Improving Long-Term Metrics in Recommendation Systems using Short-Horizon Reinforcement Learning

Bogdan Mazoure ✉ [1,2,†], Paul Mineiro[3], Pavithra Srinath[6,7,†], Reza Sharifi Sedeh[4,†], Doina Precup[1,2,5], and Adith Swaminathan[3]

[1] McGill University
[2] Quebec AI Institute (Mila)
[3] Microsoft Research
[4] Meta
[5] DeepMind
[6] The Trade Desk
[7] Imperial College London

Abstract. We study session-based recommendation scenarios where we recommend items to users during sequential interactions to improve their long-term utility. Optimizing a long-term metric is challenging because the learning signal (whether the recommendations achieved their desired goals) is delayed and confounded by interactions across many sessions. Optimizing session-specific metrics as a proxy can be suboptimal because they ignore the effects of the recommendation policy across sessions. We develop a reinforcement learning (RL) algorithm called Short Horizon Policy Improvement (SHPI) that approximates policy-induced changes in user behavior across sessions. SHPI modifies episodic RL algorithms by additionally giving a termination bonus in each session. To remain computationally tractable we estimate termination bonuses from logged observational data, and SHPI finds policy improvements that other RL methods can miss. Results on four recommendation tasks show that SHPI can outperform matrix factorization for offline metrics, bandits for myopic online metrics, and RL for long-term metrics.

1 Introduction

High quality recommendations can provide personalized experiences for users and drive increased usage, revenue and utility for service providers. These recommendations do not occur as isolated interactions, but rather as repeated sequential interactions with users. For instance, mobile health interventions nudge users over time to achieve their long-term goals [13] and e-commerce sites personalize product suggestions within user sessions to maximize their likelihood of purchases [16].

† Work done while the author was at Microsoft.
✉ Correspondence to bogdan.mazoure@mail.mcgill.ca.

Traditionally, recommender systems use immediately measurable user feedback (e.g., implicit feedback like clicks or explicit feedback like item ratings) to identify good recommendations. For sequential recommendations, these myopic feedbacks are imperfect proxies for Long-Term Rewards (LTR). Consider an e-commerce site optimizing for total monthly conversions by its user population:

$$LTR = \frac{Conv}{Month} = \frac{Conv}{Rec}\frac{Recs}{Session}\frac{Sessions}{Month}.$$

Recommenders that optimize immediate user feedback are too myopic because they greedily optimize the first term even if that lowers the other two terms.

Consequently, reinforcement learning (RL) techniques have been applied to optimize for LTR [2]. These techniques require many user interactions, with the sample complexity of RL growing with the horizon of the problem [27]. Fortunately, user interactions in many recommendation applications typically occur in short bursts called sessions. Episodic RL algorithms are well-suited for optimizing session-specific metrics where each session is treated as an independent episode. Real-world recommenders that optimize session-specific metrics perform better than click-optimizing policies [2], yet often encounter an offline-online gap where policies that perform well on offline data fail during online deployment [6]. As we will show, this is because episodic RL for session-based recommendations can still be too myopic.

Fig. 1 illustrates an example where each session length is 2 and there are a maximum of 5 sessions in a month. When users visit the site, they have a limited amount of patience to interact and find an appropriate product. Within a user session, we may recommend attractive products which advertises our item inventory for a user's future shopping needs or we may recommend more functional products that address their immediate need. Consider first a deployed system μ that recommends items uniformly at random. All items have some non-zero user exposure and so users initiate sessions for all their intents. However, uniform random is not a good recommendation policy so μ sees very few conversions.

Next consider a recommender that has been tuned to optimize for conversions myopically using data collected from μ. It infers that there is a small subset of items that have good conversion rates and only recommends these most popular items. However, users exposed to only a small subset of items may never initiate sessions for other intents, leading to fewer sessions and eventually sub-optimal total conversions.

Suppose instead we optimized session-based recommendations using episodic RL, where each episode corresponds to a user session. Such a policy would recommend functional items which increase the chance of a conversion within each session. This is an optimal policy *if* the distribution of sessions matches what is seen in the training data. Unfortunately, the deployed policy can influence which sessions are initiated by users. Therefore, although the deployed policy has a 100% session conversion rate, it has fewer sessions overall.

One non-solution is to ignore sessions and instead treat the duration of the LTR as the RL problem horizon. In the example above this would treat each month of user activity as an episode. However, the sample complexity of RL

