Exploration with Expert Policy Advice

Ashwin Khadke, Arpit Agarwal, Anahita Mohseni-Kabir, Devin Schwab
Robotics Institute, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, USA.

Abstract

Exploration for Reinforcement Learning is a challenging problem. Random exploration is often highly inefficient and in sparse reward environments may completely fail. In this work, we developed a novel method which incorporates expert advice for exploration in sparse reward environments. In our formulation, the agent has access to a set of expert policies and learns to bias its exploration based on the experts' suggested actions. By incorporating expert suggestions the agent is able to quickly learn a policy to reach rewarding states. Our method can mix and match experts' advice during an episode to reach goal states. Moreover, our formulation does not restrict the agent to any policy set. This allows us to aim for a globally optimal solution. In our experiments, we show that using expert advice indeed leads to faster exploration in challenging grid-world environments.

The field of Reinforcement Learning (RL) has made a number of major breakthroughs in recent years. Mnih et al. (2013) introduced Deep Q-Networks that successfully learned to play Atari games. More recently RL techniques have been applied to learn to play Go at human performance (Silver et al. 2016; 2017). There have also been successes in high-dimensional control applications (Heess et al. 2017). Despite these many breakthroughs, how to efficiently explore a domain, is still an open problem.

During the learning process an agent must try different actions, in order to learn both how actions affect the world and what actions should be included in the final policy. An agent that efficiently samples interesting and unique parts of the state-action space can converge to a good policy with fewer samples, an important metric when considering applications such as robotics where sample collection has a real cost.

One potential way to reduce the number of samples that must be collected, is to transfer knowledge from similar tasks that the agent has already learned. However, overly biasing with previous experience can lead an agent to miss better policies for the specific task. Moreover, if a task is sufficiently different, the bias may hurt performance.

In this work we propose an algorithm for transferring previous knowledge from agents trained in similar tasks to efficiently learn a new task. The goal is to collect samples more efficiently, while still being able to learn a good policy in the case of bad advice. Our method works by biasing the exploration strategy based on the previous policies. This exploration strategy draws parallels with Randomized Weighted Majority Algorithm (RWMA) (Littlestone and Warmuth 1994) for prediction in sequential trials with expert advice. RWMA maintains a set of weights that capture the utility of each expert's advice. Our strategy for using expert policies is similar. However, our method differs in two ways, firstly, the weights we maintain are state dependent and secondly, we use long term returns instead of immediate feedback in updating them.

We show that our algorithm speeds up policy learning on grid-world environments, Frozen Lake and Four Rooms, compared to baselines.

Related Work

A large body of work on exploration for RL exists. Some of the earlier approaches include, ϵ-greedy exploration (Sutton and Barto 1998) in which the agent randomly samples an action ϵ fraction of the time, and Boltzman distributed exploration (Thrun 1992) in which actions are sampled from the Boltzman distribution over the agent's Q-value estimates. Another idea is to perturb policy parameters with Gaussian noise and sampling actions from such a policy to explore (Plappert et al. 2017). Count based methods such as (Bellemare et al. 2016) and (Tang et al. 2017) encourage visiting infrequently visited states through an additional reward. A slightly different approach is to provide Intrinsic Motivation rewards (Chentanez, Barto, and Singh 2005). The agent builds a forward model while learning a policy and receives additional reward proportional to the error in forward model predictions. None of these methods use policies learned in similar tasks or any prior knowledge about the environment to guide exploration which is unlike our approach.

There have been works that use prior knowledge about a task to improve policy learning. This prior knowledge could be in the form of previously learned policies that achieve sub-goals potentially relevant to the current task (Sutton, Precup, and Singh 1999), or could be in the form of a task decomposition which breaks down the problem into simpler sub-problems (Dietterich 2000). However, these methods only find policies that are composed of the specified sub-policies or which adhere to the given task decomposi-

tion. Our approach uses previously learned policies simply for exploration and is capable of finding globally optimal solutions. Although the idea of using a given task decomposition or previously learned policies just for exploration has been examined in (Schwab and Ray 2017) and (Fernández and Veloso 2005) respectively. Schwab and Ray's approach only samples actions allowed by the task decomposition. Fernández and Veloso's approach, viz Policy Library, is the most similar to our work. It maintains a collection of policies and picks one to follow ϵ-greedily during an entire episode. The more reward a policy collects, the more likely it is to be picked in future episodes. Our method attempts to improve the Policy Library algorithm by allowing the agent to switch the expert policy for sampling within a single episode. This way if one expert is good for the beginning of an episode, and another expert is good for the end of an episode, our approach can leverage the information from both.

