Learning Symbolic Representations for Reinforcement Learning of Non-Markovian Behavior

Phillip J.K. Christoffersen, Andrew C. Li, Rodrigo Toro Icarte, Sheila A. McIlraith
University of Toronto & Vector Institute for Artificial Intelligence, Toronto, Canada
`{phill,andrewli,rntoro,sheila}@cs.toronto.edu`

Abstract

Many real-world reinforcement learning (RL) problems necessitate learning complex, temporally extended behavior that may only receive reward signal when the behavior is completed. If the reward-worthy behavior is known, it can be specified in terms of a non-Markovian reward function—a function that depends on aspects of the state-action history, rather than just the current state and action. Such reward functions yield sparse rewards, necessitating an inordinate number of experiences to find a policy that captures the reward-worthy pattern of behavior. Recent work has leveraged Knowledge Representation (KR) to provide a symbolic abstraction of aspects of the state that summarize reward-relevant properties of the state-action history and support learning a Markovian decomposition of the problem in terms of an automaton over the KR. Providing such a decomposition has been shown to vastly improve learning rates, especially when coupled with algorithms that exploit automaton structure. Nevertheless, such techniques rely on *a priori* knowledge of the KR. In this work, we explore how to automatically *discover* useful state abstractions that support learning automata over the state-action history. The result is an end-to-end algorithm that can learn optimal policies with significantly fewer environment samples than state-of-the-art RL on simple non-Markovian domains.

1 Introduction

Deep RL has shown promise at learning complex behavior in many settings, including game-playing [1], robotics [2], and control systems [3]. These algorithms typically take advantage of a Markov assumption — that it is enough to consider only the current state when deciding which action to take. However, many real-world tasks are inherently temporally extended. The pattern of behavior the RL agent must learn depends not only upon the current state but also on past states and actions. For example, an agent that needs to get through a locked door to get high reward must have previously located and acquired the key. Unfortunately, learning such temporally extended behavior can be incredibly challenging since the agent must learn to discern relevant features from its state-action history; these can be arbitrarily far removed from the present state, and may depend on this history in complex ways and without intermediate reward signal to aid learning. The standard deep RL solution to learning such temporally extended behavior is to use a recurrent neural network (RNN), which learns an abstract hidden state in order to summarize environment histories, but RNNs require much data to train, and are difficult to tune.

By contrast, in recent work, an algorithm for learning temporally extended behavior is proposed, where the RNN hidden states are replaced with an augmentation of the current state in terms of hand-designed propositional symbols, which incorporate domain knowledge and point the agent towards potentially reward-relevant properties of the state-action history (e.g. [4, 5, 6, 7]). In the previous example, one can augment the agent with the propositional symbol **have_keys**, indicating whether the agent has acquired the keys to the door in the past. Markovian policies on this state space now become vastly more expressive: if one can additionally condition on the truth state of **have_keys** when deciding an action, we can perform the following temporally extended behavior: go towards the keys when **have_keys** is false, then go towards the door when **have_keys** is true. But how did we know to augment the agent with **have_keys** in the first place? While this example seems simple, this is only because we contrived the reward: it is not in general clear which propositional symbols to augment an agent with, in order to achieve high performance. To address this, we propose the use of automata learning within the RL framework to *automatically* yield such propositional symbols, rather than relying on domain knowledge. We demonstrate that the trained automata dramatically accelerate policy learning, with our end-to-end approach outperforming a state-of-the-art RL algorithm (Recurrent-PPO) on several non-Markovian reward domains.

2 Related Work

Recently, the idea of specifying non-Markovian reward functions in RL via formal languages such as Linear Temporal Logic (LTL) (e.g., ([8, 9, 10, 11, 12, 13, 5, 14]) or automata (e.g., [7, 4, 15, 16]) has garnered significant attention. While these approaches rely on domain knowledge and a domain-specific vocabulary for specification of the reward function, we consider a black-box non-Markovian reward and present an automated approach to uncover the reward structure. In this approach, we train automata offline using a reward-prediction heuristic, and augment the environment states with the states of the learned automaton, as opposed to hand-designed features. An alternate black-box approach to ours is to first train an RNN, with a standard deep (recurrent) RL algorithm, and then "quantize" the hidden state of the RNN, but this learned transition model is not a direct function of the state-action history [17].

