Visual Reward Machines

Elena Umili[1], Francesco Argenziano[1], Aymeric Barbin[1] and Roberto Capobianco[1,2]

[1]Sapienza University of Rome
[2]Sony AI

Abstract

Non-markovian Reinforcement Learning (RL) tasks are extremely hard to solve, because intelligent agents must consider the entire history of state-action pairs to act rationally in the environment. Most works use Linear Temporal Logic (LTL) to specify temporally-extended tasks. This approach applies only in finite and discrete state environments or continuous problems for which a mapping between the continuous state and a symbolic interpretation is known as a symbol grounding function. In this work, we define Visual Reward Machines (VRM), an automata-based neurosymbolic framework that can be used for both reasoning and learning in non-symbolic non-markovian RL domains. VRM is a fully neural but interpretable system, that is based on the probabilistic relaxation of Moore Machines. Results show that VRMs can exploit ungrounded symbolic temporal knowledge to outperform baseline methods based on Recurrent Neural Networks (RNN) in non-markovian visual domains.

Keywords

Non-Markovian Reinforcement Learning, Neursoymbolic AI, Symbol grounding, Deep Reinforcement Learning

1. Introduction

Non-Markovian Reinforcement Learning (RL) tasks [1] are extremely difficult to solve because intelligent agents must consider the entire history of state-action pairs to act rationally in the environment. However, the current line of research in this area involves bypassing non-Markovianity by augmenting the state space with a set of *features* that encode the environment history and solving the augmented-state MDP with known RL algorithms.

Therefore, the main challenge is how to construct these features. In the case of non-symbolic-state MDPs, the most popular approach is to combine RL algorithms with the use of Recurrent Neural Networks [2][3][4], which automatically extract features from data sequences.

For problems with a symbolic finite state space, most works use LTL [5] or LTLf [6] to specify the temporally-extended task. This specification is then compiled into an automaton, and the automaton state is combined with the environment state to make the decision process Markovian. An example of this approach is the so-called Reward Machines [7]. Reward machines (RM) can also be applied to non-symbolic-state environments for which a symbol grounding function is known [8]. The latter maps the environment's raw state into a boolean interpretation over the symbols used by the specifications in LTL and makes the symbols observable. Therefore,

NeSy 2023, 17th International Workshop on Neural-Symbolic Learning and Reasoning, Certosa di Pontignano, Siena, Italy

umili@diag.uniroma1.it (E. Umili); argenziano@diag.uniroma1.it (F. Argenziano); aymeric.barbin@uniroma1.it (A. Barbin); capobianco@diag.uniroma1.it (R. Capobianco)

CEUR Workshop Proceedings (CEUR-WS.org)

summarizing RMs assumes prior knowledge of: (i) the symbol grounding function, (ii) the LTL or the Deterministic Finite Automaton (DFA) specification. Despite prior work that discards the assumption of knowing the temporal specification [9][10][11][12], no work assumes not knowing the symbol grounding function. We emphasize that learning this function for realistic environments with raw and/or high-dimensional states can be challenging and requires a large amount of labeled data.

In our previous work [13], we showed that we can exploit LTLf specifications in image sequence classification to speed up the learning process, even if the symbols in the specific alphabet are not grounded in the images. In another previous work [14], we exploited a particular recurrent neural network design inspired by Probabilistic Finite Automata (PFA) to perform DFA induction with gradient descent. In this work, we combine [13] and [14] to implement a neurosymbolic version of reward machines that we call *Visual Reward Machines* (VRM). The contribution of this paper is a novel formulation of reward machines that considers the symbol grounding function as a part of the specification. We show that we can relax the assumption of knowing the symbol grounding function and that the LTLf specifications can be exploited to increase the performance of RL algorithms in visual non-Markovian tasks, even if the specification is not grounded in the environment states.

The remainder of this paper is organized as follows: In Section 2, we review related works; in Section 3, we provide some preliminary knowledge on Reward Machines and Probabilistic Finite Automata; we define Visual Reward Machines and describe their implementation using neural networks in Section 4; we report preliminary experiments validating our method in Section 5; finally, we discuss final considerations and directions for future work in Section 6.

2. Related works

LTL formulas are widely used in Reinforcement Learning (RL) to specify non-markovian tasks [15]. Some work assumes prior knowledge of the LTL task specification [7][16], which is compiled into an automaton and monitored during the task execution to produce non-markovian rewards for the task. Some other works focus on *learning* the task machine from traces of symbolic observations and rewards received by the environment, by using known methods for automata induction [9][10][11] [12]. All these works, however, use symbolic data and do not consider the problem of discovering latent symbols in the data. For this reason, they are applicable only in symbolic-state environments or continuous problems for which a mapping between the continuous state and a symbolic interpretation is known, also called labeled MDP [8].

