Bayesian Optimisation over Multiple Continuous and Categorical Inputs

Binxin Ru [* 1]   **Ahsan S. Alvi** [* 1]   **Vu Nguyen** [1]   **Michael A. Osborne** [1]   **Stephen J Roberts** [1]

Abstract

Efficient optimisation of black-box problems that comprise both continuous and categorical inputs is important, yet poses significant challenges. Current approaches, like one-hot encoding, severely increase the dimension of the search space, while separate modelling of category-specific data is sample-inefficient. Both frameworks are not scalable to practical applications involving *multiple* categorical variables, each with *multiple* possible values. We propose a new approach, Continuous and Categorical Bayesian Optimisation (CoCaBO), which combines the strengths of multi-armed bandits and Bayesian optimisation to select values for both categorical and continuous inputs. We model this mixed-type space using a Gaussian Process kernel, designed to allow sharing of information across *multiple* categorical variables; this allows Co-CaBO to leverage all available data efficiently. We extend our method to the batch setting and propose an efficient selection procedure that dynamically balances exploration and exploitation whilst encouraging batch diversity. We demonstrate empirically that our method outperforms existing approaches on both synthetic and real-world optimisation tasks with continuous and categorical inputs.

1. Introduction

Existing work has shown Bayesian Optimisation (BO) to be remarkably successful at optimising functions with continuous input spaces (Snoek et al., 2012; Hennig & Schuler, 2012; Hernández-Lobato et al., 2015; Ru et al., 2018; Shahriari et al., 2016; Frazier, 2018; Alvi et al., 2019). However, in many situations, optimisation problems involve a mixture of continuous and categorical variables.

---
[*]Equal contribution [1]University of Oxford. Correspondence to: Binxin Ru <robin@robots.ox.ac.uk>.

For example, with a deep neural network, we may want to adjust the learning rate and the weight decay (continuous), as well as the activation function type in each layer (categorical). Similarly, in a gradient boosting ensemble of decision trees, we may wish to adjust the learning rate and the maximum depth of the trees (continuous), as well as the boosting algorithm and loss function (categorical).

Having a mixture of categorical and continuous variables presents challenges. If some inputs are categorical variables, then the common assumption that the BO acquisition function is differentiable over the input space, which allows the acquisition function to be efficiently optimised, is no longer valid. Recent research has dealt with categorical variables in different ways. The simplest approach for BO with Gaussian process (GP) (Rasmussen & Williams, 2006) surrogates is to use a one-hot encoding on the categorical variables, so that they can be treated as continuous variables, and perform BO on the transformed space (GPyOpt, 2016). Alternatively, the mixed-type inputs can be handled using a hierarchical structure, such as using random forests (Hutter et al., 2011; Bergstra et al., 2011) or multi-armed bandits (MABs) (Gopakumar et al., 2018). These approaches come with their own challenges, which we will discuss below (see Section 3). In particular, the existing approaches are not well designed for *multiple* categorical variables with *multiple* possible values. Additionally, no GP-based BO methods have explicitly considered the batch setting for continuous-categorical inputs, to the best of our knowledge.

In this paper, we present a new Bayesian optimisation approach for optimising a black-box function with multiple continuous and categorical inputs, termed Continuous and Categorical Bayesian Optimisation (CoCaBO). Our approach is motivated by the success of MABs (Auer et al., 2002a;b) in identifying the best value(s) from a discrete set of options.

Our main contributions are as follows:

- We propose a method which combines the strengths of MABs and BO to optimise black-box functions with *multiple* categorical and continuous inputs. (Section 4).

- We present a GP kernel to capture complex interac-

tions between the continuous and categorical inputs (Section 4.2). Our kernel allows sharing of information across different categories without resorting to one-hot transformations.

- We introduce a novel batch selection method for mixed input types that extends CoCaBO to the parallel setting, and dynamically balances exploration and exploitation and encourages batch diversity (Section 4.3).

- We demonstrate the effectiveness of our methods on a variety of synthetic and real-world optimisation tasks with *multiple* categorical and continuous inputs (Section 5).

2. Preliminaries

In this paper, we consider the problem of optimising a black-box function $f(\mathbf{z})$ where the input $\mathbf{z}$ consists of both continuous and categorical inputs, $\mathbf{z} = [\mathbf{h}, \mathbf{x}]$, where $\mathbf{h} = [h_1, \ldots, h_c]$ are the categorical variables, with each variable $h_j \in \{1, 2, \ldots, N_j\}$ taking one of N_j different values, and $\mathbf{x}$ is a point in a d-dimensional hypercube $\mathcal{X}$. Formally, we aim to find the best configuration to maximise the black-box function

$$\mathbf{z}^* = [\mathbf{h}^*, \mathbf{x}^*] = \arg\max_{\mathbf{z}} f(\mathbf{z}) \qquad (1)$$

by making a series of evaluations $\mathbf{z}_1, \ldots, \mathbf{z}_T$. Later we extend our method to allow parallel evaluation of multiple points, by selecting a batch $\{\mathbf{z}_t^{(i)}\}_{i=1}^b$ at each optimisation step t.

Bayesian optimisation (Brochu et al., 2010; Shahriari et al., 2016; Nguyen & Osborne, 2020) is an approach for optimising a black-box function $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ such that its optimal value is found using a small number of evaluations. BO often uses a Gaussian process (Rasmussen & Williams, 2006) surrogate to model the objective f. A GP defines a probability distribution over functions f, as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and covariance functions respectively, which encode our prior beliefs about f. Using the GP posterior, BO defines an acquisition function $\alpha_t(\mathbf{x})$ which is optimised to identify the next location to sample $\mathbf{x}_t = \arg\max_{x \in \mathcal{X}} \alpha_t(\mathbf{x})$. Unlike the original objective function $f(\mathbf{x})$, the acquisition function $\alpha_t(\mathbf{x})$ is cheap to compute and can be optimised using standard techniques.

