# Bayesian Optimization for Categorical and Category-Specific Continuous Inputs

**Dang Nguyen, Sunil Gupta, Santu Rana, Alistair Shilton, Svetha Venkatesh**
Applied Artificial Intelligence Institute ($A^2I^2$), Deakin University, Geelong, Australia
{d.nguyen, sunil.gupta, santu.rana, alistair.shilton, svetha.venkatesh}@deakin.edu.au

## Abstract

Many real-world functions are defined over both categorical and *category-specific* continuous variables and thus cannot be optimized by traditional Bayesian optimization (BO) methods. To optimize such functions, we propose a new method that formulates the problem as a multi-armed bandit problem, wherein each category corresponds to an arm with its reward distribution centered around the optimum of the objective function in continuous variables. Our goal is to identify the best arm and the maximizer of the corresponding continuous function simultaneously. Our algorithm uses a Thompson sampling scheme that helps connecting both multi-arm bandit and BO in a unified framework. We extend our method to *batch* BO to allow parallel optimization when multiple resources are available. We theoretically analyze our method for convergence and prove sub-linear regret bounds. We perform a variety of experiments: optimization of several benchmark functions, hyper-parameter tuning of a neural network, and automatic selection of the best machine learning model along with its optimal hyper-parameters (a.k.a *automated machine learning*). Comparisons with other methods demonstrate the effectiveness of our proposed method.

## Introduction

Bayesian optimization (BO) (Shahriari et al. 2016) provides a powerful and efficient framework for global optimization of expensive black-box functions. Typically, at each iteration a BO method first models the black-box function via a statistical model (e.g. a Gaussian process (GP)) and then seeks out the next function evaluation points by maximizing an easy to optimize function (a.k.a. *acquisition function*) that balances the two conflicting requirements: exploitation of current function knowledge and exploration to gain more function knowledge. A notable strength of BO is that its convergence is well studied (Bull 2011).

Most BO methods assume that the function inputs are continuous variables. In reality, however, a function may be defined over diverse input types – for example, categorical, integer or continuous. Categorical type variables are particularly challenging since they do not have a natural ordering as in integer and continuous variables. Limited work has addressed incorporation of categorical input types. In a recent work, (Golovin et al. 2017) used *one-hot encoding* for categorical variables, increasing the input dimension by one extra variable per category. Since the categories are mutually exclusive, all the extra variables are set to zero except the *active* variable (corresponding to the required category) which is set to one. After converting the categorical variables to one-hot encoding, this approach treats extra variables as continuous in $[0, 1]$ and uses a typical BO algorithm to optimize them. Unfortunately, this type of encoding imposes equal measure of covariance between all category pairs, totally ignoring the fact that they may have different or no correlations at all. Further, the recommendations can get repeated as they are generated via rounding-off at the end. To address the latter problem, (Garrido-Merchán and Hernández-Lobato 2018) assumed that the objective function does not change its values except at the designated points of 0 and 1. This is achieved by using a kernel function that computes covariances after the input is rounded off. However, this makes the resulting acquisition function step-wise, which is hard to optimize.

In many optimization problems, an additional challenge arises – each category is coupled with a different continuous search space. For example, consider the problem of *automated machine learning* (Feurer et al. 2019) where we need to automatically select the best performing machine learning model along with its optimal hyper-parameters. Each machine learning model can be viewed as a distinct value (or choice) of a *categorical variable* while the hyper-parameters of the model can be viewed as category-specific *continuous variables*. None of the current GP-based BO methods can be applied to such complex search spaces.

To incorporate categorical inputs and deal with category-specific continuous search spaces, tree-based methods have been proposed. For example, SMAC tackles this problem by using random forest in place of GP (Hutter, Hoos, and Leyton-Brown 2011). However, random forest has a well-known limitation in performing extrapolation and thus is not a good choice for BO (Lakshminarayanan, Roy, and Teh 2016). Yet another method, tree-structured parzen window based approach (TPE) (Bergstra et al. 2011) can naturally cope with both categorical and continuous variables. In contrast to GP based approaches, TPE models two likelihood functions to assess if the function value at any point would be in the range of top few observations or not. The disadvantage is that this approach requires higher number of initial data points

to model the likelihood functions effectively. Moreover, the overarching problem with all the above-mentioned methods is that none of them offered an avenue for convergence analysis and thus remained ad-hoc in nature.

In this work, we develop a BO algorithm that can handle both categorical and continuous variables even if each category involves a different set of continuous variables. The algorithm is amenable to convergence analysis. We extend this algorithm to develop a *batch* BO algorithm for the same complex search space scenario. We use a mix of multi-armed bandit (MAB) and BO formulations in our approach. Each categorical value corresponds to an arm with its reward distribution centered around the optimum of the objective function in continuous variables. Thompson sampling (TS) (Russo and Roy 2014) is used for both selecting the best arm and suggesting the next continuous point to evaluate. Among various possibilities of MAB and BO algorithms, our choice of TS is guided by the need for an unified framework to join both the MAB and BO seamlessly into a single entity for convergence analysis. Empirically also TS is known to be a competitive algorithm for both MAB and BO problems due to using a range of exploitation/exploration trade-offs (Russo and Roy 2014; Kandasamy et al. 2018). We derive the theoretical regret bounds for both the sequential and batch algorithms, and show that the growth of cumulative regret is at most sublinear. We perform a variety of experiments: optimization of several benchmark functions, hyper-parameter tuning of a neural network, and *automated machine learning*. The empirical results demonstrate the effectiveness of our method.

Compared with other methods, our method offers three key advantages:

- Optimizing black-box functions with continuous and categorical inputs in both sequential and batch settings;
- Handling the problems where each category is coupled with a different set of continuous variables; and
- Deriving the regret bounds for both sequential and batch settings.

