LTL-Based Non-Markovian Inverse Reinforcement Learning

Extended Abstract

Mohammad Afzal
TCS Research, IIT Bombay
Pune, India
afzal@cse.iitb.ac.in

Sankalp Gambhir
EPFL Switzerland
Lausanne, Switzerland
sankalp.gambhir@epfl.ch

Ashutosh Gupta
IIT Bombay
Mumbai, India
akg@cse.iitb.ac.in

Krishna S
IIT Bombay
Mumbai, India
krishnas@cse.iitb.ac.in

Ashutosh Trivedi
University of Colorado Boulder
Boulder, Colorado, United States
ashutosh.trivedi@colorado.edu

Alvaro Velasquez
University of Colorado Boulder
Boulder, Colorado, United States
alvaro.velasquez@colorado.edu

ABSTRACT

The successes of reinforcement learning in recent years are underpinned by the characterization of suitable reward functions. However, in settings where such rewards are non-intuitive, difficult to define, or otherwise error-prone in their definition, it is useful to instead learn the reward signal from expert demonstrations. This is the crux of *inverse reinforcement learning* (IRL). While eliciting learning requirements in the form of scalar reward signals has been shown to be effective, such representations lack explainability and lead to opaque learning. We aim to mitigate this situation by presenting a novel IRL method for eliciting declarative learning requirements in the form of a popular formal logic—Linear Temporal Logic (LTL)—from a set of traces given by the expert policy.

KEYWORDS

Inverse reinforcement learning; Linear temporal logic; Constraints optimization

ACM Reference Format:
Mohammad Afzal, Sankalp Gambhir, Ashutosh Gupta, Krishna S, Ashutosh Trivedi, and Alvaro Velasquez. 2023. LTL-Based Non-Markovian Inverse Reinforcement Learning: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

Learning from demonstrations has become a viable approach to learning in environments where domain experts or performant agents can provide traces of (un)desirable behavior. One important embodiment of this form of learning is known as *inverse reinforcement learning* [14] (IRL), whereby an apprentice agent learns the reward function being optimized by a given expert policy or behavior. Please refer to our full paper [1] for more detail.

Linear Temporal Logic (LTL). We focus on (a subset of) LTL [13] as the specification language due to its succinctness [2, 9] and relevance in the AI [5, 7], formal methods [2, 10], control theory [3, 15], and machine learning [6] communities. Recently, it has gained popularity [4, 6, 11] in expressing learning objectives in model-free reinforcement learning (RL). The key computational problem for the LTL-based IRL is the following: given a pair $\mathcal{S} = (P, N)$ of samples consisting of positive traces P and negative traces N (both are sets of finite words), produce the highest-ranking LTL specification consistent with the sample where rank is informed by some user-tunable notion of simplicity over the LTL specifications.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

2 QUANTIFYING EXPRESSIVE PARSIMONY

An *alphabet* Σ is a non-empty, finite set of *symbols*. A *finite word* w over Σ is a finite sequence $a_1 a_2 ... a_n$ of symbols from Σ.

Given an LTL formula φ and a finite word w, we design a valuation function $V(\varphi, w)$ that quantifies the parsimony of φ in explaining w. We use the valuation function to rank the formulae. Intuitively, a pair scores high if all of the subformulae of the formula φ contribute in accepting w in $L(\varphi)$. However, we do so in a nuanced fashion by geometrically attenuating the effect of parsimony with the length of the word. For example, $\mathsf{G}(p \vee q)$ should score well along with word $(\{p\}\{q\})^3$ but should not do well with word $(\{p\})^6$, since the subformula q did not contribute to the acceptance. Similarly, $\mathsf{G}(p \vee q)$ should score better along with word $(\{p\}\{q\})^3$ than $(\{p\})^5\{q\}$.

Let us present a valuation function first. Let $\mathcal{F}$ represent the set of NNF GF-fragment formulae over $\mathcal{P}$. We interpret LTL formulae over finite words and define the quantitative semantics in terms of a *valuation mapping* $V : \mathcal{F} \times \Sigma^* \to \mathbb{R}^+ \cup \{0\}$, where $\Sigma = 2^{\mathcal{P}}$. The valuation mapping is defined over a word $w \in \Sigma^*$ inductively:

