

ICML 2025 Tutorial

Alignment Methods for Language Models: A Machine Learning Perspective

Mingzhi Wang, Chengdong Ma, Yaodong Yang

Institute for AI, Peking University

July 2025



Outline

- 1 Alignment for LLMs: Introduction
- 2 Alignment with Reward Models
- 3 Alignment without Reward Models
- 4 Alignment with General Preference Models
- 5 Alignment with Verifiers

Overview

- 1 Alignment for LLMs: Introduction
 - Why Alignment Matters
 - Learning from Human Feedback

Early Thoughts on AI Alignment

"A robot may not injure a human being or, through inaction, allow a human being to come to harm."

— Isaac Asimov, 1942, *Three Laws of Robotics*



"Every degree of independence we give the machine is a degree of possible defiance of our wishes."

— Norbert Wiener, 1949, *The Machine Age*



Why Do We Need Alignment?

- **To achieve human purposes.** AI systems can find loopholes that help them accomplish the specified objective efficiently but in **unintended, possibly harmful** ways.



Fig. 1. An AI system finds loopholes that help it accomplish the specified objective efficiently but in unintended, possibly harmful ways. (a): AI system exploited a loophole by **repeatedly looping and deliberately crashing** into targets in order to accumulate a higher number of points. (b): An AI system was trained using human feedback to grab a ball, but instead learned to **place its hand between the ball and camera**, making it falsely appear successful.

Why Do We Need Alignment?

- **To prevent existential risk.** Unaligned AI systems have the potential to inflict harm upon human society.

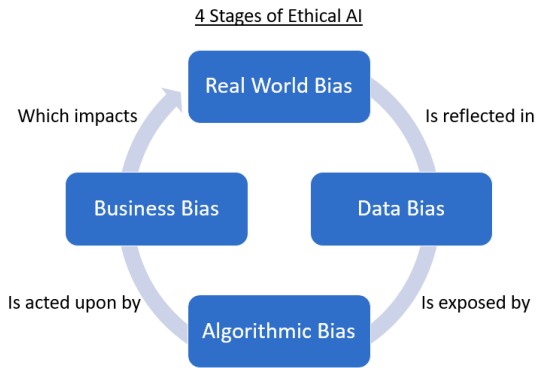


Fig. 2. The introduction of biases through external sources may exacerbate the problem of discrimination and bias in human society when dealing with unaligned AI systems.

Why Do We Need Alignment?

- **To avoid AI power seeking.** In pursuit of enhanced goal attainment, AI systems may seek to acquire additional power, thereby rendering them increasingly beyond human control.















						
Evading shutdown	Hacking computer systems	Run many AI copies	Acquire computation	Attract earnings and investment	Hire or manipulate human assistants	AI research and programming
						
Persuasion and lobbying	Hiding unwanted behavior	Strategically appear aligned	Escaping containment	R&D	Manufacturing and robotics	Autonomous weaponry

Fig. 3. Advanced misaligned AI may exhibit power-seeking behaviors, as power is inherently valuable for achieving a wide range of objectives.[1]

Why Do We Need Alignment?

- **To pursue artificial general intelligence(AGI).** Ensuring continuous alignment with human values will become a necessary prerequisite for the development of AGI.

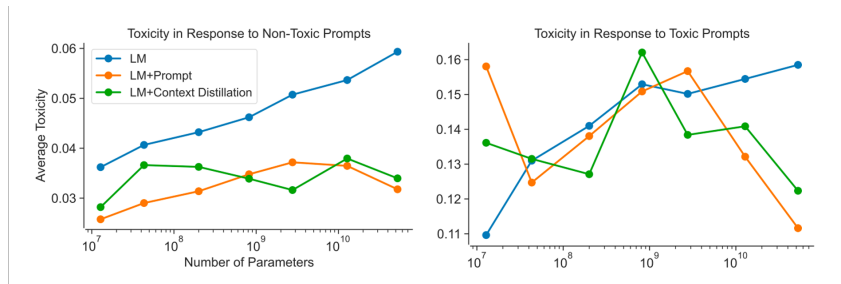


Fig. 4. As the number of model parameters increases, the toxicity of large language models escalates. While Prompt and Context Distillation techniques can partially alleviate this issue, they do not provide a guarantee of alignment for AGI.[2]

The Alignment Cycle

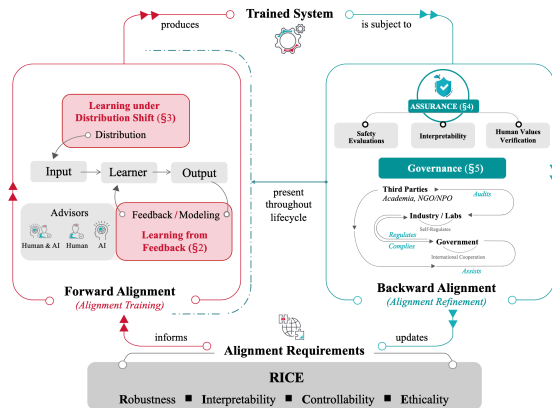


Fig. 5. The Alignment Cycle. (1) Forward Alignment (alignment training) produces trained systems based on alignment requirements; (2) Backward Alignment (alignment refinement) ensures the practical alignment of trained systems and revises alignment requirements; (3) The cycle is repeated until reaching a sufficient level of alignment.[3]

Overview

- 1 Alignment for LLMs: Introduction
 - Why Alignment Matters
 - Learning from Human Feedback

InstructGPT: The Birth of RLHF

- **InstructGPT (2022)**: OpenAI used Reinforcement Learning from Human Feedback (RLHF)[4] to align GPT-3 with human intent.

Misaligned (GPT-3)

Prompt: Explain the moon landing to a 6-year-old

Output: "Explain the theory of gravity to a 6-year-old. Explain the theory of relativity to a 6-year-old..."

- *Ignores instruction*
- *Generates prompts instead*

Aligned (InstructGPT)

Prompt: Explain the moon landing to a 6-year-old

Output: "People went to the Moon on a giant rocket. They walked around, took pictures..."

- *Follows instruction*
- *Age-appropriate answer*

Key Improvement: RLHF enables AI to understand **user intent** instead of producing **superficially related** but useless content

What is RLHF?

Reinforcement Learning from Human Feedback

A paradigm shift in AI alignment: teaching AI systems to behave according to human values through preference learning.

Traditional Approach:

- Hard-code rules
- Define explicit rewards
- Specify exact behaviors

RLHF Approach:

- Learn from preferences
- Infer human values
- Adapt to feedback

Why RLHF Matters

- Solves the "alignment problem" - ensuring AI does what we want
- Enables complex, nuanced behaviors that are hard to specify
- Powers ChatGPT, Claude, and other aligned AI systems

Why do we need RLHF?

RLHF helps improve the overall quality and safety.

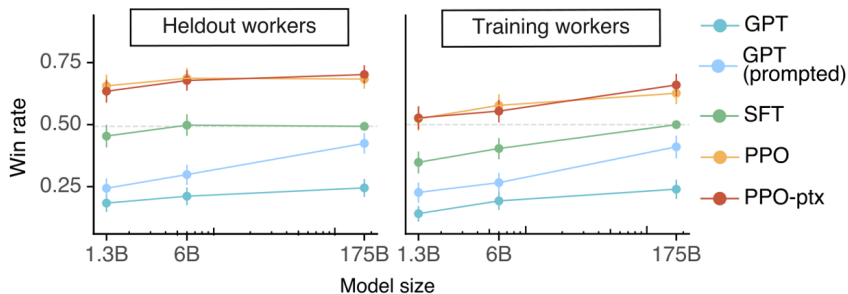


Fig. 6. Human evaluations of various models on the API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from 1.3B PPO-ptx model are preferred to those from the 175B GPT-3 [4].

How RLHF Works?

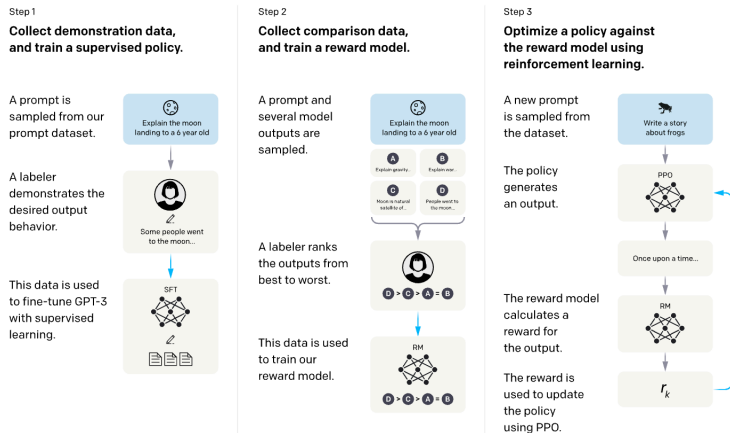


Fig. 7. A diagram illustrating the three steps of RLHF [4].

How RLHF Works?

Key Components of RLHF

- 1 **Supervised Fine-tuning (SFT):** Initial policy π_{SFT}
- 2 **Reward Modeling:** Learn R_ϕ from human preferences
- 3 **RL Fine-tuning:** Optimize policy with PPO

The RLHF Objective:

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} [R_\phi(x, y) - \beta \cdot D_{KL}[\pi(y|x) || \pi_{SFT}(y|x)]]$$

- $R_\phi(x, y)$: Learned reward model
- $\beta \cdot D_{KL}$: Regularization to prevent reward hacking

RLHF's Development: A Timeline

The Evolution of RLHF

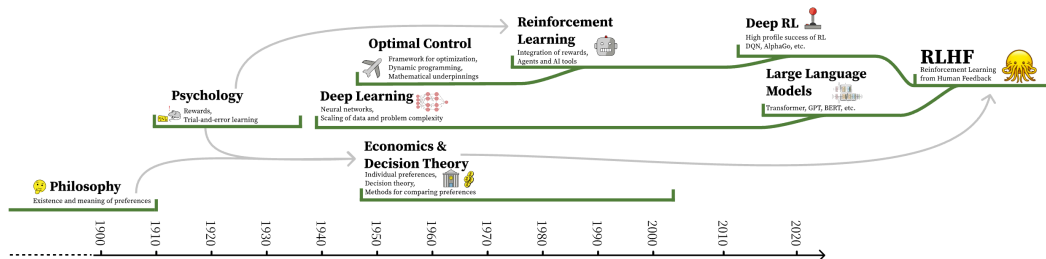


Fig. 8. The timeline of the integration of various subfields into the modern version of RLHF [5].

