ADAO2B: ADAPTIVE ONLINE TO BATCH CONVERSION FOR OUT-OF-DISTRIBUTION GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

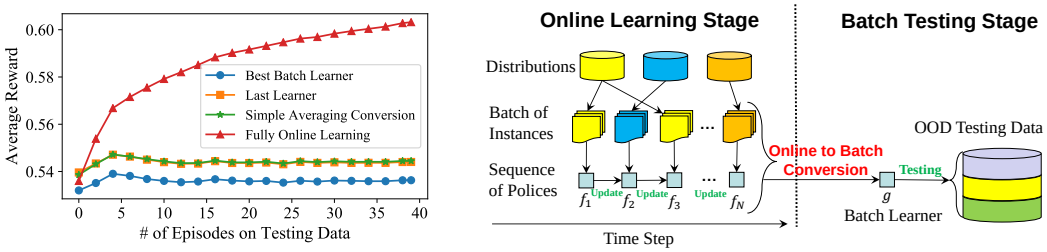
Online to batch conversion involves constructing a new batch learner by utilizing a series of models generated by an existing online learning algorithm, for achieving generalization guarantees under i.i.d assumption. However, when applied to real-world streaming applications such as streaming recommender systems, the data stream may be sampled from time-varying distributions instead of persistently being i.i.d. This poses a challenge in terms of out-of-distribution (OOD) generalization. Existing approaches employ fixed conversion mechanisms that are unable to adapt to novel testing distributions, hindering the testing accuracy of the batch learner. To address these issues, we propose AdaO2B, an adaptive online to batch conversion approach under the bandit setting. AdaO2B is designed to be aware of the distribution shifts in the testing data and achieves OOD generalization guarantees. Specifically, AdaO2B can dynamically combine the sequence of models learned by a contextual bandit algorithm and determine appropriate combination weights using a context-aware weighting function. This innovative approach allows for the conversion of a sequence of models into a batch learner that facilitates OOD generalization. Theoretical analysis provides justification for why and how the learned adaptive batch learner can achieve OOD generalization error guarantees. Experimental results have demonstrated that AdaO2B significantly outperforms state-of-the-art baselines on both synthetic data and real-world data.

1 INTRODUCTION

Online learning aims at conducting sequential decision-making by capturing the dynamic nature of data stream, which generates an updated model at each round and uses it for the next decision-making round (Cesa-Bianchi & Lugosi, 2006; Shalev-Shwartz, 2011). To achieve regret guarantees, online learning algorithms incrementally update the model upon new instances are received. However, fully updating the model in an online fashion is often computationally expensive and leads to decision instability in many real-world applications (*e.g.*, streaming recommender systems). An effective approach is to employ *online to batch conversion* (Littlestone, 1989; Cesa-Bianchi et al., 2004), where a batch learner is constructed based on the sequence of models generated by an existing online learning algorithm. During the testing process, the *batch learner* remains fixed and aims to benefit from the sequence of existing models to achieve good generalization abilities.

Classic online to batch (O2B) conversion approaches typically assume that the instances in data stream are i.i.d. according to a fixed but unknown distribution. Under this i.i.d. assumption, O2B conversion involves selecting a representative model or averaging multiple models (Dekel & Singer, 2005; Dekel, 2008; Cutkosky, 2019). However, in real-world applications, distribution shifts between the training and testing data are ubiquitous, posing new challenges of achieving out-of-distribution (OOD) generalization guarantees through O2B conversions. For instance, in streaming recommender systems, user preferences often change dynamically (Hamidzadeh & Moradi, 2021; Zhang et al., 2021a). For example, a user's preference for different categories of videos may vary due to factors like weather or mood, and the features of a video may change on different timestamp.

Existing O2B conversion technologies are not suitable for the OOD scenarios due to their fixed conversion mechanisms, which can not adjust strategies of combining or selecting models for adaptation to novel or similar testing distributions. Figure 1(a) presents an empirical study on the real-world video recommendation data, where user preferences in the testing data may differ due



(a) (b)

Figure 1: (a) Generalization performances (average rewards on testing data) of classic online to batch conversion technologies on `KuaiRec` data, where "Best Batch Learner" selects the model that achieves the highest cumulative reward among all episodes for batch testing, "Simple Averaging Conversion" uses the average over the model sequence for testing (Cesa-Bianchi et al., 2004), "Last Learner" chooses the model obtained in the last episode for testing (Shalev-Shwartz, 2007), "Fully Online Learning" keeps updating the learner on the testing data in an online manner, and the online learning backbone is sequential batch UCB (Han et al., 2020). (b) Online to batch conversion in BCB setting on out-of-distribution testing data.

to variations in data collection time, leading to OOD problems in the testing phase. The results demonstrate a significant decrease in recommendation performance for classic O2B conversions compared to the fully online learning algorithm during the testing phase. This clearly indicates that out-of-distribution data can negatively impact the testing accuracy of a batch learner. Analysis reveals that in online learning, the instances received in the data stream may be sampled from multiple distributions, violating the i.i.d. assumption. As a result, the models generated by online learning algorithms have already been trained on data from these distributions, creating an opportunity to construct a batch learner for OOD generalization.

Motivated by the potential of online learning models, this paper proposes an adaptive O2B conversion approach for OOD generalization under the bandit feedback setting of online learning. To provide a deeper understanding, we first establish the OOD generalization error bounds of O2B conversion theoretically, establishing the relationship between weighted regret and the OOD generalization error of a batch learner. Building on these theoretical results, we introduce AdaO2B, which leverages an adaptive weighting network to be aware of distribution shifts and adaptively combines the model sequence for decision-making. AdaO2B utilizes observed rewards as supervised signals and can be efficiently trained using different data selection approaches based on bandit feedback. Furthermore, AdaO2B is model-agnostic and can be applied to any contextual bandit algorithm. We summarize the major contributions as follows: (1) Rigorous theoretical analysis that establishes the relationship between the weighted regret in online learning and the OOD generalization error of a batch learner in O2B conversion; (2) A model-agnostic O2B conversion framework called AdaO2B for achieving OOD generalization guarantees under bandit feedback setting; (3) Comprehensive empirical studies showed the effectiveness of AdaO2B in terms of improving the OOD generalization ability of different bandit algorithms and its superiority over state-of-the-art O2B conversion baselines.

1.1 Related Work

Online model averaging/selection is an important topic in online learning, which aims at adjusting the model class for prediction in an online manner. Online model averaging approach has been extensively studied in online kernel learning, which typically reduces the model averaging problem to a prediction problem with expert advice (Jin et al., 2010; Orabona et al., 2010). Orabona et al. (2010) designed a variant of follow-the-regularized-leader for online model averaging in multiclass problems. Zhang & Liao (2018) presented an efficient online model selection approach using incremental sketched kernel alignment, which formulates an unbiased selection criterion of kernel models. Model selection under bandit settings has received increased attention over the past several years, which can adapt the reward model class of optimal policy (Foster et al., 2019; Ghosh et al., 2021). Since existing online model averaging/selection approaches finally obtain fixed strategies for weighting or selecting the learners, they are not suitable for adapting the OOD testing data. Besides, how to perform O2B conversion for OOD generalization under bandit settings is still an unsolved problem.

Transfer learning has received considerable attention in machine learning literature, which aims to improve the generalization performance on target domain by transferring the knowledge contained in previous related source domains (Pan & Yang, 2009; Weiss et al., 2016; Zhuang et al., 2020). There are several types of technologies that are commonly used in transfer learning, including weighted ERM (Huang et al., 2006; Reddi et al., 2015), feature mapping (Ando et al., 2005), regularization (Duan et al., 2009; Kuzborskij & Orabona, 2017). More recently, a series of works lies in the idea of transferring the model trained on streaming data for adapting the target domain (Zhao et al., 2020; Tao & Lu, 2020). Zhao et al. (2020) proposed a regularized empirical risk minimization approach on target sources, which could online update the weight of sub-models with theoretical guarantees. Tao & Lu (2020) focused on the assumption that instances in the data stream are sampled from the same distribution but not guaranteed independent, and proposed a training process for fine-tuning the final model outputted by any online learning algorithm. Unlike the O2B conversion we focused on, these works need to specify the data source or the distribution assumption and re-train the model on data from the target source. Besides, these works are concerned with the full-information feedback settings rather than bandit settings in this paper.

2 PROBLEM FORMULATION

Let $[m] = \{1, 2, \ldots, m\}$, $\mathcal{S} \subseteq \mathbb{R}^d$ be the context space whose dimension is d. Since bandit feedback is ubiquitous in real-world applications, in this paper we focus on the batched version of contextual bandit (BCB) setting, which is an extension of the classic contextual bandit setting (Perchet et al., 2015; Han et al., 2020; Esfandiari et al., 2021). In BCB setting, the sequential decision-making process is partitioned into N *episodes*, where each episode includes B decision-making steps (B is also called the *batch size*). Specifically, at step b in the n-th episode, a bandit algorithm receives a *candidate context set* $\mathcal{S}_{n,b} \subseteq \mathcal{S}$, where $\mathcal{S}_{n,b}$ contains M contexts $\mathcal{S}_{n,b} = \{\boldsymbol{s}_I\}_{I\in[M]}$ and each context corresponds to a candidate action. Then, the bandit algorithm chooses a context $\boldsymbol{s}_{I_{n,b}} \in \mathcal{S}_{n,b}$ following the *policy* (*i.e.,*decision model) $f_n : \mathcal{S}_{n,b} \to [M]$ parameterized by $\boldsymbol{\theta}_n$, where $I_{n,b} \in [M]$ can be seen as the index of the *executed action* at step b in the n-th episode. After choosing the context, algorithm will observe a *reward* $R_{n,b}$. Note that, during the decision process in each episode, the bandit policy is fixed. That is, the policy f_n is only updated at the end of the n-th episode based on a *data buffer* $\mathcal{D}_n = \{(\mathcal{S}_{n,b}, I_{n,b}, R_{n,b})\}_{b\in[B]}$.

