Policy Optimization

Nguyen Nhat Minh,
HUST
CMC AI Intership
Data Science and Artificial Intelligence

February 15, 2026

Outline

Part II: Policy Optimization Framework

Outline

Trust Policy Policy Optimization(TRPO)

TRPO

The expected return of a policy π as:

$$\eta(\pi) = \mathbb{E}_{s\sim\rho_\pi,\, a\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t, s_{t+1})\right]$$

where ρ_π is discounted visitation frequency distribution

$$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \cdots$$

And it is possible to relate the expected return of two policiesπ_θ and $\pi_{\theta_{old}}$ using advantages:

$$\eta(\theta) = \eta(\theta_{old}) + \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}[A_{\pi_{\theta_{old}}}(s, a)]$$

TRPO

They propose the approximation, given assumption that two policies π and π_{old} not different from another:

$$\eta(\theta) \approx \eta(\theta_{old}) + \mathbb{E}_{s\sim\rho_{\pi_{\theta_{old}}}, a\sim\pi_\theta}[A_{\pi_{\theta_{old}}}(s,a)]$$

Let's now define the following objective function:

$$J_{\theta_{\text{old}}}(\theta) = \eta(\theta_{\text{old}}) + \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim\pi_\theta}\left[A_{\pi_{\theta_{\text{old}}}}(s,a)\right]$$

It is easy to see that

$$J_{\theta_{\text{old}}}(\theta_{\text{old}}) = \eta(\theta_{\text{old}})$$

by definition of the advantage, and that its gradient w.r.t to θ taken in θ_{old} is the same as the one of $\eta(\theta_{\text{old}})$:

$$\nabla_\theta J_{\theta_{\text{old}}}(\theta)\big|_{\theta=\theta_{\text{old}}} = \nabla_\theta \eta(\theta)\big|_{\theta=\theta_{\text{old}}}$$

TRPO

Suppose that we can find its maximum, i.e. a policy π' that maximizes the advantage of each state-action pair over $\pi_{\theta_{\text{old}}}$. No guarantee that π' and $\pi_{\theta_{\text{old}}}$ are close enough so that the assumption stands $\Rightarrow$ make only a small step in its direction and hope for the best:

$$\pi_\theta(s,a) = (1-\alpha)\,\pi_{\theta_{\text{old}}}(s,a) + \alpha\,\pi'(s,a)$$

This is the conservative policy iteration method, where a bound on the difference between $\eta(\pi_{\theta_{\text{old}}})$ and $J_{\theta_{\text{old}}}(\theta)$ can be derived.

- 1. Adding a hard constraint on the KL divergence, leading to a constrained optimization problem (Lagrange methods can be applied):

$$\begin{aligned} &\text{maximize}_\theta && J_{\theta_{\text{old}}}(\theta) \\ &\text{subject to} && D_{KL}\left(\pi_{\theta_{\text{old}}} \,\|\, \pi_\theta\right) \le \delta \end{aligned}$$

- 2. Regularizing the objective function with the KL divergence:

$$\text{maximize}_\theta \quad L(\theta) = J_{\theta_{\text{old}}}(\pi_\theta) - C\,D_{KL}\left(\pi_{\theta_{\text{old}}} \,\|\, \pi_\theta\right)$$

TRPO

The practical implementation uses the constrained optimization problem:

$$\begin{aligned} \text{maximize}_\theta \quad & J_{\theta_{\text{old}}}(\theta) = \eta(\theta_{\text{old}}) + \mathbb{E}_{s\sim\rho_{\pi_{\theta_{\text{old}}}}, a\sim\pi_\theta}\left[A_{\pi_{\theta_{\text{old}}}}(s,a)\right] \\ \text{subject to} \quad & D_{KL}\left(\pi_{\theta_{\text{old}}} \,\|\, \pi_\theta\right) \leq \delta \end{aligned}$$

- Notice that $\eta(\theta_{\text{old}})$ does not depend on θ, so it is constant in the optimization problem. We only need to maximize the advantage of the actions taken by π_θ in each state visited by $\pi_{\theta_{\text{old}}}$.
- The problem is that π_θ is what we search, so we cannot sample actions from it.
- $\Rightarrow$ The solution is to use **importance sampling** to allow sampling actions from $\pi_{\theta_{\text{old}}}$:

$$\begin{aligned} \text{maximize}_\theta \quad & \mathbb{E}_{s\sim\rho_{\pi_{\theta_{\text{old}}}}, a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(s,a)}{\pi_{\theta_{\text{old}}}(s,a)} A_{\pi_{\theta_{\text{old}}}}(s,a)\right] \\ \text{subject to} \quad & D_{KL}\left(\pi_{\theta_{\text{old}}} \,\|\, \pi_\theta\right) \leq \delta \end{aligned}$$

TRPO

Lastly, the advantages $A_{\pi_{\theta_{\text{old}}}}(s,a) = Q_{\pi_{\theta_{\text{old}}}}(s,a) - V_{\pi_{\theta_{\text{old}}}}(s)$ depend on the value of the states encountered by $\pi_{\theta_{\text{old}}}$.
$\Rightarrow$ The state values do not depend on the policies; they are constant for each optimization step, so they can also be safely removed:

$$\begin{aligned} \text{maximize}_\theta \quad & \mathbb{E}_{s\sim\rho_{\pi_{\theta_{\text{old}}}},\, a\sim\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(s,a)}{\pi_{\theta_{\text{old}}}(s,a)} Q_{\pi_{\theta_{\text{old}}}}(s,a) \right] \\ \text{subject to} \quad & D_{KL}\left(\pi_{\theta_{\text{old}}} \,\|\, \pi_\theta\right) \leq \delta \end{aligned}$$

To solve this problem we need to use conjugate gradient algorithm and others complicated techniques in optimization.