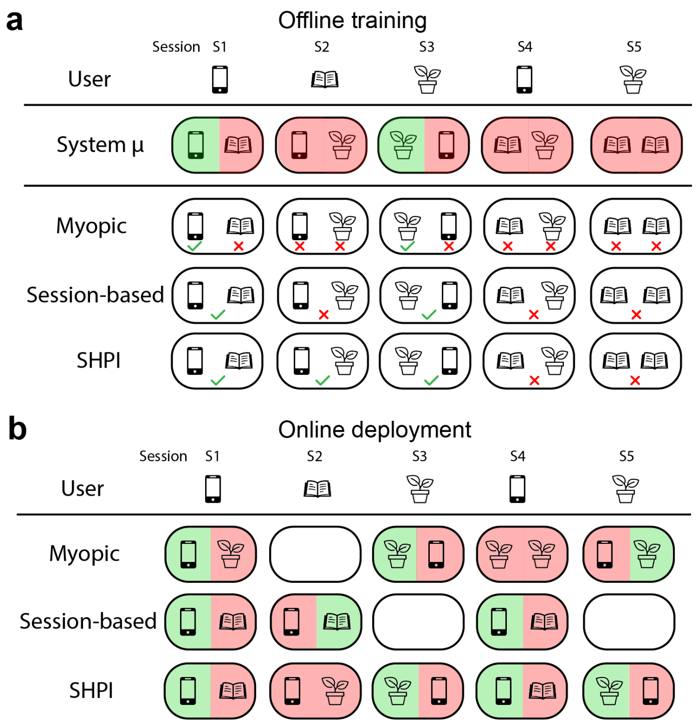

Fig. 1. Stylized example of e-commerce recommendations. A session S_i lasts for two interactions with conversions (green) or not (red). Users' propensity to initiate sessions depends on their past interactions. (a) Myopic bandit, Session-based RL and SHPI infer different training signals in logged data collected from uniform random μ. Myopic bandit infers every user interaction that converts as a positive signal. Session-based RL infers every session containing a conversion as a positive signal. SHPI additionally infers a positive signal for sessions that lead to more future user sessions by using a session-termination bonus. Session-based RL ignores that session initiation is affected by the policy and infers low reward for S_2. SHPI correctly identifies that good discovery recommendations in S_2 increase the number of sessions and thereby total conversions. (b) During online deployment, Myopic bandit and Session-based RL always recommend items with high conversion rate, missing out on other session intents. SHPI trades-off between success within a session (e.g., S_2) and initiating more sessions, and thereby manages to accumulate more conversions in a month.

scales with episode length. How can we optimize for better session-based recommendations without paying the statistical costs for the LTR horizon? Notice that the key difficulty is that the number of sessions in a month is influenced by the policy. In this paper, we approximate this dependence by first estimating long-term user behavior (e.g., $\frac{Sessions}{Month}$) separately using logged data. Then, by invoking episodic RL algorithms with an additional bonus on episode termination to capture policy-induced user shifts, we develop a practical algorithm called

Short Horizon Policy Iteration (SHPI). SHPI transforms the sparse long-term outcomes recorded in a dataset of sequential recommendations into a reliable training signal. This transformation is simple and highly composable with many other offline RL and user-embedding techniques.

We conduct experiments in four domains: a synthetic task where the reward is a challenging non-convex function of user context, a recommender system benchmark with a sub-modular reward [20], a real-world private recommendation dataset and an HIV treatment domain with delayed rewards [4]. Our experiments show that SHPI is simple to run with offline data, and outperforms session-based RL, contextual bandit and classic recommendation baselines.

2 Related Work

Recommendations for LTR. Our work builds on [25] who identified a trade-off in online bandits between exploring new items, exploiting items for immediate utility, and exploiting other items to bring back returning users. Our work extends their reasoning from online bandits to offline episodic RL, and generalizes their approach using episodic termination bonuses.

Session-based recommendations are widely-studied, see [1] for a recent survey. State of the art techniques use session summarization for user embeddings (e.g., via recurrent neural nets), and blend short and long-term metrics (e.g., via weighted rewards). SHPI is complementary to these techniques, and additionally reduces the horizon in offline LTR optimization by transforming the reward.

Hierarchical RL has been proposed for optimizing long-term metrics across sessions [26]. The key idea is to train two separate policies: a meta-controller for switching across sessions and an intra-session recommender policy [16]. Although this hierarchy conforms to the LTR session structure, credit assignment between the meta-controller and intra-session policy are difficult, complicating the meta-controller training [12]. SHPI avoids training a meta-controller, and instead modifies the intra-session policy learning using session-termination bonuses.

Offline/Batch RL techniques. Offline RL techniques like BCQ [5] and CQL [10] control extrapolation errors outside the logged dataset of user-item interactions. SHPI is composable with many of these techniques (see Appendix 6.3 for additional experiments). However SHPI is not a general-purpose offline RL algorithm. It solves for a much shorter horizon than the original problem which can be disastrous; for instance, SHPI will struggle on offline RL benchmarks which do not have session structure.

Truncated horizon reasoning in other settings. GAE [22] is a widely used online RL algorithm, but uses a value function estimate V^{π} to derive a consistent estimate of policy advantages. SHPI uses V^{μ} instead (see equation 4) which is inconsistent for solving a general RL problem. However we empirically show that it is sufficient for fixing the myopia of session-based RL. HURL [3] establishes sufficient theoretical conditions for truncated horizon reasoning that are also needed for SHPI, and additionally increases the problem horizon gradually during training. All of these works are in the online interactive setting for general RL distinct from our batch RL session-based setting.

3 Background

Notation. Lowercase letters x represent either random variables or their realizations (depending on context), uppercase bold letters $\mathbf{A}$ represent matrices, lowercase bold letters $\boldsymbol{v}$ represent vectors, calligraphic letters $\mathcal{T}$ represent tensors and/or spaces. Index notation is the equivalence $w_{a:b} = \{w_a, w_{a+1}, .., w_{b-1}, w_b\}$. For a space $\mathcal{X}$, $\Delta(\mathcal{X})$ represents the set of distributions with support $\mathcal{X}$. $\bigvee$ denotes the max operator to save space when necessary.

3.1 Recommendation as Episodic MDP

The notation used throughout this paper follows a single user u, where dependence is specified by subscript X_u; generalization to a user population is accomplished using user-specific features X_u in the user context. The user's preferences are assumed to be stationary across time. There is also a notion of a task horizon, e.g., one month. We seek a recommendation policy that maps user contexts to valid recommendations so as to maximize total utility over the horizon. We formulate this problem as an episodic Markov Decision Process (MDP) with a long horizon. The results in this paper directly translate to infinite horizon discounted objectives.

