

Bayesian Optimization Meets Bayesian Optimal Stopping

Anonymous Authors¹

Abstract

Bayesian optimization (BO) is a popular paradigm for optimizing the hyperparameters of *machine learning* (ML) models due to its sample efficiency. Many ML models require running an iterative training procedure (e.g., stochastic gradient descent). This motivates the question whether information available during the training process (e.g., validation accuracy after each epoch) can be exploited for improving the epoch efficiency of BO algorithms by early-stopping model training under hyperparameter settings that will end up under-performing and hence eliminating unnecessary training epochs. This paper proposes to unify BO (specifically, *Gaussian process-upper confidence bound* (GP-UCB)) with *Bayesian optimal stopping* (BO-BOS) to boost the epoch efficiency of BO. To achieve this, while GP-UCB is sample-efficient in the number of function evaluations, BOS complements it with epoch efficiency for each function evaluation by providing a principled optimal stopping mechanism for early stopping. BO-BOS preserves the (asymptotic) no-regret performance of GP-UCB using our specified choice of BOS parameters that is amenable to an elegant interpretation in terms of the exploration-exploitation trade-off. We empirically evaluate the performance of BO-BOS and demonstrate its generality in hyperparameter optimization of ML models and two other interesting applications.

1. Introduction

The state-of-the-art *machine learning* (ML) models have recently reached an unprecedented level of predictive performance in several applications such as image recognition, complex board games, among others (LeCun et al., 2015; Silver et al., 2016). However, a major difficulty faced by ML practitioners is the choice of model hyperparameters which

significantly impacts the predictive performance. This calls for the need to develop hyperparameter optimization algorithms that have to be sample-efficient since the training of many modern ML models consumes massive computational resources. To this end, *Bayesian optimization* (BO) is a popular paradigm due to its high sample efficiency and strong theoretical performance guarantee (Shahriari et al., 2016). In particular, the BO algorithm based on the *Gaussian process-upper confidence bound* (GP-UCB) acquisition function has been shown to achieve no regret asymptotically and perform competitively in practice (Srinivas et al., 2010).

Many ML models require running an iterative training procedure for some number of epochs such as stochastic gradient descent for neural networks (LeCun et al., 2015) and boosting procedure for gradient boosting machines (Friedman, 2001). During BO, any query of a hyperparameter setting usually involves training the ML model for a fixed number of epochs. Information typically available during the training process (e.g., validation accuracy after each epoch) is rarely exploited for improving the epoch efficiency of BO algorithms, specifically, by early-stopping model training under hyperparameter settings that will end up under-performing, hence eliminating unnecessary training epochs.

To address this challenging issue, a number of works have been proposed to make BO more epoch-efficient: Freeze-thaw BO (Swersky et al., 2014) explores a diverse collection of hyperparameter settings in the initial stage by training their ML models with a small number of epochs, and then gradually focuses on (exploits) a small number of promising settings. Despite its promising epoch efficiency, its performance is not theoretically guaranteed and its computational cost can be excessive. Multi-fidelity BO (Kandasamy et al., 2016; 2017) reduces the resource consumption of BO by utilizing low-fidelity functions which can be obtained by using a subset of the training data or by training the ML model for a small number of epochs. However, in each BO iteration, since the fidelity (e.g., number of epochs) is determined before function evaluation, it is not influenced by information that is typically available during the training process (e.g., validation accuracy after each epoch). In addition to BO, attempts have also been made to improve the epoch efficiency of other hyperparameter optimization algorithms: Some heuristic methods (Baker et al., 2017; Domhan et al., 2015; Klein et al., 2017) have predicted the final training out-

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

come based on partially trained learning curves in order to identify hyperparameter settings that are predicted to underperform and early-stop their model training. Hyperband (Li et al., 2017), which dynamically allocates the computational resource (e.g., training epochs) through random sampling and eliminates under-performing hyperparameter settings by successive halving, has been proposed and shown to perform well in practice. Both the learning curve prediction methods and Hyperband can be combined with BO to further improve the epoch efficiency (Domhan et al., 2015; Falkner et al., 2018; Klein et al., 2017), but their resulting performances are not theoretically guaranteed. Despite these recent advances, we still lack an epoch-efficient algorithm that can incorporate early stopping into BO (i.e., by exploiting information available during the training process) and yet offer a theoretical performance guarantee, the design of which is likely to require a principled decision-making mechanism for determining the optimal stopping time.

Optimal stopping is a classic research topic in statistics and operations research regarding sequential decision-making problems whose objective is to make the optimal stopping decision with a small number of observations (Ferguson, 2006). In *Bayesian optimal stopping* (BOS) or Bayesian sequential design, the decision between stopping vs. continuing is made to maximize the expected utility or, equivalently, minimize the expected loss (Powell & Ryzhov, 2012). BOS has found success in application domains such as finance (Longstaff & Schwartz, 2001), clinical design (Brockwell & Kadane, 2003; Müller et al., 2007; Wathen & Thall, 2008), and economics (Davis & Cairns, 2012). The capability of BOS in providing a principled optimal stopping mechanism makes it a prime candidate for introducing early stopping into BO in a theoretically sound and rigorous way.

This paper proposes to unify *Bayesian optimization* (specifically, GP-UCB) with *Bayesian optimal stopping* (BO-BOS) to boost the epoch efficiency of BO (Section 3). Intuitively, GP-UCB is acclaimed for being sample-efficient in the number of function evaluations while BOS can reduce the required number of epochs for each function evaluation. BO-BOS unifies the best of both worlds to yield an epoch-efficient hyperparameter optimization algorithm. Interestingly, in spite of the seemingly disparate optimization objectives of GP-UCB vs. BOS (respectively, objective function v.s. expected loss), BO-BOS can preserve the trademark (asymptotic) no-regret performance of GP-UCB with our specified choice of BOS parameters that is amenable to an elegant interpretation in terms of the exploration-exploitation trade-off (Section 4). Though the focus of our work here is on epoch-efficient BO for hyperparameter tuning, we additionally evaluate the performance of BO-BOS empirically in two other interesting applications to demonstrate its generality: policy search for reinforcement learning, and joint hyperparameter tuning and feature selection (Section 5).

2. Background and Notations

2.1. Bayesian Optimization (BO) and GP-UCB

Consider the problem of sequentially optimizing an unknown objective function $f : \mathcal{D} \rightarrow \mathbb{R}$ representing the validation accuracy over a compact input domain $\mathcal{D} \subseteq \mathbb{R}^d$ of different hyperparameter settings for training an ML model: In each iteration $t = 1, \dots, T$, an input query $\mathbf{z}_t \triangleq [\mathbf{x}_t, n_t]$ of a hyperparameter setting (comprising $N_0 < n_t \leq N$ training epochs and a vector \mathbf{x}_t of the other hyperparameters) is selected for evaluating the validation accuracy f of the ML model to yield a noisy observed output (validation accuracy) $y_t \triangleq f(\mathbf{z}_t) + \epsilon$ with i.i.d. Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and noise variance σ^2 . Since every evaluation of f is costly (Section 1), our goal is to strategically select input queries to approach the global maximizer $\mathbf{z}^* \triangleq \arg \max_{\mathbf{z} \in \mathcal{D}} f(\mathbf{z})$ as rapidly as possible. This can be achieved by minimizing a standard BO objective such as the *simple regret* $S_T \triangleq f(\mathbf{z}^*) - \max_{t \in \{1, \dots, T\}} f(\mathbf{z}_t)$. A BO algorithm is said to guarantee *no regret* asymptotically if it satisfies $\lim_{T \rightarrow \infty} S_T = 0$, thus implying that it will eventually converge to the global maximum.

To guarantee no regret, the belief of f is modeled by a *Gaussian process* (GP). Let $\{f(\mathbf{z})\}_{\mathbf{z} \in \mathcal{D}}$ denote a GP, that is, every finite subset of $\{f(\mathbf{z})\}_{\mathbf{z} \in \mathcal{D}}$ follows a multivariate Gaussian distribution (Rasmussen & Williams, 2006). Then, a GP is fully specified by its *prior* mean $\mu(\mathbf{z})$ and covariance $k(\mathbf{z}, \mathbf{z}')$ for all $\mathbf{z}, \mathbf{z}' \in \mathcal{D}$, which are, respectively, assumed w.l.o.g. to be $\mu(\mathbf{z}) = 0$ and $k(\mathbf{z}, \mathbf{z}') \leq 1$ for notational simplicity. Given a column vector $\mathbf{y}_T \triangleq [y_t]_{t=1, \dots, T}^\top$ of noisy outputs observed from evaluating f at the selected input queries $\mathbf{z}_1, \dots, \mathbf{z}_T \in \mathcal{D}$ after T iterations, the GP posterior belief of f at some input $\mathbf{z} \in \mathcal{D}$ is a Gaussian with the following *posterior* mean $\mu_T(\mathbf{z})$ and variance $\sigma_T^2(\mathbf{z})$:

$$\begin{aligned} \mu_T(\mathbf{z}) &\triangleq \mathbf{k}_T(\mathbf{z})^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_T, \\ \sigma_T^2(\mathbf{z}) &\triangleq k(\mathbf{z}, \mathbf{z}) - \mathbf{k}_T(\mathbf{z})^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_T(\mathbf{z}) \end{aligned} \quad (1)$$

where $\mathbf{K}_T \triangleq [k(\mathbf{z}_t, \mathbf{z}_{t'})]_{t, t'=1, \dots, T}$ and $\mathbf{k}_T(\mathbf{z}) \triangleq [k(\mathbf{z}_t, \mathbf{z})]_{t=1, \dots, T}^\top$. In each iteration t of BO, an input query $\mathbf{z}_t \in \mathcal{D}$ is selected to maximize the GP-UCB acquisition function (Srinivas et al., 2010) that trades off between observing an expected maximum (i.e., with a large GP posterior mean $\mu_{t-1}(\mathbf{z})$) given the current GP belief of f (i.e., exploitation) vs. that of high predictive uncertainty (i.e., with a large GP posterior variance $\sigma_{t-1}^2(\mathbf{z})$) to improve the GP belief of f over \mathcal{D} (i.e., exploration). That is, $\mathbf{z}_t \triangleq \arg \max_{\mathbf{z} \in \mathcal{D}} \mu_{t-1}(\mathbf{z}) + \sqrt{\beta_t} \sigma_{t-1}(\mathbf{z})$ where the parameter $\beta_t > 0$ is set to trade off between exploitation vs. exploration for guaranteeing no regret asymptotically with high probability.

2.2. Bayesian Optimal Stopping (BOS)

BOS provides a principled mechanism for making the Bayes-optimal stopping decision with a small number of

observations. As shall be seen in Algorithm 1, in each iteration t of BO-BOS, BOS is used to early-stop model training under the selected input hyperparameters \mathbf{x}_t that will end up under-performing, hence reducing the required number of training epochs. In a BOS problem, the goal is to decide whether to stop and conclude either hypothesis/event $\theta_t = \theta_{t,1}$ or $\theta_t = \theta_{t,2}$ corresponding to terminal decision d_1 or d_2 , or to gather one more observation via the continuation decision d_0 . Let $y_{t,n'}$ be the noisy output (validation accuracy) observed in epoch n' and $\mathbf{y}_{t,n} \triangleq [y_{t,n'}]_{n'=1,\dots,n}^\top$ be a vector of noisy outputs observed up till epoch n in iteration t . Recall from Section 2.1 that in iteration t , the ML model is trained using the selected input hyperparameter setting $[\mathbf{x}_t, n_t]$ to yield the noisy observed output (validation accuracy) y_t . So, $y_t = y_{t,n_t}$ for $t = 1, \dots, T$. After each epoch n , the posterior belief of event θ_t is updated to $\Pr(\theta_t | \mathbf{y}_{t,n})$ which will be used to compute the expected losses of terminal decisions d_1 and d_2 . Such a loss function l has to encode the cost of making a wrong decision. Define $\rho_{t,n}(\mathbf{y}_{t,n})$ as the minimum expected loss among all decisions in epoch n :

$$\rho_{t,n}(\mathbf{y}_{t,n}) \triangleq \min\{ \mathbb{E}_{\theta_t | \mathbf{y}_{t,n}}[l(d_1, \theta_t)], \mathbb{E}_{\theta_t | \mathbf{y}_{t,n}}[l(d_2, \theta_t)], c_{d_0} + \mathbb{E}_{y_{t,n+1} | \mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n+1})] \} \quad (2)$$

for $n = 1, \dots, N - 1$ where the first two terms are the expected losses of terminal decisions d_1 and d_2 , the last term sums the immediate cost c_{d_0} and expected future loss of making the continuation decision d_0 to continue model training in the next epoch $n + 1$ to yield the noisy observed output (validation accuracy) $y_{t,n+1}$, and $\rho_{t,N}(\mathbf{y}_{t,N}) \triangleq \min\{ \mathbb{E}_{\theta_t | \mathbf{y}_{t,N}}[l(d_1, \theta_t)], \mathbb{E}_{\theta_t | \mathbf{y}_{t,N}}[l(d_2, \theta_t)] \}$. Since $\rho_{t,n}$ depends on $\rho_{t,n+1}$, it naturally prompts the use of backward induction to solve the BOS problem (2) exactly, which is unfortunately intractable due to an uncountable set of possible observed outputs $y_{t,n+1}$. This computational difficulty can be overcome using approximate backward induction techniques (Brockwell & Kadane, 2003; Müller et al., 2007) whose main ideas include using summary statistics to represent the posterior belief, discretizing the space of summary statistics, and estimating the expectation terms via sampling. Appendix A describes a commonly-used approximate backward induction algorithm (Müller et al., 2007).

