Optimistic Exploration in Reinforcement Learning Using Symbolic Model Estimates – Main Supplementary file

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this appendix, Section 1 will cover the formal statements and proof sketches for various theoretical results, Section 2 will cover the implementation details including hyperparameters, Section 4 will provide a list of files included in the supplementary package and finally Section 5 will cover the various assumptions made in the work.

1 Theoretical Results

First result we are interested in establishing is the fact that for any given MDP of the form described in Section 3.1 in the main paper, there exists a corresponding symbolic model that meets the criteria discussed in Section 3.2 in the paper.

Proposition 1 *For an MDP of the form $\mathcal{M} = \langle S, T, A, I, R \rangle$, there exists a grounded symbolic model $\mathcal{M}_\downarrow = \langle F_\downarrow, A_\downarrow, I, G \rangle$, such that there exists*

1. *a mapping $\mathcal{C}$ from the state S of $\mathcal{M}$ to the states of $\mathcal{M}_\downarrow$,*
2. *a mapping $\mathcal{C}^{-1}$ from the actions $A_\downarrow$ of $\mathcal{M}_\downarrow$ to the actions A of $\mathcal{M}$, and*
3. *a mapping from valid traces in $\mathcal{M}$ to valid plans in $\mathcal{M}_\downarrow$ and vice-versa.*

Proof Sketch: We can build such a model by adding one grounded predicate (each corresponding to a unique lifted predicate of arity 0) for each state other than $\perp$ into $F_\downarrow$. Now a state in S maps (defined by $\mathcal{C}$) exactly to the symbolic state where the corresponding predicate is true and none of the other fluents are true. Now for the action, we will start with an action definition that includes conditional effect, and then convert it to a form assumed by the work. A conditional effect captures cases where an effect of an action only fires if the state meets certain criteria. Now we create one symbolic action for each MDP action. For each possible transition between states (other than $\perp$), we will add a corresponding conditional effect that takes the predicate corresponding to source state as condition and as effect the predicate corresponding to the target state. We will keep as precondition of the actions a disjunctive list of all possible states where it will not fail. For the goal action, we will have corresponding symbolic goal action whose precondition corresponds to potential states in S_G and the effect is a goal predicate. The initial state consists of only the predicate corresponding to the state I and the goal corresponds to the goal predicate. Now we can convert the actions with conditional effects to ones with no conditional effect (cf. [3]). Now the action mapping $\mathcal{C}^{-1}$, will map each of these new actions to the original MDP action from which it was defined. Now a plan is only valid in this model, if there exists a sequence of transitions from initial state to goal with non-zero probability. Similarly, for every valid trace there must exist a valid plan where each MDP action could be replaced by one of the potential symbolic actions that maps to it.

Next we will talk about the optimism of the initial model estimate

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Proposition 2 *$\mathcal{M}_0^{\mathcal{C}} = \langle F_{\downarrow}^{\mathcal{C}}, A_{\downarrow}^{\mathcal{C}}, I^{\mathcal{C}}, G^{\mathcal{C}} \rangle$. More formally, every action $a \in A_{\downarrow}^{\mathcal{C}}$ will be defined as follows: $a = \langle pre_0^a, add_0^a, del_0^a \rangle$, where $pre_0^a = del_0^a = \emptyset$ and $add_0^a = F_{\downarrow}^{\mathcal{C}}$ is optimistic for any MDP model such that there exists a mapping $\mathcal{C}$ from MDP state to symbolic states and a function $\mathcal{C}^{-1}$ mapping symbolic actions to MDP actions.*

This can be easily shown by the fact that every possible action sequence is a possible plan here.

Moving onto the update rule.

