Regret Minimization Experience Replay in Off-Policy Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In reinforcement learning, experience replay stores past samples for further reuse. Prioritized sampling is a promising technique to better utilize these samples. Previous criteria of prioritization include TD error, recentness and corrective feedback, which are mostly heuristically designed. In this work, we start from the regret minimization objective, and obtain an optimal prioritization strategy for Bellman update that can directly maximize the return of the policy. The theory suggests that data with higher hindsight TD error, better on-policiness and more accurate Q value should be assigned with higher weights during sampling. Thus most previous criteria only consider this strategy partially. We not only provide theoretical justifications for previous criteria, but also propose two new methods to compute the prioritization weight, namely ReMERN and ReMERT. ReMERN learns an error network, while ReMERT exploits the temporal ordering of states. Both methods outperform previous prioritized sampling algorithms in challenging RL benchmarks, including MuJoCo, Atari and Meta-World.

1 Introduction

Reinforcement learning (RL) [1] has achieved great success in sequential decision making problems. Off-policy RL algorithms [2, 3, 4, 5, 6] have the ability to learn from an arbitrary data distribution, and often provide better sample efficiency compared to on-policy counterparts. This is critical when the data collection process is expensive or dangerous. Experience Replay [7] enables data reuse and has been widely adopted in off-policy reinforcement learning.

Previous work [8] points out that selecting important samples from the replay buffer can benefit off-policy RL algorithms. Prioritized Experience Replay (PER) [9] quantifies such importance by the magnitude of temporal-difference (TD) error. It significantly improves the training efficiency of DQN, and has been adopted [4] and extended [10, 11, 12] by various deep reinforcement learning algorithms. Previous prioritization strategies are based on TD error [9, 10, 12] or focus on Q function learning [11]. However, such objectives can be theoretically suboptimal. For example, minimizing TD error is different from the objective of RL, which is to maximize the expected return of the policy. Even if we manage to reduce the TD error, there is no guarantee that the overall performance of the policy will improve [11].

In this paper, we first give examples to illustrate the issue of the indirect objectives in previous prioritization methods. To correct this issue, we first formulate an optimization problem that directly minimizes the regret of the current policy rather than TD error. Then, we derive the optimal prioritization strategy by solving this optimization problem. The optimal prioritization strategy indicates that we should pay more attention to experiences with higher hindsight TD error, better on-policiness and more accurate Q value. To the best of our knowledge, this paper is the first to optimize the sampling distribution of replay buffer theoretically from the perspective of regret minimization.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Based on our analysis, we provide tractable approximations to each term. The on-policiness can be estimated by training a classifier to distinguish recent transitions, which are generally more on-policy, from early ones, which are generally more off-policy. However, since the oracle Q value is inaccessible to the agent, we can not calculate the accuracy of Q value directly. Inspired by DisCor [11], we propose an algorithm named ReMERN to estimate the suboptimality of Q value with a neural network updated by Approximate Dynamic Programming (ADP).

Experiments show that ReMERN outperforms previous methods in environments with high randomness. However, training an extra neural network increases the computational cost, and requires more iterations to converge. Therefore, we propose another estimation of suboptimality based on a temporal viewpoint. In RL scenarios where states are temporally correlated, the error in Bellman update accumulates from the terminal state all across the trajectory. Since the terminal state has no bootstrapping target, it has low Bellman error. This intuition indicates that states fewer steps away from the terminal state can have lower error in updated Q value because of the more accurate Bellman target. We justify this intuition both empirically and theoretically. Then, we propose Temporal Correctness Estimation (TCE) based on the distance of each state to a terminal state, and name the overall algorithm ReMERT.

Similar to PER, ReMERN and ReMERT can be a plug-in module to all off-policy RL algorithms with a replay buffer, including but not limited to DQN [5] and SAC [2]. Experiments show that ReMERN and ReMERT substantially improve the performance of standard off-policy RL methods in various benchmarks.

2 Background

2.1 Preliminaries

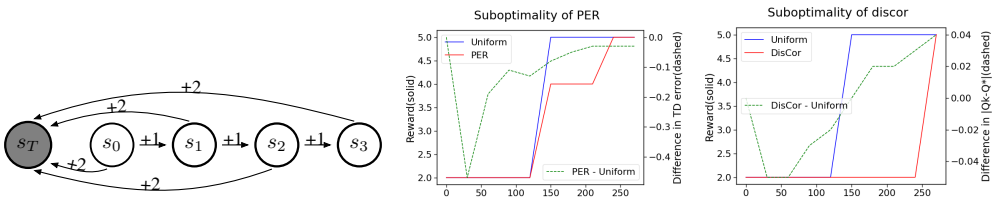
A Markov decision process (MDP) is denoted $(\mathcal{S}, \mathcal{A}, T, r, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. $T(s'|s,a)$ and $r(s,a) \in [0, \mathrm{R}_{\max}]$ are the transition and reward function. $\gamma \in (0,1)$ is the discounted factor and $\rho_0(s)$ is the distribution of the initial state. The target of reinforcement learning is to find a policy that maximizes the expected return: $\eta(\pi) = \mathbb{E}_\pi[\sum_{t\geq 0} \gamma^t r(s_t, a_t)]$, where the expectation is calculated from trajectories sampled from $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)$ for $t \geq 0$.