An episodic MDP is defined by the tuple $(\mathcal{X}, \mathcal{A}, r, \mathcal{T}, p_0, \gamma, T)$ where $\mathcal{X}$ is the state space, $\mathcal{A}$ the action space, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ the reward function, $\gamma \in [0, 1]$ the discount factor and T the maximum length of an episode. $\mathcal{T} : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$ is a Markov transition function and $p_0 \in \Delta(\mathcal{X})$ is the initial state distribution.

Unlike a general MDP, recommendation scenarios have additional structure, which can be represented using a factorization of the context or state space $x = [x_s, x_a, x_u]$ where x_s encodes within-session context, x_a encodes action-dependent features, for example the valid recommendations in that context, and x_u encodes across-session user profiles. If we observe only session-based rewards (e.g., conversions on an e-commerce site) they can be encoded using $r(x, a) = r(x_s)$.

At each $t = 0, 1, 2, .., T$, the system agent observes $x_t \in \mathcal{X}$ and picks an action $a_t \in \mathcal{A}$ based on some behavior policy $\mu : \mathcal{X} \to \Delta(\mathcal{A})$. The goal of a learning agent is to maximize the expected cumulative rewards after T timesteps. Finally we assume access to a historical dataset $\mathcal{D}$ of (context, action, reward) trajectories collected from an existing system μ (for instance, one that is tuned using bandit algorithms for optimizing clicks). We must use $\mathcal{D}$ to find a new policy π that gets reliably better cumulative rewards than μ.

3.2 RL Solution Concepts

Let $\mathbb{P}^{\mu}_{0:t}$ denote "rolling-in" with μ: $x_0 \sim p_0, a_0 \sim \mu(x_0), \dots x_t \sim \mathcal{T}(\cdot|x_{t-1}, a_{t-1})$ and $\mathbb{P}^{\mu}_{t}$ denote "rolling out" with μ: $\mathbb{P}^{\mu}_{t} = \mathbb{E}_{\mu}[\mathcal{T}(\cdot|x_{t-1}, a_{t-1})]$.

The *state value function* under any policy π is

$$V^{\pi}_t(x_t) = \mathbb{E}_{\mathbb{P}^{\pi}_t}[\sum_{m=0}^{T-t} \gamma^m r(x_{t+m}, a_{t+m})|x_t],$$

the *state-action value function* is

$$Q^{\pi}_t(x_t, a_t) = \mathbb{E}_{\mathbb{P}^{\pi}_t}[\sum_{m=0}^{T-t} \gamma^m r(x_{t+m}, a_{t+m})|x_t, a_t],$$

and the *advantage function* is $A^{\pi}_t(x_t, a_t) = Q^{\pi}_t(x_t, a_t) - V^{\pi}_t(x_t)$. Note that we have time-indexed value and advantage functions; however, we drop the index because when T is very large these functions are time-invariant under the assumption that user preferences are stationary. The *expected return* is $\eta^{\pi} = \mathbb{E}_{p_0}[V^{\pi}_0(x_0)] = \mathbb{E}_{\mathbb{P}^{\pi}_{0:t}}[\sum_{i=0}^{t-1} \gamma^i r_i + V^{\pi}_t(x_t)] = \mathbb{E}_{\mathbb{P}^{\pi}_{0:t}}[\sum_{i=0}^{t-1} \gamma^i r_i + Q^{\pi}_t(x_t, a_t)]$ for $1 \leq t \leq T$.

The online/interactive/on-policy RL problem is to find a π by interacting with users such that η^{π} is near-optimal. The offline/batch RL problem attempts to find such a π using only the logged dataset $\mathcal{D}$. In practical recommendation settings, we have a large $\mathcal{D}$ and also a small budget for online user interactions. In Sec. 4 we provide practical online and offline RL algorithms for both settings.

Policy Gradient A typical interactive RL strategy attempts to maximize $\eta^{\pi} \equiv \arg\max \mathbb{E}_{\mathbb{P}^{\pi}}[A^{\pi}(x, a)]$ by collecting samples from π interacting with the environment to estimate the expectation $\mathbb{P}^{\pi}$ and the advantages A^{π}. If online learning is impossible due to the absence of a simulator or is too costly (e.g., in medical treatment domains), an improved policy π can still be found for a given dataset gathered by behavior policy μ. One approach uses ideas inspired by the policy gradient of $\nabla\eta^{\mu}$, and essentially maximizes $\mathbb{E}_{\mathbb{P}^{\mu}}[A^{\mu}(x, a)]$. However the policy improvement step when following this strategy tends to be very small (see [9] for a discussion). Moreover, when we need to make a sequence of good recommendations within a session for user conversion, then one-step deviations from μ as estimated via $\hat{Q}^{\mu}(x_t, a^*) \approx 0$ (even for a good recommendation a^*). So policy gradient techniques may completely miss avenues for improvement over μ.

Conservative Policy Improvement Another strategy uses a performance difference lemma [9] to relate $\eta^{\pi} - \eta^{\mu} = \mathbb{E}_{\mathbb{P}^{\mu}}[-A^{\pi}(x, a)] = \mathbb{E}_{\mathbb{P}^{\pi}}[A^{\mu}(x, a)]$. Batch RL techniques that attempt to maximize policy improvement by optimizing $\mathbb{E}_{\mathbb{P}^{\mu}}[A^{\pi}(x, a)]$ face the challenge of extrapolation errors, since A^{π} estimated from samples from μ may not be reliable in important regions of the state-action space visited by π. Techniques that optimize $\mathbb{E}_{\mathbb{P}^{\pi}}[A^{\mu}(x, a)]$ like SPIBB [11] and MBS [15] suffer from distribution shift since samples collected from μ may not estimate expectations w.r.t $\mathbb{P}^{\pi}$.

