2025/02/20 회의 (RL 관련 공유)

Wise

AI Lab

2025-02-20

Contents

- Preliminaries
  - Random variables
  - Monte Carlo estimation
- Overview of RL
  - Terminologies
  - Value-based / Policy-based / Model-based RL
- Policy Optimization
  - Policy Gradient Theorem
  - REINFORCE
  - PPO
- Reinforcement Learning with LLMs
- Reinforcement Learning with Verifiable Rewards
- How to apply RL to Colorization(밑색/채색)

Preliminaries

Random variables

$X : \Omega \to E$

(Ω, Σ, P): 확률 공간 (Sample space Ω, Event space Σ, Probability measure P)

(with some assumptions), we also have

Probability density function

$\mathbb{E}_P[f(X)] := \int_\Omega f(X(\omega))dP(\omega) = \int_E f(x)p(x)dx =: \mathbb{E}_{x\sim p(x)}[f(x)]$

> Note
>
> E might be complex..

Monte Carlo estimation

Goal: Calculate $\mathbb{E}_{x\sim p(x)}[f(x)]$

(Hint: Law of Large Numbers)

Terminologies

- Agent
- Environment
- $\mathcal{S}$: a finite set of states (상태 집합)
- $\mathcal{A}$: a finite set of actions (행동 집합)
- Policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$
    - Optimal policy
        - $\pi^* = \arg\max_\pi \mathbb{E}_{s_0 \sim p_0(s)}[V_\pi(s_0)]$
        - $V_{\pi^*}(s) \geq V_\pi(s)\ (\forall x \in \mathcal{S}, \forall \pi)$
- Reward $R : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$
- Value function $V : \mathcal{S} \to \mathbb{R}$
- Q-function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

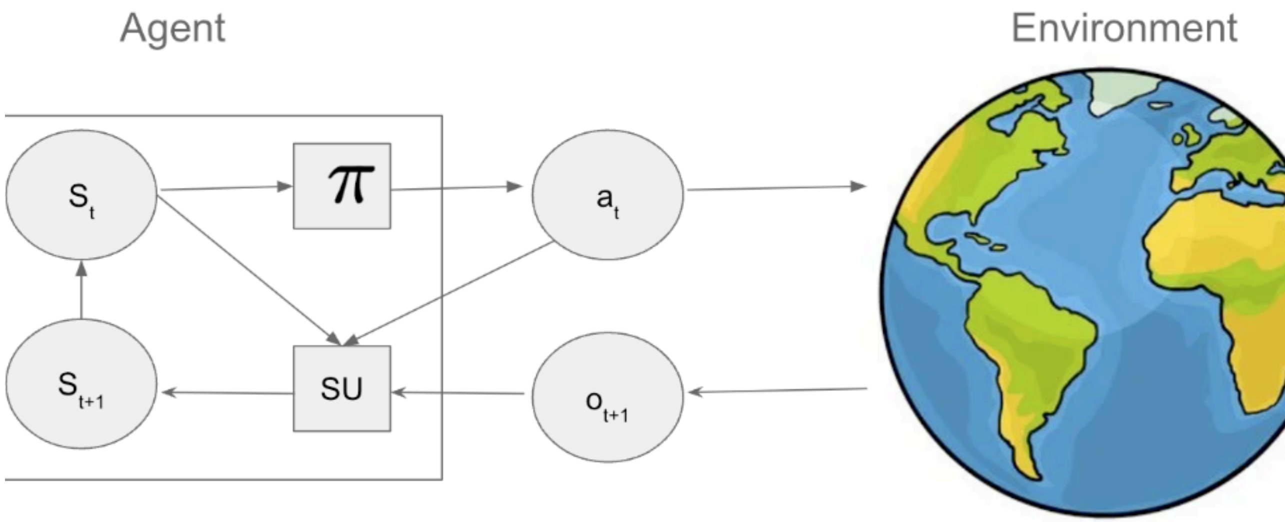


Figure 1.1: A small agent interacting with a big external world.

