Reinforcement Learning with External Knowledge by using Logical Neural Networks

Daiki Kimura , Subhajit Chaudhury , Akifumi Wachi , Ryosuke Kohita ,
Asim Munawar , Michiaki Tatsubori , Alexander Gray
IBM Research AI
daiki@jp.ibm.com

Abstract

Conventional deep reinforcement learning methods are sample-inefficient and usually require a large number of training trials before convergence. Since such methods operate on an unconstrained action set, they can lead to useless actions. A recent neuro-symbolic framework called the Logical Neural Networks (LNNs) can simultaneously provide key-properties of both neural networks and symbolic logic. The LNNs functions as an end-to-end differentiable network that minimizes a novel contradiction loss to learn interpretable rules. In this paper, we utilize LNNs to define an inference graph using basic logical operations, such as AND and NOT, for faster convergence in reinforcement learning. Specifically, we propose an integrated method that enables model-free reinforcement learning from external knowledge sources in an LNNs-based logical constrained framework such as action shielding and guide. Our results empirically demonstrate that our method converges faster compared to a model-free reinforcement learning method that doesn't have such logical constraints.

1 Introduction

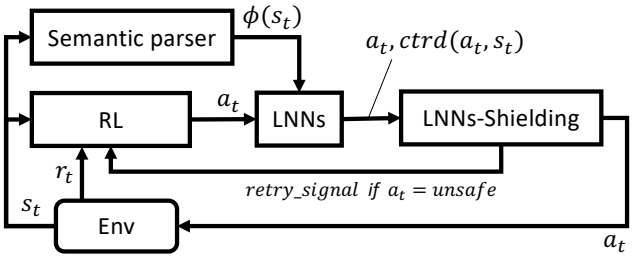
Deep reinforcement learning methods have been successfully applied to many applications, particularly computer game, text-based game, and robot control applications [Mnih *et al.*, 2015; Narasimhan *et al.*, 2015; Silver *et al.*, 2017; Kimura, 2018; Pathak *et al.*, 2017; Yuan *et al.*, 2018; Kimura *et al.*, 2018]. Such methods require a large number of training trials for converging to an optimal action policy. By default, due to a lack of external constraints, they cannot avoid unsafe or useless actions. If an agent receives the proper action list (recommendations for action), it can reduce the number of training trials. We believe we can prepare such an action list from external action constraints pertaining to the environment. Another option is safe reinforcement learning [Garcıa and Fernández, 2015], which can avoid taking unsafe and useless actions via the action constraints.

To define such action constrains, there are various techniques in reinforcement learning that use symbolic logical functions or graphs [Hasanbeig *et al.*, 2018; Hasanbeig *et al.*, 2020]. However, these techniques require all rules to be set manually, which is time-consuming. A recent neuro-symbolic framework called the Logical Neural Networks (LNNs) [Riegel *et al.*, 2020] simultaneously provides key-properties of both neural networks (learning) and symbolic logic (reasoning). It can train the constraints and rules with logical functions in the neural networks, and since every neuron in the LNNs has a component for a formula of weighted real-valued logics, it can calculate the probability and contradiction loss for each of the propositions. At the same time, trained LNNs follow symbolic rules, which means they yield a highly interpretable disentangled representation. In this paper, we define the external knowledge for the reinforcement learning within this LNNs structure, and then leverage the knowledge in the LNNs.

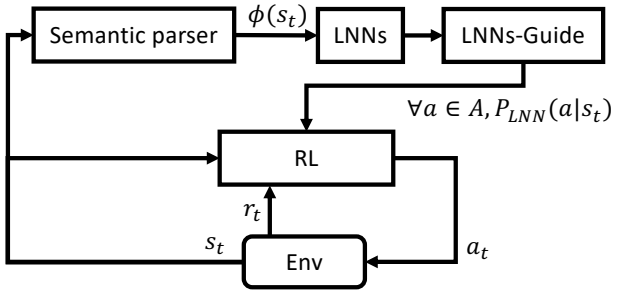
We propose an integrated reinforcement learning method with external knowledge for action shielding (avoiding useless action) and guiding (giving an action recommendation) that is defined in the LNNs. The performance of our proposed method is experimentally compared with a baseline method that doesn't use external knowledge. Our main contributions are (1) the proposal of an integrated method that uses logical guides for fast training and safe reinforcement learning via the trainable logical network and (2) the experimental demonstration for the effectiveness of external knowledge in reinforcement learning by using LNNs.

