Reward Prediction Error as an Exploration Objective in Deep RL

Riley Simmons-Edler[1,2*], **Ben Eisner**[2], **Daniel Yang**[2], **Anthony Bisulco**[2], **Eric Mitchell**[2,3], **Sebastian Seung**[2] and **Daniel Lee**[2]

[1]Princeton University
[2]Samsung AI Center NYC
[3]Stanford University
rileys@cs.princeton.edu

Abstract

A major challenge in reinforcement learning is *exploration*, when local dithering methods such as ϵ-greedy sampling are insufficient to solve a given task. Many recent methods have proposed to intrinsically motivate an agent to seek novel states, driving the agent to discover improved reward. However, while state-novelty exploration methods are suitable for tasks where novel observations correlate well with improved reward, they may not explore more efficiently than ϵ-greedy approaches in environments where the two are not well-correlated. In this paper, we distinguish between exploration tasks in which seeking novel states aids in finding new reward, and those where it does not, such as goal-conditioned tasks and escaping local reward maxima. We propose a new exploration objective, maximizing the reward prediction error (RPE) of a value function trained to predict extrinsic reward. We then propose a deep reinforcement learning method, QXplore, which exploits the temporal difference error of a Q-function to solve hard exploration tasks in high-dimensional MDPs. We demonstrate the exploration behavior of QXplore on several OpenAI Gym MuJoCo tasks and Atari games and observe that QXplore is comparable to or better than a baseline state-novelty method in all cases, outperforming the baseline on tasks where state novelty is not well-correlated with improved reward.

1 Introduction

In recent years deep reinforcement learning (RL) algorithms have demonstrated impressive performance on tasks such as playing video games and controlling robots [Mnih *et al.*, 2015; Kalashnikov *et al.*, 2018]. However, successful training for such cases typically requires both a well-shaped reward function, where the RL agent can sample improved trajectories through simple dithering exploration such as ϵ-greedy sampling, and the ability to collect many (hundreds of thousands to millions) of trials. Satisfying these preconditions often requires large amounts of domain-specific engineering. In particular, reward function design can be unintuitive, may require many iterations of design, and in some domains such as robotics can be physically impractical to implement.

The field of *exploration* methods in RL seeks to address the difficulties of reward design by allowing RL agents to learn from *unshaped* reward functions. Unshaped functions (for example, a reward of 1 when an object is moved to a target, and 0 otherwise) are usually much easier to design and implement than dense well-shaped reward functions. However, it is hard for standard RL algorithms to discover good policies on unshaped reward functions, and they may learn very slowly, if at all.

While substantial work has been conducted on designing general exploration strategies for high-dimensional Markov Decision Processes (MDPs) with sparse reward functions, few studies have distinguished between different types of tasks requiring exploration, particularly in terms of which signals in each MDP are useful for discovering new sources of reward. In this work, we consider three types of exploration challenges in particular: solving mazes, learning goal conditioning relationships, and escaping local reward maxima.

Many classical exploration tasks can be described well as **mazes**. For example, discovering the single rewarding state in a sparse reward environment, or navigating a precise series of obstacles in order to play a game. Qualitatively, the agent must search for the exit to the maze (reward), receives little or no reward before finding it, and has no learned priors. In the limit any RL task can be seen as a maze (such as by treating a single optimal trajectory as the "exit"), but such a treatment is often intractable for large MDPs.

Related but distinct are **goal conditioned tasks**. Here, the reward function is conditioned on a non-static goal specified by the environment and discovered through interaction. For example, a robot that must move an object to a set of coordinates, which differ for each episode. The agent must learn how the observation and reward are conditioned on the goal, which is made significantly harder when the underlying reward function is sparse and unshaped. Unlike a maze, correlations between observation and goal/reward provide additional information an agent can use to solve the problem.

Lastly, in a poorly-shaped reward function there may exist local maxima in the space of trajectories, where an agent cannot discover an improved policy through local exploration

*Contact Author

and must deliberately sample suboptimal trajectories to **escape local maxima**. Here, the contours of the reward function can provide information on what directions of exploration might be informative, even if exploring them does not immediately maximize reward.

This distinction is important because both goal conditioning and local maxima introduce additional information about the task that mazes do not contain — In a maze-like environment, discovering new states is explicitly linked with discovering new reward signals. Goal conditioning can provide hints as to what states are and are not rewarding (and when) through correlations in the observation. Similarly, local reward maxima are embedded within a dense reward function which provides correlations between each observation and the reward that results, and discovering this relationship may lead to improved reward. Each of these problems can be intractable using naive exploration (depending on the severity of the problem), but each in turn provides some signal that can be used to solve it. For example, goal-conditioning relationships can also be learned by goal-driven RL methods such as Hindsight Experience Replay [Andrychowicz *et al.*, 2017], which by assuming the presence of a goal can learn much faster and more sample-efficiently on that class of problems.

