Non-stationary Continuum-armed Bandits for Online Hyperparameter Optimization

Shiyin Lu
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
lusy@lamda.nju.edu.cn

Yu-Hang Zhou
Alibaba Group
Hangzhou, China
zyh174606@alibaba-inc.com

Jing-Cheng Shi
Alibaba Group
Hangzhou, China
jingcheng.sjc@alibaba-inc.com

Wenya Zhu
Alibaba Group
Hangzhou, China
wenya.zwy@alibaba-inc.com

Qingtao Yu
Alibaba Group
Hangzhou, China
qingtao.yqt@alibaba-inc.com

Qing-Guo Chen
Alibaba Group
Hangzhou, China
qingguo.cqg@alibaba-inc.com

Qing Da
Alibaba Group
Hangzhou, China
daqing.dq@alibaba-inc.com

Lijun Zhang*
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
zhanglj@lamda.nju.edu.cn

ABSTRACT

For years, machine learning has become the dominant approach to a variety of information retrieval tasks. The performance of machine learning algorithms heavily depends on their hyperparameters. It is hence critical to identity the optimal hyperparameter configuration when applying machine learning algorithms. Most of existing hyperparameter optimization methods assume a static relationship between hyperparameter configuration and algorithmic performance and are thus not suitable for many information retrieval applications with non-stationary environments such as e-commerce recommendation and online advertising. To address this limitation, we study online hyperparameter optimization, where the hyperparameter configuration is optimized on the fly. We formulate online hyperparameter optimization as a non-stationary continuum-armed bandits problem in which each arm corresponds to a hyperparameter configuration and the algorithmic performance is viewed as reward. For this problem, we develop principled methods with strong theoretical guarantees in terms of dynamic regret. The key idea is to adaptively discretize the continuous arm set and estimate the mean reward of each arm via weighted averaging. As a case application, we show how our methods can be applied to optimize the hyperparameter of vector-based candidate generation algorithm and empirically demonstrate the effectiveness and efficiency of our methods on public advertising dataset and online A/B testing. Furthermore, to the best of our knowledge, our methods are the first to achieve sub-linear dynamic regret bounds for continuum-armed bandits, which may be of independent interest.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

KEYWORDS

continuum-armed bandits, non-stationary environments, hyperparameter optimization

ACM Reference Format:
Shiyin Lu, Yu-Hang Zhou, Jing-Cheng Shi, Wenya Zhu, Qingtao Yu, Qing-Guo Chen, Qing Da, and Lijun Zhang. 2022. Non-stationary Continuum-armed Bandits for Online Hyperparameter Optimization. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3488560.3498396

1 INTRODUCTION

Machine learning is a popular and dominant approach for building information retrieval (IR) systems [4, 25, 27, 32, 42] and has been applied in various IR tasks such as e-commerce recommendation [43], web search [41], and online advertising [44]. The performance of machine learning algorithms typically depends on the configurations of hyperparameters, which are adjustable parameters that govern the learning process. Representative examples of hyperparameters in IR scenarios include thresholds of vector-based candidate generation algorithms [8] and step sizes of online learning to rank algorithms [16], to name a few. Due to the complex relationship between hyperparameter configuration and algorithmic performance, it is difficult to tune hyperparameters by standard optimization techniques such as convex optimization [39]. Instead, practitioners often resort to brute-force approaches such as grid search and random search [28, 31].

*Lijun Zhang is the corresponding author.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498396

Grid search refers to exhaustive exploration on a manually determined discretization of the hyperparameter space. In grid search, each discrete hyperparameter configuration is first evaluated and the one yielding the highest performance is then identified as the best configuration. While grid search is simple to implement and easy to parallelize, it requires human expertise to specify a suitable discretization level for balancing the computational cost of evaluating all discrete configurations and the quality of the obtained best configuration. As a refinement of grid search, random search avoids human overhead by uniformly drawing candidate hyperparameter configurations from the hyperparameter space [7]. It has been empirically demonstrated and widely recognized that random search is more effective than grid search in finding well-performing hyperparameter configurations [6].

Nevertheless, an intrinsic drawback of random search as well as grid search is that the performance of previously explored hyperparameter configurations is totally ignored when determining the next configuration to evaluate, which makes the exploration process rather inefficient. To address this drawback, recent advances in hyperparameter optimization have introduced Bayesian optimization to utilize historical information for more efficient exploration on the hyperparameter space [33]. Under Bayesian optimization framework, the relationship between hyperparameter configuration and algorithmic performance is characterized by an objective function. While the objective function is complex and remains unknown, Bayesian optimization only requires querying the objective function and uses a probabilistic surrogate model to approximate the objective function. The surrogate model is first initialized with a prior distribution that expresses one's belief on the unknown objective function and then iteratively updated according to the value of the objective function at the queried point. Equipped with the surrogate model, Bayesian optimization can exploit historical observations of the objective function and adaptively select the next hyperparameter configuration to explore. Such adaptivity also translates into superior performance than brute-force approaches on various tasks including information retrieval [23, 24, 37].

While Bayesian optimization stands as the state-of-the-art approach for optimizing hyperparameters of information retrieval systems [24], existing Bayesian optimization methods either assume a stationary objective function or lack of theoretical guarantees and hence not suitable for many non-stationary and cost-sensitive scenarios such as e-commerce recommendation and online advertising. For example, in online advertising, the click-through-rate of an ad usually changes with time and so does the objective function; a one-percent drop of click-through-rate due to a bad hyperparameter configuration can lead to million dollars losses [1]. Furthermore, Bayesian optimization is originally designed for offline hyperparameter optimization and its time complexity grows rapidly with iteration, which is prohibitive for real-time update applications.

In this paper, we study online hyperparameter optimization, where the objective function can vary with time and the hyperparameter configuration is optimized on the fly. We propose a novel bandit learning framework termed as non-stationary continuum-armed bandits for online hyperparameter optimization: each hyperparameter configuration is viewed as an arm with the corresponding algorithmic performance being reward. We develop provably effective methods under the proposed bandit learning framework. The main idea is to handle the continuous arm set by adaptive discretization and deal with the non-stationary environment via dynamic mean estimator. We rigorously analyze the performance of our methods in terms of dynamic regret and establish the first sublinear dynamic regret bounds for continuum-armed bandits, which may be of independent interest. Our methods are also efficient in the sense that each iteration only takes constant time. As a practical example, we apply our methods to optimize the hyperparameter of vector-based candidate generation algorithm. Extensive experimental comparison with existing hyperparameter algorithms on both public advertising dataset and online A/B testing demonstrates the effectiveness and efficiency of our methods.

2 RELATED WORK

Grid search is the most basic method for hyperparameter optimization and has been widely used in tuning hyperparameters of various machine learning models such as random forest [38], support vector machine [40], and deep neural networks [11]. Thanks to its simplicity and flexibility, grid search is commonly integrated in popular machine learning software packages including scikit-learn [30], libsvm [9], and pytorch [29]. Given a human-specified step size, grid search traverses the hyperparameter space step by step and evaluates each possible hyperparameter configuration. To find a well-performing hyperparameter configuration with affordable computational costs, one has to carefully set the step size and often repeats the process of grid search multiple times [45]. As an effort to overcoming this limitation, a variant of grid search termed as random search was proposed, where the hyperparameter space is randomly explored [2]. Random search inherits all advantages of grid search such as simple implementation and flexible parallelism and greatly reduces the manual burden of grid search [34]. The superiority of random search was delicately analyzed in [7], where random search was found to spend less time in exploring poor-performing regions of the hyperparameter space than grid search. [7] also empirically shows that random search is more effective than grid search in tuning hyperparameters of diverse machine learning algorithms on a variety of datasets.

