, \cdot)$ on the corresponding input values. See [24] for further details.

The BO method uses the previous predictive distribution to determine at which point $\mathbf{x}_{N+1}$ the objective function has to be evaluated next. Once this new observation has been collected, the GP is fitted again to the observed data, and the process is repeated iteratively. When the maximum number of evaluations of the objective has been reached (given by the available

computational budget), an estimate of the solution of the optimization problem can be simply obtained by optimizing the GP posterior mean for $f(\cdot)$ given in (3). Notwithstanding, as described previously, GPs have hyper-parameters that need to be adjusted during the fitting process. These include the variance of the additive Gaussian noise σ_0^2, and also other potential hyper-parameter of the covariance function $k(\cdot, \cdot)$. These can be, *e.g.*, the amplitude parameter σ^2 and the length-scales ℓ_j [24]. Rather than trying to find point estimates of these hyper-parameters, BO methods often compute an approximate posterior distribution for them using slice sampling [30]. This has been shown to give better empirical results, particularly at the first iterations of BO, in which the number of observations is very small and carrying out a point estimation of the hyper-parameters of the GP can lead to over-fitting. Given the samples from the GP hyper-parameters, the Gaussian predictive distribution defined in (3) and (4) is then simply averaged over the samples. This is the final predictive distribution of the GP probabilistic model.

The key to BO success is found in the acquisition function $\alpha(\cdot)$. For each input location, this function uses the predictive distribution given by the GP to compute the expected utility of performing an evaluation of the objective there. Therefore, the next input location where the objective is evaluated is simply chosen as $\mathbf{x}_{N+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$. Note that the acquisition function is not does not depend on $f(\cdot)$. It only depends on the predictive distribution given by the GP. Therefore, the maximization of $\alpha(\cdot)$ is very cheap. Algorithm 1 shows the detailed steps involving the BO of a black-box function.

A popular acquisition function is expected improvement (EI) [14]. EI is defined by the expected value of the utility function $u(f(\mathbf{x})) = \max(0, \nu - f(\mathbf{x}))$.

Input: Maximum number of evaluations T.

for $N = 1, 2, 3, \ldots, T$ **do**

1: if $N = 1$**:**

Choose $\mathbf{x}_N$ at random from $\mathcal{X}$.

else:

Find $\mathbf{x}_N$ by maximzing the acquisition: $\mathbf{x}_N = \arg\max_{\mathbf{x}} \quad \alpha(\mathbf{x})$.

2: Evaluate the black-box objective $f(\cdot)$ at $\mathbf{x}_N$: $y_N = f(\mathbf{x}_\mathrm{N}) + \epsilon_N$.

3: Include the point in training set: $\mathcal{D}_{1:N} = \mathcal{D}_{1:N-1} \bigcup \{\mathbf{x}_N, y_N\}$.

4: Fit again the Gaussian process model using $\mathcal{D}_{1:N}$.

end

Result: Optimize the mean of the Gaussian process to estimate the solution to the optimization problem.

Algorithm 1: Bayesian optimization of a black-box objective function.

This expectation is computed in terms of the GP predictive distribution for $f(\mathbf{x})$. Furthermore, $\nu = \min(\{y_i\}_{i=1}^N)$ is the best observation obtained so far, also called incumbent, assuming a minimization problem. In consequence, EI measures in expectation how much we will improve on the current best-found solution by evaluating the objective at a candidate point. The following expression gives the EI acquisition function in closed-form:

$$\alpha(\mathbf{x}) = \sigma(\mathbf{x})(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x}) + \phi(\gamma(\mathbf{x}))\,, \tag{5}$$

where

$$\gamma(\mathbf{x}) = (\nu - \mu(\mathbf{x}))/\sigma(\mathbf{x})\,, \tag{6}$$

and $\Phi(\cdot)$ and $\phi(\cdot)$ represent the cumulative density function (c.d.f.) and

probability density function (p.d.f.) of a standard Gaussian distribution, respectively.

Predictive Entropy Search (PES) is another commonly used acquisition function [11]. PES is an information-theoretic acquisition function. This acquisition function chooses the next input location $\mathbf{x}_{N+1}$, at which the black-box function $f(\cdot)$ has to be evaluated, as the point that gives maximum information about the global optimum $\mathbf{x}^\star$ of $f(\cdot)$. The information about this optimum is measured in terms of the differential entropy of the random variable $\mathbf{x}^\star$. This particular random variable is fully specified in terms of the corresponding posterior distribution $p(\mathbf{x}^\star|\mathcal{D}_N)$. PES selects $\mathbf{x}_{N+1}$ as the point that reduces the most the expected differential entropy of $\mathbf{x}^\star$, after performing an evaluation of the objective at that particular input location. Therefore, the PES acquisition function is:

$$\alpha(\mathbf{x}) = H[p(\mathbf{x}^\star|\mathcal{D}_N)] - \mathbb{E}_y[H[p(\mathbf{x}^\star|\mathcal{D}_N \cup (\mathbf{x}, y))]\,, \tag{7}$$

where the expectation w.r.t. y is computed using the predictive distribution of the GP at the candidate location $\mathbf{x}$..

A critical issue is, however, that computing (7) in closed-form is intractable. This is a consequence of the difficulty of evaluating the posterior distribution of the global optimum $p(\mathbf{x}^\star|\mathcal{D}_N)$. Therefore, in practice, (7) needs to be approximated. With this goal, in the literature it has been observed that (7) is just the mutual information between $\mathbf{x}^\star$ and y, $I(\mathbf{x}^\star; y)$ [11]. Note that the mutual information is a symmetric quantity. That is, $I(\mathbf{x}^\star; y) = I(y; \mathbf{x}^\star)$. Therefore, the roles of y and $\mathbf{x}^\star$ can be swapped in (7). This simplifies considerably the computation of the PES acquisition function, since now in (7) we will have to evaluate the entropy of y rather than the entropy of $\mathbf{x}^\star$. The

entropy of y can be estimated efficiently using expectation propagation (EP), as described in [11]. PES has been compared to other acquisition functions showing improved optimization results. In particular, it often shows a better trade-off between exploration (evaluating regions with high uncertainty) and exploitation (evaluating regions with low mean value) than EI.

3. Dealing with Categorical and Integer-valued Variables

In the framework described before it is assumed that the black-box $f(\cdot)$ has input variables whose values lie on the real line. This is a consequence of the covariance function of the GP, $k(\cdot, \cdot)$, which assumes that the input variables are real-valued. An issue may arise in the case that some, or all the input variables, take values in a closed discrete subset, such as, *e.g.*, the integers, or when some of the input variables are categorical. Concerning this last case, a commonly used technique is to use a one-hot encoding of the categorical variable. That is, the input dimension corresponding to that variable is replaced by additional variables, one variable per category. The only valid configurations are those in which one of the additional variables takes value one (*i.e.*, the additional variable corresponding to the active category), and all the remaining variables take value zero. For instance, suppose that an input dimension x_j is categorical, taking potential values from the set $\mathbb{C} = \{red, green, blue\}$. Dimension j is replaced in $\mathbf{x}$ with three extra variables that may take values values $(1, 0, 0), (0, 1, 0)$ and $(0, 0, 1)$, for each value in $\mathbb{C}$, respectively. In the case of integer-valued variables, a simple approach is to consider that they are real. If an integer value is strictly needed, then the real value can be rounded to the closest integer.

When some of the input variables of the black-box $f(\cdot)$ do not take real values, like integer or categorical values, some problems may arise in the BO method. In particular, a standard GP will ignore that only some input values are actually valid and will place probability mass on configurations at which the objective $f(\cdot)$ cannot be evaluated. These incorrect modeling assumptions about $f(\cdot)$ will have a negative impact on the optimization process. Furthermore, the optimization of the acquisition function $\alpha(\cdot)$ will result in candidate points $\mathbf{x}_{N+1}$ in which the categorical or integer-valued variables will be set to invalid values. In practice, some mechanism must be implemented to transform real values into integer or categorical values before the evaluation of the black-box can take place. Importantly, if this is not done with care, some problems may appear, as we explain in the next section.

