Balancing Exploration and Exploitation in Self-imitation Learning

Chun-Yao Kang^(✉) and Ming-Syan Chen

National Taiwan University, Taipei, Taiwan
cykang@arbor.ee.ntu.edu.tw, mschen@ntu.edu.tw

Abstract. Sparse reward tasks are always challenging in reinforcement learning. Learning such tasks requires both efficient exploitation and exploration to reduce the sample complexity. One line of research called self-imitation learning is recently proposed, which encourages the agent to do more exploitation by imitating past good trajectories. Exploration bonuses, however, is another line of research which enhances exploration by producing intrinsic reward when the agent visits novel states. In this paper, we introduce a novel framework *Explore-then-Exploit* (EE), which interleaves self-imitation learning with an exploration bonus to strengthen the effect of these two algorithms. In the exploring stage, with the aid of intrinsic reward, the agent tends to explore unseen states and occasionally collect high rewarding experiences, while in the self-imitating stage, the agent learns to consistently reproduce such experiences and thus provides a better starting point for subsequent stages. Our result shows that EE achieves superior or comparable performance on variants of MuJoCo environments with episodic reward settings.

Keywords: Reinforcement learning · Self-imitation learning · Exploration · Sparse reward

1 Introduction

Reinforcement learning (RL) learns an optimal policy by maximizing the expected return. These methods work well in environments with dense rewards but suffer from degenerate performance when the rewards are sparse, which means rewards are almost zero during an episode. In such cases, the agent requires both an efficient exploration method that guides itself to find potentially useful information, and an efficient exploitation technique to make better use of these experiences.

Exploration bonus is a simple way for directed exploration, which generates intrinsic rewards every step even when the external rewards are unavailable. This bonus is designed to be higher in novel states than in those visited frequently, and thus encourages the agent to explore new behavior. Even though such method still requires a tremendous amount of time to train, and the intrinsic rewards may vanish when the policy converges to a local optimum.

Another method called self-imitation learning is a recently introduced algorithm that enhances exploitation by storing and reusing useful experiences.

Once the agent occasionally generates high rewarding trajectories, they are collected and stored in a replay buffer. The policy is then trained to imitate those trajectories in the buffer and thus the resulting agent consistently reproduces past good behavior. This method is shown to have high sample efficiency, though it hurts the exploration and has a chance to stuck at local optima. Besides, self-imitation learning relies on other techniques to bootstrap the process before the first trajectory is added to the buffer.

In this paper, we propose a framework *Explore-then-Exploit* (EE) which combines random network distillation (RND) [5], a kind of exploration bonus, and generative adversarial self-imitation learning (GASIL) [7]. By integrating these two methods, the RND bonus solves the initial bootstrap problem of self-imitation and potentially prevents the policy from getting stuck at local optima. On the other hand, GASIL speeds up the convergence of the policy and provides good starting points for later exploration. Nevertheless, a direct composition does not make sense due to that the agent will prefer unpredictable actions with exploration bonuses while tend to reproduce past behaviors with self-imitation. Mixing these two objectives will confuse the agent and lead to an undesired result. Instead, we suggest to combine them in an interleaving manner. By doing so, the agent only learns one concept at each stage and switch to another one when certain criteria are reached. We formulate our framework in the form of reward interpolation and provide some heuristic methods to determine the weight between exploration and self-imitation. Finally, we evaluate our model on several MuJoCo tasks [19] with episodic rewards and empirically show that EE improves over GASIL or RND in most environments.

2 Related Works

Exploration. There have been many researchers working on exploration in RL. Count-based exploration gives a reward to rarely visited states [2, 13]. Prediction-based exploration, also known as the curiosity-driven method, predicts the agent’s dynamics and treats the prediction error as intrinsic reward [1, 4, 14, 18]. Random network distillation (RND) [5] further improves by utilizing two neural networks: a randomly initialized target network, and a predictor network trained to minimize the difference between the outputs of the two networks. The difference is then used as the exploration bonus.