Solving the BOS problem (2) yields a Bayes-optimal decision rule in each epoch n : Take the Bayes-optimal stopping decision if the expected loss of either terminal decision d_1 or d_2 is at most that of the continuation decision d_0 , that is, $\min\{ \mathbb{E}_{\theta_t | \mathbf{y}_{t,n}}[l(d_1, \theta_t)], \mathbb{E}_{\theta_t | \mathbf{y}_{t,n}}[l(d_2, \theta_t)] \} \leq c_{d_0} + \mathbb{E}_{y_{t,n+1} | \mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n+1})]$. Otherwise, continue model training to yield the noisy observed output (validation accuracy) $y_{t,n+1}$ and repeat this rule in the next epoch $n + 1$.

3. BO-BOS Algorithm

In this section, we will describe our proposed BO-BOS algorithm (Section 3.1) and define the loss function l in BOS (2)

such that it can serve as an effective early-stopping mechanism in BO (Section 3.2). We focus on problem settings where the objective function f is bounded and monotonically increasing in n :

Assumption 1. (a) $f(\mathbf{z}) \in [0, 1]$ for all $\mathbf{z} \in \mathcal{D}$ and (b) $f([\mathbf{x}, n]) \leq f([\mathbf{x}, n + 1])$ for all \mathbf{x} and $n = 1, \dots, N - 1$.

Assumption 1a is not restrictive since it applies to any bounded f with a proper transformation. Assumption 1b holds reasonably well in a number of important ML problems: (a) f represents the validation accuracy of ML models and n denotes the number of training epochs or the number of selected features during feature selection, and (b) f represents the (discounted) cumulative rewards in reinforcement learning (assuming non-negative rewards) and n denotes the number of steps taken by the agent in the environment. Our experiments in Section 5 will demonstrate that BO-BOS outperforms the state-of-the-art hyperparameter optimization algorithms in these ML problems.

3.1. Algorithm Description

In each iteration t of BO-BOS (Algorithm 1), the input hyperparameters \mathbf{x}_t are selected to maximize the GP-UCB acquisition function with the input dimension of training epochs fixed at N (line 2). The ML model is trained using \mathbf{x}_t for N_0 initial training epochs to yield the noisy observed outputs (validation accuracies) \mathbf{y}_{t,N_0} (line 3). After that, the BOS problem is solved (line 5) to obtain Bayes-optimal decision rules (see Sections 2.2 and 3.2). Then, in each epoch $n > N_0$, model training continues under \mathbf{x}_t to yield the noisy observed output (validation accuracy) $y_{t,n}$ (line 8). If both of the following conditions are satisfied (line 9),

C1. BOS decision rule in epoch n outputs stopping decision;

C2. $\sigma_{t-1}([\mathbf{x}_t, n]) \geq \sigma_{t-1}([\mathbf{x}_t, N]) / \kappa$,

then model training is early-stopped in epoch $n_t = n$. Otherwise, the above procedure is repeated in epoch $n + 1$. If none of the training epochs $n = N_0 + 1, \dots, N - 1$ satisfy both C1 and C2, then $n_t = N$ (i.e., no early stopping). Finally, the GP posterior belief (1) is updated with the selected input hyperparameter setting $\mathbf{z}_t = [\mathbf{x}_t, n_t]$ and the corresponding noisy observed output (validation accuracy) $y_t = y_{t,n_t}$ (line 11). BO-BOS then proceeds to the next iteration $t + 1$.

To understand the rationale of our choices of C1 and C2, the BOS decision rule in C1 recommends the stopping decision to early-stop model training in epoch n if it concludes that model training under \mathbf{x}_t for N epochs will produce a validation accuracy not more than the currently found maximum in iterations $1, \dots, t - 1$; this will be formally described in Section 3.2. On the other hand, C2 prefers to evaluate the validation accuracy f of the ML model with the input query $[\mathbf{x}_t, n]$ of fewer training epochs $n < N$ than $[\mathbf{x}_t, N]$ if the predictive uncertainty of $f([\mathbf{x}_t, n])$ is higher

Algorithm 1 BO-BOS

```

1: for  $t = 1, 2, \dots, T$  do
2:    $\mathbf{x}_t \leftarrow \arg \max_{\mathbf{x}} \mu_{t-1}([\mathbf{x}, N]) + \sqrt{\beta_t} \sigma_{t-1}([\mathbf{x}, N])$ 
3:   Train model using  $\mathbf{x}_t$  for  $N_0$  epochs to yield  $\mathbf{y}_{t, N_0}$ 
4:    $n \leftarrow N_0$ 
5:   Solve BOS problem (2) to obtain decision rules
6:   repeat
7:      $n \leftarrow n + 1$ 
8:     Continue model training using  $\mathbf{x}_t$  to yield  $\mathbf{y}_{t, n}$ 
9:   until  $(n = N) \vee (C1 \wedge C2)$ 
10:   $n_t = n$ 
11:  Update GP posterior belief (1) with  $\mathbf{z}_t = [\mathbf{x}_t, n_t]$  and
     $\mathbf{y}_t = \mathbf{y}_{t, n_t}$ 

```

than that of $f([\mathbf{x}_t, N])$ being scaled down by a factor of $\kappa \geq 1$, hence better improving the GP belief of f (i.e., exploration) along the input dimension of training epochs (i.e., $\{[\mathbf{x}_t, n']\}_{n'=1, \dots, N}$); the degree of preference/exploration is controlled by parameter κ . The requirement to satisfy both C1 and C2 thus reflects two complementary principles of early stopping in BO: (a) exploiting BOS’s capability to early-stop model training under hyperparameter settings that will end up under-performing and (b) exploring to improve the GP belief along the input dimension of training epochs. Both C1 and C2 are necessary for theoretically guaranteeing the no-regret performance of BO-BOS (Section 4).

3.2. BOS for Early Stopping in BO

In iteration t , let the currently found maximum or incumbent value be denoted as $y_{t-1}^* \triangleq \max_{t' \in \{1, \dots, t-1\}} y_{t'}$. In the context of early stopping in BO, BOS has to determine whether model training under \mathbf{x}_t for N epochs will produce a validation accuracy not more than the incumbent value, i.e., $f([\mathbf{x}_t, N]) \leq y_{t-1}^* - \xi_t$ given the noisy outputs (validation accuracies) $\mathbf{y}_{t, n}$ observed up till epoch $n < N$ where $y_0^* \triangleq 0$ and ξ_t is defined later in Theorem 1. To achieve this, we model it with the following three decisions: (a) d_1 : stop and conclude that $f([\mathbf{x}_t, N]) \leq y_{t-1}^* - \xi_t$, (b) d_2 : stop and conclude that $f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t$, and (c) d_0 : continue to train for one more epoch. Based on this setting, the event θ (Section 2.2) can be defined as

$$\theta_t \triangleq \begin{cases} \theta_{t,1} & \text{if } f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t, \\ \theta_{t,2} & \text{otherwise;} \end{cases}$$

about which we are uncertain due to the uncertainty regarding the value of $f([\mathbf{x}_t, N])$ when $n < N$, and the objective of BOS is to determine whether $\theta_t = \theta_{t,1}$ or $\theta_t = \theta_{t,2}$. We adopt the 0 – K loss function commonly used in clinical designs (Lewis & Berry, 1994; Jiang et al., 2013) due to its simplicity and interpretability:

$$l(d_1, \theta_t) \triangleq K_1 \mathbb{1}_{\theta_t = \theta_{t,1}}, \quad l(d_2, \theta_t) \triangleq K_2 \mathbb{1}_{\theta_t = \theta_{t,2}}, \quad (3)$$

in which the cost parameters $K_1 > 0$ and $K_2 > 0$ represent how much we would like to penalize falsely making the corresponding terminal decisions. As a result, the expected losses of the terminal decisions, d_1 and d_2 can be calculated:

$$\begin{aligned} \mathbb{E}_{\theta_t | \mathbf{y}_{t, n}} [l(d_1, \theta_t)] &= K_1 \Pr(\theta_t = \theta_{t,1} | \mathbf{y}_{t, n}), \\ \mathbb{E}_{\theta_t | \mathbf{y}_{t, n}} [l(d_2, \theta_t)] &= K_2 \Pr(\theta_t = \theta_{t,2} | \mathbf{y}_{t, n}). \end{aligned} \quad (4)$$

Note that if a cost parameter is set to $+\infty$, the corresponding expected loss is taken as $+\infty$ and the corresponding terminal decision is never taken; e.g., if $K_1 = +\infty$, decision d_1 is never selected. Based on these definitions, the expected loss at step n is defined as the minimum expected losses among all decisions in the same way as (2):

With this formulation, the resulting BOS problem can be solved with an adapted approximate backward induction algorithm. To take into account Assumption 1b, we use a kernel with prior bias towards exponentially decaying learning curves (Swersky et al., 2014) to fit a GP model on the initial N_0 validation accuracies and draw samples from the resulting GP model, which are then plugged into approximate backward induction as the forward simulation samples mainly for the estimation of $\Pr(\theta_t = \theta_{t,1} | \mathbf{y}_{t, n})$ and $\Pr(\mathbf{y}_{t, n+1} | \mathbf{y}_{t, n})$. Following some applications of BOS (Müller et al., 2007; Jiang et al., 2013), we use the average validation accuracies as the summary statistics. A detailed description of the approximate backward induction algorithm is presented in Appendix B. After the BOS problem is solved, a set of decision rules are obtained and used by C1 in the BO-BOS algorithm. Specifically, after the n -th validation accuracy is observed, we first calculate the updated summary statistics $\sum_{n'=1}^n y_{t, n'} / n$, and then take the corresponding optimal decision suggested by the BOS decision rules. If the suggested decision is d_1 (i.e., the optimal decision now is to stop the current experiment since we believe \mathbf{x}_t will end up having smaller validation accuracy than the incumbent), we immediately terminate the training of \mathbf{x}_t (assuming C2 is satisfied); otherwise, we train for one more epoch, and repeat the process until $n = N$. Note that taking decision d_2 (i.e., early-stopping because we believe \mathbf{x}_t will perform better than the incumbent) does not align with the objective of BO, which is to sequentially maximize the objective function. Therefore, when decision d_2 is suggested by BOS, to conform to the decision-making principle of BO, we do not early-stop the experiment and instead continue the training.

4. Theoretical Analysis

The goal of the theoretical analysis is to characterize the growth of the *simple regret* S_T of the proposed BO-BOS algorithm and thus show how the algorithm should be designed in order for S_T to asymptotically go to 0, i.e., for the algorithm to be *no regret*. To account for the addi-

tional uncertainty introduced by BOS, we analyze the expected regret, in which the expectation is taken with respect to the posterior probabilities from the BOS algorithms: $\Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t, n_t})$.

We make the following assumption on the smoothness of the objective function f :

Assumption 2. Assume for the kernel k , for some a and b ,

$$\Pr(\sup_{\mathbf{z} \in \mathcal{D}} |\partial f / \partial z_j| > L) \leq a \exp(-(L/b)^2)$$

for $j = 1, \dots, d$ where z_j is the j -th component of input \mathbf{z} .