## Related Background

### Bayesian Optimization

Bayesian optimization (BO) is a method to find the global optimum of an expensive, black-box function $f(x)$ as $x^* = \text{argmax}_{x\in\mathcal{X}} f(x)$, where $\mathcal{X}$ is a bounded domain in $\mathbb{R}^d$. It assumes that $f(x)$ can only be noisily evaluated through queries to the black-box. BO operates sequentially, and the next function evaluation is guided by the previous observations. Let us assume that the observations up to iteration $t$ are denoted as $\mathcal{D}_t = \{x_i, y_i\}_{i=1}^t$, where $y_i = f(x_i) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Typically, $f(x)$ is assumed to be a smooth function and modeled using a Gaussian process (GP), *i.e.* $f(x) \sim \text{GP}(m(x), k(x, x'))$, where $m(x)$ is a mean function that can be assumed to be a zero function, and $k(x, x')$ is a covariance function modeling the covariance between any two function values $f(x)$ and $f(x')$. A common covariance function is the *squared exponential* kernel, defined as $k(x, x') = \sigma^2 \exp(-\frac{1}{2l^2}||x - x'||_2^2)$, where $\sigma^2$ is a parameter dictating the uncertainty in $f(x)$, $l$ is a length scale parameter. The *predictive distribution* for $f(x)$ at any point $x$ is also a Gaussian distribution with its mean and variance given by $\mu_t(x) = \boldsymbol{k}^\mathsf{T}[K + \sigma_\epsilon^2 I]^{-1} y_{1:t}$ and $\sigma_t^2(x) = k(x, x) - \boldsymbol{k}^\mathsf{T}[K + \sigma_\epsilon^2 I]^{-1}\boldsymbol{k}$, where $K$ is a matrix of size $t \times t$ with $(i, j)$-th element defined as $k(x_i, x_j)$ and $\boldsymbol{k}$ is a vector with $i$-th element defined as $k(x_i, x)$.

BO uses a surrogate function called *acquisition function* to find the next point to evaluate. The acquisition function uses the predictive distribution to balance two contrasting goals: sampling where the function is expected to take a high value vs. sampling where the uncertainty about the function value is high. Some well-known acquisition functions are probability of improvement (Kushner 1964), expected improvement (Jones, Schonlau, and Welch 1998), GP-upper confidence bound (Srinivas et al. 2012), and predictive entropy search (Hernández-Lobato, Hoffman, and Ghahramani 2014).

### Batch Bayesian Optimization

In its standard form, BO works in a *sequential setting* where it suggests one point at each iteration. However, when parallel resources are available at each iteration, BO is extended to a *batch setting* where it suggests multiple points at each iteration. Several batch BO methods have been developed (see (González et al. 2016) for a review). Recently, Thompson sampling has become an efficient technique for batch BO (Hernández-Lobato et al. 2017; Kandasamy et al. 2018); the idea is to select a batch element by maximizing a randomly drawn function from the posterior GP. However, to the best of our knowledge, batch BO for both categorical and continuous variables has not yet been studied.

## The Proposed Method

### Problem Definition

We present a method for BO to jointly handle both categorical and category-specific continuous input variables. Formally, given an input $[c, x_c]$ and a *black-box* function $f([c, x_c])$, where $c \in \{1, ..., C\}$ is a *category* among $C$ categories and $x_c \in \mathcal{X}_c \subset \mathbb{R}^d$ are *continuous* variables corresponding to the category $c$, our goal is to find:

$$[c^*, x^*] = \underset{c\in\{1,\ldots,C\}, x_c\in\mathcal{X}_c}{\text{argmax}} \; f([c, x_c]) \tag{1}$$

A naïve approach to solve Eq. (1) is to consider $f([c, x_c])$ as a *collection of black-box functions* $\{f_c(x_c)\}_{c=1}^C$ defined on continuous domain $\mathcal{X}_c$ and then find the optimum $x_c^* = \text{argmax}_{x_c\in\mathcal{X}_c} f_c(x)$ for each function and finally get $[c^*, x^*]$ as $c^* = \text{argmax}_{c\in\{1,\ldots,C\}} f_c(x_c^*)$ and $x^* = x_{c^*}^*$. However, this approach is *inefficient* as it needs to find $x_c^*$ for all $c$. Since the function evaluations in BO are expensive, an efficient approach is needed.

Before proceeding further, we simplify our notation. Instead of using $[c, x_c]$, we simply write $[c, x]$. The notation $x \in \mathcal{X}_c$ is sufficient to resolve any ambiguity.

### Sequential Setting

To efficiently solve the optimization problem in Eq. (1), we propose a novel method that combines multi-armed bandit

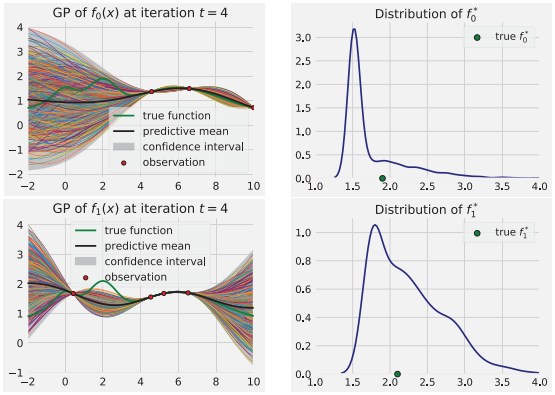
(MAB) and BO. The main idea is that instead of optimizing $f([c, x])$ exhaustively for each value of $c$, we attempt to identify (in parallel) the best value of $c$ (denoted as $c^*$), which results in the optimal function value and optimize $x^* = \text{argmax}_{x \in \mathcal{X}_{c^*}} f([c^*, x])$.