The area of transfer learning (Taylor and Stone 2009) also tries to leverage prior knowledge from similar tasks for speeding up learning in the current task. Some of the techniques reuse low-level knowledge such as Q-values (Selfridge, Sutton, and Barto 1985) and policies (Asadi and Huber 2007). Like Asadi and Huber's approach, this work attempts to leverage previous policy information. However, our work does not construct a task hierarchy, nor does it reuse the learned skills wholesale in the new environments. Learning with a task hierarchy can lead to policies with less return than globally optimal, as well as requiring the more complicated Semi-Markov Decision Process (SMDP) domain representation. Other techniques use learned high-level knowledge such as rules (Taylor and Stone 2009) and state features (Walsh, Li, and Littman 2006). Rule based systems require the behaviors of previous agents to be distilled into rules that can be evaluated and applied in new environments. But these rules may be difficult to distill. Finally, state feature approaches, such as (Walsh, Li, and Littman 2006) attempt to find similarities between states in old domain and those in the new domain, such that irrelevant changes are ignored, leading to generalization of a policy across states in different environments. Our work, attempts to use the prior expert policies directly, without finding a new state-abstraction that works across environments.

Method

We assume a standard Markov Decision Process (MDP) formulation: $M = (S, A, R, P, \gamma)$ (Puterman 2005). Where S is the set of discrete states, A is the set of discrete actions, $R : S \times A \to \mathbb{R}$ is the reward function, $P : S \times A \times S \to [0, 1]$ is the transition function, and γ is the discount factor. Furthermore, the agent has access to $e \in E$ different expert policies, $\pi_e \in \Pi_E$. Each policy corresponds to a specific MDP $m_e \in M_E$. We assume all MDPs in the set M_E have the same action space (A) and state space (S). However, the reward functions and the transition functions may vary between MDPs in the set. We also assume that the policies in Π_E are all deterministic, and that each policy π_e has an associated value function $V_e(s) \in \mathcal{V}_E$ that the agent can query when learning.

We learn a policy Π_{self} and its Q-values Q_{self} using the expert policies policies Π_E and value functions $\mathcal{V}_E$ for biasing action sampling. Our approach is to maintain a set of state dependent weights that capture the utility of suggested actions. We update these weights based on the Monte-Carlo returns observed in an episode. This approach can be used with any value-based policy learning algorithm. Here, we use Q-Learning.

Adaptive Expert Distribution Exploration

Algorithm 1, describes our method. Throughout the learning phase, the agent has a choice to either use its own Q-value estimates to pick an action or to choose an action suggested by one of the expert policies. Since our expert policies are deterministic, $V_e(s)$[1] is a relevant metric to judge the utility of the expert's recommended action in state s. Therefore, we bootstrap our estimate $\hat{V}_e(s)$ of how good an expert's suggested action is with $V_e(s)$ (Line 2). However, $V_e(s)$ is an accurate metric only if the agent is acting in the MDP m_e. For the current task, $V_e(s)$ may be an arbitrarily bad indicator of the utility of expert's suggested action. Thus, we update our estimates $\hat{V}_e(s)$ using the Monte-Carlo return obtained from state s if the agent chooses to act on the recommended action $\pi_e(s)$ (Lines 23-29). We perform this update after each episode (Line 14).

For selecting actions while learning, we compute a cumulative value $\hat{Q}$ for every action using our estimates $\hat{\mathcal{V}}_E$, and the agent's own policy (Π_{self}) and Q-value estimates (Q_{self}) (Lines 17-20). Instead of ϵ-greedy, we sample an action from a Multinomial distribution over softmax of $\hat{Q}$ (Line 21). Π_{self} and Q_{self} are updated as in Q-Learning algorithm (Line 12).