Truncated horizon We can solve an artificially shorter horizon RL problem, by either treating $H < T$ as effective task horizon or by decreasing the discount factor to $\gamma' < \gamma$. User sessions are a natural and popular choice for defining truncated episodes of length up to H. This essentially assumes that the state $x_{H+1} \sim p_0$ whereas in reality $x_{H+1} \sim \mathcal{T}(\cdot|x_H, a_H)$. We saw in Sec. 1 that when policy actions within a user session influence the user population who initiate future sessions, $\mathbb{P}^\pi[x_{H+1}] \neq p_0$.

We now identify an alternative solution strategy for truncated horizon reasoning that leverages the session structure of user interactions without assuming that $\mathbb{P}^\pi[x_{H+1}] \approx p_0$.

4 Algorithm

Can we estimate how the distribution over sessions depends on any particular recommendation policy? In some very simple cases, the answer is yes. For instance, suppose a user u had a known, intrinsic *session initiation rate* $c(x_u)$. Then estimating $\mathbb{P}^\pi[x_u]$ and computing $\mathbb{E}_{\mathbb{P}^\pi(x_u)}[c(x_u)]$ gives us the correct policy-dependent session distribution. Our key insight is that we can learn a suitable bonus function like $c(x_u)$ that, when invoked on the distribution of terminal states produced by an episodic RL agent, gives the correct long-term signal for learning. For example, suppose we are given the state value function for an optimal policy π^* for our problem, $V^*(x)$. Bellman optimality equations state that the π that maximizes the instantaneous re-shaped reward, $\pi(x) = \arg\max_{a\in\mathcal{A}}\{r(x,a) + \gamma\mathbb{E}_{x'\sim\mathcal{T}(\cdot|x,a)}[V^*(x')]\}$ is an optimal policy. In other words, if we use V^* as a bonus function then even a myopic bandit algorithm that optimizes a re-shaped reward will recover the optimal policy.

Since we do not know V^*, we begin by first estimating V^μ with classic policy evaluation techniques to use as a bonus instead (Sec. 4.1). Then we incorporate these estimates in an online on-policy iteration scheme (Sec. 4.2) and later extend it to the offline off-policy setting (Section 4.3).

4.1 Estimating Termination Bonus V^μ

Given a dataset $\mathcal{D} = \{(x_i \sim \mathbb{P}^\mu[x], a_i \sim \mu(\cdot|x_i), {x'}_i \sim \mathcal{T}(\cdot|x_i, a_i), r_i \sim r(x,a))\}_{i=1}^n$, we estimate $V^\mu(x)$ using Bellman residual minimization:

$$\hat{V}^\mu = \arg\min_V \sum_{(x,a,r,x')\in\mathcal{D}} (r + \gamma V(x') - V(x))^2 \,. \tag{1}$$

In practice we use SGD on $\mathcal{D}$ to optimize Eq. 1. When users have an intrinsic session initiation rate $c(x_u)$ and the within-session rewards are independent of session-initiations, $\hat{V}^\mu(x_u)$ can approximate the long-term value of user x_u, $V^\mu(x_u) \approx c(x_u)$. With this approximation to $c(x_u)$ in hand, we can derive $\mathbb{E}_{\mathbb{P}^\pi(x_u)}[c(x_u)]$ (by using $\hat{V}^\mu$ as a termination bonus) which gives a useful summary of long-term behavior under π during training. However using V^μ to summarize the long-term signal instead of V^* or even V^π can introduce approximation error in more general user behavior models.

4.2 Online Short-Horizon Policy Iteration

We next combine Monte Carlo samples collected from online interactions along with the evaluated V^μ model to estimate k-step advantages:

$$\begin{aligned}\mathbb{A}^\pi_{(k),t}(x_t,a_t) = \mathbb{E}_{\mathbb{P}^\pi_{t:t+k}}[\sum_{m=0}^{k-1}\gamma^m r(x_{t+m},a_{t+m}) \\ +\gamma^k V^\mu_{t+k}(x_{t+k})|x_t,a_t] - V^\mu_t(x_t).\end{aligned} \quad (2)$$

These advantages estimate the effect of starting in state x_t at time t, taking action a_t, then rolling-in for $(k-1)$-steps with π before reverting to μ till the end of the episode (remember that episode length is T). Furthermore the baseline that is subtracted from this Q-function is the estimated value of following μ from state x_t. Note that $\mathbb{A}^\pi_{(1),t} \equiv A^\mu_t$ and if $k \geq T-t : \mathbb{A}^\pi_{(k),t} \equiv A^\pi_t$. So using $k=1$ recovers the advantages used by policy gradient schemes, while $k=T$ recovers advantages in conservative policy iteration schemes. Using V^π in place of V^μ in Equation 2 will reduce SHPI to the classical n-step temporal difference estimator. However, as shown in Fig. 2b, this substitution leads to noisier advantage estimates. Further, we recover session-based episodic RL algorithms if V^μ always returns 0, and if additionally $k=1$ we recover contextual bandit algorithms. In practice we set k to be large enough to span user sessions, e.g., we set $k=3$ in the example of Sec. 1 and we use equation 1 to get V^μ.

We can use any episodic RL algorithm with these k-step advantages or termination bonuses. In Appendix 6.5 we instantiate a classic approximate policy iteration algorithm in Alg. 2. However practical deployments of RL for recommendation have limited budget for interaction and must use the logged data $\mathcal{D}$ more effectively. We now sketch our main algorithm for offline LTR optimization.