We formulate the problem of selecting the best value $c^*$ as a MAB problem (Bubeck, Cesa-Bianchi, and others 2012), where each value $c \in \{1, ..., C\}$ is considered as an arm. In the usual parlance of MAB, the arm $c$ has a reward distribution (due to noisy function evaluations) around a mean parameter $f_c^* \triangleq \max_{x \in \mathcal{X}_c} f_c(x)$. We note that $f_c^*$ is unknown due to $f_c(x)$ being a black-box. For BO of $f_c(x)$, we model it using a GP, which induces a distribution on $f_c^*$. As we increasingly get observations of $f_c(x)$, the uncertainty in $f_c^*$ reduces. From the MAB side, this means that each time an arm $c$ is played, we get an additional observation of $f_c(x)$, which improves the estimate of the mean parameter $f_c^*$. MAB algorithm allows us to play the arms optimally guided by the uncertainties in $f_c^*$. An illustration is shown in Figure 1.

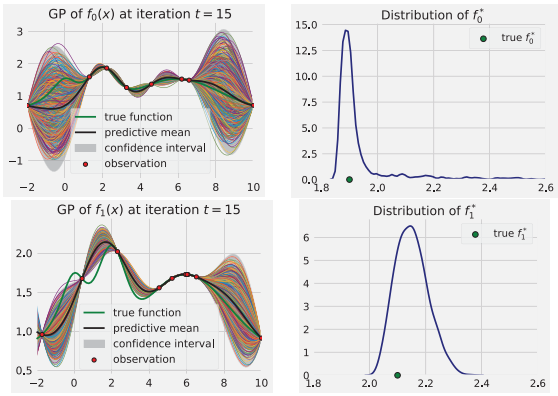
**Input:** $C$: # of arms, $B$: batch size
**begin**
  **for** $t = 1, 2, ...$ **do**
    **foreach** $c \in \{1, ..., C\}$ **do**
      fit GP$_c$ (*i.e.* $p(f_c(x) \mid \mathcal{D}_t^c)$) using $\mathcal{D}_t^c$;
    **end**
    **for** $b = 1$ **to** $B$ **do**
      **foreach** $c \in \{1, ..., C\}$ **do**
        draw $\tilde{f}_c(x) \sim p(f_c(x) \mid \mathcal{D}_t^c)$;
        obtain $\tilde{x}_c^* = \text{argmax}_{x \in \mathcal{X}_c} \tilde{f}_c(x)$;
        set $\tilde{f}_c^* = \tilde{f}_c(\tilde{x}_c^*)$;
      **end**
      choose an arm $c_{t+b} = \arg\max_c \tilde{f}_c^*$;
      suggest a point $x_{t+b} = \tilde{x}_{c_{t+b}}^*$;
      evaluate $y_{t+b} = f_{c_{t+b}}(x_{t+b}) + \epsilon_{t+b}$;
    **end**
    $\mathcal{D}_{t+B} = \mathcal{D}_t \cup \{c_{t+b}, x_{t+b}, y_{t+b}\}_{b=1}^B$;
  **end**
**end**
**Algorithm 1:** The proposed **Bandit-BO** algorithm.

We have several choices for MAB algorithms such as UCB1, Exp3, $\epsilon$-greedy, and Thompson sampling (Auer, Cesa-Bianchi, and Fischer 2002; Russo and Roy 2014) and similarly multiple choices for BO algorithms differing mainly in acquisition functions e.g. EI, GP-UCB, TS, entropy search etc (Hernández-Lobato, Hoffman, and Ghahramani 2014). We prefer to use GP-UCB[1] or TS for BO as these algorithms can be analyzed to provide theoretical upper bounds on regret. For MAB algorithm, we have decided to use TS for multiple reasons: (1) the GP directly offers a posterior distribution on the arm means $f_c^*$ and therefore using TS is feasible; (2) TS is flexible to modifications and amenable to theoretical analyses; and (3) TS is shown to achieve competitive performance in practice due to using a complete distribution for exploration/exploitation trade-off (Kandasamy et al. 2018).

[1]Although we focused only on TS for BO, but GP-UCB is also feasible both practically and theoretically. The key difference is that we get probabilistic regret bounds holding with high probability.




(a) Iteration $t = 4$.




(b) Iteration $t = 15$.

Figure 1: An illustration of our method using $C = 2$: (a) the results at $t = 4$, (b) the results at $t = 15$. In both (a) and (b), the first column shows the posterior GPs for category-0 and category-1 along with the true functions (shown in green color). The second column shows the estimated distributions of $f_0^*$ and $f_1^*$ with their true values (shown as green dots).

Using TS for both BO and MAB, our method works as follows. At each iteration $t$, for each arm $c$ we first model $f_c(x)$ via a GP using existing observations $\mathcal{D}_t^c = \{c_i, x_i, y_i = f_{c_i}(x_i) + \epsilon_i \mid c_i = c\}_{i=1}^t$. Collectively, we denote $\mathcal{D}_t = \cup_{c=1}^C \mathcal{D}_t^c$. We then randomly draw a function from the posterior GP distribution of each arm as $\tilde{f}_c(x) \sim p(f_c(x) \mid \mathcal{D}_t^c)$. Next we select an arm as $c_{t+1} = \underset{c \in \{1, ..., C\}}{\text{argmax}} \max_{x \in \mathcal{X}_c} \tilde{f}_c(x)$. Given the selected arm $c_{t+1}$ (or category), we need to recommend a value for $x_{t+1}$. Usually in BO, $x_{t+1}$ is recommended via an acquisition function. Since we are using TS as the acquisition function in BO, we recommend $x_{t+1} = \text{argmax}_{x \in \mathcal{X}_{c_{t+1}}} \tilde{f}_{c_{t+1}}(x)$. Finally, we observe the function value as $y_{t+1} = f_{c_{t+1}}(x_{t+1}) + \epsilon_{t+1}$ and update the observation set as $\mathcal{D}_{t+1}^{c_{t+1}} = \mathcal{D}_t^{c_{t+1}} \cup \{c_{t+1}, x_{t+1}, y_{t+1}\}$. We call our method **Bandit-BO**.