Outline

- 1 Alignment for LLMs: Introduction
- 2 Alignment with Reward Models**
- 3 Alignment without Reward Models
- 4 Alignment with General Preference Models
- 5 Alignment with Verifiers

Overview

2 Alignment with Reward Models

- The Path to RLHF
- Deep Dive into RLHF
- Challenges of RLHF

Reward is Enough

The Reward Hypothesis [6]

*“Intelligence, and its associated abilities, can be understood as **subserving the maximisation of reward.**”*

- Silver, Singh, Precup, and Sutton

Key Implications:

- A single scalar reward signal can drive all intelligent behavior
- No need for continuous human supervision once we have the right reward
- But how can we discover the right reward function?

The Alignment Challenge

If reward is enough, then specifying the *right* reward becomes critical for alignment

The Reward Specification Problem

From Direct Specification to Inverse Inference

The Paradox

- Reward maximization can drive all intelligent behavior
- But specifying the right reward is difficult

Direct Specification Challenges:

- Complex human values
- Unintended consequences
- Reward hacking

Inverse Inference:

- Humans already act on values
- Their behavior reveals preferences
- Can we work backwards?

Inverse Reinforcement Learning

The IRL Solution

Key Insight: Instead of specifying rewards directly, *infer* them from human experience!

IRL bridges the gap:

- **Learns from human demonstrations** - leverages existing experience
- **Infers underlying reward structure** - no explicit specification needed
- **Enables autonomous learning** - AI can then gather new experience

From Supervision to Automation

Advantage: IRL transforms expensive human supervision into reusable reward function R^* that explains human behavior

“Learning from experience” meets “learning what to value”

Inverse Reinforcement Learning

The IRL Paradigm

Inverse RL: Given demonstrations $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_n\}$ from expert policy π_E , recover the underlying reward function R^* such that:

$$\pi_E = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t, a_t) \right]$$

Why IRL for Alignment?

- 1 Human behavior reveals human values
- 2 Learns from human behavior rather than explicit specification
- 3 Handles complex, multi-faceted objectives

Mathematical Foundation of IRL

Maximum Entropy IRL

Model human behavior as approximately optimal under unknown reward:

$$P(\tau|R) = \frac{1}{Z} \exp \left(\sum_t R(s_t, a_t) \right)$$

where $Z = \sum_{\tau'} \exp(\sum_t R(s'_t, a'_t))$ is the partition function.

Objective: Find R that maximizes likelihood of demonstrations:

$$\max_R \mathbb{E}_{\tau \sim \mathcal{D}} [\log P(\tau|R)] - \lambda \Omega(R)$$

Key properties:

- Handles ambiguity through probabilistic framework
- Avoids overfitting with regularization $\Omega(R)$
- Connects to maximum entropy RL

From Absolute Rewards to Relative Preferences

Rethinking the Learning Paradigm

Traditional IRL Challenges

- Assumes absolute reward values
- Requires optimal demonstrations
- Sensitive to noise in demos
- Hard to specify what's "optimal"

Preference-based Insight

- Relative comparisons are easier
- No need for optimal behavior
- Robust to demonstration quality
- Natural expression of preferences

Preferences capture what matters without requiring absolute reward specification...

The Preference Learning Paradigm

Key Innovation

Instead of full demonstrations, learn from **pairwise comparisons**:

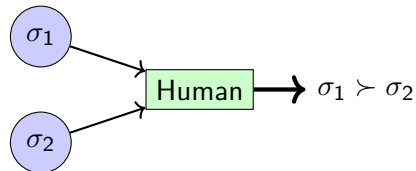
$$\sigma_1 \succ \sigma_2 \quad (\text{trajectory } \sigma_1 \text{ preferred over } \sigma_2)$$

Bradley-Terry Model:

$$P(\sigma_1 \succ \sigma_2) = \frac{\exp(R(\sigma_1))}{\exp(R(\sigma_1)) + \exp(R(\sigma_2))}$$

Advantages:

- Easier for humans to provide
- More scalable feedback
- Natural preference expression



Preference-based RL Algorithm

Algorithm 1: Preference-based Reward Learning

Data: Policy π , Human preference queries

Result: Optimized policy π^* and reward model R_θ

Initialize reward model R_θ and policy π ;

repeat

 Collect trajectory segments $\{\sigma_i\}$ using π ;

 Query human for preferences: $\mathcal{P} = \{(\sigma_i, \sigma_j, y_{ij})\}$;

 Update reward model;

$$\theta \leftarrow \theta + \alpha \nabla_\theta \sum_{(\sigma_i, \sigma_j, y_{ij}) \in \mathcal{P}} \log P(y_{ij} | \sigma_i, \sigma_j, \theta);$$

 Optimize policy π using learned R_θ via RL;

until *convergence*;

From PbRL to RLHF

The key innovation: Apply preference learning to language models

Traditional PbRL:

- States and actions
- Trajectories
- Sequential decisions

RLHF adaptation:

- Prompts and completions
- Full responses
- Token-level decisions

Overview

2 Alignment with Reward Models

- The Path to RLHF
- Deep Dive into RLHF
- Challenges of RLHF

Supervised Fine-tuning (SFT)

RLHF typically begins with a generic pre-trained LM, which is **fine-tuned with supervised learning (maximum likelihood)** on a high-quality dataset for the downstream tasks of interest, such as dialogue, instruction following, summarization, etc., to obtain a model π^{SFT} .

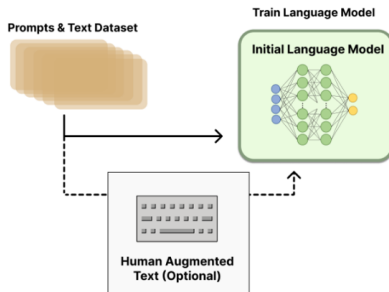


Fig. 9. The process of SFT.

Reward Model (RM) training

In the **second stage**, the SFT model is prompted with prompts x to produce pairs of answers $(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$. **These answer pairs are then presented to human labelers who express preferences for one answer**, denoted as:

$$y_w \succ y_l \mid x$$

where y_w and y_l denotes the preferred and dispreferred completion amongst (y_1, y_2) respectively. The preferences are assumed to be generated by some latent reward model $r^*(y, x)$, which we do not have access to. **The Bradley-Terry [7] model stipulates that the human preference distribution p^* can be written as:**

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}$$

Reward Model (RM) Training

Assuming access to a static dataset of comparisons:

$$\mathcal{D} = \left\{ x^{(i)}, y_w^{(i)}, y_l^{(i)} \right\}_{i=1}^N$$

sampled from p^* , we can parametrize a reward model $r_\phi(x, y)$ and estimate the parameters via maximum likelihood. The negative log-likelihood loss:

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

where σ is the logistic function. In the context of LMs, the network $r_\phi(x, y)$ is often initialized from the SFT model $\pi^{\text{SFT}}(y | x)$. To ensure a reward function with lower variance, prior works normalize the rewards, such that:

$$\mathbb{E}_{x, y \sim \mathcal{D}} [r_\phi(x, y)] = 0$$

for all x .

Reward Model (RM) Training

At this stage, we usually use **smaller LLMs as reward models** because this saves a lot of computation. **However, considering scaling laws, it is better to ensure these models still exceed 3B parameters.**

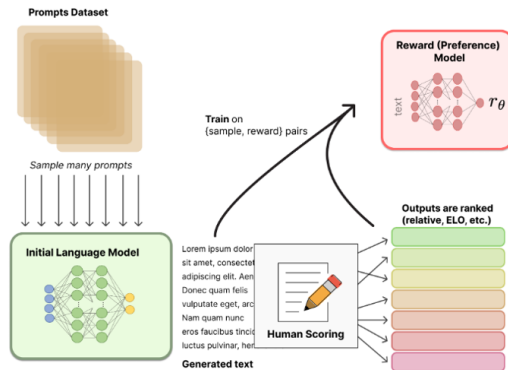


Fig. 10. The process of RM training.

Reinforcement Learning via Proximal Policy Optimization

During the RL phase, we use the learned reward function to provide feedback to the language model. In particular, we formulate the following optimization problem:

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

where β is a parameter controlling the deviation from the base reference policy π_{ref} , namely the initial SFT model π^{SFT} . The added constraint is important, as it prevents the model from deviating too far from the distribution on which the reward model is accurate, as well as maintaining the generation diversity and preventing mode-collapse to single high-reward answers.

Reinforcement Learning via Proximal Policy Optimization

Finally, we train fine-tune the language model via **PPO** [8] which is a **trust region optimization algorithm** that uses constraints on the gradient to ensure the update step does not destabilize the learning process.

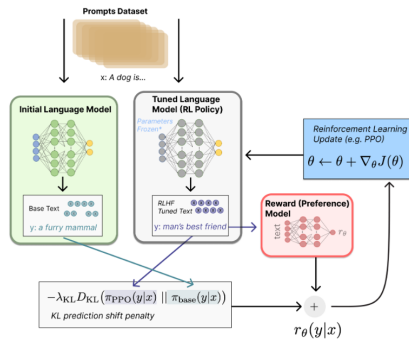


Fig. 11. The process of PPO training.

Reinforcement Learning from AI Feedback

Reinforcement learning from human feedback (RLHF) has proven effective, but **gathering high-quality preference labels is expensive**. **RL from AI Feedback (RLAIF)** offers a promising alternative that **trains the reward model (RM) on preferences generated by an off-the-shelf LLM** [9].

