Multi-Agent Exploration with Random Network Distillation

Philip Raeisghasem

Abstract—A significant part of reinforcement learning is ensuring adequate exploration. An agent cannot learn how good a new strategy is if the strategy is never tried. For multi-agent tasks, the space of possible configurations grows exponentially with the number of agents. This larger space is much harder to explore. In this paper, we apply the recently proposed Random Network Distillation exploration method to a multi-agent setting. We compare this approach to other exploration methods in the multi-agent setting. We do this both by explicitly quantifying amount of exploration and by analyzing downstream performance.

Index Terms—reinforcement learning, multi-agent, exploration, random network distillation.

1 Introduction

Reinforcement learning is a paradigm of machine learning alongside supervised and unsupervised learning. It is most applicable to problems that can be framed as an agent acting within an environment to achieve some goal. Like supervised learning, reinforcement learning happens through a feedback mechanism during an initial training phase. Unlike in supervised learning, this feedback is not in the form of 'labeled' data. There are no simple 'correct' answers to directly compare the model's performance against. Instead, the feedback loop in reinforcement learning is between the agent and the environment by way of a reward function. Broadly speaking, instead of learning whether its actions are 'right' or 'wrong', the agent learns how good its actions are.

This difference in feedback mechanism is important for allowing reinforcement learning models to learn from self-generated experience. Thanks to the advent of deep neural networks, the space of problems solvable by artificial intelligence and machine learning has exploded in size. Many of these problems, however, require much more data to solve. In supervised learning, vast datasets that require significant preprocessing are the norm, and a human is needed to label every single training example. In contrast, a human often need only specify the reward function for a given environment in reinforcement learning.

While for many problems it can be challenging to design a reward function that will lead to desired behavior, a large class of problems are amenable to very simple and sparse reward functions. If the problem is a two-player game, for example, the agent could only be rewarded at the very end, depending on whether it won the game. Using these sparse reward functions comes at the cost of the agent receiving less feedback. That is, there is a tradeoff between difficulty and human effort in embedding prior knowledge. This tradeoff is not unique to reinforcement learning, but it has given rise to unique methods in reinforcement learning for addressing the issue.

In particular, using a sparse reward function exacerbates the existing problem of exploration. While learning, an agent must always be deciding how confident it is in its current strategy. If it is very confident in the information it has received so far, it will choose to act in ways that have given it large rewards in the past. The agent is said to be exploiting its current knowledge. If the agent is not very confident, it may decide to try something new in order to gather information. The agent is then said to be exploring the environment. A sparse reward function lessens the effectiveness of exploration by limiting the number of times an agent is given any feedback after exploratory actions.

The difficulty of exploration depends on more

• *Philip Raeisghasem is with Louisiana Tech University, Department of Cyberspace Engineering. E-mail: par019@latech.edu*

than just the sparsity of the reward function, of course. Other important factors are the size and complexity of the environment as well as the number of agents acting concurrently. This last factor, the number of agents, is especially impactful. An environment's reward is a function of the states occupied and the actions taken at any given time step. If there are $|S|$ states and $|A|$ actions available in an environment with n agents, then the input space of the reward function is of size $|S|^n|A|^n$. For problems with large state or action spaces, exploration in a multi-agent setting quickly becomes a daunting task.

In this paper, we investigate the exploration of a particularly challenging class of environments for which normal methods are inadequate. We develop new methods to help make these environments tractable, and we compare their effectiveness against an existing technique.

Namely, we compare our methods to exploration by random network distillation (RND) as formulated in [1]. This method is one way of encouraging an agent to seek out states that are "surprising". Roughly speaking, an intrinsic reward bonus is given to the agent for visiting states that it has visited few times before. This intrinsic reward, when combined with the (potentially sparse) extrinsic environment reward, has been shown to result in a more thorough exploration of the environment by a single agent.

We analyze the simulated data generated by multiple agents concurrently exploring with both methods. The environment we test with is a simple 2D environment. The metrics we gather include a direct measure of exploration as well as the total reward received over time. We also collect these metrics for other exploration methods for the sake of comparison.

2 Background/Review of Literature

2.1 Single-Agent Tasks

A reinforcement learning problem is formalized as a Markov Decision Process (MDP). Much of the notation below is taken from [11]. An MDP $\mathcal{M}$ is defined as a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$. Here, $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, and $\mathcal{R}$ is a set of rewards. The dynamics function $\mathcal{P} : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ assigns probabilities to states and rewards given an agent's action and previous state. It is generally stochastic and written as a conditional probability distribution $p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$ for all $s, s' \in \mathcal{S}, r \in \mathcal{R}$ and $a \in \mathcal{A}$. The discount rate, $\gamma \in [0, 1]$, is used to control how much the agent values immediate reward over future reward.

When in a given state s at time t, an agent seeks to act in a way that will maximize its expected sum of long-term, discounted rewards. That is, it attempts to maximize

$$\begin{aligned} G_t &\doteq \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] \end{aligned}$$

where G_t is called the return at time t. The expectation is taken with respect to both the dynamics $\mathcal{P}$ of the MDP and the policy of the agent. An agent's policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, often written as conditional distribution $\pi(a|s)$, assigns a probability to every action $a \in \mathcal{A}$ when in state $s \in \mathcal{S}$.