Proposition 3 *Update rule as presented in Section* **??***, will only result in an optimistic representation.*

Proof Sketch: The important point to note is that at any point, the update rule is only applied to an optimistic representation. So, in order for it to result in a non-optimistic model, it must have removed a plan corresponding to a valid trace. Given our initial construction of $\mathcal{M}_0^{\mathcal{C}}$, we always ensure that in $\mathcal{M}_0^{\mathcal{C}}$ the execution of an action a at a state $\mathcal{C}(s)$ will result in a symbolic state that is a superset of $\mathcal{C}(s')$, where $T(s, a, s') = 1$. Note that an application of an update rule will only extend the precondition if the corresponding MDP action fails and the preconditions are extended to exclude only the current state (though the list of excluded state, action pairs grows as the number of failed samples grows). Additionally, the effect is changed only to disallow impossible transitions. Since the transitions are deterministic, only one sample is needed to determine that no other transitions are possible from that state and action. This means that the above property (the fact that the resultant symbolic state will be superset) will be preserved through updates. Which in turn means that any plan that previously corresponded to a valid trace can become invalid.

Now coming to the theorem

Theorem 1 *Algorithm 1 (as referenced in the main paper) will (a) terminate in a finite number of steps and (b) identify a path to a goal (provided one exists); as long as the diverse planner used is complete (i.e., it will return a non-empty plan set as long as there exists a valid plan).*

Proof Sketch The validity of this theorem follows from the fact that the update rule will remove any plan that doesn't correspond to a valid trace from consideration again. If the planner is complete then it will effectively iterate over all possible plans. Eventually finding one that corresponds to a path that goes to the goal. This is guaranteed to exit in finite steps, as the set of non-redundant plans is guaranteed to be finite when the state space is finite.

Now revisiting Proposition 1

Proposition 1 *Let $\bar{A} = \langle a_1, ..., a_m \rangle$ be a set of actions marked as being instances of a single lifted action $a^{\uparrow}$. Then* min_add *must be a superset of add effects of $a^{\uparrow}$ and* max_del *a superset of deletes of $a^{\uparrow}$, where* min_add *and* max_del *are calculated for $\bar{A}$*

Proof Sketch The validity is trivial. The update rule makes sure that every effect estimate will be an optimistic estimate of the true ground action effects. In the case of add effects this estimate will be a superset and for delete effects it will be a subset. Thus, the lifted representation of each set must correspond to optimistic estimates of the true lifted representation of the effects.

2 Implementation Details

All experiments were on a laptop running Mac OS v 11.06, with 2 GHz Quad-Core Intel Core i5 and 16 GB 3733 MHz LPDDR4X. We did not use CUDA in any of the experiments. For the planner, we used the FI-diverse-agl planner provided as part of the forbid iterative planner. As discussed we generated 10 plans in every planing query. The search was given a maximum threshold of 1000 iterations, but we never reached that limit given our time limit. We stop an action from being considered if it fails 10 times in a row. We will update this upperbound on number of failures if the planner returns empty plan at any point. Since we found out that the planner was slowed down by the introduction of disjunctive preconditions, we replaced the disjunctions with a set of actions (this is an equivalent compilation popular within planning). To control the growth of the precondition, we introduce an upper bound on its size, set to 10 in our experiment. Note that the true size of the preconditions in all instances we consider here is significantly smaller than our bound. We could make the bound adaptive to a domain, but we do not expect it to make any significant difference. For

all the RL baselines we used a discount factor of γ. For Q learning and R max, we used a maximum of 1000000 episodes with 200 steps per episode. For exploration, the ϵ and decay rates were set as the same as the one used by SimpleRL experiment scripts. For PPO, we used the same default values used by [2]. The environment names for the two problems we tested in minigrid where where, MazeRooms-8by8-DoorKey-v0 and MazeRooms-2by2-TwoKeys-v0. While creating the PDDL model for minigrid we combined the turn actions with the other actions (move, pickup, drop, etc.), to avoid potential conditional effects.

3 Additional Experiments

3.1 Comparison with a Symbolic Baseline

To see how our method compares against other methods for symbolic model acquisition, we look at how many samples are required by a popular model learning method (cf. [1]) to generate a model that can produce a goal reaching plan. Since we don't have access to a plan library, we will generate one through random walks on our simulator. We focused on the Blocksworld domain, and for each of the five problems, we look at the number of samples required to generate a model that allows a potential plan to the goal. Since the method generates a pessimistic approximation of the model, any plan generated by the model is guaranteed to be valid and thus the method no longer requires the use of diverse planners to generate potential plans. We placed an upper bound of 600000 on the number of samples originally collected from the simulator (this is nearly six times larger than the number of samples required by our method). Out of the five problems, we found that only the method was only able to learn a model capable of generating a valid plan in the case of the first problem instance. Even for that problem, we found that the method took on average 548287.8 samples.

