Strategic Exploration in Object-Oriented Reinforcement Learning

Ramtin Keramati [*1] Jay Ha Whang [*2] Patrick Cho [*2] Emma Brunskill [2]

Abstract

Humans learn to play video games significantly faster than state-of-the-art reinforcement learning (RL) algorithms. Inspired by this, we use an object-oriented representation for RL to learn simple dynamics model and do planning with strategic exploration. In contrast to the tabular settings which is the focus of most theoretical analysis of efficient exploration, it is not tractable to perform exact planning. We investigate how various model-based strategic exploration strategies work when combined with the popular Monte Carlo Tree Search approximate planning technique, and find preliminary evidence that optimism-based strategies can be particularly beneficial. We use the resulting technique on perhaps the hardest Atari game *Pitfall!* and our preliminary results suggest substantially improved exploration and performance over prior methods.

1. Introduction

The coupling of deep neural networks and reinforcement learning has led to exciting advances, enabling reinforcement learning agents that can reach human-level performance in many Atari2600 games (Mnih et al., 2015). However, such agents typically require hundreds of millions of time steps to learn to play well. As recently noted (Lake et al., 2017), this is in sharp contrast to people, who typically learn to play Atari games within a few episodes. Prior work on human learning for Atari highlights people's ability to generalize from few examples, explore and plan efficiently (Tsividis et al., 2017). People also seem to benefit substantially from using a higher-level object representation. Humans can strategically explore object dynamics models, and use to compute high expected reward plans.

An important question is whether we can define algorithms that can similarly leverage such strategies to enable vastly more efficient reinforcement learning. We hypothesize that the intersection of three features may be sufficient to enable such success: (a) leveraging abstract object-level representations, (b) learning (often inaccurate) models of the world dynamics that can be learned quickly and support fast planning, and (c) strategic model-based exploration using lookahead planning.

Strategic exploration methods have been studied in great detail in the tabular setting with recent results yielding near-tight probably approximately correct (Dann et al., 2017) and tight regret bounds (Azar et al., 2017). Extensions of tabular optimistic bonuses (e.g. (Strehl & Littman, 2008)) to deep model-free RL methods (Bellemare et al., 2016) show substantially improved performance in many environments over ϵ-greedy exploration. However, these methods still require millions of frames and struggle in domains that involve sparse delayed reward.

A challenge with many of these methods is propagating the optimistic reward bonuses to encourage exploration, a challenge that should be addressed by performing lookahead planning using model-based RL, such as UCT algorithm (Kocsis & Szepesvári, 2006). Unfortunately, due to the difficulty in learning accurate models in complicated domains, model-based RL has not matched the impressive performance observed by its model-free counterparts.

While prior attempts have tried using object-level representations to provide key inductive bias to accelerate learning, they do not couple their efforts with strategic exploration (Garnelo et al., 2016; Roderick et al., 2017; Cobo et al., 2013). Our work is inspired by an important exception to this, the DOORMAX algorithm (Diuk et al., 2008), which performs strategic R-max (Brafman & Tennenholtz, 2002) like exploration to learn logic-like dynamics models. This work assumes that planning can be done exactly. However, even when using object-level representations, in long horizon, sparse reward domains it will still typically be intractable to perform perfect lookahead planning.

In this paper we also use an object-oriented representation for RL (Diuk et al., 2008) and make three key contributions.

1. Given it will generally be impossible to perform exact

[*] Equal contribution [1] Institue of Computational and Mathematical Engineering, Stanford University, Stanford, California, USA [2] Department of Computer Science, Stanford University, Stanford, California, USA. Correspondence to: Ramtin Keramati <keramati@cs.stanford.edu>.

Published at the Exploration in Reinforcement Learning Workshop at the 35th International Conference on Machine Learning, Stockholm, Sweden. Copyright 2018 by the author(s).

lookahead planning, we investigate which model-based strategic exploration strategies perform best when combined with the popular approximate Upper Confidence Tree planning algorithm (Kocsis & Szepesvári, 2006). Our results highlight that optimistic strategies can be substantially better than Thompson Sampling when using MCTS as the planner.

2. Leveraging these results, we introduce a new object-oriented, model-based, optimistic strategic RL algorithm. Our algorithm takes in simple action macros (of the form "act and then wait" identical to those defined in prior work (Diuk et al., 2008)) that may mimic human performance due to reaction time. It also leverages an inductive prior that there should exist simple deterministic models of the world dynamics.

3. We evaluate our approach on *Pitfall!*, perhaps the hardest Atari2600 game with extremely sparse positive reward. To our knowledge, our approach is the first method to achieve positive reward on this game without human demonstrations.