TRPO

Algorithm 5 TRPO

Require: $\delta \in \mathbb{R}, b \in (0,1), K \in \mathbb{N}, \alpha \in (0,1], U \in \mathbb{N}, T \in \mathbb{N}$

Initialize θ and ϕ at random
$t \leftarrow 0$
while $t \leq T$ **do**
  for $i = 1, \ldots, U$ **do**
    $a \sim \pi_\theta$ ▷ sample action
    $\beta(a \mid s) \leftarrow \pi_\theta(a \mid s)$
    $s, r \sim P(s, a)$ ▷ sample next state and reward
    $t \leftarrow t + 1$
    Store $(a, s, r, \beta(a \mid s))$ in $\mathcal{D}$
  end for
  for all epochs **do**
    Compute returns R and advantages A
    $g \leftarrow \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \nabla_\theta \frac{\pi_\theta(a|s)}{\beta(a|s)} A$
    Compute $\hat{H}$ as the Hessian of the sample average KL-divergence
    Compute $d \approx \hat{H}^{-1} g$ via conjugate gradient algorithm
    $m \leftarrow 0$
    repeat
      $\theta \leftarrow \theta_{\text{old}} + b^m \sqrt{\frac{2\delta}{d^\intercal \hat{H} d}} d$
      $m \leftarrow m + 1$
    until (sample loss improves and KL constraint satisfied) or $m > K$
    $d\phi \leftarrow \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \nabla_\phi (R - V_\phi(s))^2$
    Update ϕ using $d\phi$ via gradient descent
  end for
end while

Proximal Policy Optimization(PPO))

PPO

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$. The main objective PPO paper propose is the following:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

For language models, the objective (or loss) is computed per token, which intuitively can be grounded in how one would compute the probability of the entire sequence of autoregressive predictions – by a product of probabilities:

$$J(\theta) = \frac{1}{|a|} \sum_{t=0}^{|a|} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon \right) A_t \right).$$

This is the per-token version of PPO, which also applies to other policy-gradient methods.

PPO

The original paper initialized the value estimator from the trained reward model. Since PPO is an actor-critic algorithm, the value estimator is updated concurrently with the policy, via minimizing the squared TD-error, which in this case equals the squared advantage term:

$$L_{\mathrm{TD}}(\xi) = \mathbb{E}_{(x,y)\sim D_{\phi_t}^{\mathrm{RL}}}\left[\left(r_\theta(x,y) - \beta \log \frac{\pi_{\phi_t}^{\mathrm{RL}}(y|x)}{\pi^{\mathrm{SFT}}(y|x)} - V_\xi(x)\right)^2\right]$$

which is minimized by gradient *descent* on it. Other methods than squared TD-error might be used.

The PPO Clipping Mechanism

Core Intuition: The clipping term removes the incentive for the policy to change too drastically in a single step.

Case 1: Good Action ($A_t > 0$)

- **Goal:** Increase probability $\pi(a)$.
- **Limit:** We boost π_{new} only up to $(1+\epsilon)\pi_{\text{old}}$.
- **Result:** If π_{new} exceeds this limit, the gradient becomes **0**.
- *Prevents over-reinforcing a lucky step.*

Case 2: Bad Action ($A_t < 0$)

- **Goal:** Decrease probability $\pi(a)$.
- **Limit:** We reduce π_{new} only down to $(1-\epsilon)\pi_{\text{old}}$.
- **Result:** If π_{new} drops below limit, the gradient becomes **0**.
- *Prevents destroying the policy due to a single bad sample.*

Conclusion

PPO performs updates **only** when the ratio $\frac{\pi_{\text{new}}}{\pi_{\text{old}}}$ is within the interval $[1-\epsilon, 1+\epsilon]$. Outside this range, the model stops learning to ensure stability.

PPO- Mixing pretraining gradients

- A third term is added to the objective function to prevent from catastrophic forgetting.
- RLHF incorporates the original language modeling objective: random texts x are sampled from the original pretraining dataset $\mathcal{D}_{\text{pretrain}}$, model trained to maximize the log-likelihood of the text $\log\left(\pi_\phi^{\text{RL}}(x)\right)$.

$$\mathcal{L}(\phi) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_\phi}^{\text{RL}}}\left[r_\theta(x,y) - \beta\log\left(\frac{\pi_\phi^{\text{RL}}(y\mid x)}{\pi^{\text{SFT}}(y\mid x)}\right)\right]$$

$$+\ \gamma\,\mathbb{E}_{x\sim\mathcal{D}_{\text{pretrain}}}\left[\log\left(\pi_\phi^{\text{RL}}(x)\right)\right].$$

- The final PPO-ptx Objective:

$$\mathcal{L}_{\text{PPO-ptx}}(\phi) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_{\phi_t}}^{\text{RL}}}\left[\min\left(\frac{\pi_\phi^{\text{RL}}(y\mid x)}{\pi_{\phi_t}^{\text{RL}}(y\mid x)}A(x,y),\ \text{clip}\left(\frac{\pi_\phi^{\text{RL}}(y\mid x)}{\pi_{\phi_t}^{\text{RL}}(y\mid x)},\right.\right.\right.$$

$$\left.\left.\left.1\text{-}\epsilon, 1+\epsilon\right)A(x,y)\right) - \beta\log\left(\frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right)\right] + \gamma\,\mathbb{E}_{x\sim\mathcal{D}_{\text{pretrain}}}\left[\log\left(\pi_\phi^{\text{RL}}(x)\right)\right].$$

Outline

RLHF

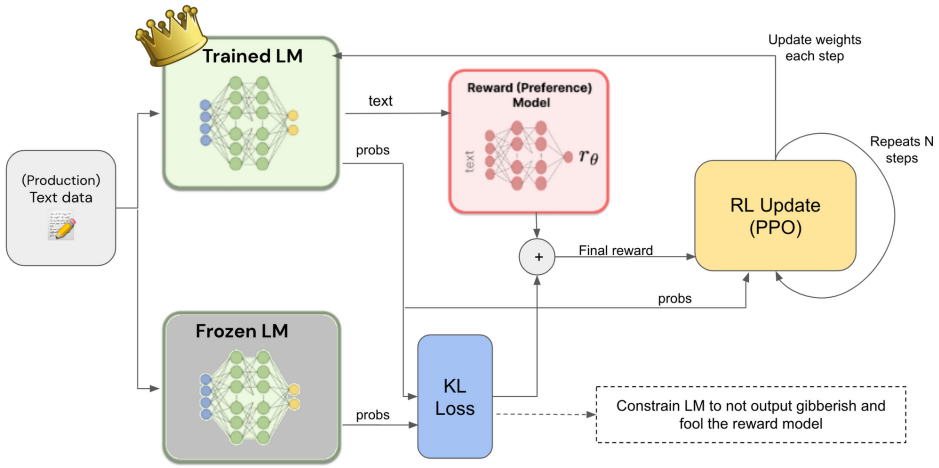

Figure: RLHF

In RLHF, two different models are trained: a reward model and a reinforcement learning (RL) policy.