A common criticism of grid search and random search is that the hyperparameter configuration to explore is independent of the hyperparameter configurations that have been evaluated, which may involve a large number of unnecessary hyperparameter evaluations [39]. To improve the exploration efficiency, advanced methods including Bayesian optimization [33] and Hyperband [26] have been proposed, where the exploration process is guided by the previously evaluated hyperparameter configurations. Bayesian optimization speeds up the identification of good hyperparameter configurations by adaptively determining the next hyperparameter to evaluate [6]. In each iteration, after observing the performance of previously evaluated hyperparameter configurations, Bayesian optimization first fits the observations into a surrogate model. Then, based on the surrogate model, an acquisition function is adopted to balance the trade-off between exploring potentially better regions and exploiting the currently promising regions [19]. According to the adopted surrogate model, Bayesian optimization methods can be divided into three categories, namely, sequential model-based algorithm configuration [19], tree-structured Parzen [6], and Spearmint

[33], where the surrogate model is random forest, Parzen density estimator, and Gaussian process, respectively. The three types of Bayesian optimization methods were empirically shown to beat random search on various tasks [12, 13, 24, 35, 37].

As an orthogonal research direction to Bayesian optimization, Hyperband [26] accelerates the optimization process by adaptively allocating the evaluation budget to different hyperparameter configurations. Hyperband is a best-arm identification (BAI) algorithm built upon the successive halving (SH) method [20]. Given a finite evaluation budget B, SH first samples n hyperparameter configurations each allocated B/n evaluation budget and then successively eliminates poorly-performing configurations while doubling the evaluation budget of the remain configurations. The difficultly of applying SH lies in balancing the trade-off between the number of configurations n and the average allocated evaluation budget across n configurations B/n. As a simple yet effective approach to this n versus B/n dilemma, Hyperband essentially performs a grid search over all feasible n and calls SH as a subroutine for each n. The effectiveness of Hyperband is empirically verified on a variety of problems [21, 36].

3 PRELIMINARY

We formulate online hyperparameter optimization as a novel non-stationary continuum-armed bandits problem, where a learner interacts with a continuous set of arms. Without loss of generality, we assume the arm set (the hyperparameter space) has been shifted and scaled to the unit interval, i.e., $[0, 1]$. The learning protocol proceeds over T rounds. In each round $t \in [T]$, the learner first chooses an arm (a hyperparameter configuration) I_t from $[0, 1]$ and then receives a reward of the chosen arm $r_t(I_t)$, which corresponds to the performance of the selected hyperparameter configuration. For each arm $a \in [0, 1]$, we denote by $\mathcal{D}_t(a)$ the probability distribution from which the reward of a in round t is drawn. Finally, the learner updates her arm selection strategy for the next round based on the received reward $r_t(I_t)$.

In each round t, let $\mu_t(a) = \mathbb{E}_{r\sim\mathcal{D}_t(a)}[r]$ denote the mean reward of an arm a. We define the arm with maximum mean reward as the optimal arm: $a_t^* = \arg\max_{a\in[0,1]} \mu_t(a)$. The learner's performance is measured by dynamic regret, which is the difference between the cumulative mean rewards of the arms chosen by the learner and that of the optimal arms:

$$\mathrm{DR}(T) = \sum_{t=1}^{T} \mu_t(a_t^*) - \sum_{t=1}^{T} \mu_t(I_t).$$

We make three mild assumptions as follows.

- The rewards are all bounded in $[0, 1]$ and the reward distributions are independent across arms and rounds.
- The mean reward function is Lipschitz continuous:
$$|\mu_t(u) - \mu_t(v)| \le |u - v|, \quad \forall u, v \in [0, 1], \forall t \in [T].$$
- The environment is piecewise stationary with reward distributions varying at most Γ times[1]:
$$\sum_{t=1}^{T} \mathbb{1}[\exists a \in [0, 1], \mathcal{D}_t(a) \neq \mathcal{D}_{t-1}(a)] \le \Gamma.$$

[1]We denote by $\mathbb{1}[\cdot]$ the indicator function.

4 METHOD

The main idea of our methods is to deal with the continuous arms by discretizing the arm set and handle non-stationary reward distributions via dynamic mean estimator. To ease understanding, we first present a basic method using static discretization and then propose a refined method with adaptive discretization.

4.1 Static Discretization

Let ρ denote the discretization resolution. We first discretize the continuous arm set $[0, 1]$ into a set of equally-spaced arms:

$$\mathcal{A} = \{\rho k \mid k = 1, \ldots, \lfloor 1/\rho \rfloor\}. \tag{1}$$

We refer to each arm in $\mathcal{A}$ as active arm and only pull active arms during the T rounds. The intuition behind the discretization in (1) is two folds. On the one hand, the number of active arms only grows inverse-linearly with the discretization resolution and thus the sampling cost on the active arm set can be controlled. On the other hand, for each inactive arm $a \in [0, 1] - \mathcal{A}$, its mean reward can be approximated by that of certain active arm $a' \in \mathcal{A}$ and the approximate error can be bounded by the discretization resolution, since by the definition of $\mathcal{A}$ and the Lipschitz continuity of mean reward function, there must exist an active arm $a' \in \mathcal{A}$ such that

$$|\mu_t(a) - \mu_t(a')| \le |a - a'| \le \rho.$$

After discretization, we associate each active arm $a \in \mathcal{A}$ with two variables, namely $\hat{\mu}_t(a)$ and $\xi_t(a)$, which denote the estimated mean of arm a's reward distribution in round t and the confidence width of the estimation, respectively. In each round $t = 1, \ldots, T$, following the principle of "optimism in the face of uncertainty" [3], we select the active arm with the maximum sum of estimated mean reward and confidence width:

$$I_t = \arg\max_{a\in\mathcal{A}} \hat{\mu}_t(a) + \xi_t(a). \tag{2}$$

As the reward distribution of each arm is non-stationary, classical mean estimators that directly average the historical rewards do not apply. To address this problem, we adopt dynamic mean estimator [15], which can converge to the time-varying mean reward. The main idea of dynamic mean estimator is to assign more weights to recent rewards and less weights to old rewards when averaging the historical rewards. Specifically, let $w_t(s)$ denote the weight assigned to the reward in round s. We estimate the mean reward of each active arm $a \in \mathcal{A}$ in round t by[2]

$$\hat{\mu}_t(a) = \frac{\sum_{s=1}^{t-1} w_t(s)\mathbb{1}[I_s = a] r_s(a)}{\sum_{s=1}^{t-1} w_t(s)\mathbb{1}[I_s = a]}. \tag{3}$$

We consider two types of weight configurations termed as hard-drop and soft-drop, respectively. In the hard-drop configuration, we simply drop the historical rewards before round $t - \lambda$:

$$w_t(s) = \begin{cases} 1, & s \ge t - \lambda \\ 0, & s < t - \lambda \end{cases} \tag{4}$$

where $\lambda \in \mathbb{N}$ is the drop threshold. By contrast, in the soft-drop configuration, all historical rewards are preserved and their weights constitute a geometric series:

$$w_t(s) = \gamma^{t-s-1} \tag{5}$$

[2]We make the convention that $0/0 = +\infty$.

