Rethinking Exploration for Sample-Efficient Policy Learning

William F. Whitney*[1], Michael Bloesch[2], Jost Tobias Springenberg[2], Abbas Abdolmaleki[2], and Martin Riedmiller[2]

[1]Department of Computer Science, New York University
[2]DeepMind, London

Abstract

Off-policy reinforcement learning for control has made great strides in terms of performance and sample efficiency. We suggest that for many tasks the sample efficiency of modern methods is now limited by the richness of the data collected rather than the difficulty of policy fitting. We examine the reasons that directed exploration methods in the bonus-based exploration (BBE) family have not been more influential in the sample efficient control problem. Three issues have limited the applicability of BBE: bias with finite samples, slow adaptation to decaying bonuses, and lack of optimism on unseen transitions. We propose modifications to the bonus-based exploration recipe to address each of these limitations. The resulting algorithm, which we call UFO, produces policies that are Unbiased with finite samples, Fast-adapting as the exploration bonus changes, and Optimistic with respect to new transitions. We include experiments showing that rapid directed exploration is a promising direction to improve sample efficiency for control.

1. Introduction

Recent progress in reinforcement learning (RL) for continuous control has led to significant improvements in sample efficiency and performance. While earlier on-policy algorithms required hundreds of millions of environment steps to solve simple tasks (Mnih et al., 2016; Schulman et al., 2015, 2017), recent off-policy algorithms have brought the sample efficiency of model-free control within range of solving tasks on real robots (Popov et al., 2017; Kalashnikov et al., 2018; Haarnoja et al., 2018a; Fujimoto et al., 2018; Haarnoja et al., 2018b; Abdolmaleki et al., 2018).

The fundamental aspect allowing this improvement lies in the difference between on-policy and off-policy methods. I.e. for every policy update, on-policy algorithms require enough new samples from the environment to make each update. By contrast, off-policy algorithms decouple data collection from policy updates: they can use data that was collected by a previous version of the policy, or even a separate exploration policy. This means that an off-policy update only requires that enough useful data has been collected from the environment *in total*.

In this paper, we suggest that with this ability to reuse data, the sample efficiency of modern off-policy RL algorithms is now limited by the richness of the data they collect rather than by the absolute number of samples. This places a heavy emphasis on the speed with which this rich data can be collected by the behavior policy. This is the role of exploration, which has been studied extensively in deep reinforcement learning (Stadie et al., 2015; Osband et al., 2016; Houthooft et al., 2016; Pathak et al., 2017; Tang et al., 2017; Burda et al., 2018; Fortunato et al., 2018; Plappert et al., 2018; Osband et al., 2019; Badia et al., 2020; Machado et al., 2020; Rashid et al., 2020; Dean et al., 2020). However, existing techniques have largely failed to influence deep RL outside the domain of so-called "hard exploration problems." Indeed, across commonly used benchmarks these methods often yield no improvements over ε-greedy (Taiga et al., 2020).

∗. Work done during an internship at DeepMind.

Much of the prior work on exploration falls into the bonus-based exploration (BBE) family of algorithms, where a standard RL algorithm is used to learn a policy for a Markov decision process (MDP) augmented with an exploration bonus. The policy attempts to maximize the sum of the environment rewards and the exploration bonus, while the exploration bonus decreases over the course of training. For the sample-efficient control setting, the standard BBE recipe for exploration has three main limitations: (1) it optimizes a single policy on a sum of exploration bonus and task rewards, leading to a policy which is biased after any finite number of samples; (2) it fails to adapt its policy and rewards within a single episode, leading to wasted samples; and (3) it provides no mechanism to encourage the agent to take transitions it has never seen.

We propose to address these issues with three modifications to the BBE recipe. First, we separate the learning of an unbiased task policy, which always estimates the optimal policy on the true task of interest, from an exploration policy, which optimizes the exploration objective. This removes the bias from the task policy, allowing it to perform well even before the exploration rewards go to zero. Second, we note that the exploration MDP is, by necessity, non-stationary; because that the reward for a particular transition (s, a, s') decreases on successive visits. Therefore to maximize the exploration reward within an episode (and thus also the amount of exploration), we propose that the exploration policy should be fast-updating, ideally optimized to convergence after every environment step. This allows the exploration policy to visit an unseen state, mark it as seen, and proceed to the next unseen state, rather than getting stuck on the first novel state it observes. Third, we use an optimistic update to incentivize the agent to reach and take transitions which it has never seen before. Taken together, these modifications to BBE yield a new algorithm for exploration, which we call UFO: unbiased, fast-adapting, optimistic.

We provide experiments on simple problems using a count-based exploration reward. Our results show that these improvements result in faster coverage of the state space and better sample efficiency for policy learning. We view scaling up our approach to real-world control tasks as a very promising direction for future work.