3.1. Naive and Basic Approaches

As described before, if the problem of interest considers some categorical or integer-valued variables, we cannot evaluate $f(\cdot)$ at all the potential input locations. We can only do the evaluation at compatible input locations with the integer or categorical-valued variables. A naive approach to take this into account within the BO method is to (i) optimize $\alpha(\cdot)$ assuming that all variables lie in the real line, and (ii) use a one-hot encoding procedure to assign values to the categorical variables. For example, the extra input variable (associated to the categorical variable) with the largest value will be set to 1 and all other extra variables will be set to 0. Similarly, we can round all the integer-valued variables to the closest integer. As an example of the first case, consider that $\mathbb{Q}_k$ is the set of extra input dimensions of $\mathbf{x}$ that correspond to the k-th categorical input variable. For each $j \in \mathbb{Q}_k$, we

simply set $x_j = 1$ if $x_j > x_i \; \forall i \in \mathbb{Q}_k$ and $i \neq j$. Otherwise, we set $x_j = 0$. The approach described is precisely the one followed by the popular software for BO Spearmint (https://github.com/HIPS/Spearmint).

The first row of Figure 1 shows, for an integer-valued input variable, that the naive approach just described can lead to a mismatch between the point where the acquisition function has the highest value, and the point where the evaluation of the objective is performed. Importantly, this can produce situations in which the BO method always evaluates the black-box objective at a location that has already been evaluated before. This is precisely the case illustrated in the first row of Figure 1. Note that we have highlighted the maximum of the acquisition function, which is the recommendation for the next new evaluation of the black-box objective. At iteration 6, the BO method will choose the same point for evaluation as the one chosen at iteration 5. This happens simply because the next and following evaluations are performed at input locations that differ from the input locations at which the acquisition function is maximum. More precisely, because the evaluation is performed at a different location, it may not reduce at all the uncertainty at the point maximizing the acquisition function. Of course, in the case of categorical variables this mismatch between the point at which the black-box is evaluated and the maximizer of the acquisition function can be a problem as well. For this reason, we do not recommend using the approach described in practice.

The problem described in the previous paragraph can be easily solved in practice. In the case of integer-valued variables, one simply has to do rounding to the closest integer inside the wrapper that evaluates the black-box. A similar approach can be followed in the case of categorical variables. Namely,

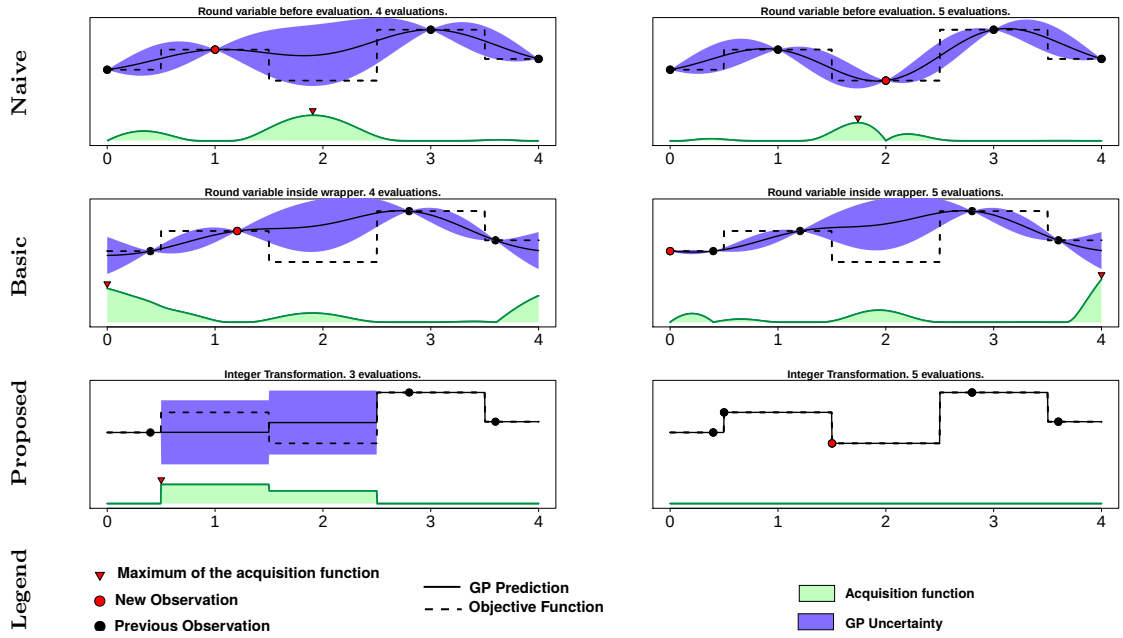

Figure 1: Different methods that deal with integer-valued variables. At the top of each image, we show a GP fit to the data (Posterior mean and 1-std confidence interval, in purple) that models a 1-dimensional objective taking values in the set $\{0, 1, 2, 3, 4\}$ and its generated acquisition function colored in green. Each column shows similar figures before and after evaluating a new point, respectively. Best seen in color.

perform the corresponding transformation inside the wrapper that evaluates the black-box. For this, (i) observe which extra input variable is the largest one, (ii) assign to that input variable value 1, and (iii) set all other extra variables equal to 0. This basic method is represented in the second row of Figure 1 for the case of an integer-valued variable. In this case, the location at which the acquisition function is maximum and the location at where the black-box objective is evaluated coincide. Therefore, the BO method will perform evaluations at different input locations each time, which is the

expected behavior. This avoids the problem of the naive approach mentioned before, where the BO method may get stuck. The problem now is, however, that the actual black-box objective becomes constant in those intervals of real values that, after being rounded to the closest integer, result in the same value. This is shown in the figures by displaying a flat objective. The GP model ignores this constant behavior which is expected to lead to sub-optimal optimization results. The same is expected to happen in the case of categorical input variables. The next section describes our proposed approach that solves the modeling problem described and leads to no uncertainty about the objective after performing just 2 evaluations.

3.2. Proposed Approach

We describe here a method that can be used to alleviate the problems of the basic approach introduced in Section 3. With this goal, we consider that the model should be constant in those regions of the input space that lead to the same input variable configuration on which the actual objective has to be evaluated. This is precisely how the actual objective behaves. This feature can be simply coded into the GP model by transforming the covariance function $k(\cdot, \cdot)$. Covariance functions are usually stationary, depending only on the distance between the input points [24]. If the distance between points is equal to zero, the correlation is maximum and the function values at both points are expected to be equal. Based on this, we suggest to transform the input points before computing the covariance function $k(\cdot, \cdot)$. The result is an alternative covariance function $k'(\cdot, \cdot)$:

$$k'(\mathbf{x}_i, \mathbf{x}_j) = k(T(\mathbf{x}_i), T(\mathbf{x}_j)) \,, \tag{8}$$

where $T(\mathbf{x})$ is a transformation in which all input variables of $f(\cdot)$ that are not real are modified as follows:

- The input variables that correspond to an integer-valued variable are simply rounded to the closest integer value.
- The input variables that correspond to the same categorical variable are assigned 0 value, unless they take the largest value among its corresponding group of extra variables. If they take the largest value, they are set equal to 1.

Note that $T(\cdot)$ performs the same transformation on an input location, $\mathbf{x}$, as the transformation described in Section 3.1 for the basic approach. In that case, however, the transformation was performed only inside the wrapper that evaluates the black-box objective. In this case, the transformation is performed in the covariance function of the GP model, which allows for a better modeling of the objective.

The beneficial properties of the proposed covariance function, $k'(\cdot, \cdot)$, when used for BO, are illustrated in the third row of Figure 1. Again, this is the case a single input variable that takes integer values. We can see that the GP model correctly identifies that the black-box objective is constant inside those intervals of real values that are rounded to the same integer value. Moreover, the level of uncertainty is just the same in those intervals, and this is reflected in the acquisition function. Furthermore, after doing a single measurement in every interval, the uncertainty about the objective function goes to zero. This better modeling of the objective is expected to be reflected in better optimization results. The same behavior is expected in the case of categorical

variables.

The transformation $T(\mathbf{x})$ rounds all integer-valued variables, that initially take values in $\mathbb{R}$, to the closest integer $k \in \mathbb{Z}$. The set of the integers, $\mathbb{Z}$, has a notion of order. That is, for all $z \in \mathbb{Z}$, we can define operators of order that involve two values: $<, >, \leq$ and $\geq$, such that $z_i < z_j$, $z_j > z_i$, $z_i \leq z_j$ and $z_j \geq z_i$, having that $z_i, z_j \in \mathbb{Z}$. The resulting transformation preserves this order. More precisely, assume an integer input variable and that $T(\mathbf{x})$ and $T(\mathbf{x}')$ only different in such integer-valued variable. The prior covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ under $k(T(\mathbf{x}), T(\mathbf{x}'))$ will be higher the closer the corresponding integer values of $T(\mathbf{x})$ and $T(\mathbf{x}')$ are the one from the other. Therefore, the GP model will be able to correctly capture the smoothness properties of the black-box objective $f(\cdot)$ when solving the optimization problem.