Many works assume to know an *imperfect* symbol grounding function for the task [17][18][19]. Namely, a function that sometimes makes mistakes in predicting symbols from states or predicts a set of probabilistic *beliefs* over the symbol set instead of a boolean interpretation. These works can be considered a preliminary step towards integration with non-symbolic domains. However, they do not assess the problem of learning the symbol grounding function, but only how to use a pre-trained imperfect symbol grounder. One work [20] uses the LTLf specification of the task without assuming any knowledge of the symbol grounding function. This work employs a neural network architecture that is *shaped as the LTL formula* to learn a representation of states,

that can be easily transferred to different LTL tasks in the same environment. The capability to transfer the representation to new tasks suggests that the representation can capture the semantics of symbols in some ways. However, the representation does not exactly ground the symbols in the environment observations.

3. Background

Reward Machines Reward machines (RM) [7] are an automata-based representation of non-Markovian reward functions. RMs provide a normal-form representation for reward specification in a diversity of formal languages. Given a finite set of propositions P representing abstract properties or events observable in the environment, RMs specify temporally extended rewards over these propositions while exposing the compositional reward structure to the learning agent. Formally, a simple Reward Machine is a tuple $RM = (2^P, Q, q_0, \delta_t, \delta_r)$, where 2^P is the automaton alphabet, Q is the set of states, q_0 is the initial state, $\delta_t : Q \times 2^P \to Q$ is the transition function and $\delta_r : Q \times 2^P \to \mathbb{R}$ is the reward function.

Probabilistic Finite Automaton A Probabilistic Finite Automaton (PFA) A_p is a tuple $(P, Q, i_p, \delta_{tp}, f_p)$, where P is the alphabet, Q is the finite set of states, $i_p : Q \to [0, 1]$ is the probability for a state to be an initial state, $\delta_{tp} : Q \times P \times Q \to [0, 1]$ is the transition probability function, and $f_p : Q \to [0, 1]$ is the probability of a state to be final. We have therefore $\sum_{q' \in Q} \delta_{tp}(q, p, q') = 1$, and $\sum_{q \in Q} i(q) = 1 \ \forall q \in Q, \forall a \in P$. We can also represent the PFA in matrix form as a *transition matrix* M_t, an *input vector* v_i and an *output vector* v_o. Matrix $M_t \in \mathbb{R}^{|P| \times |Q| \times |Q|}$ contains at index (p, q, q') the value of $\delta_{tp}(q, p, q')$. We denote as $M_t[p] \in \mathbb{R}^{|Q| \times |Q|}$ the 2D transition matrix for symbol p. The input vector $v_i \in \mathbb{R}^{1 \times |Q|}$ contains at index k the probability of state q_k to be an initial state. The output vector $v_o \in \mathbb{R}^{|Q| \times 1}$ has in position k the probability of state q_k to be accepting. Given a string $x = x[0]x[1]...x[l-1]$, we denote as $q_{p,0}, q_{p,1}...q_{p,l}$ the sequence of probabilities to visit a certain state, where $q_{p,t} \in \mathbb{R}^{1 \times |Q|}$ is a row vector containing at position k the probability to stay in state k at time t.

$$\begin{aligned} q_{p,0} &= v_i \\ q_{p,t} &= q_{p,t-1} \times M_t[x[t]] \qquad \forall t > 0 \end{aligned} \tag{1}$$

The probability of being in a final state at time t is the inner product $q_{p,t} \times v_o$. Therefore the probability of a string to be accepted is the probability to be in a final state in the last computed state $q_{p,l}$, and it is calculated as follows

$$v_i \times M_t[x[0]] \times M_t[x[1]] \times ... \times M_t[x[l-1]] \times v_o \tag{2}$$

4. Method

4.1. Visual Reward Machine definition

We define a Visual Reward Machine as a tuple $VRM = (S, P, Q, R, q_0, \delta_{tp}, \delta_{rp}, sg)$, where S is the set of input data, possibly infinite and continuous, the machine can process; P is a finite