Algorithm 1 Static Discretization with Dynamic Mean Estimator (SD^{2}ME)

Require: discretization resolution ρ, drop threshold λ, drop discount γ

Construct a set of active arms $\mathcal{A} = \{\rho k \mid k = 1, \ldots, \lfloor 1/\rho \rfloor\}$
Initialize $\hat{\mu}_1(a) = 0$ and $\xi_1(a) = +\infty$ for each $a \in \mathcal{A}$
for $t = 1, 2, \ldots, T$ **do**
  Select an arm I_t according to (2)
  Pull I_t and receive a reward $r_t(I_t)$
  switch weight configuration **do**
    case hard-drop**:** Compute $w_t(s), s = 1, \ldots, t-1$ by (4)
    case soft-drop**:** Compute $w_t(s), s = 1, \ldots, t-1$ by (5)
  end switch
  Update $\mu_{t+1}(a)$ and $\xi_{t+1}(a)$ for each $a \in \mathcal{A}$ by (3) & (7)
end for

where $\gamma \in (0, 1]$ is the drop discount.

It remains to design the confidence width $\xi_t(a)$. To this end, we resort to the Hoeffding's inequality [17]. Specifically, it can be proved that[3] in most of rounds, the following inequality holds with a high probability

$$|\hat{\mu}_t(a) - \mu_t(a)| \leq \sqrt{\frac{\log \sum_{s=1}^{t-1} w_t(s)}{\sum_{s=1}^{t-1} w_t(s) \mathbb{1}[I_s = a]}}, \quad \forall a \in \mathcal{A}. \tag{6}$$

Thus, we set the confidence width of each active arm $a \in \mathcal{A}$ as

$$\xi_t(a) = \sqrt{\frac{\log \sum_{s=1}^{t-1} w_t(s)}{\sum_{s=1}^{t-1} w_t(s) \mathbb{1}[I_s = a]}}. \tag{7}$$

We summarize the whole procedure in Algorithm 1, which is referred to as Static Discretization with Dynamic Mean Estimator (SD^{2}ME) and enjoys the following theoretical guarantee.

Theorem 4.1. *The dynamic regret of the SD^{2}ME method with* $\lambda = \lfloor 6^{1/4}(T/\Gamma)^{3/4} \rfloor$, $\gamma = 1 - 6^{-1/4}(\Gamma/T)^{3/4}$, *and*

$$\rho = \begin{cases} \sqrt[3]{6/\lambda}, & \text{hard-drop weight configuration} \\ \sqrt[3]{6(1-\gamma)}, & \text{soft-drop weight configuration} \end{cases} \tag{8}$$

satisfies $\mathbb{E}[\mathrm{DR}(T)] \leq O(\Gamma^{1/4} T^{3/4})$.

Remark 4.1. The pseudo-code in Algorithm 1 is not optimized for runtime but for ease of exposition and understanding. In fact, for each active arm $a \in \mathcal{A}$, both the estimated mean reward $\hat{\mu}_t(a)$ and the confidence width $\xi_t(a)$ can be computed in an online manner with $O(1)$ time complexity per round. Specifically, let $n_t(a) = \sum_{s=1}^{t-1} w_t(s) \mathbb{1}[I_s = a]$, $R_t(a) = \sum_{s=1}^{t-1} w_t(s) \mathbb{1}[I_s = a] r_s(a)$, and $W_t = \sum_{s=1}^{t-1} w_t(s)$. We have $\hat{\mu}_t(a) = R_t(a)/n_t(a)$ and $\xi_t(a) = \sqrt{(\log W_t)/n_t(a)}$. In the hard-drop weight configuration, we can maintain two queues of length λ (one for storing the rewards $r_s(a), s = t-\lambda, \ldots, t-1$ and the other for storing the indicators $\mathbb{1}[I_s = a], s = t-\lambda, \ldots, t-1$) and update $n_t(a)$, $R_t(a)$, and W_t incrementally as

$$\begin{aligned} n_{t+1}(a) &= n_t(a) - \mathbb{1}[I_{t-\lambda} = a] + \mathbb{1}[I_t = a] \\ R_{t+1}(a) &= R_t(a) - \mathbb{1}[I_{t-\lambda} = a] r_{t-\lambda}(a) + \mathbb{1}[I_t = a] r_t(a) \\ W_{t+1} &= \min(W_t + 1, \lambda). \end{aligned}$$

[3]The proof is postponed to the next section.

In the soft-drop weight configuration, the values of $n_t(a)$, $R_t(a)$, and W_t can be also updated on the fly:

$$\begin{aligned} n_{t+1}(a) &= \gamma n_t(a) + \mathbb{1}[I_t = a] \\ R_{t+1}(a) &= \gamma R_t(a) + \mathbb{1}[I_t = a] r_t(a) \\ W_{t+1} &= \gamma W_t + 1. \end{aligned}$$

4.2 Adaptive Discretization

While SD^{2}ME is simple, it cannot utilize the rewards received on the fly to refine the discretization, which hinders its effectiveness in dealing with complex mean reward functions. To overcome this limitation, we propose an adaptive discretization based method, where the discretization is not determined in advance but adaptively adjusted throughout the T rounds.

Specifically, we maintain a time-varying active arm set $\mathcal{A}_t \subset [0, 1]$, which is initialized to be empty and updated in each round. Similarly to SD^{2}ME, each active arm $a \in \mathcal{A}_t$ is associated with a time-varying estimated mean reward $\hat{\mu}_t(a)$ and a corresponding confidence width $\xi_t(a)$. While we compute $\hat{\mu}_t(a)$ in the same way as that of SD^{2}ME, we slightly modify the design of $\xi_t(a)$ in (7) to[4]:

$$\xi_t(a) = \sqrt{\frac{\log(2t^{1.5}/\delta^{0.5})}{\sum_{s=1}^{t-1} w_t(s) \mathbb{1}[I_s = a]}} \tag{9}$$

where $\delta \in (0, 1)$ controls the failure probability of the concentration inequality in (6).

An active arm $a \in \mathcal{A}_t$ and its associated confidence width $\xi_t(a)$ define a covering interval $[a - \xi_t(a), a + \xi_t(a)]$. By the Lipschitz continuity of the mean reward function and the concentration inequality of the dynamic mean estimator in (6), the mean reward $\mu_t(a')$ of each arm a' in the covering interval $[a - \xi_t(a), a + \xi_t(a)]$ can be approximated by the estimated mean reward $\hat{\mu}_t(a)$ of the active arm a as

$$|\hat{\mu}_t(a) - \mu_t(a')| \leq |\hat{\mu}_t(a) - \mu_t(a)| + |\mu_t(a) - \mu_t(a')| \leq 2\xi_t(a).$$

Thus, in each round, we can only pull active arms if the arm set $[0, 1]$ can be covered by the union of the covering interval of each active arm, i.e.,

$$[0, 1] \subseteq \cup_{a \in \mathcal{A}_t} [a - \xi_t(a), a + \xi_t(a)]. \tag{10}$$

Our adaptive discretization based method is designed such that (10) is guaranteed throughout the T rounds. Specifically, in each round $t = 1, \ldots, T$, we first check whether $[0, 1] \subseteq \cup_{a \in \mathcal{A}_{t-1}} [a - \xi_t(a), a + \xi_t(a)]$ holds. If yes, (10) can be guaranteed by simply setting $\mathcal{A}_t = \mathcal{A}_{t-1}$. Otherwise, we pick an arbitrary arm a' from the uncovered region $[0, 1] - \cup_{a \in \mathcal{A}_{t-1}} [a - \xi_t(a), a + \xi_t(a)]$ and add it into the active arm set, i.e., $\mathcal{A}_t = \mathcal{A}_{t-1} \cup \{a'\}$. Then, we set the estimated mean reward and the confidence width of this newly added active arm as zero and infinity, respectively. By doing so, the arm set $[0, 1]$ becomes a subset of the covering interval of a' and thus (10) holds.

