A Benchmark for Low-Switching-Cost Reinforcement Learning

Shusheng Xu[1], **Yancheng Liang**[1], **Yunfei Li**[1], **Simon Shaolei Du**[2], **Yi Wu**[1,3]

[1] IIIS, Tsinghua Unveristy, Beijing,
[2] University of Washington,
[3] Shanghai Qi Zhi institute, Shanghai

{xuss20, liangyc19, liyf20}@mails.tsinghua.edu.cn,
ssdu@cs.washington.edu,
jxwuyi@gmail.com

Abstract

A ubiquitous requirement in many practical reinforcement learning (RL) applications, including medical treatment, recommendation system, education and robotics, is that the deployed policy that actually interacts with the environment cannot change frequently. Such an RL setting is called low-switching-cost RL, i.e., achieving the highest reward while reducing the number of policy switches during training. Despite the recent trend of theoretical studies aiming to design provably efficient RL algorithms with low switching costs, none of the existing approaches have been thoroughly evaluated in popular RL testbeds. In this paper, we systematically studied a wide collection of policy-switching approaches, including theoretically guided criteria, policy-difference-based methods, and non-adaptive baselines. Through extensive experiments on a medical treatment environment, the Atari games, and robotic control tasks, we present the first empirical benchmark for low-switching-cost RL and report novel findings on how to decrease the switching cost while maintain a similar sample efficiency to the case without the low-switching-cost constraint. We hope this benchmark could serve as a starting point for developing more practically effective low-switching-cost RL algorithms. We release our code and complete results in *https://sites.google.com/view/low-switching-cost-rl*.

1 Introduction

Reinforcement Learning (RL) has been successfully applied to solve sequential-decision problems in many real-world scenarios, such as medical domains [15], robotics [7, 11], hardware placements [19, 18], and personalized recommendation [27]. In these scenarios, it is often desirable to restrict the agent from adjusting its policy too often due to the high costs and risks of deployment. For example, changing a policy in medical domains requires a thorough approval process by human experts [2]; changing policies on robots can be associated with additional risks [7]. In these settings, it is a requirement that the deployed policy, i.e., the policy used to interact with the environment, could keep unchanged as much as possible. Formally, we would like our RL algorithm to both produce a policy with the highest reward and at the same time reduce the number of deployed policy switches (i.e., a low *switching cost*) throughout the training process.

Offline reinforcement learning [14] is perhaps the most related framework in the existing literature that also has a capability of avoiding frequent policy deployment. Offline RL assumes a given transition dataset and performs RL training completely in an offline fashion without interacting with the environment at all. [17] adopt a slightly weaker offline assumption by repeating the offline

Submitted to the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks. Do not distribute.

training procedure, i.e., re-collecting transition data using the current policy and re-applying offline RL training on the collected data, for about 10 times. However, similar to the standard offline RL methods, due to such an extreme low-deployment-constraint, the proposed method suffers from a particularly low sample efficiency and even produces significantly lower rewards than online SAC method in many cases [17]. In contrast with offline RL, which optimizes the reward subject to a minimal switching cost, low-switching-cost RL aims to *reduce the switching cost while maintain a similar sample efficiency and final reward* compared to its unconstrained RL counterpart.

In low-switching-cost RL, the central question is *how to design a criterion to decide when to update the deployed policy* based on the current training process. Ideally, we would like this criterion to satisfy the following four properties:

1. **Low switching cost:** This is the fundamental mission. An RL algorithm equipped with this policy switching criterion should have a reduced frequency to update the deployed policy.
2. **High reward:** Since the deployed policy can be different from the training one, the collected data can be highly off-policy. We need this criterion to deploy policies at the right timing so that the collected samples can be still sufficient for finally achieving the optimal reward.
3. **Sample efficiency:** In addition to the final reward, we also would like the algorithm equipped with such a criterion to produce a similar sample efficiency to the unconstrained RL setting without the low-switching-cost requirement.
4. **Generality:** We would like this criterion can be easily applied to a wide range of domains rather than a specific task.

From the theoretical side, low-switching-cost RL and its simplified bandit setting have been extensively studied [3, 5, 4, 21, 6, 25, 26]. The core notion in these theoretical methods is *information gain*. Specifically, they update the deployed policy only if the measurement of information gain is doubled, which also leads to optimality bounds for the final policy rewards. We suggest the readers refer to the original papers for details of the theoretical results. We will also present algorithmic details later in Section 4.4.

However, to our knowledge, there has been no empirical study on whether these theoretically-guided criteria are in fact effective in popular RL testbeds. In this paper, we aim to provide systematic benchmark studies on different policy switching criteria from an empirical point of view. Our contributions are summarized below.