Evaluation

We first describe the environments and baseline algorithms used for evaluation and then compare the performance of our method for different sets of expert policies.

Environments

Frozen Lake In this environment (Figure 1a), we have three types of grid cells, holes, goals and normal cells and the agent is randomly spawned in a normal cell. The episode terminates if the agent enters a hole or reaches the goal. The agent gets -0.1 reward if it enters a hole, +1 reward for reaching the goal and 0 otherwise. The experts are trained to reach goals marked in blue.

Four Rooms In this environment (Figure 1b), we have four rooms separated by walls and the agent is randomly spawned at a location. The agent gets +1 reward for reaching the goal. The agent gets no other reward and the episode terminates if it reaches the goal or travels for 200 steps. The experts are trained to reach goals marked in blue.

Baselines

We implemented five baselines namely Q-Learning, Q-Learning with count-based exploration, Bootstrapped

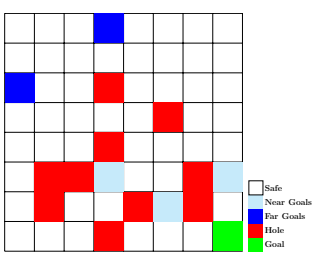
[1] Value function $V(s)$ is the expected cumulative discounted reward accrued starting from state s by following policy π.

Algorithm 1: Adaptive Expert Distribution $(\mathcal{V}_E, \Pi_E)$

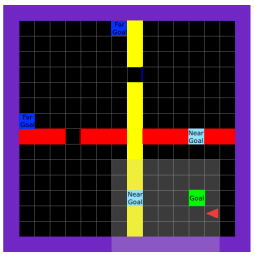
Q-Learning-with-AED $(\mathcal{V}_E, \Pi_E, \epsilon)$
  $\forall e \in E$ and $\forall s \in$ *stateSpace*, $\hat{V}_e(s) \leftarrow V_e(s)$
  $\hat{\mathcal{V}}_E \leftarrow \{\hat{V}_e | \forall e \in E\}$
  Initialize Π_{self}, Q_{self}
  for $i \in \{1, \cdots,$ *NumEpsiodes*$\}$ **do**
    Terminal $\leftarrow$ *False*, *stateVisits* $\leftarrow []$
    $s \leftarrow$ Reset()
    while *not* *Terminal* **do**
      $a \leftarrow$ SelectAction(s, $\hat{\mathcal{V}}_E$, Π_E, Π_{self}, Q_{self})
      r, s', *Terminal* $\leftarrow$ Step(s, a)
      stateVisits.append((s, a, r))
      UpdatePolicy(Π_{self}, Q_{self}, s, a, s', r, *Terminal*)
      $s \leftarrow s'$
    UpdateExpertValues(*stateVisits*, $\hat{\mathcal{V}}_E$, Π_E)
  return Π_{self}
SelectAction (s, $\mathcal{V}_E$, Π_E, Π_{self}, Q_{self})
  for $a \in$ *actionSpace* **do**
    $\hat{Q}(s, a) \leftarrow \sum_{\pi_e(s)=a, \pi_e \in \Pi_E} \hat{V}_e(s)$
  $a' \sim \Pi_{self}(s)$
  $\hat{Q}(s, a') \leftarrow \hat{Q}(s, a') + Q_{self}(s, a')$
  $a \sim \text{Mult}\Big(\big(\frac{e^{\hat{Q}(s,a_1)}}{\sum_{a''} e^{\hat{Q}(s,a'')}}, \cdots, \frac{e^{\hat{Q}(s,a_n)}}{\sum_{a''} e^{\hat{Q}(s,a'')}}\big)\Big)$
  return a
UpdateExpertValues (*stateVisits*, $\hat{\mathcal{V}}_E$, Π_E)
  $R \leftarrow 0$
  while *not* *Empty(stateVisits)* **do**
    $(s, a, r) \leftarrow$ *stateVisits*.popBack()
    $R \leftarrow r + \gamma R$
    if $\pi_e(s) == a$ **then**
      $\hat{V}_e(s) \leftarrow \alpha(R - \hat{V}_e(s))$

Q-Learning, Policy Library and a Fixed Expert Distribution version of our algorithm to show the improvements obtained by our approach.