2 Proposed Method

In this paper, we proposed two methods: safe reinforcement learning by LNNs (LNNs-Shielding) and action-guided reinforcement learning by LNNs (LNNs-Guide). Figure 1 shows the architectures of reinforcement learning with these methods, we provide a detailed explanation of each in the following subsections. Since we apply these methods in a text-based game environment that is specifically Coin-collector game of TextWorld [Côté *et al.*, 2018], our explanations include some examples from on the game. In Coin-collector game, the agent needs to find and take a coin in series of connected rooms and take the coin. Hence, the agent needs to travel though various rooms while seeking the coin.



i. Reinforcement learning with LNNs-Shielding



ii. Reinforcement learning with LNNs-Guide

Figure 1: Architecture of reinforcement learning with proposed methods.

2.1 LNNs-Shielding

LNNs-Shielding avoids unsafe or useless actions that are defined by logical conditions in the LNNs for reinforcement learning. For example, the agent had better not to take the action of "`go to west room`", if it has already visited the west room. Action shielding needs to be represented in logical operations, hence the logical function for this example is "`visited west room`" $\wedge$ "`found west room`" $\Rightarrow \neg$ "`go to west room`" ($\wedge$: AND operator, $\Rightarrow$: IMPLY operator, $\neg$: NOT operator). This shielding means that even if the agent has found the west room, it will not go to the west room if the west room has already been visited. In the LNNs, the agent first checks the current state values, then if it has already visited the west room, the agent set a true value for proposition of "`visited west room`", that means "`visited west room`" = *true*. If the agent find exit for the west room, it also set a true value for "`found west room`". Then the LNNs have a logical function for this proposition, which is "`found west room`" $\Rightarrow$ "`go to west room`".

At same time, the agent inputs current state values to RL method to obtain the action. If the RL method outputs "`go to west room`" as a selected action, the agent set a *true* value for a proposition of "`go to west room`" action in LNNs. If the agent visited and found the west room ("`visited west room`" = "`found west room`" = *true*) and RL method outputs the "`go to west room`" as a selected action, the "`go to west room`" proposition will observe a contradiction in the proposition. Because the proposition was set *true* value from RL method by the selected action candidate, it was also set *false* from logical function ("`found west room`" $\Rightarrow$ "`go to west room`"). We assume such action restrictions will help lead to faster convergence in reinforcement learning.

Let $\phi(s_t)$ be a logical propositional state input for current state s_t from a semantic parser algorithm. We can obtain these logical state values from raw text descriptions via the semantic parser such that the statement "`found west room`" is *true* when the state description has "`There is an unguarded exit to the west.`" [1]. Let $\langle lower_{n_i,s_t}, upper_{n_i,s_t} \rangle = LNN(n_i|\phi(s_t))$ be *lower* and *upper* bound values from the LNNs for a node n_i and input $\phi(s_t)$. All neurons in the LNNs return pairs of values in the range $[0, 1]$ representing *lower* and *upper* bounds on the truth values of their corresponding subformulae and propositions [Riegel *et al.*, 2020]. These values are updated using an inference function from given propositional inputs. Note that the weight and bias values in the connections are updated during the back-propagation operations. Normally, the *upper* bound is higher than the *lower* bound. However, if a neuron is observed to be contradicting the logical rules, the *lower* bound will be higher than the *upper* bound. Therefore, the contradiction value $ctrd(a_t, s_t)$ for the node n_i is defined as

$$ctrd(a_t, s_t) = \sum_{n_j \in \mathrm{to}(a_t)} \max(0, lower_{n_j,s_t} - upper_{n_j,s_t}), \tag{1}$$

where a_t represents a node for an action value at t time step from the model-free reinforcement learning method, and $\mathrm{to}(a_t)$ is all nodes connected to node a_t.