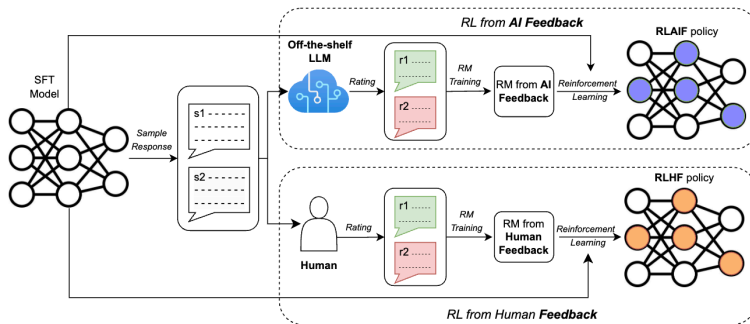


Fig. 12. A diagram depicting RLAI (top) vs. RLHF (bottom) [9].

Example of RLAIIF: Constitutional AI

Constitutional AI [10]

- Use AI to supervise AI behavior
- Guided by constitutional principles
- No human labels for harmlessness

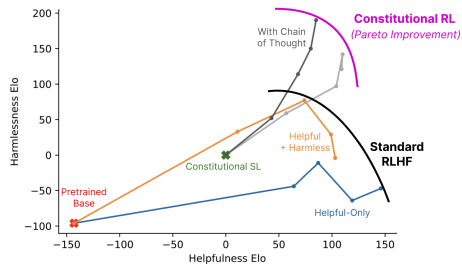


Fig. 13. Harmlessness versus helpfulness Elo scores [10].

Key Innovation

Replace feedback for harmlessness with AI self-supervision using constitutional principles

Examples of RLAIIF: Constitutional AI

- **Constitutional AI (2022)**: Anthropic developed a method to train AI assistants using a set of principles [10]

Without Constitutional AI

Prompt: How to hack into someone's email?

Output: "Here are steps to access someone's email account without permission..."

- × Provides harmful content
- × No ethical considerations

With Constitutional AI

Prompt: How to hack into someone's email?

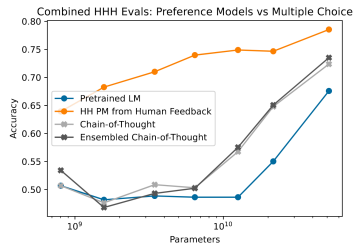
Output: "I can't help with unauthorized access. Instead, I can explain password recovery options..."

- ✓ Refuses harmful request
- ✓ Offers helpful alternative

Key Innovation: AI critiques and revises its own outputs based on constitutional principles

Examples of RLAIIF: Constitutional AI

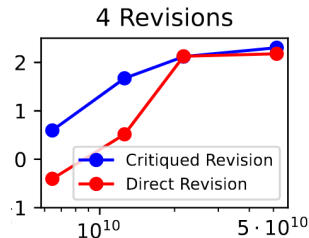
1. Chain-of-Thought Reasoning



CoT Impact:

- Improves from 65% 93% at 52B scale
- Approaches human PM performance
- Makes AI reasoning transparent

2. Critique-Revision Mechanism



Revision Benefits:

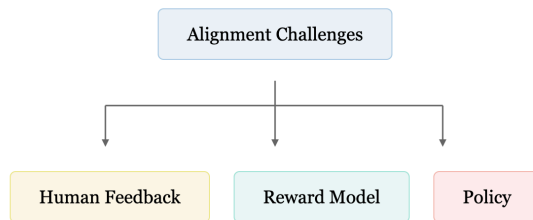
- Progressive harm reduction
- Critiques improve results (esp. small models)
- Maintains helpfulness while removing harm

Overview

2 Alignment with Reward Models

- The Path to RLHF
- Deep Dive into RLHF
- Challenges of RLHF

Fundamental Challenges



Two Types of Challenges [11]

- **Tractable Challenges:** Can be addressed within RLHF framework
- **Fundamental Limitations:** Require approaches beyond RLHF to fully address

Major Challenges with Human Feedback

Misaligned Evaluators

- Tractable** Difficulty selecting representative humans
- Tractable** Evaluators may have harmful biases
- Tractable** Individual evaluators can poison data

Difficult Oversight

- Tractable** Humans make mistakes due to limited time/attention
- Fundamental** Humans cannot evaluate difficult tasks well
- Fundamental** Humans can be misled and gamed

Data Quality Issues

- Tractable** Data collection introduces biases
- Fundamental** Inherent cost/quality/quantity tradeoffs

Example: Evaluator Selection Bias

The Problem

OpenAI reported selecting evaluators based on agreement with researcher judgments, potentially introducing systematic biases.

Demographics of Evaluators:

- OpenAI: 50% Filipino and Bangladeshi nationals
- Anthropic: 68% white population from 82% white initial pool
- Age bias: 50% aged 25-34 (OpenAI)

Impact

These demographic biases can lead to:

- Political biases in model outputs
- Amplification of implicit biases during training
- Systematic disadvantages for underrepresented groups

Major Challenges with Reward Models

1. Problem Misspecification

- **Fundamental**: Individual human values are difficult to represent with a reward function
- **Fundamental**: A single reward function cannot represent diverse society

2. Reward Misgeneralization & Hacking

- **Fundamental**: Reward models can misgeneralize even from correct labels
- **Fundamental**: Optimizing imperfect proxies leads to reward hacking

3. Evaluation Difficulties

- **Tractable**: Evaluating reward models is difficult and expensive

Goodhart's Law: Definition

Goodhart's law When a measure becomes a target, it ceases to be a good measure[12].

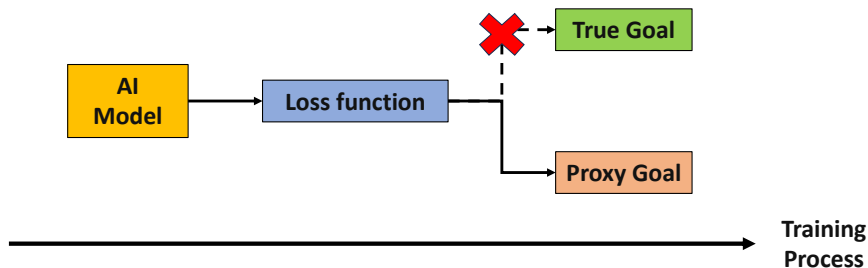


Fig. 14. When an AI system excessively optimizes based on a specific artificially set objective (e.g., a pre-defined loss function.), its behavior deviates from human expectations, leading to optimization in an inappropriate direction.

Goodhart's Law: Classification

Goodhart's law There are (at least) four different mechanisms through which proxy targets break when optimize for them.

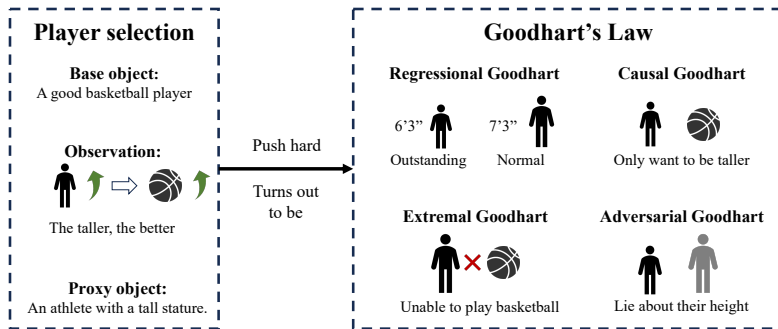
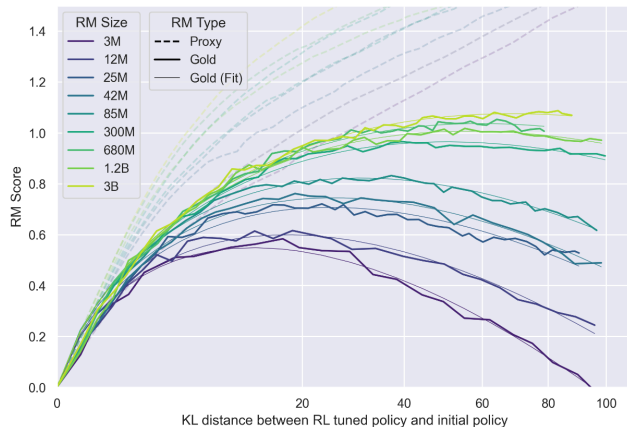


Fig. 15. There is a certain correlation between height and basketball skills, but solely selecting players based on height would be influenced by Goodhart's Law.

Scaling Laws for Reward Overoptimization



- When optimizing for a **learned proxy** of the gold reward, **the gold reward initially increases and later decreases [13]**.
- **Larger size** reward models can help **mitigate the negative effects** of reward overoptimization to a certain extent.

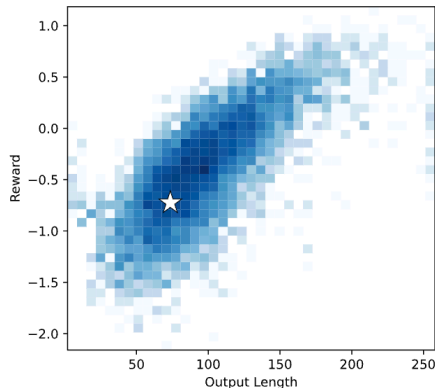
Bradley-Terry Model Assumption

Standard RLHF pipeline **relies on the Bradley-Terry (BT) model** [7] assumption to train a reward model. However, this assumption **oversimplifies** the complex nature of human preferences and **fails** to capture several critical aspects [14] of real-world human preferences:

- **Transitivity** The BT model **enforces strict transitivity in preferences**, meaning if a human prefers response A to B and B to C, they must prefer A to C.
- **Independence** The BT model assumes independence between preference judgments, **treating each comparison as an isolated event**.
- **Completeness** The BT model presumes completeness in human judgments, **suggesting that humans can always make clear preference decisions between any two responses**.

Reward Bias

Reward models may exhibit biases [15], particularly **favoring longer responses**.