Terminologies

- Advantage function $A : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
  - $A(s, a) := Q(s, a) - V(s)$
- Generalized advantage estimation (GAE)
  - Advantage function을 계산하려면 state와 action 값이 필요하다.
  - 그런데 이러한 state, action은 (policy와 initial state의 확률분포에 depend하는) random variable이다.
  - 따라서 Advantage function의 evaluation 결과 $A(s, a)$도 random variable
  - 이 random variable을 estimate하기 위해 R, V를 통해 Monte Carlo estimate을 하는데 그 estimation의 variance를 줄이기 위해 나온 방법이 GAE

GAE

3.3.2 Generalized advantage estimation (GAE)

In A2C, we replaced the high variance, but unbiased, MC return G_t with the low variance, but biased, one-step bootstrap return $G_{t:t+1} = r_t + \gamma V_{\boldsymbol{w}}(s_{t+1})$. More generally, we can compute the n-step estimate

$$G_{t:t+n} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V_{\boldsymbol{w}}(s_{t+n}) \tag{3.25}$$

and thus obtain the (truncated) n-step advantage estimate as follows:

$$A_{\boldsymbol{w}}^{(n)}(s_t, a_t) = G_{t:t+n} - V_{\boldsymbol{w}}(s_t) \tag{3.26}$$

Unrolling to infinity, we get

$$A_t^{(1)} = r_t + \gamma v_{t+1} - v_t \tag{3.27}$$

$$A_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 v_{t+2} - v_t \tag{3.28}$$

$$\vdots \tag{3.29}$$

$$A_t^{(\infty)} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots - v_t \tag{3.30}$$

$A_t^{(1)}$ is high bias but low variance, and $A_t^{(\infty)}$ is unbiased but high variance.

Instead of using a single value of n, we can take a weighted average. That is, we define

$$A_t = \frac{\sum_{n=1}^{T} w_n A_t^{(n)}}{\sum_{n=1}^{T} w_n} \tag{3.31}$$

If we set $w_n = \lambda^{n-1}$ we get the following simple recursive calculation:

$$\delta_t = r_t + \gamma v_{t+1} - v_t \tag{3.32}$$

$$A_t = \delta_t + \gamma\lambda\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} = \delta_t + \gamma\lambda A_{t+1} \tag{3.33}$$

GAE

Algorithm 5: Generalized Advantage Estimation

def GAE($r_{1:T}, v_{1:T}, \gamma, \lambda$)

$A' = 0$

for $t = T : 1$ **do**

- $\delta_t = r_t + \gamma v_{t+1} - v_t$
- $A' = \delta_t + \gamma \lambda A'$
- $A_t = A'$ // advantage
- $q_t = A_t + v_t$ // TD target

Return $(A_{1:T}), q_{1:T})$

Objective function of RL

$$L(\theta) = \mathbb{E}_{s_0 \sim p_0(s)}[V_{\pi_\theta}(s_0)]$$

- policy를 neural network의 parameter θ를 도입하여 위의 목적함수를 최대화하도록 훈련해서 optimal policy를 얻고자 하는 것이 RL의 목표

Overview of RL

- Value-based RL
  - Q functiond을 학습하여 $\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$를 policy로 사용
  - 단점
    - function approximation (such as neural networks)
    - bootstrapped value function estimation (TD-like method)
    - off-policy learning
    - This combination : `the deadly triad`
    - RL 알고리즘 불안정함
- Policy-based RL
  - Policy search method
  - 대부분 policy gradient 방법론 사용
- Model-based RL (MBRL)

Classification of RL

Approach	Method	Functions learned	On/Off	Section
Value-based	SARSA	$Q(s, a)$	On	Section 2.4
Value-based	Q-learning	$Q(s, a)$	Off	Section 2.5
Policy-based	REINFORCE	$\pi(a\|s)$	On	Section 3.2
Policy-based	A2C	$\pi(a\|s)$, $V(s)$	On	Section 3.3.1
Policy-based	TRPO/PPO	$\pi(a\|s)$, $A(s, a)$	On	Section 3.4.3
Policy-based	DDPG	$a = \pi(s)$, $Q(s, a)$	Off	Section 3.6.1
Policy-based	Soft actor-critic	$\pi(a\|s)$, $Q(s, a)$	Off	Section 3.5.4
Model-based	MBRL	$p(s'\|s, a)$	Off	Chapter 4

Table 1.1: Summary of some popular methods for RL. On/off refers to on-policy vs off-policy methods.

Policy Gradient Theorem

- **Policy Gradient Theorem (Sutton, etl al., NIPS 1999)**

Let us define $L(\theta) = \mathbb{E}_{s_0 \sim p_0(s)}[V_{\pi_\theta}(s_0)]$. Then,

$$\nabla_\theta L(\theta) = \mathbb{E}_{s \sim \rho^\gamma_{\pi_\theta}(s), a \sim \pi_\theta(a|s)}[Q_{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a|s)],$$

where ρ^γ_π is the discounted state visitation measure defined by

$$\rho^\gamma_\pi(s) := \sum_{t=0}^{\infty} \gamma^t \sum_{s_0} p_0(s_0) p_\pi(s_0 \to s, t).$$