In the case of categorical-valued variables (*e.g.*, variables with values such as *red*, *green*, *blue*) the notion of order does not apply. That is, the operators $<, >, \geq$ and $\leq$ have no meaning nor purpose. One cannot compare two different values c_1, c_2 of any categorical set $\mathbb{C}$ according to these operators. However, in a categorical set, there exists a notion of equality or difference, specified by the operators $=, \neq$. The proposed transformation preserves this notion of no order and notion of equal or different. More precisely, assume one categorical variable and that $T(\mathbf{x})$ and $T(\mathbf{x}')$ simply differ in the extra variables corresponding to the categorical variable. The prior covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ under $k(T(\mathbf{x}), T(\mathbf{x}'))$ will be equal as long as $T(\mathbf{x})$ and $T(\mathbf{x}')$ encode a different value for the categorical variable. On the other hand, if $T(\mathbf{x})$ and $T(\mathbf{x}')$ encode the same value for the categorical variable,

the covariance will be maximum, which is the expected behavior, since the objective is will take the same value at both inputs.

It is straightforward to show that the proposed transformation generates a valid covariance function or kernel. In particular, a kernel is valid if we can find an embedding $\phi(\cdot)$ such that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}} \cdot \phi(\mathbf{x})$ [29]. Assume that the original kernel is valid and hence $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}} \cdot \phi(\mathbf{x})$ for some embedding $\phi(\cdot)$. Then $k(T(\mathbf{x}), T(\mathbf{x}')) = \phi(T(\mathbf{x}))^{\mathrm{T}} \cdot \phi(T(\mathbf{x}))$, and the embedding of the resulting kernel is simply given by $\phi(T(\cdot))$.

One important feature of the proposed transformation is that it does not add any extra parameters to tune. It is simply a transformation performed over the input variables, before computing the covariance function of the GP. Furthermore, this transformation is independent of the covariance function used and is also independent of acquisition function. Therefore, it can be used with any stationary GP covariance function and any acquisition function.

The proposed approach incorporates a transformation that consists in rounding input variables to the closest integer, in the case of integer-valued variables. In the case of categorical variables, a one-hot encoding transformation is used. Therefore, its computational complexity is in $\mathcal{O}(N)$, where N is the number of observations. This is much smaller than the cost of computing the predictive distribution of the GP process, which requires the inversion of the covariance matrix, with a cost in $\mathcal{O}(N^3)$. The consequence is that the computational cost of the BO method is not expected to be affected by the proposed transformation. The dominant cost will still be the same.

3.2.1. Visualization of the Proposed Transformation

Figure 2 shows the modeling properties of the proposed transformation, in the case of a real and an integer-valued variable (8). The figure displays the mean and standard deviation of the GP posterior distribution conditioned to a set of observations when the proposed transformation is used to compute the covariance function (top). The results obtained are compared with a standard GP which does not use the proposed transformation (bottom). To generate these data, we have used samples from a GP with the covariance function in (8) and $k(\cdot, \cdot)$ the squared exponential covariance function [24]. The first dimension takes integer-valued values in $\{0, 1, 2, 3, 4\}$, while the second dimension takes real values. Therefore, the actual objective is constant in any interval of integer-valued values that are rounded to the same integer value, for the integer-valued dimension. Note that the constant behavior of the objective function is correctly captured by the posterior distribution, only when the proposed transformation is used in the computation of the covariance function (top). The predictive distribution of a standard GP (corresponding to the basic approach in Section 3) cannot capture this constant behavior (bottom).

Figure 3 shows how the proposed transformation applies in a scenario with a categorical variable that can take only two values, *e.g.*, *True* and *False*. When a one-hot encoding is used, these two values are represented as $(0, 1)$ and $(1, 0)$, respectively. Therefore, in the basic approach described before, this categorical variable is replaced by two real variables taking values in the range $[0, 1]$. Notwithstanding, any combination of values in which the first component is larger than the second will lead to the configuration $(1, 0)$. By

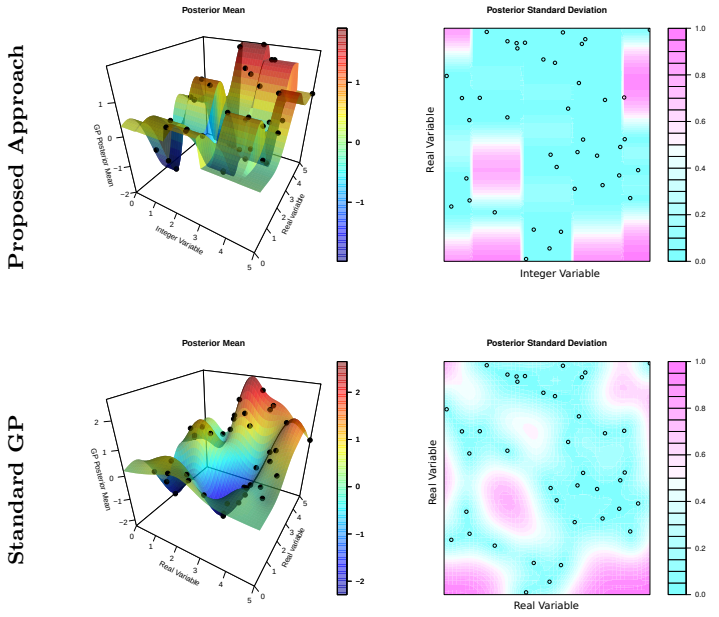

Figure 2: (top) Posterior mean and standard deviation of a GP over a 2-dimensional space in which the first dimension can only take 5 different integer values. The second dimension can take real values. The covariance function in (8) is used in this case. (bottom) Similar results for a standard GP model that does not use the proposed transformation in the covariance function. Best seen in color.

contrast, any combination of values in which the second component is larger will lead to the configuration $(0, 1)$. Therefore, the corresponding black-box objective is expected to be constant in those regions of the input space that correspond to the same configuration.

Figure 3 (top) shows that a GP with the proposed covariance function can capture the behavior described in the previous paragraph. This figure shows the posterior distribution of the GP. We observe that the uncertainty is equal to zero after just collecting one observation that corresponds to the

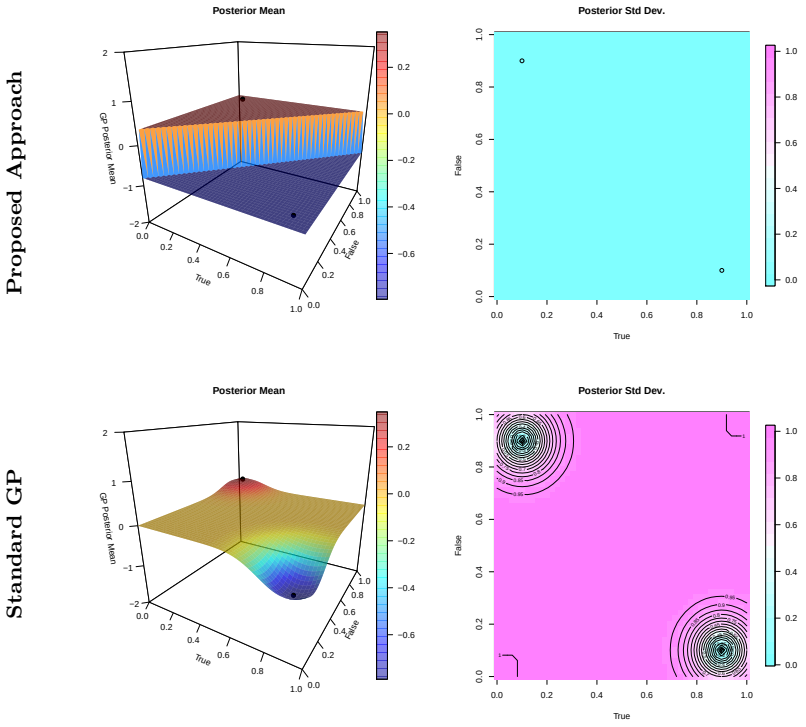

Figure 3: (top) Posterior mean and standard deviation of a GP, for a 1-dimensional binary variable, when the covariance function in (8) is used. (bottom) Same results without using the proposed transformation in the covariance function of the GP model. Best seen in color.

True value and just one observation that corresponds to the *False* value. This makes sense because the objective is expected to be constant in all the regions of the input space that lead to the same configuration, after the corresponding one-hot encoding. Figure 3 (bottom) shows that a standard GP is not able to capture this behavior, and the mean of the posterior distribution is not constant in those regions of the input space that lead to the same variable configuration. Furthermore, the standard deviation is significantly different

from 0, unlike in the proposed approach. Summing up, Figure 3 shows that a better modeling the black-box objective is done when the proposed covariance function is used in practice. In the end, this is expected to be translated into better optimization results by the BO method, as a consequence of using a better model of the objective.