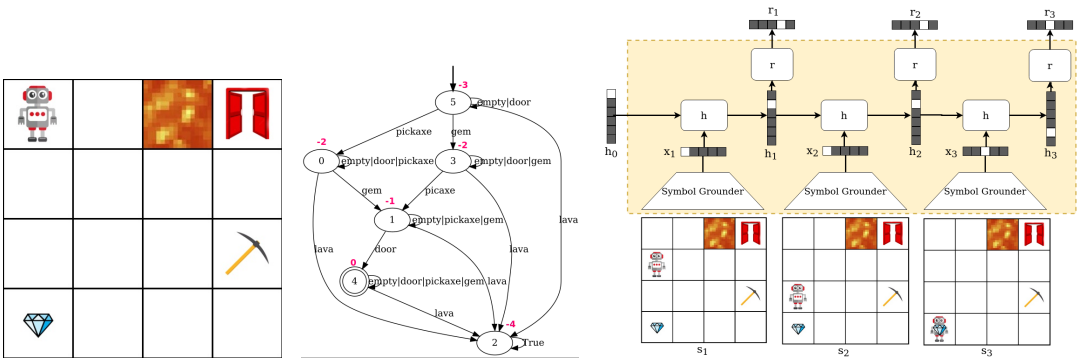


Figure 1: a) Visual Minecraft environment. b) non-markovian task specification for the Minecraft agent: the agent has to collect the pickaxe and the gem (in any order) and then go to door, while always avoiding the lava. c) Implementation of a Visual Reward Machine with convolutional and recurrent neural modules

set of symbols; Q is a finite set of states, R is a finite set of rewards; q_0 is the initial state; $\delta_{tp} : Q \times P \times Q \to [0, 1]$ is the transition probability function; $\delta_{rp} : Q \times R \to [0, 1]$ is the reward probability function and $sg : S \times P \to [0, 1]$ is the *symbol grounding* probability function.

Similarly to a RM, a VRM produces a reward sequence by processing a data sequence. Unlike RM, however, the input data do not need to be symbolic but can be of any type. The symbol grounding function maintains the link with a symbolic representation, assigning to a data instance $s \in S$ a probability value for each symbol in the alphabet P. In our setting, the Reward Machine is a probabilistic Moore Machine taking as input *probabilistic* symbols. Let us assume that all the information in the definition is known. We represent the Moore machine in matrix representation, as described in Section 3, as an input vector v_i encoding the initial state, a transition matrix M_t encoding the transition function, and an output matrix M_o representing the reward function. Given an input sequence of data $s_1, s_2, ..., s_l$, the VRM outputs a sequence of l probability vectors over the machine states $h_1, h_2, ..., h_l$ and a sequence of l probability vectors over the machine outputs (which correspond to reward values) $r_1, r_2, ..., r_l$ as follows.

$$
\begin{aligned}
h_0 &= v_i \\
x_t &= sg(s_i) \\
h_t &= \sum_{p=0}^{p=|P|} x_t[p](h_{t-1} \times M_T[p]) \\
r_t &= h_t \times M_O
\end{aligned}
\tag{3}
$$

Where we denote with $V[i]$ (or v[i]), the component i of matrix (vector) V (v).

4.2. VRM implementation with neural networks

This section describes a neural network architecture implementing Visual Reward Machines.

We define the VRM implementation as a neural network composed of two parts: a convolutional module and a recurrent module. The convolutional module implements the *symbol*

grounding function. This module handles the perception of symbols from non-symbolic states. It is implemented as a softmax-activated Convolutional Neural Network (CNN), having the output layer size equal to the number of symbols. The recurrent module corresponds to a *parametric probabilistic Moore Machine*. This module implements the reward machine with a Recurrent Neural Network (RNN) of a specific structure that allows extracting from the network parameters an equivalent Moore Machine. This network is based on an extension of [14].

In the VRM definition, we use a probabilistic relaxation of both the symbol grounding function and the Reward Machine. These probabilistic relaxations are fundamental for implementing the whole system with neural networks (NN). NNs, in fact, are intrinsically continuous since they learn through a gradient-based optimization process, and they can struggle in learning a 'crispy' symbolic logical model. In our case, the logical model we want to induce from data is the Moore Machine (MM), representing the non-markovian reward. The intuition behind our work is that Probabilistic Moore Machines (PMM) are closely related to Recurrent Neural Networks, since they calculate the next state and output using multiplications between continuous vectors and matrices, in the same way as RNNs do. However, (boolean) machines can also be represented in matrix form, considering their matrix representation comprises only one-hot row vectors. Following this idea, we define the recurrent module as a parametric PMM in which we can drive the representation to be close to one-hot during training, allowing logical induction through backpropagation. We obtain this effect using an activation function that smoothly approximates discrete output, as in prior work [21] [22] [23]. In particular, we use a modified version of the classical softmax activation functions: $softmax_\tau(x, \tau) = softmax(x/\tau)$; where $0 < \tau \leq 1$ is a temperature value controlling the activation steepness: when τ is close to 0, the τ-softmax returns outputs close to one-hot vectors; when τ is 1, the function is the softmax.