In this paper, we propose the use of reward prediction error, specifically the Temporal-Difference Error (TD-Error) of a value function, to direct exploration in MDPs that contain Goal Conditioning and Local Maxima Escape problems but do not have a strong correlation between reward discovery and state novelty. To facilitate the use of this objective in a deep reinforcement learning setting for high-dimensional MDPs, we introduce QXplore, a new deep RL exploration formulation that seeks novelty in the predicted reward landscape instead of novelty in the state space. QXplore exploits the inherent reward-space signal from TD-error in value-based RL, and directly promotes visiting states where the current understanding of reward dynamics is poor. In the following sections, we describe QXplore for continuous MDPs and demonstrate its utility for efficient learning on a variety of complex benchmark environments showcasing different exploration cases.

2 Related Work

Of the exploration methods proposed for deep RL settings, the majority provide some state-novelty objective that incentivizes an agent to explore novel states or transition dynamics. A simple approach consists of explicitly counting how many times each state has been visited, and acting to visit rarely explored states. This approach can be useful for small MDPs, but often performs poorly in high-dimensional or continuous state spaces. However, several recent works [Tang *et al.*, 2017; Bellemare *et al.*, 2016; Fu *et al.*, 2017] using count-like statistics have shown success on benchmark tasks with complex state spaces.

Another approach to environment novelty learns a model of the environment's transition dynamics and considers novelty as the error of the model in predicting future states or transitions. This exploration method relies on the assumption that any new state that can be predicted in advance is equivalent to some previously seen state in its effect on reward. Predictions of the transition dynamics can be directly computed [Pathak *et al.*, 2017; Stadie *et al.*, 2015], or related to an information gain objective on the state space, as described in VIME [Houthooft *et al.*, 2016] and EMI [Kim *et al.*, 2019].

Several exploration methods have recently been proposed that capitalize on the function approximation properties of neural network to recognize novel states. Random network distillation (RND) trains a function to predict the output of a randomly-initialized neural network from an input state, and uses the approximation error as a reward bonus for a separately-trained RL agent [Burda *et al.*, 2019]. Similarly, DORA [Fox *et al.*, 2018] trains a network to predict zero on observed states and deviations from zero are used to indicate unexplored states.

These methods have been shown to perform well on maze-solving exploration tasks such as the Atari game `Montezuma's Revenge`, where maximizing reward (game score) requires visiting each room of the game, which also maximizes the diversity of states and observations experienced. However, evaluating these methods on tasks where novelty does not correlate highly with reward, such as on other Atari games, shows little improvement over ϵ-greedy [Taiga *et al.*, 2020].

Reward prediction error has been previously used for exploration in a few cases. Previous works described using reward misprediction and model prediction error for exploration [Schmidhuber, 1991; Thrun and Möller, 1992]. However, these works were primarily concerned with model-building and system-identification in small MDPs, and used single-step reward prediction error rather than TD-error. Later, TD-error was used as a negative signal to constrain exploration to focus on states that are well understood by the value function for safe exploration [Gehring and Precup, 2013]. Related to maximizing TD-error is maximizing the variance or KL-divergence of a posterior distribution over MDPs or Q-functions, which can be used as a measure of uncertainty about rewards [Fox *et al.*, 2018; Osband *et al.*, 2018]. Posterior uncertainty over Q-functions can be used for information gain in the reward or Q-function space, but posterior uncertainty methods have thus-far largely been used for local exploration as an alternative to dithering methods such as ϵ-greedy sampling, though [Osband *et al.*, 2018] do apply posterior uncertainty to `Montezuma's Revenge` and other exploration tasks in the Atari game benchmark.

3 Preliminaries

We consider RL in the terminology of [Sutton and Barto, 1998], in which an agent seeks to maximize reward in a Markov Decision Process (MDP). An MDP consists of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a state transition function $S : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ giving the probability of moving to state s_{t+1} after taking action a_t from state s_t for discrete timesteps $t \in 0, ..., T$. Rewards are sampled from reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow R$. An RL agent has a policy $\pi(s_t, a_t) = p(a_t|s_t)$ that gives the probability of taking action a_t when in state s_t. The agent aims to learn a policy to

maximize the expectation of the time-decayed sum of reward $R_\pi(s_0) = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$ where $a_t \sim \pi(s_t, a_t)$.