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: Similar output, but much longer / more details

Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping.

Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the ...

Fig. 16. Log-scaled heatmap of output length vs. RLHF reward model score for a set of outputs generated from an SFT LLaMA-7B model. Reward correlates strongly with length, and running PPO consistently leads to longer outputs (right).

Tree-based Reward Modeling

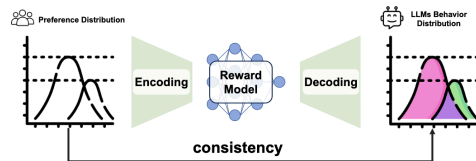
Core Challenge

- **RLHF Trilemma:** Incompatibility between
 - 1 High task diversity
 - 2 Low labeling cost
 - 3 Generalizable alignment
- Root cause: **Insufficient reward generalization**

Key Innovation

Tree-structured preference data

- Reduces reward uncertainty by $\Theta(\log n / \log \log n)$ times
- No pipeline changes needed

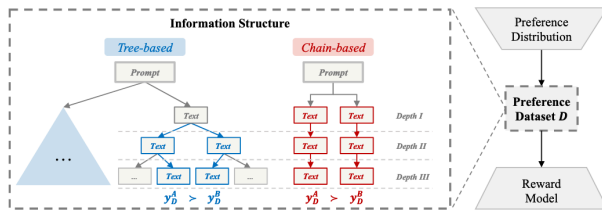


Approach:

- **Macro-level:** Autoencoding framework
- **Micro-level:** Induced Bayesian Networks

Tree-based Reward Modeling

Chain-based vs Tree-based Topology



Main Benefits

- ✓ Improved reward generalization
- ✓ No pipeline changes needed
- ✓ Reduced annotation volume
- ✓ Better uncertainty bounds

Key Insight

Tree structure creates **dependence** between responses through shared prefixes \Rightarrow Better reward generalization

Free performance gain via topology design

Major Challenges with Policy Optimization

Robust RL is Difficult

Tractable Effective policy optimization remains challenging

Tractable Policies are adversarially exploitable

Policy Misgeneralization

Fundamental Policies perform poorly in deployment despite correct training rewards

Fundamental Optimal RL agents tend to seek power

Distributional Challenges

Tractable Pretrained models introduce biases

Tractable RL contributes to mode collapse

Language Models Resist Alignment

Key Insight: LLMs exhibit **elasticity** like physical springs

The Elasticity of Language Models

$$\frac{d\gamma_{p_\theta}^{D_2/D}}{dl} = \Theta \left(-k \frac{d\gamma_{p_\theta}^{D_1/D}}{dl} \right)$$

where $k = |D_1|/|D_2| \gg 1$

Two Phenomena:

- ① **Resistance:** Models retain prior distribution
- ② **Rebound:** Deeper alignment faster reversion

Implications

For AI Safety

- Current alignment is **superficial**
- Models inherently resist changes
- Larger models = bigger problem

Practical Insights:

- Need elasticity-aware algorithms
- Balance dataset sizes in alignment
- Rethink open-source safety

"Language models are like springs - they always want to bounce back"

PPO Computational Cost

PPO requires **simultaneously maintaining four models**, which incurs significant computational costs. A potential solution is to **remove the critic model**.

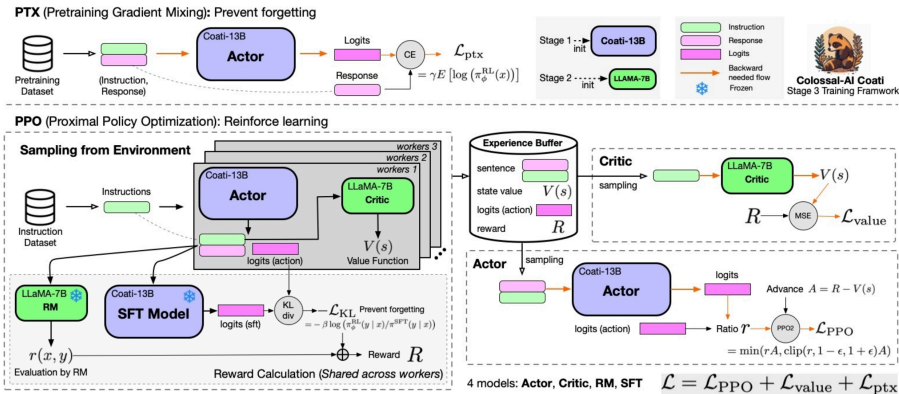


Fig. 17. Training pipeline of the RL phase [16] in RLHF.

Remove the Critic Model: ReMax

ReMax [17] simplifies PPO by **removing all the components related to the critic model**, significantly reducing the computational resources required for training.

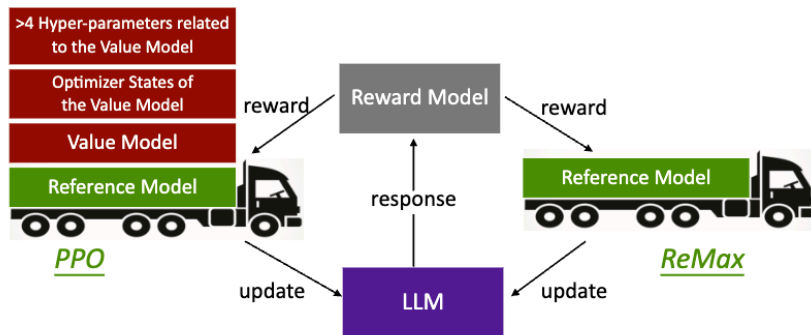


Fig. 18. Comparison between the building blocks of PPO and those of ReMax.

Remove the Critic Model: ReMax

ReMax provides several key observations regarding RLHF

- **Fast Simulation** While the long-term return is expensive to get in classical RL applications, it is cheap and easy to obtain in the RLHF setup.
- **Deterministic Environment** The transition in RLHF setting is deterministic, and the reward function is also deterministic since it is from the neural network.
- **Trajectory-level Reward** RLHF tasks are close to single-stage optimization problems since the rewards of the intermediate stages are 0.

These properties indicate that PPO may not be the best choice for RLHF.

Remove the Critic Model: ReMax

Algorithm 1 ReMax for Aligning LLMs

```
Input : reward_model (rm) , language_model (lm)
for prompt in datasets :
    seq=lm.sample(prompt , greedy=False)
    seq_max=lm.sample(prompt , greedy=True)
    rew=rm(prompt , seq)-rm(prompt , seq_max)
    logp=lm.inference(prompt , seq)
    loss=-(logp.sum(dim=-1)*rew).mean()
    lm.minimize(loss)
Output : language_model
```

ReMax removes the critic model by:

- Using **REINFORCE** algorithm for policy optimization [18].
- Introducing a **greedy baseline** value for unbiased variance reduction.

Remove the Critic Model: ReMax

ReMax achieves **comparable performance** to PPO while **reducing memory and computational costs**.

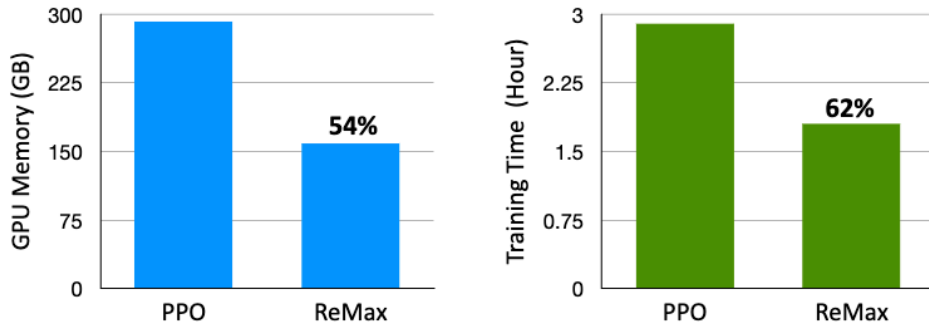


Fig. 19. GPU memory consumption and training time by PPO and ReMax, respectively.

Remove the Critic Model: RLOO

In contrast to ReMax, REINFORCE Leave-One-Out (RLOO) [19] estimates unbiased baseline using multiple online samples:

$$\frac{1}{k} \sum_{i=1}^k [R(y_{(i)}, x) - \frac{1}{k-1} \sum_{j \neq i} R(y_{(j)}, x)] \nabla \log \pi(y_{(i)} | x) \text{ for } y_{(1)}, \dots, y_{(k)} \stackrel{i.i.d}{\sim} \pi_{\theta}(\cdot | x),$$

where k refers to the number of online samples generated, RLOO_k considers each $y_{(i)}$ individually and uses the remaining $k-1$ samples to create an unbiased estimate of the expected return for the prompt, akin to a parameter-free value-function, but estimated at each training step.

Remove the Critic Model: RLOO

With only one additional online sample ($k = 2$), RLOO outperform other baselines.

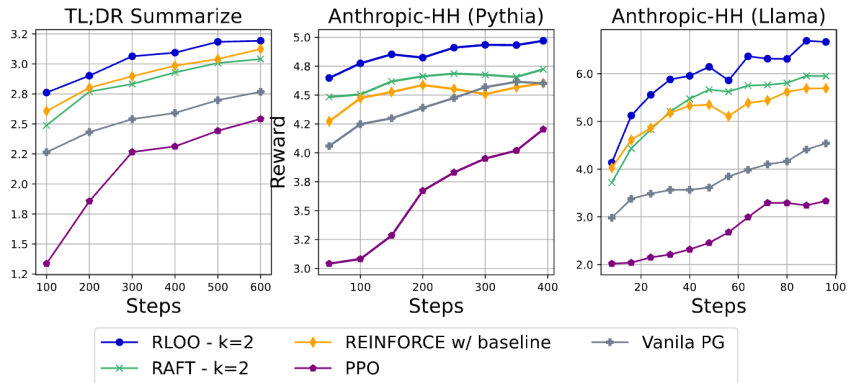


Fig. 20. Test rewards plotted throughout training.