In reinforcement learning, the agent calculates this contradiction value from a given action a_t and state s_t from the model-free LSTM-DQN++ [Yuan *et al.*, 2018] reinforcement learning method. The proposed LNNs-Shielding distinguishes whether the given action is *safe* (useful) or *unsafe* (useless) from the inference result in the LNNs. The action a_t will be discriminated as a *safe* action if the contradiction value $ctrd(a_t, s_t)$ is α or higher. The action a_t will be discriminated as an *unsafe* action if the condition value is less than α. When the action a_t is a *safe* action, the agent executes action a_t. Alternatively, when it is an *unsafe* action, the LNNs-Shielding returns the action a_t to the reinforcement learning module, and the module then calculates the next candidate for proper action. Note that the contradiction value of this next candidate will also be checked by the LNNs-Shielding. The action policy training from the reward signal is then performed by the reinforcement learning method.

2.2 LNNs-Guide

LNNs-Guide recommends the suitable actions that are defined by the logical functions in LNNs for reinforcement learning. For example, if the LNNs are trained similar rules to those in the previous LNNs-Shielding example, the LNNs-Guide can give a negative recommendation (similar to the

[1] The semantic parser is not our focus in this paper.

shielding) for the visited room. At the same time, the LNNs-Guide can recommend taking a "`go to west room`" action when the agent has found the west room, which is a positive recommendation. For this example, we implement this rule on various logical operations such as "`found west room`" $\Rightarrow$ "`go to west room`". LNNs-Guide, therefore, can additionally give positive recommendations compared to LNNs-Shielding. We assume such an action recommendation will help lead to faster convergence in reinforcement learning, and it is more effective than the LNNs-Shielding since it also has the positive recommendation function.

Let $\phi(s_t), lower_{n_i,s_t}, upper_{n_i,s_t}, ctrd(a_t, s_t)$ be the same as the definitions in the LNNs-Shielding method. The LNNs-Guide provides probabilities for recommendation of the action for each state input. The probability is calculated by

$$P_{LNN}(a|s_t) = \frac{e^{v(a,s_t)}}{\sum_{a_j \in A} e^{v(a_j,s_t)}}, \quad (2)$$

$$v(a, s_t) = \frac{lower_{a,s_t} + upper_{a,s_t}}{2} - ctrd(a, s_t), \quad (3)$$

where a is a targeted action for calculating the probability, and A is all actions. Value $v(a, s_t)$ represents the level of truth values for the propositions while discounting the contradiction value.

In reinforcement learning, the policy follows the probabilities calculated with the epsilon greedy algorithm in LSTM-DQN++ [Yuan *et al.*, 2018]. In this work, we select the action a_t by

$$a_t = \begin{cases} \text{argmax}_{a \in A} P_{LNN}(a|s_t)\, Q(s_t, a) & : \zeta \geq \epsilon \\ \text{random}_{a \in A} P_{LNN}(a|s_t) & : \text{otherwise} \end{cases}, \quad (4)$$

where ζ is the current random value for epsilon greedy, and $\text{random}_{a \in A} P$ takes an action in accordance with probabilities P. In this equation, an action is selected on the basis of q-value and action probabilities from LNNs-Guide. Training steps by the reward signal are performed in the same way as in the reinforcement learning method.

3 Evaluation

3.1 Experiments

We evaluated the performance of the proposed method through experiments conducted in the TextWorld environment [Côté *et al.*, 2018]. Our target was the Coin-collector game. We built LNNs that has useful knowledge based on external data. We fit the prepared data to the following rules and then trained the LNNs.

- ¬ "`visited all connected rooms`" $\wedge$ "`no coin in east room`" $\Rightarrow$ ¬ "`go east`"
- "`found east room`" $\Rightarrow$ "`go east`"
- ¬ "`visited all connected rooms`" $\wedge$ "`no coin in west room`" $\Rightarrow$ ¬ "`go west`"
- "`found west room`" $\Rightarrow$ "`go west`"