3.3. Optimization of the Acquisition Function

A consequence of the transformation introduced in the previous section is that the acquisition function becomes flat in some regions of the input space. These are precisely the regions in which integer-valued and categorical variables specify the same configuration, after the doing the corresponding rounding to the closest integer, or the corresponding one-hot encoding, respectively. This behavior is illustrated in Figure 1 for one integer-valued variable. The typical approach to maximize the acquisition is to evaluate it first on a grid of points. Then, the best point from the grid is chosen to start a gradient-based search using, *e.g.*, L-BFGS. This is the approach employed by the BO software Spearmint. However, in the case of a non-smooth acquisition function, sub-optimal results may be obtained. Assume that we want to optimize a function with D binary categorical inputs, and let $D = 30$, for example. It is extremely unlikely that the best point after evaluating the acquisition function is the best among the 2^D choices. Furthermore, a gradient-based optimization of the acquisition function will not leave the starting point since the acquisition function will be flat at the starting point, as described before.

To address the problem of optimizing the acquisition function, we consider a block coordinate ascent methodology which iterates between optimizing

non-real variables (integer-valued and categorical) and real variables, similar to the described one-exchange neighborhood (OEN) strategy in [13, 17]. Our methodology consists in using the grid and the L-BFGS method to optimize all real variables. Then, the OEN strategy is used to optimize the transformed integer and categorical variables. The OEN strategy is a greedy method. Specifically, one iteratively evaluates, for each non-real dimension, the corresponding neighbors of that dimension (or dimensions in the case of categorical variables). If some improvement is achieved in terms of the acquisition function, after changing to one of the neighbors, that new value is kept as the best one. This process is repeated until there is no further progress. Of course, in order to evaluate the quality of each neighbor, the real variables have to be optimized. For that task, we use again L-BFGS.

At this point, one may ask whether the proposed transformation of the GP covariance function is beneficial at all. More precisely, it can be the case that it is simply enough to optimize the acquisition function as described here to obtain improved results. To answer this question, we have also implemented a block coordinate ascent optimization methodology that ignores the transformation of integer-valued and categorical variables in the GP covariance function. We compared in a toy problem the results obtained by both approaches, the proposed approach and the alternate optimization methodology alone, which we refer to as OEN optimization only.

Figure 4 shows the evaluations performed by the proposed approach and by OEN optimization only in a 2-dimensional optimization problem with one real variable and one integer-valued variable taking 5 different values. The contour curves show the value of the acquisition function. Results are

shown after performing 10, 20 and 30 evaluations. The points at which the objective has been evaluated are displayed as blue crosses. We observe that the proposed approach performs a more evenly spaced evaluation of the input space. By contrast, the OEN optimization only strategy, that does not use the proposed transformation, focuses on doing all the evaluations in a localized region of the input space. This is a consequence of using a model (*i.e.*, a GP without the proposed transformation) that ignores that the actual objective is constant in those regions of the input space leading to the same integer value. In any case, our experiments of Section 5 show that OEN optimization only often performs better than the basic approach described in Section 3, which simply uses a grid of points combined with L-BFGS to optimize the acquisition function with respect to all input variables, independently of whether they take real, integer or categorical values.

4. Related Work

We describe here two approaches that can be used as an alternative to BO methods based on GPs, when categorical and/or integer-valued variables are present in the black-box optimization problem. These methods are the Sequential model-based optimization for general algorithm configuration (SMAC) [12] and the Tree-structured Parzen Estimator Approach (TPE) [2]. Both naturally handle categorical and integer-valued variables. SMAC is implemented in the machine learning tool AutoWeka [34]. TPE is used in the HyperOpt tool [1]. Unlike the methods described so far for BO, SMAC and TPE do not rely on a GP as the underlying model of the objective function.

SMAC uses a random forest as the surrogate probabilistic model of the

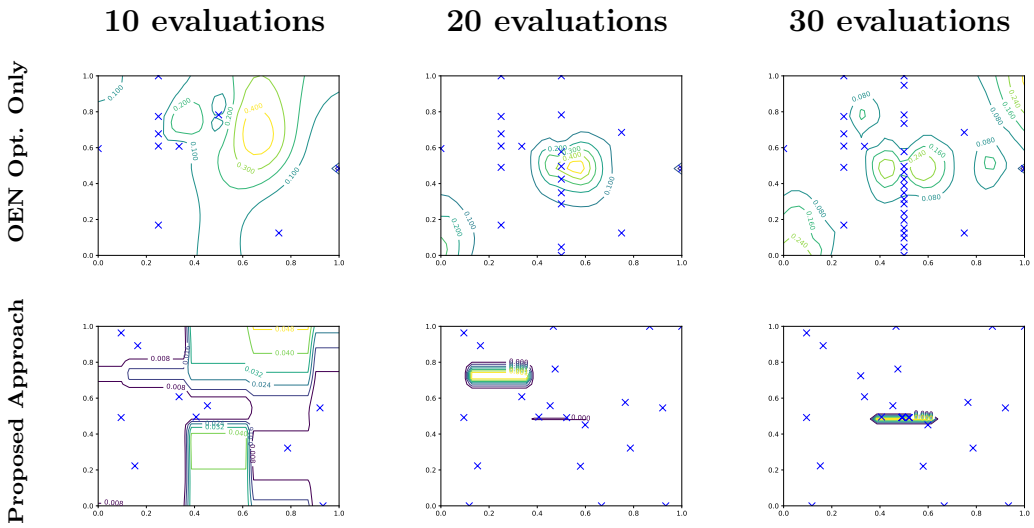

Figure 4: BO of a 2-dimensional objective with one integer-valued variable in $1, 2, 3, 4, 5$, and one real variable. The evaluations of the objective are shown by a cross symbol. We show results for OEN optimization only (top) and the proposed approach (bottom). The contour curves show the value of the acquisition function, as estimated by each method. From left to right, the value of the acquisition function after 10, 20 and 30 evaluations. Best seen in color.

objective function [3]. This model gives a predictive distribution which is used to select promising values of the parameters at which to evaluate the black-box objective. In a random forest, T random regression trees are generated. Each tree is grown using a different bootstrap sample of training data. Each bootstrap sample is obtained by drawing with replacement N instances from the observed data. Furthermore, in a random tree, at each node a random subset of variables are tested to split the data. This introduces variability in the generated trees. For a candidate test point, the random forest prediction is obtained by computing the individual predictions of each of the T trees of

the ensemble. The predictive distribution is simply a Gaussian distribution with the empirical mean and variance across individual tree predictions for that point. Given this predictive distribution, the EI criterion described in Section 2 is evaluated and maximized to select a new point at which the black-box $f(\cdot)$ should be evaluated. The random forest main advantage is that it has a smaller computational cost than a GP.

Random forest uses regression trees to compute the predictive distribution. Importantly, these trees can naturally consider categorical and integer-valued variables. Therefore, SMAC does not suffer from the limitations of GPs for handling these variables, as described in Section 3. A problem, nevertheless, is that the predictive distribution of random forest is expected to be suboptimal. In particular, it strongly relies on the randomness introduced by the bootstrap samples and the randomly chosen variables that have to be tested at each node to split the data. This result is confirmed by our experiments, in which GP-BO tends to outperform SMAC.

In SMAC, the EI criterion is optimized by a simple multi-start local search algorithm. This method considers the ten resulting configurations with locally maximal EI from previous runs and starts a local search at each of them. To manage a mixture of categorical and integer-valued variables, it employs a randomized one-exchange neighborhood search method, as described in Section 3.3. The search is stopped when none of the neighbors improves the EI criterion. The configuration with the highest EI value is chosen as the candidate on which to query the black-box function at the next iteration. More details on this method are given in [12].