Figure 1(b) illustrates the online to batch (O2B) conversion problem for out-of-distribution (OOD) generalization in the above BCB setting. In the *online learning phase*, bandit policy is incrementally updated on data steam generated from multiple distributions. O2B conversion aims to formulate a batch learner based on the collected data for OOD testing in the *batch testing phase*, where the testing distributions may be similar but different. Next, we introduce the formal definition of O2B conversion for OOD generalization in BCB setting.

Definition 1 (O2B Conversion for OOD Generalization)**.** *Consider the BCB setting that includes N episodes. Let $\mathcal{F}_N = \{f_1, f_2, \ldots, f_N\}$ be a sequence of policies (i.e, decision models) generated by performing a bandit algorithm $\mathcal{A}$, and $\mathcal{D} = \{\mathcal{D}_n\}_{n\in[N]}$ be the sequence of data buffers that store the interaction history. Assume that the context-reward pairs are generated according to a distribution $\mathbb{P}$ (may be a mixture distributions[1]). O2B conversion aims to find a O2B function f_{o2b}: $f_{\mathrm{o2b}} : \mathcal{F}_N \times \mathcal{D} \to g \in \mathcal{G}$, where g is a **batch learner** in a policy space $\mathcal{G}$ which has generalization guarantees on a testing distribution $\mathbb{Q}$ that may be different from the distribution $\mathbb{P}$.*

3 OOD GENERALIZATION ANALYSIS

In this section, we first specify the key ingredients in Definition 1 and then carry out an OOD generalization justification on why and how to conduct O2B conversion in environments with distribution shifts. The detailed proofs of the theoretical results can be found in the Appendix A.

We specify the key ingredients in Definition 1 as follows.

Reward r. Following the setups in linear contextual bandit literature (Dimakopoulou et al., 2019; Yang et al., 2021; Li et al., 2010), for any context $\boldsymbol{s}_i \in \mathcal{S} \subseteq \mathbb{R}^d$, we assume that the expectation of

[1]In Corollary 1, we will give the formulation of the mixture distribution representing the sampling process according to multiple distributions.

the observed reward R_i is determined by unknown *true reward parameters* $\boldsymbol{\theta}^* \in \mathbb{R}^d$: $\mathbb{E}[R_i \mid \boldsymbol{s}_i] = \langle \boldsymbol{\theta}^*, \boldsymbol{s}_i \rangle$. The linear reward will be extended to the convex function case in Corollary 1.

Distributions $\mathbb{P}$ and $\mathbb{Q}$. Given the linear reward assumption, the distribution shift of the context-reward pairs can be described as the shift of context distribution. Thus, we define that $\mathbb{P}$ and $\mathbb{Q}$ are the distributions on the context space $\mathcal{S}$. Besides, $\mathbb{P}$ and $\mathbb{Q}$ are unknown to the algorithm and may be both are mixtures of multiple distributions defined in Corollary 1.

Policy space $\mathcal{G}$. We assume that $\mathcal{G}$ is the set containing all possible linear combinations of policies in the sequence of policies $\mathcal{F}_N = \{f_1, f_2, \ldots, f_N\}$ generated by a bandit algorithm. Formally, the *adaptive batch learner* $g \in \mathcal{G}$ can be formulated as follows:

$$g(\boldsymbol{s}) = \sum_{n \in [N]} \beta_n \, f_n(\boldsymbol{s}) / \beta_{1:N}, \quad \forall \boldsymbol{s} \in \mathcal{S}, \tag{1}$$

where $\{\beta_n\}_{n \in [N]}$ denotes the *combination weights* that are outputs of a context-aware *weighting function* $h : \mathcal{S} \to \mathbb{R}^N$ for adapting the distribution shifts between $\mathbb{P}$ and $\mathbb{Q}$, and $\beta_{1:N}$ denotes the sum of weights over the number of episodes N, *i.e.,*$\beta_{1:N} := \sum_{n \in [N]} \beta_n$. More specifically, for the linear true reward, bandit policy f_n typically chooses action (*i.e.,*the context) according to the *estimated reward* $r_n(\boldsymbol{s}) = \langle \boldsymbol{\theta}_n, \boldsymbol{s} \rangle$, where $\boldsymbol{\theta}_n$ denotes the *estimated reward parameters* in f_n from the n-the episode. Then, the adaptive batch learner g in equation 1 can be represented by the combination of the estimated reward parameters[2]

$$\boldsymbol{\theta}_{\mathrm{ada}} = \sum_{n \in [N]} \beta_n \boldsymbol{\theta}_n / \beta_{1:N}. \tag{2}$$

For the sake of simplicity, we denote the adaptive batch learner g by $\boldsymbol{\theta}_{\mathrm{ada}}$.

Evaluation metric of OOD generalization. Given an unknown distribution (may be a mixture distribution) of the contexts $\boldsymbol{s} \in \mathcal{S}$, define the *expected reward* of a policy with estimated reward parameters $\boldsymbol{\theta}$ w.r.t. $\mathbb{Q}$ to be $\mathrm{ER}_{\mathbb{Q}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s} \sim \mathbb{Q}}[\langle \boldsymbol{\theta}, \boldsymbol{s} \rangle]$. Then, we define the *generalization error* (also called *expected risk*) of $\boldsymbol{\theta}$ w.r.t. $\mathbb{Q}$ as follows:

$$\mathrm{GE}_{\mathbb{Q}}(\boldsymbol{\theta}) = C_{\mathrm{ER}} - \mathrm{ER}_{\mathbb{Q}}(\boldsymbol{\theta}), \tag{3}$$

where C_{ER} denotes the upper bound of the absolute values of expected reward w.r.t. any distribution. Then, $\mathrm{GE}_{\mathbb{Q}}(\boldsymbol{\theta}_{\mathrm{ada}})$ can be used for evaluating the OOD generalization performance of the adaptive batch learner $\boldsymbol{\theta}_{\mathrm{ada}}$ on testing distribution $\mathbb{Q}$.

Following the above setup of O2B conversion problem, we first demonstrate the OOD generalization error bound of the adaptive batch learner $\boldsymbol{\theta}_{\mathrm{ada}}$ in equation 2.

Theorem 1 (OOD Generalization Error Bound of Adaptive Batch Learner)**.** *Consider the BCB setting with N episodes and the batch size B. Let $\boldsymbol{\theta}_n, n \in [N]$ be the reward parameters estimated by policy f_n, $\boldsymbol{\theta}^*$ be the true reward parameters, $\boldsymbol{s}_{I_{n,b}} \in \mathcal{S}_{n,b}$ be the context chosen by f_n at step b, and C_{ER} be the upper bound defined in equation 3. Define the weighted regret as*

$$\mathrm{WReg}(N, B) = \sum_{n \in [N], b \in [B]} \beta_n \left\langle \boldsymbol{\theta}^* - \boldsymbol{\theta}_n, \boldsymbol{s}_{I_{n,b}} \right\rangle, \tag{4}$$

and assume that the weighted regret is bounded by C_{WReg}, i.e.,$\mathrm{WReg}(N, B) \leq C_{\mathrm{WReg}}$*. Then,*

$$\mathrm{GE}_{\mathbb{Q}}(\boldsymbol{\theta}_{\mathrm{ada}}) - \mathrm{GE}_{\mathbb{P}}(\boldsymbol{\theta}^*) \leq \mathbb{E}_{\mathbb{P}} \left[\frac{C_{\mathrm{WReg}}}{B \beta_{1:N}} \right] + C_{\mathrm{ER}} \sqrt{2 \mathrm{D}_{\mathrm{JS}}(\mathbb{Q} \| \mathbb{P})}, \tag{5}$$

where $\mathrm{D}_{\mathrm{JS}}(\mathbb{Q} \| \mathbb{P})$ denotes the Jensen-Shannon (JS) divergence between the distributions $\mathbb{Q}$ and $\mathbb{P}$.

Remark 1 (Tighten the Bound)**.** *Theorem 1 tells us that the weighted regret upper bounds the OOD generalization error of the adaptive bath learner $\boldsymbol{\theta}_{\mathrm{ada}}$. That is, to achieve good OOD generalization performances, the batch learner's objective should be to minimize the weighted regret in equation 4 by adjusting the combination weights $\{\beta_n\}_{n \in [N]}$. Besides, a smaller JS divergence between the training distribution $\mathbb{P}$ and the testing distribution $\mathbb{Q}$ leads to a tighter bound in equation 5.*

[2]In the batch testing phase, since the batch learner is fixed, we omit the exploration term in the original policy (*e.g.*, the uniform distribution term in EXP3 policy).

Remark 2 (Simply to i.i.d.). *In equation 5, setting $\beta_n = 1$ for all $n \in [N]$ and $\mathbb{Q} = \mathbb{P}$ yields the i.i.d. risk bound of a simple averaging learner $\boldsymbol{\theta}_{\text{avg}} = \sum_{n\in[N]} \boldsymbol{\theta}_n/N$ in bandit setting. Analogous to the case of full-information O2B conversion (Shalev-Shwartz, 2007), the result in equation 5 becomes $\text{GE}_{\mathbb{P}}(\boldsymbol{\theta}_{\text{avg}}) \leq \text{GE}_{\mathbb{P}}(\boldsymbol{\theta}^*) + \mathbb{E}_{\mathbb{P}}\left[C_{\text{Reg}}/T\right]$, where C_{Reg} is the upper bound of worse-case regret and $T = N \times B$.*

As illustrated in Figure 1(b), the distributions of $\mathbb{P}$ and $\mathbb{Q}$ may be mixture distributions, yielding the following result.