Self-imitation. Self-imitation learning (SIL) [12] was recently proposed to exploit past good behaviors. This algorithm stores the transitions in a replay memory, and uses them to update the policy when the stored return is higher than the current state value estimation. Generative adversarial self-imitation learning (GASIL) [7] is a generative adversarial extension to SIL, which instead stores top-k experiences based on episode returns and formulates it as a divergence minimization problem. Combined with actor-critic methods, GASIL learns a shaped, dense reward function that can be used as an extra reward signal. Another method [6] also utilizes the generative adversarial structure, but instead

trains an ensemble of agents and uses a special optimization technique to guarantee the diversity among these agents. This work focuses on the interaction between multiple agents, while our work only considers one agent. This technique can be used simultaneously with our framework without problems.

Amplifying the Imitation Effect. In a very recent work, AIE [10] was proposed to combine RND and SIL, which has similar idea as our method. This work combines these two algorithms directly and introduces several techniques to enhance the effect of RND, whereas ours integrates GASIL in an interleaving fashion, and evaluates it on common RL benchmarks.

3 Background

3.1 Reinforcement Learning

Consider a state space $\mathcal{S}$ and an action space $\mathcal{A}$. The purpose of RL is to find a parameterized policy $\pi_\theta(a|s)$ where $s \in \mathcal{S}$ and $a \in \mathcal{A}$, which maximizes the expected discounted sum of rewards: $\eta(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ where $\gamma \in (0, 1]$ is the discount factor and r_t is the reward at time t . The objective is non differentiable and hence requires the technique called policy gradient to estimate the gradient. A commonly used gradient estimator of the objective $\eta(\pi_\theta)$ is given by

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t], \quad (1)$$

where $\hat{A}_t$ is the advantage estimation at time t . The estimator $\nabla_\theta \eta(\pi_\theta)$ can be obtained by differentiating the surrogate objective

$$\mathcal{L}^{\text{PG}}(\theta) = \mathbb{E}_\pi[\log \pi_\theta(a_t|s_t) \hat{A}_t]. \quad (2)$$

3.2 Generative Adversarial Self-imitation Learning

GASIL involves a good trajectory buffer $\mathcal{B} = \{\tau_i\}$ and a discriminator $D_\phi(s, a)$. The buffer $\mathcal{B}$ stores the top-k good trajectories according to the total trajectory reward $R = \sum_{t=0}^{\infty} r_t$, where each trajectory τ_i consists of a sequence of states and actions $\{s_0, a_0, s_1, a_1, \dots, s_t, a_t\}$. The algorithm treats the trajectories stored in the buffer $\mathcal{B}$ as the expert demonstrations, and utilizes the GAIL framework [8] to obtain a similar generative adversarial loss

$$\begin{aligned} \underset{\theta}{\operatorname{argmin}} \underset{\phi}{\operatorname{argmax}} \mathcal{L}^{\text{GASIL}}(\theta, \phi) &= \mathbb{E}_{\tau_\pi}[\log D_\phi(s, a)] \\ &+ \mathbb{E}_{\tau_E \sim \mathcal{B}}[\log (1 - D_\phi(s, a))] - \lambda \mathcal{H}(\pi_\theta) \end{aligned} \quad (3)$$

where τ_π, τ_E are the trajectories sampled from the policy π_θ and the buffer $\mathcal{B}$ respectively, and $\lambda \mathcal{H}(\pi_\theta)$ is the entropy regularization term. The discriminator D_ϕ is updated via

$$\nabla_\phi \mathcal{L}^{\text{GASIL}} = \mathbb{E}_{\tau_\pi}[\nabla_\phi \log D_\phi(s, a)] + \mathbb{E}_{\tau_E}[\nabla_\phi \log (1 - D_\phi(s, a))] \quad (4)$$

and the policy π_θ is updated via the approximate gradient

$$\begin{aligned} \nabla_\theta \mathcal{L}^{\text{GASIL}} &= \mathbb{E}_{\tau_\pi} [\nabla_\theta \log \pi_\theta(a|s) Q(s, a)] - \lambda \nabla_\theta \mathcal{H}(\pi_\theta), \\ \text{where } Q(s_t, a_t) &= -\mathbb{E}_{\tau_\pi} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \log D_\phi(s'_t, a'_t) \right]. \end{aligned} \quad (5)$$

