

Deep RL



Reinforcement Learning problem formulation

Suppose we have

- A finite state space S ;
- a finite set of actions A ;
- a probability $\mathbf{P}_a(s, s')$ on $S \times S$ for each $a \in A$;
- a function $r: S \rightarrow \mathbb{R}$ called *reward*.

This tuple is called a **Markov decision process (MDP)**.

The model above fits the reinforcement learning setting perfectly in the following sense. We have **an agent** that at each time $t \in \{0, 1, 2, 3, \dots\}$

- 1) being in some state s_t
- 2) takes an action a_t according to **policy** $\pi(\cdot | s_t)$,
- 3) moves to state s_{t+1} according to $\mathbf{P}_{a_t}(s_t, \cdot)$,
- 4) eventually gets reward $r(s_{t+1})$.

Note that policy $\pi(\cdot | s_t)$ defines a probability distribution on a space S^∞ of sequences of the form $(s_1, s_2, s_3 \dots)$.

Reinforcement Learning problem formulation

Suppose we have

- A finite state space S ;
- a finite set of actions A ;
- a probability $\mathbf{P}_a(s, s')$ on $S \times S$ for each $a \in A$;
- a function $r: S \rightarrow \mathbb{R}$ called *reward*.

This tuple is called a **Markov decision process (MDP)**.

The model above fits the reinforcement learning setting perfectly in the following sense. We have **an agent** that at each time $t \in \{0, 1, 2, 3, \dots\}$

- 1) being in some state s_t
- 2) takes an action a_t according to **policy** $\pi(\cdot | s_t)$,
- 3) moves to state s_{t+1} according to $\mathbf{P}_{a_t}(s_t, \cdot)$,
- 4) eventually gets reward $r(s_{t+1})$.

Note that policy $\pi(\cdot | s_t)$ defines a probability distribution on a space S^∞ of sequences of the form $(s_1, s_2, s_3 \dots)$.

Let's give some definitions.

Define **discounted cumulative reward** as

$$\mathcal{R}_\pi(t) = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}).$$

Define **state-action value function** as

$$Q_\pi(s_t, a_t) = \mathbf{E}_{s_{t+1}, a_{t+1}, \dots} \mathcal{R}_\pi(t).$$

Define **value function** as

$$V_\pi(s_t) = \mathbf{E}_{a_t, s_{t+1}, a_{t+1}, \dots} \mathcal{R}_\pi(t).$$

Define **advantage function** as

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s).$$

Reinforcement Learning problem is about finding a policy $\pi^*(\cdot | s)$ that solves

$$J(\pi) = \mathbf{E} \mathcal{R}_\pi(0) \rightarrow \sup_{\pi}.$$

One can show that

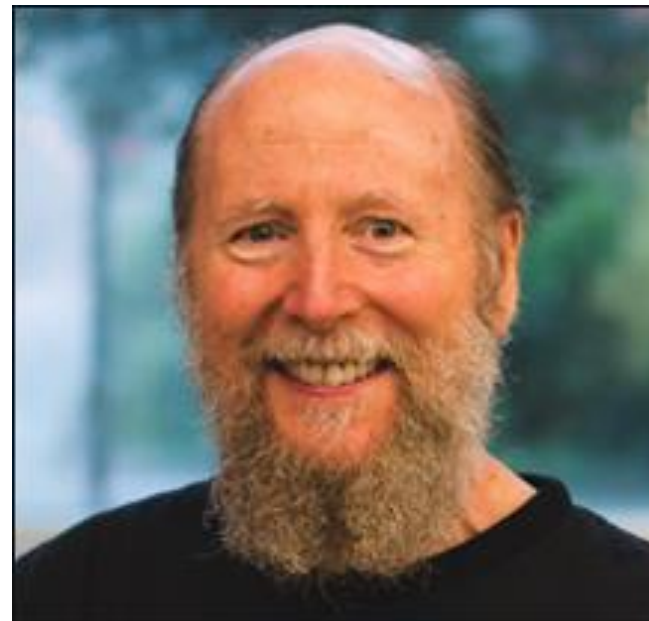
$$\frac{d}{d\theta} J(\pi_\theta) = \mathbf{E} \left[\mathcal{R}_{\pi_\theta}(0) \frac{d}{d\theta} \log \pi_\theta(a_t | s_t) \right],$$

which leads us to the simplest algorithm to update π_θ – REINFORCE.

