Optimistic Initialization for Exploration in Continuous Control

Sam Lobel, Omer Gottesman[*], Cameron Allen[*], Akhil Bagaria[*], George Konidaris

Brown University

samuel_lobel@brown.edu, omer_gottesman@brown.edu, csal@brown.edu, akhil_bagaria@brown.edu, gdk@cs.brown.edu

Abstract

Optimistic initialization underpins many theoretically sound exploration schemes in tabular domains; however, in the deep function approximation setting, optimism can quickly disappear if initialized naïvely. We propose a framework for more effectively incorporating optimistic initialization into reinforcement learning for continuous control. Our approach uses metric information about the state-action space to estimate which transitions are still unexplored, and explicitly maintains the initial Q-value optimism for the corresponding state-action pairs. We also develop methods for efficiently approximating these training objectives, and for incorporating domain knowledge into the optimistic envelope to improve sample efficiency. We empirically evaluate these approaches on a variety of hard exploration problems in continuous control, where our method outperforms existing exploration techniques. All code to reproduce experiments can be found at this link.[1]

Introduction

Reinforcement learning is the study of learning to maximize cumulative reward while interacting with an environment (Sutton and Barto 2018). Many tasks of interest have a naturally sparse reward signal, for example reaching a goal or satisfying a given condition. Learning in this case is significantly harder than when rewards are naturally dense, because the agent may not receive any guiding signal during most of its training. A common approach in these cases is to augment the sparse reward with a dense reward built using domain knowledge, to guide training in helpful directions (Randløv and Alstrøm 1998). However, there are three problems with this approach. First, it is not always clear how to construct such a reward signal. Second, the learned policy under this changed reward function may no longer be optimal (or even good) in the original task (Ng, Harada, and Russell 1999). Finally, this approach may not be enough to ensure sufficient exploration in challenging exploration domains.

In the absence of a dense learning signal, the agent needs alternative ways to motivate its decision-making. A common approach is *optimistic initialization* in which unobserved transitions are assumed to be maximally desirable until proven otherwise, thus encouraging the agent to explore (Brafman and Tennenholtz 2002; Strehl et al. 2006; Jaksch, Ortner, and Auer 2010). In the discrete state-action settings, optimistic initialization is well understood and leads to strong theoretical regret bounds (Strehl et al. 2006). When using function approximation, however, naïve attempts at optimistic initialization are quickly learned away due to generalization (Rashid et al. 2020; Machado, Srinivasan, and Bowling 2015).

We introduce Deep Optimistic Initialization for Exploration (DOIE), a novel, practical method for optimistically initializing value functions that enables precise control over the effect of generalization on optimism. Our method uses a nearest-neighbors-based optimism module that identifies the degree to which state-action pairs are similar to those already observed by the agent. We then compute an *optimistic* value function by using this similarity measure to interpolate between a learned value-estimate and an optimistic *envelope* that describes an upper bound of the optimal value function. In contrast to an engineered dense reward, this envelope can be defined using very little domain knowledge. However, through *value shaping*, more domain knowledge can be used to define a tighter envelope and speed up learning.

DOIE drastically improves exploration in sparse-reward domains, and is amenable to principled approximation schemes which allow for its use over long time scales. In the limit of complete exploration this method reduces to standard Q-learning and therefore does not modify the fixed point optimal policy. On the other extreme, with no interaction, the effective Q function is the optimistic envelope, thus satisfying optimistic initialization.

We empirically investigate our method's behavior on a variety of challenging sparse reward continuous control problems, demonstrating state-of-the-art performance on a maze navigation domain and improved sample-efficiency compared with exploratory baselines on sparse-reward tasks in the DeepMind Control Suite (Tassa et al. 2018).

Background

We consider sequential decision making problems represented as Markov Decision Processes (MDPs) denoted by

[*] These authors contributed equally.

[1] https://github.com/samlobel/optimistic-init

$\langle \mathcal{S}, \mathcal{A}, T, R, P_0, \gamma \rangle$, where $\mathcal{S}$, $\mathcal{A}$ and γ are the state space, action space, and the discount factor, respectively. The transition and reward functions are given by $T(s, a)$ and $R(s, a)$ respectively, and $P_0(s)$ is the initial state distribution. In this work we focus on *continuous control* problems with continuous state-and-action spaces. We further define the state-action space as $\mathcal{X} := \mathcal{S} \times \mathcal{A}$ and denote a point in that space as $x := \text{concat}(s, a)$.

We seek to learn an action-selection strategy, or *policy*, which results in high cumulative discounted reward. For a given policy π, we define the state-action value function, or Q-function:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[R(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))].$$

We can extract a policy from a given Q-function by choosing the action with maximum value:

$$\pi^{Q}(s) = \arg\max_{a} Q(s, a).$$

A common method of iteravely improving a Q function parameterized by θ is through temporal difference (TD) learning (Sutton 1988; Watkins and Dayan 1992):

$$\begin{aligned} \theta \;\leftarrow\; & \theta + \alpha\delta\Delta_{\theta}\hat{Q}(s, a; \theta) \\ \text{where} \quad & \delta = r + \gamma \max_{a'} \hat{Q}(s', a'; \theta) - \hat{Q}(s, a; \theta) \end{aligned} \qquad (1)$$

and (s, a, r, s') tuples are drawn from experience. The fixed point of this update equation is the Q-function that describes the *optimal* policy:

$$Q^{*}(s_t, a_t) = \mathbb{E}_{\pi^{*}}[R(s_t, a_t) + \gamma \max_{a'} Q^{*}(s_{t+1}, a')]$$

To have principled generalization with continuous states and actions, methods generally require an understanding of which interactions are similar to others (Kakade, Kearns, and Langford 2003). We formalize this by having access to a metric over the space $\mathcal{X}$, characterized by a distance function $d(x_1, x_2)$.

Related Work

The problem of exploration with sparse rewards has been thoroughly studied in tabular domains (Brafman and Tennenholtz 2002; Strehl et al. 2006; Kearns and Singh 2002). A prevailing approach for exploration in such domains is *optimism in the face of uncertainty* (OFU), where the agent assumes high value for unobserved state-actions to drive itself towards new experience (Jaksch, Ortner, and Auer 2010; Ortner and Auer 2007; Brafman and Tennenholtz 2002). In Q-learning, OFU can be implemented through *optimistic initialization*, a method in which the Q-table is initialized to some maximum value and carefully lowered towards the empirical estimates (Strehl et al. 2006).

However, tabular exploration methods are impractical in high-dimensional or continuous environments, where an infinite number of different states are realizable. Modern machine learning methods have scaled to such problems using *deep function approximation*, where the Q-value of each action is represented as the output of a neural network (Mnih et al. 2015). In continuous control problems, deriving a policy from this Q-function is non-trivial, as it requires finding the maximum action-value over a continuous space of actions. A common strategy is to train a *policy network* to find the maximum-value action for a given state; however under this framework, optimistic modifications to a Q-function are not immediately reflected in the policy. To more directly investigate the effect of optimistic Q-values, we apply our exploration algorithms on top of RBFDQN (Asadi et al. 2021), a continuous control architecture that achieves state-of-the-art performance by extracting a policy directly from its value function without using a policy network. However, as demonstrated in Appendix 3, our approach can also be applied to state-of-the-art policy-based methods, with similar results.