The Eq. (5) has a similar form as policy gradient (1), and thus can be combined together as follows:

$$\nabla_\theta \mathcal{L}^{\text{PG}} + \alpha \nabla_\theta \mathcal{L}^{\text{GASIL}} = \mathbb{E}_{\tau_\pi} [\nabla_\theta \log \pi_\theta(a|s) \hat{A}_t^\alpha] + \lambda \nabla_\theta \mathcal{H}(\pi_\theta) \quad (6)$$

where $\hat{A}_t^\alpha$ is the advantage estimation by replacing $r(s, a)$ with a modified reward $r(s, a) - \alpha \log D_\phi(s, a)$. The extra term $-\log D_\phi(s, a)$ can be seen as an intrinsic reward signal generated by the discriminator to encourage the agent to imitate the past behaviors.

3.3 Random Network Distillation

RND introduces a fixed and randomly initialized target network $f : \mathcal{S} \rightarrow \mathbb{R}^k$, which takes a state as input and outputs a k -dimensional embedding, together with a predictor network $\hat{f} : \mathcal{S} \rightarrow \mathbb{R}^k$, which is trained to minimize the MSE $\|\hat{f}(s; \theta) - f(s)\|^2$. The exploration bonus is defined as the prediction error $\|\hat{f}(s) - f(s)\|^2$, which is expected to be lower for the states similar to the frequently visited ones.

4 Explore-then-Exploit Framework

Our EE framework incorporates the exploration bonus component into the GASIL structure. We choose RND as the exploration algorithm because of the simplicity of implementation. Both GASIL and RND generate extra reward signals, which allows us to formulate our framework as an interpolation of the rewards from three different sources: (a) the external environment reward r^{ext} , (b) the imitation bonus $r^{\text{im}} = -\log D_\phi(s, a)$ which is derived from the discriminator and (c) the exploration bonus $r^{\text{exp}} = \|\hat{f}(s) - f(s)\|^2$ which is given by the predictor network. It does not make sense to directly sum up these rewards, as the imitation bonus and exploration bonus guides the agent to different directions. Instead, we use an dynamically adaptive reward

$$r = (1 - \nu) r^{\text{im}} + \nu (r^{\text{ext}} + r^{\text{exp}}) \quad (7)$$

where ν controls the ratio between exploration and self-imitation. The parameter ν is explicitly assigned to 0 or 1 to prevent these two terms from interfering with each other. In the exploration stage, we set $\nu = 1$ to completely eliminate the effect of self-imitation, which allows the agent to freely explore the environment. While in the self-imitation stage, we set $\nu = 0$ to make the agent purely rely on

Fig. 1. Square wave with period 10^6 and duty cycle 25%.

Fig. 2. Heuristic method which sets $\nu = 1$ for the first 3×10^5 steps and sets $\nu = 0$ for the remaining time

the imitation bonus. As such, the agent will quickly converge to a local optimum and begin to explore again when it switches back to the exploration stage.

It is crucial to determine when to assign the ratio ν to 0, which implies the self-imitation stage, or vice versa. In this work, we introduce a heuristic method that works well in the underlying benchmarks. We assign the ratio ν as a square wave with period T and duty cycle d . A particular setting $T = 10^6$ and $d = 25\%$ is used throughout the paper, which is shown in Fig. 1. Another method is to set the agent in the exploration stage for certain steps in the beginning and switch to the self-imitation stage for the remaining time, as shown in Fig. 2. We found that by doing so, the agent often leads to superior performance.