Our Contributions

- We conduct the first empirical study for low-switching-cost RL on environments that require modern RL algorithms, i.e., Rainbow [9] and SAC [8], including a medical environment, 56 Atari games[1] and 6 MuJoCo control tasks. We test theoretically guided policy switching criteria based on the information gain as well as other adaptive and non-adaptive criteria.
- We find that a feature-based criterion produces the best overall quantitative performance. Surprisingly, the non-adaptive switching criterion serves as a particularly strong baseline in all the scenarios and largely outperforms the theoretically guided ones.
- Through extensive experiments, we summarize practical suggestions for RL algorithms with with low switching cost, which will be beneficial for practitioners and future research.

2 Related Work

Low switching cost algorithms were first studied in the bandit setting [3, 5]. Existing work on RL with low switching cost is mostly theoretical. To our knowledge, [4] is the first work that studies this problem for the episodic finite-horizon tabular RL setting. [4] gave a low-regret algorithm with an $O\left(H^3SA\log\left(K\right)\right)$ local switching upper bound where S is the number of stats, A is the number of actions, H is the planning horizon and K is the number of episodes the agent plays. The upper bound was improved in [26, 25].

[1]There are a total of 57 Atari games. However, only 56 of them (excluding the "surround" game) are supported by the atari-py package, which we adopt as our RL training interface.

Offline RL (also called Batch RL) can be viewed as a close but parallel variant of low-switching-cost RL, where the policy does not interact with the environment at all and therefore does not incur any switching cost. Offline RL methods typically learn from a given dataset [13, 14], and have been applied to many domains including education [16], dialogue systems [10] and robotics control [12]. Some methods interpolate offline and online methods, i.e., semi-batch RL algorithms [22, 13], which update the policy many times on a large batch of transitions. However, reducing the switching cost during training is not their focus. [17] developed the only empirical RL method that tries to reduce the switching cost without the need of a given offline dataset. Given a fixed number of policy deployments (i.e., 10), the proposed algorithm collects transition data using a fixed deployed policy, trains an ensemble of transition models and updated a new deployed policy via model-based RL for the next deployment iteration. However, even though the proposed model-based RL method in [17] outperforms a collection of offline RL baselines, the final rewards are still substantially lower than standard online SAC even after consuming an order of magnitude more training samples. In our work, we focus on the effectiveness of the policy switching criterion such that the overall sample efficiency and final performances can be both preserved.

3 Preliminaries

Markov Decision Process: We consider the Markov decision model $(\mathcal{S}, \mathcal{A}, \gamma, r, p_0, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, γ is the discounted factor, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, p_0 is the initial state distribution, and $P(x'|x, a)$ denotes the transition probability from state x to state x' after taking action a. A policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a mapping from a state to an action, which can be either deterministic or stochastic. An episode starts with an initial state $x_0 \sim p_0$. At each step h in this episode, the agent chooses an action a_h from $\pi(x_h)$ based on the current state x_h, receives a reward $r(x_h, a_h)$ and moves to the next state $x_{h+1} \sim P(\cdot|x_h, a_h)$. We assume an episode will always terminate, so each episode $e = \{(x_h^e, a_h^e)|0 \le h \le H_e\}$ will always have a finite horizon H_e (e.g., most practical RL environments have a maximum episode length $H_{\max}$). The goal of the agent is to find a policy π^* which maximizes the discounted expected reward, $\pi^\star = \arg\max_\pi \mathbb{E}_e \left[\sum_{h=0}^{H_e} \gamma^h r(x_h^e, a_h^e)\right]$. Let K denote the total transitions that the agent experienced across all the episodes during training. Ideally, we also want the agent to consume as few training samples as possible, i.e., a minimal K, to learn $\pi^\star$. A Q-function is used to evaluate the long-term value for the action a and subsequent decisions, which can be defined w.r.t. a policy π by

$$Q^\pi(x, a) := r(x, a) + \mathbb{E}\left[\sum_h \gamma^h r\left(x_h, \pi\left(x_h\right)\right) \middle| x_0 = x, a_0 = a\right]. \tag{1}$$