2. Object Representation

Consider a finite horizon Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ the transition function, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, and γ the discount factor. The goal of the RL agent is to maximize the expected discounted reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ following a policy π. Additionally, inspired by human visual perception, we assume the existence of an object extractor function $f : \mathcal{S} \rightarrow \mathcal{O}$ that extracts the objects in state s.[1]

Similar to OOMDP, we define a set of object classes $\mathcal{C} = \{c_1, \ldots, c_n\}$ where each class has a set of attributes $\{c.a_1, \ldots, c.a_m\}$. Each state s consists of objects $f(s) = \{o_1, \ldots, o_k\}$ where each object $o_i \in \mathcal{C}$. The state of an object is defined by the value assignment to its attributes. Finally, the state s of the underlying MDP is the union of all object states $\cup_{i=1}^{k} o_i$.

We define the interaction function $I : \mathcal{O} \times \mathcal{O} \rightarrow \{0, 1\}$ to be an indicator that determines if two objects are interacting with each other. For simplicity, we make three assumptions: first, the interaction function is known; second, objects from the same class share the same transition, reward and interaction function; and third, each object's next state is dependent on at most pairwise object interactions and action. An object's successor state is determined by a standalone transition function $T_c(o, a)$ or a pairwise transition function $T_{c_i,c_j}(o_i, o_j, a)$ if $I(o_i, o_j) = 1$.

3. Exploration with Imperfect Planning

Planning in a MDP with known dynamics can be efficiently done by focusing on promising branches of state-action tree using the UCT algorithm. However, learning a model sufficient for planning can be hard, if not impossible, for a MDP with large state space. Object representation allows us to learn a simple predictive model of the dynamics for each object class, and also allows us to perform strategic exploration (e.g. posterior sampling and optimism in the face of uncertainty).

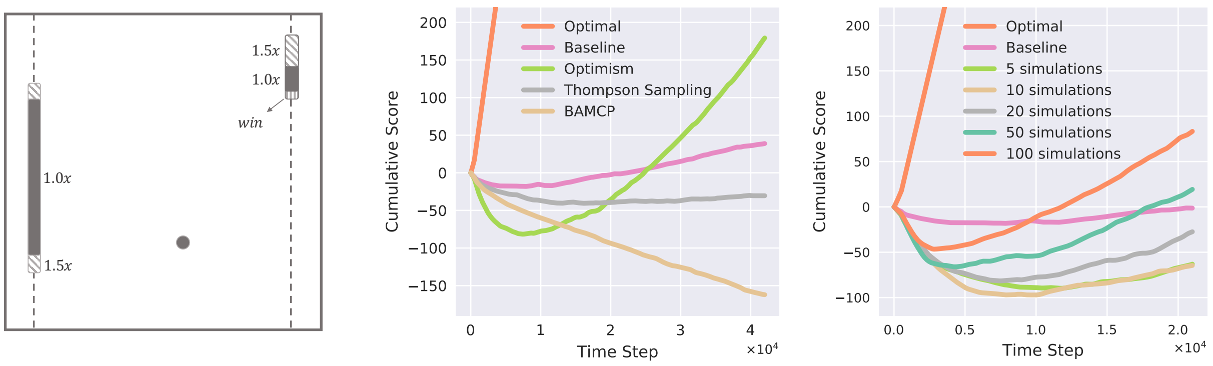
At each state s, we select the appropriate distribution over models for the corresponding object representation $f(s)$ and use UCT to pick the best action. This approach naturally lends itself to three different methods of exploration: **Thompson Sampling** (Thompson, 1933), by sampling a model at the beginning of planning; **BAMCP** (Guez et al., 2012), by sampling a model for each simulation; and **optimism based exploration** by planning multiple times each with a new sampled model and acting greedily according to the best Q value for each action across different models. We compared these methods to **Baseline**, which uses a MLE model with UCT algorithm and no exploration. Note that in all algorithms use the same input representation of the state space, which is an object level representation.

We now compare these approaches in a challenging exploration setting in order to better understand the effect of strategic exploration with approximate model-based planning. To do so, we introduce Pong Prime, a variant of the game Pong. The dynamics of this game is similar to Pong, with minor tweaks that make the game significantly harder. The enemy paddle is made 3 times larger than the player paddle, so it is impossible to score points by simply hitting the ball. Additionally, enemy and player paddles are split into two and three regions respectively, each with distinct behavior. Hitting the ball with different regions result in different speed changes to the ball. Figure 1(a) shows the speed multipliers for the regions.