Following the VRM implementation with parametric models.

$$\begin{aligned} h_0 &= v_i \\ x_t &= CNN(s_t; \theta_{sg}) \\ h_t &= \sum_{p=0}^{p=|P|} x_t[p](h_{t-1} \times softmax_\tau(\theta_{M_T}, \tau)[p]) \\ r_t &= h_t \times softmax_\tau(\theta_{M_O}, \tau) \end{aligned} \tag{4}$$

where θ_{sg}, θ_{M_T} and θ_{M_O} are the model trainable parameters. θ_{sg} are the symbol grounder parameters, while θ_{M_T} and θ_{M_O} are the parameters of the recurrent module. In particular, θ_{M_T} and θ_{M_O} are two matrices having the same dimensions of matrices M_T and M_O (namely $\theta_{M_T} \in \mathbb{R}^{|P| \times |Q| \times |Q|}$ and $\theta_{M_O} \in \mathbb{R}^{|Q| \times |R|}$) and the $softmax_\tau$ activation is applied to the last matrix dimension. A scheme of the proposed implementation is shown in Figure 1(c).

5. Preliminary experiments

In this section, we report some preliminary experiments that validate the proposed framework. The implementation code can be found at https://github.com/whitemech/VisualRewardMachine.

In particular, we explore several learning procedures. In the first experiment, we test *offline symbol grounding*, which refers to the capability to learn the *symbol grounding function* by leveraging the Moore Machine structure and sequences of states-rewards of a hand-crafted

dataset. For this test, we initialize the Moore Machine parameters θ_{M_T} and θ_{M_O} with the given task specification and train only the CNN weights by minimizing the cross-entropy between the network predictions and the reward labels.

In the second experiment, we test a combination of *RL and online symbol grounding*. In this test, we ground the symbols of the task specifics *on the fly*, meaning we use sequences of image states and rewards collected by the agent while learning to perform the task. We use the learned symbol grounding to proceed on the automaton at each step and estimate the current automaton state. The latter is combined with the environment visual state to form a *Markovian* state representation, which is used by the RL algorithm to estimate the optimal policy.

While this paper mainly focuses on learning the grounding function, the framework is very versatile, and many other learning-reasoning combinations are possible. For instance, we test learning the DFA structure from traces of *imperfectly grounded symbols*, which are symbols for which we have only a probabilistic prediction instead of a strictly boolean interpretation. Although many techniques for DFA induction exist in the literature [24][25][26], and few extensions can handle noise in the output label [27] [28], to the best of our knowledge, there are no extensions to handle noise in the *input symbols*. In our experiments, we simulate the use of an imperfect pretrained classifier for symbol grounding by corrupting the input symbols with Gaussian noise, and we train the recurrent part of the VRM on this noisy dataset. We conduct our preliminary experiments in this direction on Tomita Languages, which are a popular benchmark for DFA induction. We leave the integration of DFA induction with RL in non-Markovian environments for future research.

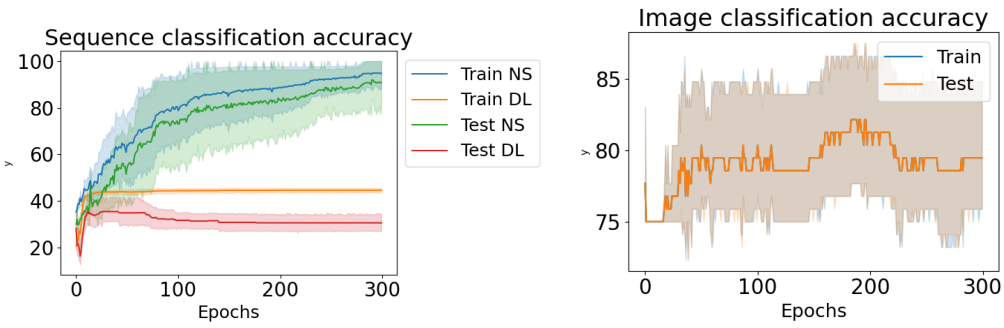
5.1. Visual Minecraft Environment

In this section, we introduce the Visual Minecraft environment that we used to conduct our preliminary experiments, as shown in Figure 1(a). This environment is an example of a non-Markovian task with non-symbolic states, similar to the one considered by [29] and [19], but with image states.

A robotic agent can navigate through a grid world environment using four movement actions: $left, right, up, down$. Each cell of the grid can either be empty or contain one of the following items: *pickaxe*, *gem*, *lava*, or *door*. The task requires the agent to collect a pickaxe and a gem (in any order), and then proceed to the door, while always avoiding the lava. The task specification is represented by the deterministic finite automaton (DFA) shown in Figure 1(b), which is expressed in terms of five symbols, $P = pickaxe, gem, lava, door, empty$. Each symbol indicates whether the corresponding item is present in the cell occupied by the agent.

