Project Report: Reinforcement Learning

Ritish Shrirao IMT2022003
Aditya Priyadarshi IMT2022075
Kandarp Dave IMT2022115

Contents

1 Introduction

Contrastive Policy Learning (ConPoLe) is a novel reinforcement learning approach designed to solve symbolic reasoning tasks such as algebra, arithmetic, and logic, where traditional RL methods struggle due to sparse rewards and unstructured textual inputs. The authors frame symbolic domains as deterministic environments with binary rewards and propose ConPoLe to learn policies by directly optimizing a contrastive loss (InfoNCE), which encourages distinguishing successful action sequences from unsuccessful ones.

LEMMA is a reinforcement learning method that augments agents with learned symbolic abstractions to solve complex mathematical reasoning tasks such as equation solving and fraction simplification. Traditional RL agents struggle with these tasks due to long action sequences and sparse rewards. LEMMA addresses this by automatically discovering high-level, reusable abstractions from prior successful solutions and incorporating them into the agent's action space.

The code for our experiments and modifications is available on github . Please refer to the README.md file for instructions on how to run the code.

2 Literature Review

1. **ConPoLe**

   The ConPoLe paper addresses symbolic reasoning tasks by formulating them as Reinforcement Learning (RL) problems. The core idea is to learn a policy that can solve problems step-by-step, like simplifying an algebraic equation. The environment is modeled as a deterministic Markov Decision Process (MDP):

   - **States (S)**: Problem states are represented as unstructured text (e.g., "2x + 1 = 5").
   - **Actions ($A(s)$)**: Actions are applications of low-level axioms (e.g., "subtract 1 from both sides").
   - **Transition Function ($T(s_t, a)$)**: Deterministic, mapping a state s_t and action a to a new state s_{t+1}.
   - **Reward Function ($R(s)$)**: A binary reward is given: $R(s) = 1$ if the state s represents a solved problem (e.g., "x = 2"), and $R(s) = 0$ otherwise. Solved states are terminal.
   - **Policy ($\pi(a|s)$)**: The agent learns a policy to select actions that maximize the probability of reaching a solved state.

   Traditional RL algorithms struggle due to sparse rewards (only at the end) and the vast state-action space. ConPoLe overcomes this by using contrastive learning. During training, it uses beam search with the current policy to find successful trajectories (solutions) and nearly successful/failed trajectories. For each step in a successful trajectory, the action taken is considered a "positive" example. Other actions available at that step, or actions from failed branches in the beam search, are considered "negative" examples.

   ConPoLe directly optimizes the InfoNCE loss. This loss function serves to maximize a *lower bound* on the mutual information between the current state (s_t) and the positive next state (s_{t+1}) that continues on a path to the solution. The algorithm trains a parametric scoring function $f_\theta(p, s_t)$ to assign a higher score to the actual next state s_{t+1} on the solution path (the positive example) compared to other candidate next states p_i (the negative examples) drawn from a set $X = \{s_{t+1}, p_1, ..., p_N\}$. The scoring function is typically a log-bilinear model using state embeddings $\phi_\theta(\cdot)$ and a learnable weight matrix W_θ:

   $$f_\theta(p, s_t) = \exp(\phi_\theta(p)^T \cdot W_\theta \cdot \phi_\theta(s_t))$$

   The objective is to minimize the expectation of the following loss over states s_t (and their corresponding positive s_{t+1} and negative set X) encountered during training:

   $$\mathcal{L}_t(\theta) = \mathbb{E}_{s_t}\left[-\log \frac{f_\theta(s_{t+1}, s_t)}{\sum_{p\in X} f_\theta(p, s_t)}\right]$$

By minimizing this expected loss $\mathcal{L}_t(\theta)$, the policy learns to better discriminate the correct path forward. This approach directly learns a policy by contrasting successful and unsuccessful transitions, sidestepping explicit value function estimation, which is particularly advantageous in domains with sparse rewards and unique solution states.