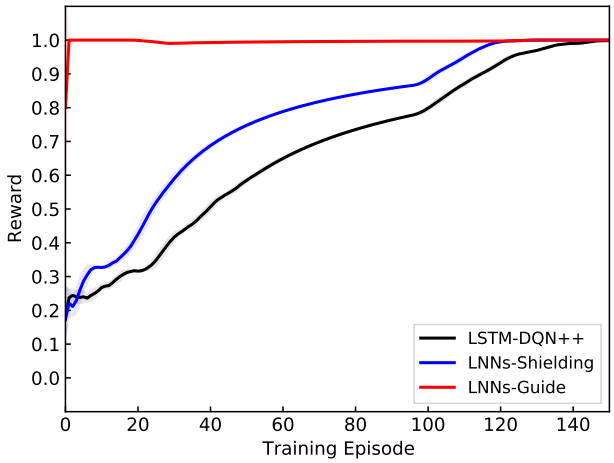


Figure 2: Reward [0-1] curves for proposed and baseline methods with moving average ($N = 5$). Shaded area represents the standard deviation value.

- ¬ "`visited all connected rooms`" $\wedge$ "`no coin in south room`" $\Rightarrow$ ¬ "`go south`"
- "`found south room`" $\Rightarrow$ "`go south`"
- ¬ "`visited all connected rooms`" $\wedge$ "`no coin in north room`" $\Rightarrow$ ¬ "`go north`"
- "`found north room`" $\Rightarrow$ "`go north`"
- "`found coin in the room`" $\Rightarrow$ "`take coin`"

The reason we have ¬ "`visited all connected rooms`" as a rule is that the agent might have any preferred action at the dead-end of a path. The agent can go back to the visited room with this proposition when it is at the dead-end. We prepared the LSTM-DQN++ [Yuan *et al.*, 2018] method as a baseline method and tested it along with the proposed methods (LNNs-Shielding, LNNs-Guide). We set $\alpha = 1$ for LNNs-Shielding.

3.2 Results

Figure 2 shows the reward curves from the proposed and baseline methods. The LNNs-Shielding avoided previously visited rooms thanks to the rules in LNNs, such as "`visited all connected rooms`" $\wedge$ "`no coin in east room`" $\Rightarrow$ ¬ "`go east`", so it had a better learning convergence than the baseline method. The LNNs-Guide converged the training extremely fast thanks to the effect of the positive recommendation, such as "`found east room`" $\Rightarrow$ "`go east`". The reason LNNs-Shielding was weaker than LNNs-Guide is that, while LNNs-Shielding only prohibited an action when the given action had a high contradiction value, the LNNs-Guide provided action recommendations at every time step. This result is in line with our expectation and leads us to conclude that LNNs-Guide is the superior method for utilizing external knowledge. However, we also feel that LNNs-Guide may produce weaker results due to incorrect or fuzzy rules in the LNNs. To alleviate such concerns, we believe the interpretability of the rules in LNNs, which is the key benefit of LNNs, would be helpful to confirm the correctness of the trained rules.

4 Conclusion

In this work, we have proposed a method that utilizes external knowledge represented in trainable logical neural networks and demonstrated through experiments that is has better convergence compared to a baseline method. For future work, we plan to apply this method to other complex games and train the policy directly in logical neural networks.

References

[Côté *et al.*, 2018] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.

[Garcıa and Fernández, 2015] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[Hasanbeig *et al.*, 2018] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.

[Hasanbeig *et al.*, 2020] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. *arXiv preprint arXiv:2002.12156*, 2020.

[Kimura *et al.*, 2018] Daiki Kimura, Subhajit Chaudhury, Ryuki Tachibana, and Sakyasingha Dasgupta. Internal model from observations for reward shaping. 2018.

[Kimura, 2018] Daiki Kimura. Daqn: Deep auto-encoder and q-network. *arXiv preprint arXiv:1806.00630*, 2018.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[Narasimhan *et al.*, 2015] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *Association for Computational Linguistics*, 2015.

[Pathak *et al.*, 2017] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[Riegel *et al.*, 2020] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. Logical neural networks, 2020.

[Silver *et al.*, 2017] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017.

[Yuan *et al.*, 2018] X. Yuan, Marc-Alexandre Côté, Alessandro Sordoni, R. Laroche, Remi Tachet des Combes, Matthew J. Hausknecht, and Adam Trischler. Counting to explore and generalize in text-based games. *ArXiv*, abs/1806.11525, 2018.