$$
\begin{aligned}
V(p, w) &= \begin{cases} 1 & \text{if } p \in w(1) \\ 0 & \text{otherwise} \end{cases} \\
V(\neg p, w) &= \begin{cases} 1 & \text{if } p \notin w(1) \\ 0 & \text{otherwise} \end{cases} \\
V(\varphi \wedge \psi, w) &= \beta \cdot V(\varphi, w) \cdot V(\psi, w) \\
V(\varphi \vee \psi, w) &= \beta \cdot \frac{V(\varphi, w) + V(\psi, w)}{2} \\
V(\mathsf{G}\varphi, w) &= \begin{cases} \beta \sum_{i=0}^{|w|} \alpha^i V(\varphi, w_i) & \text{if } V(\neg\varphi, w_t){=}0,\ \forall t \\ 0 & \text{otherwise} \end{cases} \\
V(\mathsf{F}\varphi, w) &= \begin{cases} \beta\alpha^t V(\varphi, w_t) & t{=}\min\{j \mid V(\varphi, w_j){>}0\} \\ 0 & \text{if } V(\varphi, w_t) = 0,\ \forall t \end{cases}
\end{aligned}
$$

Here, w_j is a shorthand for $w[j:]$. If $w \models \varphi$, then $V(\varphi, w)$ is non-zero. This scheme is parameterized by two discount factors: the *temporal discount factor* α and the *nesting discount factor* β.

3 LEARNING ALGORITHMS

As our main contribution, we propose learning algorithms to solve the following problem. *Given a sample $\mathcal{S}$=(P,N) over finite words, compute an LTL formula φ in the GF-fragment that best describes $\mathcal{S}$ and is consistent with $\mathcal{S}$. That is, φ has the highest score, based on the valuation described above, among all formulae such that for all $w \in P$, $w \models \varphi$ and for all $w' \in N$, $w' \not\models \varphi$.*

To achieve the goal of ranking formulae based on a quantitative notion of satisfiability, we propose the techniques of *constraint system optimization*, *optimized pattern matching* (Section 5.2 [1]), and *hybrid pattern matching*. In the first one, we get a sample and a depth d as input. We encode the syntax tree of this unknown formula of depth d along with constraints to compute the score of each node in the tree. The second one makes use of a formula template pattern provided by the user, but has unknown propositional variables. We encode constraints which allow mapping these variables to unique variables occurring in the sample. The third one is a "hybrid" approach, a middle ground incorporating both of the above techniques. We use an optimizing SMT solver to solve the constraints to find the *best* formula for the sample with the highest score according to the valuation function.

Furthermore, we also developed an alternative greedy search, called *compositional ranking*, which bypasses constraint solving and optimizations, by pruning the search space of formulae. We begin by enumerating all formulae of depth zero, i.e., all literals in our system as obtained after parsing input traces. We consider all compositions of these literals with the operators present. After enumerating the literals, we perform an "F-check" : for any φ, the F-check tests whether, in any input sample, Fφ holds. If a formula passes an F-check, it is retained to produce formulae of higher depth, else it is removed.

4 EXPERIMENTS

We have implemented the preceding algorithms in a tool called Anon. In this section, we present the results of Anon on a set of traces sampled from a grid-world environment running under OpenAI Gym. Anon is implemented in C++. For the optimization, it takes a set of positive traces, a (possibly empty) set of negative traces, and a formula template (which can simply be $\varphi(d)$, a search depth of d with no specification) or a combination, while the compositional ranking takes as input the traces along with a maximum search depth. Our implementation uses SMT solver Z3 [8] for the optimizations. All our queries to Z3 are quantifier-free. For optimization, Anon returns a formula with maximal score according to our scheme, while for compositional ranking, it returns a list of all satisfying formulae in the search space, sorted by score. In our experiments, we used a discount factor $\alpha = e^{-1}$ and also used $\beta = 0.8$ to decay each time we build deeper formulae in order to bias the ranking towards simpler formulae. We evaluated the performance of Anon on a 64-bit Linux system with an AMD Renoir Ryzen 5 (4500U) laptop CPU. We set 1000 seconds as timeout. We compare with Texada[16] and the SAT based tool Traces2LTL [12].

	Constraint Optimization	Comp. Ranking	Traces2LTL
Mean ILE	0.031	0.037	0.112
Input size	10^3	2×10^5	10^3

Non-Markovian IRL. We apply learning techniques to generate a reward function over an MDP defined as a grid world to obtain a non-markovian reward decision processes(NMRDP). After generating a randomized 10×10 grid environment labeled with propositional variables, we uniformly sample the grid taking actions compatible with an input automaton. This ensures that the generated traces satisfy a given formula. We use the same LTL properties as the previous case. We allow the MDP to randomly simulate for at least 100 steps, after which we wait for it to reach an accepting state. Through this method, we generated traces of length varying between 100 and 150, with 1000 positive and 1000 negative traces for each formula, amounting to a total trace length of at least 10^5 across all positive and negative inputs. However, for constraint system optimization and Traces2LTL [12], due to timeouts, a smaller subset was randomly selected from the traces.