2. Background

2.1 Notation

A Markov decision process (MDP) $\mathcal{M}$ consists of a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition function mapping $\mathcal{S} \times \mathcal{A}$ to distributions on $\mathcal{S}$, R is the scalar reward function on $\mathcal{S} \times \mathcal{A}$, and γ is the discount factor. We use lower-case (s, a, r) to refer to concrete realizations of states, actions, and rewards. We use $\mathcal{M}_f$ to denote the MDP $\mathcal{M}$ with the original reward function R replaced by another function f. On a discrete state space, we define the function $N(s)$ to be the number of times the agent has visited the state s at a particular point in training. Similarly for discrete action spaces we define $N(s, a)$ to be the number of times that the policy has taken action a while in state s. For convenience, we assume exploration rewards are within $[0, 1]$, and we define $\bar{r} = 1/1-\gamma$, which is the maximum discounted value possible.

2.2 Bonus-based exploration: a recipe for exploration in deep RL

Bonus-based exploration has emerged as the standard framework for exploration in the deep reinforcement learning community. In this framework, an agent learns in a sequence of MDPs $\widetilde{\mathcal{M}} = \{\mathcal{M}_{\widetilde{R}_n}\}_{n=1}^{N}$ where the reward function $\widetilde{R}_n$ changes as a function of each transition. A typical choice is $\widetilde{R}_n = R + R_n^+$, where R_n^+ is an exploration bonus which measures the "novelty" of a transition (s, a, s') given the history of all previous transitions. After taking each transition (s, a, s'), the reward $\widetilde{r} = \widetilde{R}_n(s, a, s')$ is calculated and the tuple $(s, a, s', \widetilde{r})$ is added to a replay dataset D. The agent optimizes its reward in this (non-stationary) MDP $\widetilde{\mathcal{M}}$ via some model-free RL algorithm operating on the replay dataset. The realization of a particular algorithm in this family amounts to defining a novelty function and

Algorithm 1 Bonus-based exploration

Require: replay dataset D, policy π

$n \leftarrow 0$
repeat
  for one episode **do**
    Collect $(s, a, s', r) \sim P(s, \pi(s))$
    $\widetilde{r} \leftarrow r + R_n^+(s, a, s')$
    $D \leftarrow D \cup (s, a, s', \widetilde{r})$
    $n \leftarrow n + 1$
  Train π with samples from D
until convergence

Algorithm 2 UFO exploration

Require: replay dataset D, temperature τ
Require: policy π, exploration value Q_{explore}
$n \leftarrow 0$
repeat
  for one episode **do**
    Update Q_{explore} optimistically on $\mathcal{M}_{R_n^+}$
    Set optimistic $\beta(a|s)$ by Eq. (2)
    Collect $(s, a, s', r) \sim P(s, \beta(s))$
    $D \leftarrow D \cup (s, a, s', r)$
    $n \leftarrow n + 1$
  Train π with samples from D
until $n = N$

Figure 1: Comparison of classic bonus-based exploration (BBE) with our method (UFO). BBE computes exploration bonuses at the time of visiting a transition, adds them to the real rewards, and uses a replay buffer of experience to learn a policy. UFO separates the exploration policy β from the task policy π, allowing π to be an unbiased estimate of the optimal policy throughout training. It always uses the *current* exploration function R_n^+ when updating the exploration value function, and is fast-adapting to deal with the non-stationary bonus MDP. Finally UFO's behavior policy actions and updates are optimistic, ensuring the agent will reach and take transitions it has never seen.

picking a model-free RL algorithm (Stadie et al., 2015; Houthooft et al., 2016; Bellemare et al., 2016; Pathak et al., 2017; Tang et al., 2017; Burda et al., 2018; Machado et al., 2020). We illustrate this recipe in Algorithm 1.

2.3 Count-based exploration

Classic work on exploration, including methods such as E3 (Kearns and Singh, 1998), R-MAX (Brafman and Tennenholtz, 2002), and MBIE (Strehl and Littman, 2008) leverage the count function N to achieve provably fast policy learning. For our experiments, we build on the method of model-based interval estimation with exploration bonuses (MBIE-EB) (Strehl and Littman, 2008) and its spiritual descendants (Bellemare et al., 2016). MBIE-EB is an bonus-based exploration algorithm which uses a bonus

$$R^+(s, a) = \frac{\alpha}{\sqrt{N(s, a)}} \tag{1}$$

to achieve provably efficient policy learning in the tabular case for a suitable setting of α. Count-based rewards have the downside that they encourage the agent to visit every state. Our proposed algorithm is not specific to count-based rewards, and any alternative (e.g. forward prediction error (Pathak et al., 2017) or random network distillation (Burda et al., 2018)) could be used in its place.