2. **LEMMA**

In this paper, the authors propose an algorithm named LEMMA, that develops hierarchies of abstractions. Some of the key insights of this paper include:

(a) **Abstractions dramatically reduce search complexity**

By identifying and elevating frequently used sequences of low-level axioms into single "macro-actions," LEMMA shortens solution paths.

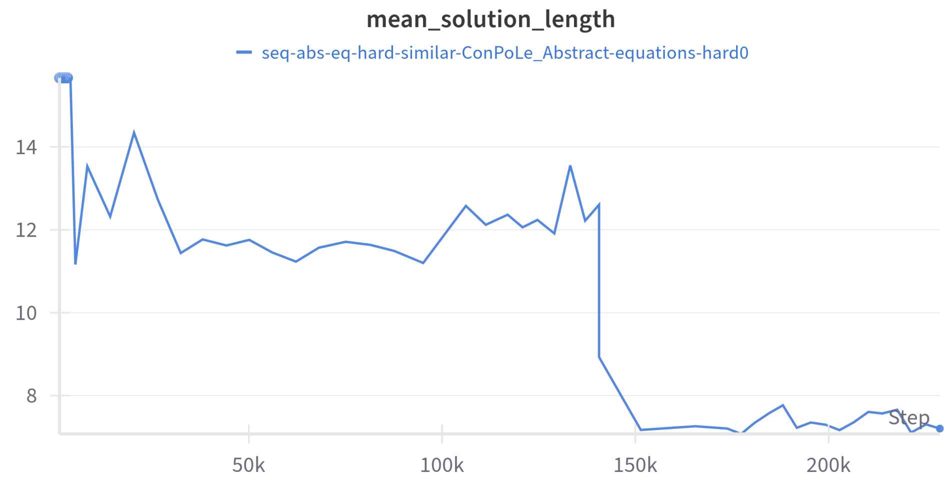

Figure 1: Mean solution length on the 'equations-hard' domain for the modified ConPoLe agent combined with *SeqAbs* abstractions. It drops from about 12 to 7 after incorporating abstractions.

(b) **Integration with Expert Iteration**

LEMMA augments any search-and-learn loop (like ConPoLe) by periodically mining solved trajectories for useful subsequences, adding them to the action set, and then retraining. This includes a brief imitation-learning phase on the abstracted data before the next search round.

(c) **Action Projection Functions**

- *SeqAbs*: An abstraction simply corresponds to a sequence of axioms that are to be applied.
- *RelAbs*: Uses sequences of axioms as well as a relative indexing of where they are applied to.

(d) **Bayesian selection of abstractions**

Candidate macros are scored by how much they increase the likelihood that a random agent would reproduce the observed solutions. A simple greedy maximization of this objective reliably picks the most useful abstractions. The github repository contains implementations of multiple algorithms for discovering abstractions from a data set of ConPoLe solutions.

(e) **Strong empirical gains on mathematical benchmarks**

On both standard and hard versions of the Common Core equation-solving and fraction-simplification tasks, LEMMA-augmented ConPoLe achieved higher success rates. In some cases, jumps from under 50% to 90% were observed. It also showed better zero-shot generalization to longer, more complex problems.

3 Motivation

1. **Similar Negative Sampling for ConPoLe:** The core idea behind contrastive learning is to learn representations that pull similar (positive) items together and push dissimilar (negative) items apart. In the context of policy learning with ConPoLe, a ”positive” is the next state on a correct solution path. ”Negatives” are alternative next states. If negative samples are too easy to distinguish from positives, the model might not learn fine-grained distinctions crucial for complex decision-making. By selecting negatives that are ”semantically similar” to the positive (correct next state) but are ultimately incorrect, we hypothesize that the model can learn a more robust decision boundary. This forces the policy to focus on subtle differences that differentiate a good step from a slightly-off but plausible-looking step. The intuition is that training against harder, more confusable negatives will lead to a more discerning and effective policy.