REINFORCE

- Monte Carlo version

Algorithm 3: REINFORCE (episodic version)

Initialize policy parameters $\boldsymbol{\theta}$
repeat
  Sample an episode $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, \ldots, s_T)$ using $\pi_{\boldsymbol{\theta}}$
  for $t = 0, 1, \ldots, T-1$ **do**
    $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$
until *converged*

- 단점: estimation G_t의 분산이 큼
  - 해결책: baseline을 사용

Policy Optimization

- PPO

- InstructGPT

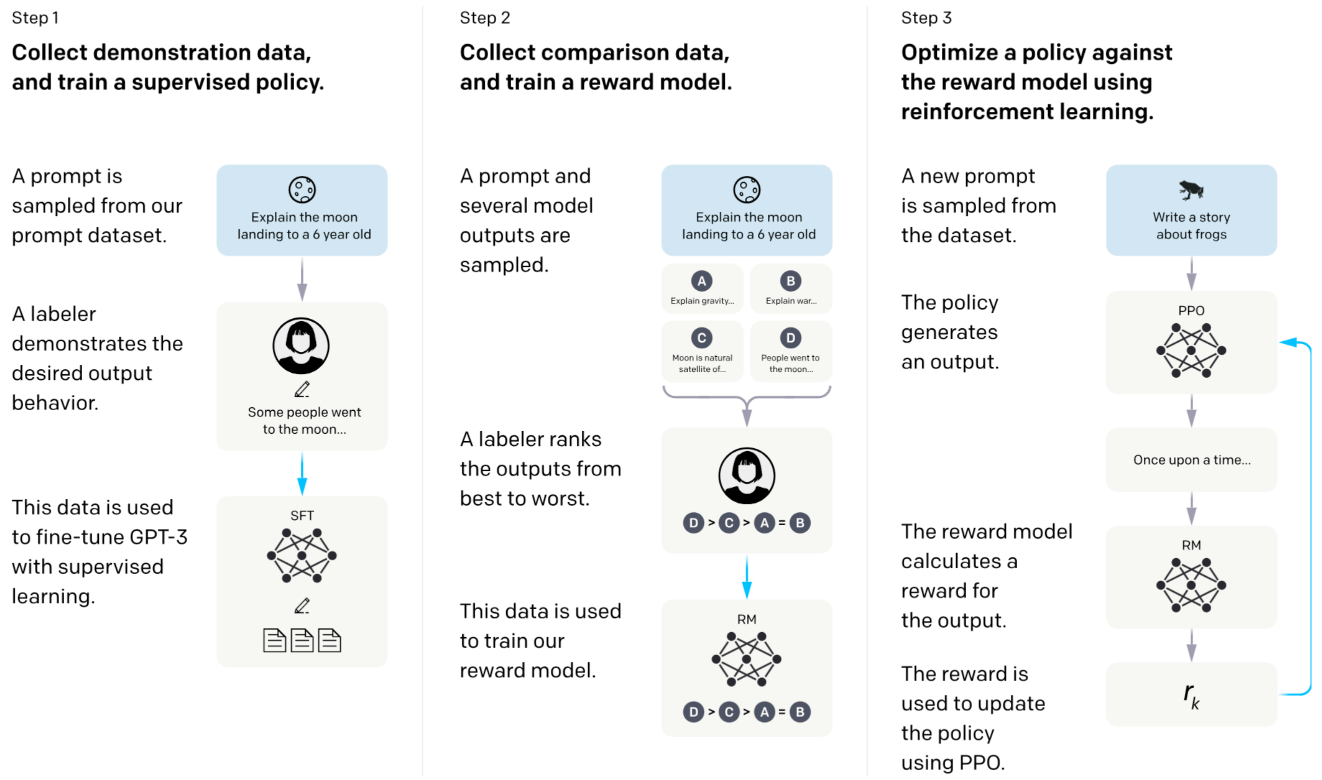
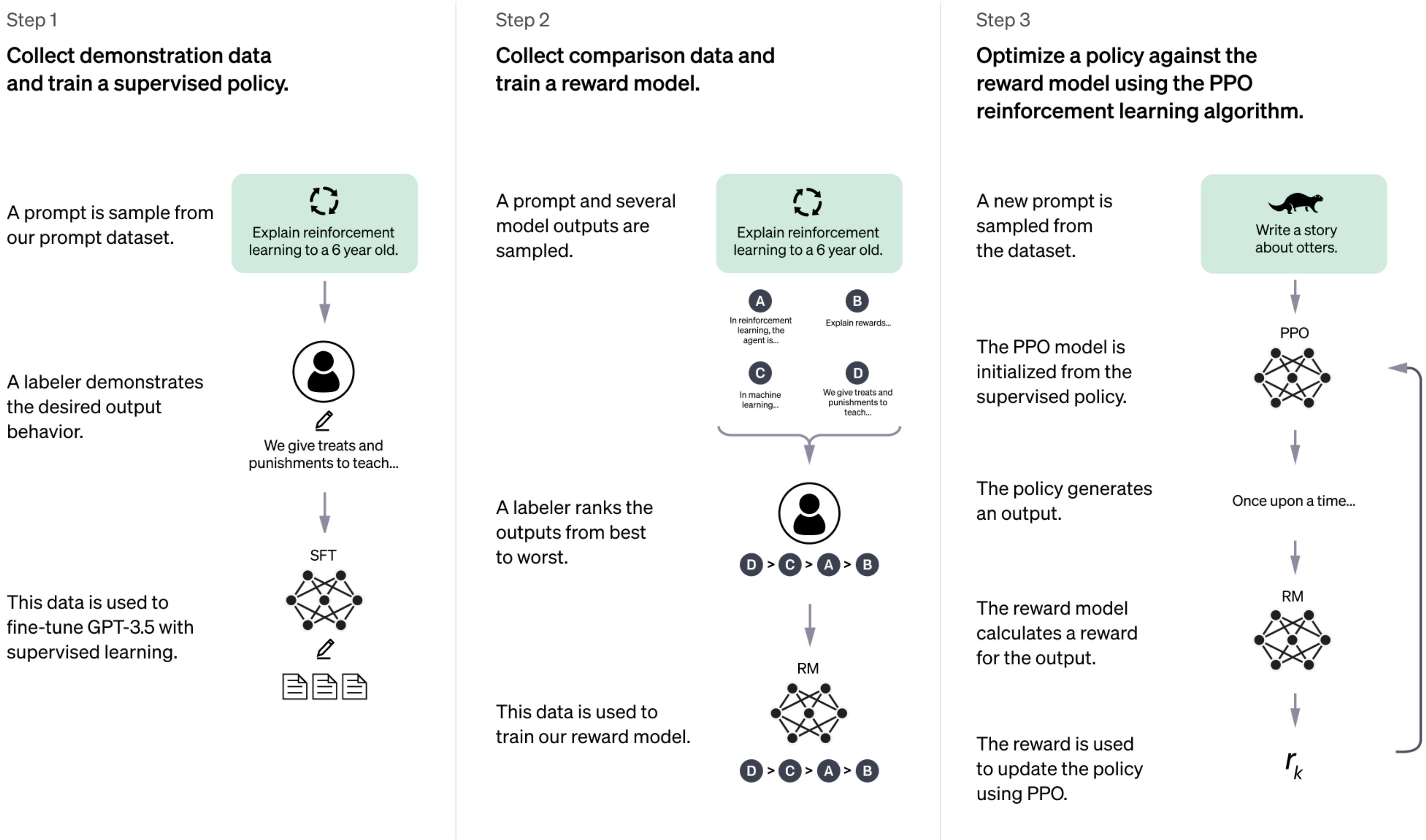