Assumption 2 is satisfied by some commonly used kernels such as the Square Exponential kernel and Matérn kernel with $\nu > 2$ (Srinivas et al., 2010). For simplicity, we assume that the underlying domain \mathcal{D} is discrete, i.e. $|\mathcal{D}| < \infty$. However, it is straightforward to extend the analysis to general compact domain by following similar analysis strategies as those in Appendix A.2. of (Srinivas et al., 2010). Theorem 1 below shows an upper bound on the expected simple regret of the BO-BOS algorithm.

Theorem 1. Suppose that Assumptions 1 and 2 hold. Let $\delta, \delta', \delta'' \in (0, 1)$, $\beta_t \triangleq 2 \log(|\mathcal{D}| t^2 \pi^2 / (6\delta))$, and $\tau_T \triangleq \sum_{t=1}^T \mathbb{1}_{n_t < N}$ be the number of BO iterations in which early stopping happens from iterations 1 to T . Let κ be the parameter used in C2. At iteration t , the BOS algorithm is run with the corresponding fixed cost parameters K_2 and c_{d_0} , as well as iteration-dependent cost parameter $K_{1,t}$, $\xi_1 \triangleq 0$, and $\xi_t \triangleq \sqrt{2\sigma^2 \log(\pi^2 t^2 (t-1) / (6\delta''))}$ for $t > 1$. Then $\forall T \geq 1$, with probability of at least $1 - \delta - \delta' - \delta''$,

$$\mathbb{E}[S_T] \leq \frac{\kappa \sqrt{TC_1 \beta_T \gamma_T}}{T} + \frac{\sum_{t=1}^T \eta_t}{T} + \frac{1}{T} N b \sqrt{\log \frac{da}{\delta'}} \tau_T$$

in which $\eta_t \triangleq \frac{K_2 + c_{d_0}}{K_{1,t}}$, $C_1 = 8 / \log(1 + \sigma^{-2})$, γ_T is the maximum information gain about the function f from any set of observations of size T , and the expectation is w.r.t. $\prod_{t \in \{t' | t'=1, \dots, T, n_{t'} < N\}} \Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t, n_t})$ used in the BOS algorithm.

Theorem 2 below states how the BOS parameters should be chosen to make BO-BOS asymptotically no-regret.

Theorem 2. In Theorem 1, if $K_{1,t}$ is an increasing sequence such that $K_{1,1} \geq K_2 + c_{d_0}$ and that $K_{1,t}$ goes to $+\infty$ in finite number of BO iterations, then, with probability of at least $1 - \delta - \delta' - \delta''$, $\mathbb{E}[S_T]$ goes to 0 asymptotically.

The proof of both theorems is presented in Appendix C. The first term in the upper bound of $\mathbb{E}[S_T]$ in Theorem 1 matches that of the simple regret of the GP-UCB algorithm (up to the constant κ). Note that the theoretical results rely on the exact solution of the BOS problems; however, in practice, a trade-off exists between the quality of the approximate backward induction and the computational efficiency.

In particular, increasing the number of forward simulation samples and making the grid of summary statistics more fine-grained both lead to better approximation quality, while increasing the computational cost. Recommended approximation parameters that work well in all our experiments and thus strike a reasonable balance between these two aspects are given in Section 5.

Interestingly, the choice of an increasing $K_{1,t}$ sequence as required by Theorem 2 is well justified in terms of the exploration-exploitation trade-off. As introduced in section 3.2, K_1 represents how much we would like to penalize the BOS algorithm for falsely early-stopping (taking decision d_1). Therefore, increasing values of K_1 implies that, as the BO-BOS algorithm progresses, we become more and more cautious at early-stopping. In other words, the preference of BOS for early stopping diminishes over BO iterations. Interestingly, this corresponds to sequentially shifting our preference from exploration (using small number of epochs) to exploitation (using large number of epochs) throughout all runs of the BOS algorithms, which is an important decision-making principle followed by many sequential decision-making algorithms such as BO, multi-armed bandit, reinforcement learning, among others.

Another intriguing interpretation of the theoretical results is that the growth rate of the $K_{1,t}$ sequence implicitly determines the trade-off between faster convergence of the BO algorithm (smaller number of BO iterations) and more computational saving in each BO iteration (smaller number of training epochs on average). In particular, if $K_{1,t}$ grows quickly, the second and third terms in the upper bound in Theorem 1 both decay fast, since η_t is inversely related to $K_{1,t}$ and large penalty for early stopping results in small τ_T ; as a result, a large number of hyperparameters are run with N epochs and the resulted BO-BOS algorithm behaves similarly to GP-UCB, which is corroborated by the upper bound on $\mathbb{E}[S_T]$ in Theorem 1 since the first term dominates. On the other hand, if the $K_{1,t}$ sequence grows slowly, then the second and third terms in Theorem 1 decay slowly; consequently, these two terms dominate the regret and the resulted algorithm early-stops very often, thus leading to smaller number of epochs on average, at the potential expense of requiring more BO iterations. Furthermore, the constant κ used in C2 also implicitly encodes our relative preference for early-stopping. Specifically, large values of κ favor early stopping by relaxing C2: $\sigma_{t-1}([\mathbf{x}_t, n_t]) \geq \sigma_{t-1}([\mathbf{x}_t, N]) / \kappa$; however, more early-stopped function evaluations might incur larger number of required BO iterations as can be verified by the fact that larger κ increases the first regret term in Theorem 1 (which matches the regret of GP-UCB). In practice, as a result of the above-mentioned trade-offs, the best choices of the BOS parameters and κ are application-dependent. In addition, to avoid abnormal behaviors of the BOS algorithms, the BOS parameters should be chosen with

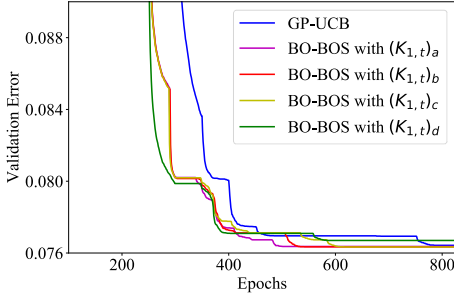


Figure 1. Best-found validation error of logistic regression v.s. the total number of epochs (averaged over 10 random initializations). $K_2 = 99$ and $c_{d_0} = 1$ are fixed; $K_{1,1} = 100$ for all four BO-BOS algorithms; for $t > 1$, the different $K_{1,t}$ sequences are: $(K_{1,t})_a = \frac{K_{1,t-1}}{0.89}$; $(K_{1,t})_b = \frac{K_{1,t-1}}{0.95}$; $(K_{1,t})_c = \frac{K_{1,t-1}}{0.99}$; $(K_{1,t})_d = \frac{K_{1,t-1}}{1.0} = K_{1,1}$.

additional care. In particular, c_{d_0} should be small, whereas K_2 and the K_1 's should be chosen to be of similar order to ensure that the penalties BOS gives to different false decisions don't deviate excessively. In Section 5, we recommend some parameters that work well in all our experiments and thus are believed to perform robustly in practice.

5. Experiments and Discussion

The performance of BO-BOS is empirically compared with that of four other hyperparameter optimization algorithms: GP-UCB (Srinivas et al., 2010), Hyperband (Li et al., 2017), multi-fidelity BO algorithm called *BO with continuous approximations* (BOCA) (Kandasamy et al., 2017), and GP-UCB with learning curve prediction using an ensemble of Bayesian parametric regression models (LC Prediction) (Domhan et al., 2015). Freeze-thaw BO is not included in the comparison here since its implementation details are complicated and not fully available. We empirically evaluate the performance of BO-BOS in hyperparameter optimization of *logistic regression* (LR) and *convolutional neural networks* (CNN), respectively, in Sections 5.1 and 5.2, and demonstrate its generality in two other interesting applications in Section 5.3. Due to lack of space, additional details of the experimental setup are deferred to Appendix D.

5.1. Hyperparameter Optimization of LR

We first tune three hyperparameters of LR trained on the MNIST image dataset. Although both the $K_{1,t}$ sequence and κ determine the trade-off between the number of BO iterations and the number of epochs on average as mentioned in section 4, for simplicity, we fix $\kappa = 2$ and investigate the impact of different sequences of $K_{1,t}$ values.

As shown in Fig. 1, the sequences $(K_{1,t})_a$, $(K_{1,t})_b$ and

$(K_{1,t})_c$ lead to similar performances, all of which outperform GP-UCB. On the other hand, the algorithm with fixed K_1 values (the $(K_{1,t})_d$ sequence), despite having fast performance improvement initially, eventually finds a worse hyperparameter setting than all other algorithms. This undesirable performance results from the fact that fixed K_1 values give constant penalty to falsely-early stopping throughout all runs of the BOS algorithms, and as the incumbent validation error decreases, the preference of the algorithm for early stopping will increase, thus preventing the resulting algorithm from beginning to exploit (running promising hyperparameter settings with N epochs). This observation demonstrates the necessity of having an increasing sequence of $K_{1,t}$ values, thus substantiating the practical relevance of our theoretical analysis (Theorem 2). The sequence $(K_{1,t})_b$, as well as the values of $K_2 = 99$ and $c_{d_0} = 1$, will be used in the following experiments if not further specified.

5.2. Hyperparameter Optimization of CNN

In this section, we tune six hyperparameters of CNN using two image datasets: CIFAR-10 (Krizhevsky, 2009) and Street View House Numbers (SVHN) (Netzer et al., 2011). Note that the goal of the experiments is not to compete with the state-of-the-art models, but to compare the efficiency of different hyperparameter tuning algorithms, so no data augmentation or advanced network architectures are used.

As shown in Figures 2a and 2b, BO-BOS outperforms all other algorithms under comparison in terms of the run-time efficiency. Note that the horizontal axis, which represents the wall-clock time, includes all the time spent during the algorithms, including the running time of machine learning models, the approximate backward induction used to solve BOS, etc. Although Hyperband is able to quickly reduce the validation error in the initial stage, it eventually converges to sub-optimal hyperparameter settings compared with both GP-UCB and BO-BOS. Similar findings have been reported in previous works (Klein et al., 2017; Falkner et al., 2018) and they might be attributed to the pure-exploration nature of the algorithm. Our BO-BOS algorithm, which trades off exploration and exploitation, is able to quickly surpass the performance of Hyperband and eventually converges to significantly better hyperparameter settings. In addition, we also ran the BOHB algorithm (Falkner et al., 2018) which combines Hyperband and BO; however, BOHB did not manage to reach comparable performance with the other algorithms in these two tasks. We observed that the sub-optimal behavior of Hyperband and BOHB observed here can be alleviated if the search space of hyperparameters is chosen to be smaller, in which case the advantage of pure exploration can be better manifested. The unsatisfactory performance of BOCA might be explained by the fact that it does not make use of the intermediate validation errors when selecting the number of epochs. In contrast, BO-BOS

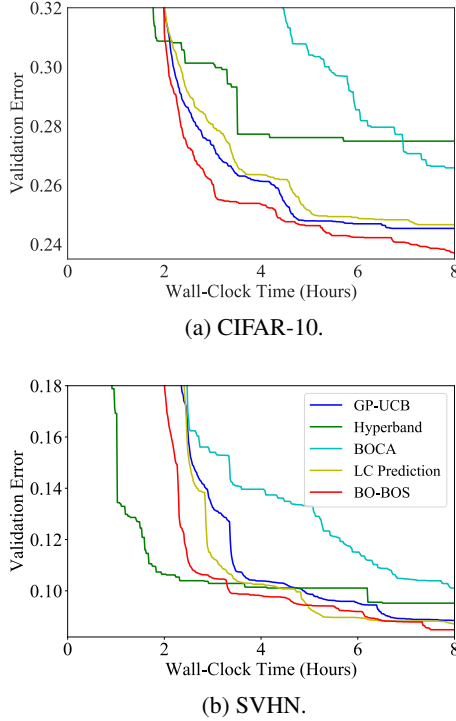


Figure 2. Best-found validation error of CNN v.s. run-time (averaged over 30 random initializations)

takes into account the observations after each training epoch when choosing the optimal stopping time, and thus is able to make better-informed decisions. Moreover, since BOCA is designed for general scenarios, the useful assumption of monotonic learning curve utilized by BO-BOS is not exploited; therefore, BOCA is expected to perform better if the fidelity levels result from data sub-sampling. LC Prediction performs similarly to GP-UCB, which might be because the predicted final values of the learning curves are used as real observations in the GP surrogate function (Domhan et al., 2015), thus invalidating the theoretical guarantee and deteriorating the convergence of GP-UCB, which offsets the computational saving provided by early stopping. BO-BOS, on the other hand, offers theoretically guaranteed convergence, thus allowing explicit control over the trade-off between the speed of convergence and the reduction in the average number of epochs as discussed in Section 4.