Deep Off-policy Reinforcement Learning: Deep Q-learning (DQN) [20] is perhaps the most popular off-policy RL algorithm leveraging a deep neural network to approximate $Q(x, a)$. Given the current state x_h, the agent selects an action a_h greedily based on parameterized Q-function $Q_\theta(x_h, a)$ and maintain all the transition data in the replay buffer.For each training step, the temporal difference error is minimized over a batch of transitions sampled from this buffer by

$$\mathcal{L}(\theta) = \mathbb{E}\left[(r_{h+1} + \gamma \max_{a'} Q_{\bar{\theta}}(x_{h+1}, a') - Q_\theta(x_h, a_h))^2\right], \tag{2}$$

where $\bar{\theta}$ represents the parameters of the target Q-network, which is periodically updated from θ. Rainbow [9] is perhaps the most famous DQN variant, which combines six algorithmic enhancements and achieves strong and stable performances on most Atari games. In this paper, we adopt a deterministic version[2] of Rainbow DQN as the RL algorithm for the discrete action domains. We also adopt count-based exploration [23] as a deterministic exploration bonus.

For continuous action domains, soft actor-critic (SAC) [8] is the representative off-policy RL algorithm. SAC uses neural networks parameterized by θ to approximate both $Q(s, a)$ and the stochastic policy $\pi_\theta(a|s)$. Q-network is trained to approximate entropy-regularized expected return by minimizing

$$\mathcal{L}_Q(\theta) = \mathbb{E}\left[(r_h + \gamma(Q_{\bar{\theta}}(x_{h+1}, a') - \alpha \log \pi(a'|x_{h+1})) - Q_\theta(x_h, a_h))^2 | a' \sim \pi(\cdot|x_{h+1})\right], \tag{3}$$

[2] Standard Rainbow adds random noise to network parameters for exploration, which can be viewed as constantly switching policies over a random network ensemble. This contradicts the low-switching-cost constraint.

Algorithm 1 General Workflow of Low-Switching-Cost RL

Initialize parameters $\theta_{\text{onl}}, \theta_{\text{dep}}$, an empty replay buffer D, $C_{\text{switch}} \leftarrow 0$
for k $\leftarrow 0$ to $K-1$ **do**
  Select a_k by $\pi_{\text{dep}}(x_k)$, execute action a_k and observe reward r_k, state x_{k+1}
  Store $(x_k$, a_k, r_k, $x_{k+1})$ in D
  Update θ_{onl} using D and an off-policy RL algorithm
  if $\mathcal{J}(\star) == \text{true}$ **then**
    Update $\theta_{\text{dep}} \leftarrow \theta_{\text{onl}},\ C_{\text{switch}} \leftarrow C_{\text{switch}} + 1$
  end if
end for

where α is the entropy coefficient. We omit the parameterization of π since π is not updated w.r.t $\mathcal{L}_Q$. The policy network π_θ is trained to maximize $\mathcal{L}_\pi(\theta) = \mathbb{E}_{a\sim\pi}\left[Q(x,a) - \alpha \log \pi_\theta(a|x)\right]$.

4 Reinforcement Learning with Low Switching Cost

In standard RL, a transition (x_h, a_h, x_h) is always collected by a single policy π. Therefore, whenever the policy is updated, a switching cost is incurred. In low-switching-cost RL, we have two separate policies, a deployed policy π_{dep} that interacts with the environment, and an online policy π_{onl} that is trained by the underlying RL algorithm. These policies are parameterized by θ_{dep} and θ_{onl} respectively. Suppose that we totally collect K samples during the training process, then at each transition step k, the agent is interacting with the environment using a deployed policy π_{dep}^k. After the transition is collected, the agent can decide whether to update the deployed π_{dep}^{k+1} by the online policy π_{onl}^{k+1}, i.e., replacing θ_{dep} with θ_{onl}, according to some switching criterion $\mathcal{J}$. Accordingly, the switching cost is defined by the number of different deployed policies throughout the training process, namely:

$$C_{\text{switch}} := \sum_{k=1}^{K-1} \mathbb{I}\{\pi_{\text{dep}}^{k-1} \neq \pi_{\text{dep}}^{k}\} \tag{4}$$

The goal of low-switching-cost RL is to design an effective algorithm that learns π^* using K samples while produces the smallest switching cost C_{switch}. Particularly in this paper, we focus on the design of the switching criterion $\mathcal{J}$, which is the most critical component that balances the final reward and the switching cost. The overall workflow of low-switching-cost RL is shown in Algorithm 1.

In the following content, we present a collection of policy switching criteria and techniques, including those inspired by the information gain principle (Sec. 4.4) as well as other non-adaptive (Sec. 4.1) and adaptive criteria (Sec. 4.2,4.3). All the discussed criteria are summarized in Algorithm 2.