To express the non-markovian reward function, we transformed the DFA into a Moore machine by assigning a reward function to its states. The reward function is maximum for the final states and decreases as the distance from the final state increases. The values of the reward function are indicated in red on the automaton states in Figure 1(b).

The environment state is an image similar to the one depicted in Figure 1(a), and it does not provide any information about the items collected by the agent up to that point in time, such as the gem or pickaxe. This makes the environment non-Markovian because it is impossible to obtain a comprehensive inventory of all the elements collected by the agent from the current state alone. Previous actions and observations must be considered. Furthermore, an additional



(a) (b)

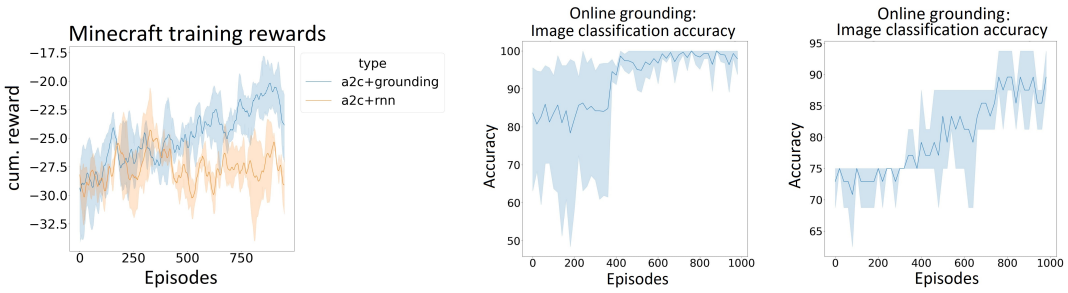
Figure 2: Results obtained by exploiting the reward machine structure to learn the symbol grounding function offline for the Visual Minecraft environment. a) Visual Reward Machine accuracy over sequences. b) Symbol grounding accuracy over single images

challenge is introduced by the fact that the specifics of the task are not directly usable since the symbols are not observable. We do not know how to recognize them in the image.

5.2. Offline symbol grounding

Dataset To apply offline symbol grounding to the Minecraft environment, we created a dataset simulating 40 episodes in the environment and collecting the images rendered by the environment. Each episode lasts for 30 steps, producing a sequence of 30 images that correspond to the environment states and a sequence of 30 reward values that are used to label the image sequence *at each step*. Episodes are balanced between positive and negative examples, with 20 episodes where the agent correctly collects all the items and goes to the door while avoiding the lava, and 20 episodes where it fails. We split the dataset into 80% for training and 20% for testing.

Results Figure 2 shows the train and test sequence classification accuracy in Figure (a) and the symbol grounding accuracy on single images in Figure (b). Let us note that the task specification has two ungroundable symbols: *gem* and *pickaxe*. Since the agent can collect these two items in any order, the framework does not receive enough supervision to distinguish between them, resulting in the image classification not achieving 100% accuracy. However, sequence classification can still achieve top accuracy even if the symbol grounder confuses the gem for the pickaxe and vice versa. We compared our approach with a purely deep-learning-based approach that learns to classify sequences end-to-end using a CNN and an LSTM, which performed poorly in the task, obtaining only 40% sequence accuracy. Investigating the reason for these poor performances, we found that it almost never predicts rewards of -1 and -2, corresponding to the scarcest reward labels in the dataset. In fact, even if we balanced the dataset between positive (reward = 0 in the last step) and negative (reward $<$ 0 in the last step) episodes, the reward labels are not balanced within the episodes. Learning classification tasks from highly biased data with neural networks can be very challenging, as shown in this experiment. However,



(a) (b) (c)

Figure 3: Results obtained by exploiting the reward machine structure to learn the symbol grounding function for the Visual Minecraft environment. a) Training rewards b) Reward prediction accuracy during training. c) Symbol grounding accuracy during training