5.3. Novel Applications of the BO-BOS Algorithm

5.3.1. POLICY SEARCH FOR REINFORCEMENT LEARNING

Thanks to its superb sample efficiency, BO has been shown to be effective for policy search in reinforcement learning (RL) (Wilson et al., 2014), especially when policy evaluation is costly such as gait optimization for robots (Lizotte et al.,

2007) and vehicle navigation (Brochu et al., 2010). In policy search, the return of a policy is usually estimated by running the agent in the environment sequentially for a fixed number of steps and calculating the cumulative rewards (Wilson et al., 2014). Therefore, the sequential nature of the policy evaluation makes our BO-BOS algorithm an excellent fit to improve the efficiency of BO for policy search in RL.

We apply our algorithm to the Swimmer-v2 task from OpenAI Gym, MuJoCo (Brockman et al., 2016; Todorov et al., 2012), and use a linear policy consisting of 16 parameters. Each episode consists of 1000 steps, and we treat every m consecutive steps as one single epoch such that $N = 1000/m$. Direct application of BO-BOS in this task is inappropriate since the growth pattern of cumulative rewards differs significantly from the evolution of the learning curves, as illustrated in Appendix D.3. Therefore, the rewards are discounted (by γ) when calculating the objective function, because the pattern of discounted return (cumulative rewards) bears close resemblance to that of the learning curves. Note that although the value of the objective function is the discounted return, we also record and report the corresponding un-discounted return, which is the ultimate objective to be maximized. As a result, N and γ should be chosen such that the value of discounted return faithfully aligns with its un-discounted counterpart. Fig. 3 plots the best (un-discounted) return in an episode against the total number of steps, in which BO-BOS (with $N = 50$ and $\gamma = 0.9$) outperforms GP-UCB. The observation that the solid red line shows better returns than the two dotted red lines might be because overly small γ (0.75) and overly large N (100) both enlarge the disparity between discounted and un-discounted returns since they both downplay the importance of long-term discounted rewards. Moreover, not discounting the rewards ($\gamma = 1$) leads to poor performance, corroborating the earlier analysis motivating the use of discounted rewards. The results demonstrate that even though BO-BOS is not immediately applicable in the original problem, it can still perform effectively if the problem is properly transformed, which substantiates the general applicability of BO-BOS. Moreover, we also applied Hyperband in this task, but it failed to converge to a comparable policy to the other methods (achieving an average return of around 2.5), which further supports our earlier claim stating that Hyperband under-performs when the search space is large because there are significantly more parameters (16) than the previous tasks.

Interestingly, both BO-BOS (with $N = 50$ and $\gamma = 0.9$) and GP-UCB use significantly less steps to substantially outperform the benchmarks¹ achieved by some recently developed deep RL algorithms, in which the best-performing algorithm (Deep Deterministic Policy Gradients) achieves

¹<https://spinningup.openai.com/en/latest/spinningup/bench.html>

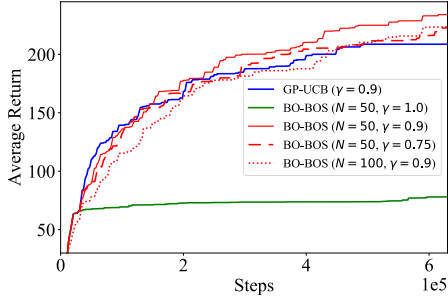


Figure 3. Best-found return (averaged over 5 episodes) v.s. the total number of steps of the robot in the environment (averaged over 30 random initializations) using the Swimmer-v2 task. The BO-BOS algorithm is run with different values of N (the maximum number of epochs) and γ (the discount factor).

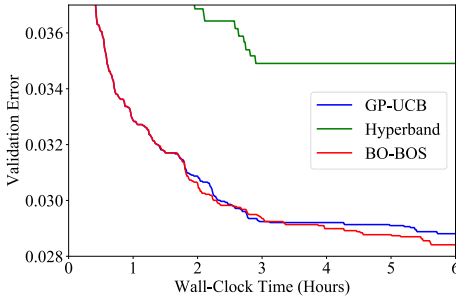


Figure 4. Best-found validation error of XGBoost v.s. run-time (averaged over 30 random initializations), obtained using joint hyperparameter tuning and feature selection.

an average return of around 120. Although the simplicity of the task might have contributed to their overwhelming performances, the results highlight the considerable potential of BO-based policy search algorithms for RL. Note that following the common practice in RL, Fig. 3 is presented in terms of the total number of steps instead of the run-time; we believe the additional computation required by BOS can be easily overshadowed in large-scale RL tasks, as demonstrated in the previous experiments. Most modern RL algorithms rely on enormous number of samples, which makes their applications problematic when sample efficiency is of crucial importance (Arulkumaran et al., 2017). As shown above, BO-BOS achieves significantly better results than popular deep RL algorithms while at the same time being far more sample-efficient, thus potentially offering practitioners a practically feasible option in solving large-scale RL problems.

5.3.2. JOINT HYPERPARAMETER TUNING AND FEATURE SELECTION

Beside hyperparameter tuning, feature selection is another important pre-processing step in ML, because it can lower the computational cost and boost the model performance (Hall, 2000). There are two types of feature selection techniques: filter and wrapper methods; the wrapper methods, such as forward selection (which starts with an empty feature set and in each iteration greedily adds the feature with the largest performance improvement), have been shown to perform well, although computationally costly (Hall & Smith, 1999). Interestingly, as a result of the sequential nature of the wrapper methods, they can be naturally incorporated into BO-BOS by simply replacing the sequential training of ML models with sequential forward selection.

In this task, we tune four hyperparameters of the gradient boosting model (XGBoost (Chen & Guestrin, 2016)) trained on an email spam dataset. We additionally compare with Hyperband since it was previously applied to random feature approximation in kernel methods (Li et al., 2017). As shown in Fig. 4, BO-BOS again delivers the best performance in this application. Consistent with Fig. 2, the wall-clock time includes all the time incurred during the algorithms. The $K_{1,t}$ sequence is made smaller than before: $K_{1,t} = \frac{K_{1,t-1}}{0.99}$, because in this setting, more aggressive early stopping is needed for BO-BOS to show its advantage. Although Hyperband works well for random feature approximation (Li et al., 2017), it does not perform favourably when applied to more structured feature selection techniques. Both hyperparameter optimization and feature selection have been shown to be effective for enhancing the performance of ML models. However, performing either of them in isolation may lead to sub-optimal performance since their interaction is un-exploited. Our results suggest that BO-BOS can effectively improve the efficiency of joint hyperparameter tuning and feature selection, making the combined usage of these two pre-processing techniques a more practical choice.

6. Conclusion

In this work, we present a unifying framework, BO-BOS, that integrates BOS into BO in a natural way, to derive a principled mechanism for optimally stopping hyperparameter evaluations during BO. We analyze the regret of the algorithm, and derive the BOS parameters that make the resulting BO-BOS algorithm no-regret. Applications of BO-BOS to hyperparameter tuning of ML models, as well as two other novel applications, demonstrate the practical effectiveness of the algorithm. For future work, we plan to further exploit the generality of the BO-BOS algorithm by applying it to more applications with costly iterative function evaluations.

References

- Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 17(6):26–38, 2017.
- Baker, B., Gupta, O., Raskar, R., and Naik, N. Accelerating neural architecture search using performance prediction. In *Proc. NIPS Workshop on Meta-Learning*, 2017.
- Brochu, E., Cora, V. M., and de Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599, 2010.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. arXiv:1606.01540, 2016.
- Brockwell, A. E. and Kadane, J. B. A gridding method for Bayesian sequential decision problems. *J. Computational and Graphical Statistics*, 12(3):566–584, 2003.
- Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proc. KDD*, pp. 785–794, 2016.
- Davis, G. A. and Cairns, R. D. Good timing: The economics of optimal stopping. *Journal of Economic Dynamics and Control*, 36(2):255–265, 2012.
- Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Domhan, T., Springenberg, J. T., and Hutter, F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proc. IJCAI*, pp. 3460–3468, 2015.
- Falkner, S., Klein, A., and Hutter, F. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proc. ICML*, pp. 1436–1445, 2018.
- Ferguson, T. S. *Optimal stopping and applications*. 2006. URL <https://www.math.ucla.edu/~tom/Stopping/Contents.html>.
- Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Ann. Statistics*, 29(5):1189–1232, 2001.
- Hall, M. A. Correlation-based feature selection of discrete and numeric class machine learning. In *Proc. ICML*, pp. 359–366, 2000.
- Hall, M. A. and Smith, L. A. Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *Proc. FLAIRS conference*, pp. 235–239, 1999.
- Jiang, F., Jack Lee, J., and Müller, P. A Bayesian decision-theoretic sequential response-adaptive randomization design. *Statistics in Medicine*, 32(12):1975–1994, 2013.
- Kandasamy, K., Dasarthy, G., Oliva, J. B., Schneider, J., and Póczos, B. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Proc. NIPS*, pp. 992–1000, 2016.
- Kandasamy, K., Dasarthy, G., Schneider, J., and Póczos, B. Multi-fidelity Bayesian optimisation with continuous approximations. In *Proc. ICML*, pp. 1799–1808, 2017.
- Klein, A., Falkner, S., Springenberg, J. T., and Hutter, F. Learning curve prediction with Bayesian neural networks. In *Proc. ICLR*, 2017.
- Krizhevsky, A. Learning multiple layers of features from tiny images. Master’s thesis, Department of Computer Science, University of Toronto, 2009.
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.
- Lewis, R. J. and Berry, D. A. Group sequential clinical trials: A classical evaluation of Bayesian decision-theoretic designs. *J. American Statistical Association*, 89(428):1528–1534, 1994.
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(1):6765–6816, 2017.
- Lizotte, D. J., Wang, T., Bowling, M. H., and Schuurmans, D. Automatic gait optimization with Gaussian process regression. In *IJCAI*, pp. 944–949, 2007.
- Longstaff, F. A. and Schwartz, E. S. Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001.
- Müller, P., Berry, D. A., Grieve, A. P., Smith, M., and Krams, M. Simulation-based sequential Bayesian design. *J. Statistical Planning and Inference*, 137(10):3140–3150, 2007.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- Powell, W. B. and Ryzhov, I. O. *Optimal learning*. John Wiley & Sons, 2012.
- Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for machine learning*. MIT Press, 2006.

- Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pp. 1015–1022, 2010.
- Swersky, K., Snoek, J., and Adams, R. P. Freeze-thaw Bayesian optimization. arXiv:1406.3896, 2014.
- Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *Proc. IEEE/RSJ IROS*, pp. 5026–5033, 2012.
- Wathen, J. K. and Thall, P. F. Bayesian adaptive model selection for optimizing group sequential clinical trials. *Statistics in Medicine*, 27(27):5586–5604, 2008.
- Wilson, A., Fern, A., and Tadepalli, P. Using trajectory data to improve Bayesian optimization for reinforcement learning. *Journal of Machine Learning Research*, 15(1): 253–282, 2014.

A. Approximate Backward Induction for Bayesian Optimal Stopping

In this section, we present a commonly-used approximate backward induction algorithm for solving the BOS problem. The algorithm uses summary statistics to compactly represent the posterior belief $\Pr(\theta_t | \mathbf{y}_{t,n})$ which is computed from the prior belief $\Pr(\theta_t)$ and the noisy outputs $\mathbf{y}_{t,n}$ observed up till epoch n in iteration t .

In the approximate backward induction algorithm introduced by Müller et al. (2007), the entire space of summary statistics is firstly partitioned into M discrete intervals in each epoch, which results in a two-dimensional domain (i.e., assuming the summary statistics is one-dimensional) with one axis being the number of epochs and the other axis representing the discretized intervals of the summary statistics. At the beginning, a number of sample paths are generated through *forward simulation*: Firstly, a large number of samples are drawn from the prior belief $\Pr(\theta_t)$. Then, for each sample drawn from $\Pr(\theta_t)$, an entire sample path is generated from epochs 1 to N through repeated sampling. In this manner, each sample path leads to a curve in the 2-D space of summary statistics and fully defines N posterior beliefs with one in each epoch. The calculation starts from the last epoch N : For each interval j , the expected loss of a terminal decision d_1 or d_2 is calculated for every sample path that ends up in the current interval (since each such sample path corresponds to a particular posterior belief) and their empirical average is used as the expected loss of the particular terminal decision. The expected loss for each interval is then taken as the minimum of the expected losses among all terminal decisions for that particular interval, which is equivalent to (2) except that decision d_0 is not available in the last epoch N .