- The reward model learns to determine what behavior is desirable based on human feedback.
- The policy guided by the reward model to determine the agent's actions.
- Both models are commonly initialized using a pre-trained autoregressive language model.

Reward Model

Human preference: The fine-tuned model generates pairs of responses (y_1, y_2) for given prompt x. Then human select their preferred response, y_w (the "winner"), over the less preferred, y_l (the "loser"). These human preferences are then used as training data for the next step.

- Create reward model $r_\phi(x, y)$ assigns a score to each response y given prompt x reflect how well the response aligns with human preferences.
- The reward model is usually initialized with a pre-trained model. Then is trained by replacing the final layer of the previous model with a randomly initialized regression head.
- This change shifts the model from original classification task to outputting a score of any given prompt and response:

$$\mathcal{L}(\theta) = -\frac{1}{\binom{K}{2}} \mathbb{E}_{(x,y_w,y_l)} \left[\log\left(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l))\right)\right]$$

$$= -\frac{1}{\binom{K}{2}} \mathbb{E}_{(x,y_w,y_l)} \left[\log\left(\frac{e^{r_\theta(x,y_w)}}{e^{r_\theta(x,y_w)} + e^{r_\theta(x,y_l)}}\right)\right].$$

Reward Modeling

Modeling Pairwise Preferences

To train this reward model, we rely on **pairwise comparisons** of responses y_w (winner) and y_l (loser). The preferences are modeled using the **Bradley–Terry framework**, which assigns a probability to the preference:

1. The probability of y_w being preferred over y_l is:

$$p_\phi(y_w > y_l \mid x) = \frac{\exp r_\phi(x, y_w)}{\exp r_\phi(x, y_w) + \exp r_\phi(x, y_l)}.$$

- $r_\phi(x, y_w)$ and $r_\phi(x, y_l)$: Rewards (scores) assigned to the winner and loser, respectively.
- The numerator $\exp r_\phi(x, y_w)$: Represents the likelihood of the winner being the preferred choice.
- The denominator ensures probabilities sum to 1 by including both options: $\exp r_\phi(x, y_w) + \exp r_\phi(x, y_l)$.

Reward Modeling

2. Rearranging, we write the probability in terms of a difference between rewards:

$$p_\phi(y_w > y_l \mid x) = \frac{1}{1 + \exp[r_\phi(x, y_l) - r_\phi(x, y_w)]}.$$

- $r_\phi(x, y_l) - r_\phi(x, y_w)$: The difference between the scores of the loser and the winner.
- $\exp(\cdot)$: Converts this difference into a scaling factor for the probability.

3. Using the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}},$$

the equation becomes:

$$p_\phi(y_w > y_l \mid x) = \sigma(r_\phi(x, y_w) - r_\phi(x, y_l)).$$

- $r_\phi(x, y_w) - r_\phi(x, y_l)$: If the winner's score is much higher than the loser's, the probability approaches 1 (the winner is strongly preferred).

Reward Modeling

To train r_ϕ, we optimize it to match human preferences as closely as possible. This is done using **maximum likelihood estimation (MLE)**:

- The loss function for the reward model is:

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right].$$

- $\mathcal{D}$: Dataset of human preferences (pairs y_w, y_l for each prompt x).
- $\log \sigma(\cdot)$: Penalizes predictions that assign low probabilities to actual human preferences.
- The goal is to **minimize the negative log-likelihood**, ensuring the reward model aligns its predictions with the collected human feedback.

Policy

- Similarly to the reward model, the human feedback policy is also initialized from a pre-trained model.
- The first step in its training is supervised fine-tuning (SFT). This step does not require the reward model. Instead, the pre-trained model is trained on a dataset that contains prompt-response pairs (x, y).
- Then, during SFT, the model is trained to auto-regressively generate the corresponding response y when given a random prompt x. The original paper recommends to SFT for only one epoch, since more than that causes overfitting.
- The dataset D_{SFT} is usually written by human contractors, who write both the prompts and responses.

Policy

The second step uses a policy gradient method to the reward model. It uses a dataset D_{RL}, which contains prompts, but not responses. Policy Gradient

1. Initialize policy $\pi_\phi^{\text{RL}} \leftarrow \pi^{\text{SFT}}$
2. For each outer iteration:
    1. Initialize empty dataset $D_\phi^{\text{RL}} \leftarrow \emptyset$
    2. For each rollout step:
        1. Sample prompt $x \sim D_{\text{RL}}$
        2. Sample response $y \sim \pi_\phi^{\text{RL}}(\cdot \mid x)$
        3. Compute reward $r_\theta(x, y)$
        4. Store $(x, y, r_\theta(x, y))$ in D_ϕ^{RL}
    3. Update ϕ to maximize:

$$\mathbb{E}_{(x,y)\sim D_\phi^{\text{RL}}} \left[r_\theta(x, y) - \beta \log \frac{\pi_\phi^{\text{RL}}(y \mid x)}{\pi^{\text{SFT}}(y \mid x)} \right]$$

Group Relative Policy Optimization(GRPO)

GRPO

It extends PPO by discarding the value network and replacing it with a group-relative baseline computed from multiple outputs sampled for the same prompt. It brings 2 benefits:

- Avoiding the challenge of learning a value function from a LM backbone, where research hasn't established best practices.
- Saves memory by not needing to keep the extra set of model weights in memory.