TPE also uses EI as the acquisition function. However, its computation is

performed in a slightly different way, using a different modeling strategy. While standard BO methods fit a regression model for $p(y|\mathbf{x})$ directly, TPE follows a generative approach. In particular, $p(\mathbf{x}|y)$ and $p(y)$ are fit in this case. Both methods are related since $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$, where $p(\mathbf{x}) = \int p(\mathbf{x}|y)p(y)dy$. To compute an estimate of $p(\mathbf{x}|y)$, TPE assigns to each dimension a probability distribution which plays the role of a prior distribution. Then, TPE replaces those distributions with non-parametric density estimators. TPE redefines $p(\mathbf{x}|y)$ by using two different densities, $\ell(\mathbf{x})$ and $g(\mathbf{x})$. $\ell(\mathbf{x})$ is estimated using only the observations in which the objective value is lower than a chosen value $y^\star$. $g(\mathbf{x})$ is estimated using all other observations. That is,

$$p(\mathbf{x}|y) = \begin{cases} \ell(\mathbf{x}) & \text{if} \quad y \leq y^\star \,, \\ g(\mathbf{x}) & \text{if} \quad y > y^\star \,. \end{cases} \tag{9}$$

Importantly, non-parametric Parzen estimators are used to obtain $\ell(\mathbf{x})$ and $g(\mathbf{x})$ in the case of continuous random variables. In the case of categorical variables, a categorical distribution is used instead. Similarly, in the case of a variable over the integer set, a distribution that considers only this domain is used instead. Therefore, the particular characteristics of integer-valued and categorical input variables are naturally considered in TPE. $y^\star$ is simply set as some quantile of the objective values y observed so far. An interesting property of this approach is that no specific model for $p(y)$ is necessary. Specifically, TPE follows a different approach to evaluate the EI acquisition function. Namely,

$$\begin{aligned} \alpha(\mathbf{x}) &= \int_{-\infty}^{y^\star} (y^\star - y) p(y|\mathbf{x}) dy \\ &= \int_{-\infty}^{y^\star} (y^\star - y) \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} dy \propto (\gamma + \frac{g(\mathbf{x})}{\ell(\mathbf{x})}(1-\gamma))^{-1} \,, \end{aligned} \tag{10}$$

where we have used that $\gamma = p(y < y^\star)$ and that $p(\mathbf{x}) = \int p(\mathbf{x}|y)p(y)dy = \gamma l(\mathbf{x}) + (1-\gamma)g(\mathbf{x})$. See [2] for further details. Importantly, both models, $\ell(\mathbf{x})$ and $g(\mathbf{x})$, represent hierarchical processes that naturally allow for continuous and discrete-valued variables. Therefore, TPE can consider categorical and integer-valued variables.

The EI acquisition function of TPE is maximized simply by choosing points with high probability under $\ell(\mathbf{x})$ and low probability under $g(\mathbf{x})$. More precisely, in TPE, at each iteration, the evaluation is performed at the candidate point with greatest EI among many generated points sampled from $\ell(\mathbf{x})$ and evaluated according to (proportionally) $\ell(\mathbf{x})/g(\mathbf{x})$. The particular form of $\ell(\mathbf{x})$ makes it easy to draw candidates with a mix between discrete and continuous variables.

In the literature, there are other BO methods based on GP that can account for integer-valued and categorical input variables. For example, [23, 17] suggest constraining the optimization of the acquisition function to consider only those values that are valid. This is essentially equivalent to the method OEN optimization only described in Section 3.3. Again, we would like to highlight that such a method is expected to give sub-optimal results for the reasons explained in that section.

A related approach to our transformation that also allows to deal with categorical variables is the GP kernel proposed in [13]. In that work, it is used a weighted Hamming distance kernel to account for this type of variables in the context of GPs. That method is equivalent to our transformation when the squared exponential covariance function is used as the base covariance function in (8). However, as we transform the input variables before feeding

them into the covariance function, our approach is more general. Specifically, it has the advantage that any valid covariance function for GPs can be used. It is not restricted to the squared exponential covariance function. More over, [13] does not include any empirical evaluation of the benefits of considering the kernel described, nor it explains how to deal with integer-valued variables.

5. Experiments

We have carried out several synthetic and real-world experiments to evaluate the performance of the proposed approach for BO with GPs. We compare this method with (i) the basic method described in Section 3. We also compare results with (ii) the basic approach that uses the OEN procedure to optimize the acquisition function (without performing the transformation in the covariance function of the GP). We refer to such a method as OEN optimization only. Each method (Proposed, Basic and OEN optimization only) has been implemented in the BO software Spearmint. The code is available online at `https://github.com/EduardoGarrido90/Spearmint`. Finally, we also compare results with two other methods that do not use GPs as the surrogate model to guide the search. These methods are the ones described in the Section 4. Namely, (iii) SMAC, as implemented in AutoWeka, and (iv) TPE, as implemented in the HyperOpt platform.

For each optimization problem considered in this section we repeat the experiment 100 times and report average results and standard deviations. The standard deviation of the mean is estimated using 200 bootstrap samples. This is done to ensure that the comparison among the methods is fair. In each of the 100 repetitions we use a different random seed. Random seeds

are shared across methods. For the GP based methods (Proposed approach, Basic and OEN optimization only) we use a Matérn covariance function and slice sampling [20] to estimate the GP hyper-parameters (length-scales, level of noise and amplitude). Specifically, we generate 10 samples (50 samples in the real-world problems) of the hyper-parameters and average the acquisition function across the sample values, as in [30]. The acquisition function that we employ in these methods is PES.

We consider synthetic and real-world problems. In the last ones, the GP model need not be optimal. This is done to guarantee a fair comparison with other probabilistic surrogate models, as the random forests used by SMAC, or the Tree Parzen Estimator used by TPE. All GP based methods have been coded in Spearmint to ensure that the comparison is not influenced by other features of the BO process. For each method, at each iteration of the optimization process, we output a recommendation. This recommendation is obtained by optimizing the GP mean in the synthetic experiments. In the real-world experiments, we simply return the best observation obtained so far. Both SMAC and TPE deliver their recommendation based on the best observation in both real-world and synthetic problems.

The experiments contained in this section are organized as follows: The first set of experiments are synthetic and the objective is sampled from a GP prior. Then, we consider three real optimization problems to compare GP based methods with non-GP based methods in scenarios where the objective need not be sampled from a GP prior. These problems include finding an optimal ensemble of trees on the digits dataset and finding an optimal deep neural network on the digits and MNIST datasets.

5.1. Synthetic Experiments

We compare the five methods described before when the objective function is sampled from a GP prior. We consider problems with 4 and 6 dimensions. We also consider noiseless evaluations and evaluations contaminated with additive Gaussian noise. The variance of the additive Gaussian noise is set equal to 0.01 in the noisy scenario. In each setting, the objective is randomly sampled from the corresponding GP prior 100 times and we report average optimization results across the different objectives.

The first batch of experiments considers 4 input variables. The first 2 variables take real values and the rest of the variables take 3 and 4 different integer values, in the integer case. In the categorical case, the last two variables take 3 different categories. In the second batch of experiments, we consider 6 input variables. The first 3 variables take real values and the other 3 take 4, 3 and 2 different integer values, in the integer case. In the categorical case, they take 3 different categories. The next section considers real, categorical and integer-variables at the same time.

In each setting, we sample the objective from a GP prior using (8) as the covariance function. Furthermore, we run each BO method (Basic, Proposed, OEN optimization only, SMAC and TPE) for 50 iterations in the problems with 4 input variables and for 100 iterations in the problems with 6 input variables. For each method, we report, as a function of the evaluations done, the logarithm of the difference, in absolute value, between the actual objective value at the recommendation and the best objective value. In each of the 100 random repetitions of the experiments, we use a different random seed to generate the objective.

The average results of each method are displayed in Figure 5 and 6, for the 4 dimensional input setting, and for the integer and categorical case, respectively. We observe that the proposed approach provides better results than the other methods. In particular, it finds points that are closer to the optimum with fewer evaluations of the black-box objective, both in the case of integer-valued (noiseless and noisy) and categorical scenarios (noiseless and noisy). Figure 7 and 8 shows similar results for the 6 dimensional input setting, for the integer and categorical case, respectively.

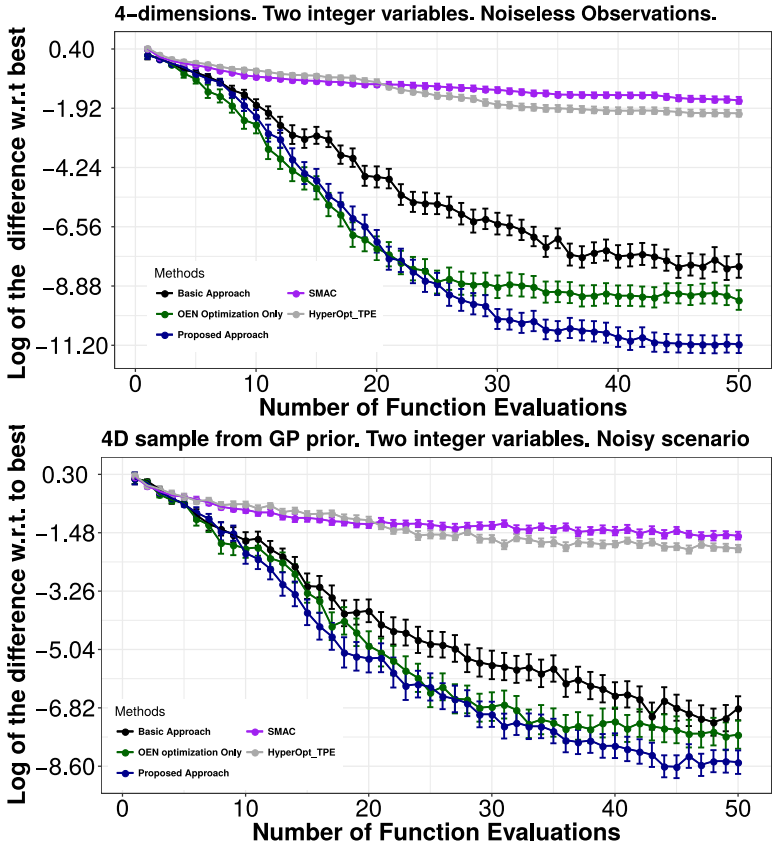

Figure 5: Average results on the synthetic experiments with 4 dimensions and 2 integer-valued variables.