A variety of works attempt to generate more scalable forms of optimism to aid exploration in the function-approximation setting. Prior work has investigated modifying the bias term of a Q-function to induce optimistic initialization (Machado, Srinivasan, and Bowling 2015), though this form of optimism can be quickly learned away in the deep function approximation setting. Various other methods have introduced novelty-bonuses to aid exploration. Notable examples include bonuses based on density estimation and pseudocounts (Bellemare et al. 2016; Ostrovski et al. 2017), the error of predicting a random neural network's output (RND) (Burda et al. 2019), and the error of predicting an environment's transition dynamics (model-predictive error, or MPE) (Houthooft et al. 2016; Pathak et al. 2017). These bonuses can be applied both during bootstrapping (Bellemare et al. 2016) and during action-selection (Rashid et al. 2020). Though these methods scale to large problems, they often do not utilize information which may be known about an environment's structure.

In more structured environments, an agent can frequently take advantage of metric-based learning (Kakade, Kearns, and Langford 2003). Recent theoretical results have bounded cumulative regret by assuming Lipschitz continuity of either the optimal Q-function or the transition function, in both deterministic (Ni, Yang, and Wang 2019) and stochastic (Pazis and Parr 2013; Touati, Taiga, and Bellemare 2020) domains. However, due to their restrictive assumptions these methods may be computationally impractical to apply as the dimensionality of the learning domain increases.

An additional way to improve exploration performance is by incorporating domain knowledge into learning algorithms. For example, in discrete MDPs, specifying tighter upper bounds for Q-values can increase learning speed (Abel et al. 2018). For online learning, potential-based reward shaping biases exploration towards user-defined "high potential" regions of the state space (Ng, Harada, and Russell 1999). When a goal-state is known, various goal-diversity methods can be used to improve exploration (Pitis et al. 2020; Trott et al. 2019). For situations where a general outline of the desired policy is known, defining initiation and termination conditions in the *options* framework (Sutton, Precup, and Singh 1999) biases exploration towards learning useful sub-policies.

Our work is situated at the intersection of these three lines

Algorithm 1 Iterative Covering Set Creation

```
Input: radius $\epsilon$
Initialize $\hat{X} = \{\}$.
for each episode do
  $s \leftarrow$ env.reset()
  for each step do
    $a = \pi(s)$
    $x = (s, a)$
    $d_{\text{min}} = \min_{(x') \in \hat{X}} d(x, x')$
    if $d_{\text{min}} > \epsilon$ then
      $\hat{X} \leftarrow \hat{X} \cup \{x\}$
    end if
    $s \sim T(s, a)$
  end for
end for
```

of progress: bonus based exploration (Taïga et al. 2020), RL in metric spaces (Kakade, Kearns, and Langford 2003) and incorporating domain knowledge using shaping (Ng, Harada, and Russell 1999). We assume continuity of the optimal Q function, and provide a practical method for neural-network based learning while maintaining optimism of transitions far from those which have previously been encountered. In contrast to other bonus-based methods, we directly take advantage of the metric structure present in many continuous control problems. We further provide a simple method for incorporating domain knowledge into the learning algorithm for faster convergence, which in challenging exploration problems can be more effective than a hand-crafted dense reward (Mataric 1994).

Optimistic Initialization in Continuous MDPs

The intuition behind DOIE is to construct a modified Q function that takes on an optimistic value for transitions far outside the agent's experience, and smoothly relaxes to empirical estimates for transitions near those which have been observed. This can be achieved through the use of *knownness*, a quantity which equals 1 for observed transitions, and smoothly decays to 0 for state-actions far away from any observations. Given a knownness function $\kappa(s, a)$ satisfying these properties, we can construct an optimistic Q-function, Q^{+}, as follows:

$$Q^{+}(s, a) = \kappa(s, a) Q(s, a) + (1 - \kappa(s, a))\, \mathrm{Q}_{\max}(s, a). \tag{2}$$

Choosing an optimistic upper-bound where $\mathrm{Q}_{\max} \geq Q^{*}$ everywhere ensures that $Q^{+}(s, a) \geq Q(s, a)$, and thus incentivizes the agent to explore regions of the state space with low knownness. $\mathrm{Q}_{\max}$ frequently can be specified using commonly-known quantities of an environment. For example, when the maximum per-timestep reward $r_{\max}$ is known,

$$\mathrm{Q}_{\max} = \frac{r_{\max}}{1 - \gamma}.$$

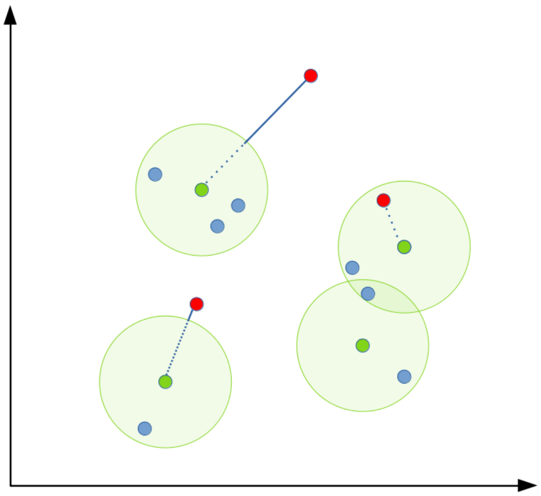

Figure 1: A visualization of $\hat{d}_{\text{min}}$ in a 2-dimensional state-action space. Green points at the center of each circle represent the filtered covering-set. Blue points represent observed state-actions. Red points represent "query points". The blue solid line is $\hat{d}_{\text{min}}$. Note that $\hat{d}_{\text{min}} = 0$ for the red point within the covering-ball, and that $\hat{d}_{\text{min}} \leq d_{\text{min}}$ always. The approximation $\hat{d}_{\text{min}}$ is equivalent to assuming we have encountered every point in each green ball. If the green balls cover the entire state-action-space, then $\hat{d}_{\text{min}} \equiv 0$.