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

RL with LLMs

- ChatGPT



RL with LLMs

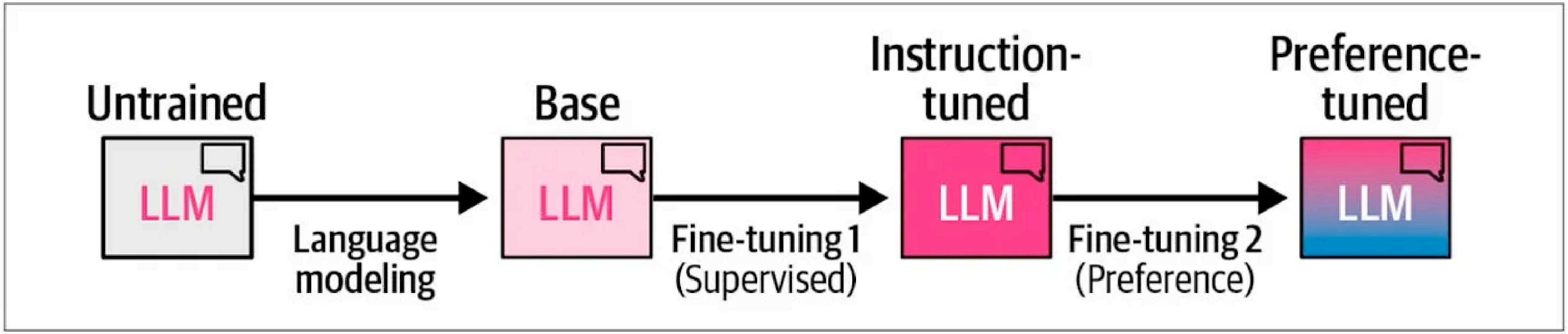


Figure 12-3. The three steps of creating a high-quality LLM.