4.1 Non-adaptive Switching Criterion

This simplest strategy switches the deployed policy every n timesteps, which we denote as "FIX_n" in our experiments. Empirically, we notice that "FIX_1000" is a surprisingly effective criteria which remains effective in most of the scenarios without fine tuning. So we primarily focus on "FIX_1000" as our non-adaptive baseline. In addition, We specifically use "None" to indicate the experiments without the low-switching-cost constraint where the deployed policy keeps synced with the online policy all the time. Note that this "None" setting is equivalent to "FIX_1".

4.2 Policy-based Switching Criterion

Another straightforward criterion is to switch the deployed policy when the action distribution produced by the online policy significantly deviates from the deployed policy. Specifically, we sample a batch of training states and count the number of states where actions by the two policies differ in the discrete action domains. We switch the policy when the ratio of mismatched actions exceeds a threshold σ_p. For continuous actions, we use KL-divergence to measure the policy differences.

4.3 Feature-based Switching Criterion

Beyond directly consider the difference of action distributions, another possible solution for measuring the divergence between two policies is through the feature representation extracted by the neural networks. Hence, we consider a feature-based switching criterion that decides to switch policies according to the feature similarity between different Q-networks. Similar to the policy-based criterion, when deciding whether to switch policy or not, we first sample a batch of states $\mathbb{B}$ from the experience replay buffer, and then extract the features of all states with both the deployed deep Q-network and the online deep Q-network. Particularly, we take the final hidden layer of the Q-network as the feature representation. For a state x, the extracted features are denoted as $f_{\text{dep}}(x)$ and $f_{\text{onl}}(x)$, respectively. The similarity score between f_{dep} and f_{onl} on state x is defined as

$$sim(x) = \frac{\langle f_{\text{dep}}(x), f_{\text{onl}}(x) \rangle}{||f_{\text{dep}}(x)|| \times ||f_{\text{onl}}(x)||}. \tag{5}$$

We then compute the averaged similarity score on the batch of states $\mathbb{B}$

$$sim(\mathbb{B}) = \frac{\sum_{x \in \mathbb{B}} sim(x)}{||\mathbb{B}||}. \tag{6}$$

With a hyper-parameter $\sigma_f \in [0, 1]$, the feature-based policy switching criterion changes the deployed policy whenever $sim(\mathbb{B}) \leq \sigma_f$.

Reset-Checking as a Practical Enhancement: Empirically, we also find an effective implementation enhancement, which produces lower switch costs and is more robust across different environments: we *only* check the feature similarity when an episode *resets* (i.e., a new episode starts) and additionally force deployment to handle extremely long episodes (e.g., in the "Pong" game, an episode may be trapped in loopy states and lead to an episode length of over 10000 steps).

Hyper-parameter Selection: For action-based and feature-based criteria, we uniformly sample a batch of size 512 from recent 10,000 transitions and compare the action differences or feature similarities between the deployed policy and the online policy on these sampled transitions. We also tried other sample size and sampling method, and there is no significant difference. For the switching threshold (i.e., the mismatch ratio σ_p in policy-based criterion and parameter σ_f in feature-based criterion), we perform a rough grid search and choose the highest possible threshold that still produces a comparable final policy reward.

4.4 Switching via Information Gain

Existing theoretical studies propose to switch the policy whenever the agent has gathered sufficient new information. Intuitively, if there is not much new information, then it is not necessary to switch the policy. To measure the sufficiency, they rely on the visitation count of each state-action pair or the determinant of the covariance matrix. We implement these two criteria as follows.

Visitation-based Switching: Following [4], we switch the policy when visitation count of any state-action pair reaches an exponent (specifically $2^i, i \in \mathbb{N}$ in our experiments). Such exponential scheme is theoretically justified with bounded switching cost in tabular cases. However, it is not feasible to maintain precise visitations for high-dimensional states, so we adopt a random projection function to map the states to discrete vectors by $\phi(x) = \text{sign}(A \cdot g(x))$, and then count the visitation to the hashed states as an approximation. A is a fixed matrix with i.i.d entries from a unit Gaussian distribution $\mathcal{N}(0, 1)$ and g is a flatten function which converts x to a 1-dimensional vector.

Information-matrix-based Switching: Another algorithmic choice for achieving infrequent policy switches is based on the property of the feature covariance matrix [21, 6], i.e., $\Lambda_h^e = \sum_{e: H_e \geq h} \psi(x_h^e, a_h^e)\psi(x_h^e, a_h^e)^T + \lambda I$, where e denotes a training episode, h means the h-th timestep within this episode, and ψ denotes a mapping from the state-action space to a feature space. For each episode timestep h, [1] switches the policy when the determinant of Λ_h^e doubles. However, we empirically observe that the approximate determinant computation can be particularly inaccurate for complex RL problems. Instead, we adopt an effective alternative, i.e., switch the policy when the least absolute eigenvalue doubles. In practice, we again adopt a random projection function to map the state to low-dimensional vectors, $\phi(x) = \text{sign}(A \cdot g(x))$, and concatenate them with actions to get $\psi(x, a) = [\phi(x), a]$.

