# Bayesian Optimization with Discrete Variables

Phuc Luong(✉), Sunil Gupta, Dang Nguyen, Santu Rana, and Svetha Venkatesh

Applied Artificial Intelligence Institute, Deakin University, Geelong, Australia
{pluong,sunil.gupta,d.nguyen,santu.rana,svetha.venkatesh}@deakin.edu.au

**Abstract.** Bayesian Optimization (BO) is an efficient method to optimize an expensive black-box function with continuous variables. However, in many cases, the function has only discrete variables as inputs, which cannot be optimized by traditional BO methods. A typical approach to optimize such functions assumes the objective function is on a continuous domain, then applies a normal BO method with a rounding of suggested continuous points to nearest discrete points at the end. This may cause BO to get stuck and repeat pre-existing observations. To overcome this problem, we propose a method (named **Discrete-BO**) that manipulates the exploration of an acquisition function and the length scale of a covariance function, which are two key components of a BO method, to prevent sampling a pre-existing observation. Our experiments on both synthetic and real-world applications show that the proposed method outperforms state-of-the-art baselines in terms of convergence rate. More importantly, we also show some theoretical analyses to prove the correctness of our method.

**Keywords:** Bayesian optimization · Gaussian process · Discrete variables · Hyper-parameter tuning

## 1 Introduction

Bayesian optimization [11,12] is an efficient approach to find a global optimizer of expensive black-box functions, i.e. the functions that are non-convex, expensive to evaluate, and do not have a closed-form to compute derivative information. For example, tuning hyper-parameters of a machine learning (ML) model can be considered as an expensive black-box function since it is time-consuming, and there is no explicit mathematical formula that maps the hyper-parameters to the accuracy of the model. Additionally, evaluating the goodness of a hyper-parameter set requires re-training of the model and assessment of the trained model on a validation set, which is expensive especially on large models like deep neural networks. BO can find the best set of hyper-parameters within a reasonable number of iterations because it makes use of observed data to predict a next point where the function should be evaluated. Generally, a BO method


J. Liu and J. Bailey (Eds.): AI 2019, LNAI 11919, pp. 473–484, 2019.
https://doi.org/10.1007/978-3-030-35288-2_38

has two main steps. First, it builds a surrogate model for the objective function and quantifies the epistemic uncertainty of the surrogate model using a Bayesian machine learning technique (e.g. using a Gaussian process). Second, it uses an acquisition function constructed from the surrogate model to decide where to sample the next function evaluation point. A summary of recent research work in BO and its applications in real-world problems can be found in [15].

When optimizing a function, most BO methods assume the input variables to be continuous because BO uses an acquisition function defined only on a continuous domain. In real-world applications, it is common to encounter problems with discrete variables, e.g. the number of trees and depth in tuning a random forest model, the number of layers/hidden units and the batch-size in tuning a neural network. Therefore, applying BO to optimize functions with discrete inputs is a challenging problem. First, when BO samples the next point for function evaluation, it suggests a continuous point that is an invalid input for the function. Second, we have exponentially many combinations of discrete values with respect to the number of variables, which causes the search space to become large and impractical to try all possible values.

**Existing Methods.** There have been a few prior attempts to develop methods to optimize expensive black-box functions defined on discrete inputs. One of them is the *Transformation* approach of Garrido-Merchán and Hernandez-Lobat [4]. This method is based on BO and assumes that the objective function does not change its values except at discrete points. Although this method can tackle the discrete inputs by simply rounding the inputs of a covariance function, it makes the acquisition function a step-wise function, which is difficult to optimize. Another work is *Sequential model-based optimization for general algorithm configuration* (SMAC) [5], which uses random forest as a surrogate model instead of a Gaussian process. It has a low computational cost and can naturally deal with discrete variables due to the tree-based structure. However, random forest is not a good choice for the surrogate model because it has a limitation in performing extrapolation [9]. Similarly, *Tree-parzen estimators* (TPE) [1] is another tree-based method which estimates the densities of good and bad candidate points in the search space and can cope with discrete variables by randomly sampling candidates from discrete distributions. Unfortunately, TPE requires a large enough number of observations, in the beginning, to model the density distribution efficiently. BO methods for multi-armed bandit [15] can tackle discrete domain. However, they do not incorporate correlation between neighbour discrete values, and they need to sample the same point again to reduce the uncertainty. Because of these disadvantages, the problem of black-box function optimization with discrete variables remains open.