3. Related Work

3.1. One-hot encoding

A common method for dealing with categorical variables is to transform them into a one-hot encoded representation,

where a variable with N choices is transformed into a vector of length N with a single non-zero element. This is the approach followed by popular BO packages like Spearmint (Snoek et al., 2012) and GPyOpt (González et al., 2016; GPyOpt, 2016).

There are two main drawbacks with this approach. First, the commonly-used kernels in the GP surrogate assume that f is continuous and differentiable in the input space, which is not the case for one-hot encoded variables, as the objective is only defined for a small subspace within this representation.

The second drawback is that one-hot encoding adds many dimensions to the search space, making the continuous optimisation of the acquisition function much harder. Additionally, the distances between one-hot encoded categorical variables tend to be large. As a result, the optimisation landscape is characterised by many flat regions, increasing the difficulty of optimisation (Rana et al., 2017).

3.2. Category-specific continuous inputs

We consider two recent approaches in handling category-specific continuous inputs, a specific setting of mixed categorical-continuous. The first approach is EXP3BO (Gopakumar et al., 2018), which can deal with mixed categorical and continuous input spaces by utilising a MAB. When the categorical variable is selected by the MAB, EXP3BO constructs a GP surrogate specific to the chosen category for modelling the continuous domain, i.e. it shares no information across the different categories. The observed data are divided into smaller subsets, one for each category, and as a result EXP3BO can handle only a small number of categorical choices and requires a large number of samples. Recently, Nguyen et al. (2019) considered extending the category-specific continuous inputs to the batch setting using Thompson sampling (Russo et al., 2018).

3.3. Hierarchical approaches

Random forests (RFs) (Breiman, 2001) can naturally consider continuous and categorical variables, and are used in SMAC (Hutter et al., 2011) as the underlying surrogate model for f. However, the predictive distribution of the RF, which is used to select the next evaluation, is less reliable, as it relies on randomness introduced by the bootstrap samples and the randomly chosen subset of variables to be tested at each node to split the data. Moreover, RFs can easily overfit and we need to carefully choose the number of trees. Another tree-based approach is Tree Parzen Estimator (TPE) (Bergstra et al., 2011), an optimisation algorithm based on tree-structured Parzen density estimators. TPE uses nonparametric Parzen kernel density estimators to model the distribution of good and bad configurations w.r.t. a reference value. Due to the nature of kernel den-

Algorithm 1 CoCaBO Algorithm

1: **Input:** A black-box function f, observation data $\mathcal{D}_0$, maximum number of iterations T
2: **Output:** The best recommendation $\mathbf{z}_T = [\mathbf{x}_T, \mathbf{h}_T]$
3: **for** $t = 1, \ldots, T$ **do**
4:     Select $\mathbf{h}_t = [h_{1,t}, \ldots, h_{c,t}] \leftarrow \text{MAB}(\{\mathbf{h}_i, f_i\}_i^{t-1})$
5:     Select $\mathbf{x}_t = \arg\max \alpha_t(\mathbf{x}|\mathcal{D}_{t-1}, \mathbf{h}_t)$
6:     Query at $\mathbf{z}_t = [\mathbf{x}_t, \mathbf{h}_t]$ to obtain f_t
7:     $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\mathbf{z}_t, f_t)$ and update the reward distributions of MAB with $(\mathbf{h}_t, f_t)$
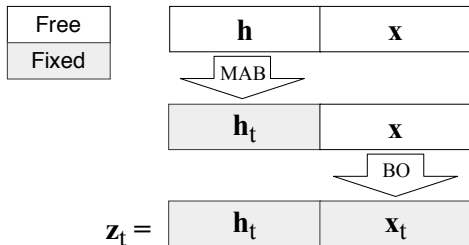8: **end for**

Figure 1: Optimisation procedure in CoCaBO.

sity estimators, TPE also supports continuous and discrete spaces.

3.4. Integer-continuous inputs

There are also other approaches for integer-continuous variables (Swiler et al., 2014; Garrido-Merchán & Hernández-Lobato, 2019; Daxberger et al., 2019; Pelamatti et al., 2020) which are related, but not essentially suitable for mixed categorical-continuous variables.

Remark. To the best of our knowledge, none of the above approaches has solved satisfactorily for the mixed type variables where (1) the continuous variables are not specific to a category and (2) we have *multiple* categorical variables each of which include *multiple* choices.

4. Continuous and Categorical Bayesian Optimisation (CoCaBO)

Our proposed method, Continuous and Categorical Bayesian Optimisation, harnesses both the advantages of multi-armed bandits to select categorical inputs and the strengths of GP-based BO in optimising continuous input spaces. The CoCaBO procedure is shown in Algorithm 1. CoCaBO first decides the values of the categorical inputs $\mathbf{h}_t$ by using MAB (Step 4 in Algorithm 1). Given $\mathbf{h}_t$, it then maximises the acquisition function to select the continuous part $\mathbf{x}_t$ which forms the next point $\mathbf{z}_t = [\mathbf{h}_t, \mathbf{x}_t]$ for evaluation, as illustrated in Figure 1.

4.1. CoCaBO bandit algorithm

The design of the MAB system in the CoCaBO algorithm is motivated by two considerations.

First, our problem setting deals with *multiple* categorical inputs $\mathbf{h} = [h_1, \ldots, h_c]$ and the value of each h_j is determined by a MAB agent. This results in a multi-agent MAB system and requires coordination. We achieved the coordination across the agents by using a joint reward function. Specifically, at each iteration, each agent decides the categorical value for its own variable among the corresponding N_j choices but the objective function is evaluated based on the joint decision of all the agents. This function value is then used to update the reward distribution for the chosen category among all the agents. A concurrent work (Carlucci et al., 2020) also uses the similar coordination in deploying multi-agent learning system and shows its effectiveness on their neural architecture search application.