Algorithm 2 Switching Criteria ($\mathcal{J}$ in Algorithm 1)

$\triangleright$ Non-adaptive Switching
input environment step counter k, fixed switching interval n
output $bool(k \bmod n == 0)$

$\triangleright$ Policy-based Switching
input deployed and online policy $\pi_{\text{dep}}, \pi_{\text{onl}}$, state batch $\mathbb{B}$, threshold σ_p
Compute the ratio of action difference or KL divergence for π_{dep} and π_{onl} on $\mathbb{B}$ as δ.
output $bool(\delta \geq \sigma_p)$

$\triangleright$ Feature-based switching
input Encoder of deployed and online policy $f_{\text{dep}}, f_{\text{onl}}$, state batch $\mathbb{B}$, threshold σ_f
Compute $sim(\mathbb{B})$ via Eq.(6)
output $bool(sim(\mathbb{B}) \leq \sigma_f)$

$\triangleright$ Visitation-based Switching
input the current visited times of state-action pair $n(\phi(x_k), a_k)$
output $bool(n(\phi(x_k), a_k) \in \{1, 2, 4, 8...\})$

$\triangleright$ Information-matrix-based Switching
input episode timestep h, current covariance matrix Λ_h^e, old $\Lambda_h^{\widetilde{e}}$ at previous switch time
Compute the least absolute eigenvalues v_h^e and $v_h^{\widetilde{e}}$
output $bool(v_h^e \geq 2 \times v_h^{\widetilde{e}})$

5 Experiments

In this section, we conduct experiments to evaluate different policy switching criteria on Rainbow DQN and SAC. For discrete action spaces, we study the Atari games and the GYMIC testbed for simulating sepsis treatment for ICU patients which requires low switching cost. For continuous control, we conduct experiments on the MuJoCo [24] locomotion tasks.

5.1 Environments

GYMIC GYMIC is an OpenAI gym environment for simulating sepsis treatment for ICU patients to an infection, where sepsis is caused by the body's response to an infection and could be life-threatening. GYMIC built an environment to simulate the MIMIC sepsis cohort, where MIMIC is an open patient EHR dataset from ICU patients. This environment generates a sparse reward, the reward is set to +15 if the patient recovers and -15 if the patient dies. This environment has 46 clinical features and a 5×5 action space.

Atari 2600 Atari 2600 games are widely employed to evaluate the performance of DQN-based agents [9]. We evaluate the efficiency of different switching criteria on a total of 56 Atari games.

MuJoCo control tasks We evaluate different switching criteria on 6 standard continuous control benchmarks in the MuJoCo physics simulator, including Swimmer, HalfCheetah, Ant, Walker2d, Hopper and Humanoid.

5.2 Evaluation Metric

For GYMIC and Atari games whose action space is discrete, we adopt Rainbow DQN to train the policy; for MuJoCo tasks with continuous action spaces, we employ SAC since it is more suitable for continuous action space. We evaluate the efficiency among different switching criteria in these environments. All of the experiments are repeated over 3 seeds. Implementation details and hyper-parameters are listed in Appendix B. All the code and the complete experiment results can be found at *https://sites.google.com/view/low-switching-cost-rl*.

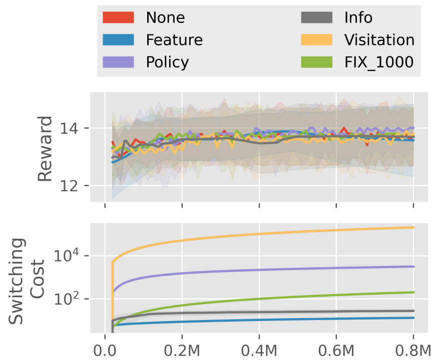

Figure 1: Results on GYMIC. *Top*: the learning curve of reward vs. steps. *Bottom*: switching cost. Note that the switching cost of "Visitation" almost overlaps with "None".