When a standard BO is used to optimize functions with discrete variables, it treats discrete variables as continuous then applies a normal BO method, and finally rounds the suggested continuous point before function evaluations. We call this approach as *BO with naive rounding* (Naive BO). This approach does not have the problems of above-mentioned works e.g. step-wise difficult to optimize acquisition function, extrapolation problems of SMAC, or the density

modelling requirement of TPE. However, this approach often starts to repeat the function evaluations at previously tried points due to rounding of a continuous suggestion to the nearest discrete value in the search space. It is illustrated in Fig. 1, and, in this paper, we aim at solving this problem.

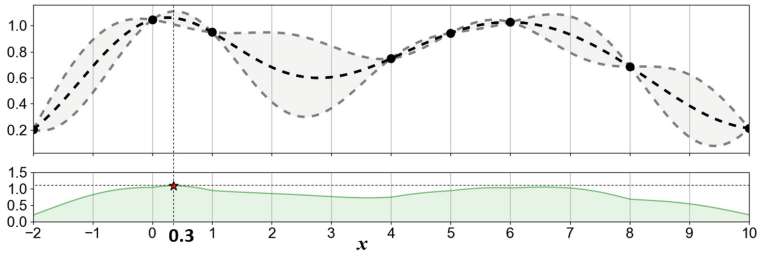



**Fig. 1.** An illustration of the repetition of pre-existing observations problem of standard BO caused by naive rounding. We can only evaluate at grid locations, and we already evaluated at $-2$, 0, 1, 4, 5, 6, 8, and 10 indicated by black dots. Using a Gaussian process, we can calculate the predictive mean (dashed line) and variance to build an acquisition function. Maximizing the acquisition suggests a continuous point (red marker) $x = 0.3$ that is invalid to evaluate, so we round to the nearest grid point $x = 0$ which is already observed. (Color figure online)

**Our Method.** We propose a novel approach, named **Discrete-BO**, to solve the repetition problem in the Naive BO method. In particular, we want the algorithm to sample points that are different from pre-existing observations by shifting the suggested point obtained after maximizing an acquisition function. There are two ways to shift the suggested point. One is to increase the value of the exploration factor of the acquisition function, and another one is to adjust the length scale of the covariance function. To select the optimal values for the exploration factor and length scale, we formalize it as an optimization problem. With these optimal exploration factor and length scale, our proposed method not only suggests valid discrete points but also avoids the repetition of observations.

Our contributions are:

- We propose **Discrete-BO** – a BO method for optimizing expensive black-box functions with discrete inputs.
- We provide theoretical analyses for a deeper understanding of our method.
- We conduct comprehensive experiments on both synthetic functions and real-world applications where our method outperforms state-of-the-art baselines.

## 2 Background

In this section, we briefly provide basic knowledge of BO with Gaussian process (GP) and the well-known upper confidence bound (GP-UCB) acquisition function since they form the basic framework of our proposed method. A detailed review of BO and acquisition functions can be found in [3,10].

**Bayesian Optimization with Gaussian Process.** BO is a well-known search strategy for finding the global optimizer of an expensive and noisy black-box function [12]. Specifically, BO finds the optimizer (i.e. the optimal input):

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}} f(x) \tag{1}$$

where $\mathcal{X}$ is a bounded domain in $\mathbb{R}^d$ and $f(x)$ is an objective function.