Glorious fathers



Andrew Barto



Richard S. Sutton

- Developed a formalism and theory for an agent acting in a stochastic environment based on MDP
- A. M. Turing Award winners in 2024 for “developing the conceptual and algorithmic foundations of reinforcement learning”
- “Reinforcement Learning: An Introduction”

[1] Trust Region Policy Optimization

J. Schulman, S. Levine, P. Moritz, M. Jordan, P. Abbeel

University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Feb 2015

Trust Region Policy Optimization

Suppose we have two policies π and $\tilde{\pi}$. They are connected via

$$J(\tilde{\pi}) = J(\pi) + \mathbf{E}_{s_0, a_0, \dots \sim \tilde{\pi}} \left[\sum_{k=0}^{\infty} \gamma^k A_{\pi}(s_t, a_t) \right],$$

which may be rewritten as

$$J(\tilde{\pi}) = J(\pi) + \sum_s p_{\tilde{\pi}}(s) \sum_a A_{\pi}(s, a) \tilde{\pi}(a|s), \quad (1)$$

where

$$p_{\tilde{\pi}}(s) = \sum_{k=0}^{\infty} \gamma^k P_{\tilde{\pi}}(s_k = s).$$

The formula (1) may be used for policy optimization, but $p_{\tilde{\pi}}(s)$ is hard to estimate. Let's use an approximation

$$L_{\pi}(\tilde{\pi}) = J(\pi) + \sum_s p_{\pi}(s) \sum_a A_{\pi}(s, a) \tilde{\pi}(a|s).$$

Consider $\{\pi_{\theta}\}_{\theta \in \Theta}$. One can see that

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = J(\pi_{\theta_0}), \quad \left. \frac{d}{d\theta} L_{\pi_{\theta_0}}(\pi_{\theta}) \right|_{\theta=\theta_0} = \left. \frac{d}{d\theta} J(\pi_{\theta}) \right|_{\theta=\theta_0}.$$

So a sufficiently small step from π_{θ_0} to π_{θ_1} that improves $L_{\pi_{\theta_0}}(\pi_{\theta})$ improves $J(\pi_{\theta_0})$ as well.

Trust Region Policy Optimization

Suppose we have two policies π and $\tilde{\pi}$. They are connected via

$$J(\tilde{\pi}) = J(\pi) + \mathbf{E}_{s_0, a_0, \dots \sim \tilde{\pi}} \left[\sum_{k=0}^{\infty} \gamma^k A_{\pi}(s_t, a_t) \right],$$

which may be rewritten as

$$J(\tilde{\pi}) = J(\pi) + \sum_s p_{\tilde{\pi}}(s) \sum_a A_{\pi}(s, a) \tilde{\pi}(a|s), \quad (1)$$

where

$$p_{\tilde{\pi}}(s) = \sum_{k=0}^{\infty} \gamma^k P_{\tilde{\pi}}(s_k = s).$$

The formula (1) may be used for policy optimization, but $p_{\tilde{\pi}}(s)$ is hard to estimate. Let's use an approximation

$$L_{\pi}(\tilde{\pi}) = J(\pi) + \sum_s p_{\pi}(s) \sum_a A_{\pi}(s, a) \tilde{\pi}(a|s).$$

Consider $\{\pi_{\theta}\}_{\theta \in \Theta}$. One can see that

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = J(\pi_{\theta_0}), \quad \left. \frac{d}{d\theta} L_{\pi_{\theta_0}}(\pi_{\theta}) \right|_{\theta=\theta_0} = \left. \frac{d}{d\theta} J(\pi_{\theta}) \right|_{\theta=\theta_0}.$$

So a sufficiently small step from π_{θ_0} to π_{θ_1} that improves $L_{\pi_{\theta_0}}(\pi_{\theta})$ improves $J(\pi_{\theta_0})$ as well.

But how small this step should be?

Theorem

Let

$$D_{\text{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{\text{KL}}(\pi(\cdot | s), \tilde{\pi}(\cdot | s)).$$

Then

$$J(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - C D_{\text{KL}}^{\max}(\pi, \tilde{\pi}),$$

where

$$C = \frac{4\gamma}{(1-\gamma)^2} \max_{s,a} |A_{\pi}(s, a)|.$$

Algorithm 1 Policy iteration algorithm guaranteeing non-decreasing expected return η

Initialize π_0 .

for $i = 0, 1, 2, \dots$ until convergence **do**

 Compute all advantage values $A_{\pi_i}(s, a)$.