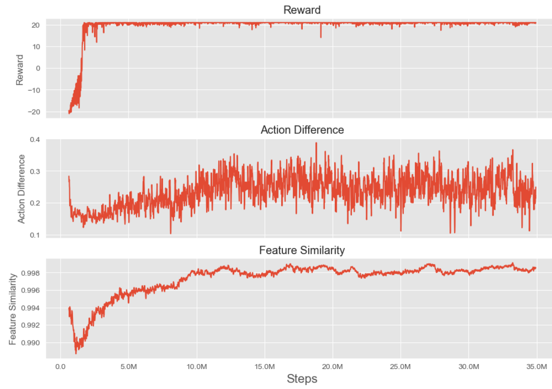

Figure 2: Action difference and feature similarity recorded on Pong. Higher feature similarity or lower action difference implies that the deployed policy and the online policy are closer.

We evaluate different policy switching criteria based on the off-policy RL backbone and measure the reward function as well as the switching cost in both GYMIC and MuJoCo control tasks. For Atari games, we plot the average human normalized rewards. Since there are 56 Atari games evaluated, we only report the average results across all the Atari games as well as 8 representative games in the main paper. Detailed results for every single Atari game can be found at our project website.

To better quantitatively measure the effectiveness of a policy switching criterion, we propose a new evaluation metric, ***Reward-threshold Switching Improvement (RSI)***, which takes both the policy performance and the switching cost improvement into account. Specifically, suppose the standard online RL algorithm (i.e., the "None" setting) can achieve an average reward of $\hat{R}$ with switching cost $\hat{C}$[3]. Now, an low-switching-cost RL criterion $\mathcal{J}$ leads to a reward of $R_{\mathcal{J}}$ and reduced switching cost of $C_{\mathcal{J}}$ using the same amount of training samples. Then, we define RSI of criterion $\mathcal{J}$ as

$$RSI(\mathcal{J}) = \mathbb{I}\left[R_{\mathcal{J}} > \left(1 - \text{sign}(\hat{R})\sigma_{\text{RSI}}\right)\hat{R}\right] \log\left(\max\left(\frac{\hat{C}}{C_{\mathcal{J}}}, 1\right)\right), \tag{7}$$

where $\mathbb{I}[\cdot]$ is the indicator function and σ_{RSI} is a reward-tolerance threshold indicating the maximum allowed performance drop with the low-switching-cost constraint applied. In our experiments, we choose a fixed threshold parameter $\sigma_{\text{RSI}} = 0.2$. Intuitively, when the performance drop is moderate (i.e., within the threshold σ_{RSI}), RSI computes the logarithm of the relative switching cost improvements; while when the performance decreases significantly, the RSI score will be simply 0.

5.3 Results and Discussions

We compare the performances of all the criteria presented in Sec. 4, including unconstrained RL ("None"), non-adaptive switching ("Fix_1000"), policy-based switching ("Policy"), feature-based switching ("Feature") and two information-gain variants, namely visitation-based ("Visitation") and information-matrix-based ("Info") criteria.

GYMIC: This medical environment is relatively simple, and all the criteria achieve similar learning curves as unconstrained RL as shown in Fig. 1. However, the switching cost of visitation-based criterion is significantly higher – it almost overlaps with the cost of "None". While the other information-gain variant, i.e., information-matrix-based criterion, performs much better in this scenario. Overall, feature-based criterion produces the most satisfactory switching cost without hurt to sample efficiency.

Atari Games: We then compare the performances of different switching criteria in the more complex Atari games. The state spaces in Atari games are images, which are more complicated than

[3]We use $\hat{C}$ instead of K here since some RL algorithm may not update the policy every timestep.

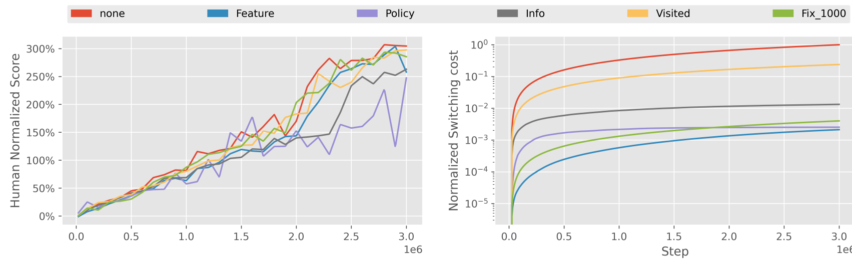

Figure 3: The average results on Atari games. We compare different switching criteria across 56 Atari games with 3 million training steps. We visualize the *human normalized reward* on the left. The figure on the right shows the average switching cost, which is normalized by the switching cost of "None" and shown in a log scale.

the low-dimensional states in GYMIC. Fig. 3 shows the average reward and switching of different switching criteria across all the 56 games, where the feature-based solution leads to the best empirical performance. We also remark that the non-adaptive baseline is particularly competitive in Atari games and outperforms all other adaptive solutions except the feature-based one. We also show the results in 8 representative games in Fig. 4, including the reward curves (odd rows) and switching cost curves (even rows). We can observe that information-gain variants produce substantially more policy updates while the feature-based and non-adaptive solutions are more stable.