## Batch Setting

We extend our sequential method to *batch setting* to recommend a set of $B$ samples at each round. We note that the number of total function evaluations $T$ = the number of rounds $N \times$ the batch size $B$. Our batch algorithm is similar to the sequential one except that at each round, we recommend $B$ samples of $c$ and $x$, each sample obtained using an independent Thompson sample from $p(f_c(x) \mid \mathcal{D}_t^c)$. Using these recommendations $\{(c_{t+b}, x_{t+b})\}_{b=1}^{B}$, we evaluate the functions as $y_{t+b} = f_{c_{t+b}}(x_{t+b}) + \epsilon_{t+b}$ and update the observation set as $\mathcal{D}_{t+B} = \mathcal{D}_t \cup \{c_{t+b}, x_{t+b}, y_{t+b}\}_{b=1}^{B}$.

Our **Bandit-BO** in both sequential and batch settings is summarized in Algorithm 1.

## Convergence Analysis

We first present the convergence analysis for the sequential setting and then extend it to the batch setting. Our analysis is developed on the previous theoretical results of (Russo and Roy 2014; Desautels, Krause, and Burdick 2014; Kandasamy et al. 2018).

**Sequential Setting** TS has been analyzed earlier mostly in the context of MAB with finite arms. An exception is (Russo and Roy 2014), which extended the analysis to GP bandits to infinitely many dependent arms. BO using GP models is a related problem where one has to decide the best point among an uncountably infinite set of points specified by $\mathcal{X}_c$. In our analysis, we extend the results of (Russo and Roy 2014) to advance the BO in a joint space of categorical and continuous variables. Our analysis provides the convergence guarantee using *Bayesian regret*, which has been used as regret measure by several earlier works (Agrawal and Goyal 2013; Bubeck and Liu 2013).

Following (Russo and Roy 2014), the Bayesian regret of our proposed **Bandit-BO** after $T$ iterations is

$$\text{BayesRegret}(T) = \mathbb{E}\sum_{t=1}^{T}[f_{c^*}(x^*) - f_{c_t}(x_t)], \qquad (2)$$

where the expectation is w.r.t. a distribution over all possible functions $f_c$ in our hypothesis space and any randomness in the algorithm, particularly the random sampling of TS. Inserting and deleting $f_{c_t}(x^*_{c_t})$ in the above expression we can write the BayesRegret$(T)$ as

$$\text{BayesRegret}(T) = \mathbb{E}\sum_{t=1}^{T}[f_{c^*}(x^*) - f_{c_t}(x^*_{c_t})] + \mathbb{E}\sum_{t=1}^{T}[f_{c_t}(x^*_{c_t}) - f_{c_t}(x_t)] = R_T^{\text{MAB}} + R_T^{\text{BO}} \qquad (3)$$

where we have defined $R_T^{\text{MAB}} \triangleq \mathbb{E}\sum_{t=1}^{T}[f_{c^*}(x^*) - f_{c_t}(x^*_{c_t})]$ and $R_T^{\text{BO}} \triangleq \mathbb{E}\sum_{t=1}^{T}[f_{c_t}(x^*_{c_t}) - f_{c_t}(x_t)]$. The reason for using the terminology of $R_T^{\text{MAB}}$ and $R_T^{\text{BO}}$ is as follows: since $f_c(x^*_c)$ is the mean of arm $c$ rewards, $f_{c^*}(x^*) - f_{c_t}(x^*_{c_t})$ denotes the regret due to choosing a sub-optimal arm at iteration $t$; and similarly, given the choice of $c_t$ at iteration $t$, $f_{c_t}(x^*_{c_t}) - f_{c_t}(x_t)$ denotes the regret of BO choosing a sub-optimal continuous point.

To provide an upper bound on BayesRegret$(T)$ in Eq. (3), we prove Lemma 1 and 2, which provide upper bounds on $R_T^{\text{MAB}}$ and $R_T^{\text{BO}}$ respectively. Before we proceed, we need to state the following two assumptions. The first assumption is required to prove regret bounds for BO in a continuous search domain (Srinivas et al. 2012; Kandasamy et al. 2018). The second assumption is required to prove Lemma 1 and 2.

**Assumption 1.** Let $\{f_c(x)\}_{c=1}^{C}$ be a set of functions such that $x \in \mathcal{X}_c \subset \mathbb{R}^d$. Further, for all $c$, let $f_c(x) \sim \text{GP}_c(0, k_c)$ with a covariance function $k_c$ such that for any sample path of $\text{GP}_c$, there exist constants $r$ and $s$ such that its partial derivatives satisfy the following condition

$$\forall L > 0, \forall i \in \{1, \ldots, d\}\ \mathbb{P}(|\partial f/\partial x_i| < L) \geq 1 - dre^{-L^2/s^2}$$

**Assumption 2.** $\forall c \in \{1, ..., C\}$, let $f_c(x) \sim \text{GP}_c(0, k_c)$ with a covariance function $k_c$ such that the maximum information gain $\gamma_{T_c}$ about $f_c$ due to any $T_c$ noisy observations is strictly sub-linear in $T_c$. Therefore, there exists an $\alpha$ such that $\gamma_{T_c} \sim \mathcal{O}(T_c^{\alpha})$ where $0 \leq \alpha < 1$.

As stated in (Srinivas et al. 2012), Assumption 1 satisfies for any covariance function that is four times differentiable. Assumption 2 satisfies for common covariance functions e.g. squared-exponential, Matérn etc.