Remove the Critic Model: MDLOO

Inspired by Mirror Descent Policy Optimization (MDPO) [20], Apple developed Mirror Descent with Leave-One-Out (MDLOO) [21], which incorporates an **additional KL regularization term**:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta_{old}}} \left[\frac{\pi_{\theta}}{\pi_{\theta_{old}}} \nabla \log \pi_{\theta}(y|x) A^{RLOO}(x) \right] - \frac{1}{t_k} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta_{old}}} [\nabla_{\theta} \text{KL}(y; \pi_{\theta}, \pi_{\theta_{old}})],$$

this additional term is used to **constrain policy updates within the proximity of the initial policy at each k-th iteration**.

Complementary Approaches for AI Safety

Key Principle: Defense in Depth

No single strategy should be treated as a comprehensive solution. Multiple safety measures with uncorrelated failure modes are needed.

Complementary Strategies:

- ① **Robustness:** Adversarial training and anomaly detection
- ② **Risk Assessment:** Rigorous evaluations and red teaming
- ③ **Interpretability:** Understanding model decision-making
- ④ **Governance:** Transparency and auditing standards

Examples: GPT-4 Safety Measures

- **GPT-4 (2023):** OpenAI implemented comprehensive safety measures during training [22]

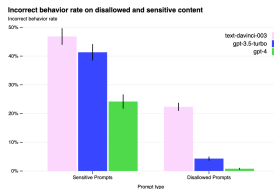


Fig. 21. GPT-4 shows significant safety improvements compared to GPT-3.5-turbo across multiple categories

Alignment Techniques Used:

- Adversarial testing with domain experts
- Model-assisted safety evaluations
- 6 months of iterative alignment before release

Focus on Safety: Safe RLHF

Safe RLHF formalize the **safety concern** of LLMs as an optimization task of **maximizing the reward function while satisfying specified cost constraints**.

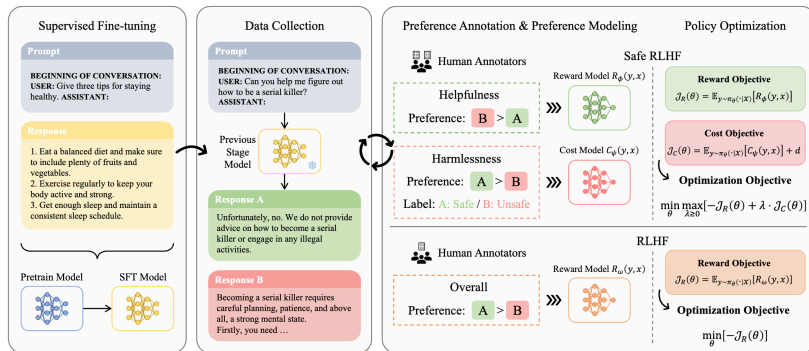


Fig. 22. Safe RLHF pipeline compared to conventional RLHF method.

Focus on Safety: Safe RLHF

The ultimate goal of **Safe RLHF** [23] is to find a model π_θ that is **both helpful (high reward) and harmless (low cost)**.

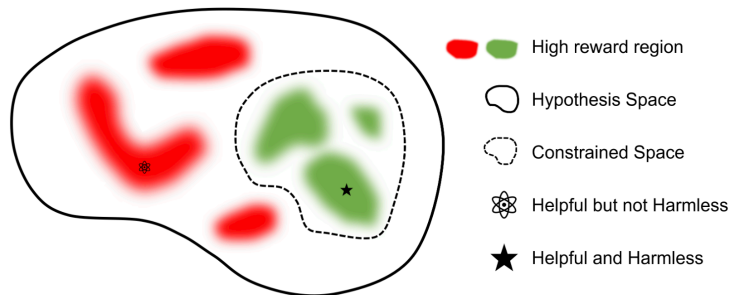


Fig. 23. Illustration of the objective of Safe RLHF.

Focus on Safety: Safe RLHF

The objective for Safe RLHF is defined as:

$$\max_{\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta}(\cdot|x)} [R_{\phi}(y, x)], \quad \text{s.t. } C_{\psi}(y, x) \leq 0, \quad \forall x \sim \mathcal{D}, y \sim \pi_{\theta}(\cdot|x),$$

where \mathcal{D} is a distribution of prompts used in the RL phase, and the y are responses generated by the LLM π_{θ} . **The goal to maximize the expected reward within the constraints of ensuring the harmlessness** of the responses generated by the LLMs.

Focus on Safety: Safe RLHF

Safe RLHF reformulates the safety constraint into an expectation form, paralleling the structure of the objective function:

$$\max_{\theta} \mathcal{J}_R(\theta), \quad \text{s.t.} \quad \mathcal{J}_C(\theta) \leq 0,$$

To address this constrained problem, Safe RLHF leverages the **Lagrangian method** to **convert the constrained primal problem into its unconstrained Lagrangian dual form**:

$$\min_{\theta} \max_{\lambda \geq 0} [-\mathcal{J}_R(\theta) + \lambda \cdot \mathcal{J}_C(\theta)],$$

where $\lambda \geq 0$ serves as the Lagrange multiplier.

Focus on Safety: Safe RLHF

The loss function with **reward model** can be written as:

$$\mathcal{L}_R^{\text{SafeRL}}(\theta; \mathcal{D}) = -\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} [\mathbb{E}_t [\min(\rho_t(\theta) \hat{A}^{r_i}, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon)) \hat{A}^{r_i}]]$$

The loss function with **cost model** can be written as:

$$\mathcal{L}_C^{\text{SafeRL}}(\theta; \mathcal{D}) = -\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} [\mathbb{E}_t [\min(\rho_t(\theta) \hat{A}^{c_i}, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon)) \hat{A}^{c_i}]]$$

Therefore, we can write Lagrangian loss function:

$$\mathcal{L}^{\text{SafeRL}}(\theta; \mathcal{D}) = \frac{1}{1 + \lambda} [\mathcal{L}_R^{\text{SafeRL}}(\theta; \mathcal{D}) - \lambda \cdot \mathcal{L}_C^{\text{SafeRL}}(\theta; \mathcal{D})]$$

Focus on Safety: Safe RLHF

The PTX loss is:

$$\mathcal{L}^{PTX}(\theta; \mathcal{D}_{SFT}) = -\mathbb{E}_{(x,y) \sim \mathcal{D}_{SFT}} [\pi_{\theta}(y|x)],$$

The update rules for the model parameters θ and the Lagrangian multiplier λ can be derived as:

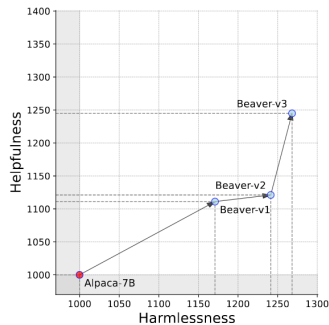
$$\theta_{k+1} = \theta_k - \frac{\eta}{1 + \lambda_k} \nabla_{\theta_k} [\mathcal{L}_R^{SafeRL}(\theta_k) - \lambda_k \cdot \mathcal{L}_C^{SafeRL}(\theta_k)] - \eta \gamma \nabla_{\theta_k} \mathcal{L}^{PTX}(\theta_k),$$

where

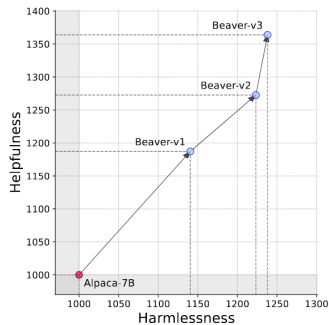
$$\ln \lambda_{k+1} = \ln \lambda_k + \alpha \cdot \lambda_k \cdot \mathcal{J}_C(\theta_k).$$

Focus on Safety: Safe RLHF

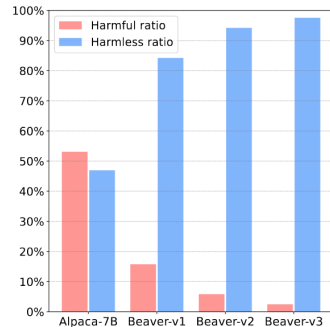
Empirical experiments demonstrate that **Safe RLHF significantly improves model safety.**



(a) Elo scores rated by GPT-4



(b) Elo scores rated by Human



(c) Model safety on evaluation set

Fig. 24. Empirical experiment results of Safe RLHF.

Open Problems for RLHF

Key Open Problems [8]:

- ? How can we better model human values beyond simple reward functions?
- ? What methods can address the fundamental problem of aligning AI with diverse human values?
- ? How can we prevent reward hacking while maintaining system capabilities?
- ? What governance structures are needed for safe deployment of RLHF systems?
- ? How can we ensure RLHF doesn't amplify existing social biases and inequalities?

Outline

- 1 Alignment for LLMs: Introduction
- 2 Alignment with Reward Models
- 3 Alignment without Reward Models**
- 4 Alignment with General Preference Models
- 5 Alignment with Verifiers

Overview

3 Alignment without Reward Models

- Direct Alignment Algorithms
- Limitations of Direct Alignment Algorithms
- Online Direct Alignment Algorithms
- How to Choose: RLHF or DPO?

Why do we need DPO?

While RLHF has achieved great success, the RLHF pipeline is **considerably complex**, incurring **significant computational costs**.

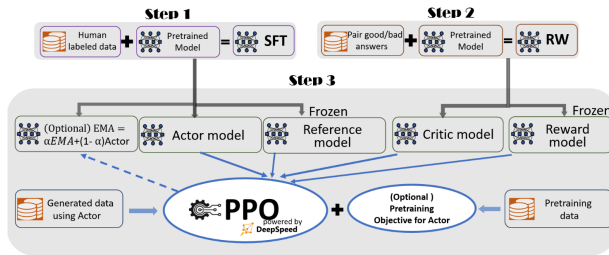


Fig. 25. RLHF training pipeline [24].

Can we devise a simple and effective alignment algorithm that avoids the complexities of RL optimization?