For a fixed policy, an MDP becomes a Markov chain, where the discounted stationary state distribution is defined as $d^\pi(s)$. With a slight abuse of notation, the discounted stationary state-action distribution is defined as $d^\pi(s,a) = d^\pi(s)\pi(a|s)$. We can then rewrite the expected return as $\eta(\pi) = \mathbb{E}_{d^\pi(s,a)}[r(s,a)]$. There exists an optimal policy π^* such that $\pi^* = \arg\max_\pi \eta(\pi)$. We use the standard definition of the state-action value function, or Q function: $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t\geq 0} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a]$. Let Q^* be the shorthand for Q^{π^*}. Q^* satisfies the Bellman equation $Q^* = \mathcal{B}^*(Q^*)$, where $\mathcal{B}^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is the Bellman optimal operator: $(B^* f)(s,a) := r(s,a) + \gamma \max_{a'} \mathbb{E}_{s'\sim P(s,a)} f(s', a')$, where $f \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$.

The regret of policy π can be defined as $\mathrm{Regret}(\pi) = \eta(\pi^*) - \eta(\pi)$. It measures the expected loss in return by following policy π instead of the optimal policy. Since $\eta(\pi^*)$ is a constant, minimizing the regret is equivalent to maximizing the expected return, and thus it can be an alternative objective of reinforcement learning.

2.2 Related Work

Extensive researches have been conducted on experience replay and replay buffer. The most frequently considered aspect is the sampling strategy. Various techniques have achieved good performance by performing prioritized sampling on the replay buffer. In model-based planning, Prioritized Sweeping [13, 14, 15] selects the next state updates according to changes in value. Prioritized Experience Replay (PER) [9] prioritizes samples with high TD error. Taking PER one step further, Prioritized Sequence Experience Replay (PSER) [10] considers information provided by transitions when estimating TD error. Emphasizing Recent Experience (ERE) [16] and Likelihood-Free Importance Weighting (LFIW) [12] prioritizes the correction of TD errors for frequently encountered states. Distribution Correction (DisCor) [11] assigns higher weights to samples with more accurate target Q value because these samples provide a more "corrective feedback". DisCor uses a neural



(a) (b) (c)

Figure 1: A simple MDP showing the objectives of PER and DisCor can slow down the training process. (a) A 5-state MDP with initial state s_0 and terminal state s_T. Except for s_3 and s_T, there are two available actions, *left* and *right*. Turning *left* leads to the terminal state s_T and +2 reward, while turning right leads to the next state and +1 reward. The optimal policy is to keep turning *right* until reaching s_3, then reach s_T. (b) Relationship between TD error (dashed line) and performance (solid line) of PER and uniformly sampling. (c) Relationship between Q error (dashed line) and performance (solid line) of DisCor and uniformly sampling.

network to estimate the accuracy of target Q value. Inspired by DisCor, SUNRISE [17] propose to use the variance of ensembled Q functions as a surrogate for the accuracy of Q value. Adversarial Feature Matching [18] focuses on sampling uniformly among state-action pairs.

Instead of proposing a new strategy of sampling, [19] proves that there exists a relationship between sampling strategy and loss function, and weighted value loss can serve as a surrogate for prioritized sampling. Other works focus on buffer capacity [20, 21]. They point out that a proper buffer capacity can accelerate value estimation and lead to better learning efficiency and performance. In fact, this can be thought as a specific example of prioritization strategies, i.e., assigning zero weights to the samples exceeding the proper buffer capacity.

3 Optimal Prioritization Strategy via Regret Minimization

3.1 A Rethink of Existing Prioritization Methods

PER and DisCor are two representative algorithms of prioritized sampling. PER prioritizes state-action pairs with high TD error, while DisCor prefers to perform Bellman update on state-action pairs that lead to more accurate Q value. However, both criteria are different from the target of RL algorithms, which is to maximize the expected return of the policy. Such difference can slow down the training process in some cases. For example, when the Bellman target is inaccurate, minimizing TD error does not necessarily improve the optimality of Q value.

To illustrate the aforementioned problems, we provide an example in the left part of Fig. 1(a). The optimal policy is to turn *right* in all states, receiving a total reward of 5. Suppose the Q values for all (s, a) pairs are initialized to zero. First, the single step reward of turning *left* is higher than turning *right* in all states, so the *left* action has a higher TD error. As a result, PER is more focused on optimizing the estimate of Q value of the *left* action, which is not the fastest training process to achieve the optimal policy. Second, since there is no bootstrapping error for the terminal state s_T, transitions with s_T as the next state have an accurate target Q-value. Therefore, DisCor focuses on state-actions pairs with *left* action, which is again not optimal.