Normally, it is expensive to directly evaluate the objective function, thus BO builds a surrogate model that is cheaper to sample. One commonly used surrogate model is Gaussian process [13]. Typically, $f(x)$ is assumed to be a smooth function and modelled by a GP, i.e. $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$, where $\mu(x)$ and $k(x, x')$ are mean and covariance functions of the distribution. In the context of BO, mean can be assumed to be a zero function and the *squared exponential* kernel (Eq. (2)) is often used for covariance function.

$$k(x, x') = \sigma^2 exp(-\frac{1}{2l^2}\|x - x'\|^2) \tag{2}$$

where $\sigma^2$ is a parameter dictating the uncertainty in $f(x)$ and $l$ is a length scale parameter which controls how quickly a function can change.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ is our observations that contain $N$ inputs $x_i$ and their corresponding function values $y_i = f(x_i) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$. By fitting the observed data into the GP, we obtain the *predictive distribution* of $f(x)$ at any point $x$ in the search space. This predictive distribution is also a Gaussian distribution characterized by the mean and variance as follows:

$$\mu(x) = \mathbf{k}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{y}, \quad \sigma^2(x) = k(x, x) - \mathbf{k}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{k} \tag{3}$$

where $\mathbf{y} = (y_1, ..., y_N)$ is a vector of the function values we have so far, $k(x, x)$ is the covariance at point $x$, $\mathbf{k} = [k(x_i, x)]_{\forall x_i \in D}$ is the covariance between the new point $x$ and all other observed points $x_i$, $\mathbf{K} = [k(x_i, x_j)_{\forall x_i, x_j \in D}]$ is the covariance matrix, $\mathbf{I}$ is an identity matrix with the same dimension as $\mathbf{K}$, and $\sigma_\epsilon^2$ is the measurement noise.

The posterior mean and variance, calculated from Eq. (3), are used to define an acquisition function $\alpha(x)$, and various types of acquisition function are presented in [3,12]. In our method, we use the well-known GP-UCB because it was theoretically analyzed to have a bounded regret [16].

**Upper Confidence Bound Acquisition Function.** This acquisition function combines the posterior mean and variance from Eq. (3) as:

$$\alpha_t^{UCB}(x) = \mu(x) + \sqrt{\beta_t}\sigma(x) \tag{4}$$

where $\beta_t$ is the exploitation-exploration trade-off factor. The author in [16] recommends the value $\beta_t = 2\log\left(t^2 2\pi^2/3\delta\right) + 2d\log\left(t^2 dbr\sqrt{\log\left(4da/\delta\right)}\right)$. Using this $\beta_t$, BO algorithm with GP-UCB achieves an upper bound on the cumulative regret with the highest probability, greater or equal to $1-\delta$, in the search space, which is a subset of $[0, r]^d$ with $r > 0$, and $a, b > 0$ are constants [16].

The acquisition function in Eq. (4) qualifies every point $x$ in the domain so that we can sample the next point $x_{t+1}$ as follows:

$$x_{t+1} = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\ \alpha_t^{UCB}(x) \tag{5}$$

Acquisition functions differ in the way of balancing exploration and exploitation. The simplest one is the *Probability of Improvement* (PI) [8], which purely exploits the area near the incumbent [3], thus it might easily fall into a local optimum. An alternative is the *Expected Improvement* (EI) [6,7], which incorporates both exploitation and exploration leading to the global solution. Essentially, all acquisition functions mentioned above assume that the objective function $f(x)$ continuous so that the Eq. (5) returns a real-valued point.

## 3 The Proposed Framework

Given an expensive black-box function $f(x)$ with discrete inputs as illustrated in Fig. 2, our goal is to find the maximum of the function as

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\ f(x) \tag{6}$$

where $\mathcal{X}$ is **a finite discrete domain** in $\mathbb{R}^d$, and we only have noisy observations in the form $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$.

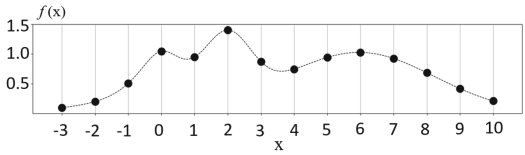



**Fig. 2.** An example of a black-box function with a discrete variable $x$. We do not know the form of this function, but we can evaluate it at discrete points in the set $\{-3, ..., 10\}$. Any other values of the variable $x$ are invalid inputs.

