Curriculum Learning and Theorem Proving

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

[1]Alfréd Rényi Institute o Mathematics, Hungarian Academy of Sciences

[2]University of Warsaw, deepsense.ai

[3]University of Innsbruck [4]Czech Technical University in Prague

Motivation

1. ATPs tend to only find short proofs - even after learning
2. AITP systems typically trained/evaluated on large proof sets - hard to see what the system has learned

- Can we build a system that learns to find longer proofs?
- What can be learned from just a few (maybe one) proof?

Aim

- Build an internal guidance system for theorem proving
- Use reinforcement learning
- Train on a single problem
- Try to generalize to long proofs with very similar structure

Domain: Robinson Arithmetic

```
%theorem: mul(1,1) = 1
fof(zeroSucc, axiom, ! [X]: (o != s(X))).
fof(diffSucc, axiom, ! [X,Y]: (s(X) != s(Y) | X = Y)).
fof(addZero, axiom, ! [X]: (plus(X,o) = X)).
fof(addSucc, axiom, ! [X,Y]: (plus(X,s(Y)) =
    s(plus(X,Y)))).
fof(mulZero, axiom, ! [X]: (mul(X,o) = o)).
fof(mulSucc, axiom, ! [X,Y]: (mul(X,s(Y)) =
    plus(mul(X,Y),X))).
fof(myformula, conjecture, mul(s(o),s(o)) = s(o)).
```

- Proofs are non trivial, but have a strong structure
- See how little supervision is required to learn some proof types

Challenge for Reinforcement learning

- Theorem proving provides sparse, binary rewards
- Long proofs provide extremely little reward

Idea

- Use curriculum learning
- Start learning from the end of the proof
- Gradually move starting step towards the beginning of proof

Reinforcement Learning Approach

- Proximal Policy Optimization (PPO)
- Actor - Critic Framework
- Actor learns a policy (what steps to take)
- Critic learns a value (how promising is a proof state)
- Actor is confined to change slowly to increase stability

PPO challenges

- Action space is not fixed (different at each step)
- Action space cannot be directly parameterized
- Guidance cannot "output" the correct action
- Guidance takes the state - action pair as input and returns a score

Technical Details

- ATP: LeanCoP (ocaml / prolog)
  - Connection tableau based
  - Available actions are determined by the axiom set (does not grow)
  - Returns (hand designed) Enigma features
- Machine learning in python
- Learner is a 3-4 layer deep neural network
- PPO1 implementation of Stable Baselines

Evaluation: STAGE 1

- $N_1 + N_2 = N_3$, $N_1 \times N_2 = N_3$
- Enough to find a good ordering of the actions
- Can be fully mastered from the proof of $1 \times 1 = 1$
- Useful:
  - Some reward for following the proof

Evaluation: STAGE 2

- RandomExpr $= N$
- Features from the current goal become important
- Couple "rare" actions
- Can be mastered from the proof of $1 \times 1 \times 1 = 1$
- Useful:
  - Features from the current goal
  - Oversample positive trajectories

Evaluation: STAGE 3

- $\text{RandomExpr}_1 = \text{RandomExpr}_2$
- More features required
- "Rare" events tied to global proof progress
- Trained on 4-5 proofs, we can learn 90% of problems
- Useful:
    - Features from the path
    - Features from other open goals
    - Features from the previous action
    - Random perturbation of the curriculum stage
    - Train on several proofs in parallel

Future work

- Extend Robinson arithmetic with other operators
- Learn on multiple proofs to master multiple strategies in parallel
- Try some other RL approaches
- Move beyond Robinson