2. **PPO for Abstraction Finding (Modified LEMMA):** The original LEMMA paper uses heuristic methods, specifically a Bayesian criterion, to discover useful abstractions (sequences of axioms) from solved trajectories. While effective, heuristic approaches might not always find the optimal set of abstractions or might require careful tuning of criteria. Our motivation for using Proximal Policy Optimization (PPO), an RL algorithm, for abstraction discovery is to frame this process as a learning problem itself. By defining an MDP where states represent the current set of discovered abstractions and actions correspond to creating new abstractions, an RL agent can potentially learn a policy to select beneficial abstractions. The reward function would be designed to reflect the utility of an abstraction (e.g., its frequency in solutions, its ability to simplify problems, its composition from simple axioms). The goal is to see if an RL agent can learn to discover abstractions that are as good as or better than those found by heuristics, potentially leading to a more adaptive and automated way of building up higher-level reasoning capabilities. PPO is chosen for its stability and good performance across various RL tasks.

4 Experiments

1. **ConPoLe with Similar Negative Sampling**

   Our modification to ConPoLe involves enhancing the selection of negative samples for the contrastive loss. The original ConPoLe uses actions from the current beam search (alternatives at the current step or from failed branches) as negatives.

   Experimental Setup: We introduced a global state replay buffer that stores string representations of all encountered states from past trajectories. When a positive action (leading to s_{t+1} from s_t) is identified:

   (a) A set of candidate state strings is sampled from this global buffer.

   (b) These state strings are converted into ‘State‘ objects.

   (c) Embeddings for these candidate states and a reference state (either s_t or s_{t+1}, configurable) are obtained using the ConPoLe agent’s Q-function’s encoder.

   (d) Cosine similarity is computed between the reference state embedding and the candidate state embeddings.

   (e) The top-k most similar candidate states (that are not the positive next state) are selected to form ”similar negative” actions. These actions originate from s_t but lead to these chosen similar (but incorrect) next states.

   (f) These ”similar negatives” are added to the existing negative samples generated by ConPoLe’s beam search mechanism (actions from other branches of the beam).

   The InfoNCE loss is then computed with this augmented set of negatives. The hypothesis is that by contrasting the correct next state with more challenging, similar-looking incorrect states, the learned policy will be more discriminative and effective.

2. **LEMMA with PPO-based Abstraction Discovery**

   The LEMMA paper finds abstractions using heuristics. Our method of finding abstractions borrows multiple concepts used by LEMMA to create a reinforcement learning algorithm for the same task. Specifically, we model the entire process of finding the *SeqAbs* abstractions as a *Markov Decision Process (MDP)*. The *MDP* formulation is as follows:

   - **States**

     Each state is a binary vector of size **362**. Due to limitations in compute power, we restricted the algorithm to find abstractions of length 2 only. This means combining two basic axioms to create an abstraction. Additionally, two axioms a and b could either form abstraction $a \sim b$ or $b \sim a$. Additionally, one extra bit was added which served as a termination bit. There are a total of **19** basic axioms for the 'equations' domain. Thus, the size of vector is calculated as:

     $$^{19}P_1 \text{ (simple axioms)} + {}^{19}P_2 \text{ (learnable abstractions)} + \\ 1 \text{ (terminate bit)} = 19 + (19 \times 18) + 1 = 19 + 342 + 1 = 362$$

     The state denotes what all abstractions are currently discovered, including the basic axioms. A "1" in the vector denotes that the corresponding abstraction has been discovered.

   - **Actions**

     In each action, the goal is to discover a new abstraction. For this, we must toggle a 0 bit in the state vector to 1. So, the action space is discrete. Each action corresponds to finding a new abstraction. If the bit is already 1, then the action is seen as invalid.

   - **Transition Function**

     The transition function is deterministic. If the action changes a bit from 0 to 1, then it is a valid action, otherwise the action is invalid. The transition denotes a new abstraction being discovered.

   - **Reward Function**

     For each abstraction, we create reward function. This reward function signifies how could the discovered abstraction is. There are a total of 4 parameters to judge an abstraction:

     (a) `Size Penalty`: Abstractions containing too many basic axioms are penalized. Currently, this component is not of much use to us since we have restricted ourselves to abstractions containing only 2 axioms.

     (b) `Frequency Bonus`: If the abstraction is used in many solutions, then it signifies that the abstraction is worth discovering. On the other hand, abstractions that are not frequently used are discouraged from being discovered.