3. Limitations of bonus-based exploration

The bonus-based exploration algorithm, illustrated in Algorithm 1, has three weaknesses which limit its usefulness for sample-efficient policy learning:

Bias with finite samples. Because they estimate the optimal policy on the modified MDP $\mathcal{M}'$, bonus-based exploration algorithms learn biased policies as long as the exploration bonus is nonzero. According to theory, the exploration bonus should be scaled very large (Strehl and Littman, 2008)

and decay slower than $1/N(s)$ (Kolter and Ng, 2009) in order to guarantee convergence to the optimal policy. This results in substantially biased policies after any feasible number of samples.

Limited within-episode adaptation. Algorithms in this family update the policy according to the schedule of the underlying model-free RL algorithm – for example at the end of each episode. This works well for the stationary MDPs that these algorithms were developed for, but the modified MDP $\mathcal{M}'$ which represents the exploration problem is non-stationary. This leads to an agent which determines the most novel state and then stays there for an entire episode. This degenerate behavior leads to potentially exploring only a single state per episode instead of visiting a sequence of new states as the reward function evolves.[1] The use of replay buffers compounds this effect, since algorithms in this family compute exploration rewards at the time the transition is collected, rather than when it is used. An algorithm which is unaware of the non-stationary nature of the MDP will maximize the return on this mixture of reward functions rather than the reward that incorporates the current bonus.

No optimism on unseen transitions. Bonus-based exploration algorithms are fundamentally about optimism in the face of uncertainty; providing a bonus on any transition that led to a novel or uncertain outcome in the past. However, they do not update the policy on transitions which have *never* been seen. As a result, they cannot directly cause an agent to take an action for the very first time! This is a significant failing in an exploration algorithm, and may explain the observation by Dabney et al. (2020) that many sophisticated exploration algorithms in the deep RL literature still rely on ε-greedy actions. This phenomenon is explored in depth by Rashid et al. (2020).

4. Maximizing sample efficiency with exploration

In this section, we describe a new algorithm called unbiased, fast-adapting, optimistic exploration, or UFO. It consists of modifications to BBE which address the limitations discussed above. Like BBE, UFO is a family of algorithms related by their structure; a particular algorithm in this family consists of a choice of an exploration reward function and an off-policy RL algorithm for the task policy. The modifications that define UFO are as follows:

1. Separately represent the current best estimate of the optimal policy for the task π_{task} and the exploration value function Q_{explore}, thereby avoiding bias in the task policy.
2. Update the exploration policy (and thus the behavior policy) aggressively on the current version of the bonus MDP after every timestep, enabling deep exploration of multiple novel locations within a single episode.
3. Impose an optimistic prior on Q_{explore} by leveraging a count function, allowing the agent to seek out transitions which it has never taken.

In this section we discuss the details of each component.

4.1 Unbiased task policy

We propose to separately learn a *task policy* π_{task}, which approximates the optimal policy for the original MDP $\mathcal{M}$, and an *exploration value function* Q_{explore}, which estimates the value of the behavior policy on the bonus MDP $\mathcal{M}_{R^+}$ only (i.e. not including the task reward). These two functions are learned separately on transitions from the same replay buffer.

1. Some implementations of bonus-based exploration may update the policy within an episode, for example via a single gradient step per environment step on transitions sampled i.i.d. from a replay. However, such a small update is typically not enough to change the qualitative behavior of the agent and adapting to the changing MDP has not been an emphasis in prior work.

The task policy for the original MDP is trained in an ordinary off-policy fashion using the replay buffer. Since it is trained only on the rewards for the true task, it is unbiased in the sense that it reflects the current best estimate of the optimal action in each state. This stands in contrast to BBE policies, which optimize the sum of task and exploration rewards, and thus represent a biased estimate of the optimal task policy until the exploration rewards go to zero. Our method is agnostic to the choice of algorithm and policy parameterization; however, it will be most effective with policy learning algorithms that work well when trained very off-policy and produce well-calibrated policy distributions.

4.1.1 Behavior policy

A good behavior policy should attempt to explore all of the transitions which are relevant for learning the optimal policy. This entails a trade-off between taking actions which are more novel and ones which are more likely to be relevant to a high-performing policy. We encode this by representing the behavior policy as a product of the task policy distribution π and a pure-exploration Boltzmann distribution:

$$\beta(a|s) \propto \pi_{\text{task}}(a|s) \exp\left\{\frac{Q^{+}_{\text{explore}}(s,a)}{\tau_{\text{explore}}}\right\} \tag{2}$$

where τ_{explore} is a temperature hyperparameter controlling the aggressiveness of the exploration and Q^{+}_{explore} is an optimistic version of the exploration value function (see Section 4.3).

The choice to parameterize β as a factored policy was made for its simplicity and ease of off-policy learning. Alternative formulations for making this trade-off while preserving the unbiased task policy are possible, and we view the form of our proposed behavior policy as just one option among many. One simple alternative would be Scheduled Auxiliary Control (SAC-X) (Riedmiller et al., 2018), which interleaves the behavior of multiple policies in time.