- *Q-Learning* is the base algorithm for our approach.
- *Q-Learning with count-based exploration* assigns additional rewards to actions proportional to $1/\sqrt{n}$, where n is the number of times an action is chosen from a particular state. Moreover, it initializes Q-values to optimistic guesses rather than starting with zero.
- *Bootstrapped Q-Learning* initializes Q-values with those provided by one of the experts and learns a policy similar to vanilla Q-Learning. We picked the expert, bootstrapping with who's Q-values, led to fastest policy learning to compare against.
- *Policy Library* either picks its current policy or an expert policy and follows it ϵ-greedily to sample actions in an episode. If experts are bad, the algorithm defaults to vanilla Q-Learning.
- *Fixed Expert Distribution* uses the same action selection strategy as our algorithm but keeps $\hat{V}_e \in \hat{\mathcal{V}}_E$ fixed throughout the learning.



(a) Frozen lake environment.



(b) Four Rooms environment.

Figure 1: Test Environments

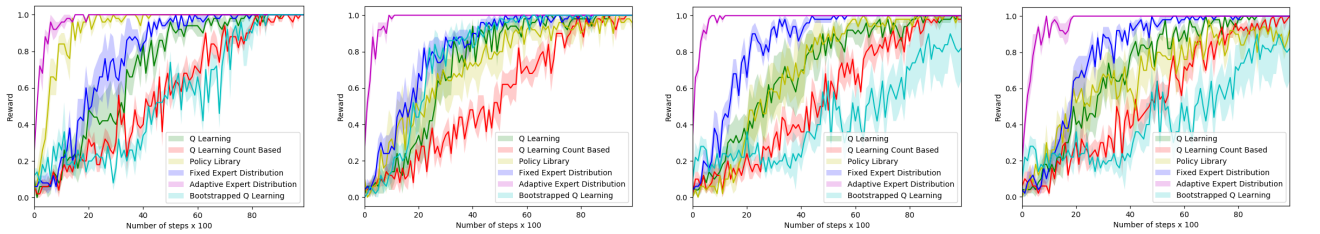
Experiments

Near Goal Experts In this experiment, we use the experts trained to reach locations that are close to the actual goal (Figure 1, marked in light blue). Getting the experts which are trained to reach slightly different goal location or halfway towards the goal are very common in robotic navigation and manipulation domains.

Figure 2a shows that our algorithm is able to explore faster than all the baselines when we have useful expert advice. Unlike Policy Library, our approach can switch between experts in a single episode. The Fixed Expert Distribution algorithm is also able to learn faster than Q-Learning. Q-Learning with count based exploration gives incentives for exploring unvisited states and thus is slow to learn. Our method is able to perform better than the baselines in a larger environment such as Four Rooms (Figure 3a).

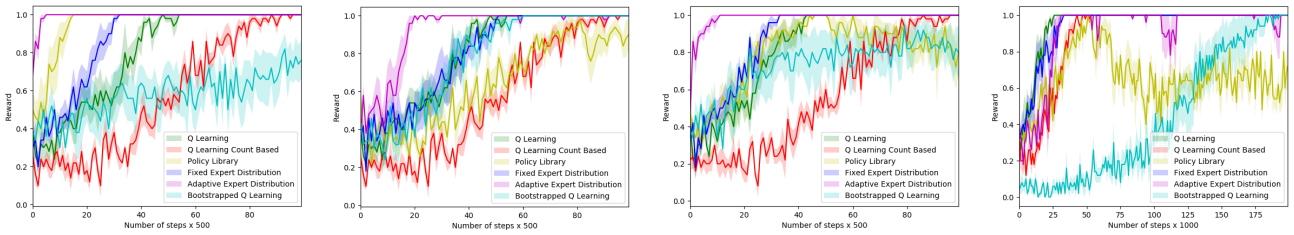
Constant Action Experts In this experiment, we have four experts. One suggests action 'left' in each state. Similarly, other three suggest 'right', 'up' and 'down' in each state respectively. We set $V_e(s) = 0$ for all these experts for every state s. Figures 2b and 3b depict the results with constant action experts on Frozen Lake and Four Rooms respectively. Our approach outperforms the baselines. Even when the experts do not provide much useful information to the agent, our method is able to quickly learn a good policy for the environments. We believe part of the reason that our algorithm outperforms ϵ-greedy is because it biases actions that return better Monte-Carlo returns. Policy Library performs very badly with such experts because it sticks with one policy to sample throughout the episode.