A value function $V_\theta(s_t)$ with parameters θ is a function which computes $V_\theta(s_t) \approx R_\pi(s_t)$ for some policy π. Temporal difference (TD) error δ_t measures the bootstrapped error between the value function at the current timestep and the next timestep as

$$\delta_t = V_\theta(s_t) - (r(s_t, a_t \sim \pi(s_t)) + \gamma V_\theta(s_{t+1})). \quad (1)$$

A Q-function is a value function of the form $Q(s_t, a_t)$, which computes $Q(s_t, a_t) = r(s_t, a_t) + \gamma \cdot \max_{a'} Q(s_{t+1}, a')$, the expected future reward assuming the optimal action is taken at each future timestep. An approximation to this optimal Q-function Q_θ with some parameters θ may be trained using a mean squared TD-error objective $L_{Q_\theta} = ||Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \cdot \max_{a'} Q'_{\theta'}(s_{t+1}, a'))||^2$ given some target Q-function $Q'_{\theta'}$, commonly a time-delayed version of Q_θ [Mnih *et al.*, 2015]. Extracting a policy π given Q_θ amounts to computing $\text{argmax}_a Q_\theta(s_t, a)$.

4 QXplore: TD-Error as Reward Signal

4.1 TD-error Objective

We first discuss why and how TD-error can be used as an exploration signal in deep RL settings on the classes of MDPs discussed above. Many Deep RL methods maintain a value function, typically a Q function, which in off-policy settings is bootstrapped to approximate the true Q function of the optimal policy. During the course of training, this Q function will naturally contain inaccuracies such that there is nontrivial Bellman error for certain s, a, s', r tuples. Intuitively, these errors indicate that the current estimate of the Q function does not correctly model the reward dynamics of the MDP per Bellman optimality. Therefore, an exploration method that prioritizes seeking out regions of the environment where the Q-function is inaccurate could aid an off-policy method in discovering novel sources of reward and propagating those improvements through the Q function.

Given a Q function with parameters θ and δ_t we define our exploration signal for a given state-action-next-state tuple as:

$$r_{x,\theta}(s_t, a_t, s_{t+1}) = |\delta_t| = |Q_\theta(s_t, a_t) - (r_\text{E}(s_t, a_t) + \gamma \text{max}_{a'} Q'_{\theta'}(s_{t+1}, a'))| \quad (2)$$

for some extrinsic reward function r_E and target Q-function Q'_θ. Notably, we use the absolute value of the TD rather than signed TD, as this is necessary to harness network extrapolation error in sparse reward environments.

Intuitively, a policy maximizing the expected sum of r_x for a fixed Q function will sample trajectories where Q_θ does not have an accurate estimate of the future rewards it will experience. This is useful for exploration because r_x will be large not only for state-action pairs producing unexpected reward, but for all state-action pairs leading to such states, providing a denser exploration reward function and allowing for longer-range exploration.

4.2 Q_x: Learning a Q-Function to Maximize TD-error

Now that we have defined a TD-error exploration formulation, we must ask, how should we maximize it? If we treat this signal as a reward function, r_x can be used to generate a new MDP where the reward function is replaced by r_x, and thus generally can be solved via any RL algorithm. For practical purposes, we choose to train a second Q-function to maximize r_x, as allows the entire algorithm to be trained off-policy and for the two Q-functions to share replay data. Additionally, this allows us to use the original Q function Q_θ as an exploitation policy at inference time, avoiding the need to trade off between exploration and exploitation because the Q function estimates are not directly affected by r_x values.

In our formulation, which we call QXplore, we define a Q-function, $Q_{x,\phi}(s, a)$ with parameters ϕ, whose reward objective is r_x. We train $Q_{x,\phi}$ using the standard bootstrapped loss function

$$L_{Q_{x,\phi}} = ||Q_{x,\phi}(s_t, a_t) - (r_x(s_t, a_t, s_{t+1}) + \gamma \text{max}_{a'} Q'_{x,\phi'}(s_{t+1}, a'))||^2. \quad (3)$$

The two Q-functions, Q_θ and Q_x, are trained in parallel, sharing replay data so that Q_θ can learn to exploit sources of reward discovered by Q_x and so that Q_x can better predict the TD-errors of Q_θ. Since the two share data, π_{Q_x} acts as an adversarial teacher for Q_θ, sampling trajectories that produce high TD-error under Q_θ and thus provide novel information about the reward landscape. A similar adversarial sampling scheme was used to train an inverse dynamics model by [Hong *et al.*, 2019], and [Colas *et al.*, 2018] use separate goal-driven exploration and reward maximization phases