4.2 Fast-adapting exploration value function

Traditional bonus-based exploration methods update the policy using a replay buffer which, at a step n, contains rewards from mixture of bonus reward functions $\{R^{+}_{1}, \ldots, R^{+}_{n}\}$, computed using different past novelty or count estimates.[2] This results in slow adaptation to the non-stationary objective of exploration. We propose to learn a separate state-action value function $Q_{\text{explore}}(s,a)$ which approximates the value of (s,a) for the behavior policy on the *current* bonus MDP $\mathcal{M}_{R^{+}_{n}}$. The non-stationarity of the bonus MDP means that after every timestep, Q_{explore} will be out of date. This is especially problematic because the rewards will change the most in precisely the part of state-space that the agent is currently in.

To illustrate this phenomenon, imagine an agent trying to solve a maze. Suppose the agent has just reached a dead end that it has never seen before. For the first timestep at that new state, it might receive a large exploration bonus for doing something new. However, if it stays there, it will not learn anything new or visit any other new states, and the reward it earns will decrease with every additional timestep in the dead end. Ideally the agent should update its value estimate for this state immediately, and as soon as this is no longer the most novel state in the maze, it should leave and go somewhere new.

We implement this form of fast adaptation as simply as possible: by performing many updates to Q_{explore} at every timestep using a large learning rate. However, this poses its own problems; most significantly, Q-learning with function approximation has a tendency to diverge if updated too aggressively with too few new samples. We use two modifications to the typical Bellman update with target networks (Mnih et al., 2015) to mitigate this issue.

2. See e.g. the code from Machado et al. (2020): https://github.com/mcmachado/count_based_exploration_sr/blob/master/function_approximation/exp_eig_sr/train.py#L204

- **Soft DoubleDQN update.** The DoubleDQN (Hasselt et al., 2016) update reduces overestimation in Q-learning by selecting and evaluating actions using different parameters. We use a soft version of the DoubleDQN update by replacing the max operator with the expectation over actions sampled from the current behavior policy (which is a function of Q_{explore}).
- **Value clipping.** To further mitigate the problem of Q-learning overestimation and divergence, we clip the Bellman targets to be within the range of possible Q values for $\mathcal{M}_{R_n^+}$. Given that the rewards r^+ are scaled to be in $[0, 1]$, any policy would have a value $Q_{\text{explore}}(s, a) \in [0, \bar{r}]$, where $\bar{r} = 1/1-\gamma$.

With these two changes, the target for estimating the value of the policy β on the MDP $\mathcal{M}_{R_n^+}$ is

$$y(s, a, s') = \text{clip}\left(R_n^+(s, a) + \gamma \mathop{\mathbb{E}}_{a' \sim \beta(\cdot|s';\theta)}\left[Q_{\text{explore}}(s', a'; \theta^-)\right], 0, \bar{r}\right). \tag{3}$$

where $R_n^+(s, a)$ is the current exploration bonus, which we recompute at update time; $Q_{\text{explore}}(s', a'; \theta^-)$ is the exploration value function target network; and $\beta(\cdot \mid s'; \theta)$ is the behavior policy calculated with the current exploration value function parameters. We then minimize the squared error between $y(s, a, s')$ and $Q_{\text{explore}}(s, a)$.

4.3 Optimistic exploration value function

We propose to make Q_{explore} optimistic by leveraging the count function in a manner similar to that proposed by Rashid et al. (2020). We assume that the value function is trustworthy for transitions with very large counts, and very untrustworthy for transitions with near-zero counts. When the count is zero we impose an optimistic prior which assumes the transition will lead to a whole episode of novel transitions; as the count increases we interpolate between this prior and the learned value function using a weighting function:

$$Q^+_{\text{explore}}(s, a) = w(s, a)Q_{\text{explore}}(s, a) + (1 - w(s, a))\bar{r} \tag{4}$$

where $\bar{r} = 1/1-\gamma$, again, is the maximum discounted return in the bonus MDP.

The weights are calculated from the counts:

$$w(s, a) = \frac{\sqrt{N(s, a)}}{\sqrt{N(s, a) + c}} \tag{5}$$

with c being the *prior count* which represents how many visits' worth of confidence we ascribe to the optimistic prior. We use this optimistic Q^+_{explore} to select actions, and in place of Q_{explore} for the Bellman updates described in Equation (3).