When an episode terminates after achieving the reward, such as in goal-directed tasks,

$$\mathrm{Q}_{\max} = r_{\text{goal}}.$$

We use this optimistic Q function for both bootstrapping:

$$Q(s, a) \leftarrow r(s, a) + \max_{a'} \gamma Q^{+}(s, a') \tag{3}$$

and action-selection:

$$\pi_{Q^{+}}(s) = \arg\max_{a} Q^{+}(s, a). \tag{4}$$

We now describe the measure of knownness for a state-action pair. As stated earlier, we would like knownness to quantify similarity of the state-action to previously observed state-actions. We assume $\mathcal{X}$ is endowed with a metric, $d(x_1, x_2)$, that quantifies such similarity, as is common in many works in continuous RL (Ni, Yang, and Wang 2019; Asadi, Misra, and Littman 2018). We define knownness as a function of the distance to the closest state-action which the agent has observed:

$$\kappa(x) = \beta(d_{\text{min}}(x)) \tag{5}$$

$$d_{\text{min}}(x) = \min_{x' \in X} d(x, x'). \tag{6}$$

where $\beta(d)$ is a kernel function such that $\beta(0) = 1$ and decays monotonically to zero as d goes to infinity, and X is the set of all previously observed state-actions. In this work we use

$$\beta(d) = \frac{1}{1 + (d/d_0)^2} \tag{7}$$

where d_0 is a lengthscale parameter that quantifies how quickly the value of Q can change. This function mimics the Gaussian function near the origin, but does not fall away exponentially as d increases.

A natural metric for such a task is L_2. We can make this metric more flexible by allowing for different weighting of different state-action dimensions in the following way:

$$d(x_1, x_2) = |Ax_1 - Ax_2|_2 \tag{8}$$

We note that this definition of knownness is similar to one presented in prior work (Nouri and Littman 2009), except that ours modulates only the next-state's value, rather than the entire update target. This is more suited to continuous control, where domains are often only mildly stochastic and thus every (s, a) in an agent's experience will be largely *known*, while the same cannot be said about every (s', a').

Covering sets for Efficient Knownness

The computational complexity of a naïve implementation of our knownness calculation scales linearly with the number of state-actions visited. For tasks which require substantial interaction to solve, this quickly becomes infeasible. We introduce a simple approximation algorithm which maintains a filtered set of state-actions that is an ϵ-covering of all seen state-actions so far (Algorithm 1). A subset $\hat{X}$ of X is an ϵ-covering if:

$$\forall x \in X \ \ \exists \hat{x} \in \hat{X} \ \ \text{s.t.} \ \ d(x, \hat{x}) \leq \epsilon.$$

We define $\hat{d}_{\min}(x)$ as the distance of x to the ϵ-ball around any member of the covering-set:

$$\hat{d}_{\min}(x) = \max\left\{\min_{x' \in \hat{X}} d(x, x') - \epsilon, \ \ 0\right\}, \tag{9}$$

and analogously the approximate knownness as:

$$\hat{\kappa}(x) = \beta(\hat{d}_{\min}(x)). \tag{10}$$

We choose $d_{\min}$ as such so that $\hat{\kappa}(x)$ is an efficiently-computable upper bound of the true knownness (proof in Appendix 1):

$$\hat{\kappa}(x) \geq \kappa(x). \tag{11}$$

Furthermore, if $\mathcal{X}$ is a compact metric space, and $\hat{X}$ is an ϵ-covering of $\mathcal{X}$, it follows that $\hat{\kappa}(x) = 0$ for all $x \in \mathcal{X}$. Thus, this approximation preserves the desirable property that knownness goes to 0 everywhere in the limit of thorough exploration. Figure 1 diagrams the filtering process and the relationship between $\hat{\kappa}(x)$ and $\kappa(x)$, and provides intuition on the lower-bound of equation 11. We examine the time, space, and performance tradeoffs of this approximation in our empirical results. Details for adaptive filtering in domains where the scale of the state space is not known a priori can be found in Appendix 2.

Value Shaping

An attractive property of optimistic initialization is that it allows for easy incorporation of domain knowledge through specifying a shaped upper bound to the Q-function. At a high level, optimistic initialization works by pinning the values of completely unknown state-actions to $\text{Q}_{\max}$, and "whittling down" the excess optimism through interaction, until $Q(s, a)$ approaches $Q^*(s, a)$. How much whittling is necessary is a function of how over-optimistic $\text{Q}_{\max}$ is. For example, if $\text{Q}_{\max}(s, a) = Q^*(s, a)$ then the initial policy of Q^+ will be optimal, and no further learning is necessary.

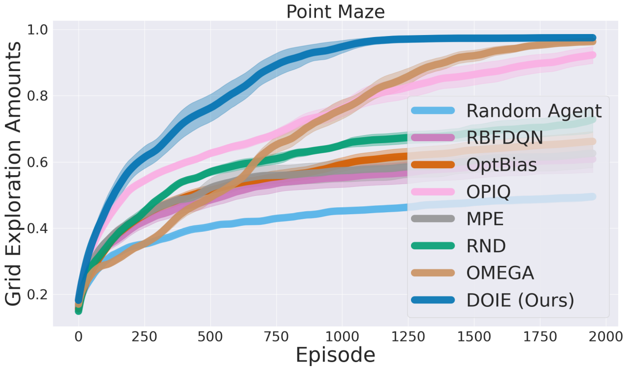

Figure 2: Fraction of state space explored by all exploration methods over 2000 episodes. While both DOIE and OMEGA eventually cover the state space, DOIE accomplishes it in roughly half the episodes.

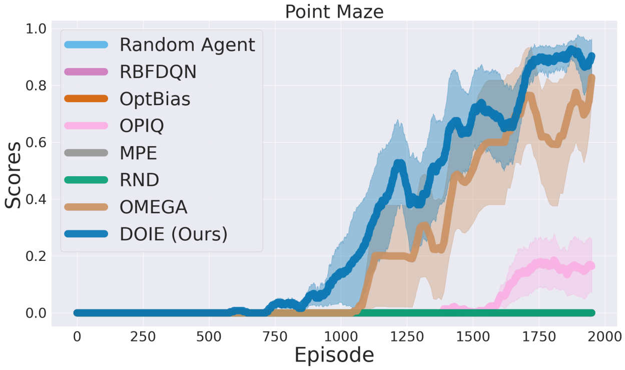

Figure 3: Success percentage of Optimistic Initialization and baselines on PointMaze over 2000 episodes. The shaded region represents the standard deviation over 5 runs. Only OMEGA and DOIE reach the goal in the allotted time.

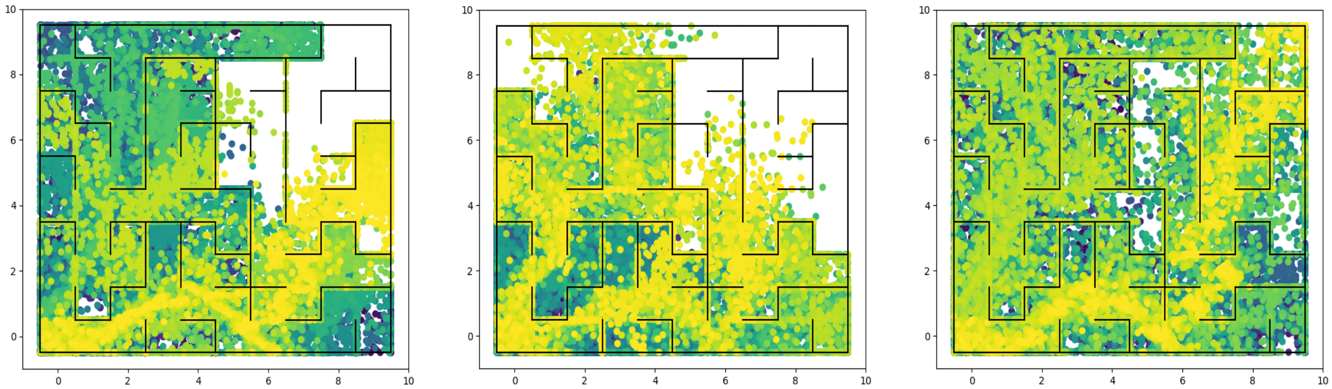

Figure 4: State visitation of the three best-performing models throughout the first 1000 episodes on PointMaze. Left: OPIQ, center: OMEGA, right: DOIE. Color denotes episode number, with blue being near the beginning of training and yellow near the end.