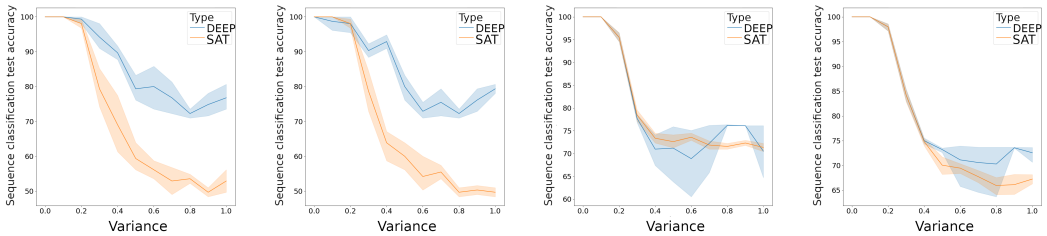
our approach is unaffected by the label imbalance. Additionally, we note that the environment highly biases the distribution of symbols and reward labels in sequences. For example, the 'empty' symbol is much more frequent than the others (because the agent cannot jump from one item to the other, but it has to walk and observe many empty cells in between). Additionally, most possible symbolic traces are unfeasible in the environment and, therefore, never observed. This can complicate the image classification task in case it would be approached with supervised learning.

5.3. Reinforcement Learning and online grounding

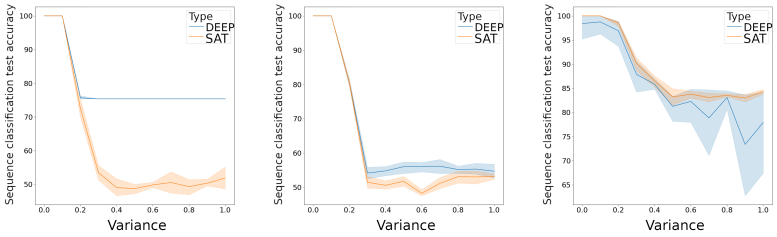
In this second experiment, we perform online symbol grounding and use it to estimate the machine state, which we exploit to speed up policy learning. Unlike in the previous experiments, the dataset is not required to be balanced, as it is collected by the agent while exploring the environment. We construct a Markov state by concatenating features extracted by a CNN from the image with the estimated automaton state. We use Advantage Actor-Critic (A2C) [30] to learn a policy on this state representation. We compare this approach, which exploits the ungrounded DFA specification, with A2C using the state of an LSTM trained end-to-end as the state representation.

Training settings We ran three experiments with three random seeds for each approach and report the mean and standard deviation. We allowed each approach to train for 1000 episodes before stopping the training. We used a learning rate of 0.0007 and an entropy coefficient of 0.0001 as hyperparameters for A2C.

The symbol grounder was trained in the same way described in the previous section, with the newly acquired data, every 40 episodes. The baseline method used a one-layer LSTM of hidden size 256. We tested different hidden sizes for the LSTM (50 and 256) and two layers instead of one, and found that the one-layer LSTM with a hidden size of 256 was the best configuration.



(a) Tomita1 (b) Tomita2 (c) Tomita3 (d) Tomita4



(e) Tomita5 (f) Tomita6 (g) Tomita7

Figure 4: Learning DFA from traces composed of imperfectly grounded symbols: results on the 7 Tomita languages

Results Figure 3 shows the results obtained in the experiment. Figure 3(a) shows the training rewards for our method (A2C+grounding) and the baseline (A2C+RNN). The figure shows that our method outperforms the baseline. Figure 3(b) and (c) show the sequence classification and the symbol grounding classification accuracy of the Visual Reward Machine obtained during the training of the RL agent. The sequence classification accuracy tends to be quite high from the beginning of training, while the symbol grounding classification accuracy achieves top results only after 800 episodes. This is reasonable since the VRM needs to observe some positive episodes to correctly ground all the symbols (e.g., the symbol *door* can be grounded only from a trajectory completing the task), and positive episodes are observable only when the RL module has learned a good policy. Training rewards increase coherently with the improvement in the VRM's performance. Despite these being only preliminary experiments, the results are encouraging. In particular, they confirm our intuition that symbolic temporal specifications can also be exploited in visual tasks for which we do not know the symbol grounding function.

5.4. Learning the machine from imperfectly grounded symbols

In this experiment, we test the capability of our framework to learn DFA specifications from sequences of imperfectly grounded symbols. Specifically, we only trained the recurrent module with traces of symbols and accepted-rejected labels generated by the Tomita languages [31]. We corrupted the one-hot representation of symbols in the training traces by adding Gaussian noise with a mean of zero and variable variance. Figure 4 compares our approach (Deep) with the SAT-based DFA-inductor (SAT) [32]. Since boolean logic induction methods like DFA-inductor

cannot handle probabilistic truth values, we discretize the corrupted inputs to the nearest one-hot vector before passing them to DFA-inductor. The figures show how the two methods' test accuracy (on the y-axis) is affected by different values of noise variance (on the x-axis). In particular, we observe that our method's test accuracy degrades less with increasing noise variance, making it more robust to noise in the symbol grounding compared to DFA-inductor.