Algorithm 1 QXplore for Continuous Actions

Input: MDP S, Q-function Q_θ with target $Q'_{\theta'}$, Q_x function $Q_{x,\phi}$ with target $Q'_{x,\phi'}$, replay buffers $\mathcal{Z}_Q$ and $\mathcal{Z}_{Q_x}$, batch size B and sampling ratios $\mathcal{R}_Q$ and $\mathcal{R}_{Q_x}$, CEM policies π_Q and π_{Q_x}, time decay parameter γ, soft target update rate τ, and environments E_Q, E_{Q_x}
while not converged **do**
  Reset E_Q, E_{Q_x}
  while E_Q and E_{Q_x} are not done **do**
    Sample environments
    $\mathcal{Z}_Q \leftarrow (s, a, r, s') \sim \pi_Q | E_Q$
    $\mathcal{Z}_{Q_x} \leftarrow (s, a, r, s') \sim \pi_{Q_x} | E_{Q_x}$
    Sample minibatches for Q_θ and $Q_{x,\phi}$
    $(s_Q, a_Q, r_Q, s'_Q) \leftarrow B * \mathcal{R}_Q$ samples from $\mathcal{Z}_Q$ and $B * (1 - \mathcal{R}_Q)$ samples from $\mathcal{Z}_{Q_x}$
    $(s_{Q_x}, a_{Q_x}, r_{Q_x}, s'_{Q_x}) \leftarrow B * \mathcal{R}_{Q_x}$ samples from $\mathcal{Z}_{Q_x}$ and $B * (1 - \mathcal{R}_{Q_x})$ samples from $\mathcal{Z}_Q$
    Train
    $r_{x,\theta} \leftarrow |Q_\theta(s_{Q_x}, a_{Q_x}) - (r_{Q_x} + \gamma Q'_{\theta'}(s'_{Q_x}, \pi_Q(s'_{Q_x})))|$
    $L_Q \leftarrow ||Q_\theta(s_Q, a_Q) - (r_Q + \gamma Q'_{\theta'}(s'_Q, \pi_Q(s'_Q)))||^2$
    $L_{Q_x} \leftarrow ||Q_{x,\phi}(s_{Q_x}, a_{Q_x}) - (r_{x,\theta} + \gamma Q'_{x,\phi'}(s'_{Q_x}, \pi_{Q_x}(s'_{Q_x})))||^2$
    Update $\theta \propto L_Q$
    Update $\phi \propto L_{Q_x}$
    $\theta' \leftarrow (1 - \tau)\theta' + \tau\theta$
    $\phi' \leftarrow (1 - \tau)\phi' + \tau\phi$
  end while
end while

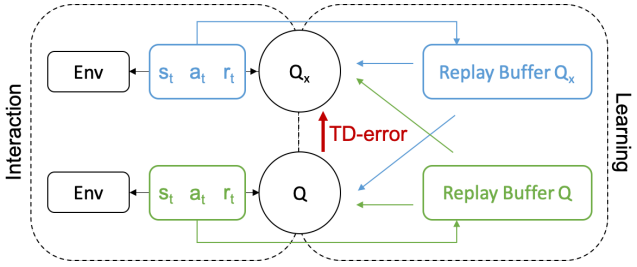


Figure 1: Method diagram for QXplore. We define two Q-functions which sample trajectories from their environment and store experiences in separate replay buffers. Q is a standard state-action value-function, whereas Q_x's reward function is the unsigned temporal difference error of the current Q on data sampled from both replay buffers. A policy defined by Q_x samples experiences that maximize the TD-error of Q, while a policy defined by Q samples experiences that maximize discounted reward from the environment.

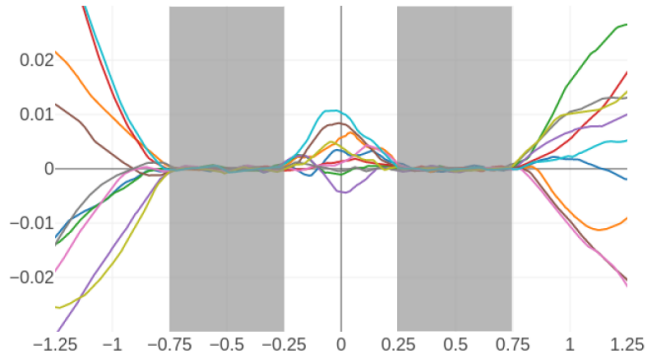


Figure 2: A neural network trained to predict a constant value does not interpolate or extrapolate well outside its training range, which can be exploited for exploration. Predictions of 3-layer MLPs of 256 hidden units per layer trained to imitate $f(x) = 0$ on $R \rightarrow R$ with training data sampled uniformly from the range $[-0.75, -0.25] \cup [0.25, 0.75]$. Each line is the final response curve of an independently trained network once its training error has converged (MSE $<$ 1e-7).

for efficient learning. However, to our knowledge adversarial sampling policies have not previously been used for exploration. To avoid off-policy stability issues due to the different reward objectives, we sample a fixed ratio of experiences collected by each policy for each training batch. Our full method is described for the continuous-action domain in Algorithm 1 and a schematic of the method is shown in Figure 1.