Previous work by Toro Icarte et al. [4] and Xu et al. [18] share many of the motivations of our work but perform poorly in noisy environments. The work most similar to ours is by Gaon & Brafman [6], which we build on in several key ways. First, the (off-the-shelf) automata-learning approaches they employ are sensitive to noisy data and often learn large, sample-sensitive automata even when the reward structure is simple. We make use of recent advances in automata-learning which are robust to noise and regularize the size of the automaton, which we demonstrate in Section 5. Furthermore, [6] lacked experimental comparisons against state-of-the-art RL. In our experiments with non-Markovian goals, we outperform state-of-the-art RL based on RNNs.

3 Preliminaries

An MDP is a tuple $\mathcal{M} = (S, A, P, R, \gamma)$, where S is a set of states, A a set of actions, $P : S \times A \times S \to [0, 1]$ the state-action transition function, $R : S \to \mathbb{R}$ the reward, and $0 \leq \gamma \leq 1$ the discounting factor [19]. In such a setup, the reward R is considered Markovian, due to its dependence only on the most recent state. We will consider the following extension of the MDP: an NMRDP [20] (non-Markovian Reward Decision Process) $\mathcal{N} = (S, A, P, R, \gamma)$ is as before, but where the reward $R : \mathcal{H} \to \mathbb{R}$, where $\mathcal{H} = (S \times A)^*$ is the set of finite histories with states S and actions A: in other words, the agent can be rewarded for behavior which is arbitrarily far removed from its current experience. Further, we define a proposition as a function $P : \mathcal{H} \to \{\text{True}, \text{False}\}$, in an NMRDP. Intuitively, propositions correspond to facts about the state-action history in a given episode, such as "the agent has at some point reached the top right corner" or "within the 3 most recent timesteps, the agent took action x". While the number of possible propositions grows double-exponentially in the length of the episode, domain knowledge is often used to specify a relevant set of such propositions, under which the reward is Markovian. The RL domains we consider have non-Markovian goals, i.e. there exists $\mathcal{G} \subset \mathcal{H}$ where $R(g) = 1$ for $g \in \mathcal{G}$, and $R(h) = 0$ for $h \in \mathcal{H} - \mathcal{G}$. Intuitively, we want to create a policy which makes the agent attain a goal history as soon as possible.

4 Algorithm

We use the algorithm described in Algorithm 1 named AutRL. Following each period of Markovian learning, the sampled traces are used to train (offline) a deterministic finite automaton (DFA) M to predict whether a given sequence achieves reward 0 or 1. We leverage the DFA-learning approach from [21] due to its efficiency, its propensity to learn small DFAs with few transitions, and its robustness to noise. Tabular Q-learning is used for `markov_learn`. Intuitively, a DFA with state

Algorithm 1: AutRL

```
dfa ← empty_automata;
$\pi$ ← uniform_random_policy;
traces ← $\emptyset$ ;
while true do
    sample_traces ← sample($\pi$, N) ;
    append traces with sample_traces;
    if sample_traces inconsistent with dfa then
        dfa ← aut_learn(traces);
    end
    $\pi$ ← markov_learn(sample_traces $\times$ dfa);
end
```

space Q that accurately discriminates reward 1 traces from reward 0 traces (which we define as *consistent*) must model all parts of the state-action history relevant to the goal, and therefore the augmented state space $S \times Q$ must make the problem Markovian. We remark that the resultant DFAs are functions $\mathcal{H} \to Q$ and are learned end-to-end without domain knowledge. Examples of this can be seen in Section 5. We also provide a convergence guarantee for this algorithm in Appendix A.