Clearly, being given $Q^*(s, a)$ as an initialization is an unrealistic scenario, because learning Q^* is the goal of interaction. However, it is often possible to use domain knowledge to lessen the distance between $\mathrm{Q}_{\max}$ and Q^* while maintaining the crucial property that $\mathrm{Q}_{\max}(s, a) \geq Q^*(s, a)$ everywhere. In our empirical results, we provide an example of value shaping which accelerates learning for a task where the agent has access to the location of its terminal goal-state.

Empirical Results

We test our method on three sets of domains. First, we investigate our method's exploration performance in detail on a challenging point navigation task, achieving state-of-the-art sample efficiency. We then compare our method to a variety of exploration baselines on modified versions of the benchmark tasks in the DeepMind Control Suite (Tassa et al. 2018). Finally, we investigate the efficacy of reward and value shaping in MountainCar (Singh and Sutton 1996). Details on architectures, training procedures, resource usage and shaping functions are included in Appendix 4. All code used to generate results is included as supplementary material.

PointMaze

PointMaze (Trott et al. 2019) is a challenging continuous control problem with sparse rewards where an agent attempts to navigate a 2D maze (pictured in Figure 4) from the lower-left corner to the upper-right corner. As in prior work, the agent has 50 steps before it is reset to the starting region; an optimal agent would reach the goal in roughly 25 steps (Trott et al. 2019). Neither uniform random exploration nor ϵ-greedy training (Sutton and Barto 2018) solves this problem within 5,000 episodes, so some form of directed exploration is necessary for efficient learning.

We compare against the following bonus-based exploration methods:

- Random Network Distillation (RND), a bonus derived from the error of predicting a randomly-initialized neural network's output (Burda et al. 2019).
- Model-Predictive Error (MPE), a bonus derived from the error of predicting the environment's transition model (Pathak et al. 2017).
- Pseudocounts with action-selection bonus (OPIQ), with counts calculated from a discretization of the normalized state-action space (Rashid et al. 2020).

We also compare against a previous method which enforces optimistic initialization through the value function's bias term (OptBias) (Machado, Srinivasan, and Bowling 2015), and against OMEGA (Pitis et al. 2020), a goal-conditioned exploration algorithm which achieves previous state-of-the-art performance on this domain. For OMEGA we use publicly available code; for all other methods we integrate the novelty-bonus or initialization into an RBFDQN (Asadi et al. 2021) base agent. Finally, we include ϵ-greedy RBFDQN and a random agent as non-exploratory baselines. Details of all architectures and bonuses are included in Appendix 4.

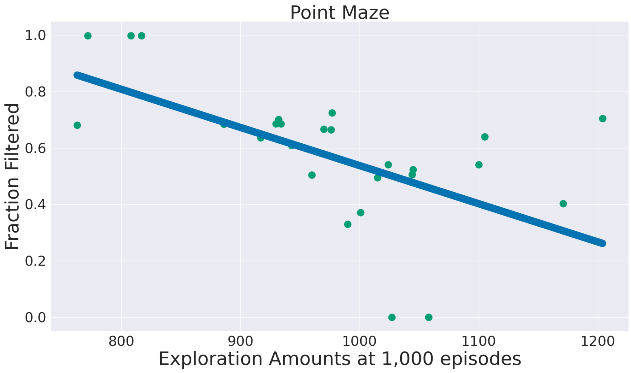

Figure 5: Comparison of exploration amounts versus knownness speedup for various approximation filter radii. As filtering becomes more aggressive, exploration is less effective.

Ability to Explore We begin by investigating the ability of our method to efficiently explore the state space. We visualize the agent's trajectories through training in Figure 4. By the 1000th episode, optimistic initialization has guided the agent through the majority of the state space, while none of the baseline exploration methods have yet explored in the region of the goal. We quantify this exploration in Figure 2 by partitioning the maze into a 40 by 40 grid, and tracking the number of these squares that each agent visits throughout training. Optimistic initialization quickly covers the state space after 1000 episodes, while the strongest baseline takes 2000 episodes to cover similar area.

Performance Results Figure 3 demonstrates the learning performance of each method over 2000 episodes. Besides OMEGA, the remaining baseline methods are unable to reach the goal even after extensive hyperparameter tuning. This is consistent with the exploration graphs presented in the prior section. OMEGA is able to reach the goal consistently, but first reaches the goal nearly 300 episodes after our method.

Effect of approximation Naïve nearest-neighbor calculations involve computing the distance between a query point and every point the agent has encountered, and thus exact nearest-neighbors is impractical for long learning interactions. Earlier, we described a filtering method that makes this process much faster at the expense of the granularity of the knownness calculation. We investigate the relationship between filtering amount and exploration by plotting the speedup of the knownness calculation versus exploration amounts at 500 episodes for a variety of filtering radii (Figure 5). As expected, we see that increasing the amount filtered decreases exploration performance. Our experiments are performed with intermediate values of filtering, so as to train acceptably quickly while still providing the benefits of exhaustive exploration.

DeepMind Control Suite

To gauge the general exploration ability of our method, we test our method on modified versions of four sparse-reward tasks from the DeepMind control suite (Tassa et al. 2018): Pendulum, Hopper Stand, Acrobot, and Ball in Cup (Figure 6). Normally, each of these environments would be initialized, on reset, to a random state, which actually provides good coverage of the state space (including near the goal) and largely obviates the need for directed exploration. To make the problems more challenging, we modify the environments by performing 1000 *no-op* actions at the beginning of each episode in order to settle to a state far from the rewarding region, thus increasing the exploration required to solve the task. We compare to all methods used for PointMaze except for OMEGA, as these tasks are not goal-directed. The tasks vary from low-dimensional (Pendulum, $|\mathcal{X}| = 4$) to relatively high-dimensional (Hopper Stand, $|\mathcal{X}| = 19$). Our method performs competitively on all domains, outperforming all baselines at early-stage learning on Acrobot, Ball in Cup, and Hopper Stand, and learning more quickly than all but RND on Pendulum.