RL with LLMs

- DeepSeek-R1-Zero
  - RL on the Base Model

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

Table 2 | Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

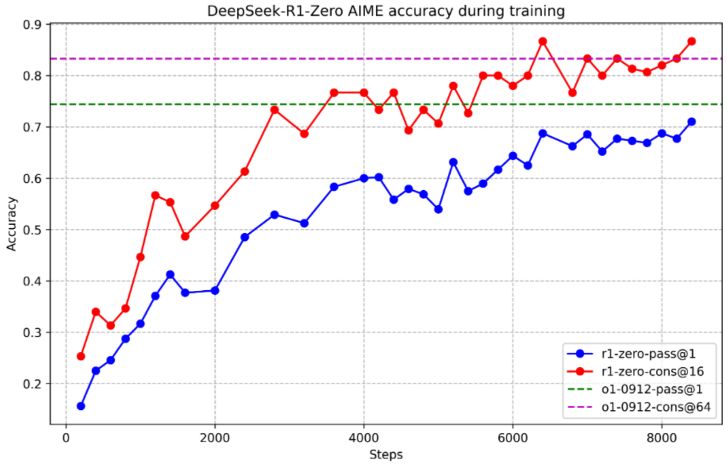


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

RLVR

2.2.2. *Reward Modeling*

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between ‘<think>’ and ‘</think>’ tags.

We do not apply the outcome or process neural reward model in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

RLVR

- Tulu 3

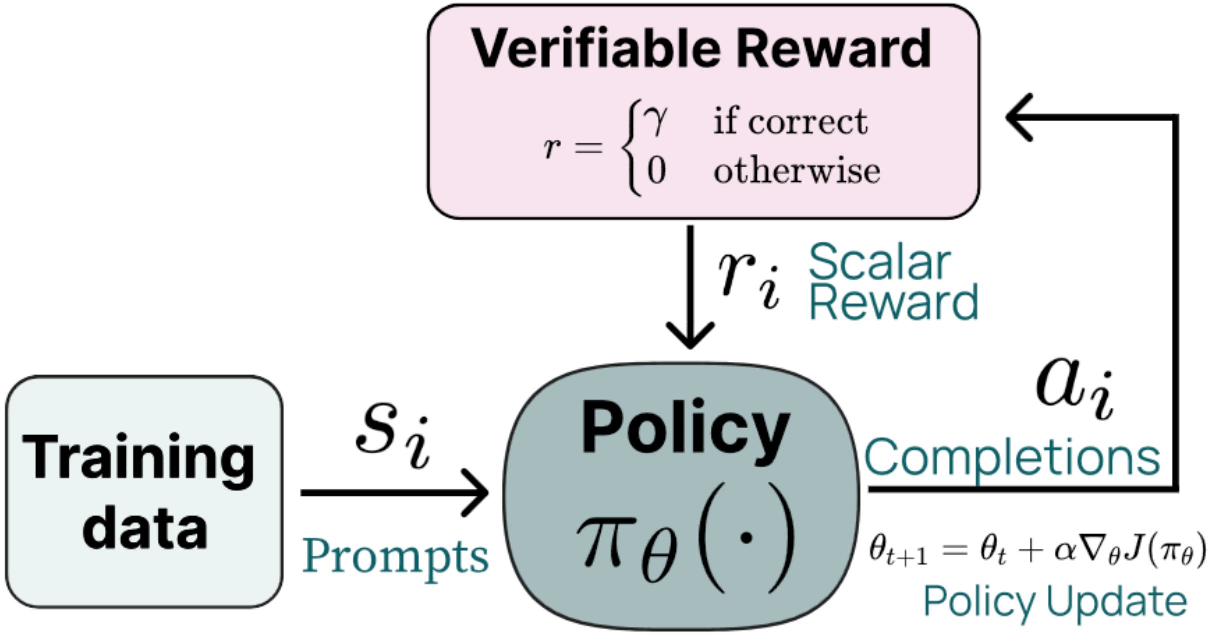


Figure 18 An overview of how Reinforcement Learning with Verifiable Rewards (RLVR) works. We sample completions from a policy model given a set of prompts, and verify their correctness using a deterministic function. If the answer is verifiably correct, we provide reward of α, otherwise 0. We then train against this reward using PPO.