We perform value iterations on this MDP. To simulate function approximation in Deep RL and avoid convergence in few iterations, the learning rate is set to 0.1. Prioritized sampling is substituted by weighted Bellman update, as introduced in [19]. The results are shown in Fig. 1(b) and Fig. 1(c). According to the results, PER indeed minimizes TD error more efficiently, and DisCor results in a more accurate estimate of Q value, as indicated by their objectives. However, they both need more iterations to converge than uniform sampling. Therefore, their objectives can be inefficient in certain cases.

3.2 Problem Formulation by Minimizing Regret

As shown in section 3.1, an indirect objective can cause suboptimal performance in certain cases. In this section, we aim to find an optimal prioritization weight w_k that can directly minimize the policy regret $\eta(\pi^*) - \eta(\pi_k)$. Since the detailed process of Bellman update is irrelevant to w_k, we assume that the optimal result of Bellman update is available in each iteration, which is calculated by the $\arg\min$ operator. Therefore, the optimization problem with respect to w_k can be written as:

$$\begin{aligned} \min_{w_k} \quad & \eta(\pi^*) - \eta(\pi_k) \\ \text{s.t.} \quad & Q_k = \arg\min_{Q} \ \mathbb{E}_{\mu}[w_k(s,a) \cdot (Q - \mathcal{B}^* Q_{k-1})^2], \end{aligned} \tag{1}$$

where $\pi_k(s) = \arg\max_a Q_k(s,a)$ is the policy corresponding to Q_k and μ is the data distribution of the replay buffer. The prioritization weight w_k modifies the distribution μ and determines how the Bellman error $(Q - \mathcal{B}^* Q_{k-1})^2$ is calculated. Q_k is the estimate of Q value after the Bellman update at iteration k, and is optimal with respect to Bellman target $\mathcal{B}^* Q_{k-1}$.

Then we manage to solve this optimization problem. The solution is presented in Thm. 1. The formal version of the theorem and detailed proof is derived in Appendix A.

Theorem 1 (Informal)**.** *Under mild conditions, the solution w_k to a relaxation of the optimization problem 1 in MDPs with discrete action spaces is*

$$w_k(s,a) \propto \underbrace{\frac{d^{\pi_k}(s,a)}{\mu(s,a)}}_{(a)} \underbrace{(2 - \pi_k(a|s))}_{(b)} \underbrace{\exp\left(-\left|Q_k - Q^*\right|(s,a)\right)}_{(c)} \underbrace{\left|Q_k - \mathcal{B}^* Q_{k-1}\right|(s,a)}_{(d)}. \tag{2}$$

In MDPs with continuous action spaces, the solution is

$$w_k(s,a) \propto \underbrace{\frac{d^{\pi_k}(s,a)}{\mu(s,a)}}_{(a)} \underbrace{\exp\left(-\left|Q_k - Q^*\right|(s,a)\right)}_{(c)} \underbrace{\left|Q_k - \mathcal{B}^* Q_{k-1}\right|(s,a)}_{(d)}. \tag{3}$$

Thm. 1 suggests that state-action tuples in the replay buffer should be assigned with higher importance if they have the following properties:

- **high hindsight Bellman error** (from $|Q_k - \mathcal{B}^* Q_{k-1}|(s,a)$). Q_k is the estimate of Q value after the Bellman update. This term describes the difference between the estimated hindsight Q value and the Bellman target. It is similar to the prioritization criterion of PER [9], but PER concerns more about the historical Bellman error, i.e., $|Q_{k-1} - \mathcal{B}^* Q_{k-2}|(s,a)$.
- **more on-policiness** (from $\frac{d^{\pi_k}(s,a)}{\mu(s,a)}$). An efficient update of π requires w_k to be on-policy, focusing on state-action pairs which are more likely to be visited by the current policy. Such prioritization strategy has been empirically illustrated in LFIW [12] and BCQ [22], while we obtain it directly from our theorem.
- Q_k **is close to** Q^* (from $\exp\left(-\left|Q_k - Q^*\right|(s,a)\right)$). This term indicates that state-action pairs with less accurate Q values after the Bellman update should be assigned with lower weights. Intuitively, state-action pairs that lead to suboptimal updates of the estimator of Q value should be down-weighted. Such suboptimality may arise from incorrect target Q-values or the error of function approximation of deep Q networks.
- a_k **is less likely to be selected by the policy** (from $2 - \pi_k(a|s)$). This term only exists in MDPs with a discrete action space. It offsets the effect of the on-policy term d^{π_k} to some extent and is similar to ε-greedy strategy in exploration.

Our theoretical analysis indicates that existing prioritization strategies only consider the problem partially, neglecting other terms in minimizing the regret. For example, DisCor fails to consider the on-policiness and PER ignores the accuracy of Q value. In the remaining part of this section, we present practical approximations to each term in Eq. (2) and (3).

Term (a) is the importance weight between the current policy and the behavior policy. We can calculate this term using Likelihood-Free Importance Weighting (LFIW, [12]). LFIW divides the

replay buffer into two parts, a fast buffer $\mathcal{D}_f$ and a slow buffer $\mathcal{D}_s$. It initializes a neural network $\kappa_\psi(s,a)$ and optimizes the network according to the following loss function:

$$L_\kappa(\psi) := \mathbb{E}_{\mathcal{D}_\mathrm{s}}\left[f^*\left(f'\left(\kappa_\psi(s,a)\right)\right)\right] - \mathbb{E}_{\mathcal{D}_\mathrm{f}}\left[f'\left(\kappa_\psi(s,a)\right)\right] \tag{4}$$

The updated κ_ψ is the desired importance weight.