4.3 State Novelty from Neural Network Function Approximation Error

A key question in using TD-error for exploration is what happens when the reward landscape is flat? Theoretically, in the case that $\forall(s,a), r(s,a) = C$ for some constant $C \in R$, an optimal Q-function which generalizes perfectly to unseen states will, in the infinite time horizon case, simply output $\forall(s,a), Q^\star(s,a) = \sum_{t=0}^{\infty} C\gamma^t$. This results in a TD-error of 0 everywhere and thus no exploration signal. However, using neural network function approximation, we find that perfect generalization to unseen states-action pairs does not occur, and in fact observe in Figure 2 that the distance of a new datum from the training data manifold correlates with the magnitude of the network output's deviation from $\sum_{t=1}^{\infty} C\gamma^t$ and thus with TD-error. As a result, in the case where the reward landscape is flat TD-error exploration converges to a form of state novelty exploration. This property of neural network function approximation has been used by several previous exploration methods to good effect, including RND [Burda *et al.*, 2019] and DORA [Fox *et al.*, 2018]. In particular, the exploration signal used by RND (extrapolation error from fitting the output of a random network) should be analogous to r_x (extrapolation error from fitting a constant value), meaning we should expect to perform comparably to RND when no extrinsic reward exists.

5 Experiments

We describe here the results of experiments to demonstrate the effectiveness of QXplore on continuous control and Atari benchmark tasks. We also compare to results on `SparseHalfCheetah` from several previous publications.

Finally, we discuss several ablations to QXplore to demonstrate that all components of the method improve performance.

We compare QXplore primarily with a related state of the art state novelty-based method, RND [Burda *et al.*, 2019], and with ϵ-greedy sampling as a simple baseline. Each method is implemented in a shared code base on top of TD3/dueling double deep Q-networks for the continuous/discrete action case [Fujimoto *et al.*, 2018; Wang *et al.*, 2016]. For experiments in continuous control environments, we implement and use a nonparametric cross-entropy method policy, previously described as more robust to hyperparameter variance, with the same architecture and hyperparameters as prior work [Simmons-Edler *et al.*, 2019; Kalashnikov *et al.*, 2018]. We experimented with a variant using DDPG-style parametric policies [Lillicrap *et al.*, 2015] for both Q_θ and Q_x , but found preventing Q_θ's policy from converging to poor local maxima difficult, consistent with previously reported stability issues in that class of algorithms [Simmons-Edler *et al.*, 2019; Islam *et al.*, 2017]. For all experiments, we set the data sampling ratios of Q_θ and Q_x, $\mathcal{R}_Q$ and $\mathcal{R}_{Q_x}$ respectively, at 0.75, the best ratio among a sweep of ratios 0.0, 0.25, 0.5, and 0.75 on `SparseHalfCheetah`. For continuous control tasks, we used a learning rate of 0.0001 for both Q-functions, the best among all paired combinations of 0.01, 0.001, and 0.0001, and fully-connected networks of two hidden layers of 256 neurons to represent each Q-function, with no shared parameters. For Atari benchmark tasks, we used the dueling double deep Q-networks architecture and hyperparameters described by Wang *et al.*.

5.1 Experimental Setup

We benchmark on five continuous control tasks using the MuJoCo physics simulator that each require exploration due to sparse or unshaped rewards. First, the `SparseHalfCheetah` task originally proposed by VIME [Houthooft *et al.*, 2016]. This task requires an agent to move 5 units (several hundred

Episodes until mean reward of	QXplore	VIME	EX2	EMI
50	3000	10000*	4740*	**2580***
100	**3400**	x*	6180*	4520*
200	**4000**	x*	x*	8440*
300	**10000**	x*	x*	x*

Table 1: Number of episodes required to reach mean reward milestones on `SparseHalfCheetah` for several methods. QXplore reaches higher rewards than previously published results. Results marked with "*" are previously published numbers. Results marked with "x" indicate that the mean reward was not achieved.

timesteps of actions) forward to receive reward, receiving 0 reward otherwise, and is maze-like in this regard. Next, we benchmark on three goal-directed OpenAI gym tasks, `FetchPush`, `FetchSlide` and `FetchPickAndPlace`, originally proposed in HER [Andrychowicz *et al.*, 2017]. Lastly, we test a variant of `SparseHalfCheetah` that we refer to as `LocalMaxEscape` where a local reward maximum has been introduced — the agent receives 0 reward for every timestep it is between -1 and 1 units from the origin, -1 reward if it moves outside that range, but 100 reward per timestep if it moves 5 units forward, similar to `SparseHalfCheetah`. We chose these tasks as they are challenging exploration problems highlighting the different cases we are interested in that are relatively simple to control, but still involve large continuous state spaces and continuous actions. Guided by a recent study suggesting that exploration in the Atari game benchmark suite doesn't improve performance on most tasks [Taiga *et al.*, 2020], we evaluated on a pair of "hard" exploration games, `Venture` and `Gravitar`, as well as an easy game, `Pong`, to show that QXplore can also function in this very different domain. We ran five random seeds for each experiment and plot the mean and plus/minus 1 standard deviation bounds for each set of runs, applying a Gaussian filter to each mean/stdev for readability.