The optimization in Eq. (6) is particularly challenging because $f(x)$ is defined only on discrete points. However, since most of the real-world functions may still take correlated values on discrete points in a neighbourhood, one can fit a continuous function passing through these discrete points as shown in dotted line in Fig. 2. We can use a GP to model this continuous function and then use an acquisition function to suggest the next point for function evaluation. Since this point may not be part of the discrete input set, the function is evaluated at the closet point in the discrete set. But as discussed in Sect. 1 (see Fig. 1), this naive approach may repeatedly evaluate the function at the same point making the algorithm become stuck and causing inefficiency in the algorithm.

Before presenting a solution to this problem, we first provide a detail insight into what determines the next sample when using the GP-UCB acquisition function. In particular, since GP-UCB is a combination of $\mu_t(x)$ and $\sigma_t(x)$, its maximum value is determined strictly by one of the three scenarios.

*Case 1.* $\mu_t(x)$ dominates $\alpha_t(x)$: The maximizer of $\alpha_t(x)$ is determined completely by $\mu_t(x)$, and $\sigma_t(x)$ has no effect on the solution. If the naive rounding scheme experiences repetition of any previously evaluated function values, we can artificially increase the weighting $\beta_t$ to increase the effect of $\sigma_t(x)$ in determining the maximizer. Although this "artificial" increase of $\beta_t$ introduces some extra exploration in the optimization, it makes the algorithm going (without repetitions) while ensuring the convergence guarantee [16].

*Case 2.* $\sigma(x)$ dominates $\alpha_t(x)$: The maximizer of $\alpha_t(x)$ is determined completely by $\sigma_t(x)$, and $\mu_t(x)$ has no effect on the solution. Thus, the repetition is not possible because close to the existing observations, the $\sigma_t(x)$ will be small and will not achieve maximum.

*Case 3.* $\mu(x)$ and $\sigma(x)$ are balanced: Both $\mu(x)$ and $\sigma(x)$ have influence in determining the maximizer. In the event of any repetition, adjusting $\beta_t$ can make it $\sigma(x)$ dominated, which will stop repetitions. However, in this case, it may be possible to adjust the GP kernel length scale as an additional control to avoid repetitions as by increasing/decreasing length scale will allow us to not use excessively high values of $\beta_t$. We note that changing the length scale may cause slight misspecification of the GP prior, however, the convergence of the algorithm still remains guaranteed.

In three cases, the solution for avoiding the repetitions requires adjusting $\beta_t$ and/or the kernel length scale. Importantly, it needs to be set to reasonable values as their values directly affect the optimization efficiency. A large value of $\beta$ causes more exploration, making the algorithm less efficient. Similarly, the length scale $l$ controls the smoothness of a surrogate model, and a large misspecification leads to a requirement of more samples to get accurate function estimation.

**Proposed Method**

Based on the above observations and analyses, we propose our **Discrete-BO** algorithm. Our idea is to avoid sampling pre-existing observations by increasing the exploration-exploitation trade-off factor $\beta$ and adjusting the length scale $l$ of the covariance function. In the following we describe an optimal way to adjust the $\beta$ and the length scale $l$.

**Optimizing $\beta$ and $l$.** In our proposed algorithm, whenever a suggestion is repeated, we find new values of $\beta$ and $l$ so that the rounded maximizer of the acquisition function differs from the previously suggested rounded maximizer. Although we can adjust $\beta$ and $l$ using random search or grid search, it will be computationally expensive to find the appropriate combination of $\beta$ and $l$.