**Lemma 1** (Upper bound on $R_T^{\text{MAB}}$)**.** *Let $f_c(x) \sim GP_c(0, k_c), c = 1, ..., C$ and $\mathcal{D}_t = \{(c_i, x_i, y_i)\}_{i \leq t}$ be the noisy function observation set suggested by our method in the **sequential setting** under the observation model $y_i = f_{c_i}(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$, then under Assumptions 1 and 2, we have $R_T^{MAB} \leq \mathcal{O}\left(\sqrt{CT^{\alpha+1}\log T}\right)$.*

*Proof.* The proof is provided in supplementary material. □

**Lemma 2** (Upper bound on $R_T^{\text{BO}}$)**.** *Let $f_c(x) \sim GP_c(0, k_c), c = 1, ..., C$ and $\mathcal{D}_t = \{(c_i, x_i, y_i)\}_{i \leq t}$ be the noisy function observation set suggested by our method in the **sequential setting** under the observation model $y_i = f_{c_i}(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$, then under Assumptions 1 and 2, we have $R_T^{BO} \leq \mathcal{O}\left(\sqrt{CT^{\alpha+1}\log T}\right)$.*

*Proof.* The proof is provided in supplementary material. □

Finally, the *overall Bayesian regret* for our **Bandit-BO** is stated in Theorem 3.

**Theorem 3** (BayesRegret$(T)$ for Sequential Setting)**.** *Under Assumptions 1 and 2, the Bayesian regret for our method after $T$ iterations in the **sequential setting** is bounded as*

$$\textit{BayesRegret}\,(T) \leq \mathcal{O}\left(\sqrt{CT^{\alpha+1}logT}\right) \qquad (4)$$

*Proof.* The proof follows by combining the results of Lemma 1 and 2 with Eq. (3). We note that the regret grows only sub-linearly in both $T$ and $C$. □

**Efficiency of using MAB:** The Bayesian regret of our proposed **Bandit-BO** algorithm is only sub-linear in $T$ as seen in Eq. (4). We compare it with *two extreme settings*: the *first extreme* where the optimal arm (or category) is known, and the *second extreme* where each arm gets equal allocation e.g. visiting each arm in a round-robin fashion or sampling an arm uniformly randomly. For the first extreme, an oracle who knows the optimal arm $c^*$ will allocate all $T$ iterations to the arm $c^*$ and therefore will have $R_T^{\text{MAB}} = 0$ and $R_T^{\text{BO}} \leq \mathcal{O}(\sqrt{T^{\alpha+1}\log T})$. On the other hand, the second extreme being a naïve algorithm allocating equal budget to each arm will incur $R_T^{\text{MAB}} = \sum_{c \neq c^*} \Delta_c \frac{T}{C} = \mathcal{O}(T)$ (where $\Delta_c = f_{c^*}^* - f_c^*$ denoting the sub-optimality of each arm) and $R_T^{\text{BO}} \leq \mathcal{O}(\sqrt{T^{\alpha+1}\log T})$. This results in a total regret that grows linearly in $T$. Thus, our method with regret upper bound $\left(\sqrt{CT^{\alpha+1}\log T}\right)$ is a significantly better algorithm than equal-budget allocation algorithms and comparable to the Oracle with just an extra sub-linear factor $\sqrt{C}$.

**Batch Setting** The main difference between the analysis of a sequential and batch algorithm arises from the way function values are observed. Unlike a sequential setting where we observe the function value immediately after recommending a sample, a batch setting with batch size $B$ gets to observe the functions values only after recommending $B$ samples. Due to the late feedback on the function knowledge, $\sigma^c_{t_c}(x)$, the predictive variance of $f_c(x)$ in the batch setting, is higher than that in the sequential setting at any iteration $t_c$ prior to which there was a recommendation without function value observation. Desautels et al. (Desautels, Krause, and Burdick 2014) showed that this gap in the function knowledge (or increased uncertainty) due to the batch setting can be bounded for any $x \in \mathcal{X}$ as

$$\left(\sigma^c_{t_c}(x)\right)_{\text{batch}} \leq \left(\sigma^c_{t_c}(x)\right)_{\text{seq}} \psi_B, \tag{5}$$

where $\psi_B$ is a sub-linear term in $B$ related to the maximum information gain potentially brought by any $B$ samples. Eq. (5) gives us $\sum_{t_c=1}^{T_c} \left(\sigma^c_{t_c}(x)\right)^2_{\text{batch}} \leq \psi_B^2 \sum_{t_c=1}^{T_c} \left(\sigma^c_{t_c}(x)\right)^2_{\text{seq}}$. Since $\sum_{t_c=1}^{T_c} \left(\sigma^c_{t_c}(x)\right)^2_{\text{seq}} \leq \gamma_{T_C}$ (Srinivas et al. 2012), we have $\sum_{t_c=1}^{T_c} \left(\sigma^{t_c}_c(x)\right)^2_{\text{batch}} \leq \psi_B^2 \gamma_{T_C}$. For deriving regret bounds for the batch setting, to bound $\sum_{t_c=1}^{T_c} \left(\sigma^{t_c}_c(x)\right)^2_{\text{batch}}$, we can use $\psi_B^2 \gamma_{T_C}$ instead of $(\gamma_{T_c})_{\text{batch}}$. Since $\psi_B$ is independent of $T_c$, we can extend Theorem 3 of sequential setting to the batch setting as stated in the following Theorem.

**Theorem 4** (BayesRegret($T$) for Batch Setting)**.** *Under Assumptions 1 and 2, the Bayesian regret for our method after $T$ iterations in the **batch setting** is given as*

$$\textit{BayesRegret}\,(T) \leq \mathcal{O}\left(\psi_B \sqrt{CT^{\alpha+1} log T}\right)$$

*Proof.* The proof is provided in supplementary material. □

## Discussion

Our algorithm is capable of handling cases where the search space for continuous variables for each category is either identical (*category-independent* continuous search spaces) or different (*category-specific* continuous search spaces). We used an independent GP to model the continuous function $f_c(x)$ for each category $c$. When the search spaces for continuous variables are identical for all categories, it may be useful to incorporate any correlations across categories. In our algorithm, it is possible to incorporate such correlations through a multi-task Gaussian process (MTGP) (Bonilla, Chai, and Williams 2008). MTGP allows to use a covariance function of the form $k(c, x, c', x')$, which can be factorized as $k(c, c') \times k(x, x')$, and through $k(c, c')$ we can incorporate the correlation across the categories. A possible example of $k(c, c')$ is Hamming kernel that was used by (Wang et al. 2016). However, in this paper, since our focus is to provide a general algorithm that is applicable to both category-specific and category-independent continuous search spaces, we have ignored the correlation aspect. This is because correlation across any two categories having different continuous search spaces does not make sense.