With (10) guaranteed, we can only pull arm from the active arm set $\mathcal{A}_t$. In each round $t = 1, \ldots, T$, we first select an active arm in a similar way to that of SD^{2}ME, with the only difference of doubling

[4]This modification is essential in deriving a dynamic regret bound that holds with high probability rather than in expectation.

Algorithm 2 Adaptive Discretization with Dynamic Mean Estimator (AD^{2}ME)

Require: drop threshold λ, drop discount γ
1: Initialize the active arm set as $\mathcal{A}_0 = \varnothing$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: **if** $[0,1] \subseteq \cup_{a\in\mathcal{A}_{t-1}}[a - \xi_t(a), a + \xi_t(a)]$ **then**
4: Set $\mathcal{A}_t = \mathcal{A}_{t-1}$
5: **else**
6: Pick an arm $a' \in [0,1] - \cup_{a\in\mathcal{A}_{t-1}}[a - \xi_t(a), a + \xi_t(a)]$
7: Set $\mathcal{A}_t = \mathcal{A}_{t-1} \cup \{a'\}$, $\hat{\mu}_t(a') = 0$, and $\xi_t(a') = +\infty$
8: **end if**
9: Select an arm I_t according to (11)
10: Pull I_t and receive a reward $r_t(I_t)$
11: **switch** weight configuration **do**
12: **case** hard-drop**:** compute $w_t(s), s = 1, \ldots, t-1$ by (4)
13: **case** soft-drop**:** compute $w_t(s), s = 1, \ldots, t-1$ by (5)
14: **end switch**
15: Update $\mu_{t+1}(a)$ and $\xi_{t+1}(a)$ for each $a \in \mathcal{A}_t$ by (3) & (9)
16: **end for**

the confidence width:

$$I_t = \arg\max_{a\in\mathcal{A}_t} \hat{\mu}_t(a) + 2\xi_t(a). \tag{11}$$

Then, we pull the selected arm and receive a reward $r_t(I_t)$. Finally, we update the estimated mean reward and the confidence width of each active arm according to (3) and (9).

The whole procedure is summarized in Algorithm 2. We call this algorithm as Adaptive Discretization with Dynamic Mean Estimator (AD^{2}ME). We derive a high probability dynamic regret bound for AD^{2}ME as follows, which is stronger than the expected dynamic regret bound of SD^{2}ME.

Theorem 4.2. *With probability at least $1-\delta$, the dynamic regret of AD^{2}ME with*

$$\begin{cases} \lambda = \lfloor 2(T/3\Gamma)^{3/4} \rfloor, & \textit{hard-drop weight configuration} \\ \gamma = 1 - (3\Gamma/T)^{3/4}, & \textit{soft-drop weight configuration} \end{cases} \tag{12}$$

satisfies $\mathrm{DR}(T) \le O(\Gamma^{1/4}T^{3/4})$.

Remark 4.2. In AD^{2}ME, both the estimated mean reward $\hat{\mu}_t(a)$ and the confidence width $\xi_t(a)$ of each active arm a can be also computed online in the same way as Remark 4.1. Furthermore, we would like to point out that the operation of checking $[0,1] \subseteq \cup_{a\in\mathcal{A}_{t-1}}[a-\xi_t(a), a+\xi_t(a)]$ at Step 3 of AD^{2}ME can be performed with constant time complexity in each round. Specifically, we can maintain a sorted and doubly linked list $L = \ell_1 \leftrightarrow \ell_2 \leftrightarrow \cdots \leftrightarrow \ell_{|\mathcal{A}_t|}$ with $\ell_1 < \ell_2 < \cdots < \ell_{|\mathcal{A}_t|}$ to store the active arm set $\mathcal{A}_t$. In the first round, we initialize $L = \ell_1$ and set $\xi_1(\ell_1) = +\infty$. Then, in each round $t = 2, \ldots, T$, let ℓ_k be the node storing the arm pulled at round $t-1$, i.e., $\ell_k = I_{t-1}$. By the definition of the confidence width, all active arms $a \in \mathcal{A}_{t-1}$ except for I_{t-1} satisfy $\xi_t(a) \ge \xi_{t-1}(a)$. Thus, if the linked list is sorted before round t and $[0,1] \subseteq \cup_{a\in\mathcal{A}_{t-1}}[a - \xi_{t-1}(a), a + \xi_{t-1}(a)]$, we only need to check whether any of the following two inequalities holds.

$$\ell_k - \xi_t(\ell_k) > \begin{cases} 0, & \ell_k \text{ is the head node;} \\ \ell_{k-1} + \xi_t(\ell_{k-1}), & \ell_k \text{ is not the head node.} \end{cases}$$

$$\ell_k + \xi_t(\ell_k) < \begin{cases} 1, & \ell_k \text{ is the tail node;} \\ \ell_{k+1} - \xi_t(\ell_{k+1}), & \ell_k \text{ is not the tail node.} \end{cases}$$

If no inequality holds, we simply keep the linked list L unchanged, so that L is sorted after round t and $[0,1] \subseteq \cup_{a\in\mathcal{A}_t}[a - \xi_t(a), a + \xi_t(a)]$. Otherwise, taking the case of $\ell_k + \xi_t(\ell_k) < \ell_{k+1} - \xi_t(\ell_{k+1})$ as an example, we first pick a new active arm $a' \in \big(\ell_k + \xi_t(\ell_k), \ell_{k+1} - \xi_t(\ell_{k+1})\big)$ and create a new node ℓ' to store a'. Then, we update the pointers of ℓ_k, ℓ_{k+1}, and ℓ' to $\ell_k \leftrightarrow \ell' \leftrightarrow \ell_{k+1}$ so as to ensure L is sorted after round t. Finally, we set $\xi_t(a') = +\infty$ to guarantee $[0,1] \subseteq \cup_{a\in\mathcal{A}_t}[a - \xi_t(a), a + \xi_t(a)]$.

5 ANALYSIS

Due to space limitation, we only prove Theorem 4.1 and Theorem 4.2 for the hard-drop weight configuration and the proofs for the soft-drop weight configuration can be done in a similar way.

5.1 Proof of Theorem 4.1

We first partition the T rounds into Γ epochs such that in each epoch the reward distributions are stationary:

$$[1, T] = [\tau_1, \tau_2) \cup [\tau_2, \tau_3) \cup \ldots \cup [\tau_\Gamma, T+1)$$

where $\tau_i, i = 1, \ldots, \Gamma$ denotes the i-th change point of the reward distributions, i.e., $\tau_1 < \cdots < \tau_\Gamma$ and $\exists a \in [0,1], \mathcal{D}_{\tau_i} \ne \mathcal{D}_{\tau_i - 1}(a)$. For notational convenience, we additionally define $\tau_{\Gamma+1} = T + 1$.