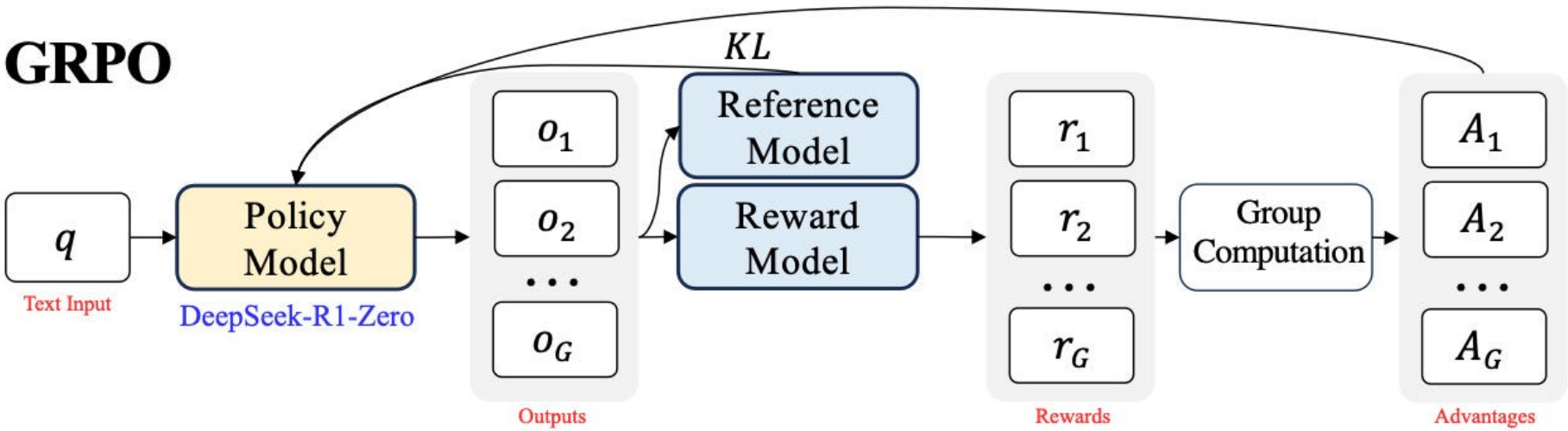

Figure: GRPO

Group Relative Baseline.

- For each prompt p, we sample a group of G full continuations $a_1, \ldots, a_G \sim \pi_\theta(\cdot \mid p)$, where each continuation $a_{i,t} = (a_{i,1}, \ldots, a_{i,t})$ is a sequence of tokens indexed by timestep t.
- The frozen reward model $r_\phi(p, a_{i,t})$ then assigns a scalar score to each token $a_{i,t}$ conditioned on the prompt. These sequence-level rewards are then normalized across the group to compute a group-relative advantage signal:

$$\hat{A}_{i,t} = \tilde{r}_{i,t} = \frac{r_{i,t} - \mathrm{mean}(r_{\cdot,t})}{\mathrm{std}(r_{\cdot,t})}, \tag{11}$$

where $\mathrm{mean}(\cdot)$ and $\mathrm{std}(\cdot)$ denote the mean and standard deviation functions used to compute the group-relative advantage.

GRPO

The same $\hat{A}_i$ is reused for every token $a_{i,t}$ in the continuation, producing the clipped surrogate:

$$\mathcal{L}_{\mathrm{GRPO}} = \mathbb{E}_{p\sim\mathcal{D}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|a_i|}\sum_{t=1}^{|a_i|}\min\Big(\rho_{i,t}\hat{A}_{i,t},\ \mathrm{clip}(\rho_{i,t},1-\epsilon,1+\epsilon)\hat{A}_{i,t}\Big)\right]$$

$- \beta\,\mathbb{E}_p\left[D_{\mathrm{KL}}(\pi_\theta(\cdot\mid p)\,\|\,\pi_{\mathrm{ref}}(\cdot\mid p))\right],$

The group-relative advantage $\hat{A}_i$ replaces the value baseline V, roughly halving memory and retaining a low-variance learning signal.

Prompt-level KL estimator: Instead of injecting a token-wise penalty to reward (PPO: $\beta \log \frac{\pi_\theta}{\pi_{\mathrm{ref}}}$),GRPO adds a separate prompt-level regulariser:

$$D_{\mathrm{KL}}(p) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|a_i|}\sum_{t=1}^{|a_i|}\log\left(\frac{\pi_\theta(a_{i,t}\mid s_{i,t})}{\pi_{\mathrm{ref}}(a_{i,t}\mid s_{i,t})}\right).$$

Why Move Beyond RLHF?

The Fundamental Flaw:
RLHF relies on an intermediate proxy—the **Reward Model**.

The Indirect Path (RLHF)

1. Collect Human Data.
2. Train a Reward Model (RM) to mimic humans.
3. Use PPO to optimize the Policy against the RM.

$\rightarrow$ Errors accumulate at every step.

The Core Issues

- **Mismatch:** The Reward Model is essentially a "lossy compression" of human preferences.
- **Cost:** Collecting pairwise preference data is expensive and slow.
- **Complexity:** Managing the PPO feedback loop is computationally heavy.

Question: Can we optimize the policy directly on the data without the Reward Model?

Outline

Direct Alignment Algorithms (DAA)

Definition

DAA is a new class of algorithms that seeks to **directly optimize** LLMs on human feedback in a supervised manner, replacing traditional policy-gradient methods.

Key Mechanism: Removing the Middleman

- Eliminates the need for a separate **Reward Model (RM)**.
- Trains end-to-end on human-labeled or curated outputs.
- Avoids "proxy objectives" and reduces risks of **reward hacking**.

Core Advantages:

- **Simpler Pipelines:** Easier to train than RLHF.
- **Tighter Alignment:** Directly targets human-preferred behavior.
- **Interpretability:** Optimization process is more transparent.

Direct Preference Optimization(DPO)

KL-Regularized RL

DPO begins from the KL-regularized reinforcement learning objective used in traditional RLHF:

$$\pi^*(y|x) = \arg\max_{\pi} \mathbb{E}_{x,y\sim\pi}\Big[r_\phi(x,y) - \beta\,\mathrm{KL}(\pi(y|x) \,\|\, \pi_{\text{ref}}(y|x))\Big],$$

where:

- $r_\phi(x,y)$ is a reward model parameterized by ϕ,
- π_{ref} is the reference (SFT) model,
- $\beta > 0$ is a temperature parameter controlling conservativeness,
- KL regularization prevents drifting too far from the base model.