6. Conclusion

In conclusion, this paper presents Visual Reward Machines, a neurosymbolic framework that provides non-Markovian rewards for visual tasks. VRMs can be used for both learning and reasoning and are based on the probabilistic relaxation of the symbol grounding function and the logical temporal specification. We have demonstrated that VRMs can leverage the machine's prior knowledge and agent experience to learn the grounding function both offline and during RL training. Specifically, we have shown that we can use the ungrounded specifics to outperform baseline methods based on RNNs. Additionally, we have demonstrated that our model can estimate the finite state machine from noisy input symbol sequences, which can occur when we can only evaluate an imperfect pretrained symbol classifier. In future research, we aim to provide further experimental evaluation to support our method and integrate DFA-induction with Reinforcement Learning in non-Markovian environments to handle partially known symbol-grounding functions and unknown temporal specifications.

Acknowledgments

This work is partially supported by the ERC Advanced Grant WhiteMech (No. 834228), by the EU ICT-48 2020 project TAILOR (No. 952215), by the PRIN project RIPER (No. 20203FFYLK) and by the PNRR MUR project PE0000013-FAIR.
This work has been carried out while Francesco Argenziano and Aymeric Barbin were enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome.

References

[1] F. Bacchus, C. Boutilier, A. Grove, Rewarding behaviors, Portland, OR, 1996, pp. 1160–1167. URL: behaviors.pdf.

[2] N. Heess, J. J. Hunt, T. P. Lillicrap, D. Silver, Memory-based control with recurrent neural networks, CoRR abs/1512.04455 (2015). URL: http://arxiv.org/abs/1512.04455. arXiv:1512.04455.

[3] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

[4] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, R. Munos, Recurrent experience replay in distributed reinforcement learning, in: International Conference on Learning Representations, 2019. URL: https://openreview.net/forum?id=r1lyTjAqYX.

[5] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 46–57. URL: https://doi.org/10.1109/SFCS.1977.32. doi:10.1109/SFCS.1977.32.

[6] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, p. 854–860.

[7] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, S. A. McIlraith, Ltl and beyond: Formal languages for reward function specification in reinforcement learning, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 6065–6073. URL: https://doi.org/10.24963/ijcai.2019/840. doi:10.24963/ijcai.2019/840.

[8] C. Wang, Y. Li, S. L. Smith, J. Liu, Continuous motion planning with temporal logic specifications using deep neural networks, 2020. URL: https://arxiv.org/abs/2004.02610. doi:10.48550/ARXIV.2004.02610.

[9] M. Gaon, R. Brafman, Reinforcement learning with non-markovian rewards, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 3980–3987. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5814. doi:10.1609/aaai.v34i04.5814.

[10] Z. Xu, B. Wu, A. Ojha, D. Neider, U. Topcu, Active finite reward automaton inference and reinforcement learning using queries and counterexamples, in: Machine Learning and Knowledge Extraction - 5th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, August 17-20, 2021, Proceedings, 2021, pp. 115–135. URL: https://doi.org/10.1007/978-3-030-84060-0_8. doi:10.1007/978-3-030-84060-0_8.

[11] A. Ronca, G. P. Licks, G. D. Giacomo, Markov abstractions for PAC reinforcement learning in non-markov decision processes, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, 2022, pp. 3408–3415. URL: https://doi.org/10.24963/ijcai.2022/473. doi:10.24963/ijcai.2022/473.

[12] D. Furelos-Blanco, M. Law, A. Jonsson, K. Broda, A. Russo, Induction and exploitation of subgoal automata for reinforcement learning, J. Artif. Int. Res. 70 (2021) 1031–1116. URL: https://doi.org/10.1613/jair.1.12372. doi:10.1613/jair.1.12372.

[13] E. Umili, R. Capobianco, G. D. Giacomo, Grounding ltlf specifications in images, in: Proceedings of the 16th International Workshop on Neural-Symbolic Learning and Reasoning as part of the 2nd International Joint Conference on Learning & Reasoning (IJCLR 2022), Cumberland Lodge, Windsor Great Park, UK, September 28-30, 2022, 2022, pp. 45–63. URL: http://ceur-ws.org/Vol-3212/paper4.pdf.

[14] E. Umili, R. Capobianco, Deepdfa: a transparent neural network design for dfa induction, 2023. doi:10.13140/RG.2.2.25449.98401.

[15] M. L. Littman, U. Topcu, J. Fu, C. L. I. Jr., M. Wen, J. MacGlashan, Environment-independent task specifications via GLTL, CoRR abs/1704.04341 (2017). URL: http://arxiv.org/abs/1704.04341. arXiv:1704.04341.

