Bayesian Optimisation over Multiple Continuous and Categorical Inputs

Binxin Ru *
Department of Engineering Science
University of Oxford
robin@robots.ox.ac.uk

Ahsan S. Alvi*
Department of Engineering Science
University of Oxford
asa@robots.ox.ac.uk

Vu Nguyen
Department of Engineering Science
University of Oxford
vu@robots.ox.ac.uk

Michael A. Osborne
Department of Engineering Science
University of Oxford
mosb@robots.ox.ac.uk

Stephen J Roberts
Department of Engineering Science
University of Oxford
sjrob@robots.ox.ac.uk

Abstract

Efficient optimisation of black-box problems that comprise both continuous and categorical inputs is important for meta-learning, yet poses significant challenges. 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, each with *multiple* possible values; this allows CoCaBO 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. 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 [26, 15, 16, 23, 25, 11, 2]. However, in many meta-learning situations, optimisation problems involve a mixture of continuous and categorical variables. For example, with a deep neural network, we may want to adjust the learning rate and the number of units in each layer (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 (both continuous), as well as the boosting algorithm and loss function (both categorical). Having a mixture of categorical and continuous variables presents unique challenges. If some inputs are categorical variables, then the common assumption that the BO acquisition function is differentiable and continuous over the input space, which allows the acquisition function to be efficiently optimised, is no longer valid.

*These authors contributed equally.

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada.

Recent research has dealt with categorical variables in different ways. The simplest approach for BO with Gaussian process (GP) 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 [4]. Usually we use a RBF or Matérn kernel in the GP surrogate and both of these kernels assume the unknown function to be defined everywhere in the input space. For a one-hot transformed input space this is not the case, as the function is only defined in a subspace therein. The second drawback affects the optimisation of the acquisition function. By inflating the dimensionality of the search space we make the problem sparse. As a result, the optimisation landscape is characterised by many flat regions, which is difficult to optimise [21].

Alternatively, the mixed-type inputs can be handled using a hierarchical structure, such as random forests [17, 5] or independent GP surrogates for different categories (EXP3BO) [14]. These approaches also come with their own challenges, e.g. the predictive distribution of random forests (RFs) relying heavily on the randomness introduced by the bootstrap samples, as well as the risk of RFs overfitting the data. These 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 main contributions are as follows: First, we propose a novel method which combines the strengths of MABs and BO to optimise black-box functions with *multiple* categorical and continuous inputs. (Section 3). Second, we present a GP kernel to capture complex interactions between the continuous and categorical inputs (Section 3.1). Our kernel allows sharing of information across different categories without resorting to one-hot transformations. Third, we introduce a novel batch selection method for mixed input types that extends CoCaBO to the parallel setting, that dynamically balances exploration and exploitation and encourages batch diversity (Section 3.2). Finally, we demonstrate the effectiveness of our methods on a variety of synthetic and real-world optimisation tasks with *multiple* categorical and continuous inputs (Section 4).

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 $h_i \in \{1, 2, \ldots, N_i\}$ taking one of N_i 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

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

by making a series of T evaluations $\mathbf{z}_1$,..., $\mathbf{z}_T$. 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.

3 Continuous and Categorical Bayesian Optimisation (CoCaBO)

Our proposed method, Continuous and Categorical Bayesian Optimisation (CoCaBO), harnesses both the advantages of multi-armed bandits (MABs), to select categorical inputs, and the strength of GP-based BO in optimising continuous input spaces. CoCaBO first decides the values of the categorical inputs $\mathbf{h}_t$ by using a MAB. Given $\mathbf{h}_t$, we then maximise 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. The algorithm of CoCaBO is presented in Appendix B. We define the MAB's reward for each category as the best function value observed so far from that category. We chose the EXP3 [3] method as the MAB because it can deal with adversarially-defined rewards, which better models the non-stationary nature of the rewards observed during an optimisation [1]: the best-so-far function value at each iteration is either equal to, or better than, previous values. In contrast, Thompson Sampling [27], UCB and ϵ-greedy assume that the rewards are i.i.d. samples from stationary distributions, thus not suitable for our application.