5. Related work

Bonus-based exploration. In the last few years there have been many bonuses proposed in the bonus-based exploration framework. Stadie et al. (2015), Pathak et al. (2017), and Burda et al. (2018) propose to use prediction error of a learned model to measure the novelty of new states, with the key differences being the state representation used for making predictions. Stadie et al. (2015) make predictions in the raw observation space, while Pathak et al. (2017) use a learned embedding from an inverse dynamics model, and Burda et al. (2018) use a randomly-initialized neural network to embed the states. Houthooft et al. (2016) propose to measure how informative a new observation is by using a variational formulation of the information gain of the policy. Bellemare et al. (2016) and Ostrovski et al. (2017) define a continuous analogue of a count function which they call pseudocounts, then use them to calculate the count-based bonuses of Strehl and Littman (2008). Tang et al. (2017) also use

the bonus formula from Strehl and Littman (2008), but compute counts using a locality-sensitive hashing function to bin the states. Machado et al. (2020) use the norm of learned successor features as a bonus, and show that it implicitly counts state visits. This work focuses on the updates and representation of the behavior policy, and our proposed algorithm can be used in conjunction with any of these bonuses from the literature. The Never Give Up (NGU) algorithm (Badia et al., 2020) uses an exploration bonus with an episodic structure, encouraging a behavior policy to visit each state infinitely often while rapidly adapting the reward within each episode. Instead of a single policy, NGU learns an ensemble of policies with different scales of the exploration bonus, allowing it to act greedily with respect to a task policy. However, NGU is designed to maximize asymptotic performance rather than sample efficiency, and indeed does not learn faster than a baseline early in training.

Optimism. Classic exploration methods (Kearns and Singh, 1998; Brafman and Tennenholtz, 2002; Strehl and Littman, 2008; Jaksch et al., 2008), depend on an optimistically-defined model. Model-free methods with theoretical guarantees, including Strehl et al. (2006) and Jin et al. (2018) similarly use optimistically-initialized Q functions. Optimism is difficult to ensure with neural networks, since even an optimistic initialization will rapidly wash out due to generalization from other observed points. Recently Rashid et al. (2020) proposed a novel method for ensuring optimism in Q learning with function approximation by leveraging a count function. Our method uses a similar formulation for learning an optimistic exploration Q function. However, by separating the exploration policy from the task policy, our method leaves the task policy unbiased in the few-sample regime.

Temporally-extended actions. While undirected exploration methods such as ϵ-greedy explore large spaces only slowly, the use of temporally-extended actions which result in less dithering can explore faster. Schoknecht and Riedmiller (2003) propose to augment the action space with multi-step actions, which consist of repeating a primitive action for a specified duration, and show that this reduces the first passage time for reaching faraway states on a random walk. Neunert et al. (2020) propose a method for learning policies in hybrid continuous-discrete action spaces and allow the policy to choose to repeat the given action for multiple steps, resulting in faster exploration without limiting the expressivity of the policy. Dabney et al. (2020) describe a temporally-extended version of ϵ-greedy exploration which, for a fraction $1 - \epsilon$ of transitions, samples a random action to take as well as a random *duration* for that action. They then train off-policy RL on the collected data. Whitney et al. (2020) use a learned k-step temporally-extended action space, where each point in the space represents a state reachable in k timesteps and a sequence of actions to get to that state, and show that uniform exploration in this space reaches faraway states with higher probability. While these methods improve significantly over the single-step ϵ-greedy method, they are unable to perform directed exploration and rapidly reach faraway unvisited states.

Randomized value functions. Thompson sampling is an appealing formulation of exploration that samples from the posterior distribution over value functions and then explores greedily (Thompson, 1933). Modern works including Osband et al. (2016) and Osband et al. (2019) extend this idea to neural networks in the full RL setting. Other works (Fortunato et al., 2018; Plappert et al., 2018) similarly use neural networks with noisy parameters to sample actions for exploration.

6. Experiments

We provide experiments on a simple environment illustrating the speed of exploration and the sample efficiency of policy learning. For each experiment, we run two versions: (i) with no environment rewards, and (ii) with environment rewards. The former allows us to show the "pure" exploration performance of each method without confounding by the task learning. The latter allows us to show how well exploration cashes out into performance on the end task.

6.1 Representations of policies and counts

Throughout we use deep neural networks for the task policy and for the exploration value function to demonstrate that our approach is practical for use with deep learning. We learn the task policy as a Double DQN (Hasselt et al., 2016) and select actions from the Boltzmann distribution on the Q values:

$$\pi_{\text{task}}(a \mid s) \propto \exp\left\{\frac{Q_{\text{task}}(s,a)}{\tau_{\text{task}}}\right\}. \tag{6}$$

In practice we implement this by reweighting samples from the uniform distribution over the action space, which allows us to use the same procedure for discrete and continuous low-dimensional action spaces. Though re-weighting uniform action samples would scale poorly to large action spaces, a dedicated continuous RL algorithm that represents the policy explicitly could be used in this case. At test time we set τ_{task} very low, resulting in a nearly deterministic policy.