We note that our implementation using Q-learning converges to an optimal policy as the number of environment samples approaches infinity (assuming $\mathcal{G}$ is regular, as above) due to the optimal convergence guarantees of Q-learning on MDPs [19]). Note that while we search for consistent DFAs, this condition is not necessary to make the learning problem Markovian. For this reason, we relax the inconsistency condition analyzed above, replacing the DFA only under weak performance (i.e. low average reward) at the end of a given epoch of `markov_learn`.

5 Experimental Results

The purpose of our experiments was to evaluate our AutRL algorithm, which leverages a learned symbolic representation, relative to a state-of-the-art RNN-based deep RL algorithm. The two metrics were the quality of the policies in terms of maximizing reward, and their sample efficiency. We tested on four non-Markovian domains, similar to those used in the experiments of [6] as follows.

Multi-Armed Bandit: A single-state environment with two actions $(\mathrm{left}, \mathrm{right})$ and episodes of length 6. A reward of 1 is obtained only if the 6 actions performed are precisely $\mathrm{left}, \mathrm{right}, \mathrm{right}, \mathrm{left}, \mathrm{right}, \mathrm{left}$ in that order.

Hallway: A 1×10 grid, aligned left-to-right, with actions $\mathrm{left}, \mathrm{right}$, and with episodes of length 30. The agent spawns at a random location on the left half of the grid and must first reach the right-most square, and then reach the left-most square to obtain a reward of 1. Before reaching the right-most square, the optimal action in every state is right, but after reaching the right-most square, the optimal action becomes left in every state.

Gridworld: A 5×5 gridworld (with x, y-coordinates from 0 to 4) with an episode length of 20 and actions $\mathrm{up}, \mathrm{down}, \mathrm{left}, \mathrm{right}$. The agent must first reach $(4, 0)$ and then the opposite corner $(0, 4)$ to collect a reward of 1. We also tested a noisy version of this environment where the reward was randomly withheld in 10% of successful traces, and actions had a random outcome 10% of the time.

We compared AutRL against Recurrent-PPO ([22]) (with an LSTM policy) using the OpenAI Baselines implementation [23]. As shown in all of the below environments, AutRL converges to high reward policies in fewer environment samples than Recurrent-PPO does. Notice that AutRL, despite being automata-based, outperforms Recurrent-PPO even on the Stochastic Gridworld, showing that AutRL is robust to noisy environments. In Figure 2 are two such examples of learned automata on two different runs of the Hallway environment.

