Synthesizing Programmatic Reinforcement Learning Policies with Memory-Based Decision Trees

Anonymous authors
Paper under double-blind review

Abstract

Programmatic reinforcement learning (PRL) has been explored for representing policies through programs as a means to achieve interpretability and generalization, meaning involving higher order constructs such as control loops. Despite promising outcomes, current state-of-the-art PRL methods are hindered by sample inefficiency and very little is known on the theoretical front about programmatic RL. A burning question is studying the trade-offs between sizes of programmatic policies and their performances. Hence, the motivation of this work is to construct programmatic policies with the shortest paths to the target region, ensuring near optimal behavior. Alongside this, we also investigate how we can reason with programmatic policies by using two learning systems: Reward Prediction Error (RPE) and Action Prediction Error (APE). How can we learn programmatic policies that can generalize better? The goal of this paper is to give first answers to these questions, initiating a theoretical study of programmatic RL. Our main contributions are construction a near optimal policy using memory-based decision trees, and studying the generalizability and size performance.

1 Introduction

Reinforcement Learning (RL) is a very popular and successful field of machine learning where the agent learns a policy in an unknown environment through numerical rewards, modelled as a Markov decision process (MDP). In the tabular setting, the environment is given explicitly, which implies that typically policies are also represented explicitly, meaning as functions mapping each state to an action (or distribution of actions). Such a representation becomes quickly intractable when the environment is large and makes it hard to compose policies or reason about them. In the general setting, the typical assumption is that the environment can be simulated as a black-box. Deep reinforcement learning algorithms which learn policies in the form of large neural networks have been scaled to achieve expert-level performance in complex board and video games (Silver et al. 2018; Vinyals et al. 2019). However, they suffer from the same drawbacks as neural networks which means that the learned policies are vulnerable to adversarial attacks (Qu et al. 2021) and do not generalize to novel scenarios (Sunderhauf et al. ¨ 2018). Moreover, big neural networks are very hard to interpret and their verification is computationally infeasible.

On the other hand, Deep reinforcement learning (DRL) has had a massive impact on the field of machine learning and has led to remarkable successes in the solution of many challenging tasks (Mnih et al., 2015; Silver et al., 2016; 2017). While neural networks have been shown to be very effective in learning good policies, the expressivity of these models makes them difficult to interpret or to be checked for consistency for some desired properties, and casts a cloud over the use of such representations in safety-critical applications.

To alleviate these pitfalls, a growing body of work has emerged which aims to learn policies in the form of programs (Verma et al. 2018; Bastani, Pu, and Solar-Lezama 2018; Verma et al. 2019; Zhu et al. 2019; Inala et al. 2020; Landajuela et al. 2021; Trivedi et al. 2021; Qiu and Zhu 2022;

Andriushchenko et al. 2022; Liang et al. 2023), under the name “programmatic reinforcement learning”. Programmatic policies can provide concise representations of policies which are easier to read, interpret, and verify. Furthermore, their short size compared to neural networks would mean that they can also generalize well to out-of-training situations while also smoothing out erratic behaviors. The goal of the line of work we initiate here is to lay new paradigms for programmatic reinforcement learning.

In this work we focus on developing a short-optimal policy using memories. We form, represent, and save each event and experience with a memory represented in a decision tree as a node. An interesting concept rise when knowing when exactly to form a memory, meaning when does the agent form a memory and how? We introduce the concept of the memory being "significant". We also seek to study how can we enable agents to reason with programmatic polices by integrating two separate learning systems: Reward Prediction Error (RPE), and Action Prediction Error (APE), which we believe can represent the agent’s reasoning abilities. This is an in-progress work and we leave experimental results for the future.