Second, we adopt the EXP3 (Auer et al., 2002b) method as the MAB algorithm because it makes comparatively fewer assumptions on reward distributions and can deal with the general setting of adversarially-defined rewards. The general adversarial bandit setting is important for our application as it enables the MAB part of our algorithm to handle the non-stationary nature of the rewards observed during the optimisation (Allesiardo et al., 2017); the function value for the given values of categorical variables will improve over iterations as we apply BO on the continuous space. In contrast, Thompson Sampling (Thompson, 1933), UCB and ϵ-greedy assume that the rewards are i.i.d. samples from stationary distributions (Allesiardo et al., 2017), thus are not suitable for our application.

The resultant multi-agent EXP3 algorithm has a cumulative regret of $\mathcal{O}\left(c\sqrt{TN\log(N)}\right)$, which is sublinear in T and increases with the number of categorical inputs c as well as the number of choices N for each categorical input, as derived in Appendix C. By using the MAB to decide the values for categorical inputs, we only need to optimise the acquisition function over the continuous subspace $\mathcal{X} \in \mathbb{R}^d$. In comparison to one-hot based methods, whose acquisition functions are defined over $\mathbb{R}^{(d+\sum_i^c N_i)}$, our approach enjoys a significant reduction in the difficulty and cost of optimising the acquisition function[1].

In Figure 2, we demonstrate the effectiveness of our approach in dealing with categorical variables via a simple synthetic example *Func-2C* (described in Section 5), which

---

[1]To optimise the acquisition function to within ζ accuracy using a grid search or branch-and-bound optimiser, our approach requires only $\mathcal{O}(\zeta^{-d})$ calls and one-hot approaches require $\mathcal{O}(\zeta^{-(d+\sum_i^c N_i)})$ calls. The cost-saving grows exponentially with the number of categories c and number of choices for each category N_i.

comprises two categorical inputs, h_1 ($N_1 = 3$) and h_2 ($N_2 = 5$), and two continuous inputs. The optimal function value lies in the subspace when both categorical variables $h_1 = h_2 = 2$. The categories chosen by CoCaBO at each iteration, the histogram of all selections and the rewards for each category are shown for 200 iterations. We can see that CoCaBO successfully identifies and focuses on the correct categories.

4.2. CoCaBO kernel design

We propose to use a combination of two separate kernels: $k_z(\mathbf{z}, \mathbf{z}')$ will combine $k_h(\mathbf{h}, \mathbf{h}')$, a kernel defined over the categorical inputs, with $k_x(\mathbf{x}, \mathbf{x}')$, for the continuous inputs.

For the categorical kernel, we propose using an indicator-based similarity metric, $k_h(\mathbf{h}, \mathbf{h}') = \frac{\sigma}{c} \sum_{i=1}^{c} \mathbb{I}(h_i - h_i')$, where σ is the kernel variance and $\mathbb{I}(h_i, h_i') = 1$ if $h_i = h_i'$ and is zero otherwise. This kernel can be derived as a special case of a RBF kernel, which is explored in Appendix D.

There are several ways of combining kernels that result in valid kernels (Duvenaud et al., 2013). One approach is to sum them together. Using a sum of kernels, that are each defined over different subsets of an input space, has been used successfully for BO in the context of high-dimensional optimisation in the past (Kandasamy et al., 2015). Simply adding the continuous kernel to the categorical kernel $k_x(\mathbf{x}, \mathbf{x}') + k_h(\mathbf{h}, \mathbf{h}')$, though, provides limited expressiveness, as this translates in practice to learning a single common trend over $\mathbf{x}$, and an offset depending on $\mathbf{h}$.

An alternative approach is to use the product of the two kernels $k_x(\mathbf{x}, \mathbf{x}') \times k_h(\mathbf{h}, \mathbf{h}')$. This form allows the kernel to encode couplings between the continuous and categorical domains, allowing a richer set of relationships to be captured, but if there are no overlapping categories in the data, which is likely to occur in early iterations of BO, this would cause the product kernel to be zero and prevent the model from learning.

We therefore propose the CoCaBO kernel as a mixture of the sum and product kernels:

$$k_z(\mathbf{z}, \mathbf{z}') = (1 - \lambda)\big(k_h(\mathbf{h}, \mathbf{h}') + k_x(\mathbf{x}, \mathbf{x}')\big) + \lambda k_h(\mathbf{h}, \mathbf{h}')k_x(\mathbf{x}, \mathbf{x}'), \qquad (2)$$

The trade-off between them is controlled by a parameter $\lambda \in [0, 1]$, which can be optimised jointly with the GP hyperparameters (see Appendix E).

It is worth highlighting a key benefit of our formulation over alternative hierarchical methods discussed in Section 3.3. Rather than dividing our data into a subset for each combination of categories, we instead use all of our acquired data at every stage of the optimisation, as our kernel

---

Algorithm 2 CoCaBO batch selection

1: **Input:** Observation data $\mathcal{D}_{t-1}$
2: **Output:** The batch $\mathcal{B}_t = \{\mathbf{z}_t^{(1)}, \dots, \mathbf{z}_t^{(b)}\}$
3: $\mathbf{H}_t = \{\mathbf{h}_t^{(1)}, \dots, \mathbf{h}_t^{(b)}\} \leftarrow$ Batch MAB$(\mathcal{D}_{t-1})$
4: $(\mathbf{u}_1, v_1), \dots, (\mathbf{u}_q, v_q)$ are the unique categorical values in $\mathbf{H}_t$ and their counts
5: Initialise $\mathcal{B}_t = \varnothing$ and $\mathcal{D}'_{t-1} = \mathcal{D}_{t-1}$
6: **for** $j = 1, \dots, q$ **do**
7: $\quad \{\mathbf{x}_i\}_{i=1}^{v_j} \leftarrow$ KB$(\mathbf{u}_j, \mathcal{D}'_{t-1})$
8: $\quad \mathbf{Z}_j = \{\mathbf{u}_j, \mathbf{x}_i\}_{i=1}^{v_j}$ and $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \mathbf{Z}_j$
9: $\quad \mathcal{D}'_t \leftarrow \mathcal{D}'_{t-1} \cup \{\mathbf{Z}_j, \mu(\mathbf{Z}_j)\}$
10: **end for**
11: **Output:** $\mathcal{B}_t$

---

is able to combine information from data within the same category as well as from different categories, which improves its modelling performance (Section 5.3).