4.3 Offline Short-Horizon Policy Iteration

In the offline problem setting we must recover a policy π that improves over μ using only the dataset $\mathcal{D}$. This can be impossible in general if μ is not sufficiently exploratory and the collected dataset $\mathcal{D}$ is not sufficiently large. We make the following standard off-policy learning assumptions: (1) the logging policy μ has non-negligible probability of sampling all actions (we will additionally clip importance weights in Eqn 3 to limit variance from sampled low-probability actions) (2) $\mathcal{D}$ is large enough for non-degenerate importance sampling estimates.

Since the training dataset was collected by μ and not π, we use PDIS [18] to rewrite the expectation in terms of μ:

$$\begin{aligned}\mathbb{A}^\pi_{(k),t}(x_t,a_t) = \mathbb{E}_{\mathbb{P}^\mu_{t:t+k}}[\sum_{m=0}^{k-1} w^{t+m}_{t+1}\gamma^m r(x_{t+m},a_{t+m}) \\ +w^{t+k}_{t+1}\gamma^k V^\mu_{t+k}(x_{t+k})|x_t,a_t] - V^\mu_t(x_t),\end{aligned} \quad (3)$$

Algorithm 1: Offline SHPI

```
Input : Batch $\mathcal{D}$, horizon $k$, model class $\mathcal{F}$, iterations $J$, clipping
        coefficients $q_1, q_2$
$\hat{V}^\mu \leftarrow$ Eq.(1);
/* Randomly initialize $\pi^{(0)} \neq \mu$                    */
$\pi^{(0)} \sim \Delta(\mathcal{A})$ ;
for $j = 1, .., J$ do
    for $\tau = \{(x_t, a_t, r_t)\}_{t=1}^T \sim \mathcal{D}$ do
        $\hat{\mathbf{w}} \leftarrow \texttt{clip}(\{\prod_{m=t+1}^{t+k} \frac{\pi^{(j-1)}(a_m|x_m)}{\mu(a_m|x_m)}\}_{t=1}^T, q_1, q_2)$
    end
    $\hat{\mathbb{A}}^\pi_{(k)} \leftarrow$ Eq.(4) ;
    $\hat{f} \leftarrow \arg\min_{f \in \mathcal{F}} \sum_{(x,a)\in\mathcal{D}} \{f(x,a) - \hat{\mathbb{A}}^\pi_{(k)}(x,a)\}^2$ ;
    $\pi^{(j)}(x) \leftarrow \arg\max_{a\in\mathcal{A}} \hat{f}(x,a)$;
end
```

where $w_{t_1}^{t_2} = \prod_{m=t_1}^{t_2} \frac{\pi(a_m|x_m)}{\mu(a_m|x_m)}$ when $t_1 \leq t_2$, 1 otherwise. To mitigate potentially large values of $w_{t_1}^{t_2}$ we use weight clipping [8]. While q_1, q_2 can be tuned independently for each domain, we set $q_1 = 0.5$ and $q_2 = 2$ in all experiments. The finite sample version of our estimator from n Monte Carlo trajectories is

$$\begin{aligned}\hat{\mathbb{A}}^\pi_{(k),t} := \frac{1}{n}\sum_{i=1}^{n}[\sum_{m=0}^{k-1} \hat{w}_{i,t+1}^{t+m}\gamma^m \hat{r}(x_{i,t+m}, a_{i,t+m}) \\ + \hat{w}_{i,t+1}^{t+k}\gamma^k \hat{V}^\mu(x_{i,t+k})] - \hat{V}^\mu(x_{i,t}) \ .\end{aligned} \tag{4}$$

As in the online version of SHPI (Sec 4.2), we set k to be large enough to span user sessions, e.g., we set $k = 3$ in the example of Sec. 1 and we use equation 1 to get V^μ. The approximate policy iteration algorithm called Offline SHPI is sketched in Alg. 1. We first estimate V^μ, and initialize our policy away from μ. Then we interleave advantage estimation using Eq. 4 with policy improvement steps. Note how the policy improvement step is rewarded implicitly with the termination bonus of $V^\mu(x_{k+1})$ inside its advantages.

We answer two questions about SHPI: (i) Using perfect estimates of $\hat{w}, \hat{r}, \hat{V}^\mu$, does SHPI guarantee a policy improvement (is the algorithm *consistent*?) and (ii) if some state-action pairs are poorly covered by μ, is it possible to find an improved policy (is the algorithm *efficient*)? In Sec. 4.4 we show that SHPI is both consistent and efficient.

4.4 Analysis

Since SHPI is essentially an approximate policy iteration scheme with a modified advantage function, it inherits the convergence issues of API with function approximation. In particular, under the assumptions that the true $\mathbb{A}^{\pi}_{(k)}$ lies in $\mathcal{F}$ (realizability) and that the minimizer f^* incurs low error over $\mathcal{D}$ (completeness), then Alg. 1 will converge to a fixed point $\tilde{\pi}$. In practice, we use a maximum iteration number J and return $\tilde{\pi} = \pi^{(J)}$. Note also that we initialize Alg. 1 away from μ so that A^{μ} and $\mathbb{A}^{\pi^0}_{(k)}$ do not coincide.

We now describe sufficient conditions for SHPI to work well, in increasing order of their generality. These conditions build on a long line of work tracing back to Blackwell optimality in the known MDP setting [7]. The first condition suggests that SHPI returns a near-optimal policy in a special class of MDPs, regardless of the V^{μ} we use. The second condition builds on recent work [3] to show that if $\hat{V}^{\mu}$ has favorable properties then SHPI returns a near-optimal policy. Finally, the third condition suggests that by tuning k we might achieve a better bias-variance trade-off and SHPI can return a larger policy improvement over μ (but not necessarily optimal) than existing methods.