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,

Algorithm 1 CoCaBO batch selection

Input: Surrogate data $\mathcal{D}_{t-1}$
Output: The batch $\mathcal{B}_t = \{\mathbf{z}_t^{(1)}, \ldots, \mathbf{z}_t^{(b)}\}$
$\mathbf{H}_t = \{\mathbf{h}_t^{(1)}, \ldots, \mathbf{h}_t^{(b)}\} \leftarrow$ EXP3-M($\mathcal{D}_{t-1}$)
$(\mathbf{u}_1, v_1), \ldots, (\mathbf{u}_q, v_q)$ are the unique categorical values in $\mathbf{H}_t$ and their counts
Initialise $\mathcal{B}_t = \varnothing$ and $\mathcal{D}'_{t-1} = \mathcal{D}_{t-1}$
for $j = 1, \ldots, q$ **do**
    $\{\mathbf{x}_i\}_{i=1}^{v_j} \leftarrow \text{KB}(\mathbf{u}_j, \mathcal{D}'_{t-1})$
    $\mathbf{Z}_j = \{\mathbf{u}_j, \mathbf{x}_i\}_{i=1}^{v_j}$ and $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \mathbf{Z}_j$
    $\mathcal{D}'_t \leftarrow \mathcal{D}'_{t-1} \cup \{\mathbf{Z}_j, \mu(\mathbf{Z}_j)\}$
end for
return $\mathcal{B}_t$

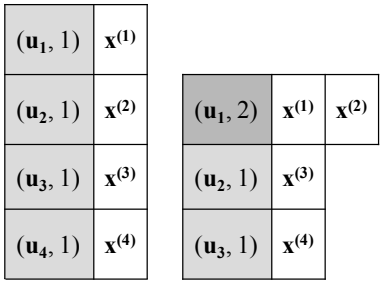


Figure 1: Two example cases for selecting a batch ($b = 4$)

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[2].

3.1 CoCaBO kernel design

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

There are several ways of combining kernels that result in valid kernels [10]. 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 [18]. Simply adding the continuous kernel to the categorical kernel $k_z(\mathbf{z}, \mathbf{z}') = k_x(\mathbf{x}, \mathbf{x}') + k_h(\mathbf{h}, \mathbf{h}')$, 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 $k_z(\mathbf{z}, \mathbf{z}') = k_x(\mathbf{x}, \mathbf{x}') \times k_h(\mathbf{h}, \mathbf{h}')$. 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 our CoCaBO kernel to automatically exploit the strengths and avoid the weaknesses of the sum and product kernels by a trade-off parameter λ which can be optimised jointly with the GP hyperparameters:

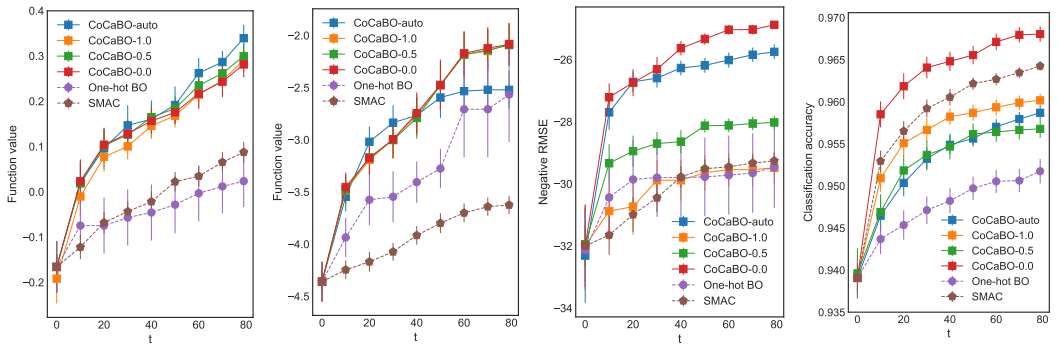
$$k_z(\mathbf{z}, \mathbf{z}') = (1 - \lambda)\left(k_h(\mathbf{h}, \mathbf{h}') + k_x(\mathbf{x}, \mathbf{x}')\right) + \lambda k_h(\mathbf{h}, \mathbf{h}')k_x(\mathbf{x}, \mathbf{x}'), \tag{2}$$

where $\lambda \in [0, 1]$ controls the relative contribution of the sum and product kernels.