Our kernel is able to learn the covariances k_z between observations in the joint input space $\mathcal{Z} = \mathcal{H} \times \mathcal{X}$. When two data points come from different categories, k_z is dominated by k_x (in the additive term). When there is some overlap in categorical choices between data points, then k_z is determined by contributions from both k_x and k_h (in both additive and multiplicative terms). In comparison, the one-hot encoded kernel converts all values of the categorical variables into additional continuous dimensions, allowing it to give nonzero covariances between points from different categories. We compare the regression performance of the CoCaBO kernel and a one-hot encoded kernel on some synthetic functions in Section 5.1 and show that our kernel leads to a superior predictive performance over one-hot kernel.

4.3. Batch CoCaBO

Our focus on optimising computer simulations and modelling pipelines provides a strong motivation to extend CoCaBO to select and evaluate multiple tasks at each iteration, in order to better utilise available hardware resources (Snoek et al., 2012; Wu & Frazier, 2016; Shah & Ghahramani, 2015; Contal et al., 2013).

The batch CoCaBO algorithm uses the "multiple plays" formulation of EXP3, called EXP3.M (Auer et al., 2002b), which returns a batch of categorical choices, and combines it with the Kriging Believer (KB)[2] (Ginsbourger et al., 2010) batch method to select the batch points in the continuous domain. We choose KB for the batch creation, as it can consider already-selected batch points, including those with different categorical values, without making restric-

---

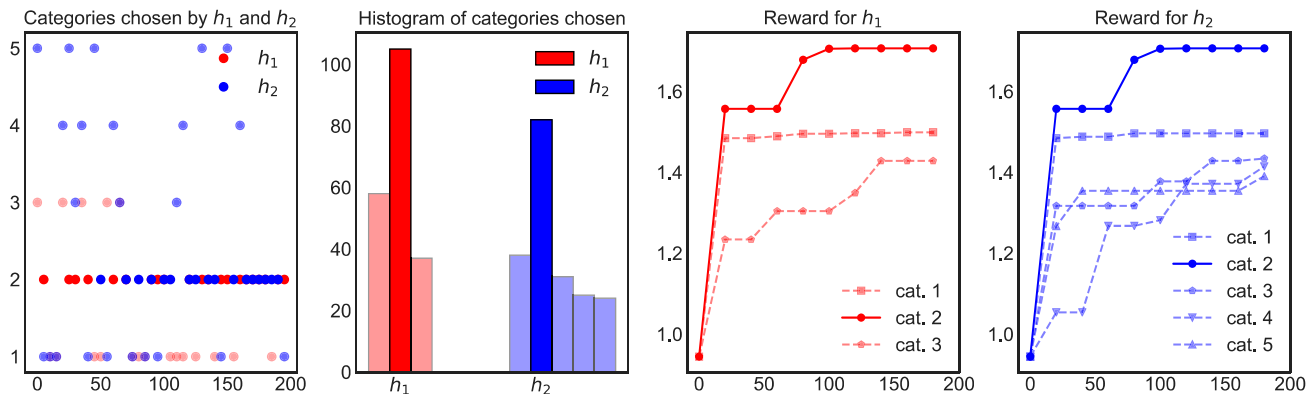[2]Note that our approach can easily utilise other batch selection techniques if desired.

Figure 2: CoCaBO correctly optimises the two categorical inputs h_1 (Red) and h_2 (Blue) of the *Func-2C* test function over 200 iterations. The *best* category is $h_i = 2$ for both h_1 ($N_1 = 3$) and h_2 ($N_2 = 5$), and is highlighted in all plots. The left subplot shows the selections made by CoCaBO, showing how the both categorical inputs increasingly focus on the best categories as the algorithm progresses. The second left subplot shows the histogram of categories selected, with the best category being chosen the most frequently. The right subplots show the reward for each categorical value for h_1 and h_2 across iterations. Again, we see the correct category being identified for both categorical inputs for the highest rewards.
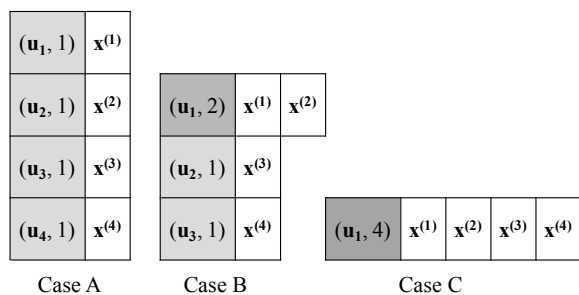

Figure 3: Three example cases for selecting a batch of 4 unique points ($b = 4$). $\mathbf{u}_j$ represents a unique categorical point. If all the categorical points in the batch are different, we select 1 continuous location $\mathbf{x}^{(i)}$ conditioned on each $\mathbf{u}_j$ (Case A). If 2 out of a batch of 4 categorical points are equal to $\mathbf{u_1}$, we sequentially select 2 continuous locations $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ conditioned on $\mathbf{u_1}$ using KB and collect only 1 continuous location for the other two categorical data $\mathbf{u_2}$ and $\mathbf{u_3}$ (Case B).

tive assumptions as do other popular techniques, e.g. local penalisation (González et al., 2016; Alvi et al., 2019) assumes that f is Lipschitz continuous. A detailed description of KB algorithm is in Appendix F. Our novel contribution is a method for combining the batch points selected by EXP3.M with batch BO procedures for continuous input spaces. Assume we are selecting a batch of b points $\mathcal{B}_t = \{\mathbf{z}_t^{(i)}\}_{i=1}^b$ at iteration t. A simple approach is to select a batch of categorical variables $\mathbf{H}_t = \{\mathbf{h}_t^{(i)}\}_{i=1}^b$ and then choose a corresponding continuous variable for each categorical point as in the sequential algorithm above,

thus forming $\{\mathbf{z}_t^{(i)}\}_{i=1}^b = \{\mathbf{h}_t^{(i)}, \mathbf{x}_t^{(i)}\}_{i=1}^b$. However, such a batch method may not identify b unique locations, as some values in $\{\mathbf{h}_t^{(i)}\}_{i=1}^b$ may be repeated and you will get identical $\mathbf{x}$ values for repeated $\mathbf{h}_t^{(i)}$ values. This is even more problematic when the number of possible combinations for the categorical variables, $\prod_{i=1}^c N_i$, is smaller than the batch size b, as we would never identify a full batch of unique points.