For term (b) and (d), since π_k and Q_k are the policy and the estimate of Q value after the update, they cannot be calculated directly. Therefore, we approximate them by the upper and lower bounds. For term (b), $1 \leq 2 - \pi_k(a|s) \leq 2$. For term (d), a viable approximation is to bound it between the minimum and maximum Bellman errors obtained at the previous iteration, $c_1 = \min_{s,a} |Q_{k-1} - \mathcal{B}^* Q_{k-2}|$ and $c_2 = \max_{s,a} |Q_{k-1} - \mathcal{B}^* Q_{k-2}|$. As shown in DisCor, we can restrict the support of state-action pairs (s,a) used to compute c_1 and c_2 in the support of replay buffer, to ensure that both c_1 and c_2 are finite. With these approximation, we can derive a lower bound for w_k, which will be detailed in Sec. 3.3 and Sec. 3.4.

In the next two subsections, we will provide two practical algorithms to estimate $|Q_k - Q^*|$.

3.3 Regret Minimization Experience Replay Using Neural Network (ReMERN)

DisCor shows Δ_k can be a surrogate of $|Q_k - Q^*|$, which is defined as:

$$\Delta_k = \sum_{i=1}^{k} \gamma^{k-i} \left(\prod_{j=i}^{k-1} P^{\pi_j} \right) |Q_i - \mathcal{B}^* Q_{i-1}| \tag{5}$$

$$\Longrightarrow \Delta_k = |Q_k - \mathcal{B}^* Q_{k-1}| + \gamma P^{\pi_{k-1}} \Delta_{k-1} \tag{6}$$

According to Eq. (6), $\gamma[P^{\pi_{k-1}}\Delta_{k-1}](s,a) + c_2$ is an upper bound of $|Q_k - Q^*|$. This is because Δ_k is proven to be the upper bound of $|Q_k - Q^*|$ [11] and c_2 is the upper bound of $|Q_k - \mathcal{B}^* Q_{k-1}|$. Recall that $2 - \pi_k(a|s) \geq 1$ and $|Q_{k-1} - \mathcal{B}^* Q_{k-2}| \geq c_1$, and we can derive the final expression for this tractable approximation for $w_k(s,a)$ by simplifying all constants:

$$w_k(s,a) \propto \frac{d^{\pi_k}(s,a)}{\mu(s,a)} \exp\left(-\gamma \left[P^{\pi_{k-1}} \Delta_{k-1}\right](s,a)\right), \tag{7}$$

This approximation applies to MDPs with discrete action space and MDPs with continuous action space. Using the lower bound of $w_k(s,a)$ may down-weight more transitions, but will never up-weight a transition by mistake.

We use a neural network to estimate Δ_{k-1}. As shown in Eq. (6), Δ_{k-1} can be calculated from a bootstrapped target, which inspires us to use ADP algorithms to update it. We name this method ReMERN (**Re**gret **M**inimization **E**xperience **R**eplay using **N**eural Network). The pseudo code for ReMERN is presented in Appendix C. ReMERN is applicable to all value-based off-policy algorithms with replay buffer. Specifically, we multiply w_k to the TD error to reweight the samples in DQN [5] (discrete action space) and SAC [2] (continuous action space).

3.4 Regret Minimization Experience Replay Using Temporal Structure (ReMERT)

ReMERN uses neural network as the estimator of $|Q_k - Q^*|$. However, training a neural network is time consuming and suffers from large estimation error without adequate iterations. To mitigate this issue, we propose another estimation of $|Q_k - Q^*|$ from a different perspective.

3.4.1 The Temporal Property of Q Error

$|Q_k - Q^*|$ can be decomposed using triangle inequality: $|Q_k - Q^*| \leq |Q_k - \mathcal{B}^* Q_{k-1}| + |\mathcal{B}^* Q_{k-1} - Q^*|$. The first term is the optimization error, which depends on the optimization algorithm. In the training process, the optimization error decreases rapidly [23, 24] to a small value compared to the second term, so it can be ignored, and the second term becomes dominating. In the second term, $\mathcal{B}^* Q_{k-1}$ is the target Q-value estimate, and $|\mathcal{B}^* Q_{k-1} - Q^*|$ describes the distance from the target Q-value to the ground-truth Q value. As clarified previously, the ground-truth Q value is inaccessible to the agent. However, we can estimate it with prior knowledge. Since the target Q-value at the terminal state consists of the reward only, there is no bootstrapping error and $|\mathcal{B}^* Q_{k-1} - Q^*| = 0$.

Moving backward through the trajectory, the accuracy of the Q-value estimation decreases as the error of Bellman update accumulates. These Q values are then utilized to compute the target Q-value, leading to more erroneous Bellman updates and larger $|\mathcal{B}^* Q_{k-1} - Q^*|$. Such error can accumulate through the MDP. Consequently, states closer to the terminal state tend to have a more accurate Bellman target. This motivates us to estimate the incorrectness of the estimated Q value using the temporal information of a given state-action tuple (s_t, a_t).