Our framework only involves the reward interpolation, and thus can be plugged into any actor-critic based algorithm such as A2C [11] or PPO [16]. We demonstrate the combination of our method with PPO in Algorithm 1. Note that the rewards used to determine the ranking of the trajectories stored in the replay buffer do not include the exploration bonuses. More specifically, the total trajectory reward is defined as $R = \sum_{t=0}^{\infty} r_t^{\text{ext}}$.

Algorithm 1. *Explore-then-Exploit*

```

Initialize network  $\theta_\pi, \phi, \theta_f, \theta_{\hat{f}}$ 
Initialize top-k replay buffer  $\mathcal{B} \leftarrow \emptyset$ 
Initialize episode buffer  $\mathcal{E} \leftarrow \emptyset$ 
Initialize  $\nu \leftarrow 1$ 
 $M \leftarrow$  number of initial steps for initializing observation normalization
for  $m \leftarrow 1, M$  do
    Sample  $a_t$  uniformly from  $\mathcal{A}$ 
    Generate a step transition  $s_t, a_t, s_{t+1}$ 
    Update observation normalization parameters using  $s_{t+1}$ 
end for
for each iteration do
    for each step do
        Execute an action  $s_t, a_t, r_t^{\text{ext}}, s_{t+1} \sim \pi_\theta(a_t|s_t)$ 
        Calculate the imitation bonus  $r_t^{\text{im}} = -\log D_\phi(s_t, a_t)$ 
        Calculate the exploration bonus  $r_t^{\text{exp}} = \|\hat{f}(s_{t+1}) - f(s_{t+1})\|^2$ 
        Normalize the intrinsic reward  $r_t^{\text{exp}}$  and update the running estimate of reward
        standard deviation
        Calculate the integrated reward  $r_t = (1 - \nu) r_t^{\text{im}} + \nu (r_t^{\text{ext}} + r_t^{\text{exp}})$ 
        Store the transition  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, a_t, r_t)\}$ 
    end for
    Update  $\mathcal{B}$  by the full trajectory stored in  $\mathcal{E}$  associated with the episode reward
     $R = \sum_{t=0}^{\infty} r_t^{\text{ext}}$ 
    Calculate the returns and the advantages
    Calculate the ratio  $\nu$  for the next iteration according to predefined rules
    Update observation normalization parameters using  $\mathcal{E}$ 
    for each minibatch do
        Update  $\theta_\pi$  wrt PPO objective on  $\mathcal{E}$ 
        Update  $\theta_{\hat{f}}$  wrt RND loss on  $\mathcal{E}$ 
    end for
    for each epoch do
        Sample a minibatch  $s_1$  from  $\mathcal{B}$ 
        Sample a minibatch  $s_2$  from  $\mathcal{E}$ 
        Update  $\phi$  wrt discriminator loss (Eq. 4) using  $s_1$  and  $s_2$ 
    end for
    Clear episode bufer  $\mathcal{E} \leftarrow \emptyset$ 
end for

```

5 Experiments and Results

The experiments are designed to answer the following questions:

1. Is EE better than running RND or GASIL alone?
2. Is the RND exploration bonus itself necessary, or a random exploration also works?
3. Is the effect of interleaving fashion critical?

5.1 Implementation Details

We evaluated our method on several OpenAI Gym [3] MuJoCo continuous control tasks. The specs of environments used are listed in Table 1. All of the benchmarks were modified as episodic reward environments, which means that rather than providing the per timestep reward r_t , we provided the whole episode reward $R = \sum_{t=0}^{\infty} r_t$ at the last step of an episode and zero rewards in other steps.