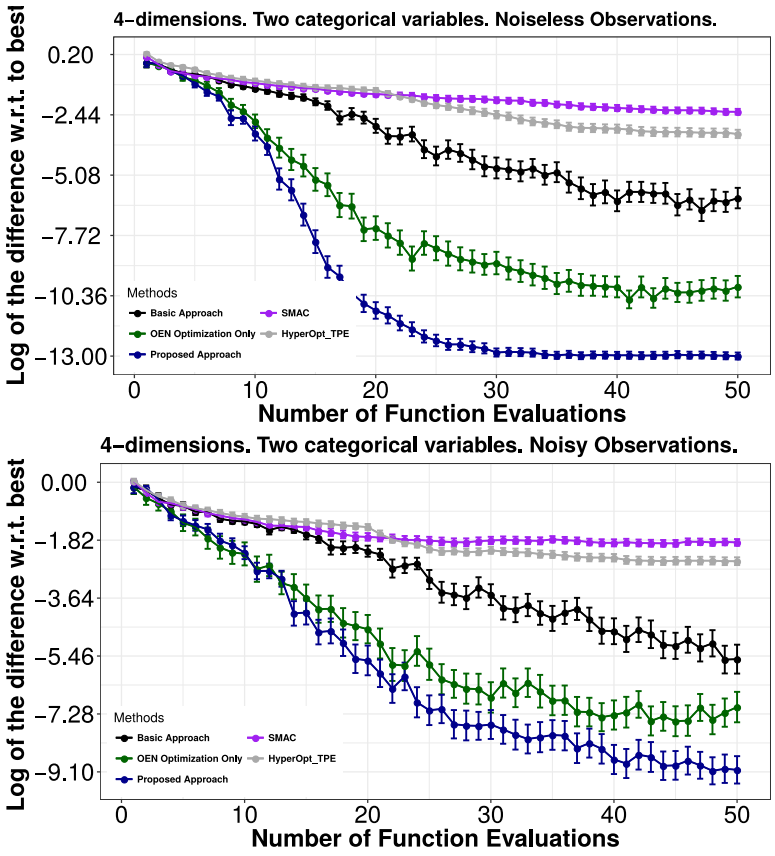

Figure 6: Average results on the synthetic experiments with 4 dimensions and 2 categorical variables

We observe that GP based BO outperforms the non-GP based BO, being the proposed approach better than the basic approach or the OEN optimization only method. This last method works better than the basic approach, showing that optimizing the acquisition function with the OEN procedure achieves better results. In the 6-dimensional scenario the difference in performance between the basic approach, OEN optimization only and the proposed approach is slightly higher than in the 4-dimensional scenario. TPE and

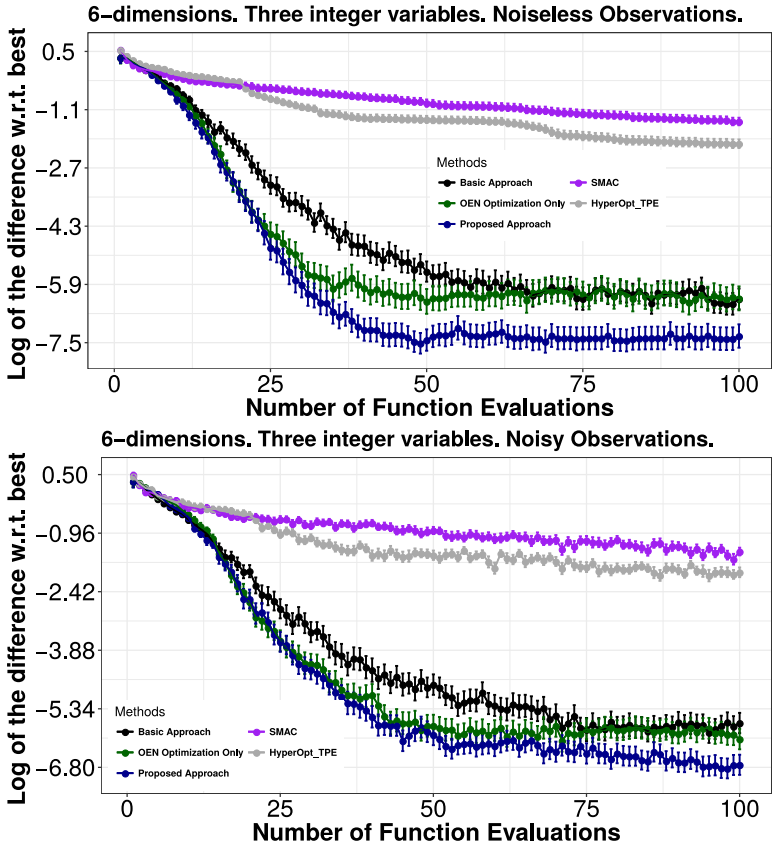

Figure 7: Average results on the synthetic experiments with 6 dimensions and 3 integer-valued variables.

SMAC also behave worse than the other methods in the 6-dimensional case. Finally, in the noisy setting, the methods are more equal but the proposed approach works slightly better. TPE and SMAC also deliver worse results in the noisy setting.

Note that TPE and SMAC do not assume a GP for the underlying model, which could be in a disadvantage in these experiments. Nevertheless, we think that it is still interesting to contrast results with them in this scenario in

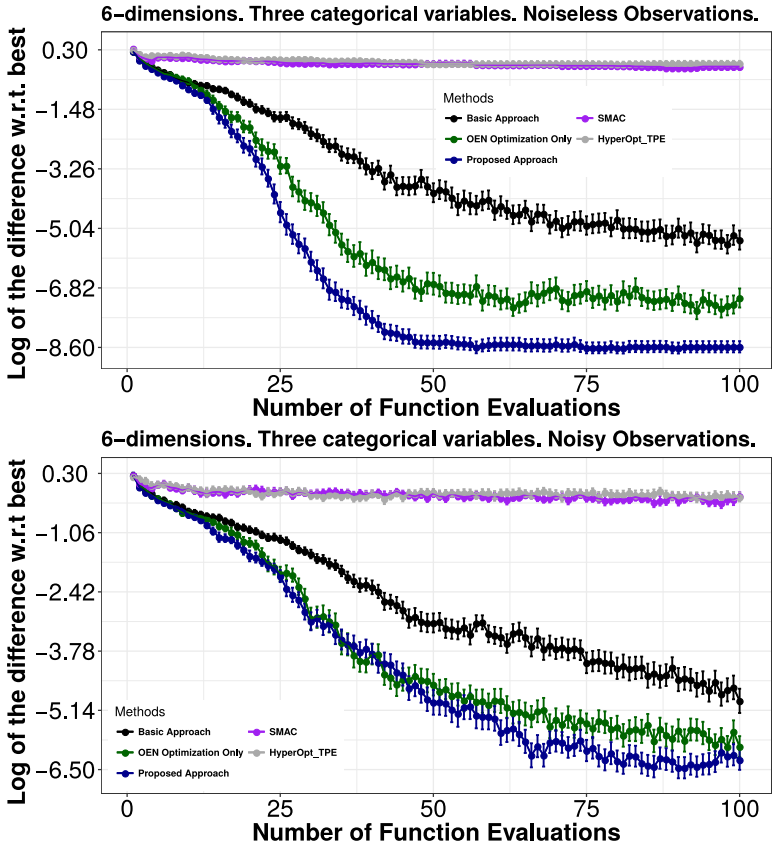

Figure 8: Average results on the synthetic experiments with 6 dimensions and 3 categorical variables.

which the exact solution of the optimization problem can be easily obtained and where the level of noise can be controlled. In Section 5.3, we perform experiments in which the objective function need not be sampled from a GP. The goal is to show the advantages of the proposed method in a wider range of scenarios.