Next, the algorithm proceeds backwards epoch by epoch from $n = N - 1$ all the way to $n = 1$. In epoch n , the expected loss of each terminal decision is calculated in the same way as that in the last epoch N , as described above. To calculate the expected loss of the continuation decision for an interval, for each sample path that passes through this interval, the expected loss for the next-epoch interval is recorded (this is equivalent to drawing samples from $\Pr(\mathbf{y}_{t,n+1} | \mathbf{y}_{t,n})$) and the average of the next-epoch losses is calculated and summed with the cost c_{d_0} of observing the noisy output $\mathbf{y}_{t,n+1}$ to yield the expected loss of the continuation decision d_0 (this is equivalent to estimating $\mathbb{E}_{\mathbf{y}_{t,n+1} | \mathbf{y}_{t,n}} [\rho_{t,n+1}(\mathbf{y}_{t,n+1})] + c_{d_0}$ via sampling). Subsequently, the minimum of the expected losses among all terminal and continuation decisions is used as the expected loss for the current interval (this is again the implementation of (2)) and the corresponding decision is recorded as the optimal decision when the summary statistics fall into this interval. Then, the backward induction algorithm continues backwards until epoch $n = 1$ is reached. After the algorithm is finished, the optimal decisions computed in each epoch and each interval will form the optimal decision rules which serve as the output of the approximate backward induction algorithm.

B. Approximate Backward Induction for the BO-BOS algorithm

In this section, we describe the approximate backward induction algorithm that is used to solve the BOS problem in each iteration of the BO-BOS algorithm, which is adapted from the algorithm introduced in section A (Müller et al., 2007).

To integrate Assumption 1 into the approximate backward induction algorithm, we adopt the kernel introduced in (Swersky et al., 2014) to incorporate the inductive bias that the learning curve of validation error is approximately exponentially decreasing in the number of training epochs, which can be expressed as

$$k(n, n') = \int_0^\infty e^{-\lambda n} e^{-\lambda n'} \phi(d\lambda) = \frac{\beta^\alpha}{(n + n' + \beta)^\alpha} \quad (5)$$

in which n and n' are different epoch numbers, ϕ is a probability measure over λ which is chosen to be a prior Gamma distribution, and α and β are parameters for the prior Gamma distribution. In this work, we make use of the kernel by drawing samples from the Gaussian process that is fit on the learning curve using this kernel, and plugging-in the samples as the forward simulation samples in approximate backward induction. In particular, when evaluating a hyperparameter setting \mathbf{x}_t , we first train the ML model for a fixed number (N_0) of initial training epochs (e.g., $N_0 = 8$ in all experiments when $N = 50$), and use the observed initial validation errors to calculate α and β in Equation 5 through Bayesian update (assuming the validation errors follow the conjugate distribution to the Gamma distribution, i.e., the exponential distribution); then, we draw a large number of samples from the resulted Gaussian process model in the range $n = [N_0 + 1, N]$, which will be used as the forward simulation samples. Figure 5 plots some forward simulation samples drawn from the posterior Gaussian process with the kernel in Equation 5 as described above. The figure demonstrates that the assumption of monotonic learning curve made in Assumption 1 is well expressed by the kernel.

Following the practices in related applications of BOS (Brockwell & Kadane, 2003; Müller et al., 2007; Jiang et al., 2013),

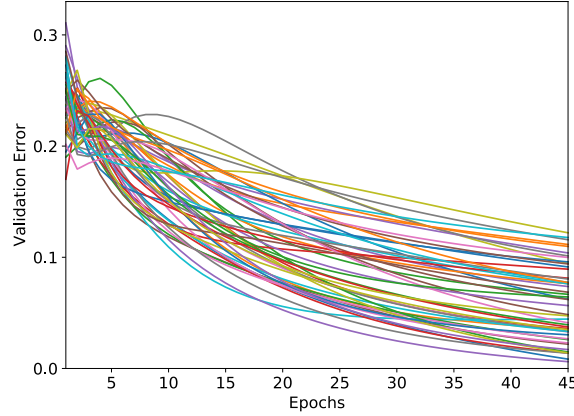


Figure 5. Some forward simulation samples drawn from the posterior GP.

we use the average validation error (or equivalently the average validation accuracy) up to the current epoch n as the summary statistics S_n . Firstly, we divide the entire space of summary statistics into a number of discrete intervals at each epoch, resulting in a two-dimensional domain with one axis being the number of training epochs, and the other being the discretized intervals of the summary statistics (the average validation error). Next, a large number (100,000 is used in all the experiments) of forward simulation samples are generated with the kernel in Equation 5 in the way described above. Similar to algorithm proposed in (Müller et al., 2007), each forward simulation sample path corresponds to one curve in the 2-D space of summary statistics. Subsequently, we start the backward induction procedure from the last step N : for each interval, we inspect all the sample path curves that end up in this interval, and use the percentage of these samples that end up having larger validation accuracy at N than the incumbent value as the estimated posterior probability $\Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n})$, which is then used to calculate the expected losses of the terminal decisions d_1 and d_2 using Equation 4. Next, the minimum of these two expected losses is taken as the expected loss of the particular interval. Subsequently, we proceed backwards from $n = N - 1$ to $n = N_0 + 1$ such that at each n , for every interval, the expected losses of d_1 and d_2 are calculated in the same way as the last step as described above. The expected loss of the continuation decision d_0 is calculated in the same way as in (Müller et al., 2007): for each sample path that passes through the current interval, the expected loss of the next-step interval of the particular sample path at $n + 1$ is recorded, then all the recorded next-step expected losses are averaged, whose summation with the observation cost c_{d_0} is taken as the expected loss of d_0 . Note that this step is equivalent to approximating the term $\mathbb{E}_{y_{t,n+1} | \mathbf{y}_{t,n}} [\rho_{t,n+1}(\mathbf{y}_{t,n})]$ in Equation 2 through Monte Carlo sampling from the distribution $\Pr(y_{t,n+1} | \mathbf{y}_{t,n})$. Next, the expected loss of the current interval is calculated by taking the minimum over all three decisions following Equation 2, and corresponding decision is recorded as the optimal decision to take when the summary statistics falls into this particular interval. Then, the algorithm proceeds further backwards until $n = N_0 + 1$.

After solving the BOS problem through approximate backward induction as described above, we have obtained the decision rules that tell us whether to stop the evaluation of \mathbf{x}_t after each training epoch. A typical set of obtained decision rules is visualized in Fig. 6, in which the yellow, white and red regions correspond to decisions d_1 , d_0 and d_2 respectively. In particular, after collecting the n -th validation error, we first calculate the updated summary statistics at step n (the average validation error); next, we find the corresponding interval of the the updated summary statistics, and examine which region it belongs to, and take the corresponding decision suggested by the decision rules. Specifically, if the summary statistics falls into the yellow region for any n , we take decision d_1 and immediately terminate the evaluation of x_t (assuming C2 is satisfied); otherwise, if the summary statistics falls into any other region, we continue to train for one more epoch, and repeat this process at the next step. This procedure, together with C2, constitutes lines 7 and 8 of Algorithm 1. The black region in Figure 6 results from the assumption of monotonic learning curve (Assumption 1) and the use of the kernel in Equation 5, which restrict the regions that can be reached by the forward simulation samples.

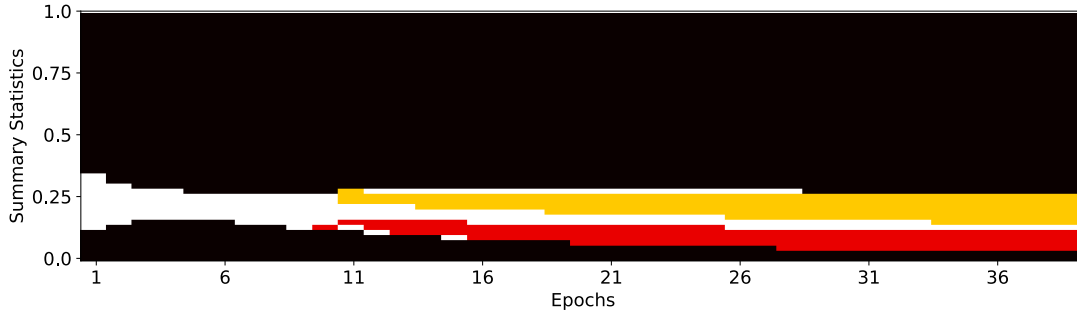


Figure 6. Example decision rules obtained from BOS.

C. Proof of Theorems 1 and 2

In this section, we prove the theoretical results in this paper.

C.1. Regret Decomposition

In this work, it is natural and convenient to define the instantaneous regret at step t as $r_t = f(\mathbf{z}^*) - f_t^*$, in which \mathbf{z}^* is the location of the global maximum: $\mathbf{z}^* = \operatorname{argmax}_{\mathbf{z}} f(\mathbf{z})$, and f_t^* is the maximum observed function value from iterations 1 to t : $f_t^* = \max_{t'=1, \dots, t} f(\mathbf{z}_{t'})$. Subsequently, the cumulative regret and simple regret after T iterations are defined as $R_T = \sum_{t=1}^T r_t$ and $S_T = \min_{t=1, \dots, T} r_t$ respectively. As a result, as long as we can show that R_T grows sub-linearly in T , then we can conclude that the average regret $\frac{R_T}{T}$ asymptotically goes to 0; therefore, S_T vanishes asymptotically since it is upper-bounded by the average regret: $S_T \leq \frac{R_T}{T}$. In contrast to the more commonly used definition of instantaneous regret: $r_t = f(\mathbf{z}^*) - f(\mathbf{z}_t)$, the slightly modified definition introduced here is justified in the sense that the induced definition of simple regrets, which is the ultimate goal of the theoretical analysis, obtained in both cases are equivalent, i.e., $\min_{t=1, \dots, T} f(\mathbf{z}^*) - f_t^* = \min_{t=1, \dots, T} f(\mathbf{z}^*) - f(\mathbf{z}_t)$.

The instantaneous regret defined above can be further decomposed as

$$\begin{aligned} r_t &= f(\mathbf{z}^*) - f_t^* = f(\mathbf{z}^*) - \max_{t'=1, \dots, t} f(\mathbf{z}_{t'}) \\ &= f(\mathbf{z}^*) - \max\{f_{t-1}^*, f(\mathbf{z}_t)\} \end{aligned} \quad (6)$$

Note that in our algorithm, the BO iterations can be divided into two types: **1)** t^+ such that $n_{t^+} = N$: those iterations that are not early-stopped; and **2)** t^- such that $n_{t^-} < N$: those that are early-stopped. For all t^+ , it follows from Equation 6 that $r_t = f(\mathbf{z}^*) - \max\{f_{t-1}^*, f(\mathbf{z}_t)\} \leq f(\mathbf{z}^*) - f(\mathbf{z}_t) = f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t]) = f(\mathbf{z}^*) - f([\mathbf{x}_t, N]) \triangleq r_{t^+}$; for all t^- , from Equation 6, we have that $r_t = f(\mathbf{z}^*) - \max\{f_{t-1}^*, f(\mathbf{z}_t)\} \leq f(\mathbf{z}^*) - f_{t-1}^* \triangleq r_{t^-}$. In the following, we will focus on the analysis of the sum of all r_{t^+} and all r_{t^-} : $R'_T = \sum_{t^+} r_{t^+} + \sum_{t^-} r_{t^-}$. As a result of the definition, R'_T is an upper bound of R_T , therefore, sub-linear growth of R'_T implies that R_T also grows sub-linearly.