Table 1. State and action space of OpenAI Gym MuJoCo tasks

Environment	State dimension	Action dimension
Walker2d	17	6
Swimmer	8	2
Hopper	11	3
HalfCheetah	17	6
Ant	111	8
Humanoid	376	17

We implemented the following agents based on this PPO implementation [17]:

- *PPO*: The proximal policy optimization [16] baseline.
- *PPO + RND*: PPO combined with RND bonus [5].
- *PPO + GASIL*: PPO combined with GASIL [7].
- *EE_interval*: Our method where ν is assigned to be the square wave with period 10^6 and duty cycle 25%.
- *EE_first_exp*: Our method where ν is assigned to be 1 for the first 3×10^5 steps and to be 0 for the remaining steps.

The hyperparameters used in our experiments are shown in Table 2. Every feed-forward networks including the actor-critic network, the discriminator and the predictor has 2 hidden layers with 64 neurons. Note that the parameter α is only used in PPO + GASIL, not in EE.

Table 2. EE hyperparameters on MuJoCo.

Hyperparameter	Value
Rollout length	2048
Discount factor γ	0.99
GAE [15] parameter	0.95
Optimization algorithm	Adam [9] with learning rate 0.0003
Optimization epochs of PPO	10
Optimization epochs of RND	1
Minibatch size	64
Entropy regularization coefficient	0
PPO clip range	[0.8, 1.1]
Discriminator minibatch size	128
Number of discriminator updates	5
Size of replay buffer $\mathcal{B}$	10
Scale of imitation bonus α used in GASIL	0.8

5.2 Episodic MuJoCo

We first evaluated 5 types of agents on 6 MuJoCo tasks. The result in Fig. 3 shows that *EE* performs better than all of the baseline on *Walker2d*, *Hopper* and *HalfCheetah*, and performs comparably with *GASIL* on *Swimmer*. This is because the *Swimmer* task is relatively simple that exploration is not even necessary. However, on more complicated tasks such as *Walker2d* and *Hopper*, the benefit of integrating exploration and self-imitation is significant. For *Ant* and *Humanoid*, all of the 5 agents fail to learn a meaningful policy. This is mainly due to the high dimensions of the observation space, which makes the networks difficult to train.

In Fig. 3, we see that the *EE_interval* has an obvious performance drop when the agent is in the exploration stage and begins to climb again when it switches back to the self-imitation stage. This is the expected behavior since the agent tends to select unpredictable behavior, which potentially causes early termination of an episode. Unfortunately, *EE_interval* performs slightly worse than *EE_first.exp*, which seems to indicate that the interleaving one does not have the advantage over the non-interleaving one. One possible reason is that MuJoCo environments do not have the sequentially dependent structure, which means that reliably producing certain rewards does not make it easier to obtain the subsequent rewards. In this case, it is not beneficial at all to first converge to a good policy and then begin to explore from that state.

Fig. 3. Learning curves for PPO, RND, GASIL, and our method EE with two different scheduling on 6 OpenAI MuJoCo tasks with episodic rewards. Mean and standard deviation over 5 random seeds are plotted.

5.3 Effect of RND

In Fig. 3, it should be noticed that adding the RND bonus alone does not take any notable effect, which gives rise to the question of the effectiveness of RND. We carried out another experiment to investigate this phenomenon. We modified the behavior of *EE_first_exp* agent in the exploration stage as follows: *no exp* indicates that the RND bonus is removed from the reward interpolation, which means that the agent only relies on external reward r^{ext} ; *random exp* indicates that the agent always takes a random action; *EE_first_exp* remains unchanged. The result in Fig. 4 shows that integrating GASIL with RND indeed amplifies the effect of exploration compared to a purely random one.

Fig. 4. Learning curves for EE with 3 different exploration behaviors on 3 MuJoCo tasks. Mean and standard deviation over 5 random seeds are plotted.