## Experiments

We conduct experiments to show the performance of our proposed **Bandit-BO** for both synthetic and real-world applications in sequential and batch settings.

We compare our method with four state-of-the-art baselines that use different ways to deal with categorical variables: **One-hot-Encoding** (Golovin et al. 2017), **Merchan-Lobato** (Garrido-Merchán and Hernández-Lobato 2018), **SMAC** (Hutter, Hoos, and Leyton-Brown 2011), and **TPE** (Bergstra et al. 2011). For One-hot-Encoding and Merchan-Lobato methods, batch recommendation is made using Thompson sampling. For SMAC, we form the batch using "hallucinated" observations similar to (Desautels, Krause, and Burdick 2014). For TPE, we form the batch using the likelihood based sampling as described in (Bergstra et al. 2011). In our experiments, we randomly initialize two points for GP fitting for each category, resulting in $2C$ initial points in total. The initialization points are kept identical across all methods for a fair comparison. We repeat each method 10 times and report the average result along with the standard error.

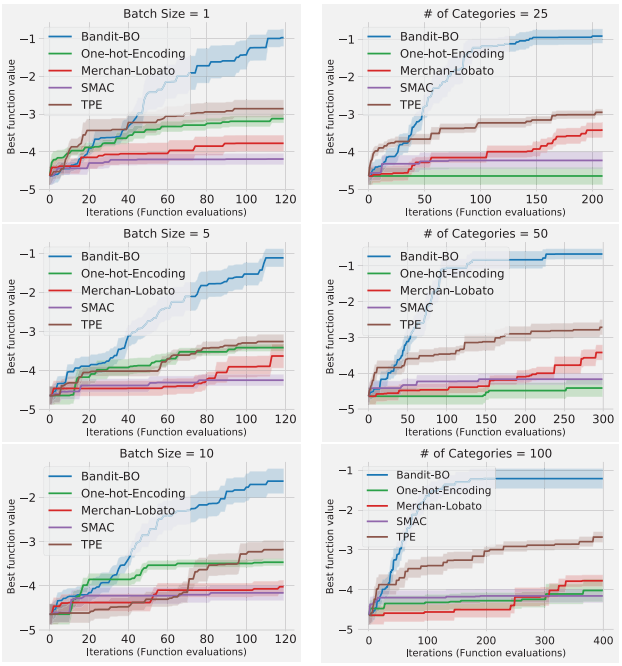
### Synthetic Applications

The first experiment illustrates how our algorithm performs with different batch sizes and numbers of categories.

**Synthetic function** Our synthetic function is created by *modifying* Ackley-5d function in 5 continuous variables by an extra categorical variable. We shift the function for each category by a value $c$ as $f([c, x]) = -20\exp(-0.2\sqrt{\frac{1}{5}\sum_{i=1}^{5} z_i^2}) - \exp(\frac{1}{5}\sum_{i=1}^{5}\cos(2\pi z_i)) + 20 + \exp(1) + c$, where $z_i = x_i + c$.

In the supplementary material, we provide additional experiments for *two more synthetic functions*.

**Study of varying batch sizes** Figure 2(a) shows the optimization results for different batch sizes while fixing the number of categories to 6. Our method **Bandit-BO** is significantly




(a) (b)

Figure 2: Optimization results for the *modified* Ackley-5d function (a) for *different batch sizes*: $B = 1$ (sequential), $B = 5$, and $B = 10$ (the number of categories is fixed to 6) and (b) for *different numbers of categories*: $C = 25$, $C = 50$, and $C = 100$ (the batch size is fixed to 5).

better compared to the other methods. It shows consistent improvements over $T = 120$ iterations. One-hot-Encoding, Merchan-Lobato, and SMAC methods do not perform well. TPE is the second-best method; however its best found function value is significantly lower than that of **Bandit-BO**.

**Study of varying numbers of categories** In Figure 2(b), we show the optimization results for different *large* numbers of categories while fixing the batch size to 5. We can see that **Bandit-BO** clearly outperforms the other methods even with a very large number of categories (e.g. $C = 100$). TPE is still a second-best method. The performance of Merchan-Lobato becomes worse as $C$ is increased. Interestingly, One-hot-Encoding shows an improvement as $C$ is increased up to 100, where it is better than SMAC.

## Real-world Applications

The second experiment shows the efficiency of our method in two real-world machine learning (ML) applications.

**Hyper-parameter tuning for a feed-forward neural network on a regression task** Our goal is to find the optimal set of hyper-parameters for a feed-forward neural network on the *protein structure* dataset[2], which has 27,438 training points, 9,146 testing points, and nine features. We define the black-box function as a mapping between the model hyper-parameters and the *mean squared error* (MSE) on a held-out testing set. We build the network with two hidden layers and train it using Adam (Kingma and Ba 2015) for 100 epochs. We optimize eight hyper-parameters as shown in Table 1. We report the average MSE with standard errors for each method as shown in Figure 3. We note that to create arms from four categorical variables, we use the cross product of their values, resulting in 36 arms (choices) in total.