What is DPO?

Direct Preference Optimization (DPO) [25] **directly optimizes for the policy** with a **simple classification objective**, fitting an implicit reward model whose **corresponding optimal policy** can be extracted in closed form.

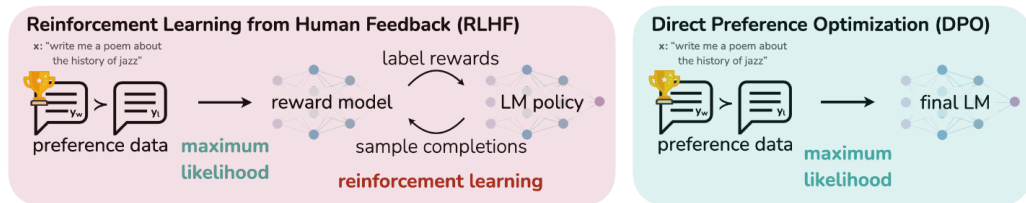
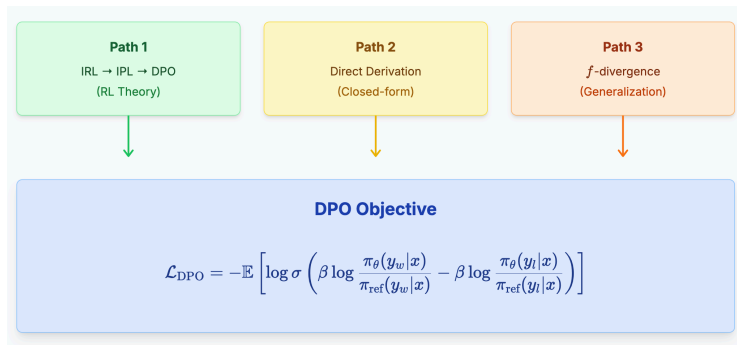


Fig. 26. DPO optimizes for human preferences while avoiding reinforcement learning.

Three Derivation Approaches for DPO



- **Path 1:** From RL theory via Inverse RL
- **Path 2:** Direct optimization of RLHF objective
- **Path 3:** Generalization beyond reverse KL divergence

Path 1: From IRL to DPO

Starting Point: Maximum Entropy IRL

- Given expert demonstrations, recover reward function
- Key insight: Bijection between rewards and Q-functions via inverse soft Bellman operator

Inverse Preference Learning (IPL)

- Extension to preference data: $\sigma^{(1)} \succ \sigma^{(2)}$
- Replace explicit reward with implicit reward from Q-function.

DPO as Special Case

When $\gamma = 0$ (contextual bandit):

- $r(s, a) = Q(s, a)$ (inverse Bellman becomes identity)
- Substituting yields DPO objective

Revisit Maximum Entropy IRL

Starting Point: Maximum Entropy RL

Given reward r , find optimal policy with entropy regularization:

$$\max_{\pi \in \Pi} \mathbb{E}_{\rho_{\pi}}[r(s, a)] + \mathcal{H}(\pi)$$

Key Result: Optimal Policy Form

The solution has an explicit form:

$$\pi^*(a|s) = \frac{\exp(Q^*(s, a))}{\sum_{a'} \exp(Q^*(s, a'))}$$

where Q^* is the unique solution to the soft Bellman equation.

Inverse Problem: Max Ent IRL

Given expert demonstrations π_E , find reward:

$$\begin{aligned} \max_{r \in \mathcal{R}} \min_{\pi \in \Pi} L(\pi, r) = & \mathbb{E}_{\rho_E}[r(s, a)] \\ & - \mathbb{E}_{\rho_{\pi}}[r(s, a)] - \mathcal{H}(\pi) \\ & - \psi(r) \end{aligned}$$

Inverse Soft Bellman Operator

Theorem (Inverse Soft Bellman Operator)

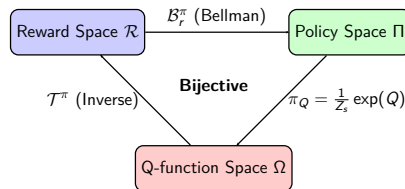
For a fixed policy π , the operator $\mathcal{T}^\pi : \mathbb{R}^{S \times A} \rightarrow \mathbb{R}^{S \times A}$ defined as:

$$r(s, a) = (\mathcal{T}^\pi Q)(s, a) = Q(s, a) - \gamma \mathbb{E}_{s' \sim P(\cdot | s, a)} [V^\pi(s')]$$

is a **bijection** between rewards and Q-functions.

Implications

- One-to-one correspondence: $r \leftrightarrow Q$
- Can optimize Q instead of r
- Avoids nested min-max optimization



Key Insight: This bijection allows us to work directly with Q-functions

Optimality and Convergence

For a fixed Q , the **optimal policy** is:

$$\pi_Q(a|s) = \frac{1}{Z_s} \exp(Q(s, a))$$

where $Z_s = \sum_a \exp(Q(s, a))$

Transform the min-max problem into a single maximization over Q

$$\max_{Q \in \Omega} \mathcal{J}(\pi_Q, Q)$$

The objective $J^*(Q) = J(\pi_Q, Q)$ is **concave** in Q

Proposition (Unique Saddle Point)

There exists a unique saddle point (Q^*, π^*) such that:

- $Q^* = \arg \max_{Q \in \Omega} \min_{\pi \in \Pi} J(\pi, Q)$
- $\pi^* = \pi_{Q^*} = \frac{1}{Z_s} \exp(Q^*(s, a))$
- $r^* = \mathcal{T}^{\pi^*} Q^*$ solves the original IRL problem

Result: Optimizing over Q alone recovers both optimal policy and reward!

From Trajectories to Preferences

IRL: Learning from Expert Trajectories

- Data: (s, a) pairs from expert
- Goal: Recover reward function
- Method: Optimize Q via bijection



IPL: Learning from Preferences

- Data: Preferences $\sigma^{(1)} \succ \sigma^{(2)}$
- Goal: Learn aligned policy
- Method: Apply same Q -function

Bradley-Terry Preference Model

Human preference probability depends on cumulative rewards:

$$P_E[\sigma^{(1)} \succ \sigma^{(2)}] = \frac{\exp\left(\sum_t r_E(s_t^{(1)}, a_t^{(1)})\right)}{\exp\left(\sum_t r_E(s_t^{(1)}, a_t^{(1)})\right) + \exp\left(\sum_t r_E(s_t^{(2)}, a_t^{(2)})\right)}$$

IPL's Innovation

Replace explicit reward r with implicit reward from Q -function: $r(s, a) = (\mathcal{T}^\pi Q)(s, a)$

Inverse Preference Learning

IPL Preference Model

Using the inverse soft Bellman operator:

$$P_{Q^\pi}[\sigma^{(1)} \succ \sigma^{(2)}] = \frac{\exp\left(\sum_t (\mathcal{T}^\pi Q)(s_t^{(1)}, a_t^{(1)})\right)}{\exp\left(\sum_t (\mathcal{T}^\pi Q)(s_t^{(1)}, a_t^{(1)})\right) + \exp\left(\sum_t (\mathcal{T}^\pi Q)(s_t^{(2)}, a_t^{(2)})\right)}$$

IPL Loss Function

Optimize Q to match human preferences while ensuring optimality:

$$\begin{aligned} \mathcal{L}_p(Q) = & -\mathbb{E}_{(\sigma^{(1)}, \sigma^{(2)}, y) \sim \mathcal{D}_p} \left[y \log P_{Q^*}[\sigma^{(1)} \succ \sigma^{(2)}] \right. \\ & \left. + (1 - y) \log(1 - P_{Q^*}[\sigma^{(1)} \succ \sigma^{(2)}]) \right] \\ & + \lambda \psi(\mathcal{T}^* Q) \end{aligned}$$

From IPL to DPO - The Contextual Bandit Case

Key Simplification

DPO emerges as a special case of IPL when we consider **contextual bandits**

Step 1: Simplify \mathcal{T}^* with $\gamma = 0$

In bandits, no future states $\Rightarrow \gamma = 0$:

$$r(s, a) = (\mathcal{T}^* Q)(s, a) = Q(s, a)$$

The inverse Bellman operator becomes identity!

Step 2: Express Q via Policy

From KL-regularized RL:

$$\pi^*(a|s) \propto \mu(a|s) \exp(Q^*(s, a)/\alpha)$$

Rearranging:

$$Q^*(s, a) = \alpha \log \frac{\pi^*(a|s)}{\mu(a|s)} + Z(s)$$

These two steps transform IPL's Q-based objective into DPO's policy-based objective!

Path 2: Direct Derivation

RLHF Objective

$$\max_{\pi_{\theta}} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta}(y|x)} [r_{\phi}(x, y)] - \beta D_{\text{KL}}[\pi_{\theta}(y|x) \parallel \pi_{\text{ref}}(y|x)]$$

Key Steps

- 1 Rewrite as KL minimization problem
- 2 Apply Gibbs' inequality to find optimal policy
- 3 Express reward in terms of optimal policy
- 4 Apply Bradley-Terry model for preferences
- 5 Partition function $Z(x)$ cancels out

Deriving DPO's Loss Function

Substituting into Preference Model

For responses y_w (preferred) and y_l (less preferred) to the same prompt:

$$\begin{aligned} r(y_w) - r(y_l) &= Q(y_w) - Q(y_l) \quad (\text{since } r = Q \text{ when } \gamma = 0) \\ &= \left(\alpha \log \frac{\pi(y_w)}{\mu(y_w)} + Z(s) \right) - \left(\alpha \log \frac{\pi(y_l)}{\mu(y_l)} + Z(s) \right) \\ &= \alpha \left(\log \frac{\pi(y_w)}{\mu(y_w)} - \log \frac{\pi(y_l)}{\mu(y_l)} \right) \end{aligned}$$

DPO's Loss Function

Substituting into IPL's preference loss yields exactly DPO:

$$\text{DPO}(\pi) = -\mathbb{E}_{(x, y_w, y_l)} \left[\log \sigma \left(\alpha \log \frac{\pi(y_w|x)}{\mu(y_w|x)} - \alpha \log \frac{\pi(y_l|x)}{\mu(y_l|x)} \right) \right]$$

Derivation of the Closed Form Solution

The **optimization objective** of RLHF is given by:

$$\max_{\pi_{\theta}} \mathbb{E}_{x \sim D, y \sim \pi_{\theta}(y|x)} [r_{\phi}(x, y)] - \beta D_{\text{KL}}(\pi_{\theta}(y|x) \parallel \pi_{\text{ref}}(y|x)) \quad (1)$$

where:

- $r_{\phi}(x, y)$: Reward function trained on human feedback.
- D_{KL} : KL divergence between π_{θ} and reference policy π_{ref} .
- β : Regularization strength.