Corollary 1 (OOD Generalization Error for Mixture Distributions). *Assume that the conditions in Theorem 1 hold, and $\mathbb{P}, \mathbb{Q}$ are two mixture distributions with distribution densities $p(\boldsymbol{s}) = \sum_{i\in[W]} \pi_i \cdot p_i(\boldsymbol{s})$ and $q(\boldsymbol{s}) = \sum_{i\in[W]} \tau_i \cdot q_i(\boldsymbol{s})$, respectively, where $\{p_i\}_{i\in[W]}$ and $\{q_i\}_{i\in[W]}$ are densities of Gaussian distributions in $\mathbb{P}$ and $\mathbb{Q}$, $\boldsymbol{\pi} = \{\pi_i\}_{i\in[W]}$ and $\boldsymbol{\tau} = \{\tau_i\}_{i\in[W]}$ denotes multi-dimensional Bernoulli distributions. Then,*

$$\text{GE}_{\mathbb{Q}}(\boldsymbol{\theta}_{\text{ada}}) - \text{GE}_{\mathbb{P}}(\boldsymbol{\theta}^*) \leq \mathbb{E}_{\mathbb{P}}\left[\frac{C_{\text{WReg}}}{B\beta_{1:N}}\right] + C_{\text{ER}}\sqrt{2\log K + K\sum_{i\in[W]} \pi_i\tau_i \cdot \text{D}_{\text{BKL}}(p_i\|q_i)}, \quad (6)$$

where $\text{D}_{\text{BKL}}(p_i\|q_i) := \frac{1}{2}\left[\text{D}_{\text{KL}}(p_i\|q_i) + \text{D}_{\text{KL}}(q_i\|p_i)\right]$ denotes the bidirectional Kullback-Leibler (KL) divergence between p_i and q_i (Liang et al., 2021), and $\text{D}_{\text{KL}}(p\|q)$ denotes the KL divergence between the distributions p and q.

Remark 3 (Weighted Bidirectional KL Divergence). *The weight term $\pi_i\tau_i$ can be seen as the probability of drawing distributions p_i and q_i for online learning and OOD testing, respectively. Then, the weighted bidirectional KL divergence in equation 6 indicates that, if distributions in $\mathbb{P}$ and $\mathbb{Q}$ with similar probability of occurrence have similar distribution densities, tighter OOD generalization error bound can be achieved.*

We further give a more general version of the OOD generalization error bound in Theorem 1, where the expectation of true rewards is convex w.r.t. the reward parameters.

Corollary 2 (OOD Generalization Error for Convex Rewards). *Consider the convex reward $r(\boldsymbol{\theta}, \boldsymbol{s})$ parameterized by $\boldsymbol{\theta}$ as well as its expected reward $\text{ER}_{\mathbb{Q}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s}\sim\mathbb{Q}}\left[r(\boldsymbol{\theta}, \boldsymbol{s})\right]$ (bounded by C_{ER}), and the generalization error $\text{GE}_{\mathbb{Q}}(\boldsymbol{\theta})$. Assume that $\mathbb{E}[\nabla_{\boldsymbol{\theta}_n} r(\boldsymbol{\theta}_n, \boldsymbol{s})] = \nabla_{\boldsymbol{\theta}^*} r(\boldsymbol{\theta}^*, \boldsymbol{s})$ holds for $\forall n \in [N]$. Define the weighted regret as $\text{WReg}(N, B) = \sum_{n\in[N], b\in[B]} \beta_n \left\langle \boldsymbol{\theta}^* - \boldsymbol{\theta}_n, \nabla_{\boldsymbol{\theta}^*} r(\boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}})\right\rangle$, and assume that the weighted regret is bounded by C_{WReg}, i.e., $\text{WReg}(N, B) \leq C_{\text{WReg}}$. Then, given the new definitions of C_{WReg} and C_{ER} for convex rewards, the adaptive batch learner $\boldsymbol{\theta}_{\text{ada}}$ enjoys the same upper bound of OOD generalization error as in equation 5.*

The above results provide theoretical guidance to implement the adaptive O2B learner, and we will derive guiding principles as well as the algorithm implementation in the next section.

4 AdaO2B: The Proposed Algorithm

From Remark 1&3, we can derive the following guiding principles for designing the adaptive O2B algorithm: (a) The data for training the adaptive batch learner should be as close as possible to the testing distribution; (b) The combination weights $\beta_n, n \in [N]$ should be computed based on the received candidate contexts at each step, which can be aware of the changes in test distribution; (c) The objective of training the weighting function should contain the component of minimizing the weighted regret in equation 4 or its surrogate. Next, we provide a practical implementation of the adaptive batch learner $\boldsymbol{\theta}_{\text{ada}}$ in equation 2 named AdaO2B.

4.1 Data Selection

We use subsets of the whole sequences of policies and data buffers in Definition 1 for O2B conversion. Specifically, to reduce the difference between the training distribution of O2B conversion and the testing distribution, we propose the following three approaches for maintaining the data buffers (denoted by $\overline{\mathcal{D}}_{\mathcal{K}} := \{\mathcal{D}_n\}_{n\in\mathcal{K}} \subseteq \mathcal{D}$) and the candidate policy set (denoted by $\mathcal{F}_{\mathcal{K}} := \{f_n\}_{n\in\mathcal{K}} \subseteq \mathcal{F}_N$), where $\mathcal{K} \subseteq [N]$ denotes the index set of the selected data buffers which is also the index set of the policies trained on these selected buffers, and $K := |\mathcal{K}|$ denotes the cardinality of $\mathcal{K}$ (*i.e.,*the

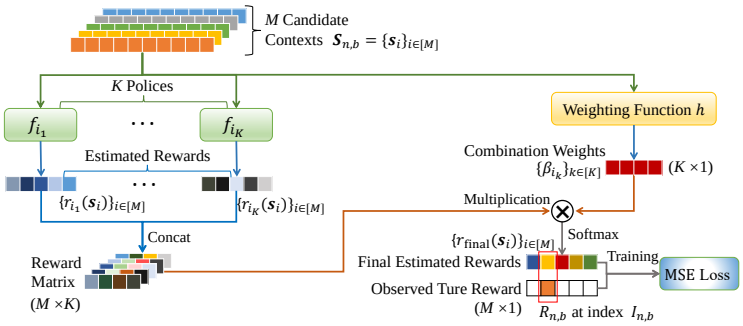


Figure 2: The training process of the proposed AdaO2B based on the data collected at step b in the n-th episodes, where $\{i_k\}_{k\in[K]}$ denotes the index set $\mathcal{K}$ specified in Section 4.1.

cardinality of $\overline{\mathcal{D}}_{\mathcal{K}}$ and $\mathcal{F}_{\mathcal{K}}$).

1) Sliding window approach. Motivated by the fact that streaming data from neighboring periods typically has similar distributions in the streaming applications (Bendada et al., 2020; Zhang et al., 2021b), we use the data buffers and policies from the most recent K episodes for training the adaptive O2B learner, *i.e.,*$\mathcal{K} = \{N-K+1, N-K+2, \ldots, N\}$. More specifically, the training process is based on $\overline{\mathcal{D}}_{\mathcal{K}} = \{\mathcal{D}_n\}_{n=N-K+1}^{N}$ that are the data buffers in the recent K episodes, and the candidate policy set $\mathcal{F}_{\mathcal{K}} = \{f_n\}_{n=N-K+1}^{N}$.

2) Reservoir sampling approach. Without knowing the number of episodes N, we can use reservoir sampling approach (Knuth, 1998) to determining which data buffers and the corresponding candidate policies are stored for O2B conversion, such that every data buffer in $\overline{\mathcal{D}}_{\mathcal{K}}$ (as well as every policy in $\mathcal{F}_{\mathcal{K}}$) has the same probability of being selected in an online fashion.

3) Data-dependent approach. Since the decrease or low growth of the rewards received in two episodes indicates severe distribution shifts, we can maintain the candidate policies with average rewards of low growth rates and the corresponding data buffers. Specifically, we present a data-dependent approach for selecting the elements in $\overline{\mathcal{D}}_{\mathcal{K}}$ and $\mathcal{F}_{\mathcal{K}}$: the index set $\mathcal{K}$ is constructed by selecting the indices with the bottom-K values of $(\overline{R}_{n+1} - \overline{R}_n)/\overline{R}_n, \forall n \in [N]$, where $\overline{R}_n$ denotes the average reward over all steps in the n-th episode.

4.2 Model Formulation and Training

Formally, by only retaining the exploitation term in the policy for batch testing, the k-th policy in $\mathcal{F}_{\mathcal{K}} = \{f_n\}_{n\in\mathcal{K}}$ can be written as $f_{i_k} = \arg\max_{j\in[M]} \langle \boldsymbol{\theta}_{i_k}, \boldsymbol{s}_j \rangle, \quad \boldsymbol{s}_j \in \mathcal{S}_{i_k,b}$, which makes decision according to the estimated reward using the estimated reward parameters $\boldsymbol{\theta}_{i_k}$ given the contexts. Then, given the index set $\mathcal{K} = \{i_k\}_{k\in[K]}$ defined in Section 4.1, the adaptive batch learner in equation 2 can be expressed as: $\boldsymbol{\theta}_{\text{ada}} = \sum_{k\in[K]} \beta_{i_k}\boldsymbol{\theta}_{i_k}/\beta_{\mathcal{K}}, \quad i_k \in \mathcal{K}$, where $\beta_{\mathcal{K}} = \sum_{k\in[K]} \beta_{i_k}$. The combination weights $\{\beta_{i_k}\}_{k\in[K]}$ are obtained using the context-aware weighting function $h : \mathcal{S} \to \mathbb{R}^K$. We implement the weighting function h using one MLP (Multi-Layer Perceptron), denoted by MLP_h. As shown in Figure 2, at each step, the weighting function h (*i.e.,*MLP_h) takes the candidate contexts as input, and obtains a K-dimensional vector $\{\beta_{i_k}\}_{k\in[K]}$.