For our experiments, given an NMRDP M in the form of a grid-world, we can compute the optimal policy for three different Deterministic Rabin Automaton (DRA), objectives by computing three product MDPs (Section 6.3 [1]). The first DRA objective is what we are trying to learn. We will denote the optimal policy here as π^*_{true} computed on $M \times A_{\text{true}}$, where A_{true} is the DRA representation of the LTL objective we are trying to learn. Then, we have the policy π^*_{QL} computed on $M \times A_{\text{QL}}$, where A_{QL} is the DRA learnt using Anon. Finally, we have the policy π^*_{T2L} computed on $M \times A_{\text{T2L}}$, where A_{T2L} is the DRA learnt using Traces2LTL [12]. We will take these three policies to generate our results in the form of the inverse learning error (ILE). We compute these value functions using uniformly random sampling of trajectories from every state in the NMRDP. We can then take a simple ratio (MeanILE in Table above) of the number of trajectories satisfied by A_{true}, and divide it with the total number of trajectories, and report an average over multiple runs and inputs. In particular, we will compute two ILE values, comparing $||V_{\pi^*_{\text{true}}} - V_{\pi^*_{\text{QL}}}||_2$ and $||V_{\pi^*_{\text{true}}} - V_{\pi^*_{\text{T2L}}}||_2$. Our experiments demonstrate that the former is smaller than the latter, thereby providing evidence that our approach generalizes better for non-Markovian IRL than a competing one adapted to the IRL.

5 CONCLUSION

In this paper, we presented a novel scheme to quantitatively evaluate LTL formulae. Our evaluation schema is designed such that the score received by a word is proportional to how well it represents the formula. Thus, words which are "good representatives" score higher than words which merely satisfy the formula (and hence qualify to be "poor representatives"). One of our contributions is to use this schema to mine LTL formulae from the traces of reactive systems. Our approach presents a viable solution to non-Markovian inverse reinforcement learning (IRL) in settings where the reward signal can be captured as LTL formulae.

REFERENCES

[1] Mohammad Afzal, Sankalp Gambhir, Ashutosh Gupta, Krishna S, Ashutosh Trivedi, and Alvaro Velasquez. 2023. LTL-Based Non-Markovian Inverse Reinforcement Learning. https://doi.org/10.48550/ARXIV.2110.13616

[2] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

[3] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic. 2020. Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning. In *ICRA*. IEEE, 10349–10355.

[4] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10349–10355.

[5] Ronen I Brafman and Giuseppe De Giacomo. 2019. Planning for LTLf/LDLf Goals in Non-Markovian Fully Observable Nondeterministic Domains.. In *IJCAI*. 1602–1608.

[6] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning.. In *IJCAI*, Vol. 19. 6065–6073.

[7] Giuseppe De Giacomo and Moshe Y Vardi. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. Association for Computing Machinery, 854–860.

[8] Leonardo de Moura and Nikolaj Bjorner. 2008. Z3: An Efficient SMT Solver. In *TACAS*. LNCS, Vol. 4963. Springer Berlin Heidelberg, 337–340. http://dx.doi.org/10.1007/978-3-540-78800-3_24

[9] Paul Gastin and Denis Oddoux. 2001. Fast LTL to Büchi automata translation. In *International Conference on Computer Aided Verification*. Springer, 53–65.

[10] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. 2019. Omega-Regular Objectives in Model-Free Reinforcement Learning. In *TACAS 2019, Proceedings, Part I (LNCS, Vol. 11427)*. Springer, 395–412.

[11] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2021. Mungojerrie: Reinforcement Learning of Linear-Time Objectives. *arXiv preprint arXiv:2106.09161* (2021).

[12] D. Neider and I. Gavran. 2018. Learning Linear Temporal Properties. In *2018 Formal Methods in Computer Aided Design (FMCAD)*. 1–10.

[13] Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 46–57.

[14] Stuart Russell. 1998. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*. 101–103.

[15] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. 2014. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *CDC*. 1091–1096.

[16] Texada 2015. TEXADA. https://github.com/ModelInference/texada.