Active Grammatical Inference for Non-Markovian Planning

Noah Topper[1], George Atia[1], Ashutosh Trivedi[2], and Alvaro Velasquez[3]

[1]University of Central Florida [2]University of Colorado Boulder [3]Air Force Research Laboratory

Abstract

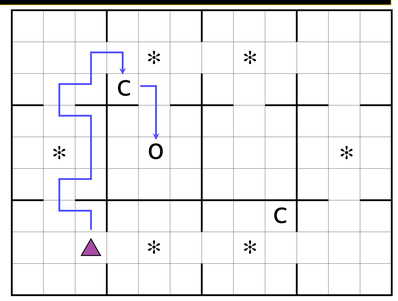
Angluin's active automata learning algorithm L* has found novel application recently in learning reward machines for non-Markovian RL. We consider a similar problem, but in the planning setting, where the transition dynamics are explicitly known, but the planning objective is an unknown non-Markovian reward signal that must be inferred from experiments. We show how we can then speed up L* and apply value iteration for faster learning. We compare against recent RL solutions and establish complexity results that illustrate the difference in runtime between grammatical inference in planning and RL settings.

Related Work

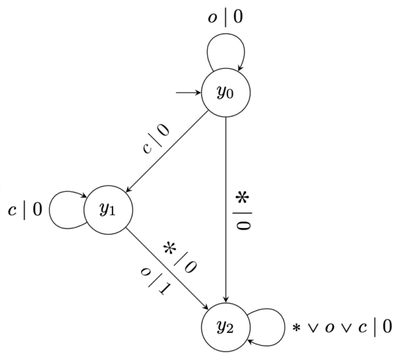
[Brafman'19] Non-Markovian planning with goal given as LTL formula
[Icarte'18] Non-Markovian RL with goal given by a reward machine
[Xu'21] Leveraging L* to learn a reward machine for non-Markovian RL

Problem to Solve

Assume we are given a non-Markovian reward decision process (NMRDP). This is much like an MDP, except each state is labeled and the reward is a function over the history of labels. We assume we have exact knowledge of the transition dynamics but can only get rewards by querying certain trajectories in the model.



We assume the reward is a regular function, meaning it can be represented by a Mealy machine (like a DFA but with outputs for each transition). For simplicity, we will assume a binary reward function, which can then be represented by a DFA.



L* is the classic algorithm for learning the minimal DFA representation of a regular language. It assumes a teacher that can answer membership and equivalence queries. For membership queries, the teacher answers if a given string is in the language or not. After a number of these, the learner forms a hypothesis DFA. For equivalence queries, the teacher checks if the hypothesis matches the target language. If not, it must return a counterexample.

For us, our target language is the set of traces that yield reward in the environment. We must find a way to implement membership and equivalence queries. We can then apply L* to learn a DFA representation of the reward. Finally, we can apply value iteration to the product of the NMRDP and this DFA to learn an optimal policy.

Methodology

We need only specify how to implement our membership and equivalence queries.

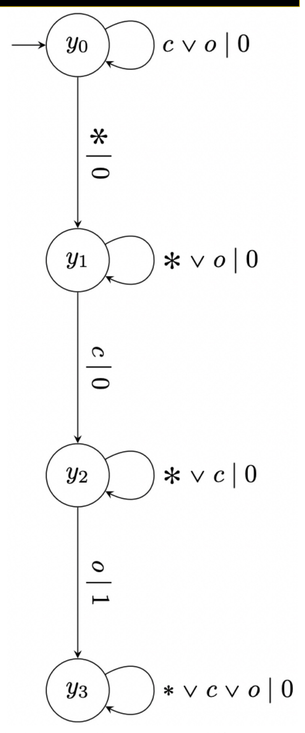
Since we have an explicit model of the environment, answering the membership queries is simple. Given query $a_1 \ldots a_n$, we first perform a BFS in the model for a state labeled a_1. From there we BFS to a state labeled a_2, and so on. At the end, we observe the resulting reward to answer the membership query.

We can also check exact equivalence, to a limited degree. Given a hypothesis DFA, consider all traces in the alphabet up to some length Ω. Run exact membership queries on each, as above, to see what rewards they get. Compare to the hypothesized reward.

We then run value iteration and start optimizing in the environment. We can continue to be on the lookout for longer counterexamples during this time, in which case we pass back to the learning phase.

Complexity

Previous treatments of this problem have dealt with the model-free (or sampling model) case and applied RL-based methods. In those settings, we simply feed in actions and get new states, labels, and rewards as observations.