The solution to this optimization problem is known (from entropy-regularized RL):

$$\pi^*(y|x) = \frac{\pi_{\text{ref}}(y|x)\,\exp\left(r_\phi(x,y)/\beta\right)}{Z(x)},$$

where:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x)\exp\left(r_\phi(x,y)/\beta\right).$$

DPO

1. **Substitute the KL-divergence formula:**

$$\max_{\pi_\theta} \left\{ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[r_\phi(x,y) - \beta \log \left(\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \right) \right] \right\}.$$

2. **Extract the constant $-\frac{1}{\beta}$ and apply an identity transformation:**

$$\min_{\pi_\theta} \left\{ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r_\phi(x,y) \right] \right\}.$$

3. **Continue the transformation:**

$$\min_{\pi_\theta} \left\{ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - \log e^{r_\phi(x,y)/\beta} \right] \right\}.$$

4. **Obtain:**

$$\min_{\pi_\theta} \left\{ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x) \exp(r_\phi(x,y)/\beta)} \right] \right\}.$$

DPO

Define a new distribution $\pi_r(y|x)$ as:

$$\pi_r(y|x) = \frac{\pi_{\text{ref}}(y|x)\exp\left(r_\phi(x,y)/\beta\right)}{Z(x)},$$

where $Z(x)$ is the normalization constant.

5. **Continue with an equivalent transformation of the PPO loss:**

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)\exp\left(\frac{r_\phi(x,y)}{\beta}\right)\frac{1}{Z(x)}Z(x)}\right].$$

6. **Simplify:**

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\pi_r(y|x)} - \log Z(x)\right].$$

DPO

7. **Ignore the $\log Z(x)$ term (independent of π_θ):**

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\pi_r(y|x)}\right].$$

8. **Express the objective as a KL divergence:**

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}} \left[D_{\mathrm{KL}}\big(\pi_\theta(y|x) \,\|\, \pi_r(y|x)\big)\right].$$

- Since the KL divergence is always non-negative and equals zero if and only if the two distributions are identical, the optimal condition is achieved when

$$\pi_\theta(y \mid x) = \pi_r(y \mid x).$$

- If the parameters of the reward model r_ϕ are fixed, then the optimal solution of PPO is given by

$$\pi_r(y \mid x) = \frac{\pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{r_\phi(x,y)}{\beta}\right)}{Z(x)}.$$

DPO

- In practice, however, the reward function $r(x, y)$ used in alignment training is not chosen arbitrarily.
- Specifically, we first train an ideal reward model r_ϕ^* and then, based on this optimal reward model, we further train an aligned policy.
- Consequently, the optimal reward model r_ϕ^* and the aligned policy it induces, denoted by π_r^*, continue to satisfy the relationship:

$$\pi_r^*(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x) \exp\left(\frac{r_\phi^*(x,y)}{\beta}\right)}{Z(x)}.$$

DPO

Although we have formally derived

$$\pi_r(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x)\ \exp\left(\frac{r_\phi(x,y)}{\beta}\right)}{Z(x)},$$

In practice this explicit solution is difficult to apply directly because:

1. **Estimating $Z(x)$ is challenging:** requires sampling all possible responses y for a given prompt x to compute $\exp(r_\phi(x, y)) \cdot \pi_{\text{ref}}(y \mid x)$,which is extremely costly.
2. **The original goal was to aboid training the reward model:** We intended to learn an aligned model in one step rather than first training a reward function. However, π_r still depends on the reward function r.

DPO

If we have the optimal aligned model π^*, we can derive its corresponding reward function r_ϕ^*

$$r_\phi^*(x, y) = \beta \log \left(\frac{\pi^*(y \mid x)\, Z(x)}{\pi_{\text{ref}}(y \mid x)} \right) + \beta \log Z(x).$$

$\rightarrow$Allows to represent π^* in terms of r_ϕ^*, $\rightarrow$ bridging the gap between training the reward model and the aligned model.

Since can express the optimal reward model r_ϕ^* in terms of the optimal aligned model π^*, then can directly substitute π^* into the reward model's training objective.

$\rightarrow$While training a reward model, in fact directly obtaining the optimal aligned model in one fell swoop.

DPO

The remaining challenge shifts to training the reward model itself.

1. **Generating Only Two Responses**
   For a given prompt x, generate two responses, <prompt x, chosen y1, reject y2>. Human annotations indicate which move is better, and the objective is to have the reward model assign a higher score to the chosen move while scoring the rejected move lower.
2. **Generating K Responses ($K > 2$)**
   For the same prompt x, generate multiple responses, such as <prompt x, y1, ..., yK>. Suppose human annotators provide a preference ranking τ (e.g., $y_2 > y_3 > y_1 > \cdots > y_K$). We aim for the reward model to give the highest overall score to the true ranking τ while scoring any other ordering lower.

Generating 2 responses:

$$
\begin{aligned}
L(r_\phi, D) &= -\mathbb{E}_{x,y_w,y_l \in D}\Big[\log(P(y_w > y_l \mid x))\Big] \\
&= -\mathbb{E}_{x,y_w,y_l \in D}\left[\log\left(\frac{e^{r(x,y_w)}}{e^{r(x,y_w)} + e^{r(x,y_l)}}\right)\right] \\
&= -\mathbb{E}_{x,y_w,y_l \in D}\left[\log\left(\frac{1}{1 + e^{-(r(x,y_w)-r(x,y_l))}}\right)\right] \\
&= -\mathbb{E}_{x,y_w,y_l \in D}\Big[\log\left(\sigma(r(x,y_w) - r(x,y_l))\right)\Big].
\end{aligned}
$$

Assuming we have found the optimal reward model of the form

$$
r^*_\phi(x,y) = \beta \log\left(\frac{\pi^*(y \mid x)\, Z(x)}{\pi_{\text{ref}}(y \mid x)}\right) + \beta \log Z(x),
$$