In this setting, the optimal policy is to always hit the ball with the lower region, since the player instantly wins a point by doing so. The game is deterministic and model free methods with ϵ-greedy exploration (e.g. DDQN) consistently lose the game with lowest possible score across 1000 episodes using either pixel or objects' location as an input.

The correct model class for dynamics of each paddle is a linear model with 3 action history. We assume that the learned dynamics model uses this true model class (i.e. performs linear regression on the transitions we observe). Figure 1(b) compares the performance of different exploration strategies to the baseline, which performs UCT with the maximum likelihood model parameters for the linear model with the 3 step action history. We perform 500 total tree searches

[1] Function f can be any computer vision object detection algorithm.



(a)

(b)

(c)

Figure 1. (a) Pong Prime environment (b) Comparison of different exploration methods (c) Effect of planning power on optimism-based exploration

for all runs in Figure 1(b) (i.e. Thompson Sampling (TS) uses 500 simulations and 1 model, BAMCP 1 simulation and 500 models, optimism 100 simulations and 5 models).

Both BAMCP and TS perform worse than the MLE model. We hypothesize that this result is due to approximate planning, as in the limit of infinite simulations, BAMCP is guaranteed to converge to the Bayes optimal solution (Guez et al., 2012). Similarly, with full horizon planning, we should be able to compute the exact value for the model sampled with TS, and there are strong guarantees that such a method will converge to the optimal policy.

However in practice, especially in large domains or domains with real-time constraints, the amount of computation, and therefore the quality of the computed plan, will be significantly limited. In particular, if it is infeasible to use a depth that mimics the game horizon, or perhaps even to reach a local reward, then TS may suffer. This is because TS samples a single model, which means that parts of the model may be overly optimistic, while other parts may be pessimistic. Hence, when performing a limited number of simulations using MCTS, we may not go down branches of the tree that “observe” the optimistic parts of the sampled model. Therefore, the computed estimates of the Q value at the root node may not be optimistic, which is often a key part of proofs of the effectiveness of TS methods, and very helpful empirically.

BAMCP faces similar challenges, but suffers further in this domain because the true domain is deterministic. This means that for TS, optimism, and MLE approaches, the tree constructed will only have one child node (the deterministic next state) for any chosen action. In contrast, BAMCP samples a different deterministic model at each simulation, and for the same action node, those models may each deterministically predict different next states. Hence, BAMCP with M sampled models and planning horizion H, potentially builds a tree of size O($(|A|M)^H$), in contrast to the other methods that build a tree of at most size O($|A|^H$).

Optimism-based exploration significantly outperforms other approaches. We suspect it is more robust to approximate planning, since optimism is built into *every* node, allowing it to distinguish even locally between actions that may need exploration, in the absence of observing long delayed reward.

Indeed as we demonstrate in Figure 1(c) for the optimistic method, as planning power increases through more simulations, the performance of optimism-based exploration also increases. We expect that with sufficient computations the optimstic method should eventually learn the optimal policy for this domain.

4. Strategic Model Based RL

Another main challenge of performing efficient model based planning is learning accurate state and reward models. In order to learn simple yet sufficient models for planning, we use object representations to learn dynamics for each object separately (as discussed in Section 2). We also use the notion of meta-actions ("act and then wait") to simplify the model learning process. Based on the results of Section 3, we coupled Model-based RL with optimism based exploration, and evaluated our algorithm on *Pitfall!*, an environment with extremely sparse positive reward where efficient exploration is necessary.

Table 1 compares our method to other state-of-the-art algorithms. Our average score across all episodes and all runs is -281.07, which is roughly on par with count-based exploration (Bellemare et al., 2016). Our average score for the best episode across all runs in 80.52, which is higher than all scores that were reported at the time of evaluation.

Figure 2 shows an increasing number of rooms being discovered across episodes. On average, the agent discovers 17 rooms within 50 episodes. The best out of 100 runs discov-

Method	Ours‡	**Ours**$^{\dagger *}$	DQfD†	Count Based†	A3C†	DQN†
Performance	-281.07 ± 395.56	**80.52**	50.8	-259.09	-6.98	-86.85

Table 1. Comparison of our method to state-of-the-art algorithm, † is evaluation time performance, ‡ is training time performance and $^{\dagger *}$ is the average of the best episode for each run (for our algorithm). We expect $^{\dagger *}$ and † to be the best performance of each algorithm, averaged across runs. We run DQN using both objects' location and pixels as an input and reported the best result. We read the reported results for DQfD, Count based and A3C.