 Solve the constrained optimization problem

$$\pi_{i+1} = \arg \max_{\pi} [L_{\pi_i}(\pi) - C D_{\text{KL}}^{\max}(\pi_i, \pi)]$$

 where $C = 4\epsilon\gamma/(1-\gamma)^2$

$$\text{and } L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$$

end for

Trust Region Policy Optimization (cont.)

In practice, step sizes in the Algorithm 1 are very small because of the large penalizing coefficient C . In order to surpass it, reformulate the problem as

$$L_{\theta_0}(\theta) \rightarrow \sup_{\theta}, \quad D_{\text{KL}}^{\text{mean}}(\pi, \tilde{\pi}) \leq \delta.$$

We use $D_{\text{KL}}^{\text{mean}}$ instead of $D_{\text{KL}}^{\text{max}}$ because the latter entails too many constraints.

In terms of optimization

$$\begin{aligned} L_{\theta_0}(\theta) &\sim \sum_s p_{\theta_0}(s) \sum_a A_{\theta_0}(s, a) \pi_{\theta}(a|s) \\ &= \sum_s p_{\theta_0}(s) \sum_a A_{\theta_0}(s, a) \frac{\pi_{\theta}(a|s)}{\pi_{\theta_0}(a|s)} \pi_{\theta_0}(a|s) \\ &= \mathbf{E}_{s, a \sim \pi_{\theta_0}} \left[A_{\theta_0}(s, a) \frac{\pi_{\theta}(a|s)}{\pi_{\theta_0}(a|s)} \right]. \end{aligned}$$

So eventually we get

$$\begin{aligned} \mathbf{E}_{s, a \sim \pi_{\theta_0}} \left[A_{\theta_0}(s, a) \frac{\pi_{\theta}(a|s)}{\pi_{\theta_0}(a|s)} \right] &\rightarrow \sup_{\theta}, \\ \mathbf{E}_{s \sim \pi_{\theta_0}} D_{\text{KL}} \left(\pi_{\theta_0}(\cdot | s), \pi_{\theta}(\cdot | s) \right) &\leq \delta. \end{aligned}$$

Trust Region Policy Optimization (cont.)

In practice, step sizes in the Algorithm 1 are very small because of the large penalizing coefficient C . In order to surpass it, reformulate the problem as

$$L_{\theta_0}(\theta) \rightarrow \sup_{\theta}, \quad D_{\text{KL}}^{\text{mean}}(\pi, \tilde{\pi}) \leq \delta.$$

We use $D_{\text{KL}}^{\text{mean}}$ instead of $D_{\text{KL}}^{\text{max}}$ because the latter entails too many constraints.

In terms of optimization

$$\begin{aligned} L_{\theta_0}(\theta) &\sim \sum_s p_{\theta_0}(s) \sum_a A_{\theta_0}(s, a) \pi_{\theta}(a|s) \\ &= \sum_s p_{\theta_0}(s) \sum_a A_{\theta_0}(s, a) \frac{\pi_{\theta}(a|s)}{\pi_{\theta_0}(a|s)} \pi_{\theta_0}(a|s) \\ &= \mathbf{E}_{s,a \sim \pi_{\theta_0}} \left[A_{\theta_0}(s, a) \frac{\pi_{\theta}(a|s)}{\pi_{\theta_0}(a|s)} \right]. \end{aligned}$$

So eventually we get

$$\begin{aligned} \mathbf{E}_{s,a \sim \pi_{\theta_0}} \left[A_{\theta_0}(s, a) \frac{\pi_{\theta}(a|s)}{\pi_{\theta_0}(a|s)} \right] &\rightarrow \sup_{\theta}, \\ \mathbf{E}_{s \sim \pi_{\theta_0}} D_{\text{KL}} \left(\pi_{\theta_0}(\cdot | s), \pi_{\theta}(\cdot | s) \right) &\leq \delta. \end{aligned}$$

- Note that to solve the problem we need to estimate $A_{\theta_0}(s, a)$.
- One may replace $A_{\theta_0}(s, a)$ with $Q_{\theta_0}(s, a)$ in the objective.
- To approximately solve the problem, one may linearly approximate the objective and quadratically approximate the constraint, and use the conjugate gradient method.
- To solve the problem, one may use Monte-Carlo method for estimating the expectation.