Next, note that for all t^- such that $n_{t^-} < N$ (when \mathbf{x}_t is early-stopped),

$$\begin{aligned} r_{t^-} &= f(\mathbf{z}^*) - f_{t-1}^* = \underbrace{f(\mathbf{z}^*) - f([\mathbf{x}_t, N])}_{(1)} + \underbrace{f([\mathbf{x}_t, N]) - f_{t-1}^*}_{(2)} \\ &\leq \underbrace{f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])}_{(1)} + \underbrace{f([\mathbf{x}_t, N]) - f_{t-1}^*}_{(2)} \end{aligned} \quad (7)$$

in which (1) results from Assumption 1. As a result, R'_T can be re-written as

$$\begin{aligned}
 R'_T &\stackrel{(1)}{=} \sum_{\{t|n_t=N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t < N\}} [f(\mathbf{z}^*) - f_{t-1}^*] \\
 &= \sum_{\{t|n_t=N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t < N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t < N\}} [f([\mathbf{x}_t, N]) - f_{t-1}^*] \\
 &\stackrel{(2)}{\leq} \sum_{\{t|n_t=N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t < N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])] + \sum_{\{t|n_t < N\}} [f([\mathbf{x}_t, N]) - f_{t-1}^*] \\
 &\stackrel{(3)}{=} \sum_{t=1}^T [f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])] + \sum_{\{t|n_t < N\}} [f([\mathbf{x}_t, N]) - f_{t-1}^*] \\
 &\triangleq \sum_{t=1}^T r_{t,1} + \sum_{\{t|n_t < N\}} r_{t,2} \\
 &\triangleq R_{T,1} + R_{T,2}
 \end{aligned} \tag{8}$$

in which (1) makes use of the definition of R'_T , (2) results from Equation 7 and (3) follows by combining the first two terms on the previous line. The first term following (3) of Equation 8 is summed over all time steps, whereas the second term is only summed over those time steps that are early-stopped ($n_t < N$). As mentioned earlier, in the sequel, we will attempt to prove an upper bound on the expected value of R'_T ,

$$\mathbb{E}[R'_T] \leq \mathbb{E}[R_{T,1}] + \mathbb{E}[R_{T,2}] \tag{9}$$

in which the expectation is taken with respect to the posterior probabilities used in the BOS problem, corresponding to those iterations that are early-stopped: $\Pi_{t \in \{t' | t'=1, \dots, T, n_{t'} < N\}} \Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t})$. Note that the probability distributions are independent across all the early-stopped iterations, therefore, for each early-stopped iteration t , the expectations of both $r_{t,1}$ and $r_{t,2}$ are only taken over the specific distribution: $\Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t})$; whereas for each not-early-stopped iteration t , $\mathbb{E}[r_{t,1}] = r_{t,1}$ (whereas $r_{t,2}$ is absent). In the next two sections, we will prove upper bounds on $\mathbb{E}[R_{T,1}]$ and $\mathbb{E}[R_{T,2}]$ respectively.

C.2. Upper Bound on $\mathbb{E}[R_{T,1}]$

In this section, we will upper-bound the term $\mathbb{E}[R_{T,1}]$. As mentioned in the main text, for simplicity, we will focus on the case in which the underlying domain \mathcal{D} is discrete, i.e., $|\mathcal{D}| < \infty$. To begin with, we will need a supporting lemma showing a uniform upper bound over the entire domain.

Lemma 1. Suppose that $\delta \in (0, 1)$ and $\beta_t \triangleq 2 \log(|\mathcal{D}| t^2 \pi^2 / 6\delta)$. Then, with probability $\geq 1 - \delta$

$$|f(\mathbf{z}) - \mu_{t-1}(\mathbf{z})| \leq \beta_t^{1/2} \sigma_{t-1}(\mathbf{z}) \quad \forall \mathbf{z} \in \mathcal{D}, t \geq 1.$$

The proof of lemma 1 makes use of standard Gaussian tail bounds and a number of union bounds, and the proof is identical to the proof of lemma 5.1 in (Srinivas et al., 2010). The next supporting lemma makes use of the Lipschitz continuity of f to bound the differences between function values whose inputs only differ by the dimension corresponding to the number of training epochs.

Lemma 2. Suppose that Assumption 2 holds and let $\delta' \in (0, 1)$. Then, with probability $\geq 1 - \delta'$,

$$|f([\mathbf{x}, N]) - f([\mathbf{x}, n])| \leq Nb \sqrt{\log \frac{da}{\delta'}} \quad \forall \mathbf{x} \in \mathcal{D}, n = 1, \dots, N.$$

Proof. Let $\mathbf{z} = [\mathbf{x}, n]$ denote the input to the objective function f . Assumption 2, together with a union bound over $j = 1, \dots, d$, implies that with probability $\geq 1 - dae^{-(\frac{\delta'}{b})^2}$,

$$|f(\mathbf{z}) - f(\mathbf{z}')| \leq L \|\mathbf{z} - \mathbf{z}'\|_1$$

Since $[\mathbf{x}, N]$ and $[\mathbf{x}, n]$ differ only by the dimension corresponding to the number of training epochs, we have that

$$|f([\mathbf{x}, N]) - f([\mathbf{x}, n])| \leq LN$$

Then, the lemma follows by letting $\delta' = dae^{-(\frac{L}{\delta})^2}$. \square

The next lemma bounds $E[r_{t,1}]$ by the Gaussian process posterior standard deviation with some scaling constants.

Lemma 3. *Let $\delta, \delta' \in (0, 1)$ and $\kappa \geq 1$ be the constant used in C2 in the BO-BOS algorithm. Then, at iteration t of the BO-BOS algorithm, we have that, with probability $\geq 1 - \delta - \delta'$,*

$$\mathbb{E}[r_{t,1}] \leq 2\kappa\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) + Nb\sqrt{\log \frac{da}{\delta'}} \mathbb{1}_{n_t < N}.$$

Proof. Firstly, with probability $\geq 1 - \delta$,

$$f(\mathbf{z}^*) \stackrel{(1)}{=} f([\mathbf{x}^*, N]) \stackrel{(2)}{\leq} \mu_{t-1}([\mathbf{x}^*, N]) + \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}^*, N]) \stackrel{(3)}{\leq} \mu_{t-1}([\mathbf{x}_t, N]) + \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) \quad (10)$$

in which (1) follows from Assumption 1 which states that, for each \mathbf{x} , the function value is monotonically non-decreasing in the number of training epochs, which implies that at the (unknown) global maximum \mathbf{z}^* , the dimension corresponding to the number of epochs is equal to N . (2) makes use of Lemma 1, whereas (3) is due to the way \mathbf{x}_t is selected in the algorithm, i.e., $\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x}} \mu_{t-1}([\mathbf{x}, N]) + \sqrt{\beta_t}\sigma_{t-1}([\mathbf{x}, N])$. As a result, we have that with probability $\geq 1 - \delta - \delta'$

$$\begin{aligned} \mathbb{E}[r_{t,1}] &= \mathbb{E}[f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])] \stackrel{(1)}{\leq} \mathbb{E}[\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) + \mu_{t-1}([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \\ &= \mathbb{E}[\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) + \mu_{t-1}([\mathbf{x}_t, N]) - f([\mathbf{x}_t, N]) + f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \\ &\stackrel{(2)}{\leq} \mathbb{E}[2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N])] + \mathbb{E}[f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \\ &\stackrel{(3)}{\leq} \mathbb{E}[2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N])] + Nb\sqrt{\log \frac{da}{\delta'}} \mathbb{1}_{n_t < N} \\ &\stackrel{(4)}{\leq} 2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) + Nb\sqrt{\log \frac{da}{\delta'}} \mathbb{1}_{n_t < N} \\ &\stackrel{(5)}{\leq} 2\kappa\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) + Nb\sqrt{\log \frac{da}{\delta'}} \mathbb{1}_{n_t < N} \end{aligned} \quad (11)$$

in which (1) follows from Equation 10, and (2) results from Lemma 1 and the linearity of the expectation operator. $\mathbb{1}_{n_t < N}$ in (3) is the indicator function, which takes the value of 1 if the event $n_t < N$ is true and 0 otherwise. (3) is obtained by analyzing two different cases separately: if $n_t = N$ (\mathbf{x}_t is not early-stopped), then $\mathbb{E}[f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \stackrel{(1)}{=} f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t]) = 0$, in which (1) is because the expectation is taken w.r.t. only those iterations that are early-stopped; if $n_t < N$ (\mathbf{x}_t is early-stopped), then $\mathbb{E}[f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \leq \mathbb{E}[Nb\sqrt{\log \frac{da}{\delta'}}] = Nb\sqrt{\log \frac{da}{\delta'}}$ with probability $\geq 1 - \delta'$ following Lemma 2. (4) is due to the fact that $\sigma_{t-1}([\mathbf{x}_t, N])$ only depends on the observations up to step $t - 1$ and is not dependent on the probability $\Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t, n_t})$. (5) follows from the design of the algorithm; in particular, if $n_t < N$, then $\kappa\sigma_{t-1}([\mathbf{x}_t, n_t]) \geq \sigma_{t-1}([\mathbf{x}_t, N])$ is guaranteed by C2; otherwise, if $n_t = N$, then $\kappa\sigma_{t-1}([\mathbf{x}_t, n_t]) \geq \sigma_{t-1}([\mathbf{x}_t, n_t]) = \sigma_{t-1}([\mathbf{x}_t, N])$ since $\kappa \geq 1$. \square

Subsequently, we can upper bound $\mathbb{E}[R_{T,1}] = \sum_{t=1}^T \mathbb{E}[r_{t,1}]$ by extensions of Lemma 5.3 and 5.4 from (Srinivas et al., 2010), which are presented here for completeness. The following lemma connects the information gain about the objective function with the posterior predictive variance, whose proof results from straightforward extension of Lemma 5.3 of (Srinivas et al., 2010).

Lemma 4. *Let \mathbf{y}_T be a set of observations of size T , and let \mathbf{f}_T be the corresponding function values. The information gain about \mathbf{f}_T from observing \mathbf{y}_T is*

$$I(\mathbf{y}_T; \mathbf{f}_T) = \frac{1}{2} \sum_{t=1}^T \log[1 + \sigma^{-2}\sigma_{t-1}^2([\mathbf{x}_t, n_t])].$$

Next, we use the following lemma to bound the sum of the first term of the expected instantaneous regret as given in Lemma 3.

Lemma 5. *Let $\delta \in (0, 1)$, $C_1 \triangleq \frac{8}{\log(1+\sigma^{-2})}$, $\beta_t \triangleq 2\log(|\mathcal{D}|t^2\pi^2/6\delta)$, and $\gamma_T \triangleq \max_{A \in \mathcal{D}, |A|=T} I(\mathbf{y}_A; \mathbf{f}_A)$ is the maximum information gain about f from any subset of size T . Then,*

$$\sum_{t=1}^T 2\beta_t^{1/2} \sigma_{t-1}([\mathbf{x}_t, n_t]) \leq \sqrt{TC_1\beta_T I(\mathbf{y}_T; \mathbf{f}_T)} \leq \sqrt{TC_1\beta_T \gamma_T}.$$

Proof. Firstly, we have that

$$\begin{aligned} (2\beta_t^{1/2} \sigma_{t-1}([\mathbf{x}_t, n_t]))^2 &= 4\beta_t \sigma_{t-1}^2([\mathbf{x}_t, n_t]) \stackrel{(1)}{\leq} 4\beta_T \sigma^2[\sigma^{-2} \sigma_{t-1}^2([\mathbf{x}_t, n_t])] \\ &\stackrel{(2)}{\leq} 4\beta_T \sigma^2 \frac{\sigma^{-2}}{1+\sigma^{-2}} \log[1 + \sigma^{-2} \sigma_{t-1}^2([\mathbf{x}_t, n_t])] \\ &\leq \beta_T \frac{8}{1+\sigma^{-2}} \frac{1}{2} \log[1 + \sigma^{-2} \sigma_{t-1}^2([\mathbf{x}_t, n_t])] \end{aligned} \quad (12)$$

in which (1) holds since β_t is monotonically increasing in t ; (2) results from the fact that $\sigma^{-2}x \leq \frac{\sigma^{-2}}{1+\sigma^{-2}} \log[1 + \sigma^{-2}x]$ for $x \in (0, 1]$, whereas $0 < \sigma_{t-1}^2([\mathbf{x}_t, n_t]) \leq 1$. Next, summing over $t = 1, \dots, T$, we get

$$\begin{aligned} \sum_{t=1}^T (2\beta_t^{1/2} \sigma_{t-1}([\mathbf{x}_t, n_t]))^2 &\leq \beta_T \frac{8}{1+\sigma^{-2}} \frac{1}{2} \sum_{t=1}^T \log[1 + \sigma^{-2} \sigma_{t-1}^2([\mathbf{x}_t, n_t])] \\ &\stackrel{(1)}{\leq} \beta_T \frac{8}{1+\sigma^{-2}} I(\mathbf{y}_T; \mathbf{f}_T) \stackrel{(2)}{\leq} C_1 \beta_T \gamma_T \end{aligned} \quad (13)$$

in which (1) results from Lemma 4, and (2) follows from the definitions of C_1 and γ_T . Next, making use of the Cauchy-Schwarz inequality, we get

$$\begin{aligned} \sum_{t=1}^T 2\beta_t^{1/2} \sigma_{t-1}([\mathbf{x}_t, n_t]) &\leq \sqrt{T} \sqrt{\sum_{t=1}^T (2\beta_t^{1/2} \sigma_{t-1}([\mathbf{x}_t, n_t]))^2} \\ &\leq \sqrt{C_1 T \beta_T \gamma_T} \end{aligned} \quad (14)$$

which completes the proof. \square

Next, putting everything together, we get the follow lemma on the upper bound on $\mathbb{E}[R_{T,1}]$.