To justify our intuition on the temporal property of Q error, we use a gridworld MDP from [25] and visualize the mean error of the target Q-value (i.e., $|\mathcal{B}^* Q_{k-1} - Q^*|$) across different actions in Fig. 2. We use DQN to update Q values. In this gridworld MDP, an agent starts at the red triangle on the top-left and terminates at the green rectangle on the top-right. The agent can't go through the wall, which is plotted as gray grids. The darker a grid is, the higher error of Q function it has. This figure illustrates that states closer to the terminal state, indeed has lower Q error, which confirms our intuition that $|\mathcal{B}^* Q_{k-1} - Q^*|$ is related to the position of (s, a) in the trajectory.

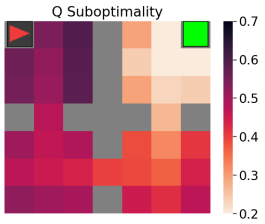


Figure 2: The visualized error of target Q-value in a GridWorld Environment. The Q error is visualized by the color of the grid.

To introduce our algorithm, we first define *Distance to End*.

Definition 1 (Distance to End)**.** *Given a MDP $\mathcal{M}$, $\tau = \{s_t, a_t\}_{t=0}^{T}$ is a trajectory generated by policy π in $\mathcal{M}$. The distance to end of (s_t, a_t), denoted by $h_\tau^\pi(s_t, a_t)$, is $T - t$ in this trajectory.*

Our intuition states that the value of $|Q_k - Q^*|$ has a positive correlation with distance to end. To formulate this intuition, we propose the following theorem.

Theorem 2 (Informal)**.** *For an MDP, the distribution of the sampled trajectories by policy π_k is $q_k(\tau)$. Under mild conditions, ignoring some terms that converge to zero when $k \to \infty$,*

$$|Q_k(s,a) - Q^*(s,a)| \leq \frac{\gamma R_{\max}}{1-\gamma} + \mathop{\mathbb{E}}_{q_k(\tau)} \left(f(h_\tau^{\pi_k}(s,a)) \big(L_{Q_{k-1}} + c\big) + \gamma^{h_\tau^{\pi_k}(s,a)+1} c \right) \tag{8}$$

holds with high probability, where $c = \max_{s,a} \big(Q^*(s,a^*) - Q^*(s,a)\big)$, $f(t) = \frac{\gamma - \gamma^t}{1-\gamma}$ and $L_{Q_{k-1}} = \mathbb{E}[|Q_{k-1} - \mathcal{B}^* Q_{k-2}|]$.

The formal version of the theorem and its proof are in Appendix B. The theorem states that $|Q_k - Q^*|$ is upper bounded by a function of distance to end and expected Bellman error with high probability.

3.4.2 A Practical Implementation

The upper bound of $|Q_k - Q^*|$ gives us a hint for approximation. Using an upper bound as the surrogate may down-weight some transitions, but will never up-weight a transition that should not be up-weighted. Thus, the upper bound can serve as a surrogate to $|Q_k - Q^*|$. Then, we show how to calculate the upper bound.

At iteration k, the Q function is updated according to: $Q_k = (1-\alpha) Q_{k-1} + \alpha(\mathcal{B}^* Q - Q_{k-1})$, where α is the learning rate. $|Q_k - Q_{k-1}|$ is a relatively small value because α is usually a small hyperparameter. Since there is limited difference between Q_k and Q_{k-1}, the corresponding π_k and π_{k-1} are also similar. Therefore, we can use $h_\tau^{\pi_{k-1}}$ and q_{k-1} instead of $h_\tau^{\pi_k}$ and q_k. In this way, $|Q_k(s,a) - Q^*(s,a)|$ can be approximated using the Temporal Correctness Estimation (TCE) algorithm, which is:

$$\begin{aligned} |Q_k(s,a) - Q^*(s,a)| &\approx \mathbb{E}_{q_{k-1}(\tau)} \mathrm{TCE}_c(s,a) \\ &= \mathbb{E}_{q_{k-1}(\tau)} \left(\frac{\gamma R_{\max}}{1-\gamma} + f(h_\tau^{\pi_{k-1}}(s,a)) \big(L_{Q_{k-1}} + c\big) + \gamma^{h_\tau^{\pi_{k-1}}(s,a)+1} c \right), \end{aligned} \tag{9}$$

where c is a constant in Thm. 2. In our practical implementation, c is a hyperparameter.

