Curriculum Learning and Theorem Proving

Zsolt Zombori[1], Adrián Csiszárik[1], Henryk Michalewski[3], Cezary Kaliszyk[2], and Josef Urban[4]

[1] Alfréd Rényi Institute of Mathematics, Budapest
[2] University of Innsbruck
[3] University of Warsaw, deepsense.ai
[4] Czech Technical University in Prague

1 Introduction

We propose a proof guidance approach for ATP systems based on reinforcement learning (RL). Unlike previous approaches, we specifically target long proofs, i.e., our aim is to be able to find proofs that are hundreds or even thousands of steps long.

2 Learning Setup

Theorem proving can be seen as a special kind of reinforcement learning environment where states are proof states and actions are inference rules that are applicable in a given state. What makes this setup particularly challenging is that proof trajectories can be long with extremely sparse rewards: only once a proof has been found.

We propose to use curriculum learning on the length of proofs to counterbalance sparse rewards. We assume that we have a few problems for which proof trajectories are already available. We start exploration from the end of these trajectories, where rewards are reasonably close, then we gradually move backwards towards the beginning of the proof. This approach has already been successfully applied to Atari games in [5] and in many other reinforcement learning experiments.

Another peculiarity of theorem proving as a reinforcement learning environment is that the action space is discrete, but potentially infinite. Depending on the proof system, the set of actions can grow indefinitely during the proof (such as selecting the given clause in saturation-style provers), while in other systems the action space is fixed for a given problem (which is the case in many tableau calculi). However, even in the latter scenario, actions can be different from problem to problem.

In this current work we focus on providing guidance for LeanCoP/FCoP [4, 7, 2, 3] theorem prover that is based on the connection tableau. In FCoP, the action space is roughly correlated with the size of the problem set, extended with axioms to handle equality. While this can be large for large problems, only a few actions are available in a particular state. We describe in Section 3 how we approach this action space.

A well known challenge of combining theorem provers with learning systems is embedding discrete proof objects (states and actions) into continuous Euclidean spaces that learners can work on. We use the hand engineered feature representation that is provided by FCoP [3], mostly based on triples occurring in the proof tree, first used in [1]. In the longer run, we believe that finding better representations can yield significant improvement.

We focus on Proximal Policy Optimization (PPO) [6], a successful reinforcement learning technique, which is a variant of the actor-critic framework. We jointly train a value and a policy model that evaluate proof states and actions, respectively. In PPO, the policy is forced to stay close to its previous version at each update, which results in a more stable training.

3 Handling the Action Space

As described earlier, we are dealing with a discrete action space that can be large and that may not be known in advance. This does not fit into the traditional RL setups where either 1) the action space is small and fixed and a policy can output a probability vector, or 2) the action space is continuous and the policy can directly parameterize actions.

We have briefly experimented with using a fixed action space, which can be used for narrow problem types, such as addition/multiplication problems in Robinson arithmetic. Here the policy can directly compute a probability for all actions, irrespective of whether the action is a valid move.

However, most of our work focuses on a more robust setup, that we called the *action selector model*. The policy receives the state, as well as all available actions as input and returns a probability vector over the available actions. This model can easily generalize to yet unseen actions, since actions appear as inputs and the model just selects one of them. It can also handle large action spaces, provided not too many actions are available at once. Also note that the policy can make decisions based on what actions are available. Another advantage of this approach is that we exploit the features of actions, not only those of the space. Instead of having to learn “Given such state features, you should make the this action”, we now can learn “If state and action features are related in a particular way, then this is the action to be selected”. The fact that all proof objects are embedded into the same feature space, makes this approach particularly comfortable. This model is very similar to the action selection in [3].

Implementing the action selector model poses technical difficulties, since different number of actions are available at each step. However, we can exploit the fact that at any proof state only a few actions are available, which can be upper bounded by a small constant C. The policy always assumes C action inputs and the unused slots are filled with zeros (empty actions).

4 Experiments

In our first set of experiments, we wanted to show the feasibility of our approach on problems that require long proofs, but that have a rather simple structure. These experiments target arithmetic formulae in Robinson arithmetic. We use successor notation, which yields very long expressions. Trying other representations is future work. For these problems, we upper-bounded the number of available actions by 22, though typically it was not more than 5-6.

Our learners are implemented in Python and the FCoP theorem prover is accessed through its Python interface PyCoP. We experimented with decision trees and neural networks and the latter provided better results. Our networks are simple multi-layer perceptrons, using 4 layers of 512 neurons, with ReLU nonlinearity.

4.1 Generalizing from Long Proofs

We trained on the proof of $7 \times 2 = 14$ (51 steps) and successfully generalized to formulae $A \times B = C$, $A + B = C$ with arbitrarily large numbers. E.x. the proof of $29 \times 29 = 841$ (2641 steps) was found in 13.5 sec. It is instructive to see what this time was spent on: FCoP proof steps (3 sec), guidance neural network evaluation (9 sec), feature extraction (1.5 sec). Training took around 10^5 steps.

4.2 Generalizing from Short Proofs

We were interested to see how large the training problem needs to be in order to generalize. With a little tuning, we managed to obtain the very same results of Subsection 4.1 when trained on $1 \times 1 = 1$ (9 steps). This opens up the possibility of fully unsupervised learning: the 9 step long proof of this simple problem can be found with various brute force methods, which then can be used to learn general addition and multiplication. Training took 2×10^5 steps.

4.3 Generalizing to Complex Formulae

Next we experimented with more complex formulae of the shape $F = N$, where N is a number and F contains a random sequence of operators and operands. We generated a set S of 66 problems with 3 operators and each operand bounded to be in $[0, 2]$. After training on $1 \times 1 = 1$, the system could prove most of S and we continued training on them. By the end of this two step training process, we generated new random sequences with larger numbers and more operators, all of which the system could solve. Training took around 2×10^6 steps. Some examples of test problems:

- $((8 + 5) \times 8) \times 5 = 520$: 16856 steps, 95.7 sec
- $((7 \times 9) + 3) \times 8 = 528$: 13937 steps, 78.9 sec

5 Conclusion

We built a reinforcement learning system using PPO that learns to provide guidance for the FCoP connection prover. The system learns to prove equations involving addition and multiplication in Robinson arithmetic from extremely little supervision and can generalize to complex formulae. In the future, we hope to be able to extend these results to more complex problem domains.

References

[1] Jan Jakubuv and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In *CICM*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[2] Cezary Kaliszyk and Josef Urban. FEMaLeCoP: Fairly efficient machine learning connection prover. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 88–96. Springer, 2015.

[3] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olsák. Reinforcement learning of theorem proving. In *NeurIPS*, pages 8836–8847, 2018.

[4] Cezary Kaliszyk, Josef Urban, and Jiři Vyskočil. Certified connection tableaux proofs for hol light and tptp. In *Proceedings of the 2015 Conference on Certified Programs and Proofs*, CPP '15, pages 59–66, New York, NY, USA, 2015. ACM.

[5] Tim Salimans and Richard Chen. https://blog.openai.com/learning-montezumas-revenge-from-a-single-demonstration/.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[7] Josef Urban, Jirí Vyskocil, and Petr Stepánek. MaLeCoP: Machine learning connection prover. In Kai Brünnler and George Metcalfe, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, volume 6793 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2011.