5.2. Computational Time Analysis

In the problem involving 4 dimension we have also recorded the computational time required by each method (Proposed, Basic and OEN optimization only). The results obtained are displayed in Table 1. We can observe that the basic approach is the method that requires most average time per iteration. This is an unexpected result, since the proposed transformation should increase the cost of the method by at least a very small amount (recall that the cost of the proposed transformation is linear in N, the number of observations, while the fitting process of the GPs has a cost in $\mathcal{O}(N^3)$, which is the effective bottle-neck of BO).

The explanation for these results is found in the optimization of the acquisition function. More precisely, we have observed that in the case of the basic approach, the optimization of the acquisition function requires more steps of the the L-BFGS algorithm. In particular, Figure 1 shows that if we do not consider the transformation of the proposed approach, the shape of the acquisition function is more complex and hence more difficult to optimize. The proposed approach transforms the predictive distribution of the GP, and in consequence, also the acquisition function. The result is an acquisition that is easier to optimize using L-BFGS. A profile analysis of Spearmint shows that the optimization of the acquisition function takes 3 times less iterations, on average, when the proposed transformation is considered. The computational cost of optimizing the acquisition function represents a large fraction of the total execution time of BO methods. Therefore, in the end, the proposed approach takes the less computational time.

Finally, we observe that in the case of the OEN optimization only method,

the computational cost is also higher than in the cost of the proposed approach. The explanation for this is similar to the one described before. The shape of the acquisition function is more complex in the OEN optimization only method than in the proposed approach, as illustrated by Figure 4. Again, we have observed that optimizing the acquisition function in Spearmint using L-BFGS takes more time for OEN optimizing only than for the proposed approach.

Table 1: Average time per iteration in seconds of each method.

Basic Approach	**OEN Optimization Only**	**Proposed Approach**
26.9 ± 6.1	27.2 ± 8.5	14.6 ± 2.2

5.3. Hyper-parameter Tuning of Machine Learning Algorithms

We compare all methods on the practical problem of finding the optimal parameters of a gradient boosting ensemble [5] and a deep neural network (DNN) on the digits dataset. This dataset contains 1,797 data instances, 64 dimensions and 10 class labels. It has been extracted from the python package scikit-learn [22]. In the case of the DNN, we also consider the MNIST dataset [16]. This dataset has $60,000$ data instances, 768 dimensions and 10 class labels.

When finding an optimal gradient boosting ensemble on the digits dataset, the objective that is considered for optimization is the average test log-likelihood of the ensemble. This objective is estimated using a 10-fold cross-validation procedure. Importantly, model bias can be a problem for all approaches in this setting since the actual form of the objective is unknown. In this problem we consider a maximum of 200 evaluations. A table with

the parameters to be optimized, their range and their type is displayed in Table 2. These parameters include: The logarithm of the learning rate, the maximum depth of the generated trees and the minimum number of samples used to split a node in the tree building process. Note that while the first parameter takes real values, the other two take integer values only.

Table 2: Ensemble of trees names, types and range of their optimized parameters

Name	**Type**	**Range**
Log Learning Rate	Real	$[-10, 0]$
Maximum Tree Depth	Integer	$[1, 6]$
Minimum Number of Samples to Split	Integer	$[2, 6]$

To guarantee a fair comparison among the different methods, we consider 100 repetitions of the experiments. In each repetition we consider a different 10-fold cross-validation split of the data. Figure 9 shows the average results obtained across the 100 repetitions. The value displayed is the logarithm of the difference, in absolute value, between the test log-likelihood of the recommendation made and the best observed test-log likelihood, for that particular split. We observe that the proposed approach is able to find gradient boosting ensembles that perform better than those found by the basic approach, using a smaller number of evaluations of the objective. Furthermore, the proposed approach also performs better than SMAC, TPE, or the OEN optimization only method. This illustrates the benefits of using a better model of the objective, which in the end is translated into better optimization results.

In the task of finding an optimal deep neural network on the digits and

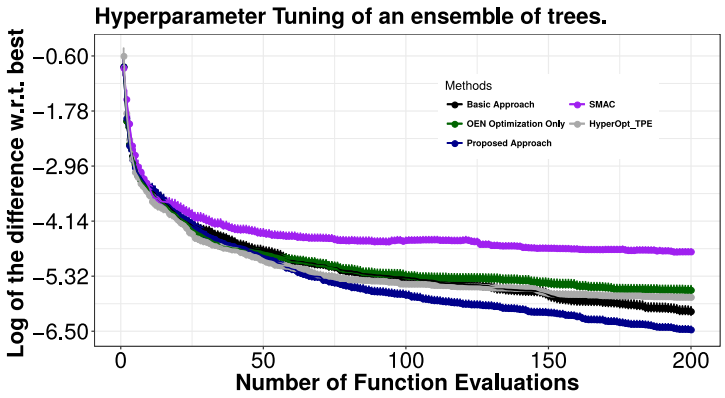

Figure 9: Average results on task of finding an optimal gradient boosting ensemble on the digits dataset.

MNIST dataset, the objective considered is the network test log-likelihood. This objective is estimated 10-fold cross-validation in the case of the Digits dataset. In the MNIST dataset, a validation set of $10,000$ instances, extracted from the training set, is used instead. We consider 125 and 150 evaluations of the objective for the digits and the MNIST dataset, respectively. The parameters that are optimized, their range and their type is displayed in Table 3. These parameters are: The logarithm of the learning rate, the number of hidden layers and the activation function. The first parameter is real-valued. The second and third parameters are integer-valued and categorical, respectively. The number of units in each layer of the deep neural network is set to 75.

The average results obtained in the two problems are displayed in Figure 10. The figure shows the logarithm of the difference, in absolute value, between the test log-likelihood of the recommendation made and the best observed test-log likelihood. We observe that again, the basic approach is

Table 3: Description, type and range of the deep neural network parameters optimized.

Name	**Type**	**Range**
Log Learning Rate	Real	$[-10, 0]$
Activation Function	Categorical	Linear, Sigmoid, Tanh or ReLU
Number of hidden layers	Integer	$[1, 3]$

significantly outperformed by the proposed approach. More precisely, the proposed approach can find parameter values that lead to a neural network with a better test log-likelihood on the left-out dataset, using a smaller number of evaluations of the objective. The proposed approach also outperforms SMAC and TPE on the digits dataset. However, on the MNIST dataset, the proposed approach gives similar results to those of TPE. We believe that better results of TPE on the MNIST dataset are explained as a consequence of model bias. Specifically, it can be the case that the GP model is not able to capture the properties of the objective. In any case, in both problems the proposed approach outperforms OEN optimization only and the basic approach.

5.4. Analysis of the Results and Limitations of the Proposed Approach

We provide in this section a brief summary and analysis of all the experimental results obtained so far. Similarly, we describe some of the limitations of the method proposed in this paper to account for integer-valued and categorical variables, in the context of BO with on GPs.

A first set of synthetic experiments has provided a controlled environment to evaluate the benefits of the proposed approach. In this setting, in which there is no model bias, the specific consideration of integer-valued and

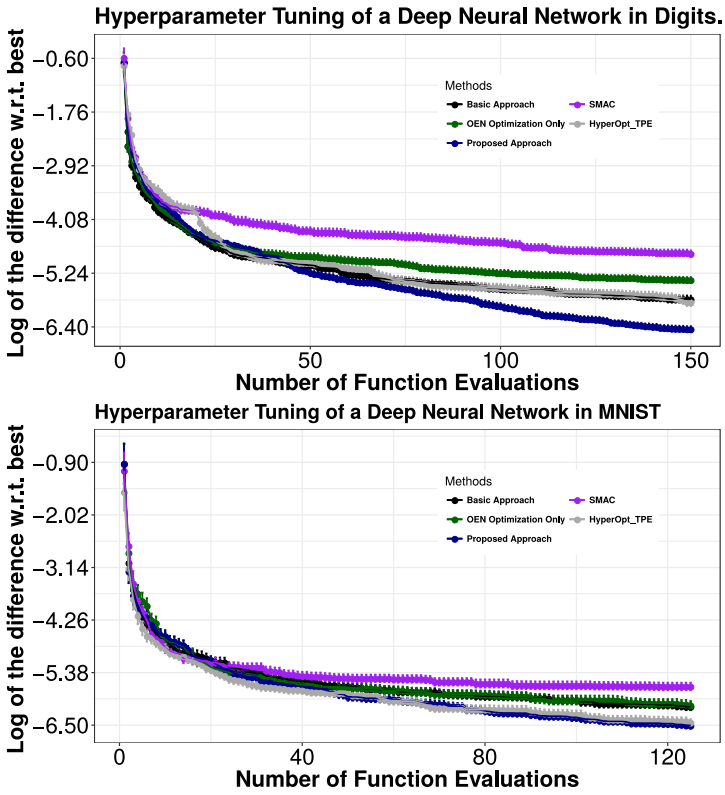

Figure 10: Average results on the Digits and MNIST dataset using deep neural networks.

categorical variables in the BO optimization process significantly improves the results obtained with respect to a basic approach that assumes that integer-valued and categorical variables take real values. In particular, with a smaller number of evaluations, the proposed approach is able to find better solutions of the optimization problem. These improvements have also been observed in a noisy evaluation setting, in which one does not obtain the value of the objective, but an estimate that is contaminated with noise. The proposed approach also provides better results than OEN optimization only,

a BO method that optimizes the acquisition function taking into account that only some configurations for the objective are valid. This highlights the importance of considering an adequate model of the objective, and shows that it is not enough to constrain the evaluation of the objective at valid configurations. In this setting, GP based BO optimization also outperforms other BO methods such as SMAC and TPE that can naturally consider integer-valued and categorical variables.