[2] Proximal Policy Optimization Algorithms

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov
OpenAI

Jul 2017

Proximal Policy Optimization

Recall that in the TRPO we reformulated the initial problem so that the objective penalization moved to constraints. The reason for this was a tricky penalizing coefficient C . But what if we somehow reconsider the objective penalization?

Denote the ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}$ as $r_t(\theta)$. Then the TRPO target is

$$\mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} [r_t(\theta) A_{\theta_0}(s_t, a_t)].$$

Define

$$L_\theta^{\text{CLIP}} = \mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} \min [r_t(\theta) A_{\theta_0}(s_t, a_t), \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon) A_{\theta_0}(s_t, a_t)].$$

Define

$$L_\theta^{\text{KL}} = \mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} \left[r_t(\theta) A_{\theta_0}(s_t, a_t) - \beta D_{\text{KL}}(\pi_{\theta_0}(\cdot | s_t), \pi_\theta(\cdot | s_t)) \right].$$

Also, define

$$L_\theta^{\text{CLIPKL}} = \mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} \left[\min [r_t(\theta) A_{\theta_0}(s_t, a_t), \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon) A_{\theta_0}(s_t, a_t)] - \beta D_{\text{KL}}(\pi_{\theta_0}(\cdot | s_t), \pi_\theta(\cdot | s_t)) \right].$$

Proximal Policy Optimization

Recall that in the TRPO we reformulated the initial problem so that the objective penalization moved to constraints. The reason for this was a tricky penalizing coefficient C . But what if we somehow reconsider the objective penalization?

Denote the ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}$ as $r_t(\theta)$. Then the TRPO target is

$$\mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} [r_t(\theta) A_{\theta_0}(s_t, a_t)].$$

Define

$$L_\theta^{\text{CLIP}} = \mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} \min [r_t(\theta) A_{\theta_0}(s_t, a_t), \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_{\theta_0}(s_t, a_t)].$$

Define

$$L_\theta^{\text{KL}} = \mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} [r_t(\theta) A_{\theta_0}(s_t, a_t) - \beta D_{\text{KL}}(\pi_{\theta_0}(\cdot | s_t), \pi_\theta(\cdot | s_t))].$$

Also, define

$$L_\theta^{\text{CLIPKL}} = \mathbf{E}_{s_t, a_t \sim \pi_{\theta_0}} [\min [r_t(\theta) A_{\theta_0}(s_t, a_t), \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_{\theta_0}(s_t, a_t)] - \beta D_{\text{KL}}(\pi_{\theta_0}(\cdot | s_t), \pi_\theta(\cdot | s_t))].$$

Algorithm 1 PPO, Actor-Critic Style

```

for iteration=1, 2, ... do
  for actor=1, 2, ..., N do
    Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
  end for
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{\text{old}} \leftarrow \theta$ 
end for

```

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	0.82
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1$.	0.71
Fixed KL, $\beta = 3$.	0.72
Fixed KL, $\beta = 10$.	0.69

How do we apply all this to the LLMs?

Reinforcement Learning with Human (AI) Feedback

Suppose we have an already fine-tuned model, which we denote as π_0 , and we want to train it to match human preferences.

1. At first sample pairs of outputs (y_1, y_2) from π_0 given prompts x .
2. Then, for each sampled pair, get feedback from human (AI) about which answer is more preferable and form a dataset $\mathcal{D} = \{(x, y_w, y_l)\}$.

3. Train a model r that takes inputs x, y and outputs reward. One can do it based on the Bradley-Terry model so that

$$\mathbf{P}\{y_1 \text{ is more preferable}\} = \frac{\exp(r(x, y_1))}{\exp(r(x, y_1)) + \exp(r(x, y_2))}.$$

Then the loss is

$$- \sum_{(x, y_w, y_l)} \log \sigma[r(x, y_1) - r(x, y_2)].$$

4. Use

$$r(x, y) - \beta [\log \pi_\theta(y|x) - \log \pi_{\theta_0}(y|x)]$$

as reward function and train π with one of RL methods.

Reinforcement Learning with Human (AI) Feedback

Suppose we have an already fine-tuned model, which we denote as π_0 , and we want to train it to match human preferences.

1. At first sample pairs of outputs (y_1, y_2) from π_0 given prompts x .
2. Then, for each sampled pair, get feedback from human (AI) about which answer is more preferable and form a dataset $\mathcal{D} = \{(x, y_w, y_l)\}$.