2 Algorithm

We are now ready to state our main theorem. It asserts that our algorithm, is a programmatic policy algorithm, ignoring lower order terms. In other words, after this many episodes interacting with the MDP. **Algorithm overview.** Our algorithm proceeds with following high level steps:

1. Form a set of memories m which allow us to visit all “significant” states with reasonable probability.
2. collect a sufficient amount of data and store each significant memory in the decision tree.
3. compute the empirical transition matrix $\hat{P}$ and the programmatic policy using the collected data.
4. for each reward function r, find a near-optimal policy by invoking a planning algorithm with transitions $\hat{P}$ and reward r.

The first two steps are performed in the exploration phase, while the latter two steps are performed in the planning phase.

2.1 Exploration Phase

The goal of exploration is to visit all possible states so that the agent can gather sufficient information in order to find the optimal policy eventually. However, rather different from the bandit setting where agent can select an arbitrary arm to pull, it is possible that certain state in the MDP is very difficult to reach no matter what policy the agent is taking. Therefore, we first introduce the concept of the state being “significant”.

Definition 3.2. A memory m in step h is δ-significant if there exists a policy π, so that the probability to reach m following policy π is greater than δ. In symbol: $maxP_h^{\pi}(m) \geq \delta$ Intuitively, with limited budget of samples and runtime, one can only hope to store all significant memories. On the other hand, since insignificant memories can be rarely stored no matter what policy is used, they will not significantly change the value from the initial states. Thus, for the sake of finding near-optimal policies, it is sufficient to visit all significant states with proper significance level ϵ.

Theorem 2.1 There exists absolute constant $c > 0$ such that for any $\epsilon > 0$ and $p \in (0; 1)$, if we set $N_0 \geq cS^2AH^4l_0^3\delta$ where $l_0 := log(SAH = (p\delta))$, then with probability at least 1p, that Algorithm returns a dataset D consisting of N trajectories $Z_{n\,n=1}^{\,N}$, which are i.i.d sampled from a distribution μ

satisfying:

$$\forall \delta - significant(s,h), \max \frac{P_h^\pi(s,a)}{\mu_h(s,a)} \leq 2SHA \tag{1}$$

Theorem 2.1 claims that using Algorithm 2, we can collect data from a underlying distribution μ, which ensures that for policy π, the ratio $P_h^\pi(s;a) = \mu_h(s;a)$ will be upper bounded for any significant state and action. That is, all significant state and action will be visited by distribution μ with reasonable amount of probability. Notice as δ becomes smaller, there will be more significant states and the condition (1) becomes stronger. As a result we need to take larger N_0. As we will see later, the δ we take eventually will be $\epsilon = (2SH^2)$, where ϵ is the suboptimality of the policy we find in the planning phase.

Concretely, for each state s at step h, our algorithm first creates a reward function r that is always zero except for the state s at step h. Then we can simulate a standard MDP by properly feeding this designed reward r when an agent interacts with the environment. It is easy to verify that the optimal policy for the MDP with this reward r is precisely the policy that maximizes the probability to reach (s; h).

2.2 Reasoning Learning Systems: RPE and APE

We rely on two separate learning systems: one for evaluating outcomes and another for reinforcing repeated actions. Known as reward prediction error (PRE) and action Prediction Error(APE), these systems help the agent to reason about his own actions and policy. While RPE helps the agent learn from outcomes. APE strengthens behaviors we repeat often, enabling more efficient multitasking by freeing up cognitive resources. In summary, Action Prediction Error (APE) allows habits to form by reinforcing frequent actions.

2.3 Constructing Programmatic Policies

What remains to be done is derive from the tree a programmatic policy; one could say “compress” paths, or find regularity. This is the purpose of Algorithm 1. that provides the main idea for the construction of the policy synthesis procedure.