DPO

Reward Model Loss

$$= \max_{\pi^*} \left\{ \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi^*(y_{\text{win}} \mid x)}{\pi_{\text{ref}}(y_{\text{win}} \mid x)} + \beta \log Z(x) \right.\right.\right.$$

$$\left.\left.\left. - \beta \log \frac{\pi^*(y_{\text{loss}} \mid x)}{\pi_{\text{ref}}(y_{\text{loss}} \mid x)} - \beta \log Z(x) \right) \right] \right\}$$

$$= \max_{\pi_\theta} \left\{ \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi^*(y_{\text{win}} \mid x)}{\pi_{\text{ref}}(y_{\text{win}} \mid x)} - \beta \log \frac{\pi^*(y_{\text{loss}} \mid x)}{\pi_{\text{ref}}(y_{\text{loss}} \mid x)} \right) \right] \right\}.$$

After a minor adjustment and setting our trainable aligned model as π_θ ,the final objective becomes: Reward Model Loss

$$= \max_{\pi_\theta} \left\{ \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_{\text{win}} \mid x)}{\pi_{\text{ref}}(y_{\text{win}} \mid x)} - \beta \log \frac{\pi_\theta(y_{\text{loss}} \mid x)}{\pi_{\text{ref}}(y_{\text{loss}} \mid x)} \right) \right] \right\}$$

$$= \text{DPO Loss.}$$

DPO

Generating many responses: Define the probability that the true ranking τ beats all other orderings as:

$$P(\tau \mid y_1, y_2, \ldots, y_K, x) = \prod_{k=1}^{K} \frac{e^{r(x, y_{\tau_k})}}{\sum_{j=k}^{K} e^{r(x, y_{\tau_j})}}.$$

Next, substitute the optimal reward function $r_\phi^*(x, y)$ into the above equation. First, express $r_\phi^*(x, y)$ as

$$r_\phi^*(x, y) = \beta \log \left(\frac{\pi^*(y \mid x)\, Z(x)}{\pi_{\text{ref}}(y \mid x)} \right) + \beta \log Z(x),$$

then the probability becomes

$$P(\tau \mid y_1, y_2, \ldots, y_K, x) = \prod_{k=1}^{K} \frac{e^{r_\phi^*(x, y_{\tau_k})}}{\sum_{j=k}^{K} e^{r_\phi^*(x, y_{\tau_j})}}.$$

DPO

We then express r_ϕ^* in terms of π^*, so that the expression can be rewritten as

$$P(\tau \mid y_1, y_2, \ldots, y_K, x) = \prod_{k=1}^{K} \frac{\exp\left(\beta \log \frac{\pi^*(y|x)\, Z(x)}{\pi_{\text{ref}}(y|x)}\right)}{\sum_{j=k}^{K} \exp\left(\beta \log \frac{\pi^*(y|x)\, Z(x)}{\pi_{\text{ref}}(y|x)}\right)}.$$

Finally, for the entire dataset, the goal is to maximize the probability of the true ranking over the dataset. Thus, for the multi-response case, the DPO objective function can be written as:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta, \pi_{\text{ref}}) = -\mathbb{E}_{\tau, y_1, y_2, \ldots, y_K, x \sim D} \left[\log \prod_{k=1}^{K} \frac{\exp\left(\beta \log \frac{\pi_\theta(y|x)\, Z(x)}{\pi_{\text{ref}}(y|x)}\right)}{\sum_{j=k}^{K} \exp\left(\beta \log \frac{\pi_\theta(y|x)\, Z(x)}{\pi_{\text{ref}}(y|x)}\right)} \right].$$

Identity preference optimization(IPO)

DPO Flaw: "The Unbounded Drive"

In DPO, the loss function is derived from the Bradley-Terry model:

$$\mathcal{L}_{DPO} = -\log \sigma \left(\beta \log \frac{\pi(y_w)}{\pi_{ref}(y_w)} - \beta \log \frac{\pi(y_l)}{\pi_{ref}(y_l)} \right)$$

The sigmoid function (σ) only reaches 1 when the input is $+\infty$. This means DPO is never essentially "satisfied." Even if the model prefers the winning answer (y_w) much more than the losing answer (y_l), DPO constantly tries to widen that gap to infinity to drive the loss to absolute zero.

The Consequence: The model drifts further and further away from the reference model (π_{ref}) just to squeeze out tiny improvements in the probability gap. This creates the **overfitting**. The KL divergence regularization becomes weak because the objective incentivizes ignoring it to maximize the reward margin.

The general framework:Ψ-PO

Instead of assuming the Bradley-Terry model (Sigmoid) is the only way to model preferences, Azar et al. introduce a generalized mapping function Ψ.

The General Relationship:
We define the relationship between the ground truth preference probability p^* and the implicit reward gap h_π:

$$p^*(y_w \succ y_l|x) = \Psi(h_\pi(x, y_w) - h_\pi(x, y_l))$$

Mapping Choices:

1. **DPO (Logistic):** $\Psi(q) = \sigma(q) = \frac{1}{1+e^{-q}}$
   - *Problem:* Output is bounded in $[0, 1]$, but input required to reach 1 is ∞.
2. **IPO (Identity):** $\Psi(q) = q$
   - *Solution:* The mapping is linear and unbounded, allowing us to solve for a finite gap.

Deriving IPO - The Identity assumption

Applying the Identity Map: If we assume Ψ is the Identity function, the relationship simplifies to:

$$p^*(y_w \succ y_l|x) = h_\pi(x, y_w) - h_\pi(x, y_l)$$

The Optimization Goal: We want to find a policy π such that its log-ratio gap matches the ground truth preference p^* (regularized). Instead of maximizing likelihood (classification), we minimize the **squared error** (regression) between the predicted gap and the target label.

The Functional Objective:

$$\min_\pi \mathbb{E}_{(x,y_w,y_l)} \left[\left(\underbrace{h_\pi(x, y_w) - h_\pi(x, y_l)}_{\text{Predicted Gap}} - \underbrace{\text{Target}}_{\text{Label}} \right)^2 \right]$$

Determine the Target

The optimal policy π^* takes the following form:

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

$$\Rightarrow \frac{1}{\beta}r(x,y) = \log\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \log Z(x)$$

Let $h_{\pi^*}(x,y) = \log\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)}$.

$$\Rightarrow \frac{1}{\beta}r(x,y) = h_{\pi^*}(x,y) + \text{const}_x$$

$$\Rightarrow \frac{1}{\beta}(r(x,y_w) - r(x,y_l)) = h_{\pi^*}(y_w) - h_{\pi^*}(y_l)$$

Interpretation: The difference in the model's log-likelihood ratios (h) must theoretically match the scaled difference in ground truth rewards.