Fix an epoch i. Let $O_i = \{a \in \mathcal{A} \mid \mu_{\tau_i}(a) = \max_{a'\in\mathcal{A}} \mu_{\tau_i}(a')\}$ be all optimal active arms during the epoch $[\tau_i, \tau_{i+1})$ and $o_i \in O_i$ be an optimal active arm. We denote the mean reward gap between an active arm a and the optimal active arm o_i by $\Delta_i(a) = \mu_{\tau_i}(o_i) - \mu_{\tau_i}(a)$. Define $\mathcal{A}_\rho = \{a \in \mathcal{A} \mid \Delta_i(a) \ge \rho\}$. We have

$$\begin{aligned} \sum_{t=\tau_i}^{\tau_{i+1}-1} [\mu_t(o_i) - \mu_t(I_t)] &= \sum_{t=\tau_i}^{\tau_{i+1}-1} \Delta_i(I_t) \le \lambda + \sum_{t=\tau_i+\lambda}^{\tau_{i+1}-1} \Delta_i(I_t) \\ &\le \lambda + (\tau_{i+1} - \tau_i)\rho + \sum_{a\in\mathcal{A}_\rho} \Delta_i(a) \sum_{t=\tau_i+\lambda}^{\tau_{i+1}-1} \mathbb{1}[I_t = a]. \end{aligned} \tag{13}$$

Fix an arm $a \in \mathcal{A}_\rho$. Define $\Lambda_i(a) = 4\Delta_i^{-2}(a)\log(\lambda+1)$ and $n_t(a) = \sum_{s=1}^{t-1} w_t(s)\mathbb{1}[I_s = a] = \sum_{s=t-\lambda}^{t-1} \mathbb{1}[I_s = a]$. On one hand, we have

$$\sum_{t=\tau_i+\lambda}^{\tau_{i+1}-1} \mathbb{1}[I_t = a, n_t(a) < \Lambda_i(a)] \le \lfloor (\tau_{i+1} - \tau_i)/\lambda \rfloor \Lambda_i(a). \tag{14}$$

On the one hand, by the arm selection rule in (2) and the Hoeffding's inequality [17], for $t \in [\tau_i + \lambda, \tau_{i+1})$ we have

$$\begin{aligned} &\mathbb{E}\big[\mathbb{1}[I_t = a, n_t(a) \ge \Lambda_i(a)]\big] = \Pr[I_t = a, n_t(a) \ge \Lambda_i(a)] \\ &\le \Pr[\hat{\mu}_t(a) + \xi_t(a) \ge \hat{\mu}_t(o_i) + \xi_t(o_i), 2\xi_t(a) < \mu_t(o_i) - \mu_t(a)] \\ &\le \Pr[\hat{\mu}_t(a) > \mu_t(a) + \xi_t(a)] + \Pr[\hat{\mu}_t(o_i) < \mu_t(o_i) - \xi_t(o_i)] \\ &\le \exp[-2n_t(a)\xi_t^2(a)] + \exp[-2n_t(o_i)\xi_t^2(o_i)] = 2/\lambda^2. \end{aligned} \tag{15}$$

Combining (13)–(15) and $\Lambda_i(a) = 4\Delta_i^{-2}(a)\log(\lambda+1)$ gives

$$\begin{aligned} &\mathbb{E}\left[\sum_{t=\tau_i}^{\tau_{i+1}-1} [\mu_t(o_i) - \mu_t(I_t)]\right] \\ &\le \lambda + (\tau_{i+1} - \tau_i)\rho + (\tau_{i+1} - \tau_i) \sum_{a\in\mathcal{A}_\rho} \left[\frac{4\log(\lambda+1)}{\lambda\rho} + \frac{2}{\lambda^2}\right]. \end{aligned} \tag{16}$$

Let o_i^* be an optimal arm during epoch i: $\mu_t(o_i^*) = \mu_t(a_t^*), \forall t \in [\tau_i, \tau_{i+1})$. By the definition of the active arm set $\mathcal{A}$ in (1), there must exist $a' \in \mathcal{A}$ such that $|o_i^* - a'| \le \rho$. Thus, we have

$$\mathbb{E}\left[\sum_{t=\tau_i}^{\tau_{i+1}-1}[\mu_t(a_t^*) - \mu_t(o_i)]\right] = \mathbb{E}\left[\sum_{t=\tau_i}^{\tau_{i+1}-1}[\mu_t(o_i^*) - \mu_t(o_i)]\right] \quad (17)$$

$$\le \mathbb{E}\left[\sum_{t=\tau_i}^{\tau_{i+1}-1}[\mu_t(o_i^*) - \mu_t(a')]\right] \le \mathbb{E}\left[\sum_{t=\tau_i}^{\tau_{i+1}-1}|o_i^* - a'|\right] \le (\tau_{i+1} - \tau_i)\rho.$$

Adding (16) and (17) and summing over epoch $i = 1, \ldots, \Gamma$ gives

$$\mathbb{E}[\mathrm{DR}(T)] = \mathbb{E}\left[\sum_{t=1}^{T}[\mu_t(a_t^*) - \mu_t(I_t)]\right] \le \lambda\Gamma + 2\rho T + \frac{6\log(\lambda+1)T}{\lambda\rho^2}.$$

5.2 Proof of Theorem 4.2

Following the previous subsection, we partition the T rounds into Γ stationary epochs: $[1, T] = [\tau_1, \tau_2) \cup [\tau_2, \tau_3) \cup \cdots \cup [\tau_\Gamma, \tau_{\Gamma+1})$. Let $\mathcal{T}$ be a subset of the T rounds: $\mathcal{T} = \cup_{i=1}^{\Gamma}[\tau_i+\lambda, \tau_{i+1})$. We first introduce the following lemma, which is a straightforward consequence of the Hoeffding's inequality [17] and the union bound [10].

Lemma 1. *With probability at least $1-\delta$, for all rounds $t \in \mathcal{T}$ and all active arms $a \in \mathcal{A}_t$, $|\hat{\mu}_t(a) - \mu_t(a)| \le \xi_t(a)$.*

Based on this lemma, we then show that the confidence width of each active arm is lower bounded by its mean reward gap.

Lemma 2. *Let o_i^* be an optimal arm during epoch i, i.e., $\mu_{\tau_i}(o_i^*) = \max_{a\in[0,1]}\mu_{\tau_i}(a)$ and denote the mean reward gap of each arm a by $\Delta_i(a) = \mu_{\tau_i}(o_i^*) - \mu_{\tau_i}(a)$. For all epochs $i = 1, \ldots, \Gamma$, we have*

$$\pi\xi_t(a) \ge \Delta_i(a), \quad \forall t \in [\tau_i + 2\lambda, \tau_{i+1}), \forall a \in \mathcal{A}_t.$$

Proof. The proof is postponed to appendix. □

We are now ready to prove Theorem 4.2. Fix an epoch i. We partition $[\tau_i+2\lambda, \tau_{i+1})$ into $[v_0, v_1) \cup [v_1, v_2) \cup \cdots \cup [v_J, \tau_{i+1})$ with $J = \lfloor(\tau_{i+1} - \tau_i - 2\lambda)/\lambda\rfloor$ and $v_j = \tau_i + (2+j)\lambda$, $j = 0, \ldots, J$. Define $n_t(a) = \sum_{s=t-\lambda}^{t-1}\mathbb{1}[I_s = a]$. By Lemma 2, for each $j = 1, \ldots, J$ and $a \in \mathcal{A}_{v_j}$, we have $\Delta_i(a) \le \pi\xi_{v_j} \le \pi\sqrt{\log(2T^{1.5}/\delta^{0.5})/n_{v_j}(a)}$. It follows that $\sum_{t=v_{j-1}}^{v_j-1}[\mu_t(a_t^*) - \mu_t(I_t)] = \sum_{a\in\mathcal{A}_{v_j}:n_{v_j}(a)>0} n_{v_j}(a)\Delta_i(a) \le \pi\sqrt{\lambda^{4/3}\log(2T^{1.5}/\delta^{0.5})}$. Summing this over $j = 1, \ldots, J$ gives

$$\sum_{t=\tau_i}^{\tau_{i+1}-1}[\mu_t(a_t^*) - \mu_t(I_t)] \le 3\lambda + \lambda^{-1/3}(\tau_{i+1} - \tau_i)\pi\sqrt{\log(2T^{1.5}/\delta^{0.5})}.$$