Similar to the derivation of ReMERN, we can simplify the expression of $w_k(s,a)$ as:

$$w_k(s,a) \propto \frac{d^{\pi_k}(s,a)}{\mu(s,a)} \exp\Big(-\mathbb{E}_{q_{k-1}(\tau)} \mathrm{TCE}_c(s,a) \Big) \tag{10}$$

$\mathbb{E}_{q(\tau)}(\text{TCE}_c(s,a))$ is not easy to calculate, because an accurate estimation requires an average of $\text{TCE}_c(s,a)$ over all trajectories generated by the current policy, which is not sample efficient. Therefore, to get an instant approximation, we modify Eq. (10) to be a tractable approximation for w_k:

$$w_k(s,a) \propto \mathbb{E}_{q_{k-1}(\tau)} \frac{d^{\pi_k}(s,a)}{\mu(s,a)} \exp\Big(-\text{TCE}_c(s,a)\Big) \tag{11}$$

In this equation, the expectation is put outside the exponential, which introduces an extra bias. This bias increases as the variance of $h^{\pi}_{\tau}(s,a)$ increases. However, this bias can be neglected when $h^{\pi}_{\tau}(s,a)$ is relatively fixed. In other words, Eq. (11) is accurate when the temporal ordering of state-action pairs is relatively fixed. In the following, we will call this kind of property as with stable temporal ordering of states. We verify this property in Appendix D. Then we formulate our algorithm ReMERT (**Re**gret **M**inimization **E**xperience **R**eplay using **T**emporal Structure), the pseudo code is presented in Appendix C. Similar to ReMERN, we use DQN and SAC as base algorithms for ReMERT in our experiments.

4 Experiments

In this section, we conduct experiments to evaluate ReMERN and ReMERT. Both algorithms are combined with DQN and SAC, and we include them as baselines. We first compare the performance of ReMERN and ReMERT to prior methods in continuous control benchmarks including Meta-World [26], MuJoCo [27] and Deepmind Control Suite (DMC) [28]. We also evaluate our methods in Arcade Learning Environments with discrete action spaces. Then, we dive into our algorithms and design several experiments, such as Gridworld tasks and MuJoCo with reward noise, to demonstrate some key properties of ReMERN and ReMERT. A detailed description of the environments and experimental details are listed in Appendix D.

4.1 Performance on Continuous Control Environments

In MuJoCo and DMC tasks, ReMERT outperforms baseline methods on four of six tasks and achieves comparable performance in the rest two tasks, i.e. HalfCheetah and Hopper, as shown in Fig. 3. The marginal improvement of ReMERT in HalfCheetah mainly comes from the absence of a strong correlation between Q-loss and time step. In HalfCheetah, there is no specific terminal state, so the agent always reaches the max length of the trajectory, which gives a fake "distance to end" for every state. In Hopper, there is not much difference of the $|Q_k - Q^*|$ term between all the

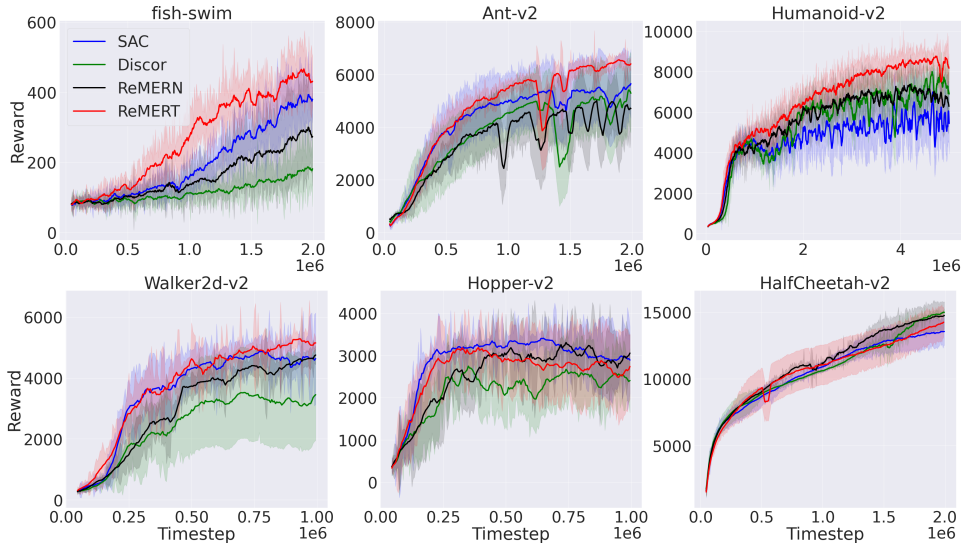


Figure 3: Performance of ReMERT, ReMERN with SAC and DisCor as baselines on continuous control tasks.

sampled state-action pairs, as shown in Appendix D, so the state-action pairs are not sampled very unequally. Besides, Hopper is a relatively easy task, in which prioritizing the samples have minor impact on the overall performance of the RL algorithm. The performance of ReMERN is better than DisCor, but is not as good as ReMERT. This verifies our theory and the existence of large estimation error induced by updating neural network with ADP algorithms.

The Meta-World benchmark [26] includes many robotic manipulation tasks. We select 8 tasks for evaluation, and plot the result in Fig. 4. The performance of PER can be found in its paper [9]. Current state-of-the-art off policy RL algorithms such as SAC performs poorly on this benchmark because the goals of tasks have high randomness. Although DisCor [11] shows preferable performance in these tasks compared to SAC and PER, ReMERN obtains a significant improvement over DisCor in the training speed and asymptotic performance. In this evaluation, we exclude ReMERT for comparison because the randomized target position in Meta-World contradicts its assumption.