Table 1: Hyper-parameters for the neural network.

| Type | Hyper-parameter | Values |
|---|---|---|
| Categorical | Activation/Layer 1 | {tanh, relu} |
| | Activation/Layer 2 | {tanh, relu} |
| | Layer 1 Size | $2^{\{4,6,9\}}$ |
| | Layer 2 Size | $2^{\{4,6,9\}}$ |
| Continuous | Initial Learning Rate | $10^{[-4,-1]}$ |
| | Batch Size | $2^{[3,6]}$ |
| | Dropout/Layer 1 | $[0.0, 0.6]$ |
| | Dropout/Layer 2 | $[0.0, 0.6]$ |

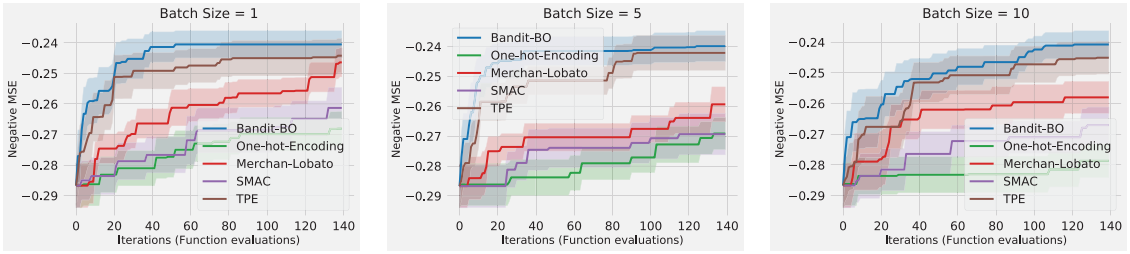
From the results in Figure 3, we can see that our method **Bandit-BO** performs the best in both sequential and batch settings. Similar to the results on the synthetic function, TPE is the second-best method. When the batch size is increased up to 5, TPE is slightly comparable to our method. Merchan-Lobato performs well with all batch sizes, where it is much better than One-hot-Encoding and SMAC. One-hot-Encoding is slightly comparable to SMAC as the batch size is increased up to 5 (Figure 3(b)).

***Automated Machine Learning*: Automatic selection of the best ML model along with its optimal hyper-parameters** Given a dataset and several candidate ML models, e.g. decision tree, random forest, logistic regression, support vector machine, etc, our goal is to determine which model along with its optimized hyper-parameters produces the highest *accuracy* on the dataset. We formulate this task as a black-box function $f([c, x_c])$ optimization, where $c$ indexes a "ML model" and $x_c$ is a set of hyper-parameters specified for that model (the detail of ML models and their hyper-parameters are provided in the supplementary material). We emphasize that $x_c$ is different for different models e.g. $x_c$ can be the "penalty parameter" when $c$ is a "linear support vector machine" while $x_c$ can be the "initial learning rate" and the "regularization parameter" when $c$ is a "logistic regression". Under such complex search space, single GP-based BO methods such as One-hot-Encoding and Merchan-Lobato cannot work. In contrast, as discussed earlier our method straightforwardly works with this setting thanks to fitting different GPs for different values $c$. We optimize the black-box function on 30 benchmark datasets[3], compared with three well-known state-of-the-art *automated machine learning* packages, namely **Hyperopt-sklearn**[4] (using TPE for optimiza-

[2]https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure

[3]Download from https://www.openml.org. Each dataset is randomly split into 80% for training and 20% for testing.

[4]https://github.com/hyperopt/hyperopt-sklearn




(a) (b) (c)

Figure 3: Results of the hyper-parameter tuning for the neural network – best value (*negative* MSE) vs. iteration for batch sizes: (a) $B = 1$ (sequential), (b) $B = 5$, and (c) $B = 10$.

Table 2: Characteristics ($|D|$: the number of samples, $|F|$: the number of features, and $|L|$: the number of labels) of the first 16 benchmark datasets along with *classification accuracy* (standard error) of our method **Bandit-BO** and other methods. Bold font marks the best performance in a row. The results for full 30 datasets are reported in the supplementary material.

| **Dataset** | **Format** | $\|D\|$ | $\|F\|$ | $\|L\|$ | **Bandit-BO** | **Hyperopt-sklearn** | **Auto-sklearn** | **TPOT** |
|---|---|---|---|---|---|---|---|---|
| *wine* | tabular | 178 | 13 | 3 | **98.33 (0.00)** | 97.78 (0.01) | 97.50 (0.01) | 96.67 (0.01) |
| *breast_cancer* | tabular | 569 | 30 | 2 | **97.02 (0.01)** | 95.44 (0.00) | 96.40 (0.00) | 96.84 (0.00) |
| *analcatdata_authorship* | text | 841 | 70 | 4 | **99.76 (0.00)** | 99.47 (0.00) | 99.41 (0.00) | 99.53 (0.00) |
| *diabetes* | tabular | 768 | 8 | 2 | **77.40 (0.01)** | 73.70 (0.02) | 76.95 (0.01) | 77.01 (0.01) |
| *electricity* | tabular | 45,312 | 8 | 2 | **92.29 (0.00)** | 92.21 (0.00) | 90.89 (0.00) | 90.94 (0.00) |
| *wall_robot_navigation* | trajectory | 5,456 | 24 | 4 | **99.73 (0.00)** | **99.73 (0.00)** | 99.43 (0.00) | 99.46 (0.00) |
| *vehicle* | tabular | 846 | 18 | 4 | **81.71 (0.01)** | 78.71 (0.01) | 80.24 (0.01) | 78.12 (0.01) |
| *cardiotocography* | tabular | 2,126 | 35 | 10 | **100.0 (0.00)** | **100.0 (0.00)** | 99.98 (0.00) | **100.0 (0.00)** |
| *artificial_characters* | text | 10,218 | 7 | 10 | 90.47 (0.01) | **90.94 (0.00)** | 82.49 (0.00) | 87.75 (0.01) |
| *monks1* | tabular | 556 | 6 | 2 | **100.0 (0.00)** | 99.82 (0.00) | 99.73 (0.00) | **100.0 (0.00)** |
| *monks2* | tabular | 601 | 6 | 2 | 98.26 (0.01) | 97.69 (0.01) | 97.36 (0.01) | **99.92 (0.00)** |
| *steel_plates_fault* | tabular | 1,941 | 33 | 2 | **100.0 (0.00)** | **100.0 (0.00)** | **100.0 (0.00)** | **100.0 (0.00)** |
| *phoneme* | tabular | 5,404 | 5 | 2 | **90.23 (0.00)** | 90.21 (0.00) | 89.25 (0.00) | 89.58 (0.00) |
| *waveform* | tabular | 5,000 | 40 | 3 | **86.45 (0.00)** | 86.42 (0.00) | 86.19 (0.00) | 86.28 (0.00) |
| *balance_scale* | tabular | 625 | 4 | 3 | **98.48 (0.01)** | 97.20 (0.01) | 89.04 (0.01) | 92.32 (0.01) |
| *digits* | image | 1,797 | 64 | 10 | 98.25 (0.00) | **98.67 (0.00)** | 98.08 (0.00) | 97.86 (0.00) |