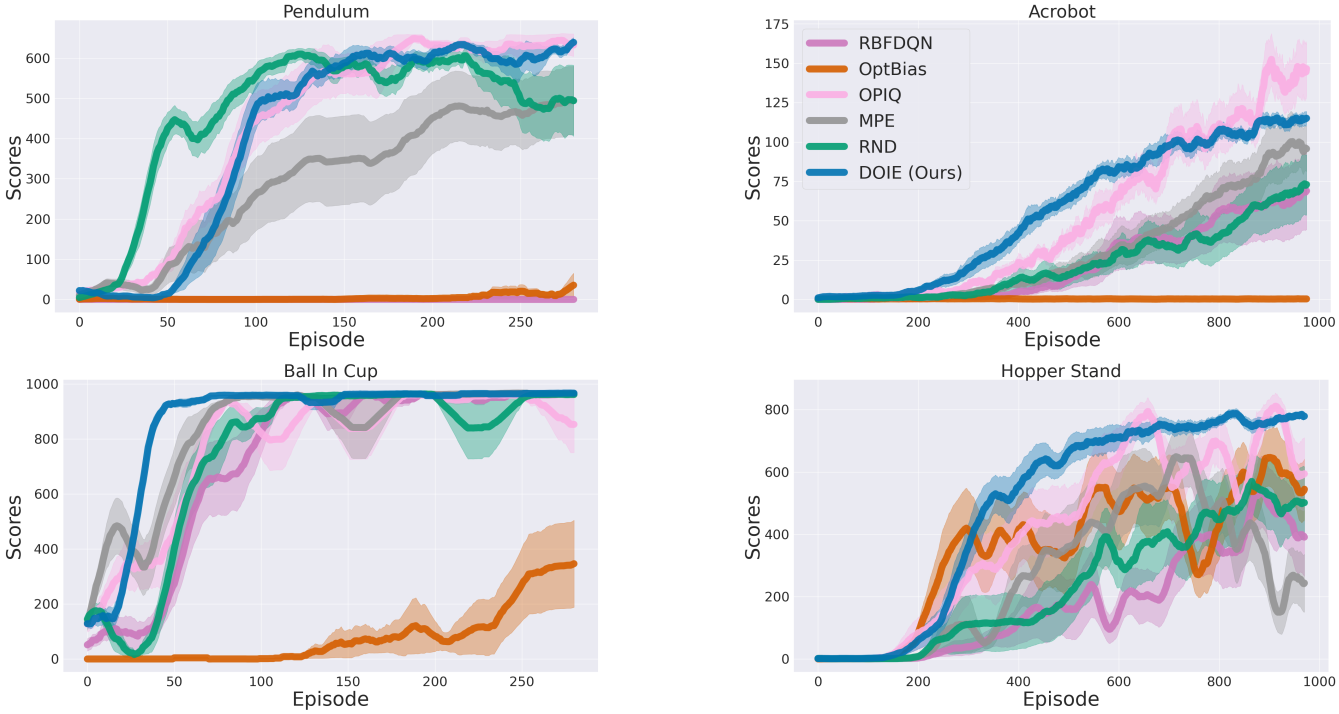

Figure 6: Performance on tasks in the DM Control Suite. The shaded region represents the standard deviation over 8 runs. Colors correspond to the same methods across domains.

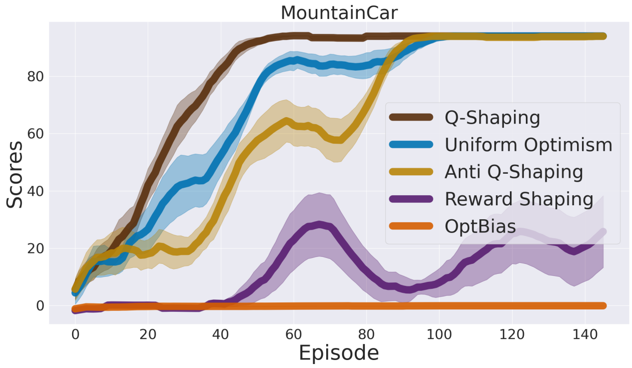

Figure 7: Learning curves for uniform optimistic initialization (blue) as well as shaping (brown) and anti-shaping (gold) on MountatinCar. Optimistic initialization allows us to use value shaping for more efficient learning, while eventually learning an optimal policy is robust to value-shaping misspecification. In comparison to *reward* shaping, value shaping is much more effective at finding a solution. For this task, OptBias and RBFDQN (not shown) were unable to solve the task in the allotted time. The shaded region represents the standard deviation over all 10 runs.

Value Shaping

We now demonstrate how our method allows us to incorporate domain knowledge through specifying a state-dependent $\mathrm{Q}_{\max}$, which we call "value shaping." We test our method on MountainCar, a low-dimensional yet challenging continuous control exploration problem that requires first moving away from the goal in order to build momentum. Using the goal-position s_g, the agent's max speed $v_{\max}$, and the reward for reaching the goal r_g, we can calculate an upper bound of the optimal Q-function Q^*:

$$\mathrm{Q}_{\max}(s, a) \; = \; r_g \gamma^{|s_g - s| / v_{\max}}. \tag{12}$$

We limit the episode length to 80 steps, which necessitates directed exploration in order to reach the goal. Our results are presented in Figure 7.

In comparison to a uniform $\mathrm{Q}_{\max}$, when we provide a tighter upper bound through value shaping we see the agent explore much more quickly. We also include a reversal of this shaping (anti-shaping), which maintains the upper-bound property $\mathrm{Q}_{\max} \geq Q^*$ but sets the upper-bound value higher *away* from the goal. Though this deliberately provides the agent with bad advice, notably the agent still learns to reach the goal, albeit more slowly. While value shaping specifies regions of the state-space which should be explored first, since the optimism is learned away the agent still eventually explores until it finds the solution in all runs.

We contrast this to augmenting the environment with a dense reward, which is a common strategy for solving challenging sparse-reward tasks (Randløv and Alstrøm 1998). We add to the sparse reward a shaped potential-based reward (Ng, Harada, and Russell 1999) that is proportional to the progress made towards the goal in a given timestep. This shaping provably does not change the optimal learned Q-function (Ng, Harada, and Russell 1999), and provides roughly the same information as our value-shaping. Despite this, the agent is quite slow to complete the task, and only finds a solution in 3 out of 10 runs in the allotted time. Through observation we discover that the agent constantly actuates to the right, myopically maximizing its cumulative reward but not gathering the data necessary to learn the optimal solution. We posit this problem remains in more complicated tasks, and that this common augmentation procedure may frequently learn policies which are locally, but not globally, optimal.

Conclusion

Optimistic initialization is foundational to reinforcement learning in discrete MDPs; in this paper we develop techniques that successfully apply this concept to deep neural networks used to solve continuous control problems. By taking advantage of the smoothness of metric spaces common to many such problems, we develop a simple, yet effective method for sample-efficient exploration. Thorough experimentation in six domains demonstrates that the proposed method outperforms several popular exploration algorithms. We additionally introduce an approximation technique that uses covering sets to preserve many desirable training properties while making our method computationally practical on long time-horizons. Finally, via *value shaping*, we show how optimistic initialization can be used to incorporate domain knowledge into Q-estimates, further accelerating learning in challenging problems. We view the synthesis of these techniques as a step towards sample-efficient exploration in sparse-reward continuous control.