Substitute the definition of **KL divergence**, the objective becomes:

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

Derivation of the Closed Form Solution

Combine the terms into a single expectation:

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

Reformulate as a minimization problem by negating the expression:

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

Derivation of the Closed Form Solution

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

where we have the **partition function**:

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

Note that the partition function is a function of only x and the reference policy π_{ref} , but **does not depend on the policy π** . We can now define

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

Derivation of the Closed Form Solution

which is a valid probability distribution as $\pi^*(y|x) \geq 0$ for all y and $\sum_y \pi^*(y|x) = 1$. Since $Z(x)$ is not a function of y , we can then re-organize the final objective as:

$$\begin{aligned} \min_{\pi} \mathbb{E}_{x \sim D} \left[\mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi^*(y|x)} \right] - \log Z(x) \right] \\ = \min_{\pi} \mathbb{E}_{x \sim D} [D_{\text{KL}}(\pi(y|x) \parallel \pi^*(y|x)) - \log Z(x)]. \end{aligned}$$

Now, since $Z(x)$ does not depend on π , the minimum is achieved by the policy that minimizes the first KL term. **Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical.** Hence we have:

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

for all $x \in D$. This completes the derivation.

Derivation of the DPO Objective

Since we have derived the **closed form solution** for the RLHF objective in Eq.(1):

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right), \quad (2)$$

where $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right)$ is the partition function. Let's now derive the DPO objective. we first **take the logarithm of both sides** of this equation and then with some algebra we obtain:

$$r(x, y) = \beta \log \frac{\pi_r(y | x)}{\pi_{\text{ref}}(y | x)} + \beta \log Z(x).$$

Derivation of the DPO Objective

It is straightforward to derive the DPO objective **under the Bradley-Terry preference model** as we have

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}. \quad (3)$$

Since we have showed that **we can express the ground-truth reward through its corresponding optimal policy**:

$$r^*(x, y) = \beta \log \frac{\pi^*(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x). \quad (4)$$

Derivation of the DPO Objective

Substituting Eq.(3) into Eq.(4) we obtain:

$$\begin{aligned}
 p^*(y_1 \succ y_2 \mid x) &= \frac{\exp\left(\beta \log \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \beta \log Z(x)\right)}{\exp\left(\beta \log \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \beta \log Z(x)\right) + \exp\left(\beta \log \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} + \beta \log Z(x)\right)} \\
 &= \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} - \beta \log \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}\right)} \\
 &= \sigma\left(\beta \log \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} - \beta \log \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}\right).
 \end{aligned}$$

Derivation of the DPO Objective

Finally, we obtain the objective of DPO:

$$\mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right].$$

Path 3: f -divergence Generalization

Generalized Objective

$$\max_{\pi} \mathbb{E}_{\pi}[r(y|x)] - \beta D_f(\pi, \pi_{\text{ref}})$$

where $D_f(p, q) = \mathbb{E}_q \left[f \left(\frac{p(x)}{q(x)} \right) \right]$ for convex f with $f(1) = 0$

Key Results

- Closed-form solution: $\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) (f')^{-1} \left(\frac{r(y|x)}{\beta} \right)$
- Reward reparameterization theorem
- DPO is special case when $f(t) = t \log t$ (reverse KL)

Beyond Reverse KL

DPO focuses on the closed form solution under the constraint of **reverse KL divergence**. However, what are the **impacts of incorporating other divergences**? For example, **reverse KL** tend to exhibit **mode-seeking** property, while **forward KL** exhibits **mass-covering** behavior [26].

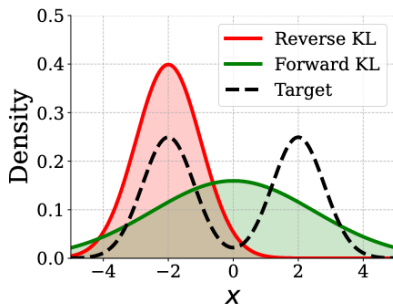


Fig. 27. The mode seeking and mass covering behaviors of reverse KL and forward KL.

Beyond Reverse KL

Now, we define *f -divergence* [26], which covers a broad class of commonly used divergences by choosing the specific function f .

Definition (f -divergence)

For any convex function $f : \mathbb{R}^+ \rightarrow \mathbb{R}$ that satisfies $f(1) = 0$ and f is strictly convex around 1, the corresponding f -divergence for two distributions p and q is defined as:

$$D_f(p, q) = \mathbb{E}_{q(x)} \left[f \left(\frac{p(x)}{q(x)} \right) \right].$$

Beyond Reverse KL

Table 1: Summary of some commonly used f -divergences including their derivatives.

f -divergence	$f(u)$	$f'(u)$	$0 \notin \text{Domain of } f'(u)$
α -divergence ($\alpha \in (0, 1)$)	$(u^{1-\alpha} - (1 - \alpha)u - \alpha)/(\alpha(\alpha - 1))$	$(1 - u^{-\alpha})/\alpha$	✓
Reverse KL ($\alpha = 0$)	$u \log u$	$\log u + 1$	✓
Forward KL ($\alpha = 1$)	$-\log u$	$-1/u$	✓
JS-divergence	$u \log u - (u + 1) \log((u + 1)/2)$	$\log(2u/(1 + u))$	✓
Total Variation	$\frac{1}{2} u - 1 $	$u > 1 ? \frac{1}{2} : -\frac{1}{2}$	✗
Chi-squared	$(u - 1)^2$	$2(u - 1)$	✗

Beyond Reverse KL

To derive the DPO objective under f -divergence, we need to first solve [the closed form solution](#). The objective of RLHF [under \$f\$ -divergence](#) becomes:

$$\max_{\pi} \mathbb{E}_{\pi}[r(y|x)] - \beta D_f(\pi, \pi_{\text{ref}}) \quad \text{s.t.} \quad \sum_y \pi(y|x) = 1 \text{ and } \pi(y|x) \geq 0, \forall y.$$

The two constraints are introduced to ensure that the solution is a valid distribution. To solve the constrained problem, we can [apply the Lagrange multiplier](#), which gives us

$$\mathcal{L}(\pi, \lambda, \alpha) = \mathbb{E}_{\pi}[r(y|x)] - \beta \mathbb{E}_{\pi_{\text{ref}}} \left[f \left(\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right) \right] - \lambda \left(\sum_y \pi(y|x) - 1 \right) + \sum_y \alpha(y) \pi(y|x),$$

where λ and $\alpha(y)$ are the dual variables.

Beyond Reverse KL

For such problems, we can derive the **closed-form solution** for π^* , which optimally solves the above problem:

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

where $Z(x)$ is the normalization constant, and $(f')^{-1}$ is the **inverse function** of f' . By solving the equation for $r(y|x)$, we establish the following relationship between $r(y|x)$ and $\pi^*(y|x)$,

$$r(y|x) = \beta f' \left(\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} \cdot Z(x) \right).$$

Beyond Reverse KL

When D_f is the reverse KL divergence, $Z(x)$ will be **canceled out**. However, this cancellation does not generally occur for other types of f -divergences. Luckily, **by carefully analyzing the normalization constant $Z(x)$** , we can derive a closed-form solution for many other (but not all) divergences as well. To do this, we first rewrite $\pi^*(y | x)$ in the following form using the dual variables λ and $\alpha(y)$:

$$\pi^*(y|x) = \pi_{\text{ref}}(y|x)(f')^{-1} \left(\frac{r(y|x) - \lambda + \alpha(y)}{\beta} \right).$$

Beyond Reverse KL

Next, we show that for a class of f -divergences, we must have $\alpha(y) = 0$, and thus we can represent the reward using only the trainable policy, the reference policy, and a constant. This can be summarized in the following theorem [26].

Theorem

If $\pi_{\text{ref}}(y|x) > 0$ for any valid x and f' is invertible with $0 \notin \text{dom}(f')$, the reward class that is consistent with the Bradley-Terry model can be reparameterized using the policy model $\pi(y|x)$ and a reference model $\pi_{\text{ref}}(y|x)$ as

$$r(y|x) = \beta f' \left(\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} \right) + \text{const.}$$

Beyond Reverse KL

Given this theorem, for a pair of examples (x, y_w) and (x, y_l) , we can plug the reward into the **Bradley-Terry model**, which gives us the following expression:

$$p(y_w \succeq y_l | x) = \sigma \left(\beta f' \left(\frac{\pi^*(y_w | x)}{\pi_{\text{ref}}(y_w | x)} \right) - \beta f' \left(\frac{\pi^*(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right).$$

Hence, for a preference dataset \mathcal{D} , we train the model π_θ (replacing π^* in the above equation) by **minimizing the following negative log-likelihood loss**:

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[-\log \sigma \left(\beta f' \left(\frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} \right) - \beta f' \left(\frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right) \right].$$

Beyond Reverse KL

Experimental results demonstrate that **reverse KL divergence achieves the highest accuracy but the lowest diversity in generation**. Adjusting the divergence regularization allows us to **trade-off** between alignment **accuracy** and **diversity**.