3. Train a model r that takes inputs x, y and outputs reward. One can do it based on the Bradley-Terry model so that

$$\mathbf{P}\{y_1 \text{ is more preferable}\} = \frac{\exp(r(x, y_1))}{\exp(r(x, y_1)) + \exp(r(x, y_2))}.$$

Then the loss is

$$- \sum_{(x, y_w, y_l)} \log \sigma[r(x, y_1) - r(x, y_2)].$$

4. Use

$$r(x, y) - \beta [\log \pi_\theta(y|x) - \log \pi_{\theta_0}(y|x)]$$

as reward function and train π with one of RL methods.

- Use x from $\mathcal{D}$.
- The model r is often initialized from π_0 with the addition of a linear layer on top of the final transformer layer that produces a prediction for the reward.
- Reward looks the way it does because initially we aim to solve

$$\mathbf{E}_{y \sim \pi_\theta(\cdot|x)} r(x, y) - \beta D_{\text{KL}}(\pi_\theta(\cdot | s_t), \pi_{\theta_0}(\cdot | s_t)) \rightarrow \sup_{\theta}.$$

[3] Direct Preference Optimization: Your Language Model is Secretly a Reward Model

R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, C. Finn
Stanford University, CZ Biohub

May 2023

Direct Preference Optimization

The PPO is a reward-based method. It means that to use it we need to estimate the reward function. To overcome this limitation, one can use reward-free methods, e.g. DPO.

Recall that initially we aim to solve

$$\mathbf{E}_{y \sim \pi_{\theta}(\cdot|x)} r(x, y) - \beta D_{\text{KL}}(\pi_{\theta}(\cdot | s_t), \pi_{\theta_0}(\cdot | s_t)) \rightarrow \sup_{\theta}.$$

One can show that the solution for this problem is

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\theta_0}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right),$$

where Z is the partition function. From this follows that

$$r(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\theta_0}(y|x)} + \beta \log Z(x).$$

Under the Bradley-Terry model, the latter means that

$$\begin{aligned} & \mathbf{P}\{y_1 \text{ is more preferable}\} \\ &= \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\theta_0}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\theta_0}(y_1|x)}\right)}. \end{aligned}$$

Thus, we got to solve

$$-\mathbf{E}_{(x, y_w, y_l)} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_2|x)}{\pi_{\theta_0}(y_2|x)} - \beta \log \frac{\pi_{\theta}(y_1|x)}{\pi_{\theta_0}(y_1|x)} \right) \right] \rightarrow \inf_{\theta}.$$

Direct Preference Optimization

The PPO is a reward-based method. It means that to use it we need to estimate the reward function. To overcome this limitation, one can use reward-free methods, e.g. DPO.

Recall that initially we aim to solve

$$\mathbf{E}_{y \sim \pi_{\theta}(\cdot|x)} r(x, y) - \beta D_{\text{KL}}(\pi_{\theta}(\cdot | s_t), \pi_{\theta_0}(\cdot | s_t)) \rightarrow \sup_{\theta}$$

One can show that the solution for this problem is

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\theta_0}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right),$$

where Z is the partition function. From this follows that

$$r(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\theta_0}(y|x)} + \beta \log Z(x).$$

Under the Bradley-Terry model, the latter means that

$$\begin{aligned} & \mathbf{P}\{y_1 \text{ is more preferable}\} \\ &= \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\theta_0}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\theta_0}(y_1|x)}\right)}. \end{aligned}$$

Thus, we got to solve

$$-\mathbf{E}_{(x, y_w, y_l)} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_2|x)}{\pi_{\theta_0}(y_2|x)} - \beta \log \frac{\pi_{\theta}(y_1|x)}{\pi_{\theta_0}(y_1|x)} \right) \right] \rightarrow \inf_{\theta}.$$

- DPO vs PPO?
PPO is still considered as the industry standard.
- DPO along with PPO?
There are approaches (e.g., [\[4\] Zhong et al., 2025](#)) that try to combine PPO with DPO.

LLM setting

What is an LLM state space?

It is simply the sequences of tokens that might fill in the LLM's context. One can check that for a dictionary of K tokens and the context length of N , number of states is $\frac{K(K^N - 1)}{K - 1}$.

What is an LLM action space?

There are two main approaches:

- a) Action is a generation of next token/group of tokens;
- b) Action is a generation of completion.