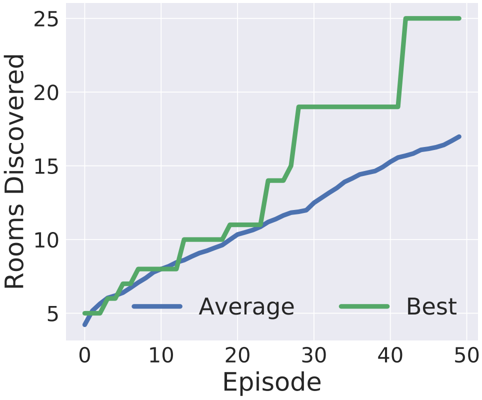


Figure 2. Number of Discovered Rooms

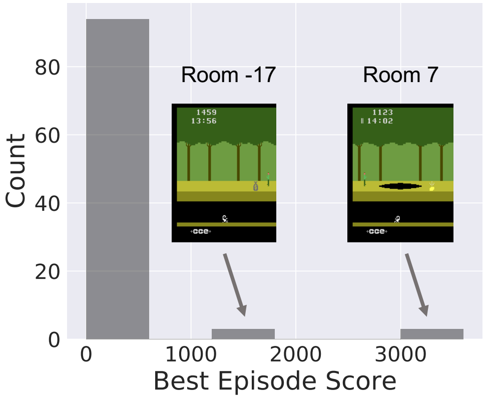


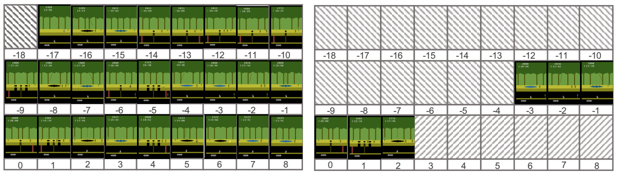
Figure 3. Histogram of Best Episode Scores

ers 25 rooms within 50 episodes. Figure 4 shows all the 26 rooms that were discovered across all 100 runs. On the other hand, Figure 4(b) shows DDQN with ϵ-greedy exploration visits at most 6 rooms with either pixels or objects' location as an input.

Moreover, Figure 3 shows that our method manages to get 6 positive rewards, 3 of which are situated in room 6 and another 3 which are situated in room -17. To the best of our knowledge, this is the first approach which manages to get positive rewards on *Pitfall!* without human demonstrations. Furthermore, from video demonstrations shown by DQfD (Hester et al., 2017), the agent seems to only get the reward in room 6 and not the reward in room -17. In comparison, our approach explores both the left and right side of the map, and gets the rewards on both sides of the map equally often.[2]

5. Conclusion and Future Work

An immediate obvious improvement would be to incorporate an estimate of the leaf node value during tree search. This addition was critical to the success and computational efficiency of earlier MCTS methods, such as on the game Go (Silver et al., 2016). Incorporating an estimate of the future value at the leaves will allow the agent to avoid prohibitive lookahead planning.



Strategic Model Based RL

(a)

DDQN with ϵ-greedy

(b)

Figure 4. (a) Rooms Discovered via Strategic Exploration (b) Rooms Discovered via DDQN with ϵ-greedy

To conclude, we presented an object-oriented framework that allows the RL agent to quickly learn to explore in an environment with large state space and sparse reward. Our work combines object oriented representation and optimism-based strategic exploration. We demonstrate that optimistic planning may be particularly beneficial when planning is necessarily approximate. We also demonstrate the first, to our knowledge, approach that can obtain positive reward on Pitfall without human demonstrations.

Acknowledgement: The research reported here was supported in part by a NSF CAREER award.

References

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International Con-*

[2] Sample videos of the agent reaching the two closest positive rewards can be found here:
`https://youtu.be/GvenPZMJiTg` (4000 reward)
`https://youtu.be/74F-ta5LyuA` (2000 reward)

ference on Machine Learning, pp. 263–272, 2017.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

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

Cobo, L. C., Isbell, C. L., and Thomaz, A. L. Object focused q-learning for autonomous agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1061–1068. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

Dann, C., Lattimore, T., and Brunskill, E. Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 5713–5723, 2017.

Diuk, C., Cohen, A., and Littman, M. L. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 240–247. ACM, 2008.

Garnelo, M., Arulkumaran, K., and Shanahan, M. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*, 2016.

Guez, A., Silver, D., and Dayan, P. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pp. 1025–1033, 2012.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Roderick, M., Grimm, C., and Tellex, S. Deep abstract q-networks. *arXiv preprint arXiv:1710.00459*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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

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

Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., and Gershman, S. J. Human learning in atari. 2017.