tion) (Komer, Bergstra, and Eliasmith 2019), **Auto-sklearn**[5] (using SMAC for optimization) (Feurer et al. 2019), and Tree-Based Pipeline Optimization Tool (**TPOT**[6]) (Olson and Moore 2019). All methods are applied to the same training and test sets and repeated 10 times.

Table 2 shows the classification results on 16 datasets (the results for 30 datasets are reported in the supplementary material), where **Bandit-BO** clearly results in better classification compared with other methods. More specifically, **Bandit-BO** achieves up to 4%, 9%, and 6% improvements over Hyperopt-sklearn, Auto-sklearn, and TPOT respectively. On four large datasets (*electricity*, *wall_robot_navigation*, *phoneme*, and *waveform*), our method is better than three baselines. The improvements are more significant on five small datasets (*wine*, *breast_cancer*, *diabetes*, *vehicle*, and *balance_scale*). The classification performances of Hyperopt-sklearn, Auto-sklearn, and TPOT are comparable. In the supplementary material, we report the overall accuracy of each method across 30 datasets, where **Bandit-BO** is the best method (92.25% accuracy). The overall classification results of Hyperopt-sklearn (91.48% accuracy), Auto-sklearn (91.82% accuracy), and TPOT (91.77% accuracy) are quite similar.

[5]https://github.com/automl/auto-sklearn

[6]https://github.com/EpistasisLab/tpot

## Conclusion

We have introduced a novel BO method to globally optimize expensive black-box functions involving both categorical and continuous variables. We formulated the problem as a MAB problem, where each category corresponds to an arm with its reward distribution centered around the optimum of the objective function in continuous variables. Our solution uses Thompson sampling, which connects both MAB and BO in a unified framework. Our method is capable of handling optimization problems where each category is associated with a different continuous search space. We also extended our method for batch optimization. We rigorously analyzed the convergence providing sub-linear regret bounds. Our experiments using several synthetic and real-world applications demonstrate the usefulness of our proposed method.

## Acknowledgment

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

Agrawal, S., and Goyal, N. 2013. Further optimal regret bounds for thompson sampling. In *AISTATS*, 99–107.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2-3):235–256.

Bergstra, J.; Bardenet, R.; Bengio, Y.; and Kégl, B. 2011. Algorithms for hyper-parameter optimization. In *NIPS*, 2546–2554.

Bonilla, E. V.; Chai, K. M.; and Williams, C. 2008. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, 153–160.

Bubeck, S., and Liu, C.-Y. 2013. Prior-free and prior-dependent regret bounds for thompson sampling. In *NIPS*, 638–646.

Bubeck, S.; Cesa-Bianchi, N.; et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5(1):1–122.

Bull, A. 2011. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research* 12:2879–2904.

Desautels, T.; Krause, A.; and Burdick, J. 2014. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research* 15(1):3873–3923.

Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; and Hutter, F. 2019. *Auto-sklearn: Efficient and Robust Automated Machine Learning*. Springer International Publishing. 113–134.

Garrido-Merchán, E., and Hernández-Lobato, D. 2018. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *arXiv preprint arXiv:1805.03463*.

Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; and Sculley, D. 2017. Google vizier: A service for black-box optimization. In *KDD*, 1487–1495. ACM.

González, J.; Dai, Z.; Hennig, P.; and Lawrence, N. 2016. Batch bayesian optimization via local penalization. In *AISTATS*, 648–657.

Hernández-Lobato, M.; Requeima, J.; Pyzer-Knapp, E.; and Aspuru-Guzik, A. 2017. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In *ICML*, 1470–1479.

Hernández-Lobato, M.; Hoffman, M.; and Ghahramani, Z. 2014. Predictive entropy search for efficient global optimization of black-box functions. In *NIPS*, 918–926.

Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, 507–523. Springer.

Jones, D.; Schonlau, M.; and Welch, W. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13(4):455–492.

Kandasamy, K.; Krishnamurthy, A.; Schneider, J.; and Póczos, B. 2018. Parallelised bayesian optimisation via thompson sampling. In *AISTATS*, 133–142.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Komer, B.; Bergstra, J.; and Eliasmith, C. 2019. *Hyperopt-Sklearn*. Springer International Publishing. 7–111.

Kushner, H. 1964. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering* 86(1):97–106.

Lakshminarayanan, B.; Roy, D. M.; and Teh, Y. W. 2016. Mondrian forests for large-scale regression when uncertainty matters. In *Artificial Intelligence and Statistics*, 1478–1487.

Olson, R., and Moore, J. 2019. *TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning*. Springer International Publishing. 151–160.

Russo, D., and Roy, B. 2014. Learning to optimize via posterior sampling. *Mathematics of Operations Research* 39(4):1221–1243.

Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.; and Freitas, N. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE* 104(1):148–175.

Srinivas, N.; Krause, A.; Kakade, S.; and Seeger, M. 2012. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory* 58(5):3250–3265.

Wang, Z.; Hutter, F.; Zoghi, M.; Matheson, D.; and de Feitas, N. 2016. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research* 55:361–387.