Lemma 6. *Suppose that Assumptions 1 and 2 hold. Let $\delta, \delta' \in (0, 1)$, $C_1 = 8/\log(1 + \sigma^{-2})$, $\beta_t = 2\log(|\mathcal{D}|t^2\pi^2/6\delta)$, $\kappa \geq 1$ be the constant used in C2 in the BO-BOS algorithm, and $\gamma_T = \max_{A \in \mathcal{D}, |A|=T} I(\mathbf{y}_A; \mathbf{f}_A)$. Let $\tau_T = \sum_{t=1}^T \mathbb{1}_{n_t < N}$ be the number of BO iterations in which early stopping happens from iterations 1 to T . Assume that f is a sample from a GP, and $y(\mathbf{z}) = f(\mathbf{z}) + \epsilon \forall \mathbf{z} \in \mathcal{D}$ in which $\epsilon \sim N(0, \sigma^2)$. Then, with probability $\geq 1 - \delta - \delta'$,*

$$\mathbb{E}[R_{T,1}] = \sum_{t=1}^T \mathbb{E}[r_{t,1}] \leq \kappa \sqrt{TC_1\beta_T \gamma_T} + Nb \sqrt{\log \frac{da}{\delta'}} \tau_T \quad \forall T \geq 1.$$

Proof.

$$\begin{aligned}
 \mathbb{E}[R_{T,1}] &\stackrel{(1)}{=} \sum_{t=1}^T \mathbb{E}[r_{t,1}] \stackrel{(2)}{\leq} \sum_{t=1}^T [2\kappa\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) + Nb\sqrt{\log \frac{da}{\delta'}} \mathbb{1}_{n_t < N}] \\
 &\stackrel{(3)}{\leq} \kappa\sqrt{TC_1\beta_T\gamma_T} + \sum_{t=1}^T Nb\sqrt{\log \frac{da}{\delta'}} \mathbb{1}_{n_t < N} \\
 &= \kappa\sqrt{TC_1\beta_T\gamma_T} + Nb\sqrt{\log \frac{da}{\delta'}} \sum_{t=1}^T \mathbb{1}_{n_t < N} \\
 &= \kappa\sqrt{TC_1\beta_T\gamma_T} + Nb\sqrt{\log \frac{da}{\delta'}} \tau_T
 \end{aligned} \tag{15}$$

in which (1) follows from the linearity of the expectation operator, (2) results from Lemma 3, and (3) follows from Lemma 5. \square

C.3. Upper Bound on $\mathbb{E}[R_{T,2}]$

In this section, we prove an upper bound on $\mathbb{E}[R_{T,2}]$. A few supporting lemmas will be presented and proved first. To begin with, the next lemma derives the appropriate choice of the incumbent values used in the Bayesian optimal stopping algorithm in different iterations of the BO-BOS algorithm.

Lemma 7. *Let the objective function f be a sample from a GP and $y(\mathbf{z}) = f(\mathbf{z}) + \epsilon \forall \mathbf{z} \in \mathcal{D}$ in which $\epsilon \sim N(0, \sigma^2)$. Let $\delta'' \in (0, 1)$. At iteration $t > 1$, define $f_{t-1}^* \triangleq \max_{t'=1, \dots, t-1} f(\mathbf{z}_{t'})$ and $y_{t-1}^* \triangleq \max_{t'=1, \dots, t-1} y_{t'}$; for iteration $t = 1$, define $f_0^* \triangleq 0$ and $y_0^* \triangleq 0$. Then with probability $\geq 1 - \delta''$,*

$$f_{t-1}^* \geq y_{t-1}^* - \xi_t \quad \forall t \geq 1$$

in which

$$\xi_t = \sqrt{2\sigma^2 \log \frac{\pi^2 t^2 (t-1)}{6\delta''}} \quad \forall t > 1$$

and $\xi_1 = 0$.

Proof. The lemma trivially holds for $t=1$. Assume we are at iteration $t > 1$ of the BO-BOS algorithm, and let $t' \in \{1, 2, \dots, t-1\}$. Since $y_{t'} = f(\mathbf{z}_{t'}) + \epsilon$, in which $\epsilon \sim N(0, \sigma^2)$, we have that $y_{t'} \sim N(f(\mathbf{z}_{t'}), \sigma^2)$. Making use of the upper deviation inequality for Gaussian distribution and the definition of ξ_t , we get

$$\Pr[y_{t'} \geq f(\mathbf{z}_{t'}) + \xi_t] \leq e^{-\frac{\xi_t^2}{2\sigma^2}} = \frac{6\delta''}{\pi^2 t^2 (t-1)} \tag{16}$$

Denote the event that $\{\exists t' \in \{1, 2, \dots, t-1\} \text{ s.t. } y_{t'} \geq f(\mathbf{z}_{t'}) + \xi_t\}$ as \mathcal{A}_t . Next, taking a union bound over the entire observation history $t' \in \{1, 2, \dots, t-1\}$, we get

$$\begin{aligned}
 \Pr[\mathcal{A}_t] &\leq \sum_{t'=1}^{t-1} \Pr[y_{t'} \geq f(\mathbf{z}_{t'}) + \xi_t] \\
 &\leq (t-1) \frac{6\delta''}{\pi^2 t^2 (t-1)} = \frac{6\delta''}{\pi^2 t^2}
 \end{aligned} \tag{17}$$

which implies that at iteration t , with probability $\geq 1 - \frac{6\delta''}{\pi^2 t^2}$, $y_{t'} - f(\mathbf{z}_{t'}) < \xi_t \forall t' \in \{1, 2, \dots, t-1\}$, which further suggests that $y_{t-1}^* - f_{t-1}^* \leq \gamma_t$ at iteration t . Next, taking a union bound over $t \geq 1$, we get

$$\Pr[\exists t \geq 1 \text{ s.t. } \mathcal{A}_t \text{ holds}] \leq \sum_{t \geq 1} \Pr[\mathcal{A}_t] \leq \sum_{t \geq 1} \frac{6\delta''}{\pi^2 t^2} = \delta'' \tag{18}$$

which suggests that, with probability $\geq 1 - \delta''$, $y_{t-1}^* - f_{t-1}^* \leq \xi_t \forall t \geq 1$, and thus completes the proof. \square

The next lemma shows that, with appropriate choices of the incumbent value, the posterior probability used in Bayesian optimal stopping is upper-bounded.

Lemma 8. *If in iteration t of the BO-BOS algorithm, the Bayesian optimal stopping algorithm is run with the incumbent value $y_{t-1}^* - \gamma_t$ and the corresponding cost parameters K_1 , K_2 and c_{d_0} , and the algorithm early-stops after $n_t < N$ epochs, then with probability $\geq 1 - \delta''$,*

$$\Pr(f([\mathbf{x}_t, N]) > f_{t-1}^* | \mathbf{y}_{t,n_t}) \leq \frac{K_2 + c_{d_0}}{K_1} \quad \forall t \geq 1. \quad (19)$$

Proof. Recall that when running the Bayesian optimal stopping algorithm, we only early-stop the experiment ($n < N$) when we can safely conclude that the performance of the currently evaluated hyperparameter x_t will end up having smaller validation accuracy than the currently found optimum f_{t-1}^* ; i.e., when the expected loss of decision d_1 is the smallest among all decisions. Therefore, when the evaluation of x_t is early-stopped after $n_t < N$ epochs, we can conclude that

$$\begin{aligned} & K_1 \Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t}) \\ & \leq \mathbb{E}_{\mathbf{y}_{t,n_t+1} | \mathbf{y}_{t,n_t}} \left[\min \{ K_1 \Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t+1}), K_2 \Pr(f([\mathbf{x}_t, N]) \leq y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t+1}), \right. \\ & \quad \left. \mathbb{E}_{\mathbf{y}_{t,n_t+2} | \mathbf{y}_{t,n_t+1}} [\rho_{n_t+2}] + c \} \right] + c_{d_0} \\ & \leq \mathbb{E}_{\mathbf{y}_{t,n_t+1} | \mathbf{y}_{t,n_t}} [K_2 \Pr(f([\mathbf{x}_t, N]) \leq y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t+1})] + c_{d_0} \\ & \leq K_2 \mathbb{E}_{\mathbf{y}_{t,n_t+1} | \mathbf{y}_{t,n_t}} [\Pr(f([\mathbf{x}_t, N]) \leq y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t+1})] + c_{d_0} \\ & \leq K_2 + c_{d_0} \end{aligned} \quad (20)$$

Equation 20, together with Lemma 7, implies that

$$\begin{aligned} \Pr(f([\mathbf{x}_t, N]) > f_{t-1}^* | \mathbf{y}_{t,n_t}) & \leq \Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t}) \\ & \leq \frac{K_2 + c_{d_0}}{K_1} \end{aligned} \quad (21)$$

which holds uniformly for all $t \geq 1$ with probability $\geq 1 - \delta''$. \square

Subsequently, we use the next Lemma to upper-bound $\mathbb{E}[R_T^2]$ by the BOS cost parameters. We set K_2 and c_{d_0} as constants, and use different values of K_1 in different iterations t of the BO-BOS algorithm, which is represented by $K_{1,t}$.

Lemma 9. *In iteration t of the BO-BOS algorithm, define $\frac{K_2 + c_{d_0}}{K_{1,t}} \triangleq \eta_t$. Then, with probability $\geq 1 - \delta''$,*

$$\mathbb{E}[R_{T,2}] \leq \sum_{t=1}^T \eta_t \quad \forall T \geq 1.$$

Proof. Recall that according to Assumption 1, the value of the objective function f is bounded in the range $[0, 1]$. In iteration t , assume we early-stop the evaluation of \mathbf{x}_t after $n_t < N$ epochs, then

$$\mathbb{E}[f([\mathbf{x}_t, N]) - f_{t-1}^* | \mathbf{y}_{t,n_t}] \stackrel{(1)}{\leq} \mathbb{E}[\mathbb{1}_{f([\mathbf{x}_t, N]) - f_{t-1}^* > 0} | \mathbf{y}_{t,n_t}] \stackrel{(2)}{=} \Pr(f([\mathbf{x}_t, N]) > f_{t-1}^* | \mathbf{y}_{t,n_t}) \quad (22)$$

Step (1) in Equation 22 is because $x \leq \mathbb{1}_{x>0} \quad \forall x \in [-1, 1]$ and substituting $x = f([\mathbf{x}_t, N]) - f_{t-1}^*$; (2) is due to the fact that the expectation of a Bernoulli random variable is equal to the corresponding probability value. As a result, with probability $\geq 1 - \delta''$

$$\begin{aligned} \mathbb{E}[R_{T,2}] & \stackrel{(1)}{=} \sum_{\{t | n_t < N\}} \mathbb{E}[r_{t,2}] \stackrel{(2)}{=} \sum_{\{t | n_t < N\}} \mathbb{E}[f([\mathbf{x}_t, N]) - f_{t-1}^* | \mathbf{y}_{t,n_t}] \stackrel{(3)}{\leq} \sum_{\{t | n_t < N\}} \Pr(f([\mathbf{x}_t, N]) > f_{t-1}^* | \mathbf{y}_{t,n_t}) \\ & \stackrel{(4)}{\leq} \sum_{\{t | n_t < N\}} \eta_t \leq \sum_{t=1}^T \eta_t \end{aligned} \quad (23)$$

in which (1) follows from the linearity of expectation, (2) holds because the Expectation of $r_{t,2}$ is taken over $\Pr(f([\mathbf{x}_t, N]) > y_{t-1}^* - \xi_t | \mathbf{y}_{t,n_t})$, (3) results from Equation 22, and (4) follows from Lemma 8. This completes the proof. \square

C.4. Putting Things Together

In this section, we put everything from the previous two sections together to prove the main theorems.

C.4.1. PROOF OF THEOREM 1

Theorem 1 can be proven by combining Lemmas 6 and 9, and making use of the fact that $S_T \leq \frac{R_T}{T}$.