In addition, we also noticed that the policy-based solution is particularly sensitive to its hyper-parameter in order to produce desirable policy reward, which suggests that the neural network features may change much more smoothly than the output action distribution.

To validate this hypothesis, we visualize the action difference and feature difference of the unconstrained Rainbow DQN on the Atari game "Pong" throughout the training process in Fig. 2. Note that in this case, the deployed policy is synced with the online policy in every training iteration, so the difference is merely due to a single training update. However, even in a unconstrained setting, the difference of action distribution fluctuates significantly. By contrast, the feature change is much more stable. We also provide some theoretical discussions on feature-based criterion in Appendix C.

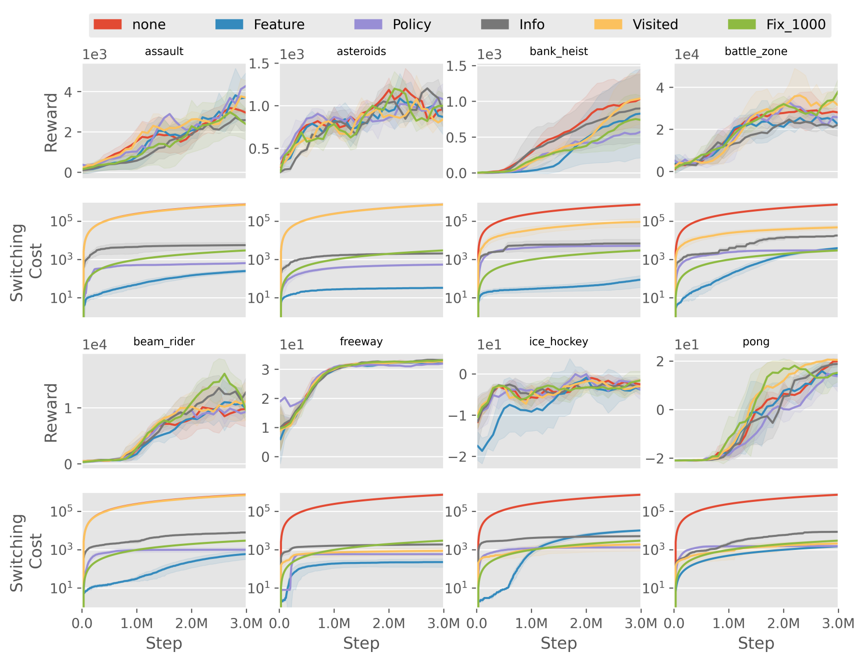

Figure 4: The results on several representative Atari games. In each environment, we visualize the training reward over the steps on the top and the switching cost in a log scale at the bottom.

MuJoCo Control: We evaluate the effectiveness of different switching criteria with SAC on all the 6 MuJoCo continuous control tasks. The results are shown in Fig. 5. In general, we can still observe

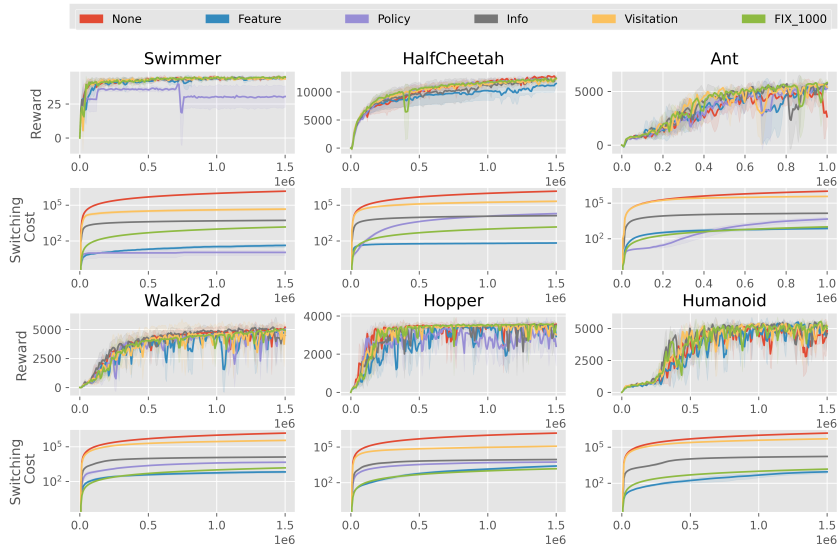

Figure 5: The results on MuJoCo tasks.

that the feature-based solution achieves the lowest switching cost among all the baseline methods while the policy-based solution produces the most unstable training. Interestingly, although the non-adaptive baseline has a relatively high switching cost than the feature-based one, the training curve has the less training fluctuations, which also suggests a future research direction on incorporating training stability into the switching criterion design.