Further summing this inequality over $i = 1, \ldots, \Gamma$ leads to

$$\mathrm{DR}(T) = \sum_{i=1}^{\Gamma}\sum_{t=\tau_i}^{\tau_{i+1}-1}[\mu_t(a_t^*) - \mu_t(I_t)] \le 3\Gamma\lambda + \frac{\pi T\sqrt{\log(2T^{1.5}/\delta^{0.5})}}{\lambda^{1/3}}.$$

6 APPLICATION: CANDIDATE GENERATION

As a case application, we show how our methods can be applied to tune the hyperparameter of the vector-based candidate generation algorithm. In industrial retrieval scenarios such as web search, e-commerce recommendation, and online advertising, a common practice for balancing the trade-off between effectiveness and efficiency is to rank items by a two-phased strategy, where a large corpus is first filtered by a candidate generation algorithm and the generated candidate items are then sorted by a ranking method. Taking web search as an example, the goal of candidate generation algorithm is to preserve relevant items w.r.t. user's query while discarding irrelevant items. Let v_q and v_d denote the embedding vector of a query q and a document d, respectively. Then, the relevance between q and d can be measured via the cosine similarity $\frac{\langle v_q, v_d\rangle}{\|v_q\|\|v_d\|}$ and documents with relevance exceeding the truncation threshold β are designated as candidate items: $C_q = \left\{d\colon \frac{\langle v_q, v_d\rangle}{\|v_q\|\|v_d\|} \ge \beta\right\}$.

The key here is to tune the hyperparameter $\beta \in [0, 1]$. On the one hand, large β leads to overall high quality of candidate items but may prevent some user-desired items from entering into the ranking phase. On the other hand, small β allows a large number of candidate items but may introduce irrelevant items into the ranking phase. Thus, the tuning of β is non-trivial and requires principled approaches. Since the whose corpus as well as user preference varies with time, the optimal value of β is also not fixed. Furthermore, it is reasonable to believe that similar values of β lead to similar search performance. Thus, we can formulate the tuning of β as a non-stationary continuum-armed bandits problem with each possible configuration of β being an arm.

7 EXPERIMENT

In this section, we provide extensive experimental results to demonstrate the effectiveness and efficiency of our methods for online hyperparameter optimization.

7.1 Setup

7.1.1 Dataset. We conduct experiments on the Ali-Display-Ad dataset [14, 44], which was collected from the online advertising logs of taobao for 8 days and has been publicly released[5]. The dataset consists of 26 million records of interactions between 1061768 users and 846811 ads. Each record comprises the identities of the served user and the displayed ad in the interaction, as well as the time stamp of the interaction and the user's click feedback on the ad. The basic information of both users and ads is also provided in the dataset. Specifically, each user is associated with 8 profile features such as age level, shopping level, and gender. Each ad is described by 5 features including the identity of the advertiser, the campaign of the ad, the category, the brand, and the price of the advertised commodity. All the 13 features except for the price of the advertised commodity are categorical. Following common industrial practice, we also convert the price of the advertised commodity into a categorical feature by splitting the prices of all commodities into 5 equal-sized bins. As the Ali-Display-Ad dataset is sufficiently large, we simply drop the records with missing values of any feature.

7.1.2 Experimental Framework. As described in the previous section, to run the vector-based candidate generation algorithm on the Ali-Display-Ad dataset, an encoding model that can transform the raw features of both user and ad into embedding vectors is required. We take DSSM [18] as the encoding model, which is widely used in online advertising and consists of two sub-networks, one for

[5] https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

encoding user's features and the other for encoding ad's features. Each sub-network is comprised of four fully-connected layers with 500, 300, 300, and 128 neurons, respectively and all fully-connected layers are followed by the *tanh* activation function. Following the approach in [18], we train the DSSM model as follows: We first choose a user u, an ad a^+ that was clicked by the user and four ads $a_1^-, \ldots, a_4^-$ that were not clicked by the user uniformly at randomly from the Ali-Display-Ad dataset. Then, the raw features of user and ads are fed into the corresponding sub-networks to obtain embedding vectors of the user and ads $v_u, v_{a^+}, v_{a_1^-}, \ldots, v_{a_4^-}$. The relevance between user and ad is measured via the cosine distance and the loss is defined as $\ell = -\log \frac{\mathcal{M}(v_u, v_{a^+})}{\mathcal{M}(v_u, v_{a^+}) + \sum_{i=1}^4 \mathcal{M}(v_u, v_{a_i^-})}$ where $\mathcal{M}(x, y) = \frac{\langle x, y \rangle}{\|x\| \|y\|}$ denotes the cosine of the angle between x and y. We train the DSSM model with batch size 1024 by the Adam optimizer [22] for 10000 epochs.

Equipped with the trained DSSM model, we are now ready to construct an online hyperparameter optimization framework, which proceeds in a sequence of rounds $t = 1, \ldots, T$. In each round t, we first select a user u_t and collect the ads $A_t = A_t^+ \cup A_t^-$ that were displayed to this user, where A_t^+ and A_t^- denote the set of clicked and unclicked ads, respectively. To comply with the real-world non-stationary environment, the users are not sampled uniformly at random but picked in order according to the most frequent hour in their interaction time stamps. Then, the trained DSSM model is employed to output the embedding vectors of both the user and ads, denoted as v_{u_t} and $\{v_a : a \in A_t^+ \cup A_t^-\}$. Next, we invoke the examined hyperparameter optimization algorithm, which outputs a truncation threshold β_t. Intuitively, we wish the truncation threshold could preserve as many clicked ads while filtering out as many unclicked ads as possible. Thus, we measure the quality of the truncation threshold via F-score [5], which is the harmonic mean of precision and recall, defined as follows

$$\mathrm{F}_t = \frac{\mathrm{precision}_t \cdot \mathrm{recall}_t}{\mathrm{precision}_t + \mathrm{recall}_t} = \frac{2 \sum_{a \in A_t^+} \mathbb{1}\{\mathcal{M}(v_{u_t}, v_a) \geq \beta_t\}}{|A_t^+| + \sum_{a \in A_t} \mathbb{1}\{\mathcal{M}(v_{u_t}, v_a) \geq \beta_t\}}.$$

Finally, we reveal F_t to the examined hyperparameter optimization algorithm as its instantaneous reward. The overall performance of the examined hyperparameter optimization algorithm is measured by its cumulative rewards over $T = 10000$ rounds: $\sum_{t=1}^T \mathrm{F}_t$.

7.1.3 Baselines. We compare our methods with the following hyperparameter optimization algorithms:

- **Grid Search.** We split the T rounds into two equal-length phases, one for exploration and the other for exploitation. Specifically, during the first $T/2$ rounds, ten evenly spaced points over the unit interval $[0, 1]$ are explored in a round-robin fashion. Then, the one that achieves the maximum average F-score is exploited in the last $T/2$ rounds.
- **Random Search.** We sample ten points uniformly at random from the unit interval $[0, 1]$. Similarly to grid search, the first $T/2$ rounds are taken to explore the sampled points and the empirically best one is used in last $T/2$ rounds.
- **Bayesian Optimization.** Following [24], we adopt Gaussian process as the surrogate model and examine two covariance functions, namely squared exponential (SE) function and Matérn function. To make fair comparison, we take upper confidence bound (UCB) as the acquisition function. In each round t, Bayesian Optimization needs to compute the inverse covariance matrix, whose time complexity grows cubicly with t. To control the computation time within a reasonable limit, we train Bayesian optimization during the first $T/10$ rounds and use its output in the remaining rounds.
- **Hyperband.** Following [26], we set the proportion of candidate hyperparameter configurations discarded in each round as 1/3. As Hyperband is originally designed for offline scenarios, we take the first $T/2$ rounds to train Hyperband and use its output in the last $T/2$ rounds. Accordingly, the maximum amount of resource that can be allocated to a single hyperparameter configuration is set as $\frac{T}{2(\ln T + 1)^2}$.