To incorporate the weighted regret in equation 4 into the training objective, we transform the weighted regret into a loss function. Given the index set $\mathcal{K}$, for the candidate context set $\mathcal{S}_{i_k,b}$ collected at step b in the i_k-th episodes, the weighted regret truncated with K becomes

$$\sum_{k\in[K]} \beta_{i_k} \left\langle \boldsymbol{\theta}^* - \boldsymbol{\theta}_{i_k}, \boldsymbol{s}_j \right\rangle, \quad \boldsymbol{s}_j \in \mathcal{S}_{i_k,b}. \tag{7}$$

Multiplying equation 7 by $1/\beta_{\mathcal{K}}$, we get

$$\left\langle \sum_{k\in[K]} \beta_{i_k}/\beta_{\mathcal{K}}\boldsymbol{\theta}^*, \boldsymbol{s}_j \right\rangle - \left\langle \boldsymbol{\theta}_{\text{ada}}, \boldsymbol{s}_j \right\rangle, \quad \boldsymbol{s}_j \in \mathcal{S}_{i_k,b}, \tag{8}$$

where the first term can be seen as an estimate of the true reward, and the second term is the *final estimated reward* estimated by $\boldsymbol{\theta}_{\text{ada}}$, denoted by $r_{\text{final}}(\boldsymbol{s}) = \langle \boldsymbol{\theta}_{\text{ada}}, \boldsymbol{s} \rangle$. Then, as a surrogate objective of minimizing equation 8, we perform the training process of $\boldsymbol{\theta}_{\text{ada}}$ by minimizing the difference between the observed true reward $R_{n,b}$ in data buffers and the final estimated reward. More specifically, in the adaptive batch learner $\boldsymbol{\theta}_{\text{ada}}$, model parameters that need to be trained include the

Algorithm 1 AdaO2B

Require: Batch size B, number of episodes N, bandit algorithm $\mathcal{A}$, number of candidate policies K
Ensure: Adaptive batch learner $\boldsymbol{\theta}_{\text{ada}}$

Initialize the policy $\boldsymbol{\theta}_1$ using uniform distribution, and the candidate policies set $\mathcal{F}_{\boldsymbol{\theta}} \leftarrow \emptyset$
// Online Learning Phase
for $n = 1$ **to** N **do**
  for $b = 1$ **to** B **do**
    Receive candidate context set $\mathcal{S}_{n,b}$ and choose context $s_{I_{n,b}} \in \mathcal{S}_{n,b}$ following the policy $\boldsymbol{\theta}_n$
    Observe the reward $R_{n,b}$
  end for
  Store the interactions into a data buffer $\mathcal{D}_n \leftarrow \{(\mathcal{S}_{n,b}, I_{n,b}, R_{n,b})\}_{b\in[B]}$
  Store the policy $\mathcal{F}_{\boldsymbol{\theta}} \leftarrow \mathcal{F}_{\boldsymbol{\theta}} \cup \{\boldsymbol{\theta}_n\}$ if $n \in \mathcal{K}$
  Update the policy $\boldsymbol{\theta}_n$ as $\boldsymbol{\theta}_{n+1} \leftarrow \Delta(\boldsymbol{\theta}_n)$ on $\mathcal{D}_n$ using bandit algorithm $\mathcal{A}$
end for
// Online to Batch Conversion Phase
Collect K data buffers $\overline{\mathcal{D}}_{\mathcal{K}} \leftarrow \{\mathcal{D}_n\}_{n\in\mathcal{K}}$
Compute the estimated rewards $r_n(\boldsymbol{s})$ using policies in $\mathcal{F}_{\boldsymbol{\theta}}$ for all $n \in \mathcal{K}$ and $\boldsymbol{s} \in \cup_{n\in\mathcal{K},b\in[B]}\{\mathcal{S}_{n,b}\}$
Compute the combination weights $\{\beta_n\}_{n\in\mathcal{K}}$ using MLP_h for each candidate context set in $\overline{\mathcal{D}}_{\mathcal{K}}$
Compute the final estimated rewards $\{r_{\text{final}}(\boldsymbol{s}_i)\}_{i\in[M]}$ for each candidate context set in $\overline{\mathcal{D}}_{\mathcal{K}}$
Optimizing the loss in equation 9 on $\overline{\mathcal{D}}_{\mathcal{K}}$ using Adam and output the model parameters $\boldsymbol{\Theta}_h$ of MLP_h
return $\mathcal{F}_{\boldsymbol{\theta}} = \{\boldsymbol{\theta}_n\}_{n\in\mathcal{K}}, \boldsymbol{\Theta}_h$

parameters in the weighting function MLP_h. All these trainable parameters in MLP_h are denoted as $\boldsymbol{\Theta}_h$, and trained based on $\overline{\mathcal{D}}_{\mathcal{K}}$ specified in Section 4.1. Finally, the task of training the weighting function h amounts to minimize the following mean squared error (MSE) loss:

$$\mathcal{L}_{\boldsymbol{\Theta}_h} = \frac{1}{\left|\overline{\mathcal{D}}_{\mathcal{K}}\right|} \sum_{(I_{n,b},R_{n,b})\in\overline{\mathcal{D}}_{\mathcal{K}}} \left[R_{n,b} - r_{\text{final}}(\boldsymbol{s}_{I_{n,b}})\right]^2 + \lambda\|\boldsymbol{\Theta}_h\|_2^2, \tag{9}$$

where $\|\boldsymbol{\Theta}_h\|_2^2$ is a regularizer for avoiding over-fitting, and $\lambda \geq 0$ is the regularization parameter. Besides, as shown in Figure 2, each final estimated reward can be efficiently computed through the multiplication of the reward matrix that concatenates the estimated rewards, and the vector of combination weights. Then, the obtained final estimated rewards are normalized through a Sofmax function. Finally, Adam (Kingma & Ba, 2014) is used to conduct the optimization. We summarize the above steps in Algorithm 1, called AdaO2B. To facilitate the understanding of the whole process, we involve the online learning phase in AdaO2B.

5 Experiments

We conducted experiments to test the performance of AdaO2B on synthetic data and real-world data. The implementation details are provided in the Appendix C.

5.1 Experimental Settings

Baselines. AdaO2B was compared with several classic online to batch conversion algorithms as well as their variants, including: **Best Batch Learner (BBL)** selects the model that achieves the highest cumulative reward among all episodes for batch testing, which can be seen as a data-driven version of the random sampling conversion (Dekel & Singer, 2005). **Last Learner (LL)** directly chooses the model obtained in the last episode for testing (Shalev-Shwartz, 2007). **Simple Averaging Conversion (SAC)** simply uses the average over the model sequence for batch testing (Cesa-Bianchi et al., 2004; Shalev-Shwartz, 2011). **Voting Conversion (VC)** decides the executed action according to the majority of the candidate policies (Freund & Schapire, 1999; Dekel & Singer, 2005). **Constant Weight (CW)** outputs a weighted average batch learner with constant weights. CW records the average reward over all steps in each episode for every generated model, normalizes these average rewards using $\mathrm{Softmax}$ and uses the normalized average reward as the combination weights for averaging the model sequence.

Table 1: Comparisons of average reward (w.r.t. all episodes) for AdaO2B and baselines on synthetic and real-world KuaiRec dataset. Bold values are the best of the proposed algorithm, while the best values in baselines are underlined. '$*$': improvements over baselines are statistical significant (t-test, p-value< 0.05) compared to the best baseline.

Model Type	Dataset	Synthetic			KuaiRec		
	Algorithm	SBUCB	EXP3-B	BLTS-B	SBUCB	EXP3-B	BLTS-B
Oracle	FOL	0.7644	0.6665	0.7651	0.6032	0.5819	0.6063
Baselines	BBL	0.7354	0.6388	0.7453	0.5363	0.5691	0.5633
	LL	<u>0.7388</u>	<u>0.6441</u>	<u>0.7494</u>	0.5440	0.5583	0.5642
	SAC-S	0.7356	0.6404	0.7397	0.5445	0.5604	0.5661
	SAC-R	0.6929	0.6216	0.7019	0.5417	0.5691	<u>0.5699</u>
	SAC-D	0.7070	0.6230	0.7130	0.5426	0.5701	0.5647
	VC-S	0.7365	0.6396	0.7402	0.5441	0.5595	0.5659
	VC-R	0.6969	0.6086	0.7040	<u>0.5457</u>	<u>0.5723</u>	0.5667
	VC-D	0.7091	0.6194	0.7165	0.5421	0.5712	0.5655
	CW-S	0.7332	0.6418	0.7404	0.5445	0.5603	0.5658
	CW-R	0.7004	0.6228	0.7054	0.5424	0.5688	0.5675
	CW-D	0.7098	0.6252	0.7166	0.5427	0.5701	0.5647
Ours	AdaO2B-S	**0.7848*** (+6.23%)	0.6971* (+8.23%)	**0.7811*** (+4.23%)	0.5556* (+1.81%)	**0.5861*** (+2.41%)	0.5806* (+1.88%)
	AdaO2B-R	0.7692* (+4.11%)	0.7154* (+11.07%)	0.7672* (+2.38%)	**0.5563*** (+1.94%)	**0.5861*** (+ 2.41%)	**0.5832*** (+2.33%)
	AdaO2B-D	0.7777* (+5.27%)	**0.7380*** (+14.58%)	0.7783* (+3.86%)	0.5532* (+1.37%)	0.5833* (+ 1.92%)	0.5796* (+1.7%)