Condition 1 (most restrictive) If the MDP$(\mathcal{T}, R, p_0, T, \gamma)$ admits a Blackwell-optimal policy for some unknown $k^* < T$ (equivalently for a $\gamma^* < \gamma$), then there exists a $k < T$ such that SHPI is near-optimal.

This condition holds when user session initiations are independent of the recommendation policy or if the task/reward horizon T is long but the environment is reset to p_0 after $H < T$. Since the definition of Blackwell optimality says that a policy that is optimal for γ^* remains optimal for $\gamma \geq \gamma^*$, any episodic RL algorithm can find a near-optimal solution if $k \geq \frac{1-\gamma}{1-\gamma^*}T$. Both session-based RL as well as SHPI will thus be consistent for large enough k.

Condition 2 (less restrictive) From Corollary 4.1 of [3] if $\hat{V}^{\mu}$ has a small error $\|\hat{V}^{\mu} - V^*\|_{\infty} \leq \epsilon$ then the policy returned by SHPI is near optimal, $V^* - V(\pi_{SHPI}) \leq \epsilon\frac{(1-\lambda\gamma)^2}{(1-\gamma)^2}$ where $\lambda = \frac{(k-1)T}{(T-1)k}$. SHPI can thus be better than session-based RL when $\hat{V}^{\mu}$ has a smaller l_{∞} error than a constant function. This is true, for instance when μ is on the value improvement path towards π^* or if $\hat{V}^{\mu}$ is an *improvable* bonus function (see [3]).

Condition 3 (least restrictive) For all π, there exists $1 \leq k \leq T$ such that the mean squared error of estimating $\mathbb{A}^{\pi}_{(k)}$ is lower than estimating $\mathbb{A}^{\mu}$ and $\mathbb{A}^{\pi}$. Therefore by tuning k, we can have a good bias-variance trade-off so that SHPI finds the tightest lower bound on policy improvement. Experiments in Sec. 5.4 confirm that k set to session lengths can give better estimates and better policy improvements than policy gradient and conservative policy iteration.

5 Experiments

We conducted experiments on three benchmarks and a private dataset: **(i)** a synthetic recommendation problem with a complex reward function [23], **(ii)** the RecoGym environment with a sub-modular reward function [20], and **(iii)** the HIV treatment simulator proposed in [4]. We consider only the more practical offline problem setting (i.e., an algorithm cannot interact with its environment to collect additional training data; all algorithms are still evaluated via online deployment in the environment) and report results for Offline SHPI (Algorithm 1) in all experiments[8].

5.1 Experimental setup

On all four domains, we first pre-train a policy (deep SARSA) using a large number of online interactions. It is then mixed with random uniform actions with probability $\varepsilon \in [0, 1]$, in order to simulate various levels of performance; we call this policy μ, or logging policy. This agent is then deployed in the environment to gather a fixed dataset $\mathcal{D}$ which are used to train all compared methods. Finally, all experiments report undiscounted test performance of learned policies on true environment rewards over 200 rollouts and 5 random seeds.

Test domains The first domain implements a toy recommendation scenario in which each of the action vectors, with randomly sampled coordinates, affects the context X_t of a user in an additive way. The policy's input context is a moving average of previous contexts; the reward function r depends only on the accumulated X_t and is characterized by the Styblinski-Tang function (see Appendix for more details).

The second domain, RecoGym, is a personalization benchmark to study click-based recommender systems using RL. We optimize a long-term reward signal which is a submodular function that depends on all previously clicked items.

The third domain is the HIV treatment simulator which relies on differential equations to simulate the administration of treatment to a group of patients. While it has relatively small state and action spaces, the HIV simulator is considered a proxy for real-world off-policy recommendation in the healthcare domain [14].

Our fourth and last domain consists of a private dataset X. The dataset X contains 120,000 unique user interactions collected from a live recommender system over the span of 3 weeks for 10,000 users. The recommendation setting is realistic, with $|\mathcal{A}| = 112$ and $\mathcal{X} \subseteq \mathbb{R}^{47}$. The dataset is accompanied by a long-term value estimator $\hat{V}^{\mu} \in [-1.5, 1]$, (where 1 is the best possible score). The LTR corresponds to the likelihood of a user to re-subscribe to a service.

Each domain stresses the basic assumptions in SHPI : the toy scenario exactly fits our assumptions, RecoGym is reproducible and semi-synthetic with sessions that still conform to SHPI assumptions, the private dataset tests scalability and our assumptions may be invalid, while the HIV simulator violates the session-structure entirely.

Choice of baselines For all domains, we include a standard contextual bandit agent without exploration. Next, we include BCQ [5] as a batch RL baseline and CQL [10] as a state-of-the-art batch RL approach. BCQ is general enough to be

[8] Code to reproduce results can be found at https://github.com/bmazoure/shpi.

suitable for all target domains without extensive tuning, while CQL performs well with near-optimal datasets. Since RecoGym gives access to both short- and long-term signals, we also compared with three common recommender baselines that can learn from myopic signals in RecoGym: (1) bandit with matrix factorization (*Bandit+MF*)[9], (2) IPS estimator via neural net regression (*NN-IPS*)[10] and (3) RepeatNet [19], a strong recurrent neural network approach.

5.2 Performance Trends across Domains

In Table 1 we compare SHPI performance to that of bandit and offline RL methods on a large-scale version of the toy domain with $|\mathcal{S}| = 100$ and $|\mathcal{A}| = 300$. Table 2 shows returns collected by all algorithms in the HIV domain.