We configure our methods according to Theorem 4.1 and Theorem 4.2. While the actual non-stationarity Γ remains unknown in the experiments, we simply replace Γ with an estimated non-stationarity $\tilde{\Gamma} = 10$ in (8) and (12).

7.1.4 Research Questions. We aim to answer the following research questions via experiments:

(1) How does our methods perform in comparison with the baseline algorithms?
(2) To what extent does the mismatch between the estimated non-stationarity $\tilde{\Gamma}$ and the actual non-stationarity Γ affect the performance of our methods?
(3) How about the efficiency of our methods compared to the baseline algorithms?

7.2 Results and Analysis

We start with the first research question and run our methods and each baseline algorithm in the online hyperparameter optimization experimental framework ten times. The average performance as well as the standard deviation of each algorithm is listed in Table 1. We can observe that the two brute-force algorithms grid search and random search behave the worst. This is because the two algorithms lack guidance in exploring on the unit interval $[0, 1]$ and hence waste too much time on bad truncation thresholds. This drawback is addressed by the Bayesian optimization and Hyperband algorithms where the exploration process is guided by historical observations of previously tried truncation thresholds. As can be seen, such informative exploration indeed translates into improved cumulative F-score. Nevertheless, we can also see that our methods with both soft-drop and hard-drop weight configurations achieve significantly higher cumulative F-score than the Bayesian optimization and Hyperband algorithms. This is expected as online advertising is a typical non-stationary scenario where the overall click-through-rate varies with time period and so does the optimal truncation threshold, while Bayesian optimization and Hyperband are designed for stationary scenarios assuming the existence of a global optimal truncation threshold over all time periods. By contrast, both of our methods SD^{2}ME and AD^{2}ME take the non-stationarity into account and track the time-varying optimism truncation threshold via dynamic mean estimator. Taking a closer look at the performance of our methods, there is clear performance improvement of AD^{2}ME over SD^{2}ME, which validates the

Table 1: Performance comparison between our methods and the baseline algorithms on the Ali-Display-Ad dataset. * denotes significant improvement over all baseline algorithms at $p < 0.01$. The unit of computation time is second.

Algorithm	Cumulative F-score	Computation Time
Grid Search	3396 ± 0.0	517
Random Search	3467 ± 64.9	528
Hyperband	3550 ± 49.2	546
Bayesian-Matérn	3564 ± 14.5	43426
Bayesian-SE	3533 ± 20.7	37283
SD^{2}ME-hard	3673 ± 5.9*	612
SD^{2}ME-soft	3671 ± 4.3*	568
AD^{2}ME-hard	3714 ± 15.2*	745
AD^{2}ME-soft	3720 ± 19.8*	671

superiority of adaptive discretization for online hyperparameter optimization.

We then answer the second research question by running our methods with different estimated non-stationarity ranging from 5 to 25. Except for the estimated non-stationarity, the other experimental setup is the same as that in the above investigation of the first research question. We plot the performance of our methods under each estimated non-stationarity $\tilde{\Gamma} = 5, 10, 15, 20, 25$ in Figure 1. The curve of cumulative F-score versus estimated non-stationarity varies slowly, which reveals that our methods are relatively robust to the mismatch between the estimated non-stationarity and the actual non-stationarity of the experimental environment.

Finally, we investigate the last research question. To this end, we record the overall computation time of each algorithm, which is also listed in Table 1 for saving space. The first observation is that Bayesian optimization takes much longer time than the other algorithms. This is due to the time-consuming operation of inverting the covariance matrix, which makes the overall time complexity of Bayesian optimization scale with $O(T^3)$. By contrast, the other algorithms all perform constant-time operations in each round, leading to an overall time complexity of $O(T)$. Further comparing our methods with grid search, random search and Hyperband, we see that the computation time of our methods are slightly higher, which is acceptable in light of the improvement on online performance. We can also observe that within our methods, soft-drop weight configuration is more efficient than hard-drop weight configuration, which is intuitive as the latter needs to additionally maintain queues as discussed in Remark 4.1.

7.3 Online A/B Testing

The candidate generation is an important stage for search, recommendation, and advertising on e-commerce platforms. In practice, several candidate generation algorithms are always kept in online service, each providing a set of candidate items for the subsequent ranking stage. Among them, the vector-based candidate generation algorithm has attracted much attention recently. As discussed in Section 6, the performance of the vector-based candidate generation algorithm heavily depends on the truncation threshold. A common practice is to manually choose the truncation threshold based on historical data, which may fail in the non-stationary environment.

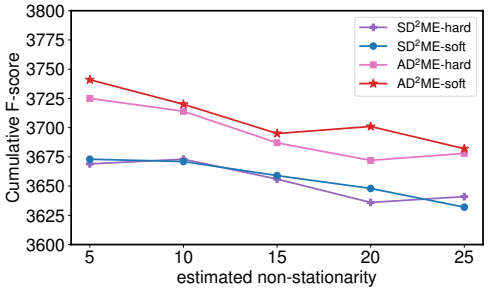

Figure 1: Performance of our methods with different $\tilde{\Gamma}$

We deployed our methods on an international e-commerce platform, to optimize the truncation threshold of the vector-based candidate generation algorithm. In this platform, the queries are categorized into several disjointed classes according to their natural categories. Thus, we run our methods for each category separately to obtain meticulous thresholds. Our goal is to maximize the user conversion rate, which is defined as the ratio of the buyer number to the visitor number in the platform. Moreover, to decrease the gap between the conversion rate of users and that of items, we introduce a novel reward formulation where every arm benefits from the conversion of the same user-query pair in the future 10 minutes time window. The results of online A/B testing demonstrate that compared to the manually fine-tuning, our methods gain 2.24% more user conversion rate and 2.38% more GMV per visitor over 14 days.

8 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel bandit-type formulation for online hyperparameter optimization, where each hyperparameter configuration is viewed as an arm and the algorithmic performance is modelled as reward. We name the proposed bandit formulation as non-stationary continuum-armed bandits, for which we develop two effective and efficient methods. Theoretically, we prove that both of our methods can achieve sub-linear dynamic regret bounds, which, to the best of our knowledge, is the first theoretical result for continuum-armed bandits in non-stationary environments and may be of independent interest. We also empirically demonstrate the effectiveness and efficiency of our methods by extensive experiments on both public advertising dataset and online A/B testing.

Currently, we only consider online optimization of single hyperparameter. While multiple hyperparameters can be optimized separately by our methods, the correlation structure between different hyperparameters remains to be investigated. We leave the study of online and joint optimization of multiple hyperparameters as a future work.

ACKNOWLEDGMENTS

This work was partially supported by NSFC (62122037), JiangsuSF (BK20200064), and the Open Research Projects of Zhejiang Lab (NO. 2021KB0AB02). We thank the anonymous reviewers for their constructive suggestions.

REFERENCES

[1] Deepak Agarwal, Bo Long, Jonathan Traupman, Doris Xin, and Liang Zhang. 2014. Laser: A scalable response prediction platform for online advertising. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 173–182.

[2] Richard Loree Anderson. 1953. Recent advances in finding best operating conditions. *J. Amer. Statist. Assoc.* 48, 264 (1953), 789–798.