[16] G. De Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications, Proceedings of the International Conference on Automated Planning and Scheduling 29 (2021) 128–136. URL:

https://ojs.aaai.org/index.php/ICAPS/article/view/3549.

[17] M. Cai, S. Xiao, B. Li, Z. Li, Z. Kan, Reinforcement learning based temporal logic control with maximum probabilistic satisfaction, 2021 IEEE International Conference on Robotics and Automation (ICRA) (2020) 806–812.

[18] C. K. Verginis, C. Köprülü, S. Chinchali, U. Topcu, Joint learning of reward machines and policies in environments with partially known semantics, CoRR abs/2204.11833 (2022). URL: https://doi.org/10.48550/arXiv.2204.11833. doi:10.48550/arXiv.2204.11833. arXiv:2204.11833.

[19] A. C. Li, Z. Chen, P. Vaezipoor, T. Q. Klassen, R. T. Icarte, S. A. McIlraith, Noisy symbolic abstractions for deep RL: A case study with reward machines, CoRR abs/2211.10902 (2022). URL: https://doi.org/10.48550/arXiv.2211.10902. doi:10.48550/arXiv.2211.10902. arXiv:2211.10902.

[20] Y. Kuo, B. Katz, A. Barbu, Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of LTL formulas, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021, 2020, pp. 5604–5610. URL: https://doi.org/10.1109/IROS45743.2020.9341325. doi:10.1109/IROS45743.2020.9341325.

[21] E. Jang, S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017. URL: https://openreview.net/forum?id=rkE3y85ee.

[22] H. Walke, D. Ritter, C. Trimbach, M. Littman, Learning finite linear temporal logic specifications with a specialized neural operator, 2021. URL: https://arxiv.org/abs/2111.04147. doi:10.48550/ARXIV.2111.04147.

[23] R. Kusters, Y. Kim, M. Collery, C. de Sainte Marie, S. Gupta, Differentiable rule induction with learned relational features, in: A. d'Avila Garcez, E. Jiménez-Ruiz (Eds.), Proceedings of the 16th International Workshop on Neural-Symbolic Learning and Reasoning as part of the 2nd International Joint Conference on Learning & Reasoning (IJCLR 2022), Cumberland Lodge, Windsor Great Park, UK, September 28-30, 2022, volume 3212 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 30–44. URL: http://ceur-ws.org/Vol-3212/paper3.pdf.

[24] D. Angluin, Learning regular sets from queries and counterexamples, Information and Computation 75 (1987) 87–106. URL: https://www.sciencedirect.com/science/article/pii/0890540187900526. doi:https://doi.org/10.1016/0890-5401(87)90052-6.

[25] M. J. H. Heule, S. Verwer, Exact dfa identification using sat solvers, in: Proceedings of the 10th International Colloquium Conference on Grammatical Inference: Theoretical Results and Applications, ICGI'10, Springer-Verlag, Berlin, Heidelberg, 2010, p. 66–79.

[26] K. J. Lang, B. A. Pearlmutter, R. Price, Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm, in: ICGI 1998: Grammatical Inference, number 1433 in Lecture Notes in Computer Science book series (LNCS), Springer, 1998, pp. 1–12. URL: https://mural.maynoothuniversity.ie/10250/, cite as: Lang K.J., Pearlmutter B.A., Price R.A. (1998) Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In: Honavar V., Slutzki G. (eds) Grammatical Inference. ICGI 1998. Lecture Notes in Computer Science, vol 1433. Springer, Berlin, Heidelberg.

[27] S. Lucas, T. Reynolds, Learning deterministic finite automata with a smart state labeling evolutionary algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 1063–1074. doi:10.1109/TPAMI.2005.143.

[28] V. I. Ulyantsev, I. Zakirzyanov, A. A. Shalyto, Bfs-based symmetry breaking predicates for dfa identification, in: Language and Automata Theory and Applications, 2015.

[29] J. Corazza, I. Gavran, D. Neider, Reinforcement learning with stochastic reward machines, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022) 6429–6436. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20594. doi:10.1609/aaai.v36i6.20594.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, PMLR, 2016, pp. 1928–1937.

[31] M. Tomita, Dynamic construction of finite automata from examples using hill-climbing, in: Proceedings of the Fourth Annual Conference of the Cognitive Science Society, Ann Arbor, Michigan, 1982, pp. 105–108.

[32] I. Zakirzyanov, A. Morgado, A. Ignatiev, V. I. Ulyantsev, J. Marques-Silva, Efficient symmetry breaking for sat-based minimum dfa inference, in: LATA, 2019.