We use the same reweighting procedure for sampling from the behavior policy defined in Equation (2). We draw samples from the task policy, re-weight them according to their value under the pure-exploration Boltzmann distribution, and sample the behavior action from the re-weighted distribution. For training we set $\tau_{\text{explore}} = \tau_{\text{task}} = 0.1$. We found that the results were fairly robust to the setting of τ_{explore} over the range $[0.01, 1]$, while τ_{task} was somewhat more sensitive.

We represent the count function $N(s,a)$ as a lookup table. For continuous state and action spaces we discretize into uniform-sized bins. In order to scale to higher-dimensional spaces alternative exploration rewards could be used, such as pseudo-counts (Bellemare et al., 2016; Ostrovski et al., 2017) or random network distillation (Burda et al., 2018).

6.2 2D point environment

We use a simple 2D point environment to allow clear visualization of state-wise quantities such as visit counts and value functions. This environment is a point-to-point navigation problem with fixed start and goal locations. We use two versions of this environment: a discrete version, with a finite number of states and actions, and a continuous version, with real-valued state and action spaces. The discrete version is a grid-world environment where the agent's action space consists of the four neighboring cells. In the continuous version the agent can set a 2D continuous velocity within a specified range.

A sparse reward is given when the agent reaches the goal location, which makes the exploration problem very difficult to solve with undirected exploration like ε-greedy. For example, in the discrete environment with diagonally opposite start and goal, the probability of seeing a reward under a uniform policy can be upper-bounded using the CDF of the binomial distribution by considering trajectories that include sufficient step towards the goal. With grid size 40×40 and episode length 100, even this extremely generous bound gives a probability 2×10^{-9} of seeing a reward per episode.

6.3 Experimental endpoints

The quantitative measures of success in exploration we use are as follows:

1. Fraction of states visited. This coverage measure shows how well exploration is working, with better exploration visiting more states sooner. For continuous state spaces we measure coverage on a discretized version of the states.

2. Policy test performance. This measures the extent to which the learned policy is able to solve the real task, which is what we care about at the end of the day.

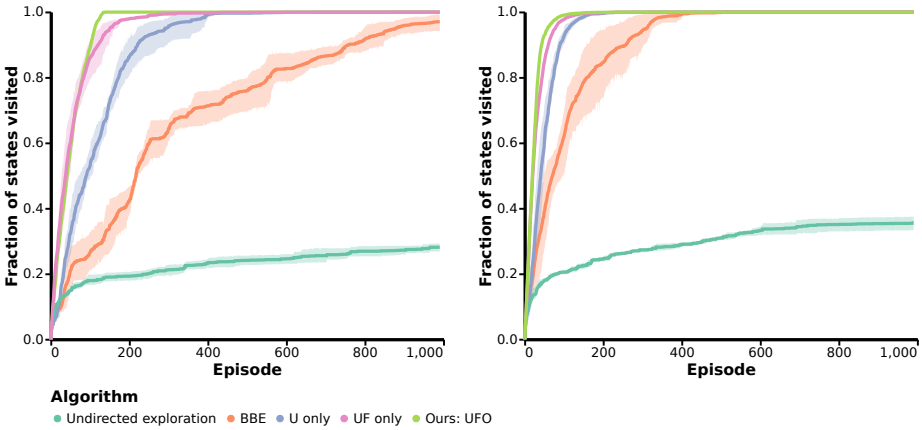


Figure 2: State coverage results in the no-reward (pure exploration) setting. **Left:** The discrete 40×40 grid-world environment. **Right:** The continuous point environment, using 40 bins in each dimension for discretization. Starting from the "undirected exploration" uniform random actions baseline, we see that classic bonus-based exploration (BBE) yields a significant improvement in state coverage on both environments. Even greater gains then come from applying our unbiased task policies (U) and fast adaptation (F) modifications. However, UF still has difficulty reaching the last few states. Finally, we see that our combined algorithm, with the addition of optimism, visits every state rapidly. Compared to the full UFO, UF takes on average $4.2\times$ as long to visit every state in the discrete environment, and $1.2\times$ as long in the continuous environment. Solid lines show the mean over four seeds and shaded regions show the min and max.

6.4 Studying the impact of each modification

We provide a set of ablation experiments that build up from a pure bonus-based exploration algorithm to the full UFO method. This shows that each modification we make individually yields an improvement and when taken together results in significantly reduced sample requirements. We refer to our ablations by the set of modifications to the base BBE algorithm they include, e.g. UF refers to an algorithm with an unbiased task policy and fast adaptation, but no optimism.

Unbiased task policy. A BBE algorithm learns a policy to maximize the sum of an exploration reward and a task reward. The unbiased task policy (U) modification to the BBE algorithm separates this into two policies, one for pure exploration and one for pure task. This gives two benefits. The first is that the task policy is always *unbiased*, in that it attempts to maximize an estimate of the task reward. In Figure 3 (right), it is visible that with pure bonus-based exploration the task policy is biased even after 1000 episodes, leading to suboptimal performance. The second benefit is that we are free to update the pure exploration policy more aggressively, since it does not need to converge in the limit. We use a larger learning rate for updating the exploration policy than the task policy, leading to faster coverage in the no-reward case shown in Figure 2.