Table 1. Evaluation of RL methods on the large toy domain.

ε	Logging	Bandit	SHPI (Ours)	BCQ	CQL
0.3	1186 ± 1535	1648 ± 732	7235 ± 325	495 ± 831	1231 ± 527
0	2191 ± 180	-509 ± 269	2151 ± 1207	3847 ± 979	3974 ± 341

Note that SHPI achieves better results when learning from suboptimal data, as opposed to data collected from a near-optimal policy. This highlights an instability when learning from an expert policy; can SHPI be composed with complementary off-policy RL techniques for stable extrapolation? In the Appendix, we show experimental results for SHPI with trust region proximal regularization [21], which indeed leads to stabler learning from optimal data.

Table 2. Evaluation of RL methods on the HIV simulator.

ε	Logging	Bandit	SHPI (Ours)	BCQ	CQL
0.3	0.69 ± 0.02	0.34 ± 0.11	0.85 ± 0.17	0.32 ± 0.15	0.37 ± 0.14
0	0.73 ± 0.01	0.57 ± 0.10	0.50 ± 0.11	0.39 ± 0.13	0.51 ± 0.11

Additionally, we tested SHPI on the private dataset X. We first fitted a model to state transitions and rewards in the dataset. This model is more realistic than RecoGym: it incorporates a complex LTR function (with values in $[-1.5, 1]$), richer user features and realistic state transitions based on real user interactions. We ran all algorithms on a batch simulated from this environment by an ε-greedy policy and report online evaluation results of the resulting policies (Table 3).

[9] see github.com/criteo-research/reco-gym/blob/master/recogym/agents/bandit_mf.py

[10] see github.com/criteo-research/reco-gym/blob/master/recogym/agents/nn_ips.py

Table 3. Evaluation of RL methods on the private dataset X.

Rewards	Logging	Bandit	SHPI (Ours)	BCQ	CQL
Raw	-0.69 ± 0.3	-1.50 ± 0.03	-0.29 ± 0.21	-1.02 ± 0.29	-0.64 ± 0.19
Rescaled	1	0.45	2.37	0.68	1.08

The results above suggest that SHPI scales to real-world datasets, and performs better than bandit and offline RL methods on realistic datasets in many recommendation scenarios.

5.3 Optimizing LTR vs Myopic Proxy Signals

Throughout the paper, we focused on maximizing long-term metrics like user conversions which are very sparse signals of success. There is a popular alternative approach which uses denser, noisy but correlated signals instead of the sparse long-term metrics, akin to reward-shaping in RL [24]. In this experiment, we compare the performance of policies optimizing LTR with that of policies optimizing correlated immediate rewards (we use clicks) within the RecoGym simulator. Table 4 shows resultswhere all algorithms are trained on their respective reward (click or long-term rewards) but evaluated on the LTR metric.

Table 4. Evaluation on the long-term signal in RecoGym after training on short- vs long-term signals.

	Online RL	Offline RL				Recommendation baselines		
Signal	Logging	Bandit	SHPI (Ours)	BCQ	CQL	Bandit+MF	NN IPS	RepeatNet
Click	0.05 ± 0.25	0.11 ± 0.16	0.02 ± 0.23	-0.03 ± 0.05	0.13 ± 0.03	-0.20 ± 0.21	0.11 ± 0.31	-0.02 ± 0.08
LTR	-0.13 ± 0.28	-0.02 ± 0.18	0.33 ± 0.17	0.08 ± 0.14	0.123 ±0.1	0.06 ± 0.19	0.25 ± 0.15	0.221 ± 0.12

The Bandit+MF and NN-IPS baselines are trained to greedily maximize the signal made available to them by the simulator, but without any reasoning beyond the current timestep. Under click rewards, we see that all algorithms are taking myopic decisions to maximize immediate rewards and do not optimize the long-term submodular metric. This suggests that the common practice of optimizing noisy, correlated, myopic rewards can be very sub-optimal and, when possible, we recommend directly collecting long-term metrics and optimizing them using SHPI.

5.4 Influence of k on SHPI

We further conduct ablations on the role that k plays in the estimation error of the long-term advantages. To do so, we sample 200 states from $\mathbb{P}^{\mu}$ in the toy task and perform 30 online rollouts to estimate ground truth advantages for each s using Eqn. 4 ($\mathbb{A}^\mu$ with $k = 1$; $\mathbb{A}^\pi$ with $k = 200$; and $\mathbb{A}_k$ with $k = 5$). We report the mean squared error between offline and online estimates in Figure 2 (right), as well as respective returns of the resulting SHPI policy in Figure 2 (left). The results show that using $k = 5$ yields lower MSE than $\mathbb{A}^\mu$ or $\mathbb{A}^\pi$, although the benefit is less significant as $\mathcal{D}$ is collected from a more deterministic μ. The improved estimation also leads to better-performing policies on the toy task.

ε	0.3	0
$\hat{\mathbb{A}}_k$	8411 ± 32	8346 ± 180
$\hat{\mathbb{A}}^\mu$	8097 ± 34	8285 ± 269
$\hat{\mathbb{A}}^\pi$	1499 ± 8005	4761 ± 6186

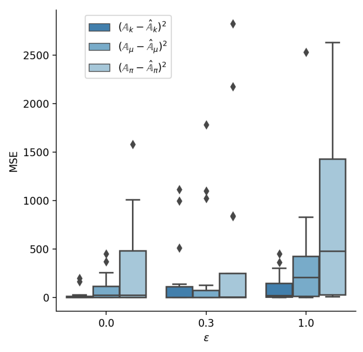

Fig. 2. Mean-squared error (right) and returns (left) obtained by SHPI using estimates of $\mathbb{A}^\pi_{(k)}$, $\mathbb{A}^\mu$, $\mathbb{A}^\pi$, averaged across states and for different logging policies.