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

[4] Manpreet Singh Bajwa, Ravi Rana, and Geetanshi Bagga. 2020. Machine Learning-Based Information Retrieval System. In *The International Conference on Recent Innovations in Computing*. Springer, 13–22.

[5] Steven M Beitzel. 2006. *On understanding and classifying web queries*. Citeseer.

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, Vol. 24. Neural Information Processing Systems Foundation.

[7] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[8] Fedor Borisyuk, Krishnaram Kenthapadi, David Stein, and Bo Zhao. 2016. CaSMoS: A framework for learning candidate selection models over structured queries and documents. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 441–450.

[9] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 1–27.

[10] Louis Comtet. 2012. *Advanced Combinatorics: The art of finite and infinite expansions*. Springer Science & Business Media.

[11] Nuno Dionísio, Fernando Alves, Pedro M Ferreira, and Alysson Bessani. 2019. Cyberthreat detection from twitter using deep neural networks. In *2019 International Joint Conference on Neural Networks*. IEEE, 1–8.

[12] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, Vol. 10. 3.

[13] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

[14] Kun Gai, Xiaoqiang Zhu, Han Li, Kai Liu, and Zhe Wang. 2017. Learning piece-wise linear models from large scale data for ad click prediction. *arXiv preprint arXiv:1704.05194* (2017).

[15] Aurélien Garivier and Eric Moulines. 2011. On upper-confidence bound policies for switching bandit problems. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*. PRML, 174–188.

[16] Artem Grotov and Maarten de Rijke. 2016. Online learning to rank for information retrieval: Sigir 2016 tutorial. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1215–1218.

[17] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.

[18] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[19] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*. Springer, 507–523.

[20] Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, Arthur Gretton and Christian C. Robert (Eds.), Vol. 51. PMLR, Cadiz, Spain, 240–248.

[21] Abdul Rehman Khan, Ameer Tamoor Khan, Masood Salik, and Sunila Bakhsh. 2021. An Optimally Configured HP-GRU Model Using Hyperband for the Control of Wall Following Robot. *International Journal of Robotics and Control Systems* 1, 1 (2021), 66–74.

[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Aarti Singh and Jerry Zhu (Eds.), Vol. 54. PMLR, Fort Lauderdale, FL, USA, 528–536.

[24] Dan Li and Evangelos Kanoulas. 2018. Bayesian optimization for optimizing retrieval systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 360–368.

[25] Hang Li and Zhengdong Lu. 2016. Deep learning for information retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1203–1206.

[26] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18, 1 (2017), 6765–6816.

[27] Tie-Yan Liu. 2011. Learning to rank for information retrieval. (2011).

[28] Debanjan Mahata, Jasper Friedrichs, Rajiv Ratn Shah, and Jing Jiang. 2018. Detecting personal intake of medicine from twitter. *IEEE Intelligent Systems* 33, 4 (2018), 87–95.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32 (2019), 8026–8037.

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

[31] BH Shekar and Guesh Dagnew. 2019. Grid search-based hyperparameter tuning and classification of microarray cancer data. In *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*. IEEE, 1–8.

[32] Luo Si and Rong Jin. 2011. Machine Learning for Information Retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 1293–1294.

[33] Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems* (2012).

[34] Francisco J Solis and Roger J-B Wets. 1981. Minimization by random search techniques. *Mathematics of operations research* 6, 1 (1981), 19–30.

[35] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 847–855.

[36] Lazar Valkov, Rodolphe Jenatton, Fela Winkelmolen, and Cédric Archambeau. 2018. A simple transfer-learning extension of Hyperband. In *NIPS Workshop on Meta-Learning*.

[37] Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang, and Sridhar Mahadevan. 2015. Efficient hyper-parameter optimization for NLP applications. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2112–2117.

[38] Xiashuang Wang, Guanghong Gong, Ni Li, and Shi Qiu. 2019. Detection analysis of epileptic EEG using a novel random forest model combined with grid search optimization. *Frontiers in human neuroscience* 13 (2019), 52.

[39] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[40] Lei Yao, Zhanpeng Fang, Yanqiu Xiao, Junjian Hou, and Zhijun Fu. 2021. An intelligent fault diagnosis method for lithium battery systems based on grid search support vector machine. *Energy* 214 (2021), 118866.

[41] Hema Yoganarasimhan. 2020. Search personalization using machine learning. *Management Science* 66, 3 (2020), 1045–1070.

[42] Weinan Zhang, Xiangyu Zhao, Li Zhao, Dawei Yin, Grace Hui Yang, and Alex Beutel. 2020. Deep Reinforcement Learning for Information Retrieval: Fundamentals and Advances. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2468–2471.

[43] Qi Zhao, Yi Zhang, Daniel Friedman, and Fangfang Tan. 2015. E-commerce recommendation with personalized promotion. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 219–226.

[44] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, New York, NY, USA, 1059–1068.

[45] Marc-André Zöller and Marco F Huber. 2021. Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research* 70 (2021), 409–472.

A PROOF OF LEMMA 2

Proof. Fix an epoch i, a round $t \in [\tau_i + 2\lambda, \tau_{i+1})$, and an active arm $a \in \mathcal{A}_t$. Let t' be the last round when a is pulled, i.e., $I_{t'} = a$ and $a \notin \{I_{t'+1}, \ldots, I_{t-1}\}$. If $t' < t - \lambda$, we have $\xi_t(a) = \sqrt{\log\left(2t^{1.5}/\delta^{0.5}\right)/\sum_{s=t-\lambda}^{t-1} \mathbb{1}[I_s = a]} = +\infty$. Thus, the inequality $\pi\xi_t(a) \geq \Delta_i(a)$ trivially holds.

Below, we consider the case $t' \geq t - \lambda$. On the one hand, since (10) is guaranteed throughout the T rounds, there must exist an

active arm $a' \in \mathcal{A}_{t'}$ such that its covering interval contains the optimal arm o_i^*, i.e., $o_i^* \in [a' - \xi_{t'}(a'), a' + \xi_{t'}(a')]$. By the Lipschitz continuity of mean reward function, we have

$$\mu_{t'}(o_i^*) - \mu_{t'}(a') \leq |o_i^* - a'| \leq \xi_{t'}(a'). \tag{18}$$

On the other hand, according to the arm selection rule in (11), $I_{t'} = a$ implies $\hat{\mu}_{t'}(a) + 2\xi_{t'}(a) \geq \hat{\mu}_{t'}(a') + 2\xi_{t'}(a')$. As $t' \geq t - \lambda$, we have $t' \geq \tau_i + \lambda$ and hence $t \in \mathcal{T}$. Applying Lemma 1 gives $\hat{\mu}_{t'}(a) - \mu_{t'}(a) \leq \xi_{t'}(a)$ and $\mu_{t'}(a') - \hat{\mu}_{t'}(a') \leq \xi_{t'}(a')$. By these three inequalities, we obtain

$$\mu_{t'}(a) + 3\xi_{t'}(a) \geq \mu_{t'}(a') + \xi_{t'}(a'). \tag{19}$$

Combining (18) and (19) leads to $\Delta_i(a) = \mu_{t'}(o_i^*) - \mu_{t'}(a) \leq 3\xi_{t'}(a)$.

Finally, if a is pulled not more than ten times during $[t'-\lambda, t'-1]$, we have $\sum_{s=t-\lambda}^{t-1} \mathbb{1}[I_s = a] \leq 11$ and hence $\pi\xi_t(a) \geq 1 \geq \Delta_i(a)$. Otherwise, we have $\xi_t(a)/\xi_{t'}(a) \geq \sqrt{11/12} \geq 3/\pi$. □