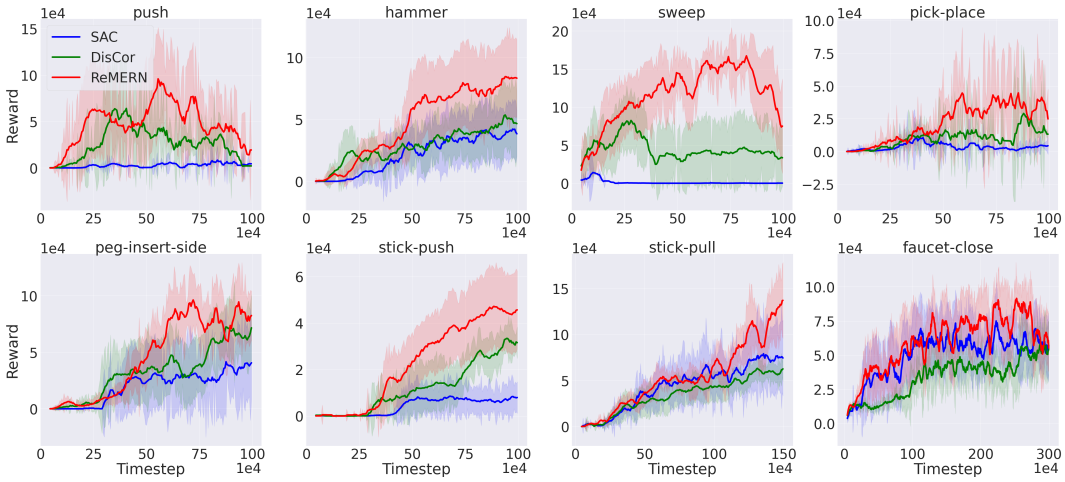


Figure 4: Performance of ReMERN, standard SAC and DisCor in eight Meta-World tasks. From left to right: push, hammer, sweep, peg-insert-side, stick push, stick pull, faucet close.

4.2 Performance on Arcade Learning Environments

Atari games are suitable for verifying our theory for MDPs with discrete action space. The tested games have a relatively stable temporal ordering of states because the initial state and the terminal state have little randomness, so that the assumption of ReMERT is satisfied. As shown by Tab. 1, ReMERT outperforms DQN in all the selected games. The results also suggest that ReMERT can be applied to environments with high dimensional state spaces. Results of more Atari games are listed in Appendix D. We do not include ReMERN for comparison because DisCor which is a composing part of ReMERN has no open-source code available for discrete action space.

4.3 Demonstration of Key Properties of ReMERN and ReMERT

4.3.1 Influence of Environment Randomness

Fig. 4 and Tab.1 show that ReMERN has a better performance on Meta-World than on Mujoco tasks. We attribute this to the robustness of ReMERN in environments with high randomness. For a highly

Table 1: DQN vs ReMERT on Atari. DQN (Nature) is the performance in the DQN paper [5]. DQN (Baseline) is the performance of our baseline program [29].

Method	Enduro	KungFuMaster	Kangaroo	MsPacman	Qbert
DQN (Nature)	301±24.6	23270±5955	6740±2959	2311±525	10596±3294
DQN (Baseline)	1185±100	29147±7280	6210±1007	3318±647	13437±2537
ReMERT (Ours)	**1303**±258	**35544**±8432	**7572**±1794	**3481**±1351	**14511**±1138

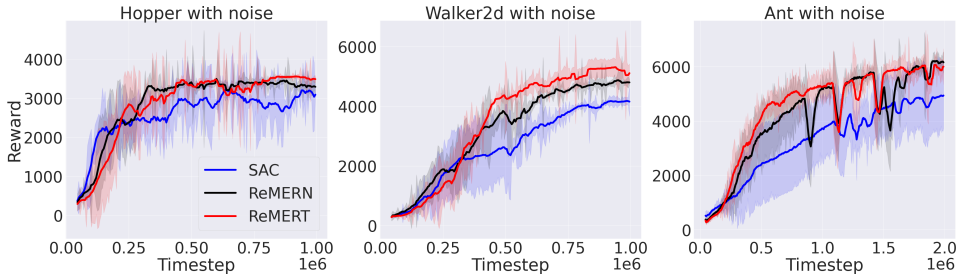


Figure 5: Performance of ReMERN, ReMERT and SAC on three continuous control tasks with reward noise.

stochastic environment, the estimation of Q value is difficult. When the estimation of Q value is inaccurate, the target Q-value is also inaccurate, leading to a suboptimal update process in off-policy RL algorithms. Thanks to the extra information provided by the error network, ReMERN estimates the Q value more accurately than other methods. However, for less stochastic environments like MuJoCo environments, the accuracy of error network might become the bottleneck of ReMERN.

To show this empirically, we add Gaussian noise to the reward function in MuJoCo environments. The details of the experimental setup are listed in Appendix D. Fig. 5 show that: (1) ReMERN performs better than SAC in stochastic environments, which verifies our analysis. (2) ReMERT has comparable performance to ReMERN, because the temporal property is not affected by the noise.