Near Goal + Constant Action Experts In this experiment, we combined the expert sets from the previous two settings to test how our algorithm performs when the number of experts increases. The obtained results as shown in Figures 2c and 3c, indicate that our algorithm is able to choose the good experts to follow when exploring, even if we have other experts to choose from. However, the Fixed Expert Distribution algorithm maintains similar performance as in the previous experiment. This is because the sampling distribution remains constant unlike our algorithm which modifies the sampling distribution based on episodic returns.



(a) Near Goal Experts (b) Constant Action Experts (c) Near Goal + Constant Action Experts (d) Far Goal + Constant Action Experts

Figure 2: Results for Frozen Lake environment.



(a) Near Goal Experts (b) Constant Action Experts (c) Near Goal + Constant Action Experts (d) Far Goal + Constant Action Experts

Figure 3: Results for Four Rooms environment.

Far Goal + Constant Action Experts In this experiment, we use experts trained to reach goal locations far away from the actual goal (Figure 1, marked in dark blue) as well as experts that suggest the same action in each state. In this setting no expert provides useful advice. Moreover, acting on the suggested actions by some of these experts would actually drive the agent away from the goal. Figures 2d and 3d show that even when advice is bad our approach is better than, if not at par with, vanilla Q-Learning

Future work and conclusion

Through various experiments we showed that our Adaptive Expert Distribution algorithm quickly learns the optimal policy in presence of experts. Our work is orthogonal to policy learning algorithms. Therefore our exploration process could be potentially used to mix best training algorithm on different MDPs. Our algorithm is also additive in terms of experts. Therefore we can potentially keep growing our expert set to perform better exploration as the complexity of our task improve. In future work, we are planning to work on following ideas:

- Learn in continuous state and action domains while incorporating expert advice;
- Extend the formulation to accommodate stochastic expert policies;
- Investigate the idea of safe exploration using expert policies that warn against taking damaging actions such as running into walls;
- Apply our approach to use policies learned in simulation as experts while learning on a robotic system;
- Address the problem when experts are trained on different state spaces.

References

Asadi, M., and Huber, M. 2007. Effective control knowledge transfer through learning skill and representation hierarchies. In *IJCAI*, volume 7, 2054–2059.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 1471–1479.

Chentanez, N.; Barto, A. G.; and Singh, S. P. 2005. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*. MIT Press. 1281–1288.

Dietterich, T. G. 2000. Hierarchical Reinforcement Learning with MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research* 13:227–303.

Fernández, F., and Veloso, M. 2005. Building a library of policies through policy reuse. Technical report, Carnegie Mellon University, School of Computer Science.

Heess, N.; TB, D.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S. M. A.; Riedmiller, M.; and Silver, D. 2017. Emergence of locomotion behaviours in rich environments. *CoRR*.

Littlestone, N., and Warmuth, M. 1994. The weighted majority algorithm. *Information and Computation* 108(2):212 – 261.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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.

Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R. Y.; Chen, X.; Asfour, T.; Abbeel, P.; and Andrychowicz, M. 2017. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*.

Puterman, M. L. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ USA: John Wiley & Sons, Inc.

Schwab, D., and Ray, S. 2017. Offline reinforcement learning with task hierarchies. *Machine Learning* 106(9-10):1569–1598.

Selfridge, O. G.; Sutton, R. S.; and Barto, A. G. 1985. Training and tracking in robotics. In *IJCAI*, 670–672.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

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.

Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, O. X.; Duan, Y.; Schulman, J.; DeTurck, F.; and Abbeel, P. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, 2750–2759.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(Jul):1633–1685.

Thrun, S. B. 1992. Efficient exploration in reinforcement learning.

Walsh, T. J.; Li, L.; and Littman, M. L. 2006. Transferring state abstractions between mdps. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*.