5.2 Exploration Benchmark Performance

We show the performance of each method on each task in Figure 3. QXplore performs comparable to RND on the `SparseHalfCheetah` task, in line with our expectation, but performs much better comparatively on the `Fetch` tasks — only on `FetchPush`, the easiest task, did RND find non-random reward. We believe this is because TD-error drives exploration behavior that helps the agent to uncover the goal-conditioning relationship, whereas state novelty is goal-agnostic and does not aid in discovery of the relationship. QXplore also strongly outperformed RND on `LocalMaxEscape`, as negative rewards far from the origin increase TD error and drive rapid discovery of the global optimum.

To validate our performance and sample efficiency, we compare QXplore to previously published `SparseHalfCheetah` performance numbers in Table 1. Because to our knowledge no previous work has evaluated off-policy Q-learning based methods on `SparseHalfCheetah`, we compare to previous methods built on top of TRPO [Schulman *et al.*, 2015]. Due to the difference in baseline algorithms, we compare the number of episodes of interaction required to reach a given level of reward, though QXplore was not intended to be performant with respect to this metric. While some decrease in sample efficiency is expected due to differing baseline methods (TRPO [Schulman *et al.*, 2015] versus TD3 [Fujimoto *et al.*, 2018]), compared to the results reported by Kim *et al.* for EMI [Kim *et al.*, 2019] and EX2 [Fu *et al.*, 2017], and by Houthooft *et al.* for VIME [Houthooft *et al.*, 2016] on the `SparseHalfCheetah` task, QXplore reaches almost every reward milestone faster, and achieves a peak reward (300) not achieved by any previous method. This shows that off-policy Q-learning combined with TD-error exploration can result in sample efficient as well as flexible exploration.

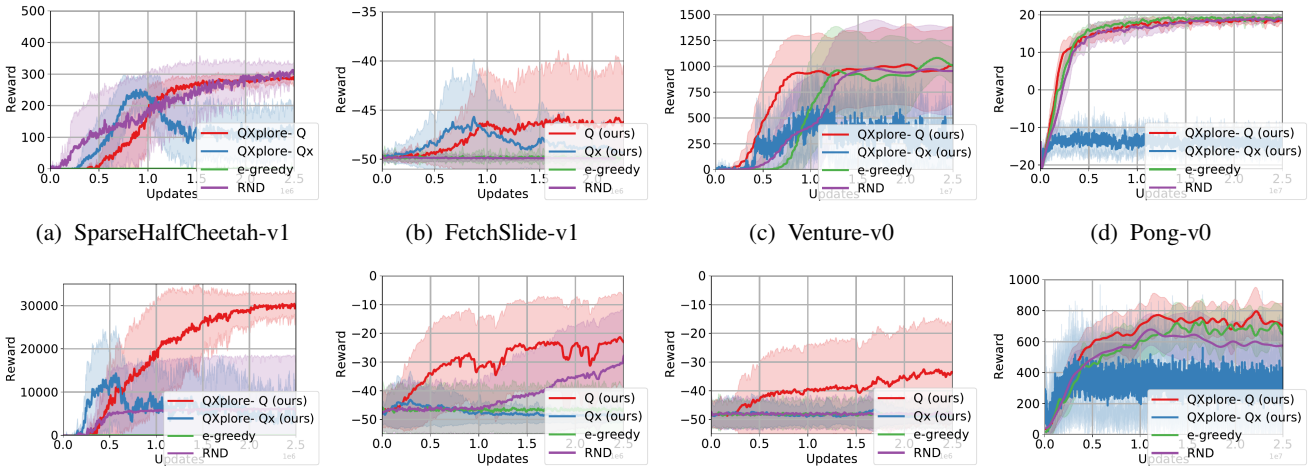
We also implemented a continuous-control adaptation of DORA [Fox *et al.*, 2018] and tested it on `SparseHalfCheetah`. DORA performed poorly, possibly because it was not intended for use with continuous action spaces, and thus we did not test it on other tasks.

As a comparison to a published off-policy Q-learning exploration method, we compared to GEP-PG [Colas *et al.*, 2018], which used separate exploration and exploitation phases similar to QXplore. We downloaded the author's implementation (built on top of DDPG) and tested it on `SparseHalfCheetah` using the parameters for the `HalfCheetah-v2` task it was originally tested on. GEP-PG reached a validation reward of 120.2 after 4000 episodes, broadly comparable to our QXplore and RND implementations.

Finally, while the main focus of our evaluation is on continuous control tasks, we also evaluated QXplore on several games in the Atari Arcade Learning Environment [Bellemare *et al.*, 2013] to verify that QXplore extends to tasks with image observations and discrete action spaces. We implemented QXplore and RND on top of dueling double DQN [Wang *et al.*, 2016], using hyperparameters and network architectures from the Dopamine implementation of DQN [Castro *et al.*, 2018]. We show the results in Figure 3 after 25 million training steps. Based on the findings of [Taiga *et al.*, 2020], we did not expect to improve significantly compared to the baseline in this domain. Indeed, we find that QXplore performs comparably to the baseline and RND implementations on `Pong` and `Gravitar`, while outperforming them modestly on `Venture`, where QXplore converges faster, perhaps due to Q_x focusing on reward-adjacent states more than ϵ-greedy or RND.