     (c) `Simplicity`: Simple axioms like *add* and *add0* can be effectively merged to form abstractions. However, merging comparatively complex abstractions like *dist* and *subsub* is not very impactful. We rate each basic axiom on a scale of 1 to 10 to denote their simplicity. Simple axioms like *refl* have a simplicity of 10. The simplicity of an abstraction is the sum of simplicities of its component axioms.

     (d) `Similarity`: If we only look at the simplicity, then complex axioms like *sub_self* and *eval* might get ignored. To overcome this, we also look at the similarity of the axioms being merged. Two similar axioms can form a good abstraction if merged.

     We first assign each axiom a conceptual category like `identity`, `inverse`, etc. Then, we assign the pairwise similarity to these categories on a scale of 1 to 10. Similar categories like (`identity`, `definition`) are assigned high similarity.

     The similarity between two axioms is taken as the similarity of their respective conceptual categories.

     All of these components are appropriately scaled using hyperparameters. The final reward function is:

     $$\texttt{size\_penalty} + \texttt{freq\_bonus} + \max(\texttt{simplicity},\ \texttt{similarity})$$

- **Discount Factor**
  The discount factor is taken to be 1.
- **Policy**
  The goal is to find an optimal policy which can find appropriate abstractions, given a set of solutions from an agent, like *ConPoLe*.

Using this *MDP* formulation, we have used the *Proximal Policy Optimization (PPO)* algorithm to find the best policy. The final state denotes the abstractions which have been learnt.

5 Results

The experiments were conducted on an RTX 3070 laptop GPU, with runs taking 1-3 days for 10^7 steps based on different configurations.

Our experiments focused on the challenging ‘equations-hard‘ domain. Detailed logs and interactive plots for our experiments are available on our Weights & Biases project page: wandb.

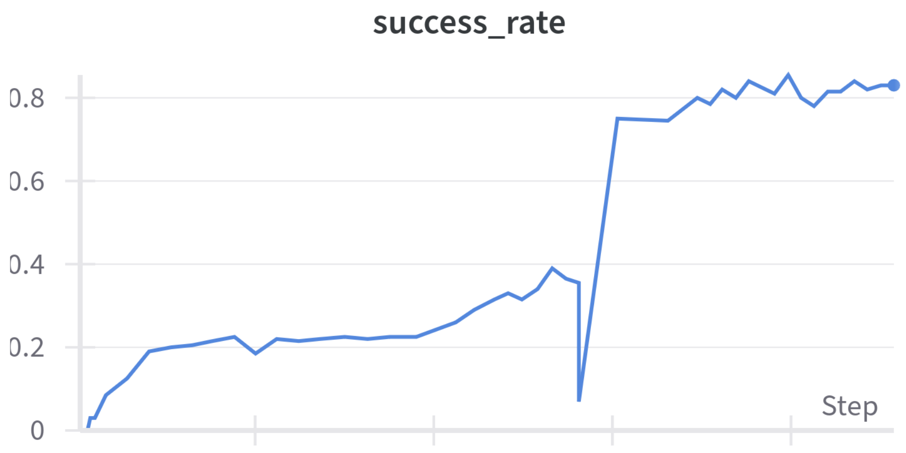

Figure 2: Success rate progression on the ‘equations-hard‘ domain for the modified ConPoLe agent combined with *SeqAbs* abstractions. A notable characteristic observed is a temporary decrease in success rate immediately following the discovery and integration of new abstractions. This dip is attributed to the agent adapting to an expanded action space, after which performance typically recovers and surpasses previous levels.