Acknowledgements

We thank Kavosh Asadi, Saket Tiwari, Rafael Rodriguez-Sanchez, and other members of BigAI for their valuable input. This research was supported in part by an NSF Graduate Research Fellowship under grant #2040433, NSF grants #1717569 #1955361 and CAREER award #1844960, DARPA grant W911NF1820268, and ONR PERISCOPE MURI Contract N00014-17-1-2699. This research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The content is solely the responsibility of the authors and does not necessarily represent the official views of DARPA, the NSF, the ONR, or the AFOSR.

References

Abel, D.; Jinnai, Y.; Guo, S. Y.; Konidaris, G.; and Littman, M. 2018. Policy and value transfer in lifelong reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 20–29. PMLR.

Asadi, K.; Misra, D.; and Littman, M. 2018. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, 264–273. PMLR.

Asadi, K.; Parikh, N.; Parr, R. E.; Konidaris, G. D.; and Littman, M. L. 2021. Deep radial-basis value functions for continuous control. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 6696–6704.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*.

Brafman, R. I.; and Tennenholtz, M. 2002. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231.

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*.

Ecoffet, A.; Huizinga, J.; Lehman, J.; Stanley, K. O.; and Clune, J. 2021. First return, then explore. *Nature*, 590(7847): 580–586.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596. PMLR.

Houthooft, R.; Chen, X.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; and Abbeel, P. 2016. VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*.

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4).

Kakade, S.; Kearns, M. J.; and Langford, J. 2003. Exploration in metric state spaces. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 306–312.

Kearns, M.; and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2): 209–232.

Machado, M. C.; Srinivasan, S.; and Bowling, M. 2015. Domain-independent optimistic initialization for reinforcement learning. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Mataric, M. J. 1994. Reward functions for accelerated learning. In *Machine learning proceedings 1994*, 181–189. Elsevier.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*.

Ni, C.; Yang, L. F.; and Wang, M. 2019. Learning to control in metric space with optimal regret. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 726–733. IEEE.

Nouri, A.; and Littman, M. 2009. Multi-resolution exploration in continuous spaces. In *Advances in Neural Information Processing Systems*, volume 21, 1209–1216.

Ortner, P.; and Auer, R. 2007. Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in Neural Information Processing Systems*, 19: 49.

Ostrovski, G.; Bellemare, M. G.; Oord, A.; and Munos, R. 2017. Count-based exploration with neural density models. In *International conference on machine learning*, 2721–2730. PMLR.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2778–2787. PMLR.

Pazis, J.; and Parr, R. 2013. PAC optimal exploration in continuous space Markov decision processes. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Pitis, S.; Chan, H.; Zhao, S.; Stadie, B.; and Ba, J. 2020. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 7750–7761. PMLR.

Randløv, J.; and Alstrøm, P. 1998. Learning to drive a bicycle using reinforcement learning and shaping. In *International Conference on Machine Learning*. Citeseer.

Rashid, T.; Peng, B.; Boehmer, W.; and Whiteson, S. 2020. Optimistic exploration even with a pessimistic initialisation. In *International Conference on Learning Representations*.

Singh, S. P.; and Sutton, R. S. 1996. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1): 123–158.

Stadie, B. C.; Levine, S.; and Abbeel, P. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Strehl, A. L.; Li, L.; Wiewiora, E.; Langford, J.; and Littman, M. 2006. PAC model-free reinforcement learning. *International Conference on Machine Learning*.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1): 9–44.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Taïga, A. A.; Fedus, W.; Machado, M. C.; Courville, A. C.; and Bellemare, M. G. 2020. On Bonus Based Exploration Methods In The Arcade Learning Environment. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; Casas, D. d. L.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690*.

Touati, A.; Taiga, A. A.; and Bellemare, M. G. 2020. Zooming for efficient model-free reinforcement learning in metric spaces. *CoRR*.

Trott, A.; Zheng, S.; Xiong, C.; and Socher, R. 2019. Keeping your distance: solving sparse reward tasks using self-balancing shaped rewards. In *Advances in Neural Information Processing Systems*, volume 32, 10376–10386.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3-4): 279–292.

Technical Appendix

1 Proofs

Proof of Equation 11: Since $\beta(d)$ decays monotonically for $d \geq 0$ and $\hat{\kappa}(x) = \beta(\hat{d}_{\min}(x))$, to prove $\hat{\kappa}(x) \geq \kappa(x)$ we can show that $\hat{d}_{\min}(x) \leq d_{\min}(x)$.

Let $x^* \in X$ be the closest point to x, meaning $d_{\min}(x) \equiv |x - x^*|$. Let $\hat{x}^* \in \hat{X}$ be (one of) the points in the covering set for which $|\hat{x}^* - x^*| \leq \epsilon$. Then,

$$\begin{aligned} |x - \hat{x}^*| &\leq |x - x^*| + |x^* - \hat{x}^*| && \text{[Triangle Inequality]} \\ |x - \hat{x}^*| &\leq |x - \hat{x}^*| + \epsilon && \text{[Defn. covering set]} \\ |x - \hat{x}^*| - \epsilon &\leq |x - x^*|. && \text{[Rearranging]} \end{aligned}$$

Now, we can relate $\hat{d}_{\min}$ with $d_{\min}$. If

$$\min_{x' \in \hat{X}} d(x, x') - \epsilon \leq 0,$$

then $\hat{d}_{\min} = 0 \leq d_{\min}$. Otherwise,

$$\begin{aligned} \hat{d}_{\min}(x) &= \min_{x' \in \hat{X}} d(x, x') - \epsilon \\ &\leq |x - \hat{x}^*| - \epsilon \\ &\leq |x - x^*| \\ &= d_{\min}(x). \end{aligned}$$

2 Adaptive Filtering

When the approximate size of the state space is known, one can normalize the state space as in equation 8 and choose an appropriate length-scale d_0 through domain knowledge or hyperparameter sweep. However, in many problems the relative sizes of each dimension may vary significantly and are not known in advance. For example, in the Acrobot domain the velocity features take on roughly 50x the range of values compared with the position features. Without prior knowledge, we would like to explore along each dimension equally, however in such situations a single feature dominates the knownness calculation. To overcome this imbalance, we use a simple adaptive rescaling procedure: after each episode we update the distance metric with a new diagonal normalizing matrix A based on the current minimum and maximum values observed for each feature,

$$A_{ii} = \frac{1}{\max\limits_{x \in X} x_i - \min\limits_{x \in X} x_i + \delta} \ ,$$

where δ is a small constant (we use 10^{-6}) to prevent divide-by-zero errors when a feature has a constant value. This transforms the interaction data so that each feature has a range close to 1 during parameter updates. Since the denominator is monotonically increasing, this ensures that updates to A only ever shrink the distances between points; thus any summarizing set S which covers interaction history X under A^t will still cover X under the updated A^{t+1}. Therefore we can use Algorithm 1 with either fixed or adaptive scaling.