It is worth highlighting a key benefit of our formulation over alternative hierarchical methods: rather than dividing our data into a subset for each combination of categories, we instead leverage all of the acquired data at every stage of the optimisation. Our kernel is able to combine information from data within the same category, as well as from different categories, which improves its modelling performance, especially as the number of categorical inputs grows. We demonstrated this superior performance of our CoCaBO kernel by comparing the regression performance of the CoCaBO kernel and a one-hot encoded kernel on synthetic functions in Appendix D.1.

3.2 Batch CoCaBO

[2]To optimise the acquisition function to within ζ accuracy via grid search or branch-and-bound optimiser, our approach requires only $\mathcal{O}(\zeta^{-d})$ calls [18] 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.



(a) *Func-3C* (b) *Ackley-5C* (c) *SVM-Boston* (d) *XG-MNIST*

Figure 2: Performance of CoCaBOs against existing methods in the batch setting ($b = 4$).

Our focus on optimising hyperparameters of machine learning algorithms provides a strong motivation to extend CoCaBO to select and evaluate multiple tasks at each iteration, in order to better utilise available computing resources [26, 29, 24, 9].

The batch CoCaBO algorithm uses the "multiple plays" formulation of EXP3, called EXP3.M [3], which returns a batch of categorical choices, and combines it with the Kriging Believer (KB)[3] batch method [12] to select the batch points in the continuous domain. 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 (i.e. $\mathbf{z}_t^{(i)} \neq \mathbf{z}_t^{(j)} \forall i \neq j$), as some values in $\{\mathbf{h}_t^{(i)}\}_{i=1}^b$ may be repeated.

Our novel batch selection method, outlined in Algorithm 1, allows us to create a batch of unique locations by allocating multiple continuous batch points to more desirable categories. The key idea is to first collect the q unique categorical choices $\{\mathbf{u}_j\}_{j=1}^q$ and how often they occur $\{v_j\}_{j=1}^q$ from the batch $\mathbf{H}_t$. The counts v_j define how many continuous batch points will be selected for each categorical choice $\mathbf{u}_j$. This is illustrated in Figure 1 for two possible scenarios.

4 Experiments

We compared CoCaBO against GP-based Bayesian optimisation with one-hot encoding (One-hot BO) [4] and SMAC [17]. For all the baseline methods, we used their publicly available Python packages. We did not compare against EXP3BO [14] because we focus on problems involving *multiple* categorical inputs with *multiple* possible values, and EXP3BO can handle only one categorical input with few possible values. In all experiments, we tested different λ values (1.0, 0.5, 0.0, auto) for our method, CoCaBO-λ, where $\lambda =$ auto means λ is optimised as a hyperparameter. We tested all these methods on synthetic and real problems which involve d continuous inputs and c categorical inputs. Due to space constraints, the sequential experimental results are provided in Appendix D.2.

We show the optimisation performance of our proposed CoCaBO methods and baseline methods on two synthetic functions, Func-3C ($d = 2, c = 3$) and Ackley-5C ($d = 1, c = 5$), and two real-world optimisation tasks, SVM-Boston ($d = 3, c = 3$) and XG-MNIST ($d = 5, c = 3$). These functions are defined in Appendix C. The mean and standard error over 20 random repetitions are presented in Figure 2 and all methods used a batch size of $b = 4$. It is evident that CoCaBO methods outperform all other competing approaches in these problems. We note that despite CoCaBO-$auto$ still remaining very competitive, the strong performance of CoCaBO-0.0 suggests independence between the categorical and continuous input spaces in the real-world tasks, making the additive kernel structure sufficient.

[3]Note that our approach can easily utilise other batch selection techniques for the continuous variables.

5 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.

References

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

[2] Ahsan S Alvi, Binxin Ru, Jan Calliess, Stephen J Roberts, and Michael A Osborne. Asynchronous batch Bayesian optimisation with improved local penalisation. *arXiv preprint arXiv:1901.10452*, 2019.