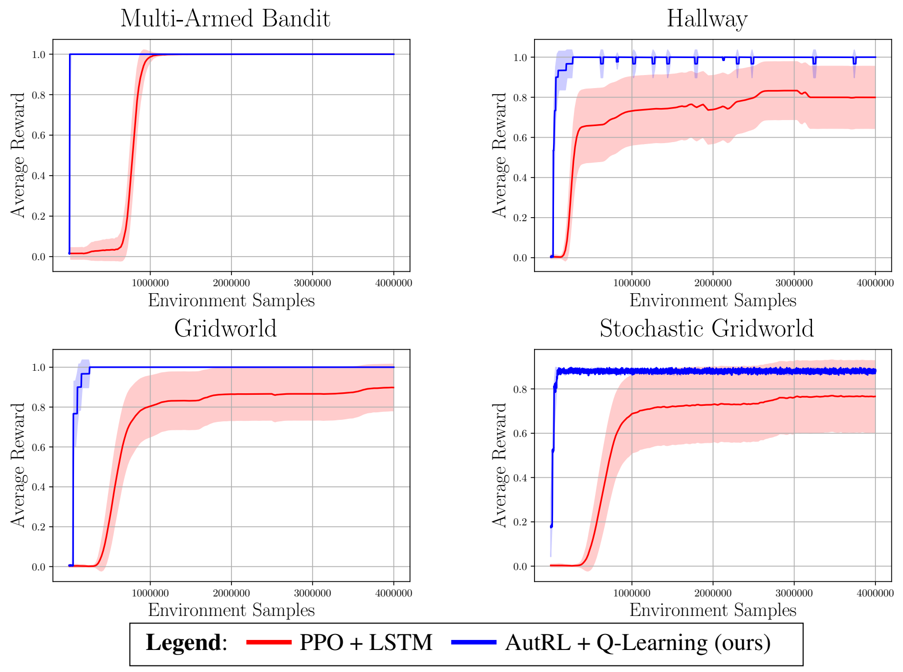

Figure 1: The results from the conducted experiments. The error bars are 95% confidence intervals over 30 runs. AutRL learns a superior policy to the LSTM-based PPO in a way that is, at most, over order of magnitude more sample-efficient. Learning is far more stable across runs with AutRL: the error bars for AutRL are far tighter than those for PPO, indicating very little variation in the learning trajectory across runs.

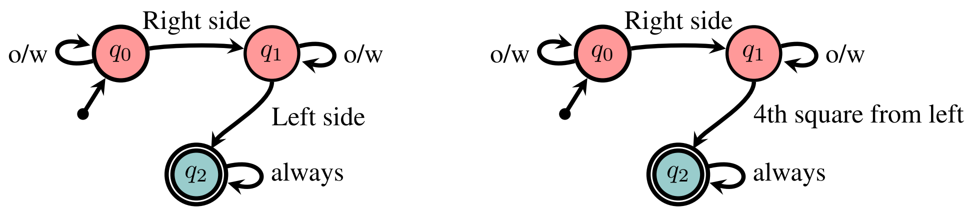

Figure 2: Two distinct DFAs learned from the Hallway environment. Both are enough to make the domain Markovian: however, the one on the right is not a perfect classifier of $\mathcal{G}$ for this domain. This shows that the condition in the statement of AutRL above is strictly stronger than necessary: there are automata which do not perfectly decide $\mathcal{G}$, but which discern enough relevant information to learn optimal, temporally extended behavior in non-Markovian settings.

6 Concluding Remarks

In this work, we show how to learn a KR that provides a useful symbolic abstraction in support of realizing temporally extended behavior in standard RL agents which rely on a Markov assumption. We provide an end-to-end RL algorithm which, in deterministic and stochastic domains, is demonstrably more sample-efficient than the state of the art. We further provide theoretical guarantees of optimal convergence for our approach (under conditions outlined in the theorem of Appendix A), along with insight into the structure of DFAs which are learned by this method.

This work reveals several promising directions for future research. While we considered environments where the reward function is non-Markovian, learning in the more general setting of POMDPs is an important problem related to this work, to which the DFA methods leveraged here do not apply. Further, AutRL leverages DFAs, which can only represent regular languages, thus learning efficiently when $\mathcal{G}$ is not a regular language in $S \times A$, as well as learning KR in high-dimensional and continuous state spaces, remain major open challenges to this methodology.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and Microsoft Research. The third author also acknowledges funding from ANID (Becas Chile). Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute for Artificial Intelligence (`www.vectorinstitute.ai/partners`). Finally, we thank the Schwartz Reisman Institute for Technology and Society for providing a rich multi-disciplinary research environment.

References

[1] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[2] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the Third IEEE-RAS International Conference On Humanoid Robots*, pages 1–20, 2003.

[3] Robert H Crites and Andrew G Barto. Improving elevator performance using reinforcement learning. In *Proceedings of the 9th Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1017–1023, 1996.

[4] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. In *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 15523–15534, 2019.

[5] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 19, pages 6065–6073, 2019.

[6] Maor Gaon and Ronen Brafman. Reinforcement learning with non-Markovian rewards. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 3980–3987, 2020.

[7] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila 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.

[8] Bruno Lacerda, David Parker, and Nick Hawes. Optimal and dynamic planning for Markov decision processes with co-safe ltl specifications. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 13, pages 1511—-1516, 2014.

[9] Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341*, 2017.

[10] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila McIlraith. Decision-making with non-Markovian rewards: From LTL to automata-based reward shaping. In *Proceedings of the Tenth International Symposium on Combinatorial Search (SoCS 2017)*, pages 279—-283, 2017.