We have also analyzed the computational time required by each GP based BO method, obtaining unexpected results. In particular, besides providing better optimization results, the proposed approach also improves the average time per iteration of GP based BO methods. We believe that the reason for this is that a better modeling of the objective leads to a better acquisition function that is easier to optimize. More precisely, we have observed that when the proposed transformation is used, the L-BFGS method that is used to optimize the acquisition function needs less iterations to achieve convergence, resulting in faster decision times.

A second set of experiments considers real-world problems in which the hyper-parameters of several machine learning methods need to be adjusted. These experiments are free of any model bias since the actual objective is unknown. The results obtained in this setting also point out the beneficial properties of the proposed approach, which is able to find better configurations with a smaller number of evaluations than the basic approach or OEN optimization only. In some of these experiments, however, TPE seems to give similar results to those of the proposed approach. We hypothesized that this can be a consequence of GPs being a sub-optimal model of the objective in

this setting.

We now point out some limitations of the proposed approach. First, in the case of categorical variables, the proposed transformation adds as many extra variables as categories. This is expected to make the optimization process more difficult. Therefore, we recommend that the proposed transformation is used only with categorical variables taking a small number of categories. Second, if there is an integer-valued variable that takes values in a wide interval of integers (*e.g.*, between 1 and 5000) the proposed transformation is not expected to be very useful. The reason is simply that in such a setting there are not that many differences between an integer-valued variable and a real variable. So one should not expect significant benefits with respect to the basic approach for BO. Finally, if the number of categories or the range of integer values is very large, the optimization of the acquisition function is expected to be very expensive. More precisely, the One Exchange Neighborhood method will have to evaluate a large number of neighbors in this setting, which is expensive. Therefore, we only recommend the proposed approach when the number of categories or potential integer values is fairly small.

6. Conclusions

We have addressed in this paper the problem of using GP with integer-valued and categorical variables in the context of BO. Standard GPs assume real variables and cannot accurately account for this type of input variables. A naive method that is often used in the context of BO with GPs to account for categorical and integer-valued variables is to simply assume real variables.

Then, given a candidate point at which to evaluate the objective, a one-hot encoding or a rounding to the closes integer is done in the case of categorical or integer-valued variables, respectively. Notwithstanding, we have shown that under this approach the BO method may get stuck. More precisely, it will always evaluate the objective at the same candidate point at each iteration.

The problem described is a consequence of a mismatch between the regions of the input space that have high values of the acquisition function and the points at which the black-box objective is evaluated. A basic method to avoid this problem is to do the one-hot encoding or the rounding to the closest integer inside the wrapper that is used to evaluate the objective. This technique has the limitation that it makes the objective constant in those regions of the input space that lead to the same variable configuration. This constant behavior cannot be modeled by standard GPs.

In this paper we have modified the covariance function of the underlying GPs to account for those regions of the input space in which the objective should be constant. Our proposed method simply performs a transformation of the input variables that rounds integer-valued variables to the closest integer and that does a one-hot encoding of categorical variables. The consequence of this method is that the Euclidean distance between the points that lead to the same variable configuration becomes zero. This enforces maximum correlation between the function values at those input points under the GP, obtaining the expected constant behavior.

The proposed approach has been compared with the basic approach described before, and also with SMAC and TPE, that account naturally for

categorical and integer-valued variables. Several experiments illustrate that the proposed approach outperforms the basic approach and SMAC, and is most of the times better or at least equivalent to TPE. The proposed approach also performs better than a strategy that constrains the optimization of the acquisition function to evaluate the objective only at those points that are feasible (*i.e.*, the OEN optimization only method). We show that such a method turns out to be sub-optimal in practice. In particular, the GP model considered ignores that the objective is constant in those regions of the input space that lead to the same parameter configuration. We conclude that the proposed approach results in a better model of the objective, which is then translated into better optimization results by the BO method.

Acknowledgments

The authors gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at Universidad Autónoma de Madrid. The authors also acknowledge financial support from Spanish Plan Nacional I+D+i, grants TIN2016-76406-P and TEC2016-81900-REDT.

[1] J. Bergstra, D. Yamins, and D. D. Cox. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20, 2013.

[2] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[3] L. Breiman. Random Forests. *Machine learning*, pages 5–32, 2001.

[4] L. Cornejo-Bueno, E. C. Garrido-Merchán, D. Hernández-Lobato, and S. Salcedo-Sanz. Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. *Neurocomputing*, 275:818–828, 2018.

[5] J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of statistics*, pages 1189–1232, 2001.

[6] R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian Optimization for Sensor Set Selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219, 2010.

[7] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Predictive entropy search for multi-objective bayesian optimization with constraints. *Neurocomputing*, 2019.

[8] Mitsuo Gen and Lin Lin. Genetic algorithms. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–15, 2007.

[9] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

[10] J. M. Hernández-Lobato, M. A. Gelbart, B. Reagen, R. Adolf, D. Hernández-Lobato, P. N. Whatmough, D. Brooks, G.-Y. Wei, and R. P. Adams. Designing neural network hardware accelerators with decoupled objective evaluations. In *NIPS Workshop on Bayesian Optimization*, 2016.

[11] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *Advances in Neural Information Processing Systems*, pages 918–926, 2014.

[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523, 2011.

[13] Frank Hutter. *Automated configuration of algorithms for solving hard computational problems*. PhD thesis, University of British Columbia, 2009.

[14] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[15] James Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766, 2010.

[16] Y. LeCun. The MNIST Database of Handwritten Digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[17] Julien-Charles Lévesque, Audrey Durand, Christian Gagné, and Robert Sabourin. Bayesian optimization for conditional hyperparameter spaces. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 286–293. IEEE, 2017.

[18] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans. Automatic Gait Optimization with Gaussian Process Regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[19] J. Močkus. On Bayesian Methods for Seeking the Extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404, 1975.

[20] I. Murray and R. P. Adams. Slice Sampling Covariance Hyperparameters of Latent Gaussian models. In *Advances in Neural Information Processing Systems 23*, pages 1732–1740. 2010.

[21] Ali Namadchian, Mehdi Ramezani, and Navid Razmjooy. A new meta-heuristic algorithm for optimization based on variance reduction of guassian distribution. *Majlesi Journal of Electrical Engineering*, 10(4):49, 2016.

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

[23] T. Rainforth, T. A. Le, J. W. van de Meent, M. A. Osborne, and F. Wood. Bayesian Optimization for Probabilistic Programs. In *Advances in Neural Information Processing Systems*, pages 280–288, 2016.

[24] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[25] Navid Razmjooy, Mohsen Khalilpour, and Mehdi Ramezani. A new meta-heuristic optimization algorithm inspired by fifa world cup competitions: theory and its application in pid designing for avr system. *Journal of Control, Automation and Electrical Systems*, 27(4):419–440, 2016.

[26] Navid Razmjooy and Mehdi Ramezani. An improved quantum evolutionary algorithm based on invasive weed optimization. *Indian J. Sci. Res*, 4(2):413–422, 2014.

[27] A. Shah, A. Wilson, and Z. Ghahramani. Student-t Processes as Alternatives to Gaussian Processes. In *Artificial Intelligence and Statistics*, pages 877–885, 2014.

[28] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[29] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[30] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2171–2180. 2012.

[31] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable Bayesian Optimization Using Deep Neural Networks. In *International Conference on Machine Learning*, pages 2171–2180, 2015.

[32] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[33] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[34] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[35] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.