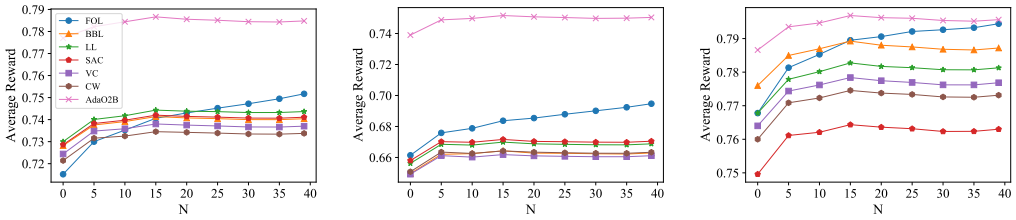
Similarly to the proposed AdaO2B, SAC, VC and CW all need to maintain the candidate policies and retained data buffers for conversion. We denote these algorithms based on different data selection approaches (defined in Section 4.1) as "algorithm name-S/R/D", where "S/R/D" denote the Sliding window approach, Reservoir sampling approach, and Data-dependent approach, respectively. For example, AdaO2B based on sliding window approach is denoted by "AdaO2B-S". AdaO2B as well as the above O2B baselines is model-agnostic, which can be applied to the following bandit backbones: **Sequential Batch UCB (SBUCB)** is a batched extension of the classic LinUCB (Li et al., 2010) where it is continuously fed with data batches (Han et al., 2020). **Batched EXP3 (EXP3-B)** is a batched version of the adversarial bandit EXP3 (Bistritz et al., 2019). **BLTS-B** uses the Thompson sampling for selecting parameters of estimated rewards (Dimakopoulou et al., 2019). To compare the performance between O2B conversion and fully online learning, we introduce the following version as oracle for each bandit backbone: **Fully Online Learning (FOL)** keeps updating the policy on the testing data in an online manner.

Data. We conducted experiments on a synthetic dataset which simulates the OOD scenario and a real-world short video recommendation dataset Kuairec[3]. Detailed descriptions about these two datasets are provided in Appendix C.2 due to the space limitation.

5.2 Experimental Results and Analyses

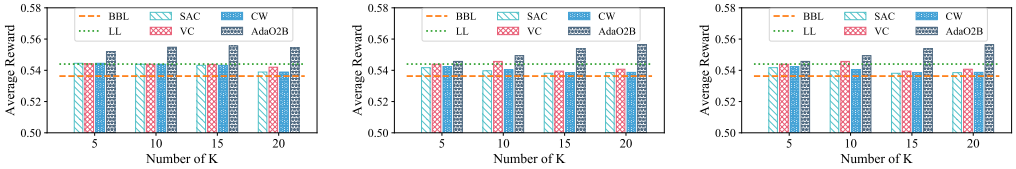
Results & discussions on synthetic data. Table 1 reports the average reward w.r.t. all episodes for the proposed AdaO2B and the baselines on the synthetic dataset. From the result, we can observe that AdaO2B significantly outperformed all the baselines with all three different bandit backbones in terms of the average rewards. Specifically, AdaO2B outperformed the best baseline (LL) by 6.23% with SBUCB, 14.58% with EXP3-B, and 4.23% with BLTS-B. Figure 3 (a)–(c) illustrate the curves of average rewards on testing data, where AdaO2B achieved the highest rewards over all episodes of testing. These results verified the effectiveness and model agnosticism of AdaO2B for capturing the distribution shifts and improving the performance of online to conversion in the out-of-distribution scenario. Furthermore, the proposed AdaO2B even outperformed FOL on synthetic data. The reason is that FOL used the exploration and exploitation trade-off strategy on testing data, where the exploration may hurt the accuracy of the bandit policy in a synthetic testing environment. For the three data selection method used in AdaO2B, we can conclude that: (1) the data-dependent approach

[3]https://kuairec.com



(a) SBUCB on Synthetic Dataset (b) EXP3-B on Synthetic Dataset (c) BLTS-B on Synthetic Dataset

Figure 3: Average rewards of baselines and the proposed AdaO2B equipped with the best data selection approach on testing data of synthetic dataset, where N denotes the number of episodes in the batch testing phase.



(a) KuaiRec; Data selection: S (b) KuaiRec; Data selection: R (c) KuaiRec; Data selection: D

Figure 4: Performance comparison of AdaO2B and all baselines with different number of candidate policies K and different data selection process on KuaiRec dataset. The bandit backbone is SBUCB. More results with other backbones (i.e., EXP3-B and BLTS-B) can be found in Appendix C.6

achieved higher average rewards than the reservoir sampling approach, indicating the advantage of the data-dependent approach for slight distribution shifts; (2) Sliding window approach typically outperformed other data selection approaches since the distribution of the recent data buffers was more similar to that of the testing data in this synthetic environment.

Results & discussions on real-world Kuairec dataset. The results of average rewards of the baselines and the proposed AdaO2B on the KuaiRec dataset are reported in Table 1. We can observe that, AdaO2B outperformed the best baseline (VC-R and SAC-R) by 1.94% with SBUCB, 2.41% with EXP3-B, and 2.33% with BLTS-B. Figure 6 in Appendix C.5 shows the curves of average rewards on KuaiRec dataset, where AdaO2B achieved the highest average reward for all bandit backbones approximating the performances of the oracle FOL. The results verified the effectiveness of the model-agnostic AdaO2B framework in improving existing bandit models for real-world online to batch conversion problem. Note that FOL with EXP3-B has lower average rewards than FOL with other bandit backbones. This phenomenon is due to the randomized exploration term used in the EXP3-B policy that is more suitable for online adversarial environments. Besides, AdaO2B equipped with the reservoir sampling approach had the best performance than other data selection approach, indicating that a simple reservoir sampling approach could capture the severe distribution drift in real-world applications.

Parameter analysis. This section empirically studied the impact of the number of candidate policies (*i.e.,*K). A larger K means the, more candidate policies and data buffers are stored for conducting the O2B conversion. We conducted experiments to test the performance of AdaO2B with different K values. The results illustrated in Figure 4 indicate that AdaO2B equipped with data selection approaches are all not sensitive to the parameter K, demonstrating the effectiveness of the online to batch conversion in AdaO2B. Besides, the data-dependent approach had lower variances w.r.t. the parameter K than the other two data selection approaches.

6 Conclusion

This paper aims to address the out-of-distribution generalization problem in online to batch conversion. Specifically, we propose an adaptive online to batch conversion approach called AdaO2B. The proposed AdaO2B is aware of distribution shifts through adaptively combining the model sequence using a weighting network, takes rigorous theoretical analyses as guidance, and achieves OOD generalization guarantees under the bandit feedback setting. Experimental results demonstrated the effectiveness of AdaO2B in out-of-distribution scenarios.

REFERENCES

Rie Kubota Ando, Tong Zhang, and Peter Bartlett. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(11), 2005.

Walid Bendada, Guillaume Salha, and Théo Bontempelli. Carousel personalization in music streaming apps with contextual bandits. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 420–425, 2020.

Ilai Bistritz, Zhengyuan Zhou, Xi Chen, Nicholas Bambos, and Jose Blanchet. Online EXP3 learning in adversarial bandits with delayed feedback. In *Advances in Neural Information Processing Systems 32*, pp. 11349–11358, 2019.

Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 1991.

Ashok Cutkosky. Anytime online-to-batch, optimism and acceleration. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1446–1454, 2019.

Ofer Dekel. From online to batch learning with cutoff-averaging. In *Advances in Neural Information Processing Systems 21*, 2008.

Ofer Dekel and Yoram Singer. Data-driven online to batch conversions. *Advances in Neural Information Processing Systems 18*, 2005.

Maria Dimakopoulou, Zhengyuan Zhou, Susan Athey, and Guido Imbens. Balanced linear contextual bandits. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 3445–3453, 2019.

Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 289–296, 2009.

Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian, and Vahab S. Mirrokni. Regret bounds for batched bandits. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 7340–7348, 2021.

Dylan J Foster, Akshay Krishnamurthy, and Haipeng Luo. Model selection for contextual bandits. *Advances in Neural Information Processing Systems 32*, 2019.

Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

Bent Fuglede and Flemming Topsoe. Jensen-Shannon divergence and Hilbert space embedding. In *Proceedings of the 2004 International Symposium on Information Theory*, pp. 31, 2004.

Avishek Ghosh, Abishek Sankararaman, and Ramchandran Kannan. Problem-complexity adaptive model selection for stochastic linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pp. 1396–1404, 2021.

Javad Hamidzadeh and Mona Moradi. Online recommender system considering changes in user's preference. *Journal of AI and Data Mining*, 9(2):203–212, 2021.

Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose H. Blanchet, Peter W. Glynn, and Yinyu Ye. Sequential batch learning in finite-action linear contextual bandits. *CoRR*, abs/2004.06321, 2020.

John R. Hershey and Peder A. Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pp. IV–317–IV–320, 2007.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems 19*, 2006.

Olivier Jeunen, David Rohde, Flavian Vasile, and Martin Bompaire. Joint policy-value learning for recommendation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1223–1233, 2020.

Rong Jin, Steven C.H. Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *Proceedings of the 21st International Conference on Algorithmic Learning Theory*, pp. 390–404, 2010.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. E. Knuth. *The Art of Computer Programming, Vol 2, Seminumerical Algorithms*. Addison-Wesley (2nd edition), 1998.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Ilja Kuzborskij and Francesco Orabona. Fast rates by transferring from auxiliary hypotheses. *Machine Learning*, 106(2):171–195, 2017.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 661–670, 2010.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. R-drop: Regularized dropout for neural networks. In *Advances in Neural Information Processing Systems 34*, pp. 10890–10905, 2021.

