Learning Reward Machines for Partially Observable Reinforcement Learning (Abridged Report)

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

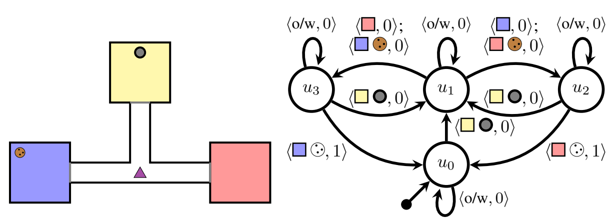
Reward Machines (RMs), originally proposed for specifying problems in Reinforcement Learning (RL), provide a structured, automata-based representation of a reward function that allows an agent to decompose problems into subproblems that can be efficiently learned. In this work, we show that RMs can be learned from experience, instead of being specified by the user, and that the resulting problem decomposition can be used to effectively solve partially observable RL problems.

1 Motivation and Research Problem

The use of neural networks for function approximation has led to many recent advances in *Reinforcement Learning (RL)*. Such *deep RL* methods have allowed agents to learn effective policies in many complex environments including board games [9], video games [5], and robotic systems [1]. However, RL methods (including deep RL) often struggle when the environment is *partially observable*. This is because agents in such environments usually require some form of memory to learn optimal behaviour [10]. Recent approaches for giving memory to an RL agent either rely on recurrent neural networks [6, 3, 13, 8] or memory-augmented neural networks [7, 4].

2 Technical Contribution: On Reward Machines and How to Learn Them

In this work, we show that *Reward Machines (RMs)* [11] are another useful tool for providing memory in a partially observable environment. We propose a method for learning an RM directly from experience in a partially observable environment, in a manner that allows the RM to serve as memory for an RL agent. To ground this discussion, consider the following problem:



(a) Cookie domain. (b) Perfect RM.

Figure 1: A partially observable environment and a RM.

Example 2.1 (The cookie domain)**.** *The* cookie domain*, shown in Figure 1a, has three rooms connected by a hallway. The agent (purple triangle) can move in the four cardinal directions. There is a button in the yellow room that, when pressed, causes a cookie to randomly appear in the red or blue room (unless the environment already contains a cookie, in which case it gets randomly moved to the red or blue room). There is no cookie at the beginning of the episode. The agent receives a reward of* $+1$ *for each time it reaches a cookie (which removes the cookie). This is a partially observable environment since the agent can only see what it is in the room that it is currently in.*

RMs decompose problems into a set of high-level states U and define how to transition from one RM state to another using if-like conditions. These conditions are over a set of binary properties

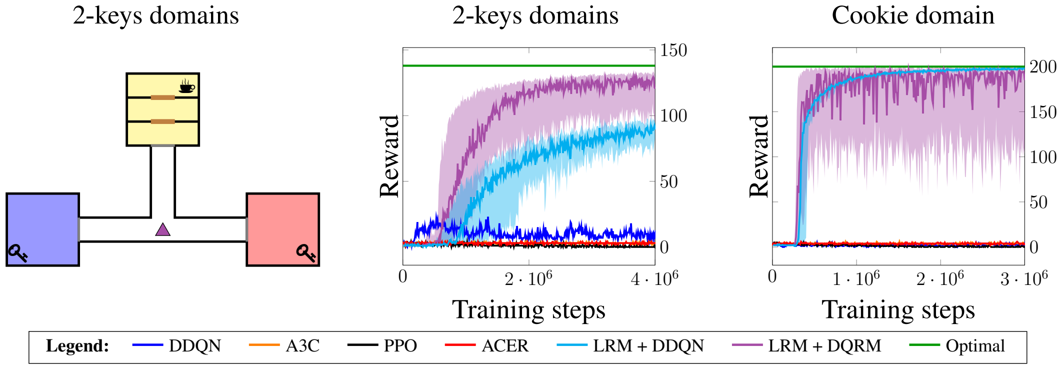


Figure 2: Total reward collected every $10,000$ training steps.

$\mathcal{P}$ that the agent can detect from the current observation. For example, in the cookie domain, $\mathcal{P} = \{$, , , , , , $\}$. These properties are true in the following situations: , , , or is true if the agent is in a room of that color; is true if the agent is in the same room as a cookie; is true if the agent just pushed the button; and is true if the agent just ate a cookie.

Figure 1b shows a possible RM for the cookie domain. It has an initial state u_0. The edge labels provide a visual representation of the state-transition and reward-transition functions of the RM. For example, label $\langle$, $1\rangle$ between state u_2 and u_0 represents that if the RM is in state u_2 and the agent just ate a cookie in room , then the agent will receive a reward of 1 and the RM will transition to u_0. Any properties not listed in the label are false. We also use multiple labels separated by a semicolon to describe different conditions for transitioning and the label "o/w" stands for "otherwise."

When learning a policy for a given RM, one simple technique is to learn a policy $\pi(a|o, u)$ that considers the current observation $o \in O$ and the current RM state $u \in U$ when selecting the next action $a \in A$. Interestingly, a partially observable problem might be non-Markovian over O, but Markovian over $O \times U$ for some RM. To learn RMs, our overall idea is to search for an RM that can be effectively used as external memory by an agent. This is an RM that remembers sufficient information about the history to make accurate Markovian predictions about the next observation.

The RM in Figure 1b is *perfect* w.r.t. this criterion. Intuitively, every transition in the cookie domain is Markovian except for transitioning from one room to another. Depending on different factors, when entering to the red room there could be a cookie there (or not). This RM encodes such information using 4 states, where u_0 represents the state where the agent knows that there is no cookie, at u_1 the agent knows that there is a cookie in the blue or the red room, at u_2 the agent knows that there is a cookie in the red room, and at u_3 the agent knows that there is a cookie in the blue room. Since keeping track of more information will not result in better predictions, this RM is *perfect*.

We formalized the problem of learning a perfect RM as a discrete optimization problem which, given a set of traces and detectors for the symbols in $\mathcal{P}$, returns a perfect RM (under certain conditions). In our experiments, we solved the discrete optimization problem using Tabu search [2].

3 Experimental Results

We tested our approach on two partially observable grid environments. The first environment is the *cookie domain* described in §2. Each episode is $5,000$ steps long, during which the agent should attempt to get as many cookies as possible. The second environment is the *2-keys domain* (Figure 2). In this domain, the agent receives a reward of $+1$ when it reaches the coffee. To do so, it must open the two doors (shown in brown). Each door requires a different key to open it. Initially, the two keys are randomly located in either the blue room, the red room, or split between them.

We tested two versions of our Learned Reward Machine (LRM) approach: LRM+DDQN and LRM+DQRM, and compared against 4 baselines: DDQN [12], A3C [6], ACER [13], and PPO [8]. DDQN used the concatenation of the last 10 observations as a limited memory. A3C, ACER, and PPO used an LSTM to summarize the history. Note that the output of the binary detectors was also given to the baselines. As Figure 2 shows, LRM approaches largely outperformed all the baselines.

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[2] F. Glover and M. Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[3] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[4] A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee. Memory augmented control networks. *arXiv preprint arXiv:1709.05706*, 2017.

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

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

[7] J. Oh, V. Chockalingam, S. Singh, and H. Lee. Control of memory, active perception, and action in minecraft. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2790–2799, 2016.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

[10] S. P. Singh, T. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

[11] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2112–2121, 2018.

[12] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2094–2100, 2016.

[13] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.