A generalization of the MDP is the Partially Observed Markov Decision Process (POMDP), wherein an agent doesn't necessarily receive complete information about the state of the environment. A POMDP is formally defined as a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \Omega, \mathcal{O}, \gamma)$. Here Ω is a set of observations and $\mathcal{O} : \Omega \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a conditional distribution $\mathcal{O}(o \mid s, a)$ over observations $o \in \Omega$ for given $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The agent's policy is then $\pi(a|o)$, a distribution over actions given the current observation. When an agent is situated in some (for example) 2D or 3D space, an observation is often its "field of view". It may not see the whole environment from a single position.

2.2 Multi-Agent Tasks

The formalism for the extension of the MDP to the multi-agent setting is called a Markov game. Markov games were heavily inspired by similar constructions in game theory called stochastic games [12]. We consider a partially observed version of the Markov game. Similar to a POMDP, a Markov game for N agents is defined by a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \boldsymbol{\mathcal{P}}, \Omega, \mathcal{O}, \gamma)$. The main difference is in the form of the environment dynamics $\boldsymbol{\mathcal{P}}$. Here we have that $\boldsymbol{\mathcal{P}} : \mathcal{S} \times \mathcal{R}^N \times \mathcal{S} \times \mathcal{A}^N \to [0, 1]$. The state $\mathcal{S}$ now encapsulates state information for the environment and all agents. Each agent i receives its own reward $r_i \in \mathcal{R}$, takes its own observation $o_i \sim \mathcal{O}_i(o|s, \boldsymbol{a})$ and has its own policy $\pi_i(a_i|o_i)$, where $\boldsymbol{a} = \{a_1, \dots, a_N\}$ is the joint action of all agents. Each agent seeks to maximize its own

expected return

$$G_i = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{i,t+1}\right].$$

There are several ways to classify different instances of Markov games. Depending on the environment dynamics, a Markov game could be either cooperative, competitive, or somewhere in between. Classic competitive settings include two-player, zero-sum games like chess and go. A modern cooperative setting might be that of a traffic junction on a road.

Agents in a Markov game can be classified according to how much information is shared between them. On one extreme is the case of independent learners [13] [14] [15], when no information is shared at all and each agent regards the others as part of the environment. Another extreme is the case where each agent knows every other agent's model/policy parameters [16]. A common paradigm is to even have them share parameters.

Agents could also be classified by the mechanism by which they share information. They could have explicit communication channels [17] [18] [19], learn cooperatively at training time [20] [21] [22], or model each other through observation [23] [24] [25].

2.3 Deep Reinforcement Learning

While classical reinforcement learning algorithms do exist that can find optimal agent policies in a simple environment, modern problems are often too complex to be solved without the representational capacity and generality of neural networks. This complexity usually comes in the form of high dimensionality of the state space and possibly the action space. Learning optimal behavior for a robot given a video stream of input, for example, is far beyond the capabilities of classical algorithms.

Not until 2015 did anyone successfully scale reinforcement learning up to high-dimensional observations. In their paper [10], researchers from DeepMind successfully utilized the power of deep neural networks to play Atari video games directly from pixels on the screen. The weights within the neural network comprise the parameters of the agent's policy model. The techniques they employed are still being built upon today.

2.4 Exploration

Subsection text here. [2] [3] [4] [5] [6] [7] [8]

2.5 Random Network Distillation

Subsection text here.

3 Problem Description

4 Model Description

5 Data and Experimental Results

6 Conclusion and Future Work

The conclusion goes here. [9]

References

[1] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018.

[2] N. Jaques, A. Lazaridou, E. Hughes, Ç. Gülçehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas, "Intrinsic social motivation via causal influence in multi-agent RL," *CoRR*, vol. abs/1810.08647, 2018.

[3] S. Sukhbaatar, I. Kostrikov, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," *CoRR*, vol. abs/1703.05407, 2017.

[4] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *CoRR*, vol. abs/1705.05363, 2017.

[5] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, "#exploration: A study of count-based exploration for deep reinforcement learning," *CoRR*, vol. abs/1611.04717, 2016.

[6] J. Z. Leibo, J. Pérolat, E. Hughes, S. Wheelwright, A. H. Marblestone, E. A. Duéñez-Guzmán, P. Sunehag, I. Dunning, and T. Graepel, "Malthusian reinforcement learning," *CoRR*, vol. abs/1812.07019, 2018.

[7] M. Dimakopoulou, I. Osband, and B. V. Roy, "Scalable coordinated exploration in concurrent reinforcement learning," *CoRR*, vol. abs/1805.08948, 2018.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 EP –, Feb 2015.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2nd ed., 2018.

[12] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[13] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *CoRR*, vol. abs/1710.03748, 2017.

[14] J. Z. Leibo, V. F. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," *CoRR*, vol. abs/1702.03037, 2017.

[15] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *CoRR*, vol. abs/1511.08779, 2015.
[16] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," *CoRR*, vol. abs/1709.04326, 2017.
[17] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," *CoRR*, vol. abs/1605.07736, 2016.
[18] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *CoRR*, vol. abs/1605.06676, 2016.
[19] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," *CoRR*, vol. abs/1703.10069, 2017.
[20] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *CoRR*, vol. abs/1706.02275, 2017.
[21] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *CoRR*, vol. abs/1705.08926, 2017.
[22] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," *CoRR*, vol. abs/1803.11485, 2018.
[23] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," *CoRR*, vol. abs/1802.09640, 2018.
[24] Z. Hong, S. Su, T. Shann, Y. Chang, and C. Lee, "A deep policy inference q-network for multi-agent systems," *CoRR*, vol. abs/1712.07893, 2017.
[25] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine theory of mind," *CoRR*, vol. abs/1802.07740, 2018.