Nick Littlestone. From on-line to batch learning. In *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pp. 269–284, 1989.

Francesco Orabona, Marco Fornoni, and Nicolò Caputo, Barbara Cesa-Bianchi. OM-2: An online multi-class multi-kernel learning algorithm. In *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 43–50, 2010.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

Vianney Perchet, Philippe Rigollet, Sylvain Chassang, and Erik Snowberg. Batched bandit problems. In *Proceedings of the 28th Conference on Learning Theory*, pp. 1456, 2015.

Sashank Reddi, Barnabas Poczos, and Alex Smola. Doubly robust covariate shift correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Shai Shalev-Shwartz. *Online learning: Theory, algorithms, and applications*. PhD thesis, The Hebrew University of Jerusalem, 2007.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

Changjian Shui, Qi Chen, Jun Wen, Fan Zhou, Christian Gagné, and Boyu Wang. Beyond h-divergence: Domain adaptation theory with Jensen-Shannon divergence. *CoRR*, abs/2007.15567, 2020. URL `https://arxiv.org/abs/2007.15567`.

Yufei Tao and Shangqi Lu. From online to non-i.i.d. batch learning. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 328–337, 2020.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

Jiaqi Yang, Wei Hu, Jason D. Lee, and Simon Shaolei Du. Impact of representation learning in linear bandits. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Yuya Yoshikawa and Yusaku Imai. A nonparametric delayed feedback model for conversion rate prediction. *arXiv preprint arXiv:1802.00255*, 2018.

Shuai Zhang, Hongyan Liu, Jun He, Sanpu Han, and Xiaoyong Du. Deep sequential model for anchor recommendation on live streaming platforms. *Big Data Mining and Analytics*, 4(3):173–182, 2021a.

Xiao Zhang and Shizhong Liao. Online kernel selection via incremental sketched kernel alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3118–3124, 2018.

Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 41–50, 2021b.

Xiao Zhang, Sunhao Dai, Jun Xu, Zhenhua Dong, Quanyu Dai, and Ji-Rong Wen. Counteracting user attention bias in music streaming recommendation via reward modification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2504–2514, 2022.

Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou. Handling concept drift via model reuse. *Machine Learning*, 109(3):533–568, 2020.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A Detailed Proofs in Theoretical Analysis

A.1 Proof of Theorem 1

To facilitate the presentation of the proofs, we first provide the definition of Jensen-Shannon (JS) divergence.

Definition 2 (Jensen-Shannon (JS) Divergence (Fuglede & Topsoe, 2004))**.** *Given two distributions $\mathbb{Q}$ and $\mathbb{P}$, the Jensen-Shannon divergence between $\mathbb{Q}$ and $\mathbb{P}$ is*

$$\mathrm{D}_{\mathrm{JS}}(\mathbb{Q}\|\mathbb{P}) := \frac{1}{2}\left[\mathrm{D}_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{H}) + \mathrm{D}_{\mathrm{KL}}(\mathbb{P}\|\mathbb{H})\right],$$

where $\mathbb{H} = (\mathbb{Q}+\mathbb{P})/2$, and $\mathrm{D}_{\mathrm{KL}}(\mathbb{X}\|\mathbb{Y})$ denotes the Kullback-Leibler (KL) divergence between the distributions $\mathbb{X}$ and $\mathbb{Y}$.

Proof of Theorem 1. Letting

$$\bar{\boldsymbol{\theta}}_n = \sum_{i\in[n]} \beta_i \boldsymbol{\theta}_i / \beta_{1:n},$$

it is obvious that $\boldsymbol{\theta}_{\mathrm{ada}} = \bar{\boldsymbol{\theta}}_N$. We first bound the weighted regret of $\{\bar{\boldsymbol{\theta}}_n\}_{n\in[N]}$ as follows:

$$\begin{aligned}
&\sum_{n\in[N],b\in[B]} \beta_n \left[\left\langle \boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}}\right\rangle - \left\langle \bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}\right\rangle\right] \\
&= \sum_{n\in[N],b\in[B]} \left\{\beta_n \left\langle \boldsymbol{\theta}^* - \boldsymbol{\theta}_n, \boldsymbol{s}_{I_{n,b}}\right\rangle + \beta_n \left\langle \boldsymbol{\theta}_n - \bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}\right\rangle\right\} \\
&= \mathrm{WReg}(N,B) + \sum_{n\in[N],b\in[B]} \beta_{1:(n-1)} \left\langle \bar{\boldsymbol{\theta}}_n - \bar{\boldsymbol{\theta}}_{n-1}, \boldsymbol{s}_{I_{n,b}}\right\rangle,
\end{aligned} \tag{10}$$

where in the last equality we used $\bar{\boldsymbol{\theta}}_n = [\beta_{1:(n-1)}\bar{\boldsymbol{\theta}}_{n-1} + \beta_n\boldsymbol{\theta}_n]/\beta_{1:n}$. Here, with a slight abuse of notations, we define $\beta_{1:0} = 0$. From equation 10 we have

$$\begin{aligned}
&\sum_{n\in[N],b\in[B]} \beta_n \left\langle \boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}}\right\rangle \\
&= \mathrm{WReg}(N,B) + \sum_{n\in[N],b\in[B]} \beta_{1:(n-1)} \left\langle \bar{\boldsymbol{\theta}}_n - \bar{\boldsymbol{\theta}}_{n-1}, \boldsymbol{s}_{I_{n,b}}\right\rangle + \sum_{n\in[N],b\in[B]} \beta_n \langle \bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}\rangle. \\
&= \mathrm{WReg}(N,B) + \sum_{n\in[N],b\in[B]} \left[\beta_{1:n} \left\langle \bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}\right\rangle - \beta_{1:(n-1)} \left\langle \bar{\boldsymbol{\theta}}_{n-1}, \boldsymbol{s}_{I_{n,b}}\right\rangle\right] \\
&\le C_{\mathrm{WReg}} + \sum_{n\in[N],b\in[B]} \left[\beta_{1:n} \left\langle \bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}\right\rangle - \beta_{1:(n-1)} \left\langle \bar{\boldsymbol{\theta}}_{n-1}, \boldsymbol{s}_{I_{n,b}}\right\rangle\right].
\end{aligned} \tag{11}$$

We now take expectations on both sides of equation 11 and obtain that

$$B\beta_{1:N}\mathrm{ER}_{\mathbb{P}}(\boldsymbol{\theta}^*) \le \mathbb{E}_{\mathbb{P}}\left[C_{\mathrm{WReg}}\right] + B\sum_{n\in[N]} \left[\beta_{1:n}\mathrm{ER}_{\mathbb{P}}(\bar{\boldsymbol{\theta}}_n) - \beta_{1:(n-1)}\mathrm{ER}_{\mathbb{P}}(\bar{\boldsymbol{\theta}}_{n-1})\right]. \tag{12}$$

Note that the second term on the right side of equation 12 telescopes and becomes

$$B\beta_{1:N}[\mathrm{ER}_{\mathbb{P}}(\boldsymbol{\theta}^*) - \mathrm{ER}_{\mathbb{P}}(\bar{\boldsymbol{\theta}}_N)] \le \mathbb{E}_{\mathbb{P}}\left[C_{\mathrm{WReg}}\right]. \tag{13}$$

Combining $\mathrm{GE}_{\mathbb{P}}(\bar{\boldsymbol{\theta}}_N) - \mathrm{GE}_{\mathbb{P}}(\boldsymbol{\theta}^*) = \mathrm{ER}_{\mathbb{P}}(\boldsymbol{\theta}^*) - \mathrm{ER}_{\mathbb{P}}(\bar{\boldsymbol{\theta}}_N)$, $\boldsymbol{\theta}_{\mathrm{ada}} = \bar{\boldsymbol{\theta}}_N$, and equation 13 yields that

$$\mathrm{GE}_{\mathbb{P}}(\boldsymbol{\theta}_{\mathrm{ada}}) \le \mathrm{GE}_{\mathbb{P}}(\boldsymbol{\theta}^*) + \mathbb{E}_{\mathbb{P}}\left[\frac{C_{\mathrm{WReg}}}{B\beta_{1:N}}\right]. \tag{14}$$

Motivated by the information theoretical tools in OOD generalization (Shui et al., 2020), we have

$$\begin{aligned}
&\mathrm{GE}_{\mathbb{Q}}(\boldsymbol{\theta}_{\mathrm{ada}}) - \mathrm{GE}_{\mathbb{P}}(\boldsymbol{\theta}_{\mathrm{ada}}) \\
&\le \frac{\max_{\mathbb{T},\boldsymbol{\theta}} \mathrm{GE}_{\mathbb{T}}(\boldsymbol{\theta}) - \min_{\mathbb{T},\boldsymbol{\theta}} \mathrm{GE}_{\mathbb{T}}(\boldsymbol{\theta})}{\sqrt{2}} \sqrt{\mathrm{D}_{\mathrm{JS}}(\mathbb{Q}\|\mathbb{P})} \\
&\le \frac{2C_{\mathrm{ER}}}{\sqrt{2}} \sqrt{\mathrm{D}_{\mathrm{JS}}(\mathbb{Q}\|\mathbb{P})}.
\end{aligned} \tag{15}$$