Table 1: RSI (Eq. 7, $\sigma = 0.2$) for different criteria over different domains. We take unconstrained RL (i.e., "None") performance as the RSI reference, so the RSI value for "None" is always zero.

Avg. RSI	Feature	Policy	Info	Visitation	FIX_1000
GYMIC	**9.63**	4.16	8.88	0.0	6.91
Atari	**3.61**	2.82	2.11	1.81	3.15
Mujoco	**8.20**	3.45	4.83	1.92	6.91

Average RSI Scores: Finally, we also report the RSI scores of different policy switching criteria on different domains. For each domain, we compute the average value of RSI scores over each individual task in this domain. The results are reported in Table 1, where we can observe that the feature-based method consistently produces the best quantitative performance across all the 3 domains.

6 Conclusion

In this paper, we focus on low-switching-cost reinforcement learning problems and take the first empirical step towards designing an effective solution for reducing the switching cost while maintaining good performance. By systematic empirical studies on practical benchmark environments with modern RL algorithms, we find the existence of a theory-practice gap in policy switching criteria and suggest a feature-based solution can be preferred in practical scenarios. Thanks to the strong research nature of this work, we believe our paper does not produce any negative societal impact.

We remark that our paper not only provides a benchmark for future research but also raises many interesting methods. For example, although feature-based solution achieves the best overall performance, it does not substantially outperform the the naive non-adaptive baseline. It still has a great research room towards designing a more principled switching criteria. Another direction is to give provable guarantees for these policy switching criteria that work for methods dealing with large state space in contrast to existing analyses about tabular RL [4, 26, 25]. We believe our paper is just the first step on this important problem, which could serve as a foundation towards great future research advances.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2312–2320, 2011.

[2] Daniel Almirall, Scott N Compton, Meredith Gunlicks-Stoessel, Naihua Duan, and Susan A Murphy. Designing a pilot sequential multiple assignment randomized trial for developing an adaptive treatment strategy. *Statistics in medicine*, 31(17):1887–1902, 2012.

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

[4] Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. Provably efficient q-learning with low switching cost. In *Advances in Neural Information Processing Systems*, pages 8002–8011, 2019.

[5] Nicolo Cesa-Bianchi, Ofer Dekel, and Ohad Shamir. Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems*, pages 1160–1168, 2013.

[6] Minbo Gao, Tianle Xie, Simon S Du, and Lin F Yang. A provably efficient algorithm for linear markov decision process with low switching cost. *arXiv preprint arXiv:2101.00494*, 2021.

[7] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[9] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

[10] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[11] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *CoRR*, abs/2104.08212, 2021.

[12] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[13] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[14] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[15] Mufti Mahmud, M. Shamim Kaiser, Amir Hussain, and Stefano Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE Trans. Neural Networks Learn. Syst.*, 29(6):2063–2079, 2018.

[16] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, pages 1077–1084, 2014.

[17] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021.

[18] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim M. Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Anand Babu, Quoc V. Le, James Laudon, Richard C. Ho, Roger Carpenter, and Jeff Dean. Chip placement with deep reinforcement learning. *CoRR*, abs/2004.10746, 2020.

[19] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2430–2439. PMLR, 2017.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, S. Petersen, C. Beattie, A. Sadik, Ioannis Antonoglou, H. King, D. Kumaran, Daan Wierstra, S. Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[21] Yufei Ruan, Jiaqi Yang, and Yuan Zhou. Linear bandits with limited adaptivity and learning distributional optimal design. *arXiv preprint arXiv:2007.01980*, 2020.

[22] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Reinforcement learning with soft state aggregation. In *Advances in neural information processing systems*, pages 361–368, 1995.

[23] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, F. Turck, and P. Abbeel. Exploration: A study of count-based exploration for deep reinforcement learning. *ArXiv*, abs/1611.04717, 2017.

[24] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[25] Zihan Zhang, Xiangyang Ji, and Simon S. Du. Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon. *arXiv preprint arXiv:2009.13503*, 2020.

[26] Zihan Zhang, Yuan Zhou, and Xiangyang Ji. Almost optimal model-free reinforcement learning via reference-advantage decomposition, 2020.

[27] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A deep reinforcement learning framework for news recommendation. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 167–176. ACM, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]