Fast adaptation of the exploration value function. The next modification we add is fast adaptation (F), giving us a UF algorithm. The U algorithm, like BBE, performs updates only between episodes, and uses a replay which contains old exploration bonuses calculated at action selection time. By contrast, an algorithm with fast adaptation updates the exploration policy aggressively after each transition, and calculates exploration rewards at update time. This allows it to have a fresher estimate of the optimal exploration policy. Comparing results from U and UF, we see that fast adaptation leads to faster state coverage (Figure 2) and thus faster task learning (Figure 3). Qualitatively the trajectories of policies with fast adaptation visit multiple novel states per episode and cover more states (Figure 4).

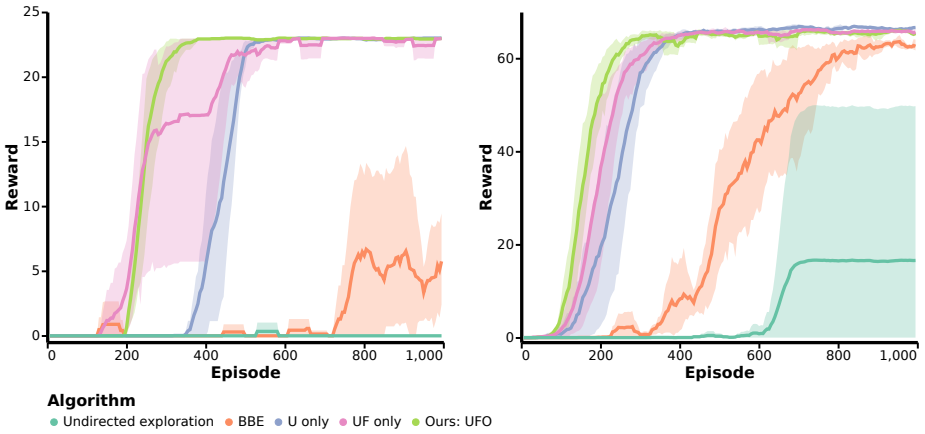
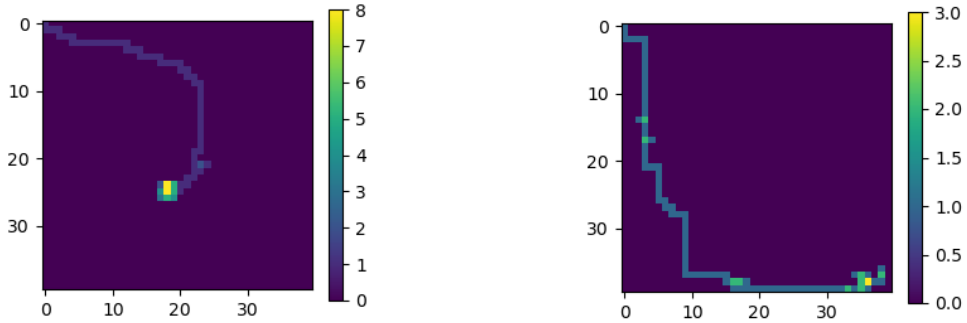


Figure 3: Results of policy learning with different exploration algorithms. **Left:** The discrete 40x40 grid-world environment. **Right:** The continuous point environment. Each additional modification to the bonus-based exploration algorithm performs incrementally better on the end task. Solid lines show the mean over four seeds and shaded regions show the min and max.

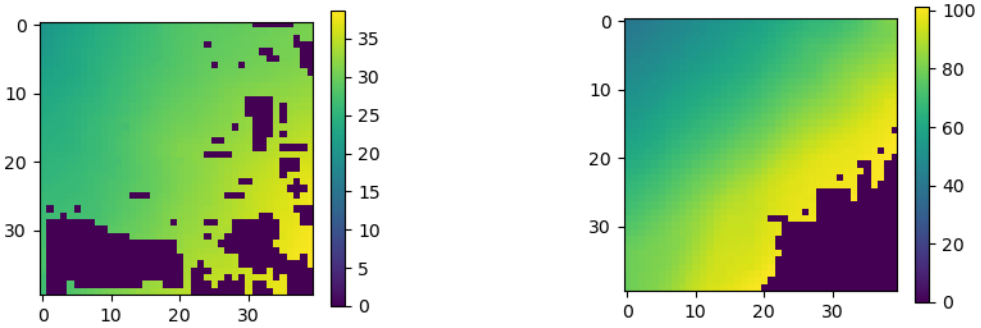


(a) Trajectory without fast adaptation

(b) Trajectory with fast adaptation

Figure 4: Two trajectories in 2D state space showing the difference in behavior without (left) and with (right) fast adaptation. While a slow-adapting policy goes to a single high-novelty point and stays there, a fast-adapting policy can visit multiple new points within an episode. This subtle change makes a significant difference in exploration speed. The color of a state indicates the number of visits to that state in this episode.