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

[4] The GPyOpt authors. GPyOpt: A Bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

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

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

[7] Eric Brochu, Vlad M Cora, and Nando De Freitas. 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.

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

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

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

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

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

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

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

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

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

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

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

[19] Brian Kulis and Michael I Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. *arXiv preprint arXiv:1111.0352*, 2011.

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

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

[22] C E Rasmussen and C K I Williams. *Gaussian processes for machine learning*. 2006.

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

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

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

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

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

[28] Taishi Uchiya, Atsuyoshi Nakamura, and Mineichi Kudo. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*, pages 375–389. Springer, 2010.

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

Appendices

A Notation summary

Table 1: Notation list

Notation	Type	Meaning
σ_l^2, σ^2	scalar	lengthscale for RBF kernel, noise output variance (or measurement noise)
$\mathcal{X} \in \mathbb{R}^d$	search domain	continuous search space where d is the dimension
d	scalar	number of the continuous dimensions
c	scalar	number of categorical dimensions
$\mathbf{x}_t$	vector	a continuous selection by BO at iteration t
N_c	scalar	number of choices for categorical variable c
$\mathbf{h}_t = [h_{t,1}, ..., h_{t,c}]$	vector	vector of categorical variables
$\mathbf{z}_t = [\mathbf{x}_t, \mathbf{h}_t]$	vector	hyperparameter input including continuous and categorical variables
$\mathcal{D}_t$	set	observation set $D_t = \{z_i, y_i\}_{i=1}^t$

B Sequential CoCaBO Algorithm

Our proposed method, Continuous and Categorical Bayesian Optimisation, harnesses both the advantages of multi-armed bandits to select categorical inputs and the strength of GP-based BO in optimising continuous input spaces. The CoCaBO procedure is illustrated in Algorithm 2 and Figure 3.

Algorithm 2 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 $\mathbf{h}_t = [h_{1,t}, \ldots, h_{c,t}] \leftarrow$ EXP3($\{\mathbf{h}_i, f_i\}_i^{t-1}$)
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: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\mathbf{z}_t, f_t)$
8: **end for**

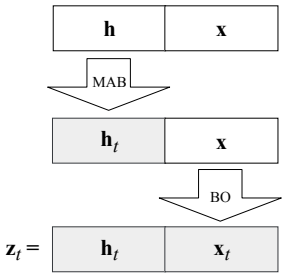


Figure 3: Optimisation procedure in CoCaBO

C Detailed description on the optimisation problems

C.1 Synthetic test functions

We generated several synthetic test functions: *Func-2C*, *Func-3C* and a *Ackley-cC* series, whose input spaces comprise both continuous variables and multiple categorical variables. Each of the categorical inputs in all three test functions have multiple values. *Func-2C* is a test problem with 2 continuous inputs ($d = 2$) and 2 categorical inputs ($c = 2$). The categorical inputs decide the linear combinations between three 2-dimensional global optimisation benchmark functions: beale (bea),

Table 2: Continuous and categorical input range of the synthetic test functions

Function f	Inputs $\mathbf{z} = [\mathbf{h}, \mathbf{x}]$	Input values
Func-2C ($d = 2, c = 2$)	h_1	$\{\text{ros}(\mathbf{x}), \text{cam}(\mathbf{x}), \text{bea}(\mathbf{x})\}$
	h_2	$\{+\text{ros}(\mathbf{x}), +\text{cam}(\mathbf{x}), +\text{bea}(\mathbf{x}), +\text{bea}(\mathbf{x}), +\text{bea}(\mathbf{x})\}$
	$\mathbf{x}$	$[-1, 1]^2$
Func-3C ($d = 2, c = 3$)	h_1	$\{\text{ros}(\mathbf{x}), \text{cam}(\mathbf{x}), \text{bea}(\mathbf{x})\}$
	h_2	$\{+\text{ros}(\mathbf{x}), +\text{cam}(\mathbf{x}), +\text{bea}(\mathbf{x}), +\text{bea}(\mathbf{x}), +\text{bea}(\mathbf{x})\}$
	h_3	$\{+5 \times \text{cam}(\mathbf{x}), +2 \times \text{ros}(\mathbf{x}), +2 \times \text{bea}(\mathbf{x}), +3 \times \text{bea}(\mathbf{x})\}$
	$\mathbf{x}$	$[-1, 1]^2$
Ackley-cC for $c = \{2, 3, 4, 5\}$ ($d = 1$, $N_i = 17$)	h_i for $i = 1, 2, \ldots, 5$	$\{z_i = -1 + 0.125 \times (j - 1),\ \text{for } j = 1, 2, \ldots, 17\}$
	$\mathbf{x}$	$[-1, 1]$