Algorithm 1 Synthesizing a programmatic policy

1: **Input** A branch in the tree $([a_1, b_1], ..., [a_p, b_p])$ and the corresponding sequence of edges $(e_1, ..., e_p)$ and regions $(R_1, ..., R_{(p1)}$ VisitedEdges = ϕ
2: **for** i=1 to p-1 **do**
3:     **if** $e_i \notin VisitedEdges$ **then**
4:         add e_i to VisitedEdges
5:         start new Do Until block with local goal e_i
6:     **else**
7:         add to the top of the instruction From e_i a new target $[a_{i+1}, b_{i+1}]$ with preference a_{i+1}
8:     **end if**
9: **end for**
10: **End Procedure**

3 Conclusion

This work is a step to study and learn programmatic reinforcement learning, and more specifically a step to address the question of designing domain-specific languages for policies which are optimal and shorter, investigating the size of programmatic and how it changes and affects the performance. We introduced the concept of "memories" as weighted nodes in a decision tree. We also integrated two learning systems RPE and APE as a reasoning mechanism for the agent. We leave empirical experiments and results for future study.

References

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulc¸ehre, ¨ C¸ .; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wunsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, ¨ T. P.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782): 350–354.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419): 1140– 1144.

Qu, X.; Sun, Z.; Ong, Y.; Gupta, A.; and Wei, P. 2021. Minimalistic Attacks: How Little It Takes to Fool Deep Reinforcement Learning Policies. IEEE Transactions on Cognitive and Developmental Systems, 13(4): 806–817.

Sunderhauf, N.; Brock, O.; Scheirer, W. J.; Hadsell, R.; Fox, ¨ D.; Leitner, J.; Upcroft, B.; Abbeel, P.; Burgard, W.; Milford, M.; and Corke, P. 2018. The limits and potentials of deep learning for robotics. International Journal on Robotics Research, 37(4-5): 405–420.

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. Human-level control through deep reinforcement learning. Nature, 518(7540): 529, 2015.

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. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. Mastering chess and Shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815, 2017.

Verma, A.; Le, H. M.; Yue, Y.; and Chaudhuri, S. 2019. Imitation-Projected Programmatic Reinforcement Learning. In Annual Conference on Neural Information Processing Systems, NeurIPS, 15726–15737.

Verma, A.; Murali, V.; Singh, R.; Kohli, P.; and Chaudhuri, S. 2018. Programmatically Interpretable Reinforcement Learning. In International Conference on Machine Learning, ICML, volume 80 of Proceedings of Machine Learning Research, 5052–5061. PMLR.

Bastani, O.; Pu, Y.; and Solar-Lezama, A. 2018. Verifiable Reinforcement Learning via Policy Extraction. In Annual Conference on Neural Information Processing Systems, NeurIPS, 2499–2509.

Zhu, H.; Xiong, Z.; Magill, S.; and Jagannathan, S. 2019. An inductive synthesis framework for verifiable reinforcement learning. In ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI, 686– 701. ACM.

Inala, J. P.; Bastani, O.; Tavares, Z.; and Solar-Lezama, A. 2020. Synthesizing Programmatic Policies that Inductively Generalize. In International Conference on Learning Representations, ICLR. OpenReview.net.

Landajuela, M.; Petersen, B. K.; Kim, S.; Santiago, C. P.; Glatt, R.; Mundhenk, T. N.; Pettit, J. F.; and Faissol, D. M. 2021. Discovering symbolic policies with deep reinforcement learning. In International Conference on Machine Learning, ICML, volume 139 of Proceedings of Machine Learning Research, 5979–5989. PMLR.

Qiu, W.; and Zhu, H. 2022. Programmatic Reinforcement Learning without Oracles. In International Conference on Learning Representations, ICLR. OpenReview.net.

Trivedi, D.; Zhang, J.; Sun, S.; and Lim, J. J. 2021. Learning to Synthesize Programs as Interpretable and Generalizable Policies. In Annual Conference on Neural Information Processing Systems, NeurIPS, 25146–25163.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as Policies: Language Model Programs for Embodied Control. In IEEE International Conference on Robotics and Automation, ICRA, 9493–9500. IEEE.

Andriushchenko, R.; Ceska, M.; Junges, S.; and Katoen, J. 2022. Inductive synthesis of finite-state controllers for POMDPs. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI, volume 180 of Proceedings of Machine Learning Research, 85–95. PMLR