4.3.2 Analysis of TCE on Deterministic Tabular Environments

To analyze the effect of the principle behind TCE, we evaluate the Q error in Gridworld with image input. We plot the $|Q_k - Q^*|$ error of standard DQN, DQN with DisCor, DQN with TCE and DQN with oracle at some time in the training process in Fig. 6. TCE is combined with DQN to estimate term (c) in Eq. (2) , and the other terms are ignored to compute w_k. DQN with oracle uses the ground-truth error $|Q_k - Q^*|$ to calculate the prioritization weight. The result shows that DQN with TCE achieves a more accurate Q value estimator than those of standard DQN and DQN with DisCor, while DQN with oracle $|Q_k - Q^*|$ achieves the most accurate Q value estimator. The lower efficiency of DQN with DisCor is due to the slower convergence speed of the error network. This proves the principle behind our theory effective, and TCE is a decent approximation of $|Q_k - Q^*|$.

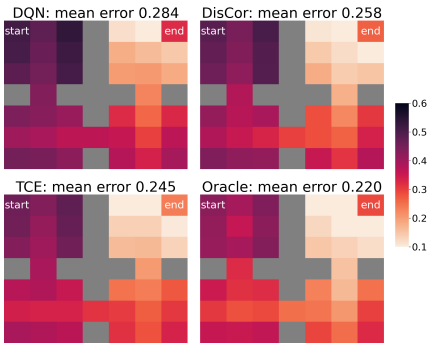


Figure 6: TCE and DisCor in Gridworld

5 Conclusion and Future Work

In this work, we first revisit the existing methods of prioritized sampling and point out that the objectives of these methods are different from the objective of RL, which can lead to a suboptimal training process. To solve this issue, we analyze the prioritization strategy from the perspective of regret minimization, which is equivalent to return maximization in RL. Our analysis gives a theoretical explanation for some prioritization methods, including PER, LFIW and DisCor. Based on our theoretical analysis, we propose two practical prioritization strategies, ReMERN and ReMERT, that directly aims to improve the policy. ReMERN is robust to the randomness of environments, while ReMERT is more computational efficient and more accurate in environments with a stable temporal ordering of states. Our approaches obtain superior results compared to previous prioritized sampling methods. Future work can be conducted in the following two directions. First, the framework to obtain the optimal distribution in off-policy RL can be generalized to model-based RL and offline RL. Second, since the two proposed algorithms are suitable to different kinds of MDP, finding an criterion to automatically switch between them can further improve the performance.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1856–1865, 2018.

[3] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018.

[4] 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 Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the 32nd Conference on Artificial Intelligence*, pages 3215–3222. AAAI Press, 2018.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

[6] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017.

[7] Long Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8:293–321, 1992.

[8] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2530–2539. PMLR, 2018.

[9] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the 4th International Conference on Learning Representations*, 2016.

[10] Marc Brittain, Joshua R. Bertram, Xuxi Yang, and Peng Wei. Prioritized sequence experience replay. *CoRR*, abs/1905.12726, 2019.

[11] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Proceedings of 33rd conference on Neural Information Processing Systems*, 2020.

[12] Samarth Sinha, Jiaming Song, Animesh Garg, and Stefano Ermon. Experience replay with likelihood-free importance weights. *CoRR*, abs/2006.13169, 2020.

[13] David Andre, Nir Friedman, and Ronald Parr. Generalized prioritized sweeping. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *proceedings of the 10th conference on Neural Information Processing Systems*, pages 1001–1007. The MIT Press, 1997.

[14] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Mach. Learn.*, 13:103–130, 1993.

[15] Harm van Seijen and Richard S. Sutton. Planning by prioritized sweeping with small backups. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 361–369. JMLR.org, 2013.

[16] Che Wang, Yanqiu Wu, Quan Vuong, and Keith Ross. Striving for simplicity and performance in off-policy DRL: output normalization and non-uniform sampling. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10070–10080. PMLR, 2020.

[17] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *arXiv preprint arXiv:2007.04938*, 2020.

[18] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2021–2030. PMLR, 2019.

[19] Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2020.

[20] Shangtong Zhang and Richard S. Sutton. A deeper look at experience replay. *CoRR*, abs/1712.01275, 2017.

[21] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3061–3071. PMLR, 2020.

[22] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 2019.

[23] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Proceedings of the 20th Conference on Neural Information Processing*, pages 161–168. Curran Associates, Inc., 2007.

[24] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Proceedings of the 26th Conference on Neural Information Processing*, pages 315–323, 2013.

[25] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

[26] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the 3rd Conference on Robot Learning*, 2019.

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

[28] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm-control: Software and tasks for continuous control, 2020.

[29] Jiayi Weng, Huayu Chen, Alexis Duburcq, Kaichao You, Minghao Zhang, Dong Yan, Hang Su, and Jun Zhu. Tianshou. https://github.com/thu-ml/tianshou, 2020.

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? [N/A]
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...
   (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)? [No]
   (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? [N/A]
   (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]