Our batch selection method, outlined in Algorithm 2, allows us to create a batch of unique points $\mathcal{B}_t = \{\mathbf{z}_t^{(i)}\}_{i=1}^b$ (i.e. $\mathbf{z}_t^{(i)} \neq \mathbf{z}_t^{(j)}$ if $i \neq j$) by allocating multiple continuous batch points to more desirable categories. The key idea is to first collect the q unique categorical points $\{\mathbf{u}_j\}_{j=1}^q$ and how often they occur $\{v_j\}_{j=1}^q$ from the batch $\mathbf{H}_t$. The count v_j defines how many continuous batch points will be selected via KB for each unique categorical point $\mathbf{u}_j$. This process is illustrated in Figure 3 for three possible scenarios. The benefit of using KB here is that the algorithm can take into account selections across the different $\mathbf{h}$ to impose diversity in the batch in a consistent manner.

5. Experiments

We compared CoCaBO against a range of existing methods which are able to handle problems with mixed type inputs: SMAC (Hutter et al., 2011), TPE (Bergstra et al., 2011), GP-based Bayesian optimisation with one-hot encoding (One-hot BO) (GPyOpt, 2016) and EXP3BO (Gopakumar et al., 2018). We did not compare to the concurrent category-specific approach of (Nguyen et al., 2019) because its code is not released yet. However, the method (Nguyen et al., 2019) is highly similar to EXP3BO and

Table 1: Categorical and continuous inputs to be optimised for real-world tasks. N_i in the parentheses indicate the number of categorical choices that each categorical input has.

	SVM-Boston ($d = 3, c = 3$)	*XG-MNIST*($d = 5, c = 3$)	*NAS-CIFAR10* ($d = 22, c = 5$)
h	kernel type ($N_1 = 4$), kernel coefficient ($N_2 = 2$), using shrinking ($N_3 = 2$)	booster type ($N_1 = 2$), grow policies ($N_2 = 2$), training objectives ($N_3 = 2$)	Operation choice for the 5 intermediate node in the directed acyclic graph (DAG) of architecture design ($\{N_i = 3\}_i^5$)
x	penalty parameter, tolerance for stopping, model complexity	learning rate, regularisation, maximum depth, subsample, minimum split loss	creation probability for each of the 21 possible edges in the DAG, number of edges present in the DAG

Table 2: Mean and standard error of the predictive log likelihood of the CoCaBO and the One-hot BO surrogates on synthetic test functions. Both models were trained on 250 samples and evaluated on 100 test points. The CoCaBO surrogate can model the function surface better than the One-hot surrogate as the number of categorical variables increases.

	Ackley-2C	*Ackley-3C*	*Ackley-4C*	*Ackley-5C*	*Func-2C*	*Func-3C*
CoCaBO	$-69.1(\pm9.47)$	$\mathbf{-74.9(\pm3.17)}$	$\mathbf{-92.0(\pm6.31)}$	$-114(\pm8.31)$	$\mathbf{115(\pm14.1)}$	$\mathbf{167(\pm6.54)}$
One-hot	$\mathbf{-60.4(\pm5.91)}$	$-107(\pm23.7)$	$-102(\pm7.94)$	$-120(\pm10.3)$	$12.4(\pm10.7)$	$-81.0(\pm4.69)$

suffers the same limitations as EXP3BO. For all the baseline methods, we used their publicly available Python packages[3]. CoCaBO and one-hot BO both use the UCB acquisition function (Srinivas et al., 2010) with scale parameter $\kappa = 2.0$. In all experiments, we tested four different λ values for our method[4]: $\lambda = 1.0, 0.5, 0.0$, auto, where $\lambda = $ auto means λ is optimised as a hyperparameter. This leads to four variants of our method: CoCaBO-1.0, CoCaBO-0.5, CoCaBO-0.0 and CoCaBO-auto. We used a Matérn, $\nu = \frac{5}{2}$, kernel for k_x, as well as for One-hot BO, and used the indicator-based kernel discussed in Section 4.2 for k_h. For both our method and One-hot BO, we optimised the GP hyperparameters by maximising the log marginal likelihood every 10 iterations using multi-started gradient descent (see Appendix E for more details).

TPE is only used in the sequential setting ($b = 1$) because its package HyperOpt does not provide a synchronous batch implementation. We started each optimisation method except EXP3BO with 24 random initial points. EXP3BO requires much more initial data to start with because as mentioned in Section 3.2, it needs to fit a GP surrogate to each category. For example, if the problem involves 2 categorical variables, each of which has 3 categorical choices, EXP3BO needs to construct 9 independent GP surrogates by dividing the observation data into 9 subsets

[3]One-hot BO: https://github.com/SheffieldML/GPyOpt, SMAC: https://github.com/automl/pysmac, TPE: https://github.com/hyperopt/hyperopt, EXP3BO: https://github.com/shivapratap/AlgorithmicAssurance_NIPS2018

[4]https://github.com/rubinxin/CoCaBO_code

(one for each surrogate). When applying EXP3BO for our problems, we start it with 3 random initial points for each GP surrogate. For all the problems, the continuous inputs were normalised to $\mathbf{x} \in [-1, 1]^d$ and all experiments were conducted on a 36-core 2.3GHz Intel Xeon processor with 512 GB RAM.