5.3 Ablations

There are two major features of QXplore that distinguish it from prior work in exploration: the use of a pair of policies that share replay data, and the use of unsigned TD-error to drive exploration. We performed several ablations that assess the contribution of each of these features to our method and confirm their value for exploration. We show the results in Figure 4. We find that the use of separate exploration and exploitation policies along with unsigned TD-error is necessary to obtain good performance, and that ablations of these components either fail to train or substantially reduce performance. We discuss each case in detail below.



(a) SparseHalfCheetah-v1 (b) FetchSlide-v1 (c) Venture-v0 (d) Pong-v0

(e) LocalMaxEscape-v1 (f) FetchPush-v1 (g) FetchPickAndPlace-v1 (h) Gravitar-v0

Figure 3: Performance of QXplore compared with RND and ϵ-greedy sampling. QXplore outperforms RND and ϵ-greedy on the Fetch tasks and in escaping local maxima, while performing comparably on maze solving tasks and non-exploration tasks. "QXplore-Q" indicates the performance of our exploitation Q-function, while "QXplore-Qx" indicates the performance of our exploration Q-function, whose objective does not directly maximize reward but which may lead to high reward regardless.

Single-Policy QXplore. First, we tested a single-policy version of QXplore by replacing $Q_\theta(s, a)$ with a value function $V_\theta(s)$. We use a value function rather than Q-function in this case to avoid large estimation errors stemming from fully off-policy training. We observe in Figure 4 that while the policy is able to find reward quickly and converge faster, the need to satisfy both objectives results in a lower converged reward than the original QXplore method.

1-Step Reward Prediction. Second, we ran an ablation where we replace $Q_\theta(s, a)$ with a function that simply predicts the current $r(s_t, a_t)$. Using reward error instead of a value function in Q_x can still produce the same state novelty fallback behavior in the absence of reward; however, it provides only limited reward-based exploration utility. We evaluate this variant and observe in Figure 4 that it fails to sample reward. Reward prediction error is not sufficient to allow strong exploration behavior without some form of lookahead.

QXplore with State Novelty Exploration. To assess the importance of TD-error specifically in our algorithm, we replaced the TD-error maximization objective of Q_x with the random network prediction error maximization objective of RND, while still performing rollouts of both policies. The results are shown in Figure 4. We observe that while the modified Q_x samples reward, it is too infrequent to guide Q to learn the task. Qualitatively, the modified Q_x function does not display the directional preference in exploration that normal Q_x does once reward is discovered, instead sampling both directions equally.

QXplore with Signed TD-Error Objective. While we used unsigned TD-error to train Q_x, we also tested QXplore using signed TD-error. We used the negative signed TD-error $-\delta_t$ from equation 1 so that better-than-expected rewards result in positive r_x values. The results of this experiment are shown in Figure 4. While this ablation is able to converge

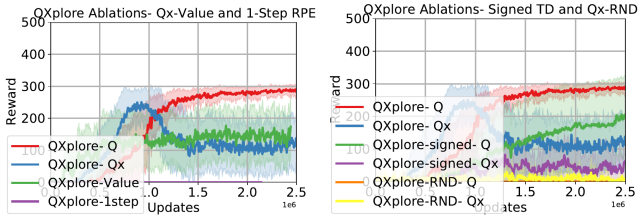


Figure 4: Plots showing several ablations of QXplore on `SparseHalfCheetah`. While several variants are able to learn the task, the full QXplore formulation performs better.

and solve the task, the unsigned TD-error performs much better on `SparseHalfCheetah`, likely due to the extrapolation error described in Figure 2 being both positive and negative.

5.4 Qualitative Behavioral Analysis

Qualitatively, on `SparseHalfCheetah` we observe interesting behavior from Q_x late in training. After initially converging to obtain high reward, Q_x appears to get "bored" and will focus on the reward threshold, stopping short or jumping back and forth across it, which results in reduced reward but higher TD-error. This behavior is distinctive of TD-error seeking over state novelty seeking, as such states are not novel compared to moving past the threshold but do result in higher TD-error. Such behavior from Q_x motivates Q to sample the state space around the reward boundary and thus learn to solve the task. Example sequences of such behaviors are shown in Figure 5.