The Final IPO Objective Function

The Formula:
Substituting the target $\frac{1}{2\beta}$ into the Least Squares framework:

$$\mathcal{L}_{IPO}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)} - \frac{1}{2\beta}\right)^2\right]$$

Key Components:

1. h_π **(The Log-Ratio):** Measures how much the model deviates from π_{ref} to favor the winner.
2. $\frac{1}{2\beta}$ **(The Margin):** A fixed, finite target value.
3. **Square** $(\cdot)^2$: Penalizes both under-confidence (gap too small) and over-confidence (gap too large).

IPO-Gradient

DPO Gradient:

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\beta\sigma(-\beta r)\nabla_\theta r$$

- As $r \to \infty$, gradient $\to 0$, but never truly stops pushing.
- If $r < 0$ (wrong prediction), gradient is massive (β).

IPO Gradient:

$$\nabla_\theta \mathcal{L}_{\text{IPO}} = 2\left(r - \frac{1}{2\beta}\right)\nabla_\theta r$$

- **Linear Scaling:** The gradient scales linearly with the error.
- **Zero Crossing:** If $r = \frac{1}{2\beta}$, the gradient is exactly **0**. The model stops learning.
- **Correction:** If $r > \frac{1}{2\beta}$, the gradient is **positive**, pushing the weights *backwards* to reduce the gap.

Kahneman-Tversky optimization(KTO)

KTO-Motivation

The Limitation of DPO/IPO (Pairwise Methods):

- **Requirement:** Requires preference pairs (x, y_w, y_l).
- **Cost:** Collecting pairs is expensive. Humans have to read two outputs and decide.
- **Data Scarcity:** most real-world data is **unary** (e.g., a "Like" on YouTube, a "Bug Report" on GitHub, a "Thumbs Up" in ChatGPT).

The KTO Solution:

- **Input:** Unpaired data (x, y, label).
- **Label:** Binary signal: **Desirable** (good) or **Undesirable** (bad).
- **Goal:** Learn alignment from "Good/Bad" flags without needing a direct comparison.

KTO

1. The Implicit Reward (r_θ):
Just like DPO, we define the reward as the log-likelihood ratio between the policy π_θ and the reference model π_{ref}.

$$r_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

2. The Reference Point (z_0):
This represents the "neutral" expectation. Theoretically, it is the partition function or the KL divergence. In practice, it acts as the threshold between "Good" and "Bad".

$$z_0 \approx \mathbb{E}_{y' \sim \pi_{\text{ref}}}[r_\theta(x, y')]$$

(In implementation, this is often the mean reward of the current batch).

KTO - HALO objectives

KTO maximizes a utility function $v(x, y)$. This function is **piecewise**, meaning it changes depending on whether the data is labeled "Desirable" or "Undesirable".

The Optimization Problem:

$$\max_{\pi_\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[v(x, y)]$$

The Value Function $v(x, y)$:

$$v(x, y) = \begin{cases} \lambda_D \sigma(r_\theta(x, y) - z_0) & \text{if } y \text{ is Desirable} \\ \lambda_U \sigma(z_0 - r_\theta(x, y)) & \text{if } y \text{ is Undesirable} \end{cases}$$

- σ: Logistic Sigmoid function.

KTO

Case A: Desirable Output (y^+)

Scenario: Human provides positive feedback (e.g., "Thumbs Up").

Equation:

$$v(x, y) = \lambda_D \sigma(\underbrace{r_\theta(x, y) - z_0}_{\text{Push Positive}})$$

Mathematical Goal:
We want $r_\theta(x, y) > z_0$. The model attempts to raise the likelihood of this output *above* the baseline expectation.

Case B: Undesirable Output (y^-)

Scenario: Human provides negative feedback (e.g., "Thumbs Down").

Equation:

$$v(x, y) = \lambda_U \sigma(\underbrace{z_0 - r_\theta(x, y)}_{\text{Push Negative}})$$

Mathematical Goal:
We want $r_\theta(x, y) < z_0$. The model attempts to lower the likelihood of this output *below* the baseline expectation.

KTO

The Asymmetry ($\lambda_U > \lambda_D$): We typically set the weight for Undesirable examples (λ_U) higher than Desirable ones (λ_D).$\Rightarrow$ This tells the model: *"It is good to generate correct answers, but it is* **unacceptable** *to generate bad answers."*

- helps prevent the model from hallucinating or generating toxic content, even if it means being slightly more conservative.
- By setting λ_U higher, the gradient for bad examples is steeper. The model prioritizes **avoiding errors** over **maximizing creativity**.
- This creates a "risk-averse" policy.

Outline

Reinforcement Learning from AI Feedback (RLAIF)

Overview of RLAIF

▷ **Motivation** Reinforcement Learning from Human Feedback (RLHF) is effective but costly and difficult to scale due to the need for human annotations. RLAIF [5] replaces human evaluators with a strong AI model to generate feedback, enabling scalable and low-cost alignment.

▷ **Core Idea** Use an AI-based evaluator (critic model) to assess and rank model outputs, and treat the AI-generated feedback as a reward signal for policy optimization.

▷ **Objective** Given a prompt x and a response y, the policy π_θ is optimized to maximize:

$$\max_\theta \ \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r_\phi^{\text{AI}}(x,y)\right] - \beta\,\mathrm{KL}\left(\pi_\theta \,\|\, \pi_{\text{ref}}\right)$$

where r_ϕ^{AI} is an AI-generated reward model and π_{ref} is a reference policy.

▷ RLAIF enables rapid iteration but may introduce bias inherited from the AI evaluator.

Reinforcement Learning from AI Feedback (RLAIF)

Pipeline RLAIF

1. **AI Preference Generation:** A reference model evaluates multiple candidate responses and provides preference labels.
2. **Reward Modeling:** A reward model is trained on AI-generated preference data.
3. **Policy Optimization:** The target model is fine-tuned via RL (e.g., PPO) using the learned reward.