Finally, combining equation 14 and equation 15 concludes the proof. □

A.2 PROOF OF COROLLARY 1

Proof of Corollary 1. For the mixture distribution $\mathbb{H} = (\mathbb{P} + \mathbb{Q})/2$, its distribution density $h(\boldsymbol{s})$ can be represented as the following sum of multiple sub-distribution densities $\{h_i\}_{i\in[W]}$:

$$h(\boldsymbol{s}) = \sum_{i\in[W]} \frac{h_i}{W}, \quad h_i = \frac{W}{2}\left[\pi_i \cdot p_i(\boldsymbol{s}) + \tau_i \cdot q_i(\boldsymbol{s})\right].$$

Using the chain rule for Kullback-Leibler (KL) divergence (Cover & Thomas, 1991), we can obtain that the KL divergence (denoted by $\mathrm{D}_{\mathrm{KL}}(\cdot\|\cdot)$) between $\mathbb{P}$ and $\mathbb{H}$ as well as $\mathbb{Q}$ and $\mathbb{H}$ can be bounded as follows:

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(\mathbb{P}\|\mathbb{H}) &\le \log W - H(\boldsymbol{\pi}) + \sum_{i\in[W]} \pi_i \cdot \mathrm{D}_{\mathrm{KL}}(p_i\|h_i), \\ \mathrm{D}_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{H}) &\le \log W - H(\boldsymbol{\tau}) + \sum_{i\in[W]} \tau_i \cdot \mathrm{D}_{\mathrm{KL}}(q_i\|h_i), \end{aligned} \tag{16}$$

where $H(\boldsymbol{\pi})$ and $H(\boldsymbol{\tau})$ denote the entropy of distributions $\boldsymbol{\pi}$ and $\boldsymbol{\tau}$, respectively.

From the variational upper bound of KL divergence (Hershey & Olsen, 2007), we can bound the KL divergences $\mathrm{D}_{\mathrm{KL}}(p_i\|h_i)$ as follows:

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(p_i\|h_i) &\le \frac{W\pi_i}{2}\mathrm{D}_{\mathrm{KL}}(p_i\|p_i) + \frac{W\tau_i}{2}\mathrm{D}_{\mathrm{KL}}(p_i\|q_i), \\ &= \frac{W\tau_i}{2}\mathrm{D}_{\mathrm{KL}}(p_i\|q_i). \end{aligned} \tag{17}$$

Similarly, we obtain the upper bound of $\mathrm{D}_{\mathrm{KL}}(q_i\|h_i)$ as

$$\mathrm{D}_{\mathrm{KL}}(q_i\|h_i) \le \frac{W\pi_i}{2}\mathrm{D}_{\mathrm{KL}}(q_i\|p_i). \tag{18}$$

Combining equation 16, equation 17, and equation 18 yields the following upper bound of the JS divergence $\mathrm{D}_{\mathrm{JS}}(\mathbb{Q}\|\mathbb{P})$:

$$\begin{aligned} &2\mathrm{D}_{\mathrm{JS}}(\mathbb{Q}\|\mathbb{P}) \\ &= \mathrm{D}_{\mathrm{KL}}(\mathbb{P}\|\mathbb{H}) + \mathrm{D}_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{H}) \\ &\le 2\log W - H(\boldsymbol{\pi}) - H(\boldsymbol{\tau}) + W\sum_{i\in[W]} \pi_i\tau_i \cdot \mathrm{D}_{\mathrm{BKL}}(p_i\|q_i) \\ &\le 2\log W + W\sum_{i\in[W]} \pi_i\tau_i \cdot \mathrm{D}_{\mathrm{BKL}}(p_i\|q_i), \end{aligned}$$

where $\mathrm{D}_{\mathrm{BKL}}(p_i\|q_i) := \frac{1}{2}\left[\mathrm{D}_{\mathrm{KL}}(p_i\|q_i) + \mathrm{D}_{\mathrm{KL}}(q_i\|p_i)\right]$ denotes the bidirectional KL divergence between p_i and q_i (Liang et al., 2021). □

A.3 PROOF OF COROLLARY 2

Proof of Corollary 2. Using the linearization trick of convex functions (Shalev-Shwartz, 2011), we can obtain that

$$\begin{aligned} &\sum_{n\in[N],b\in[B]} \beta_n \left[r(\boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}}) - r(\bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}})\right] \\ &\le \sum_{n\in[N],b\in[B]} \beta_n \left[\left\langle \boldsymbol{\theta}^* - \bar{\boldsymbol{\theta}}_n, \nabla_{\boldsymbol{\theta}^*} r(\boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}})\right\rangle\right]. \end{aligned} \tag{19}$$

Similarly, from $\mathbb{E}[\nabla_{\boldsymbol{\theta}_n} r(\boldsymbol{\theta}_n, \boldsymbol{s})] = \nabla_{\boldsymbol{\theta}^*} r(\boldsymbol{\theta}^*, \boldsymbol{s})$, $\forall n \in [N]$, we have

$$\begin{aligned} &\sum_{n\in[N],b\in[B]} \beta_{1:(n-1)} \left\langle \bar{\boldsymbol{\theta}}_n - \bar{\boldsymbol{\theta}}_{n-1}, \nabla_{\boldsymbol{\theta}^*} r(\boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}})\right\rangle \\ &\le \mathbb{E}\left\{\sum_{n\in[N],b\in[B]} \beta_{1:(n-1)} \left[r(\bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}) - r(\bar{\boldsymbol{\theta}}_{n-1}, \boldsymbol{s}_{I_{n,b}})\right]\right\}, \end{aligned} \tag{20}$$

Treating $\nabla_{\boldsymbol{\theta}^*} r(\boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}})$ as the context received at step b in the b-th episode, and substituting equation 19 and equation 20 into equation 10 in the proof of Theorem 1, we have

$$\sum_{n\in[N],b\in[B]} \beta_n \left[r(\boldsymbol{\theta}^*, \boldsymbol{s}_{I_{n,b}}) - r(\bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}})\right]$$

$$\leq \mathrm{WReg}(N, B) + \mathbb{E}\left\{\sum_{n\in[N],b\in[B]} \beta_{1:(n-1)} \left[r(\bar{\boldsymbol{\theta}}_n, \boldsymbol{s}_{I_{n,b}}) - r(\bar{\boldsymbol{\theta}}_{n-1}, \boldsymbol{s}_{I_{n,b}})\right]\right\}. \tag{21}$$

Similarly to the proof of equation 11 and equation 12, using equation 21 as a tool, we can obtain the same upper bound of the expected convex rewards as in Theorem 1, yielding the final OOG generalization error bound. □

B Diagram illustrating three data selection approaches

In this section, we present graphical representations for three data selection approaches used in managing data buffers as described in Section 4.1. These approaches include the sliding window approach, reservoir sampling approach, and data-dependent approach.

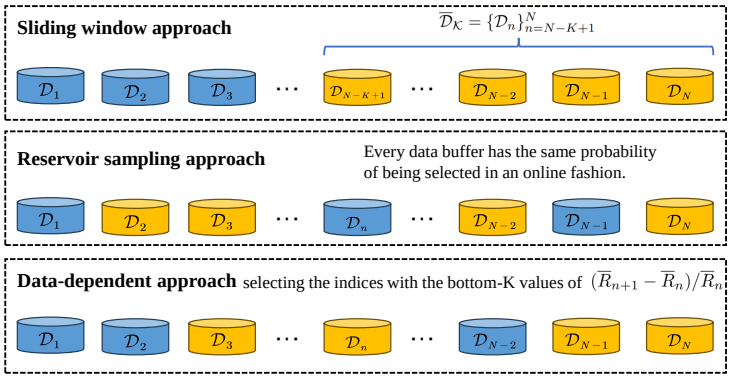


Figure 5: Diagram illustrating three data selection approaches for managing data buffers in Section 4.1.

C Detailed Experimental Settings and More Experimental Results

C.1 Implementation Details

In our implementation on both synthetic and real-world data, we trained AdaO2B based on the last 10 (*i.e.,*$K = 10$) data buffers and history policies. We tuned the hyper-parameters as follows: the learning rate was tuned among $\{1e{-}2, 1e{-}3, 1e{-}4, 1e{-}5\}$, the weight decay was tuned among $\{1e{-}3, 1e{-}4, 1e{-}5, 1e{-}6\}$, and the batch size was tuned in $\{256, 512, 1024, 2048\}$. Besides, the hidden dimensions of the MLP_h were set as $[64, 64]$. For AdaO2B, SBUCB and BLTS-B, we tuned the regularization parameter in $[0.2 : +0.2 : 2]$. For EXP3-B, we tuned the exploitation parameter in $[0.1 : +0.1 : 1]$. All the algorithms were trained on a single Tesla V100.

C.2 Data Preparation

Synthetic Dataset. To simulate the OOD scenario, we generated the synthetic data as follows: number of episodes $N = 40$, batch size $B = 5,000$ number of candidates $M = 10$, and the context dimension $d = 10$. For OL-Data, we drew candidate context set $\mathcal{S}_{n,b} \subseteq \mathbb{R}^d$ from a Gaussian distribution $\mathcal{N}(\mu_{\mathrm{s}}\mathbf{1}_d, \sigma_{\mathrm{s}}^2 \boldsymbol{I}_d)$, where in the first 20 episodes, the means of candidate contexts were $\mu_{\mathrm{s}} \in [1 : -0.4 : -2.6]$ and the standard deviation was $\sigma_{\mathrm{s}} = 0.05$. To simulate the mixture

distribution, we set the mean of the first candidate context to 1.2 in the last 20 episodes. Compared to OL-Data, the mean of the first candidate context in BT-Data was set to 1.4 for simulating the distribution shifts in testing data. The observed reward (user feedback) given the context $\boldsymbol{s}_{n,b}$ was set to a sigmoid function $\text{sigmoid}(\langle \boldsymbol{w}_{\text{r}}, \boldsymbol{s}_{n,b} \rangle)$, where each element of the coefficient vector $\boldsymbol{w}_{\text{r}} \in \mathbb{R}^d$ was sampled according to a Gaussian distribution as $\mathcal{N}(0.1, 0.01^2)$.