3.2 Kitchen Domain

As an additional experiment, we tested our method on the symbolic part of the Kitchen Domain [5]. We tested our method on the domain by creating a symbolic simulator that uses the descriptions provided in the appendix of the paper (specifically the inter-dependencies listed in Figure 5). The purely symbolic domain consisted of one action for each high-level goal possible and the preconditions were built based on the relationships described in the paper. The exact domain consisted of 15 predictions and 13 actions. The goal was the same as the one described in the paper (i.e., both banana and cabbage is cooked, they are placed on plates and the plates are served). The lifted version of our method was able to identify a valid plan in 61.46 sec using 24727.2 time steps (averaged across five runs) and the non lifted version took 393.57 secs and used 140681 samples (again averaged across five runs). Now executing the plans in the true simulator would require an additional component an additional step to drive the simulated robot to achieve each of these subgoal. However, as discussed in the paper, we can do that by using a motion level planner (like an RRT based planner).

4 Overview of Supplementary Files

The structure of the supplementary files are as follows

1. Baselines - This directory includes all the baselines we used to compare with our system
2. model-learning-simulators - includes the code to run our system and the test files used, so within this directory you would see
   (a) Domains - The test problems we used
   (b) src - the code base
   (c) experiment_scripts - the files to run to get the results reported in Table 1, 2 and Figure 1. The script files are named in a way that they are self-explanatory.
3. Minigrid - The code for the updated minigrid environment we made use of in our experiments.
4. Simple_rl_updated - Includes a modified version of Q learning that can be initialized with a plan found through our method.

5 Assumptions Made

Here we explicitly mention all the theoretical assumptions we have made in the paper and how to relax them

Model dynamics:

Deterministic model – our primary formulation and evaluation focus on deterministic domains. However, as discussed in the future works section, we can directly apply our method to stochastic environments by creating symbolic models that correspond to so-called *all outcome determinizations* of the model [6].

Observability:

We assume that the environment is fully observable. However, previous work have looked at how partial observability can be compiled into classical planning model. For optimistic estimates, we can make further simplification to assume that all unobservable facts are true, thus allowing us to directly apply our methods in such settings.

Finite state and action space:

We assume that the underlying state and action space is finite and thus can be represented exactly using a finite symbolic models. For cases where this is not true, we can still employ our symbolic model to capture an abstraction of the true state and action space.

Symbolic observations:

We assume that noise-free classifiers for each symbolic fluent are given. Previous work [4] have looked at the problem of learning such classifiers, which we can directly use in our scenario. If the classifiers are noisy, this corresponds to a special case of partial observability. As discussed above, we therefore can extend our model to handle noisy classifiers.

References

[1] Brendan Juba and Roni Stern. Learning probably approximately complete and safe action models for stochastic worlds. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*, pages 9795–9804. AAAI Press, 2022.

[2] Junkyu Lee, Michael Katz, Don Joven Agravante, Miao Liu, Tim Klinger, Murray Campbell, Shirin Sohrabi, and Gerald Tesauro. AI planning annotation for sample efficient reinforcement learning. *CoRR*, abs/2203.00669, 2022.

[3] Bernhard Nebel. On the compilability and expressive power of propositional planning formalisms. *J. Artif. Intell. Res.*, 12:271–315, 2000.

[4] Sarath Sreedharan, Anagha Kulkarni, and Subbarao Kambhampati. Explainable human–ai interaction: A planning perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 16(1):1–184, 2022.

[5] Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Regression planning networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1317–1327, 2019.

[6] Sung Wook Yoon, Alan Fern, and Robert Givan. Ff-replan: A baseline for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS*, page 352. AAAI, 2007.