Table 3: Continuous and categorical input ranges of the real-world problems

Problems	Inputs $\mathbf{z} = [\mathbf{h}, \mathbf{x}]$	Input values
SVM-Boston ($d = 3, c = 3$)	kernel type h_1	{linear, poly, RBF, sigmoid}
	kernel coefficient h_2	{scale, auto }
	shrinking h_3	{shrinking on, shrinking off}
	penalty parameter x_1	$[0, 10]$
	tolerance for stopping x_2	$10^{[10^{-6}, 1]}$
	lower bound of the fraction of support vector x_3	$[0, 1]$
	learning rate x_1	$10^{[-5, -1]}$
	number of neurons x_2	$2^{[4, 7]}$
	aleatoric variance x_3	$[0.2, 0.8]$
XG-MNIST ($d = 5, c = 3$)	booster type h_1	{gbtree, dart}
	grow policies h_2	{depthwise, loss}
	training objective h_3	{softmax, softprob}
	learning rate x_1	$[0, 1]$
	maximum dept x_2	$[1, 2, \ldots, 10]$
	minimum split loss x_3	$[0, 10]$
	subsample x_4	$[0.001, 1]$
	regularisation x_5	$[0, 5]$

six-hump camel (cam) and rosenbrock (ros)[4]. *Func-3C* is similar to *Func-2C* but with 3 categorical inputs ($c = 3$) which leads to more complicated linear combinations among the three functions. We also generated a series of synthetic functions, *Ackley-cC*, with $c = \{2, 3, 4, 5\}$ categorical inputs and 1 continuous input ($d = 1$). Here, we convert c dimensions of the $c + 1$-dimensional Ackley function into 17 categories each. The value range for both continuous and categorical inputs of these functions are shown in Table 2.

C.2 Real-world problems

We defined three real-world tasks of tuning the hyperparameters for ML algorithms: *SVM-Boston* and *XG-MNIST*.

SVM-Boston outputs the negative mean square error of support vector machine (SVM) for regression on the test set of Boston housing dataset. We use the Nu Support Vector regression algorithm in the scikit-learn package [20] and use 30% of the data for testing.

[4]The analytic forms of these functions are available at `https://www.sfu.ca/~ssurjano/optimization.html`

XG-MNIST returns classification accuracy of a XGBoost algorithm [8] on the testing set of the MNIST dataset. We use the $xgboost$ package and adopt a stratified train/test split of $7:3$.

The hyperparameters over which we optimise for each above-mentioned ML task are summarised in Table 3. One point to note is that we present the unnormalised range for the continuous inputs in Table 3 but normalise all continuous inputs to $[-1, 1]$ for optimisation in our experiments. All the remaining hyperparameters are set to their default values.

D Additional experimental results

D.1 Predictive performance of the CoCaBO posterior

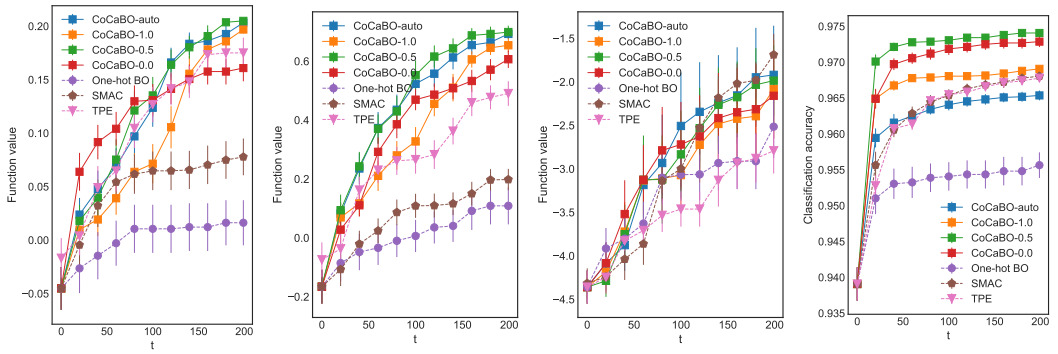
We first investigate the quality of the CoCaBO surrogate by comparing its modelling performance against a standard GP with one-hot encoding. We train each model on 250 randomly sampled data points and evaluate the predictive log likelihood on 100 test data points. The mean and standard error over 10 random initialisations are presented in Table 4. 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.