To answer a membership query, we must find some trajectory of states whose trace matches the query. We need some heuristic to do this. In the AFRAI method **[Xu'21]**, they define a reward machine incentivizing actions in the environment that induce the trace of interest. They then allow the agent to optimize for some time. As an example, consider the reward machine to the right for query "*co".

If the trace is observed during this time, we can observe the reward to answer the query. If after some threshold number of times we still cannot observe the trace, we reject it by default.

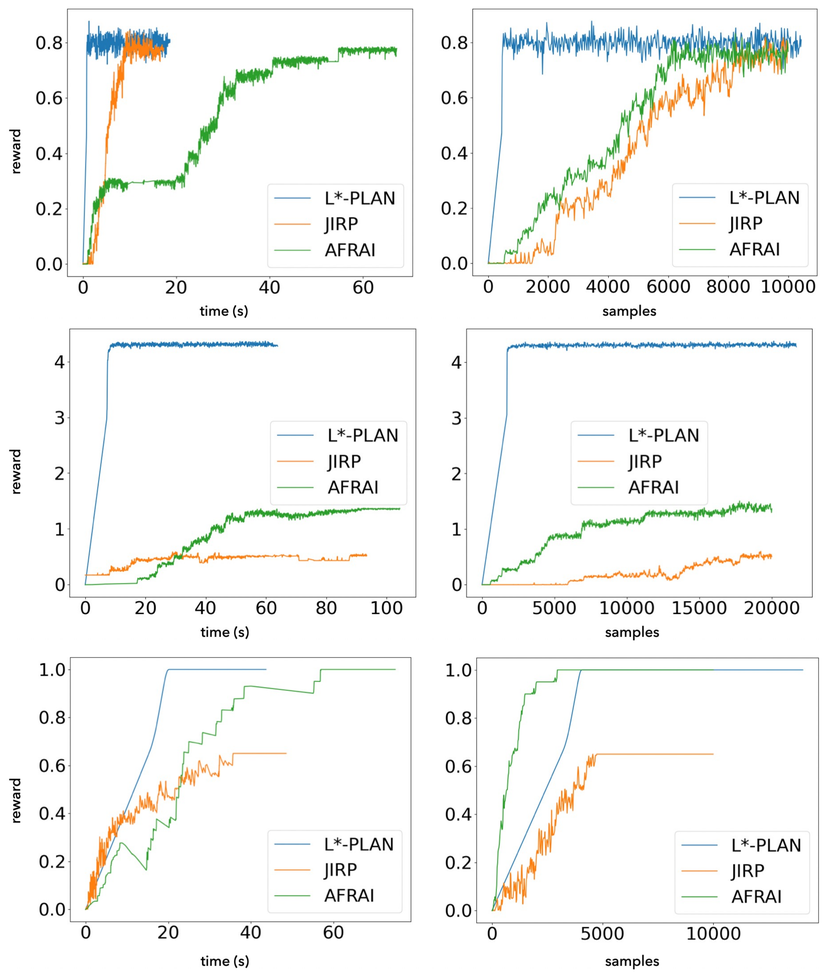
It turns out this problem, finding a policy (specifically a Markovian and deterministic one, dubbed "positional") that achieves a sequence of states that induces a given trace, is NP-complete. This motivates our attempt to use a model to bypass this problem and speed up learning.

Positional Membership Query (PMQ) Problem: Given a trace w and an NMRDP M, is there a positional policy that can achieve a sequence of states in M whose label trace is w?

Theorem: The PMQ Problem is NP-complete.

Experiments

We compared against AFRAI and JIRP **[Xu'20]**, a similar passive method. We test on three gridworlds, including the previous example. We analyze both time same sample efficiency.



We see that our method performs quite well. AFRAI seems to be competitive in terms of sample efficiency on the last problem. This comes down to how we count samples. We are including all of our additional equivalence queries, which are short and fast. They also allows us to learn exactly the correct DFA in this case, which AFRAI fails to do.

References

- Brafman,R.I.; and DeGiacomo,G. 2019. Planning for LTLf/LDLf Goals in Non-Markovian Fully Observable Nondeterministic Domains.
- Icarte, R. T. et al. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning.
- Xu, Z. et al. 2020. Joint inference of reward machines and policies for reinforcement learning.
- Xu, Z. et al. 2021. Active Finite Reward Automaton Inference and Reinforcement Learning Using Queries and Counterexamples.