KuaiRec dataset. We used the KuaiRec dataset[4], which collected from the recommendation logs of a popular video-sharing mobile app Kuaishou and is the the first dataset that contains a fully observed user-item interaction matrix. The final dataset after filtering comprises 6980 unique users, 973 unique videos, and 400,000 user-video interactions. KuaiRec also provided various side information of both users and videos. Particularly, KuaiRec provided the daily item features, *i.e.,*the same item may have different features on different date, which naturally satisfy our research target. Following the practices in (Zhang et al., 2021b; 2022; Yoshikawa & Imai, 2018), we processed the categorical features as one-hot vectors and then concatenated them to the integer features. The final dimension of the feature vectors was reduced to 50 through principal component analysis (PCA), *i.e.,*$d = 50$. For the candidate set, we retained the original item for each interaction and then randomly sampled 99 extra items from the rest item set.

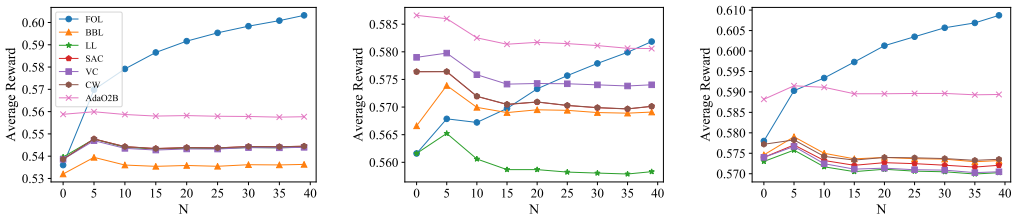
C.3 Evaluation Metrics

For both synthetic and real-world datasets, we split them into two subsets for the online learning phase (as well as the O2B conversion phase) and the batch testing phase, respectively, denoted by **OL-Data** and **BT-Data**. Following the standard practice in (Zhang et al., 2021b), we use the average reward to evaluate the accuracy of algorithms as $\frac{1}{nB}\sum_{k=1}^{n}\sum_{b=1}^{B} R_{k,b}$ for the first n episodes, where $R_{k,b}$ is the observed reward at step b in the k-th episode.

C.4 Evaluation Protocols on Real-World Data

In online recommendation, we cannot guarantee that the corresponding feedback of each recommended item to the user can be found in the log data. To overcome this issue and facilitate ground-truth evaluations, following (Jeunen et al., 2020; Zhang et al., 2022), we created a simulated online environment to test all the algorithms. More specifically, we first trained a matrix factorization (MF) model (Koren et al., 2009) using both OL-Data and BT-Data. AUC of the trained MF model were both over 83%, which assures that the online environment can provide nearly realistic feedbacks of users. At each step, the online environment received a selected context (*i.e.,*a recommended item) from the algorithm, and returned a user feedback (1 or 0) according to $\mathbb{I}(\hat{y} > \gamma)$, where $\mathbb{I}(\cdot)$ is an indicator function, $\hat{y}$ is the predicted score by the trained MF model and γ is a tuned threshold that the AUC can achieve the highest score.

C.5 Complete Results of Reward Curves on KuaiRec Dataset



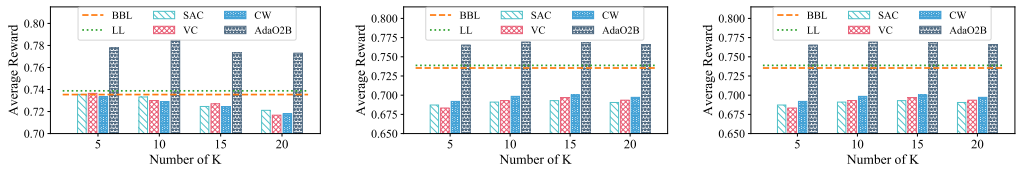
(a) SBUCB on KuaiRec Dataset (b) EXP3-B on KuaiRec Dataset (c) BLTS-B on KuaiRec Dataset

Figure 6: Average rewards of baselines and the proposed AdaO2B equipped with the best data selection approach on testing data of KuaiRec dataset, where N denotes the number of episodes in the batch testing phase.

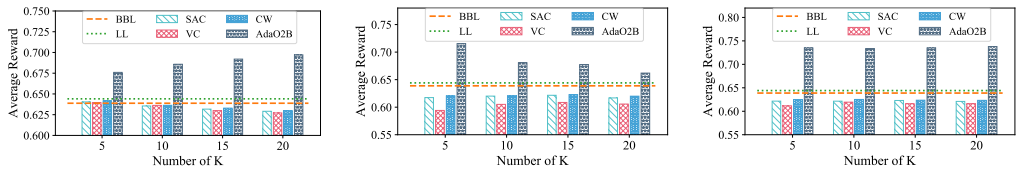
[4]https://kuairec.com

C.6 Complete Results of Parameter Analysis

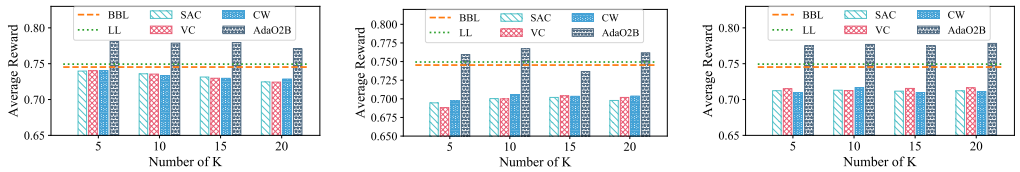
This section empirically studied the impact of the number of candidate policies (*i.e.,*K). A larger K means the, more candidate policies and data buffers are stored for conducting the O2B conversion. We conducted experiments to test the performance of AdaO2B with different K values. The results illustrated in Figure 7 and Figure 8 indicate that AdaO2B equipped with different bandit backbones and data selection approaches are all not sensitive to the parameter K, demonstrating the effectiveness of the online to batch conversion in AdaO2B. Besides, the data-dependent approach had lower variances w.r.t. the parameter K than the other two data selection approaches.



(a) Bandit backbone: SBUCB; Data selection: S (b) Bandit backbone: SBUCB; Data selection: R (c) Bandit backbone: SBUCB; Data selection: D

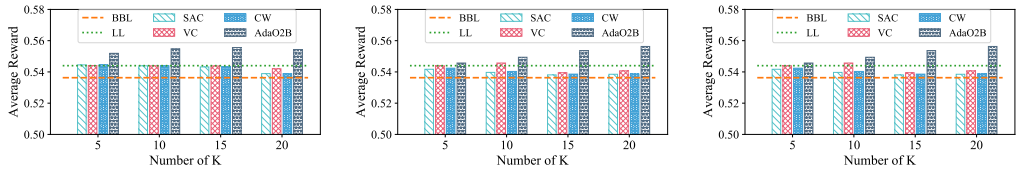


(d) Bandit backbone: EXP3-B; Data selection: S (e) Bandit backbone: EXP3-B; Data selection: R (f) Bandit backbone: EXP3-B; Data selection: D

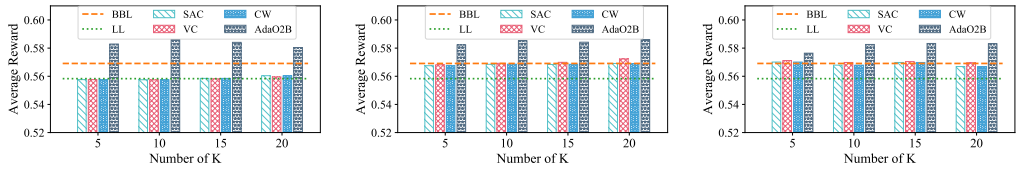


(g) Bandit backbone: BLTS-B; Data selection: S (h) Bandit backbone: BLTS-B; Data selection: R (i) Bandit backbone: BLTS-B; Data selection: D

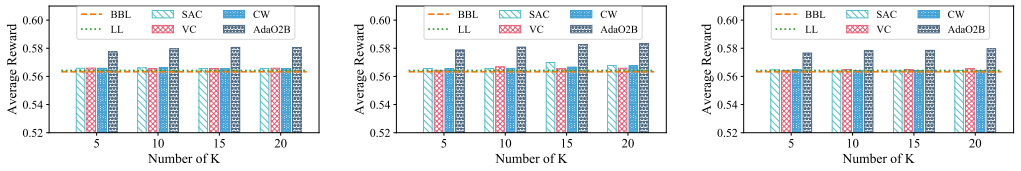
Figure 7: Performance comparison of AdaO2B and all baselines with different number of candidate policies K and different data selection process on the synthetic dataset and KuaiReC dataset. "S/R/D" denote the Sliding window approach, Reservoir sampling approach, and Data-dependent approach, respectively.



(a) Bandit backbone: SBUCB; Data selection: S (b) Bandit backbone: SBUCB; Data selection: R (c) Bandit backbone: SBUCB; Data selection: D



(d) Bandit backbone: EXP3-B; Data selection: S (e) Bandit backbone: EXP3-B; Data selection: R (f) Bandit backbone: EXP3-B; Data selection: D



(g) Bandit backbone: BLTS-B; Data selection: S (h) Bandit backbone: BLTS-B; Data selection: R (i) Bandit backbone: BLTS-B; Data selection: D

Figure 8: Performance comparison of AdaO2B and all baselines with different number of candidate policies K and different data selection process on the real-world KuaiRec dataset. "S/R/D" denote the Sliding window approach, Reservoir sampling approach, and Data-dependent approach, respectively.