Table 4: 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. We see that the CoCaBO surrogate can model the function surface better than the One-hot surrogate as the number of categorical variables increases.

	Func-2C	*Func-3C*	*Ackley-2C*	*Ackley-3C*	*Ackley-4C*	*Ackley-5C*
CoCaBO	-531 $\pm$260	**-435** $\pm$85.7	**-74.7** $\pm$9.42	**-47.2** $\pm$9.20	**-28.3** $\pm$13.7	**23.5** $\pm$5.50
One-hot	**-254** $\pm$98.0	-748 $\pm$42.4	-77.9 $\pm$14.2	-73.4 $\pm$18.3	-59.8 $\pm$18.0	7.98 $\pm$12.5

D.2 Sequential results for synthetic functions

We evaluated the optimisation performance of our proposed CoCaBO methods and other existing methods on *Func-2C*, *Func-3C* and *Ackley-5C* in the sequential setting $b = 1$. The mean and standard error over 20 random repetitions are presented in Figure 4. It is evident that CoCaBO methods perform very competitively, if not better than, all other counterparts on these synthetic problems.



(a) *Func-2C* (b) *Func-3C* (c) *Ackley-5C* (d) *XG-MNIST*

Figure 4: Performance of CoCaBOs against existing methods on synthetic test functions over $T = 200$ in the sequential setting ($b = 1$).