3 Policy Gradient Methods

We focus on value-function-only methods in this paper because for this class of methods, optimism in the Q-function is immediately represented in the policy. In contrast, for more standard policy-gradient methods, optimism must first be distilled to the policy network before it affects exploration. Therefore, value-function-only architectures such as RBFDQN (Asadi et al. 2021) are a cleaner testbed for bonus-based exploration methods. Despite this extra machinery, we find that we achieve qualitiatively similar results when each of our exploration methods (ours and baselines) are applide to TD3, a state-of-the-art policy gradient architecture. We report these results in Figure 8.

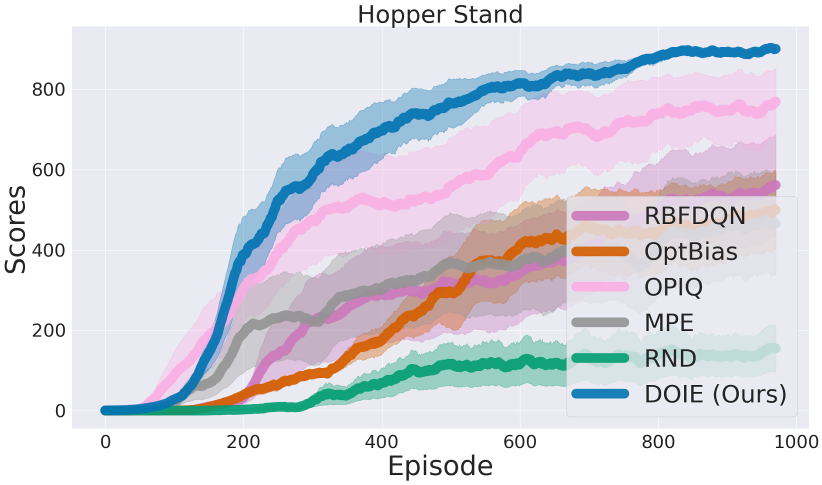

Figure 8: Learning curves for all exploration methods applied to a TD3 base-architecture, on the Hopper Stand task.

4 Architectures and Hyperparameters

Here we describe the architectures we use for all experiments. All experiments were run on a cluster, with each job having access to a 12GB GPU, 32 GB RAM, and 2 CPUs. Complete implementations and instructions for running experiments can be found in our code submission. All hyperparameters not listed are shared between all experiments, and can be found in the hyperparameters directory.

4.1 OMEGA

We use publicly available code with reported hyperparameters, to re-run experiments with OMEGA on the PointMaze domain (Pitis et al. 2020).

4.2 RBFDQN

For the results in our main paper, we apply all exploration methods besides OMEGA using an RBFDQN base agent (Asadi et al. 2021). We also include comparisons with RBFDQN using a decaying ϵ-greedy exploration strategy for all experiments, as a non-directed exploration baseline.

4.3 TD3

To demonstrate our method's applicability to policy-gradient methods we run an additional set of experiments using Twin-Delayed Deep Deterministic Policy Gradient (TD3) as a base agent (Fujimoto, Hoof, and Meger 2018). TD3 trains a policy network to maximize a Q-function, and reduces overestimation bias by using the minimum value of two Q-functions when bootstrapping. Our implementation of TD3 is included in the Code Appendix. For bonus-based methods we optimistically modify both Q-function using the same optimism module. As TD3 uses a policy network for action selection, applying a bonus during interaction for action-selection has no effect. Hyperparameters for TD3 on the "Hopper Stand" domain are included in the tables below.

4.4 DOIE

Our method, Deep Optimistic Initialization for Exploration (DOIE), modifies the effective Q-function for bootstrapping and action selection, as detailed in equation 3 and equation 4. Unless otherwise specified, we choose $\mathrm{Q}_{\max} = r_{\text{goal}}$ for goal-directed tasks, and $\mathrm{Q}_{\max} = \frac{r_{\max}}{1-\gamma}$ otherwise. We use RBFDQN as our base agent. For tasks with known sizes for the state and action spaces, we normalize the spaces to have range 1. Otherwise, we use the adaptive scaling procedure described above. We further scale the action space by some constant α, which is chosen through hyper-parameter search. The length-scale d_0 is also chosen through hyperparameter search. We choose the filtering-radius ϵ on a per-task basis such that learning is sufficiently fast. All hyperparameters are listed below in Table 1. When multiple values are in a cell, it represents the values swept, with the reported value in bold.

	PointMaze	Pendulum	Acrobot	Ball In Cup	Hopper	MountainCar	TD3 Hopper
Updates per episode	100	1,000	1,000	1,000	1,000	1,000	1,000
Action scaling	**0.01**, 0.03, 0.1, 0.3, 1.0	0.5, 1.0, 2.0, 3.0, **4.0**	0.5, 1.0, **2.0**, 3.0, 4.0	0.5, 1.0, **2.0**, 3.0, 4.0	0.5, 1.0, **2.0**, 3.0, 4.0	0.5, **1.0**, 2.0	0.5, 1.0, **2.0**
d_0	**0.1**, 0.3, 1.0, 2.0, 3.0	0.1, 0.3, 1.0, **2.0**, 3.0	0.1, **0.3**, 1.0, 2.0 3.0	0.1, 0.3, 1.0, **2.0** 3.0	0.1, 0.3, 1.0, 2.0 **3.0**	0.1, **0.3**, 1.0	0.1, 0.3, **1.0**, 3.0
Adaptive filtering	No	Yes	Yes	Yes	Yes	No	Yes
Filtering radius	0.001	0.1	0.25	0.75	0.25	0.0	0.5
Training time per run	2h 45m	3h 45m	7h 15m	3h 30m	6h 15m	1h 15m	5h 15

Table 1: Hyperparameters for Deep Optimistic Initialization for Exploration (DOIE)

4.5 OPIQ

We implement a count-based bonus using the update rule introduced in OPIQ (Rashid et al. 2020):

$$\theta \leftarrow \theta + \alpha\delta\Delta_\theta \hat{Q}(s,a;\theta)$$

$$\text{where} \quad \delta = r + \frac{\beta}{\sqrt{N(s,a)}} + \gamma \max_{a'} \hat{Q^+}(s',a';\theta) - Q(s,a;\theta)$$

$$\text{and} \quad Q^+(s,a) = Q(s,a) + \frac{C}{(N(s,a)+1)^M}$$

As in DOIE, Q^+ is used during action-selection as well as bootstrapping. The term modulated by β is the standard count-based bonus, and the modification to Q^+ ensures optimistic bootstrapping even of state-action pairs which appear zero times in the agent's experience. As is common for counting-based bonuses, we calculate our counts by discretizing the state-action space and tracking the number of visited (s,a) within each bin (Rashid et al. 2020; Ecoffet et al. 2021). For fair comparison, we use the same normalization procedure described in Appendix 2, and discretize this normalized space into hypercubes of side-length d_0. We treat d_0, β, and C as hyperparameters, and use $M = 2$ as recommended by the original OPIQ paper. We additionally scale the action-space by some constant α as we do for DOIE. All hyperparameters are listed below in Table 2. When multiple values are in a cell, it represents the values swept, with the reported value in bold.