Modified ConPoLe with Similar Negative Sampling:

- When combined with *SeqAbs* abstractions (from the original LEMMA paper's heuristic method), our modified ConPoLe achieved an accuracy of 85.5% on the ‘equations-hard‘ dataset in approximately $4.5 * 10^6$ environment steps. This is a significant improvement over the vanilla ConPoLe + *SeqAbs* which, based on the trends in the LEMMA paper (Figure 2), was below 60% accuracy at $4.5 * 10^6$ steps and required around $7.5 * 10^6$ steps to reach comparable performance (around 80-85% for ConPoLe+SeqAbs).
- Without any LEMMA abstractions, our modified ConPoLe (with similar negative sampling only) achieved an accuracy of 65.5% in $8 * 10^6$ environment steps. In contrast, the vanilla ConPoLe (without abstractions) reached about 50% accuracy in 10^7 steps on the ‘equations-hard‘ domain as per the LEMMA paper's Figure 2. [1]

[1] The performance metrics for this specific configuration could not be plotted due to some Weights & Biases logging errors during the run. The reported accuracy is the best observed success rate on the test set during the training run. It can be accessed via the logs for this run (line 3226).

These results suggest that incorporating more challenging, similar negatives helps ConPoLe learn a more effective policy, leading to faster convergence and higher final performance, both with and without learned abstractions.

Modified LEMMA (PPO-based Abstraction Discovery): Our PPO-based approach for discovering *SeqAbs* abstractions (limited to length 2) in the 'equations-hard' domain yielded limited success. With the designed reward function, the PPO agent consistently converged to discovering only one or very few dominant abstractions. Consequently, this did not lead to a noticeable increase in the performance of the base ConPoLe agent when these few learned abstractions were added. The performance was almost identical to vanilla ConPoLe. This suggests that the reward function, despite its components (size penalty, frequency bonus, simplicity, similarity), requires further tuning or a more sophisticated structure to guide the PPO agent towards a diverse and useful set of abstractions. The current reward function might be overly sensitive to initial discoveries or might not adequately balance exploration of new abstractions with exploitation of known good ones.

6 Conclusion

Our investigation yielded mixed but insightful results. The introduction of similar negative sampling into the ConPoLe framework demonstrated a tangible improvement in performance on the challenging equations-hard domain. By forcing the model to distinguish the correct next step from structurally or semantically similar (yet incorrect) alternatives, based on state embeddings, we observed accelerated learning, faster convergence, and higher overall accuracy. This benefit was apparent both when the modified ConPoLe was used standalone and when augmented with heuristic abstraction mechanisms like *SeqAbs*. This underscores the critical role of curated, challenging negative samples in enhancing the discriminative power of contrastive learning for policy optimization in symbolic reasoning.

In contrast, our attempt to replace LEMMA's heuristic abstraction discovery with a PPO-based Reinforcement Learning agent did not achieve the anticipated performance gains. Our PPO approach, consistently converged to identifying only one or very few dominant abstractions. This limitation suggests that while the MDP formulation is a principled step, the designed reward function—despite incorporating intuitive components like frequency and simplicity—was insufficient to guide the PPO agent towards a diverse and broadly useful set of abstractions. The PPO agent may have settled into local optima, identifying high-frequency 2-step sequences without adequate incentive for further exploration in the discrete, high-dimensional action space. The current reward structure likely focuses too much on individual abstraction quality rather than the value of a comprehensive set.

Therefore, while enhancing the core contrastive learning mechanism of ConPoLe through sophisticated negative sampling shows clear promise, the autonomous learning of symbolic abstractions via standard RL methods like PPO necessitates a more refined approach to reward engineering and exploration strategies. Future work for PPO-based abstraction discovery should focus on developing more nuanced reward functions that can better evaluate sets of abstractions and encourage broader exploration, potentially alongside more scalable state and action representations to accommodate longer and more complex abstractions. For ConPoLe, further exploration into different domains (for instance non-linear domains where proofs may have a tree-like structure) is required.

7 References

References

[1] Gabriel Poesia, WenXin Dong, and Noah Goodman.
Contrastive Reinforcement Learning of Symbolic Reasoning Domains.
In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), *Advances in Neural Information Processing Systems*, volume 34, pages 15946–15956.

[2] Zhening Li, Gabriel Poesia, Omar Costilla-Reyes, Noah Goodman, and Armando Solar-Lezama. *LEMMA: Bootstrapping High-Level Mathematical Reasoning with Learned Symbolic Abstractions*. NeurIPS 2022 Workshop on Math-AI.