6 Discussion

A suitable termination bonus estimator can be feasibly estimated from batch data for session-based recommendation, and when used along with episodic RL techniques we witness a policy improvement that other RL methods can miss. SHPI uses these bonuses to provide batch RL policy improvement. Empirical results show that SHPI is particularly effective in improving long-term rewards.

There are several avenues for further research. Can we tune the horizon k in a problem-dependent way? This may allow us to *discover* when session-based or even myopic bandit-based reasoning is sufficient for an application. The k-step advantages used by SHPI can also be incorporated in other batch RL algorithms (which predominantly use policy gradient or conservative policy iteration schemes). Finally, deploying SHPI in real-world recommender systems, and studying its performance when user preferences may be non-stationary, remains as future work.

References

1. Afsar, M.M., Crump, T., Far, B.: Reinforcement learning based recommender systems: A survey. arXiv preprint arXiv:2101.06286 (2021)

2. Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., Chi, E.H.: Top-k off-policy correction for a REINFORCE recommender system. In: ACM International Conference on Web Search and Data Mining. pp. 456–464 (2019)
3. Cheng, C., Kolobov, A., Swaminathan, A.: Heuristic-guided reinforcement learning. Advances in Neural Information Processing Systems **34**, 13550–13563 (2021)
4. Ernst, D., Stan, G.B., Goncalves, J., Wehenkel, L.: Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In: IEEE Conference on Decision and Control. pp. 667–672 (2006)
5. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: International Conference on Machine Learning. pp. 2052–2062 (2019)
6. Gruson, A., Chandar, P., Charbuillet, C., McInerney, J., Hansen, S., Tardieu, D., Carterette, B.: Offline evaluation to make decisions about playlist recommendation algorithms. In: ACM International Conference on Web Search and Data Mining. pp. 420–428 (2019)
7. Hordijk, A., Yushkevich, A.A.: Blackwell optimality. In: Handbook of Markov decision processes, pp. 231–267. Springer (2002)
8. Ionides, E.L.: Truncated importance sampling. Journal of Computational and Graphical Statistics **17**, 295–311 (2008)
9. Kakade, S., Langford, J.: Approximately optimal approximate reinforcement learning. In: International Conference on Machine Learning. pp. 267–274 (2002)
10. Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems **33**, 1179–1191 (2020)
11. Laroche, R., Trichelair, P., Des Combes, R.T.: Safe policy improvement with baseline bootstrapping. In: International Conference on Machine Learning. pp. 3652–3661 (2019)
12. Li, A.C., Florensa, C., Clavera, I., Abbeel, P.: Sub-policy adaptation for hierarchical reinforcement learning. International Conference on Learning Representations (2019)
13. Liao, P., Greenewald, K., Klasnja, P., Murphy, S.: Personalized heartsteps: A reinforcement learning algorithm for optimizing physical activity. ACM Interactive, Mobile, Wearable and Ubiquitous Technologies **4**, 1–22 (2020)
14. Liu, Y., Gottesman, O., Raghu, A., Komorowski, M., Faisal, A.A., Doshi-Velez, F., Brunskill, E.: Representation balancing MDPs for off-policy policy evaluation. Advances in Neural Information Processing Systems **31**, 2644–2653 (2018)
15. Liu, Y., Swaminathan, A., Agarwal, A., Brunskill, E.: Provably good batch off-policy reinforcement learning without great exploration. Advances in Neural Information Processing Systems **33**, 1264–1274 (2020)
16. Ma, Y., Narayanaswamy, B., Lin, H., Ding, H.: Temporal-contextual recommendation in real-time. In: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2291–2299 (2020)
17. Nachum, O., Chow, Y., Dai, B., Li, L.: Dualdice: behavior-agnostic estimation of discounted stationary distribution corrections. Advances in Neural Information Processing Systems pp. 2318–2328 (2019)
18. Precup, D., Sutton, R.S., Singh, S.P.: Eligibility traces for off-policy policy evaluation. In: International Conference on Machine Learning. pp. 759–766 (2000)
19. Ren, P., Chen, Z., Li, J., Ren, Z., Ma, J., De Rijke, M.: Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In: AAAI Conference on Artificial Intelligence. pp. 4806–4813 (2019)

20. Rohde, D., Bonner, S., Dunlop, T., Vasile, F., Karatzoglou, A.: Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. arXiv preprint arXiv:1808.00720 (2018)
21. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning. pp. 1889–1897 (2015)
22. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 (2015)
23. Styblinski, M., Tang, T.S.: Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. Neural Networks **3**, 467–483 (1990)
24. Vernade, C., Gyorgy, A., Mann, T.: Non-stationary delayed bandits with intermediate observations. In: International Conference on Machine Learning. pp. 9722–9732 (2020)
25. Wu, Q., Wang, H., Hong, L., Shi, Y.: Returning is believing: Optimizing long-term user engagement in recommender systems. In: ACM Conference on Information and Knowledge Management. pp. 1927–1936 (2017)
26. Xie, R., Zhang, S., Wang, R., Xia, F., Lin, L.: Hierarchical reinforcement learning for integrated recommendation. In: AAAI Conference on Artificial Intelligence. vol. 35, pp. 4521–4528 (2021)
27. Yin, M., Bai, Y., Wang, Y.X.: Near-optimal offline reinforcement learning via double variance reduction. Advances in Neural Information Processing Systems **34**, 7677–7688 (2021)