Optimistic exploration value function. Our final modification is optimism (O) in the exploration policy. The UF algorithm explores much more rapidly than the base BBE algorithm, but has a weakness: since it only updates its value estimates on transitions that it has already taken, it can't assign new transitions high value, and takes them only by chance. With the addition of optimism, the exploration policy can intentionally seek out new transitions. Qualitatively this leads to more complete coverage of the state space, which can be seen by comparing the UF and UFO results in Figure 5. As shown in Figure 2, optimism improves the number of episodes needed to cover the state space by 4.2× and 1.2× on the discrete and continuous environments, respectively. This translates into faster and more reliable policy learning (Figure 3).



(a) Q_{explore} without optimism (b) Q_{explore} with optimism

Figure 5: An exploration value function trained without (left) and with (right) optimism, evaluated at every state in the 2D state space. Q_{explore} without optimism places high value on some states near its exploration frontier, but not all. Q_{explore} with optimism places uniformly high value along the unexplored region. States marked with zero value (dark purple) are those that have never been visited.

7. Discussion

In this paper we have investigated the potential for fast exploration methods to improve the sample efficiency of reinforcement learning. Three limitations have limited the effectiveness of bonus-based exploration in this setting. A biased policy in the finite-sample regime leads to slow convergence to the optimal policy; slow adaptation of the behavior policy to a changing reward function leads to wasted samples; and lack of optimism increases the time required to visit unseen transitions. We proposed to address these limitations head-on, leading to the unbiased, fast-adapting, optimistic (UFO) exploration algorithm. In small-scale experiments, UFO results in faster state coverage when exploring and improved sample efficiency when learning a task policy. We are excited to push exploration into practical relevance in the future by scaling the UFO algorithm to high-dimensional control.

Acknowledgements

We thank many people for valuable conversations early in this project, in particular Nicolas Heess, Pablo Sprechmann, Michael Neunert, Jonas Degrave, Jan Humplik, David Abel, Alessandro Ialongo, and Giambattista Parascandolo.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Manfred Otto Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. *ArXiv*, abs/1806.06920, 2018.

Adrià Puigdomènech Badia, P. Sprechmann, Alex Vitvitskyi, Daniel Guo, B. Piot, Steven Kapturowski, O. Tieleman, Martín Arjovsky, A. Pritzel, Andew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. *ArXiv*, abs/2002.06038, 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

R. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2002.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Will Dabney, Georg Ostrovski, and A. Barreto. Temporally-extended ϵ-greedy exploration. *arXiv: Learning*, 2020.

Victoria Dean, Shubham Tulsiani, and Abhinav Gupta. See, hear, explore: Curiosity via audio-visual association. *ArXiv*, abs/2007.03669, 2020.

Meire Fortunato, Mohammad Gheshlaghi Azar, B. Piot, Jacob Menick, Ian Osband, A. Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, O. Pietquin, Charles Blundell, and S. Legg. Noisy networks for exploration. *ArXiv*, abs/1706.10295, 2018.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

H. V. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. In *J. Mach. Learn. Res.*, 2008.

C. Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient? In *NeurIPS*, 2018.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Michael Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 1998.

J. Z. Kolter and A. Ng. Near-bayesian exploration in polynomial time. In *ICML '09*, 2009.

Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. Count-based exploration with the successor representation. In *AAAI*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Jost Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics, 2020.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, S. Sidor, Richard Y. Chen, Xi Chen, T. Asfour, P. Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *ArXiv*, abs/1706.01905, 2018.

I. Popov, N. Heess, T. Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerík, T. Lampe, Y. Tassa, T. Erez, and Martin A. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *ArXiv*, abs/1704.03073, 2017.

Tabish Rashid, B. Peng, Wendelin Böhmer, and S. Whiteson. Optimistic exploration even with a pessimistic initialisation. *ArXiv*, abs/2002.12174, 2020.

Martin A. Riedmiller, Roland Hafner, T. Lampe, Michael Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and Jost Tobias Springenberg. Learning by playing - solving sparse reward tasks from scratch. In *ICML*, 2018.

Ralf Schoknecht and Martin A. Riedmiller. Reinforcement learning on explicitly specified time scales. *Neural Computing & Applications*, 12:61–80, 2003.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *ArXiv*, abs/1507.00814, 2015.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Alexander L. Strehl, L. Li, Eric Wiewiora, J. Langford, and M. Littman. Pac model-free reinforcement learning. In *ICML '06*, 2006.

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 *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJewlyStDr.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2753–2762. Curran Associates, Inc., 2017.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

William F. Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. In *ICLR*, 2020.