Reinforcement Learning for Enhanced Reasoning

Previous Approaches to Improve LLM Question Answering

- **One-shot / Direct Answering**: The model receives a question and directly predicts the final answer label.
- **Neural-Symbolic Methods**: Combine neural networks with symbolic reasoning or logic-based rules.
- **Few-shot Prompting** (e.g., GPT-3): The model is guided by a small number of in-context examples.

Limitation: Direct Answering and **Few-shot prompting** methods primarily train the model to *match the correct answer*, but they do not explicitly encourage the model to learn reasoning processes, especially for **preference-based or multi-step questions**. Otherwise, **Neural Symbolic** is a complex method to combine with LLM.

Reinforcement Learning Enhancement Reasoning

Chain of Thought

- **Chain-of-Thought** [7][8] is a prompting and reasoning paradigm that encourages large language models to generate **intermediate reasoning steps** before producing a final answer.
- Instead of directly predicting an answer, the model decomposes a complex problem into a sequence of **logical or arithmetic steps**.
- CoT significantly improves performance on tasks requiring **multi-step reasoning**, such as mathematics, logical inference, and commonsense reasoning.
- CoT can be induced via **few-shot demonstrations** or explicit prompts (e.g., *"Let's think step by step"*).

Reinforcement Learning Enhancement Reasoning

Disadvantages of Chain-of-Thought (CoT)

- **Linear reasoning only**: CoT follows a single reasoning path, which makes it difficult to recover from early reasoning mistakes.
- **No backtracking or exploration**: Once a reasoning step is generated, the model cannot revise or explore alternative reasoning paths.
- **Lack of explicit planning**: CoT does not evaluate or compare multiple candidate reasoning trajectories before reaching a final answer.
- **Error propagation**: Incorrect intermediate steps can propagate and lead to an incorrect final answer, even if later steps could correct it.
- **No explicit reward or evaluation**: CoT does not include a mechanism to score or select reasoning paths based on long-term outcomes.

Motivation: These limitations motivate planning-based reasoning methods, such as **Tree-of-Thought (ToT)** and **MCTS-based reasoning**, which explicitly explore, evaluate, and select among multiple reasoning

Reinforcement Learning Enhancement Reasoning

Tree-of-Thought (ToT)

- **Tree-of-Thought (ToT)** [9]is a reasoning framework that extends Chain-of-Thought by exploring **multiple reasoning paths** in a tree structure instead of a single linear chain.
- Each node in the tree represents a **partial reasoning state**, and each edge corresponds to a candidate **next thought** generated by the language model.
- ToT performs **explicit search** over the reasoning tree using strategies such as **Breadth-First Search (BFS)** or **Depth-First Search (DFS)**.
- Candidate thoughts are evaluated using **heuristic scoring**, often based on the language model's self-evaluation or a simple verifier.
- By comparing multiple reasoning trajectories, ToT reduces the risk of early reasoning errors and improves performance on **complex multi-step reasoning tasks**.

References

[1] L. Engstrom et al., *Implementation Matters in Deep RL: A Case Study on PPO and TRPO*, ICLR, 2019.

[2] M. G. Azar et al., *A General Theoretical Paradigm to Understand Learning from Human Preferences* (IPO), AISTATS, 2024.

[3] K. Ethayarajh et al., *KTO: Model Alignment as Prospect Theoretic Optimization*, arXiv, 2024.

[4] Z. Shi et al., *Understanding Likelihood Over-optimisation in Direct Alignment Algorithms*, arXiv, 2024.

[5] R. Rafailov et al., *Scaling Laws for Reward Model Overoptimization in Direct Alignment Algorithms*, NeurIPS, 2024.

[6] R. Rafailov et al., *Direct Preference Optimization: Your Language Model is Secretly a Reward Model* (DPO), NeurIPS, 2023.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (2nd Ed.), MIT Press, 2018.

[8] E. Brunskill, *CS234: Reinforcement Learning*, Stanford University, 2025.

References

[9] Z. Shao et al., *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*, arXiv, 2024.

[10] L. Ouyang et al., *Training Language Models to Follow Instructions with Human Feedback* (InstructGPT), NeurIPS, 2022.

[11] H. Lee et al., *RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback*, arXiv, 2023.

[12] N. Lambert, *The RLHF Book: Policy Gradient Algorithms*, rlhfbook.com, 2024.

[13] Wikipedia Contributors, *Reinforcement learning from human feedback (Direct Alignment Algorithms)*, Wikipedia, 2025.

[14] NormalUhr, *RLHF Pipeline*, Hugging Face Blog, 2024.

[15] Unsloth AI, *Reinforcement Learning (RL) Guide*, Unsloth Documentation, 2025.

[16] Christiano et al., *Deep Reinforcement Learning from Human Preferences*, NeurIPS, 2017.

References

[17] Christiano et al., *Deep Reinforcement Learning from Human Preferences*, NeurIPS, 2017.

[18] Ouyang et al., *Training Language Models to Follow Instructions with Human Feedback*, NeurIPS, 2022.

[19] Bai et al., *Constitutional AI: Harmlessness from AI Feedback*, arXiv, 2022.

[20] Rafailov et al., *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*, NeurIPS, 2023.

[21] Lee et al., *RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback*, arXiv, 2023.

[22] Dubois et al., *AlpacaEval: An Automatic Evaluator of Instruction-following Models*, arXiv, 2024.

[23] Wei et al., *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*, NeurIPS, 2022.

[24] Wang et al., *Self-Consistency Improves Chain-of-Thought Reasoning in Language Models*, ICLR, 2023.

References

[25] Yao et al., *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*, arXiv:2305.10601, 2023.

[26] Long et al., *Large Language Models as Optimizers*, arXiv, 2023.

[27] Silver et al., *Mastering the Game of Go with Deep Neural Networks and Tree Search*, Nature, 2016.

[28] Baker et al., *Emergent Tool Use from Multi-Agent Interaction*, arXiv, 2023.

[29] OpenAI, *Training Language Models to Follow Instructions with Human Feedback*, arXiv, 2022.

[30] Schulman et al., *Proximal Policy Optimization Algorithms*, arXiv, 2017.

[31] B. Peng et al., *Instruction Tuning with GPT-4*, arXiv, 2023.

THANK YOU FOR WATCHING !!!