C.4.2. PROOF OF THEOREM 2

Below we analyze the asymptotic behavior of each of the three terms in the upper bound of $\mathbb{E}[S_T]$ in Theorem 1, which is re-presented here for ease of reference.

$$\mathbb{E}[S_T] \leq \frac{\kappa \sqrt{TC_1 \beta_T \gamma_T}}{T} + \frac{\sum_{t=1}^T \eta_t}{T} + \frac{1}{T} N b \sqrt{\log \frac{da}{\delta'}} \tau_T. \quad (24)$$

The first term in the upper bound of $\mathbb{E}[S_T]$ Firstly, the first term in the upper bound matches the upper bound on the simple regret of the GP-UCB algorithm (Srinivas et al., 2010) (up to the constant κ). The maximum information gain, γ_T , has been analyzed for a few of the commonly used kernels in GP (Srinivas et al., 2010). For example, for the Square Exponential kernel, $\gamma_T = O(\log T)^{d+1}$, whereas for the Matérn kernel with $\nu > 1$, $\gamma_T = O(T^{d(d+1)/(2\nu+d(d+1))} \log T)$. Plugging both expressions of γ_T into Theorem 1, together with the expression of β_T as given in Theorem 1, shows that both kernels lead to sub-linear growth of the term $\sqrt{TC_1 \beta_T \gamma_T}$, which implies that the first term in the upper bound of $\mathbb{E}[S_T]$ asymptotically goes to 0.

The second term in the upper bound of $\mathbb{E}[S_T]$ Given that $K_{1,t}$ is an increasing sequence with $K_{1,1} \geq K_2 + c_{d_0}$, the series $\sum_{t=1}^T \eta_t = \sum_{t=1}^T \frac{K_2 + c_{d_0}}{K_{1,t}}$ grows sub-linearly, thus making the second term in the upper bound of $\mathbb{E}[S_T]$ given in Theorem 1, $\frac{\sum_{t=1}^T \eta_t}{T}$, asymptotically go to 0.

The third term in the upper bound of $\mathbb{E}[S_T]$ Next, suppose that $K_{1,t}$ becomes $+\infty$ for the first time at iteration T_0 . Since $K_{1,t}$ is a non-decreasing sequence, $K_{1,t} = +\infty$ for all $t \geq T_0$. Therefore, for $t \geq T_0$, decision d_1 will never be taken and the algorithm will never early-stop. In other words, $n_t = N$ for all $t \geq T_0$.

Therefore, we can conclude that $\tau_T \leq T_0$ for all $T \geq 1$. As a result, the last term in the upper bound on $\mathbb{E}[S_T]$ in Theorem 1 can be upper-bounded by

$$\frac{\tau_T}{T} N b \sqrt{\log \frac{da}{\delta'}} \leq \frac{T_0 N b \sqrt{\log \frac{da}{\delta'}}}{T} = O\left(\frac{1}{T}\right) \quad (25)$$

which asymptotically goes to 0 as T goes to $+\infty$, because the numerator term is a constant. Therefore, this term also asymptotically vanishes in the upper bound.

To summarize, if the BOS parameters are selected according to Theorem 2, we have that

$$\mathbb{E}[S_T] = O\left(\frac{\sqrt{T \beta_T \gamma_T}}{T} + \frac{\sum_{t=1}^T \eta_t}{T} + \frac{1}{T}\right) \quad (26)$$

and $\mathbb{E}[S_T]$ goes to zero asymptotically.

D. Additional Experimental Details

In each experiment, the same initializations (6 initial points if not further specified) are used for all BO-based methods: GP-UCB, BOCA, LC Prediction, and BO-BOS. The Square Exponential kernel is used for BOCA since the algorithm is only given for this kernel (Kandasamy et al., 2017), the other BO-based algorithms use the Matérn kernel; the kernel hyperparameters are updated by maximizing the Gaussian process marginal likelihood after every 10 BO iterations. In the BO-BOS algorithm, since the number of training epochs is an input to the GP surrogate function, some of the intermediate observations ($n < N$) can be used as additional input to GP to improve the modeling of the objective function. However, using the observation after every epoch as input leads to poor scalability. Therefore, for all experiments with $N = 50$ (which

include most of the experiments), we use the observations after first, 10-th, 20-th, 30-th and 40-th epochs as additional inputs to the GP surrogate function; whereas for the RL experiment with $N = 100$ in section 5.3.1, we use the 1-th, 20-th, 40-th, 60-th and 80-th intermediate observations as additional inputs. 100,000 forward simulation samples are used for each BOS algorithm; the grid size of the discretized summary statistics is set to 100; the incumbent value at iteration t is chosen as $y_{t-1}^* = \max_{t'=1, \dots, t-1} y_{t'}$, thus ignoring the observation noise. In the LC Prediction algorithm (Domhan et al., 2015), learning curve prediction is performed after every 2 epochs. In Hyperband (Li et al., 2017), the successive halving parameter η is set to 3 as recommended by the original authors, and the maximum number of epochs is set to $N = 80$ (we observed that setting $N = 80$ led to better performance than $N = 50$ since it allows the Hyperband algorithm to run for more epochs overall).

D.1. Hyperparameter Tuning for Logistic Regression

In the first set of experiments, we perform hyperparameter tuning for a simple ML model, logistic regression (LR). The LR model is trained using the MNIST image dataset, which consists of 70,000 images of the 10 digits, corresponding to a 10-class classification problem. Three hyperparameters are tuned: the batch size (20 to 500), L2 regularization parameter (10^{-6} to 1.0), and learning rate (10^{-3} to 0.1). We use 80% of the images as the training set and the remaining 20% as the validation set.

The reliability of the early stopping achieved by the BO-BOS algorithm is demonstrated in Fig. 7. In this figure, the green triangles correspond to the learning curves that are not early-stopped ($n_t = N$), and the red circles represent the final validation errors (after training for the maximum number of epochs N) that *could have been* reached by the early-stopped learning curves ($n_t < N$, we do not get to observe these values in practices). As displayed in the figure, the early stopping decisions made during the BO-BOS algorithm are reliable, since those early-stopped learning curves all end up having large validation error.

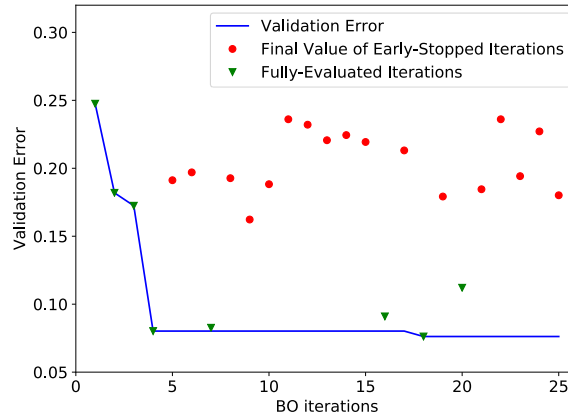


Figure 7. Illustration of the effectiveness of the early stopping decisions made during the BO-BOS algorithm.

D.2. Hyperparameter Tuning for Convolutional Neural Networks

Next, we tune the hyperparameters of convolutional neural networks (CNN) using the CIFAR-10 (Krizhevsky, 2009) and Street View House Numbers (SVHN) dataset (Netzer et al., 2011). Both tasks correspond to 10-class classification problems. For CIFAR-10, 50,000 images are used as the training set and 10,000 images are used as the validation set; for SVHN, 73,257 and 26032 images are used as the training and validation sets respectively following the original dataset partition. The CNN model consists of three convolutional layers (each followed by a max-pooling layer) followed by one fully-connected layer. We tune six hyperparameters in both experiments: the batch size (32 to 512), learning rate (10^{-7} to 0.1), learning rate decay (10^{-7} to 10^{-3}), L2 regularization parameter (10^{-7} to 10^{-3}), the number of convolutional filters in each layer (128 to 256), and the number of units in the fully-connected layer (256 to 512).

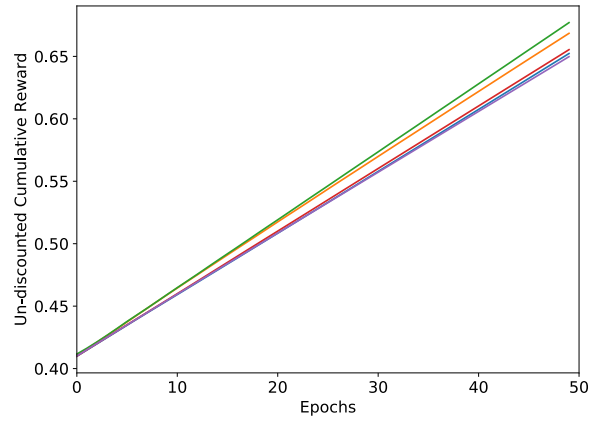
D.3. Policy Search for Reinforcement Learning

We apply our algorithm to a continuous control task: the Swimmer-v2 environment from OpenAI Gym, MuJoCo (Brockman et al., 2016; Todorov et al., 2012). The task involves controlling two joints of a swimming robot to make it swim forward as fast as possible. The state of the robot is represented by an eight-dimensional feature vector, and the action space is two-dimensional corresponding to the two joints. We use a linear policy, in which the policy is represented by an 8×2 matrix that maps each state vector to the corresponding action vector. In this setting, the input, x , to the GP-UCB and BO-BOS algorithms is the 16 parameters of the policy matrix, and the objective function is the discounted cumulative rewards in an episode. Each episode of the task consists of 1,000 steps. We set N (the maximum number of epochs) to be smaller than 1,000 by treating a fixed number of consecutive steps as one single epoch. E.g., we can set $N = 50$ or $N = 100$ by treating every 20 or 10 consecutive steps as one epoch respectively. The rewards are clipped, scaled, and normalized such that the discounted cumulative rewards of each episode is bounded in the range $[0, 1]$; for each evaluated policy, we also record the un-discounted and un-scaled cumulative rewards, which are the ultimate objective to be maximized and reported in Fig. 3 in the main text. Each policy evaluation consists of running 5 independent episodes with the given policy, and returning the average discounted cumulative rewards, i.e., average return, as the observed function value.

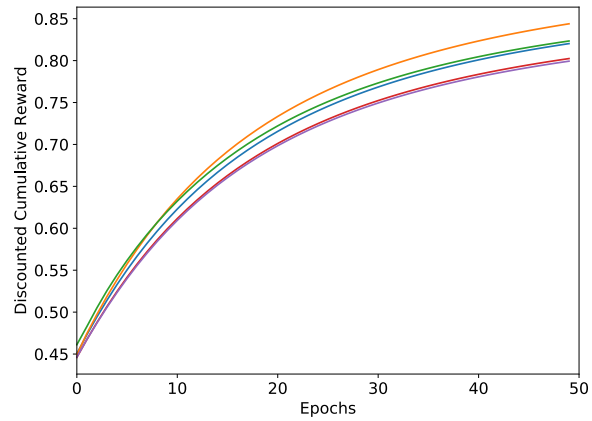
As mentioned in the main text, the rewards are discounted in order to make the objective function, the discounted cumulative rewards, resemble the learning curves of ML models, such that the BO-BOS algorithm can be naturally applied. This rationale is illustrated in Fig. 8, which plots some example un-discounted ($\gamma = 1.0$) and discounted ($\gamma = 0.9$) cumulative rewards respectively. The figures indicate that, compared with un-discounted cumulative rewards, discounted cumulative rewards bear significantly closer resemblance to the learning curves of ML models, thus supporting the claim made in the main text motivating the use of discounted rewards, as well as the experimental results shown in Fig. 3 (specifically, the poor performance of the curve corresponding to $N = 50$ and $\gamma = 1.0$).

D.4. Joint Hyperparameter Tuning and Feature Selection

In this set of experiments, we use the gradient boosting model (XGBoost (Chen & Guestrin, 2016)), tuning four hyperparameters: the learning rate (10^{-3} to 0.5), maximum depth of each decision tree (2 to 15), feature sub-sampling ratio for each tree (0.3 to 1.0), and L1 regularization parameter (0.0 to 5.0). We use the email spam dataset from the UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017), which represents a binary classification problem: whether the email is a spam or not. We use 3065 emails as the training set and the remaining 1536 emails as the validation set; each email consists of 57 features. The maximum number of features for each hyperparameter setting is set as $N = 50$.



(a) Un-discounted ($\gamma = 1.0$).



(b) Discounted ($\gamma = 0.9$).

Figure 8. Example curves of un-discounted and discounted cumulative rewards