6 Discussion and Conclusions

Here, we have proposed the use of reward prediction error as an objective for exploration in deep reinforcement learn-

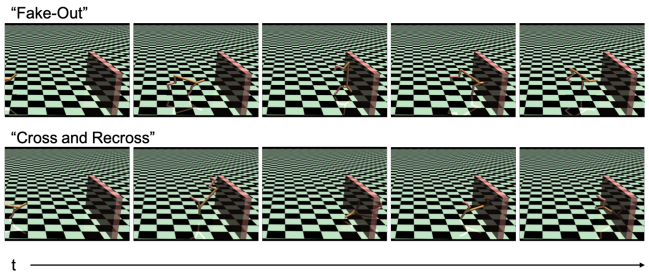


Figure 5: Example trajectories showing Q_x's behavior late in training that is distinctive of TD-error maximization. The corresponding Q network reliably achieves reward at this point. In "fake-out", Q_x approaches the reward threshold and suddenly stops itself. In "cross and recross", Q_x crosses the reward threshold going forward and then goes backwards through the threshold.

ing. We defined a deep RL algorithm, QXplore, using TD-error that is sufficient to discover solutions to multiple types of challenging exploration tasks across multiple domains. We found that QXplore performs well across all exploration task types tested compared to our state novelty baseline, although type-specific algorithms can likely perform better on some types, such as goal-directed exploration.

While QXplore is a general-purpose exploration algorithm that can be applied successfully to many tasks, several limitations remain for TD-error exploration. In the worst-case, TD-error likely performs no better than state novelty for certain "pure" exploration tasks, such as exploring a linear chain of states, though with an optimistic prior on the Q-values of unseen states it may perform comparably to state novelty. There also exist adversarial tasks where the unsigned TD-error leads to less efficient exploration compared to other possible policies, such as a task with many states that yield large negative rewards uncorrelated with positive rewards. Combining TD-error exploration with reward exploitation may help in such cases to bias the search. However, balancing the rate at which the TD-error signal disappears for a given state with the reward function's magnitude is critical to get rapid convergence, and more research into such approaches is needed. Lastly, TD-error maximization may result in "risky" exploration (in contrast to the "safe" TD-minimizing exploration of Gehring and Precup) and thus may not be well suited for tasks where failures or negative returns have real-world consequences without additional constraints on the agent's actions, or the use of signed TD-error to avoid trajectories yielding worse-than-expected returns.

We hope that our results can spur more investigation into TD-error-based exploration methods to address some of the outstanding challenges described above, as well as encourage further work on diverse exploration signals in RL and on more general exploration objectives suitable for use on heterogeneous RL tasks.

References

[Andrychowicz *et al.*, 2017] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NIPS*, 2017.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

[Bellemare *et al.*, 2016] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, pages 1471–1479, 2016.

[Burda *et al.*, 2019] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *ICLR*, 2019.

[Castro *et al.*, 2018] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018.

[Colas *et al.*, 2018] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018.

[Fox *et al.*, 2018] Lior Fox, Leshem Choshen, and Yonatan Loewenstein. {DORA} The Explorer: Directed Outreaching Reinforcement Action-Selection. In *ICLR*, 2018.

[Fu *et al.*, 2017] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *NIPS*, 2017.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

[Gehring and Precup, 2013] Clement Gehring and Doina Precup. Smart Exploration in Reinforcement Learning using Absolute Temporal Difference Errors. In *AAMAS*, 2013.

[Hong *et al.*, 2019] Zhang-Wei Hong, Tsu-Jui Fu, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. Adversarial exploration strategy for self-supervised imitation learning. In *CoRL*, 2019.

[Houthooft *et al.*, 2016] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

[Islam *et al.*, 2017] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *CoRR*, abs/1708.0, 2017.

[Kalashnikov *et al.*, 2018] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, 2018.

[Kim *et al.*, 2019] Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. Emi: Exploration with mutual information. In *ICML*, 2019.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning: Deep Deterministic Policy Gradients (DDPG). *ICLR*, 2015.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K 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. *Nature*, 518(7540):529–33, feb 2015.

[Osband *et al.*, 2018] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *NIPS*, 2018.

[Pathak *et al.*, 2017] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[Schmidhuber, 1991] Jürgen Schmidhuber. Adaptive confidence and adaptive curiosity. In *Institut fur Informatik, Technische Universitat Munchen, Arcisstr. 21, 800 Munchen 2*. Citeseer, 1991.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

[Simmons-Edler *et al.*, 2019] Riley Simmons-Edler, Ben Eisner, Eric Mitchell, Sebastian Seung, and Daniel Lee. Q-learning for continuous actions with cross-entropy guided policies. *arXiv preprint arXiv:1903.10605*, 2019.

[Stadie *et al.*, 2015] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models. *CoRR*, abs/1507.0, 2015.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. {*IEEE*} *Trans. Neural Networks*, 9(5):1054, 1998.

[Taiga *et al.*, 2020] Adrien Ali Taiga, William Fedus, Marlos C. Machado, Aaron Courville, and Marc G. Bellemare. On bonus based exploration methods in the arcade learning environment. In *ICLR*, 2020.

[Tang *et al.*, 2017] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, 2017.

[Thrun and Möller, 1992] Sebastian B Thrun and Knut Möller. Active exploration in dynamic environments. In *NIPS*, 1992.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.