In [\[5, Zekri et al., 2024\]](#) authors study the Markov chain on S derived by an LLM.

LLM setting

What is an LLM state space?

It is simply the sequences of tokens that might fill in the LLM's context. One can check that for a dictionary of K tokens and the context length of N , number of states is $\frac{K(K^N-1)}{K-1}$.

What is an LLM action space?

There are two main approaches:

- a) Action is a generation of next token/group of tokens;
- b) Action is a generation of completion.

In [\[5, Zekri et al., 2024\]](#) authors study the Markov chain on S derived by an LLM.

How does PPO loss usually look for LLMs?

Let again

$$R_{\theta}(x|y) = \frac{\pi_{\theta}(x|y)}{\pi_{\theta_{\text{old}}}(x|y)}.$$

Then

$$L_{\text{PPO}}(\theta) = \mathbf{E}_{x \sim \mathcal{D}, o_t \sim \pi_{\theta_{\text{old}}}}$$

$$\frac{1}{|o|} \sum_{t=1}^{|o|} \min[R_{\theta}(o_t|x, o_{<t})\hat{A}_t, \text{clip}(R_{\theta}(o_t|x, o_{<t}), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t],$$

where $\hat{A}_t$ is the estimator for the advantage that is computed based on the rewards $r_{\geq t}$ and a learned value function.

Recall that

$$r_t = r_{\phi}(x, o_{\leq t}) - \beta [\log \pi_{\theta}(o_t|x, o_{<t}) - \log \pi_{\text{ref}}(o_t|x, o_{<t})].$$

[6] DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

Feb 2024

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, D. Guo

DeepSeek-AI, Tsinghua University, Peking University

[7] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

Jan 2025

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang,

X. Bi, X. Zhang, X. Yu, Y. Wu, and 100+ more authors

DeepSeek-AI

Group Relative Policy Optimization

The initial GRPO loss is as follows.

$$L_{\text{GRPO}}^1(\theta) = \mathbf{E}_{x \sim \mathcal{D}, \{o_i\} \sim \pi_{\theta_{\text{old}}}}$$
$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left[\min[r_{\theta}(o_{i,t}|x, o_{i,<t}) \hat{A}_{i,t}, \text{clip}(r_{\theta}(o_{i,t}|x, o_{i,<t}), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{i,t}] \right. \\ \left. - \beta D_{\text{KL}}(\pi_{\theta}(\cdot | x, o_{i,<t}), \pi_{\theta_{\text{ref}}}(\cdot | x, o_{i,<t})) \right],$$

where

$$\hat{A}_{i,t} = \sum_{\tau \geq t} \frac{r_{i,\tau} - \text{mean}_{j,s} r_{j,s}}{\text{std}_{j,s} r_{j,s}}, \quad r_{i,t} = r_{\phi}(x, o_{i,\leq t}).$$

There are two major changes comparing to PPO:

- A group of outputs is generated to estimate the advantage, which is calculated per token only based on the rewards;
- KL penalty removed from the reward and added to the loss directly.

The consequent GRPO loss is as follows.

$$L_{\text{GRPO}}^2(\theta) = \mathbf{E}_{x \sim \mathcal{D}, \{o_i\} \sim \pi_{\theta_{\text{old}}}}$$
$$\frac{1}{G} \sum_{i=1}^G \left[\min[r_{\theta}(o_i|x) \hat{A}_i, \text{clip}(r_{\theta}(o_i|x), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_i] \right. \\ \left. - \beta D_{\text{KL}}(\pi_{\theta}(\cdot | o_i|x), \pi_{\theta_{\text{ref}}}(\cdot | o_i|x)) \right],$$

where

$$\hat{A}_i = \frac{r_i - \text{mean}_j r_j}{\text{std}_j r_j},$$

There are two major changes comparing to PPO:

- A group of outputs is generated to estimate the advantage, which is calculated per output only based on the rewards;
- KL penalty removed from the reward and added to the loss directly;
- Reward is rule-based.

Group Relative Policy Optimization (cont.)

- The GRPO might be becoming the industry standard (Qwen3: [\[8, Qwen Team, 2025\]](#), MiMo: [\[9, Xiaomi LLM-Core Team, 2025\]](#) etc.);
- Some rule-based rewards are easy (?) to hack;
- Perhaps, optimizing the GRPO loss itself leads to overthinking ([\[10, Liu et al., 2025\]](#)).