We tested all these methods on a diverse set of synthetic and real problems:

- *Func-2C* is a test problem with 2 continuous inputs ($d = 2$) and 2 categorical inputs ($c = 2$). The categorical inputs control a linear combination of three 2-dimensional global optimisation benchmark functions: beale, six-hump camel and rosenbrock. This is the function used for the illustration in Figure 2;

- *Func-3C* is similar to *Func-2C* but with 3 categorical inputs which leads to more complicated combinations of the three functions;

- *Ackley-cC*, with $c = \{2, 3, 4, 5\}$ and $d = 1$ is generated to test the performance of CoCaBO on problems with large numbers of categorical inputs and inputs with large numbers of categorical choices. Here, we convert c dimensions of the $(c + 1)$-dimensional Ackley function into 17 categories each;

- *SVM-Boston* outputs the negative mean square test error of using a support vector machine (SVM) for regression on the Boston housing dataset (Dua & Graff, 2017);
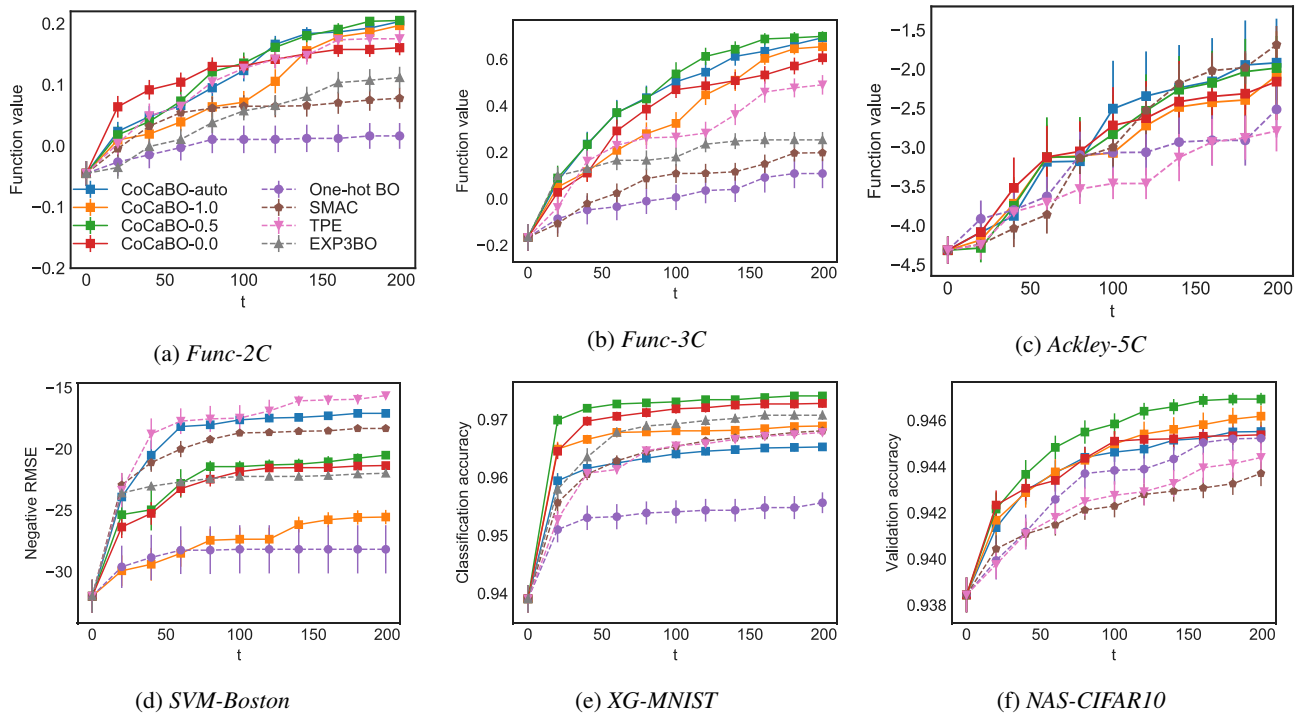
(a) *Func-2C*

(b) *Func-3C*

(c) *Ackley-5C*

(d) *SVM-Boston*

(e) *XG-MNIST*

(f) *NAS-CIFAR10*

Figure 4: Performance of CoCaBOs against existing methods on various tasks in the sequential setting ($b = 1$).

- *XG-MNIST* returns classification test accuracy of a XGBoost model (Chen & Guestrin, 2016) on MNIST (LeCun & Cortes, 2010);

- *NAS-CIFAR10* performs the architecture search on convolutional neural network topology for CIFAR10 classification. We conducted the search using the NAS-Bench-101 dataset (Ying et al., 2019) and adopted the search space proposed in (Ying et al., 2019) which encodes each unique architecture with 5 categorical variables and 22 continuous variables.

A brief summary of categorical and continuous inputs for the real-world problems is shown in Table 2 and a detailed description of all the problems is provided in Appendix G.

5.1. Predictive performance of the CoCaBO posterior

We first investigate the quality of the CoCaBO-auto surrogate by comparing its modelling performance against a standard GP with one-hot encoding. We train each model on 250 uniformly randomly sampled data points and evaluate the predictive log likelihood on 100 test data points. The mean and standard error over 20 random initialisations are presented in Table 2. The results showcase the benefit of using the CoCaBO kernel over a kernel with one-hot encoded inputs, especially when the number of categorical inputs grows. The CoCaBO kernel, which allows it to learn a richer set of variations from the data, leads to consistently

better out-of-sample predictions.

5.2. Performance of CoCaBO in sequential setting

We evaluated the optimisation performance of our proposed CoCaBO methods and other existing methods in the sequential setting. We ran each sequential optimisation method for $T = 200$ iterations. We performed 20 random repetitions for synthetic problems and 10 random repetitions for the real-world problems. The mean and standard error over all repetitions are presented in Figure 4. For almost all synthetic and real-world problems, CoCaBO methods outperform other competing approaches with CoCaBO-0.5 and CoCaBO-auto consistently demonstrating the most competitive performance.