[11] Xiao Li, Yao Ma, and Calin Belta. A policy search method for temporal logic specified reinforcement learning tasks. In *2018 Annual American Control Conference (ACC)*, pages 240–245, 2018.

[12] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 452–461, 2018.

[13] Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTLf/LDLf non-Markovian rewards. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1771–1778, 2018.

[14] Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J Pappas, and Insup Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 5338–5343. IEEE, 2019.

[15] Ronen I Brafman and Giuseppe De Giacomo. Regular decision processes: A model for non-Markovian domains. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5516–5522, 2019.

[16] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099v8*, 2019.

[17] Anurag Koul, Sam Greydanus, and Alan Fern. Learning finite state representations of recurrent policy networks. *International Conference on Learning Representations*, 2019.

[18] Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 590–598, 2020.

[19] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[20] Faheim Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, volume 13, pages 1160–1161, 1996.

[21] Maayan Shvo, Andrew C Li, Rodrigo Toro Icarte, and Sheila A McIlraith. Interpretable sequence classification via discrete optimization. *arXiv preprint arXiv:2010.02819*, 2020.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

Appendix A: Convergence Guarantee + Proof

Theorem 1. *Let $\mathcal{N}$ be an NMRDP. Under the following (realistic) conditions,* `AutRL` *converges to an optimal policy in the sample limit.*

1. `sample_traces` *visits every reachable history within $\mathcal{N}$ i.o.a.s. (infinitely often, almost surely)* ***(exploration)***
2. *For any regular language $\mathcal{L}$, when sampling traces using* `sample_traces`*, that* `aut_learn` *eventually returns an automaton perfectly deciding $\mathcal{L}$ w.p.1.* ***(consistency of*** `aut_learn`***)***
3. `markov_learn` *converges to the optimal policy in the sample limit for an MDP* ***(consistency of*** `markov_learn`***)***
4. *$\{h : R(h) = 1\} = \mathcal{G} \subset \mathcal{H}$ forms a regular language with alphabet $\Sigma = S \times A$* ***(regularity)***

Proof. Fix an NMRDP $\mathcal{N} = (S, A, P, R, \gamma)$, and assume all conditions are met in the theorem statement. Then, since $\mathcal{G}$ is a regular language, we have that the variable **dfa** is, with probability 1, eventually set to an automaton, with state space Q, which perfectly decides $\mathcal{G}$. At such a point, then we have that the states of the automaton perfectly predict the reward R: thus, the problem $\mathcal{M} = (S \times Q, A, P, R, \gamma)$, simply defined as $\mathcal{N}$ but with a state space augmented with the automaton states of **dfa**, is actually an MDP. Further, **dfa** is never thereafter reset, since no inconsistent trace can possibly be sampled from the environment subsequently. Thus, $\mathcal{M}$ is simply an MDP being trained with Markovian learning algorithm markov_learn, and thus the appropriate convergence conditions apply. □

Appendix B: Baseline Hyperparameter Tuning

For the tabular Q-learning for our approach (AutRL), we used a learning rate of 0.1 on the deterministic environment, 0.001 on the stochastic environment. Further, an exploration parameter of 0.01 was used on the deterministic environments, and 0.05 with a decaying exponential schedule (with factor 0.99) was used for the stochastic environments. The automaton learning method was set at a maximum state threshold of 5 for all environments but the multi-armed bandit (which had 14), a loop penalty of 0.01 in all environments, and a transition penalty of 0.01 for the Multi-Armed Bandit, 0.3 for both gridworlds, and 0.6 for the hallway. Further, the timeout was set to 250.

For the Recurrent-PPO baseline, we used the OpenAI Baselines [23] implementation with an LSTM policy network, an entropy coefficient of 0.001, a learning rate of 0.0003, a discounting factor of 0.99, a batch size of 16384 (with 8 minibatches per update), and all other hyperparameters set to default. We followed standard practice in tuning these hyperparameters and found that these settings consistently performed best across our domains.