Therefore, we take a systematic approach to find the new values of $\beta$ and $l$ by solving an optimization problem as follows:

$$\begin{aligned} \beta^*, l^* = & \underset{\Delta\beta\in[0,\beta_h], l\in(0,l_h]}{\operatorname{argmin}} \; g(\beta_t + \Delta\beta, l) \\ g(\beta_t + \Delta\beta, l) = & \Delta\beta + \|x_{t+1} - x'_{t+1}\|_2 + P(x'_{t+1}) \end{aligned} \tag{7}$$

The $\Delta\beta$ is the increment applied on $\beta_t$ as $\beta_t \leftarrow \beta_t + \Delta\beta$. The $x_{t+1}$ is the point suggested by the original $\beta_t$ and $l_t$. The $x'_{t+1}$ is the point suggested by the $\beta_t + \Delta\beta$ and adjusted $l$. The term $P(x'_{t+1})$ is set to a constant $C$ if $round(x'_{t+1}) \in D_t$, otherwise it is zero. We can manually set the upper limits $\beta_h$ and $l_h$.

The optimization problem in Eq. (7) has *three objectives*. The *first* objective is to minimize $\Delta\beta$ since we do not want our new $\beta_t$ to exceed much more than the original $\beta_t$ to avoid inefficiency. We can not have negative $\Delta\beta$ as this will take away convergence guarantee of the algorithm [16]. The *second* objective is to minimize the distance between $x_{t+1}$ and $x'_{t+1}$ because the algorithm should suggest a discrete point that is close to the current potential area for exploitation. The *third* objective is to minimize a penalty, which is given to make sure pre-existing observation is not sampled again. The optimization problem in (7) is a continued non-convex problem. We solve it using L-BGFG with multiple random initializations. We summarize **Discrete-BO** in Algorithm 1.

**Algorithm 1. Discrete-BO** algorithm

**Input:** GP model, initial data $D_0 = \{(x_0, y_0)\}$, upper limits $\beta_h, l_h$
**for** $t = 0, ..., n$ **do**
  $\beta_t$ is calculated as suggested for GP-UCB, $l_t$ is estimated using $D_t$
  Select the next sample $x_{t+1} = \text{argmax}_{x\in\mathcal{X}}\ \alpha_t^{UCB}(x)$ with $\beta_t, l_t$
  $x_{t+1} = round(x_{t+1})$
  **if** $x \in D_t$ **then**
    Find the optimal $\beta^*$ and $l^*$ using (7): $\beta_t \leq \beta \leq \beta_h$ and $0 < l \leq l_h$
    $x_{t+1} = \underset{x\in\mathcal{X}}{\text{argmax}}\ \alpha_t^{UCB}(x)$ with $\beta^*, l^*$
    $x_{t+1} = round(x_{t+1})$
  Query the objective function to obtain $y_{t+1}$
  Augment $D_{t+1} = \{D_t, (x_{t+1}, y_{t+1})\}$ and update statistical model GP
**end**

## 4 Theoretical Analysis

In this section, we provide a theoretical understanding of our algorithm. We start by providing a definition of the repetition of pre-existing observations.

**Definition 1.** *In a discrete domain, a sample point is repeated if the following condition holds:* $x_{t+1} = \lfloor \text{argmax}_{x\in\mathbb{R}}\, \alpha_t(x) \rceil \in D_t$, *where* $D_t$ *consists of observations up to iteration* $t$.

The following two Lemmas provide an understanding of the two extreme cases detailed at the start of this section.

**Lemma 1.** *($\mu$-dominance) If the BO algorithm repeats an observation due to the dominance of $\mu_t(x)$, there exists an increased $\beta$ that will lead to a new solution $x'_{t+1}$ of the acquisition function such that $x'_{t+1} \notin D_t$.*

*Proof.* For $\mu$-dominance case, $\sigma_t(x) \ll \mu_t(x)\ \forall x \in \mathcal{X} \subseteq \mathbb{R}$, then $\max(\alpha) \simeq \max(\mu_t(x))$. Denoting the existing observations up to iteration $t$ as $D_t$, for any $x_{t+1} \in D_t$, let us assume that $x_{t+1}$ is the rounded value of a continuous value $x_a$. Also consider another continuous value $x_b$ such that the rounded value of $x_b$ is not in $D_t$. Then $||x_a - x_{t+1}|| < ||x_b - x_{t+1}||$ and $\sigma(x_a) < \sigma(x_b)$. By sufficiently increasing $\beta_t$, we make the effect of $\sigma(x)$ term dominates the $\mu(x)$ term (See $\alpha(x) \triangleq \mu(x) + \sqrt{\beta_t}\sigma(x)$) and thus $\alpha(x_b) > \alpha(x_a)$. Hence, we can always get a solution $x_b$. Finally, we can recommend the rounded value of $x_b$ to BO.