5.4 Direct Interpolation

To justify the statement that directly mixing the imitation bonus and exploration bonus results in poor performance, we modified the reward interpolation to be $r = \alpha r^{\text{im}} + (1 - \alpha) r^{\text{ext}} + \beta r^{\text{exp}}$ where α was fixed at 0.8 throughout the experiment, and β coefficient was set to be $\{1, 2, 4, 8, 16\}$ respectively. This agent is referred to as *direct*. Figure 5 shows that *direct* method with $\beta = 1$ performs much the same as *GASIL*, which points out the fact that imitation bonus is likely to dominate the outcoming policy when given similar weights. Furthermore, the performance drops when setting higher weights on exploration bonus. This result again demonstrates that mixing different behavior will bring about inferior performance.

Fig. 5. Learning curves for agents with two types of reward interpolation trained on 3 MuJoCo tasks. Mean and standard deviation over 5 random seeds are plotted.

6 Conclusion

In this paper, we proposed *Explore then Exploit* (EE), a novel framework that combines GASIL and RND in a form of reward interpolation, and provided a heuristic way to interleaves between exploration and self-imitation stage. We demonstrated that EE significantly improves over existing single-agent methods on several continuous control tasks with episodic rewards. We also empirically justified our hypothesis that separating the objectives of exploration and imitation is better than mixing them together. Developing appropriate ways to automatically adjust the ratio between exploration and imitation will be an important future work. Further, we will apply our framework to more complicated environments such as Atari.

Acknowledgements. The authors would like to thank Dr. Kuan-Ting Lai for his helpful comments which improve the presentation of this paper.

References

1. Achiam, J., Sastry, S.: Surprise-based intrinsic motivation for deep reinforcement learning. arXiv preprint [arXiv:1703.01732](https://arxiv.org/abs/1703.01732) (2017)
2. Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R.: Unifying count-based exploration and intrinsic motivation. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 29, pp. 1471–1479. Curran Associates, Inc. (2016). <http://papers.nips.cc/paper/6383-unifying-count-based-exploration-and-intrinsic-motivation.pdf>
3. Brockman, G., et al.: Openai gym (2016)
4. Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. arXiv preprint [arXiv:1808.04355](https://arxiv.org/abs/1808.04355) (2018)
5. Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. arXiv preprint [arXiv:1810.12894](https://arxiv.org/abs/1810.12894) (2018)
6. Gangwani, T., Liu, Q., Peng, J.: Learning self-imitating diverse policies. arXiv preprint [arXiv:1805.10309](https://arxiv.org/abs/1805.10309) (2018)
7. Guo, Y., Oh, J., Singh, S., Lee, H.: Generative adversarial self-imitation learning. arXiv preprint [arXiv:1812.00950](https://arxiv.org/abs/1812.00950) (2018)
8. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: *Advances in Neural Information Processing Systems*, pp. 4565–4573 (2016)
9. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
10. Lee, G.T., Kim, C.O.: Amplifying the imitation effect for reinforcement learning of UCAV’s mission execution. arXiv preprint [arXiv:1901.05856](https://arxiv.org/abs/1901.05856) (2019)
11. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*, pp. 1928–1937 (2016)
12. Oh, J., Guo, Y., Singh, S., Lee, H.: Self-imitation learning. In: *International Conference on Machine Learning*, pp. 3875–3884 (2018)
13. Ostrovski, G., Bellemare, M.G., van den Oord, A., Munos, R.: Count-based exploration with neural density models. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML 2017*, pp. 2721–2730 (2017). [JMLR.org](https://jmlr.org/). <http://dl.acm.org/citation.cfm?id=3305890.3305962>
14. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017
15. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint [arXiv:1506.02438](https://arxiv.org/abs/1506.02438) (2015)
16. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)
17. Shangdong, Z.: Modularized implementation of deep RL algorithms in PyTorch (2018). <https://github.com/ShangdongZhang/DeepRL>
18. Stadie, B.C., Levine, S., Abbeel, P.: Incentivizing exploration in reinforcement learning with deep predictive models. arXiv preprint [arXiv:1507.00814](https://arxiv.org/abs/1507.00814) (2015)
19. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: a physics engine for model-based control. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE (2012)