Note that we did not apply EXP3BO for *Ackley-5C* and *NAS-CIFAR10* because both cases involve multiple categorical variables and each categorical variable has multiple categories, thus requiring EXP3BO to build a huge number of independent GP surrogates (17^5 and 3^5 respectively) to model the objective problem. And this in turn demands a large amount of observation data to start the optimisation. Hence, EXP3BO is not suitable for problems involving *multiple* categorical inputs with *multiple* possible values. Even in the cases where EXP3BO is compared (Figure 4a, 4b, 5d, 5e), CoCaBO outperformed EXP3BO. This justifies the gain in efficiency by using CoCaBO kernel to leverage all the data in a single surrogate in contrast with
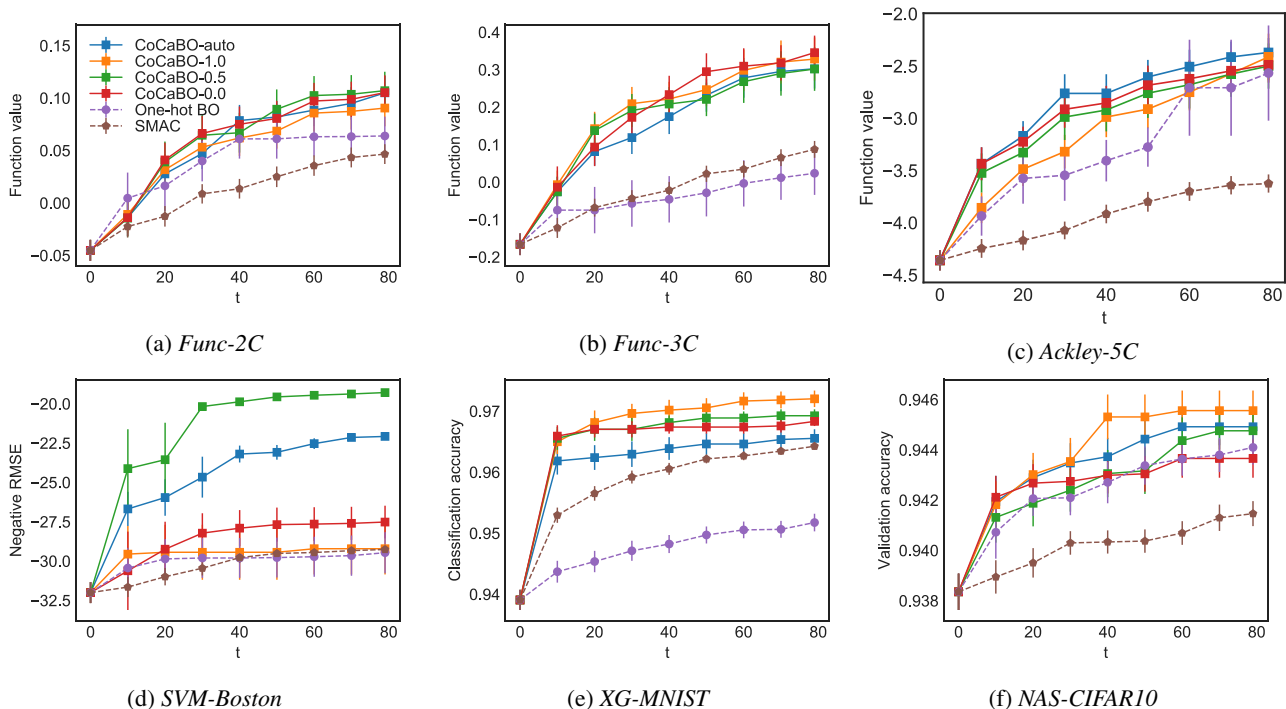
Figure 5: Performance of CoCaBOs against existing methods on synthetic and real-world tasks in the batch setting ($b = 4$).

subdividing data by categories to learn multiple separate surrogates.

Another observation is that TPE performs relatively well in the case when the number of categorical combinations ($\prod_i^c cN_i$) is small (e.g. *Func-2C*(15), *SVM-Boston*(16)) but performs clearly worse than CoCaBO when categorical combinations are large (e.g. *Func-3C*(60), *NAS-CIFAR10*(243), *Ackley-5C*(17^5)).

5.3. Performance of CoCaBO in batch setting

Now we move to experiments in the batch setting. We ran each batch optimisation with $b = 4$ for $T = 80$ iterations. The mean and standard error of the optimisation performance over 20 random repetitions are presented in Figure 5. CoCaBO methods again show competitive performance over other batch methods with CoCaBO-0.5 and CoCaBO-auto again remaining the top performing λ options. This supports our hypothesis that combining the sum and product kernels is beneficial for search over the mixed-type input space. Therefore, we recommend $\lambda = 0.5$ or $\lambda =$ auto as the default options for using our algorithm.

One point to note is that CoCaBO-1.0 (product kernel only) performs quite competitively in the batch setting for problems like XG-MNIST and NAS-CIFAR10 whose categorical variables have few category choices. At the same iteration number, the batch setting provides more sample data for the BO algorithm than the sequential setting, leading to

more frequent overlapping categories in the data, especially when the category choices are small. Hence, the product kernel, which captures the couplings between the continuous and categorical data, can perform more effectively.

6. Conclusion

Existing BO literature uses one-hot transformations or hierarchical approaches to encode real-world problems involving mixed continuous and categorical inputs. We presented a solution from a novel perspective, called Continuous and Categorical Bayesian Optimisation (CoCaBO), that harnesses the strengths of multi-armed bandits and GP-based BO to tackle this problem. Our method uses a new kernel structure, which allows us to capture information within categories as well as across different categories. This leads to more efficient use of the acquired data and improved modelling power. We extended CoCaBO to the batch setting, enabling parallel evaluations at each stage of the optimisation. CoCaBO demonstrated strong performance over existing methods on a variety of synthetic and real-world optimisation tasks with *multiple* continuous and categorical inputs. We find CoCaBO to offer a very competitive alternative to existing approaches.