**Lemma 2.** *($\sigma$-dominance) If the BO algorithm repeats an observation in general, where none of $\mu(x)$ or $\sigma(x)$ dominates, then there exists an increased $\beta$ and an adjustment in length scale $l$ that will lead to a new solution $x'_{t+1}$ of the acquisition function such that $x'_{t+1} \notin D_t$.*

*Proof.* For the case where both $\mu_t(x)$ and $\sigma_t(x)$ jointly influence the maximizer of $\alpha_t(x)$, we can always sufficiently increase $\beta_t$ and tilt the balance such that the problem can become $\sigma_t(x)$ dominated and therefore, no repetition will occur. By adjusting length scale $l$, it is possible to use a smaller increase in $\beta_t$ and get the $\sigma_t(x)$ dominance as shown in the proof of Lemma 1.

## 5 Experiments

We conduct experiments to show the performance of our proposed method **Discrete-BO** on both synthetic and real-world applications. We compare our method with existing methods such as BO with naive rounding (Naive BO), Transformation method of Garrido-Merchán and Hernandez-Lobato [4], SMAC [5], and TPE [2]. These baselines are described in Sect. 1.

In our experiments, we use the *squared exponential* kernel and randomly initialize the optimization with $d+1$ points, where $d$ is the input dimension. The initialized points are kept identical across all methods for a fair comparison. We also use the same *budget* (i.e. the number of iterations), where at each iteration $t$ we report the best function value found so far by each method. We repeat each method 10 times and report the average result along with the standard error.

### 5.1 Synthetic Applications

We first illustrate our method on synthetic functions. Since most standard benchmark functions are continuous, we need to discretize them. For example, considering the simple 1-dimension Test function in Table 1, it is a continuous function

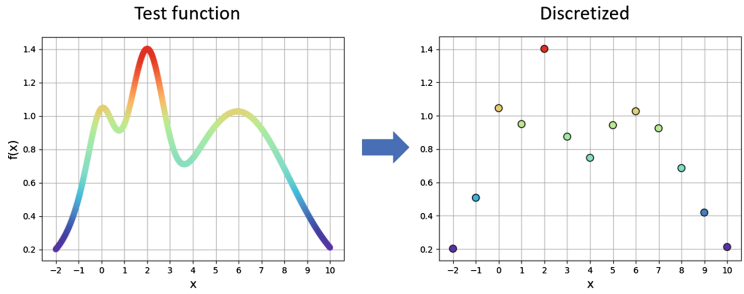



**Fig. 3.** A discretized version of the Test function in Table 1. It has one global maximum of 1.4 at $x = 2$.

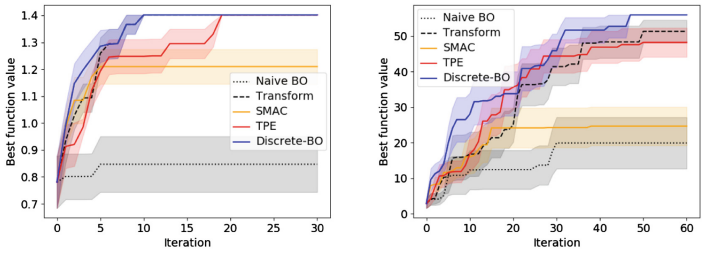
but is discretized, as shown in Fig. 3. Other functions in Table 1 such as Schubert, Eggholder, and Griewank are also discretized, and we multiply them with $-1$ so that we can find their maximum instead of minimum in their original form.