D.3 Effectiveness of CoCaBO in optimising categorical variables

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

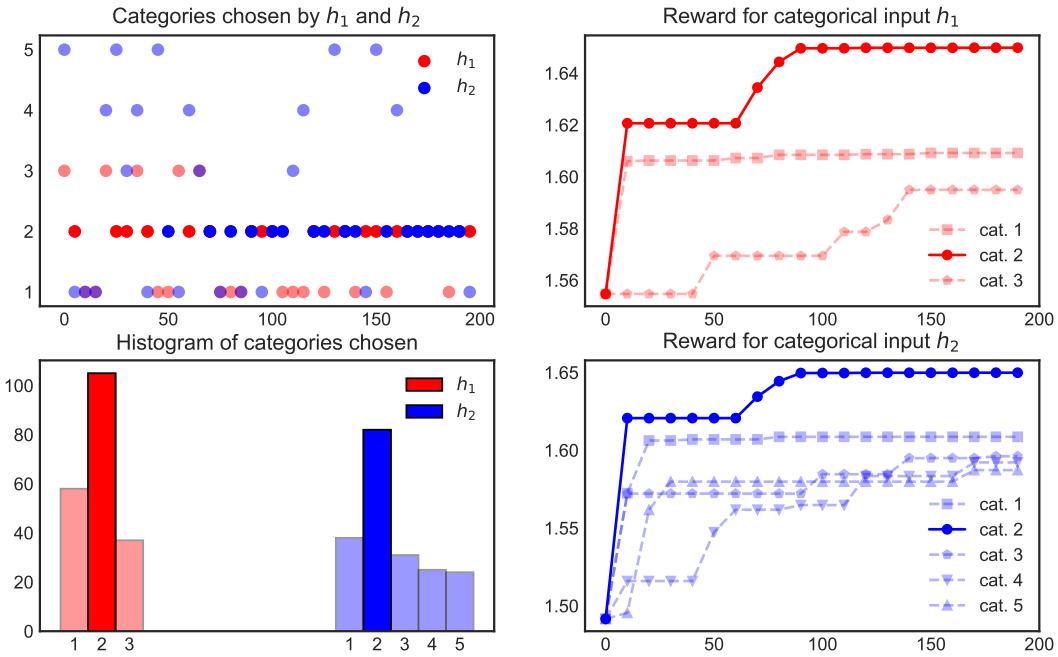


Figure 5: CoCaBO correctly optimises the two categorical inputs h_1 (Red) and h_2 (Blue) of the *Func-2C* test function over 200 iterations. The *best* category is $h_i = 2$ for both h_1 ($N_1 = 3$) and h_2 ($N_2 = 5$), and is highlighted in all plots. The top left plot shows the selections made by CoCaBO, showing how the both categorical inputs increasingly focus on the best categories as the algorithm progresses. The bottom left plot 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.

E Related work

E.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 the popular BO packages like Spearmint [26] and GPyOpt [13, 4].

There are two main drawbacks with this approach. First, the commonly-used RBF (squared exponential, radial basis function) and Matérn kernels in the GP surrogate assume that f is continuous and differentiable in the input space, which is clearly 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 the acquisition function is optimised as a continuous function. By using this extended representation, we are turning the optimisation into a significantly harder problem due to the increased dimensionality of the search space. Additionally, the one-hot encoding makes our

problem sparse, especially when we have multiple categories, each with multiple choices. This causes distances between inputs to become large, reducing the usefulness of the surrogate at such locations. As a result, the optimisation landscape is characterised by many flat regions, making it difficult to optimise [21].

E.2 Hierarchical approaches

Random forests (RFs) [6] can naturally consider continuous and categorical variables, and are used in SMAC [17] 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) [5] which is 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 density estimators, TPE also supports continuous and discrete spaces.

Another more recent approach is EXP3BO [14], 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.

F Categorical kernel relation with RBF

In this section we discuss the relationship between the categorical kernel we have proposed and a RBF kernel. Our categorical kernel is reproduced here for ease of access:

$$k_h(\mathbf{h}, \mathbf{h}') = \frac{\sigma^2}{N_c} \sum_{i=1}^{N_c} \mathbb{I}(h_i - h_i'). \tag{3}$$

Apart from the intuitive argument, that this kernel allows us to model the degree of similarity between two categorical selections, this kernel can also be derived as a special case of an RBF kernel. Consider the standard RBF kernel with unit variance evaluated between two scalar locations a and a':

$$k(a, a') = \exp\left(-\frac{1}{2}\frac{(a - a')^2}{l^2}\right). \tag{4}$$

The lengthscale in Eq. 4 allows us to define the similarity between the two inputs, and, as the lengthscale becomes smaller, the distance between locations that would be considered similar (i.e. high covariance) shrinks. The limiting case $l \to 0$ states that if two inputs are not exactly the same as each other, then they provide no information for inferring the GP posterior's value at each other's locations. This causes the kernel to turn into an indicator function as in Eq. 3 above [19]:

$$k(a, a') = \begin{cases} 1, & \text{if } a = a' \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

By adding one such RBF kernel with $l \to 0$ for each categorical variable in h and normalising the output we arrive at the form in Eq. 3.

G Bayesian optimisation and EXP3

Bayesian optimisation [7, 25] 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 [22] surrogate to model the objective f. A GP defines a probability distribution over functions f, as $f(\mathbf{x}) \sim \texttt{GP}\left(m\left(\mathbf{x}\right), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$, where $m\left(\mathbf{x}\right)$ and $k\left(\mathbf{x}, \mathbf{x}'\right)$ are the mean and covariance functions respectively, which encode our prior beliefs about f. Using the GP posterior, BO defines an acquisition function $\alpha_t\left(\mathbf{x}\right)$ 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.

EXP3 is a method proposed in [3] to deal with the non-stochastic, adversarial multi-arm bandit problem which is a more general setting and can model any form of non-stationarity [1]. In such problem setting, the rewards are chosen by an adversary at each iteration. EXP3.M is a batch version of EXP3 proposed by [28] which permits multiple arms/actions to be selected at each iteration. In our paper, we adopted EXP3 in sequential CoCaBO and EXP3.M in batch CoCaBO.