	PointMaze	Pendulum	Acrobot	Ball In Cup	Hopper	TD3 Hopper
Updates per episode	100	1,000	1,000	1,000	1,000	1,000
Action scaling	**0.01**, 0.1, 1.0	**1.0**, 2.0 4.0	**1.0**, 2.0, 4.0	1.0, 2.0, 4.0	**0.5**, 1.0 2.0	**0.5**, 1.0 2.0
d_0	**0.1**, 0.3, 1.0, 2.0, 3.0	0.03, 0.1, 0.3, 1.0	0.03, 0.1, 0.3, 1.0	0.03, 0.1, 0.3, 1.0	0.1, 0.3 **0.5**, 1.0	0.1, 0.3 **0.5**, 1.0
β	0.0001 0.001, 0.01 0.1 1.0	0.0001 0.001, 0.01 0.1 1.0	0.0001 0.001, 0.01 0.1 1.0	0.0001 0.001, 0.01 0.1 1.0	0.0001 **0.001**, 0.01 0.1 1.0	0.0001 **0.001**, 0.01 0.1 1.0
C	0.0001 0.001, 0.01 0.1 1.0	0.0001 0.001, 0.01 0.1 1.0	0.0001 0.001, 0.01 0.1 1.0	0.0001 0.001, 0.01 0.1 1.0	**0.0001** 0.001, 0.01 0.1 1.0	**0.0001** 0.001, 0.01 0.1 1.0
Adaptive filtering	No	Yes	Yes	Yes	Yes	Yes
Training time per run 6h 30m	1h 30m	2h 0m	6h 30m	3h 30m	6h 45m	5h 15m

Table 2: Hyperparameters for Optimistic Pessimistically Initialized Q-Learning (OPIQ)

4.6 RND

We implement Random Network Distillation (Burda et al. 2019)(RND) using RBFDQN as our base agent. RND modifies the effective reward used for training:

$$r_t(s,a) = r(s,a) + \alpha\,\text{RND}(s)$$

where α is a scaling hyperparameter. RND(s) is the calculated by determining the L_2 distance between two neural network outputs, one randomly initialized and frozen, and the other trained to mimic the first's output. Following the original implementation, we normalize the outputs of this network so that $\mathbb{E}[\text{RND}(s)] = 0$ and $Std[\text{RND}(s)] = 1$ over the current replay buffer. We update the trained network with the same data and frequency as we update the RBFDQN base agent. We learn a single value head which estimates the intrinsic and extrinsic expected reward. All hyperparameters are listed below in Table 3. When multiple values are in a cell, it represents the values swept, with the reported value in bold.

4.7 MPE

A variety of methods augment the training reward using the error of a learned transition model (Stadie, Levine, and Abbeel 2015; Pathak et al. 2017):

$$r_t(s,a,s') = r(s,a) + \alpha\,\text{MPE}(s,a,s')$$

$$\text{MPE}(s,a,s') = |\phi(s') - T_\theta(s,a)|$$

	PointMaze	Pendulum	Acrobot	Ball In Cup	Hopper	TD3 Hopper
Updates per episode	100	1,000	1,000	1,000	1,000	1000
RND scaling	**0.01**, 0.03, 0.1, 0.3, 1.0	0.01, 0.03, 0.1, 0.3, **1.0**	0.01, **0.03**, 0.1, 0.3, 1.0	0.01, 0.03, **0.1**, 0.3, 1.0	0.001, 0.003, **0.01**, 0.03, 0.1, 0.3, 1.0	0.001, **0.003**, 0.01, 0.03, 0.1, 0.3, 1.0
Training time per run	45m	2h 0m	6h 0m	1h 30m	4h 0m	3h 45m

Table 3: Hyperparameters for Random Network Distillation (RND)

where ϕ maps states to a compact latent space, T_θ is a function approximator trained to mimic the transition function, and α is a scaling hyperparameter. Since, unlike images, our inputs already are a compact feature space, we set ϕ to the identity transformation. We implement T_θ as outputting the residual between s and s', which we find models the environment more accurately since the state only changes slightly per timestep. Unlike RND, MPE does not generally have reward normalization. Thus, we manually inspect r_{MPE} and choose our hyperparameter sweep such that the effective intrinsic reward generally lies between 0.01 and 1.0. The exception is for the Hopper task, for which we could not generate strong results without the normalization procedure. All hyperparameters are listed below in Table 4. When multiple values are in a cell, it represents the values swept, with the reported value in bold.

	PointMaze	Pendulum	Acrobot	Ball In Cup	Hopper	TD3 Hopper
Updates per episode	100	1,000	1,000	1,000	1,000	1,000
MPE scaling	**0.01**, 0.03, 0.1, 0.3, 1.0	300, 1,000, **3,000**, 10,000, 30,000	1.0, **3.0**, 10.0, 30.0, 100	0.3, 1.0 3.0, **10.0**, 30.0	**0.003**, 0.001 0.03, 0.01 0.3, 0.1	0.003, 0.001 0.03, 0.01 0.3, **0.1**
Training time per run	45m	2h 0m	6h 0m	1h 30m	4h 0m	3h 45m

Table 4: Hyperparameters for Model-Predictive Error (MPE)

5 Value Shaping

We investigate the effect of value-shaping on learning speed in the MountainCar domain. Value shaping does not introduce any hyperparameters to the algorithm: besides changing $\mathrm{Q}_{\max}$, we use the architecture and hyperparameters tuned to perform will with DOIE. Using the goal-position s_g, the agent's max speed $v_{\max}$, and the reward for reaching the goal r_g, we can calculate an upper bound of the optimal Q-function Q^*:

$$\mathrm{Q}_{\max}(s, a) \;=\; r_g \gamma^{|s_g - s|/v_{\max}}.$$

We also include a reversal of this shaping, which we refer to as anti-shaping:

$$\mathrm{Q}_{\max}(s, a) \;=\; r_g \gamma^{-|s_g - s|/v_{\max}}.$$

This maintains the upper-bounding property, but directs exploration away from the true goal.

6 Reward Shaping

We additionally implement a dense reward for MountainCar, which we add to the sparse reward during training as a shaping signal. The dense reward is derived from a state-based potential function:

$$r_{\text{dense}}(s, a, s') = \gamma V(s') - V(s)$$

with

$$V(s) = |s - s_g|$$

being the difference in the position dimension between the current state and the goal. We modify the training reward by adding the dense reward with the environment's reward, modulated by a hyperparameter α:

$$r_t = r + \alpha r_{\text{dense}}$$

We choose $\alpha = 10.$ as the best-performing value from the range $(0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0)$. As r_{dense} is a potential-based shaped reward, it does not modify the fixed point of Bellman updates (Ng, Harada, and Russell 1999).