**Table 1.** Characteristics of discretized synthetic functions.

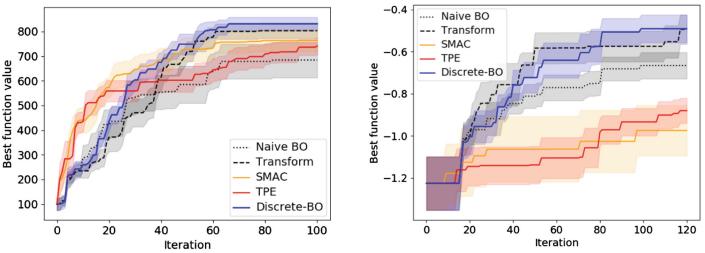
| Function | Formula | Dim | Range |
|---|---|---|---|
| Test function | $e^{-(x-2)^2} + e^{-\left(\frac{x-6}{10}\right)^2} + \frac{1}{e^{x^2}+1}$ | 1 | $x \in \{-2, \ldots, 10\}$ |
| Schubert | $\prod_{i=1}^{2}\left(\sum_{j=1}^{5} j\, cos((j+1)x_i + j)\right)$ | 2 | $x_1, x_2 \in \{-10, \ldots, 10\}$ |
| Eggholder | $-(x_2+47)\, sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right)$ $-x_1\, sin\left(\sqrt{\lvert x_1 - (x_2+47)\rvert}\right)$ | 2 | $x_1, x_2 \in \{-512, \ldots, 512\}$ |
| Griewank | $\sum_{i=1}^{3} \frac{x_i^2}{4000} - \prod_{i=1}^{3} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | 3 | $x_1, x_2, x_3 \in \{-50, \ldots, 600\}$ |

Figure 4 shows the optimization results for four functions in Table 1. From the results, we can see that our proposed method **Discrete-BO** is the best method where it significantly outperforms other methods. For example, consider the optimization result of the 1d Test function in Fig. 4(a). Our method and Transformation need only 10 iterations to find the true maximum function value of 1.4. In contrast, TPE requires a double number of iterations (20 iterations) to achieve the same result. Naive BO is unable to find this maximum due to its repetition problem as discussed in Sect. 1. SMAC is better than Naive BO.

When the number of dimensions is increased up to 2 and 3 (Fig. 4(b)–(d)), the optimization becomes a challenging problem due to the large search space. Our method still performs well and it clearly outperforms others. Transformation becomes the second-best method, which can be explained by the fact that the step-wise acquisition function created by Transformation is difficult to optimize on high dimension functions. Interestingly, Naive BO performs much better than TPE and SMAC on the 3d Griewank function.




(a) Test function (b) Schubert




(c) Eggholder (d) Griewank

**Fig. 4.** Results on synthetic functions – best function value (maximum) vs. iteration: (a) Test function, (b) Schubert, (c) Eggholder and (d) Griewank.

### 5.2 Real-World Applications

The second experiment shows the efficacy of our method in real-world applications: hyper-parameter tuning for a random forest (RF) and a neural network (NN).

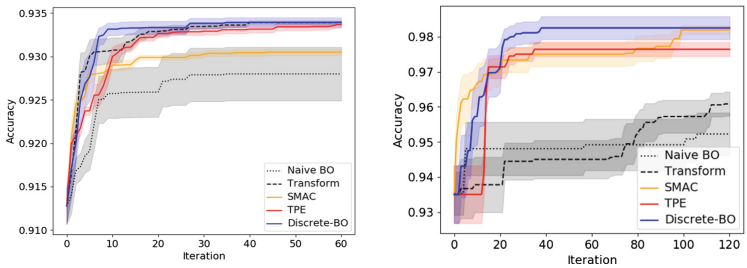
Our goal is to find the optimal set of hyper-parameters for two ML models, which achieves the best accuracy in classification. We define the black-box function as a mapping between the hyper-parameters and the classification accuracy on a held-out validation set. For RF, we used the Human Activity Recognition dataset [14], which consists of 10,299 records and 561 features. For NN, we build a network with one hidden layer and train with stochastic gradient descent. We test on the Handwritten Digits dataset [17], which has 1,797 $8 \times 8$ images and 10 labels. The discrete hyper-parameters to optimize are summarized in Table 2.