References

Allesiardo, Robin, Féraud, Raphaël, and Maillard, Odalric-Ambrym. The non-stationary stochastic multi-armed bandit problem. *International Journal of Data Science and Analytics*, 3(4):267–283, 2017.

Alvi, Ahsan, Ru, Binxin, Calliess, Jan-Peter, Roberts, Stephen, and Osborne, Michael A. Asynchronous batch bayesian optimisation with improved local penalisation. In *International Conference on Machine Learning*, pp. 253–262, 2019.

Auer, Peter, Cesa-Bianchi, Nicolo, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002a.

Auer, Peter, Cesa-Bianchi, Nicolo, Freund, Yoav, and Schapire, Robert E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002b.

Bergstra, J S, Bardenet, R, Bengio, Y, and Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011.

Breiman, Leo. Random forests. *Machine learning*, 45(1):5–32, 2001.

Brochu, Eric, Cora, Vlad M, and De Freitas, Nando. 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.

Carlucci, Fabio Maria, Esperança, Pedro M, Singh, Marco, Gabillon, Victor, Yang, Antoine, Xu, Hang, Chen, Zewei, and Wang, Jun. {MANAS}: Multi-agent neural architecture search, 2020.

Chen, Tianqi and Guestrin, Carlos. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

Contal, Emile, Buffoni, David, Robicquet, Alexandre, and Vayatis, Nicolas. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Machine Learning and Knowledge Discovery in Databases*, pp. 225–240. Springer, 2013.

Daxberger, Erik, Makarova, Anastasia, Turchetta, Matteo, and Krause, Andreas. Mixed-variable Bayesian optimization. *arXiv preprint arXiv:1907.01329*, 2019.

Dua, Dheeru and Graff, Casey. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Duvenaud, David, Lloyd, James, Grosse, Roger, Tenenbaum, Joshua, and Zoubin, Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pp. 1166–1174, 2013.

Frazier, Peter I. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Garrido-Merchán, Eduardo C and Hernández-Lobato, Daniel. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing*, 2019.

Ginsbourger, David, Le Riche, Rodolphe, and Carraro, Laurent. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pp. 131–162. Springer, 2010.

González, Javier, Dai, Zhenwen, Hennig, Philipp, and Lawrence, Neil D. Batch Bayesian optimization via local penalization. In *International Conference on Artificial Intelligence and Statistics*, pp. 648–657, 2016.

Gopakumar, Shivapratap, Gupta, Sunil, Rana, Santu, Nguyen, Vu, and Venkatesh, Svetha. Algorithmic assurance: An active approach to algorithmic testing using Bayesian optimisation. In *Advances in Neural Information Processing Systems*, pp. 5465–5473, 2018.

GPyOpt. GPyOpt: A Bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt, 2016.

Hennig, Philipp and Schuler, Christian J. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

Hernández-Lobato, José Miguel, Gelbart, Michael, Hoffman, Matthew, Adams, Ryan, and Ghahramani, Zoubin. Predictive entropy search for Bayesian optimization with unknown constraints. In *International Conference on Machine Learning*, pp. 1699–1707, 2015.

Hutter, Frank, Hoos, Holger H, and Leyton-Brown, Kevin. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011.

Kandasamy, Kirthevasan, Schneider, Jeff, and Póczos, Barnabás. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pp. 295–304, 2015.

LeCun, Yann and Cortes, Corinna. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Nguyen, Dang, Gupta, Sunil, Rana, Santu, Shilton, Alistair, and Venkatesh, Svetha. Bayesian optimization for categorical and category-specific continuous inputs. *arXiv preprint arXiv:1911.12473*, 2019.

Nguyen, Vu and Osborne, Michael A. Knowing the what but not the where in Bayesian optimization. In *International Conference on Machine Learning*, 2020.

Pelamatti, Julien, Brevault, Loïc, Balesdent, Mathieu, Talbi, El-Ghazali, and Guerin, Yannick. Overview and comparison of gaussian process-based surrogate models for mixed continuous and discrete variables: Application on aerospace design problems. In *High-Performance Simulation-Based Optimization*, pp. 189–224. Springer, 2020.

Rana, Santu, Li, Cheng, Gupta, Sunil, Nguyen, Vu, and Venkatesh, Svetha. High dimensional Bayesian optimization with elastic Gaussian process. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 2883–2891, 2017.

Rasmussen, C E and Williams, C K I. *Gaussian processes for machine learning*. 2006.

Ru, Binxin, Osborne, Michael, and McLeod, Mark. Fast information-theoretic Bayesian optimisation. In *International Conference on Machine Learning*, 2018.

Russo, Daniel J, Van Roy, Benjamin, Kazerouni, Abbas, Osband, Ian, Wen, Zheng, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Shah, Amar and Ghahramani, Zoubin. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pp. 3312–3320, 2015.

Shahriari, Bobak, Swersky, Kevin, Wang, Ziyu, Adams, Ryan P, and de Freitas, Nando. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Srinivas, Niranjan, Krause, Andreas, Kakade, Sham, and Seeger, Matthias. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1015–1022, 2010.

Swiler, Laura P, Hough, Patricia D, Qian, Peter, Xu, Xu, Storlie, Curtis, and Lee, Herbert. Surrogate models for mixed discrete-continuous variables. In *Constraint Programming and Decision Making*, pp. 181–202. Springer, 2014.

Thompson, William R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Wu, Jian and Frazier, Peter I. The parallel knowledge gradient method for batch Bayesian optimization. In *NIPS*, 2016.

Ying, Chris, Klein, Aaron, Real, Esteban, Christiansen, Eric, Murphy, Kevin, and Hutter, Frank. Nas-bench-101: Towards reproducible neural architecture search. *arXiv preprint arXiv:1902.09635*, 2019.