Figure 5(a) shows the result of hyper-parameter tuning for RF. From the result, we can see that our method **Discrete-BO** and Transformation are the best method. The performances of our method and Transformation are comparable although our method converges slightly faster than Transformation. Compared to SMAC and Naive BO, our method is significantly better. Naive BO is the worst method due to its repetition problem as explained in Fig. 1. TPE is the second-best method and it outperforms SMAC and Naive BO.

Figure 5(b) shows the result of hyper-parameter tuning for NN. Our method clearly outperforms all baselines, needing only 40 iterations to converge to the

**Table 2.** Hyper-parameters to optimize for Random Forest and Neural Network models. The last column indicates a dataset used to train and test the model.

| Model | Hyper-parameters | Dim | Dataset |
|---|---|---|---|
| Random Forest | Estimators $\in \{1, \ldots, 100\}$<br>Min samples in leaf $\in \{1, \ldots, 10\}$ | 2 | Human Activity |
| Neural Network | Hidden units $\in \{1, \ldots, 400\}$<br>Batch size $\in \{1, \ldots, 2000\}$<br>Max iterations $\in \{1, \ldots, 1000\}$ | 3 | Handwritten Digits |




(a) Random forest classifier (b) Neural network classifier

**Fig. 5.** Results of hyper-parameter tuning – best function value (accuracy) vs. iteration for two models: (a) RF and (b) NN.

optimum. Interestingly, Transformation does not perform well whereas SMAC becomes the second-best method; however, SMAC requires 100 iterations (2.5 times larger than ours). TPE is better than Transformation and Naive BO.

## 6 Conclusion and Future Work

This paper discusses the problem of BO for optimizing black-box functions with discrete variables. The naive rounding BO does not converge since it gets stuck at suggesting pre-existing observations. Our proposed method can improve the vanilla BO in a discrete domain and successfully solves the rounding problem of the vanilla BO without requiring a more complex kernel function. Our experimental results clearly demonstrate the effectiveness of our proposed method.

**Acknowledgements.** This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).

## References


1. Bergstra, J., Yamins, D., Cox, D.D.: Hyperopt: a python library for optimizing the hyperparameters of machine learning algorithms. In: Proceedings of the 12th Python in Science Conference, pp. 13–20. Citeseer (2013)
2. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)
3. Brochu, E., Cora, V.M., De Freitas, N.: 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)
4. Garrido-Merchán, E.C., Hernández-Lobato, D.: Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. arXiv preprint arXiv:1805.03463 (2018)
5. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
6. Jalali, A., Azimi, J., Fern, X., Zhang, R.: A lipschitz exploration-exploitation scheme for Bayesian optimization. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8188, pp. 210–224. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40988-2_14
7. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
8. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J. Basic Eng. **86**(1), 97–106 (1964)
9. Lakshminarayanan, B., Roy, D.M., Teh, Y.W.: Mondrian forests for large-scale regression when uncertainty matters. In: Artificial Intelligence and Statistics, pp. 1478–1487 (2016)
10. Lizotte, D.J.: Practical Bayesian optimization. University of Alberta (2008)
11. Mockus, J.: Application of Bayesian approach to numerical methods of global and stochastic optimization. J. Glob. Optim. **4**(4), 347–365 (1994)
12. Mockus, J., Tiesis, V., Zilinskas, A.: The application of Bayesian methods for seeking the extremum. Towards Glob. Optim. **2**(117–129), 2 (1978)
13. Rasmussen, C.E.: Gaussian processes in machine learning. In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds.) ML 2003. LNCS (LNAI), vol. 3176, pp. 63–71. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28650-9_4
14. Reyes-Ortiz, J.L., Anguita, D., Ghio, A., Parra, X.: Human activity recognition using smartphones data set. UCI Machine Learning Repository; University of California, Irvine, School of Information and Computer Sciences: Irvine, CA, USA (2012)
15. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016)
16. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.W.: Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. IEEE Trans. Inform. Theory **58**(5), 3250–3265 (2012)
17. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans. Syst. Man Cybern. **22**(3), 418–435 (1992)