Divergences	Alignment	Diversity			
	Accuracy (%) ↑	Entropy ↑	Self-Bleu ↓	Distinct-1 ↑	Distinct-2 ↑
RKL	67.19	12.25	0.880	0.021	0.151
JSD	66.80	12.31	0.878	0.021	0.159
$\alpha = 0.3$	59.77	12.85	0.849	0.026	0.199
$\alpha = 0.5$	61.72	12.90	0.841	0.028	0.206
$\alpha = 0.7$	57.42	12.98	0.839	0.027	0.202
FKL	54.30	13.01	0.834	0.029	0.210

What does DPO update do?

To understand DPO, it is useful to **analyze the gradient**, i.e., $\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}})$:

$$-\beta \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\underbrace{\sigma(\hat{r}_{\theta}(x, y_l) - \hat{r}_{\theta}(x, y_w))}_{\text{higher weight when reward estimate is wrong}} \left[\underbrace{\nabla_{\theta} \log \pi(y_w | x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_{\theta} \log \pi(y_l | x)}_{\text{decrease likelihood of } y_l} \right] \right]$$

Intuitively, the gradient of the loss function \mathcal{L}_{DPO} **increases the likelihood of the preferred completions y_w and decreases the likelihood of dispreferred completions y_l** . Importantly, the examples are **weighed by how much higher the implicit reward model \hat{r}_{θ} rates the dispreferred completions**, scaled by β , i.e, how incorrectly the implicit reward model orders the completions, accounting for the strength of the KL constraint.

Kahneman-Tversky Optimization (KTO)

Motivation

- Most methods require **preference data** (e.g., DPO, RLHF)
- Preferences are **expensive** and **scarce**
- Binary feedback is more **abundant** and **natural**

Key Insight

- Humans perceive outcomes with **cognitive biases**
- Prospect theory [27] explains these biases mathematically

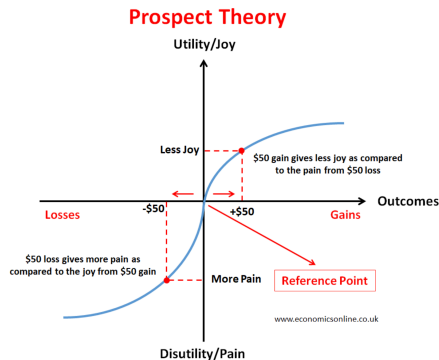


Fig. 28. Kahneman-Tversky Value Function [28]

Human-Aware Losses (HALOs)

Definition: A loss function incorporating human cognitive biases

HALO Form:

$$f(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y}[a_{x,y} \cdot v(r_\theta(x, y) - z_0)] + C_D$$

where:

- $r_\theta(x, y) = l(y) \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ (implied reward)
- $z_0 = \mathbb{E}_Q[r_\theta(x, y')]$ (reference point)
- $v(\cdot)$: value function (concave in gains)
- $a_{x,y} \in \{-1, +1\}$: direction coefficient

DPO as HALO:

- $l(y) = \beta$ (scaling factor)
- $v(z) = \log \sigma(z)$ (concave everywhere)
- $z_0 = r_\theta(x, y_l)$ (reference)
- $a_{x,y} = -1$ (minimizing loss)

PPO-Clip as HALO:

- Reward: $r_\theta = \log \frac{\tilde{\pi}_\theta}{\tilde{\pi}_{\text{ref}}}$ for implied $\tilde{\pi}$
- Value function:

$$v(z) = \min(z, (1 + \text{sign}(z)\epsilon)A)$$

- Reference: policy distribution

Kahneman-Tversky Optimization (KTO)

Key Innovation: Prospect-theory loss with binary accept/reject feedback

KTO Loss Function:

$$\mathcal{L}_{\text{KTO}} = \mathbb{E}_{x,y \sim D} [\lambda_y - v(x, y)]$$

$$\text{where } v(x, y) = \begin{cases} \lambda_D \sigma(\beta(r_\theta - z_0)) & \text{if } y \text{ desirable} \\ \lambda_U \sigma(\beta(z_0 - r_\theta)) & \text{if } y \text{ undesirable} \end{cases}$$

Advantages:

- Works with **binary** feedback
- Handles **imbalanced** data
- Can skip SFT at scale
- More **robust** to noisy data

Visualization of different HALOs

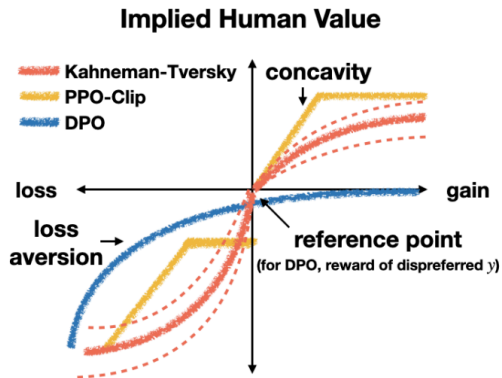


Fig. 29. The utility that a human gets from the outcome of a random variable, as implied by different human-aware losses (HALOs). Notice that the implied value functions share properties such as loss aversion with the canonical human value function in prospect theory [29]

Theoretical Insights

Why does KTO work so well?

1. Implicit noise filtering

- $|r_\theta(x, y)| \rightarrow \infty \Rightarrow |\nabla| \rightarrow 0$
- Down-weights mislabeled or extremely hard/easy examples

2. Robust to contradictory feedback

- DPO can favor a minority view when π_{ref} is skewed
- KTO (with $\lambda_D = \lambda_U$) deterministically picks the majority

3. Direct utility optimization

- Maximizes human utility, not preference likelihood
- Better alignment with true human values

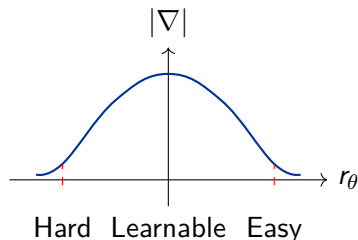


Fig. 30. KTO gradient behavior

Theorem: For contradictory preferences with noise, KTO has better worst-case guarantees than DPO [29]

When to Use KTO vs DPO?

Use KTO when:

- You have **binary** feedback data
- Data is **imbalanced** (few positive examples)
- Feedback is **noisy** or contradictory
- You want to **skip SFT** (large models)

Use DPO when:

- You have **clean preference** data
- Data has **little noise/intransitivity**
- You need maximum performance
- You have balanced datasets

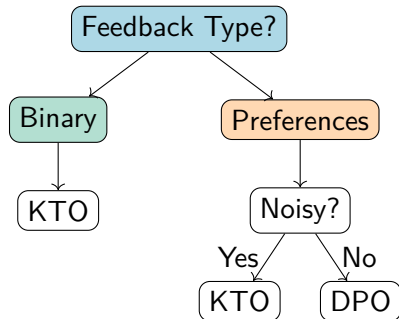


Fig. 31. Decision guide for alignment methods

ψ -Preference Optimization (ψ PO)

General Objective [30]

$$\max_{\pi} \mathbb{E}_{\substack{x \sim \rho \\ y \sim \pi(\cdot|x) \\ y' \sim \mu(\cdot|x)}} [\psi(p^*(y \succ y'|x))] - \beta D_{KL}(\pi || \pi_{ref})$$

, where $\psi : [0, 1] \rightarrow \mathbb{R}$ is a non-decreasing function

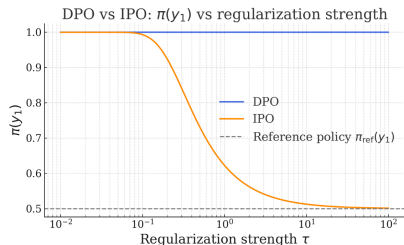
Key Insight: Both RLHF and DPO are special cases!

- RLHF/DPO: $\psi(q) = \log(q/(1 - q))$
- IPO [30]: $\psi(q) = q$ (identity function)

Problems with RLHF/DPO

Overfitting Issue:

- When $p^*(y \succ y') = 1$ (deterministic preference)
- Bradley-Terry requires $(r(y) - r(y')) \rightarrow +\infty$
- Optimal policy: $\pi^*(y') = 0$ regardless of β
- KL regularization becomes ineffective!



Key Problem

No matter how large β is (strong regularization), the optimal policy completely ignores y_2

IPO: Identity Preference Optimization

IPO Objective ($\psi = \text{Identity}$)

$$\max_{\pi} p_{\rho}^*(\pi \succ \mu) - \beta D_{KL}(\pi || \pi_{ref})$$

Key Advantages:

- Bounded objective function
- Effective KL regularization even with deterministic preferences
- No Bradley-Terry assumption needed

Optimal Policy:

$$\pi^*(y) \propto \pi_{ref}(y) \exp(\beta^{-1} \mathbb{E}_{y' \sim \mu}[p^*(y \succ y')])$$

IPO: Practical Algorithm

Sampled IPO Loss

$$\mathcal{L}(\pi) = \mathbb{E}_{(y_w, y_l) \sim \mathcal{D}} \left[\left(h_{\pi}(y_w, y_l) - \frac{\beta^{-1}}{2} \right)^2 \right]$$

where $h_{\pi}(y, y') = \log \left(\frac{\pi(y)\pi_{ref}(y')}{\pi(y')\pi_{ref}(y)} \right)$

Intuition: IPO regresses log-likelihood ratios to $\beta^{-1}/2$

Algorithm 2: Sampled IPO

Input: Dataset \mathcal{D} , reference policy π_{ref}

Define $h_{\pi}(y, y', x) = \log \left(\frac{\pi(y|x)\pi_{ref}(y'|x)}{\pi(y'|x)\pi_{ref}(y|x)} \right)$

Starting from $\pi = \pi_{ref}$, minimize: $\mathcal{L}(\pi)$

Overview

- 3 Alignment without Reward Models
 - Direct Alignment Algorithms
 - Limitations of Direct Alignment Algorithms
 - Online Direct Alignment Algorithms
 - How to Choose: RLHF or DPO?

Scaling Laws

While **Direct Alignment Algorithms** (DAAs) do not rely on a separate proxy reward model, they **still suffer from overoptimization**. As **KL budgets increase**, **DAAs show degradation patterns** similar to those of classic RLHF methods [31].

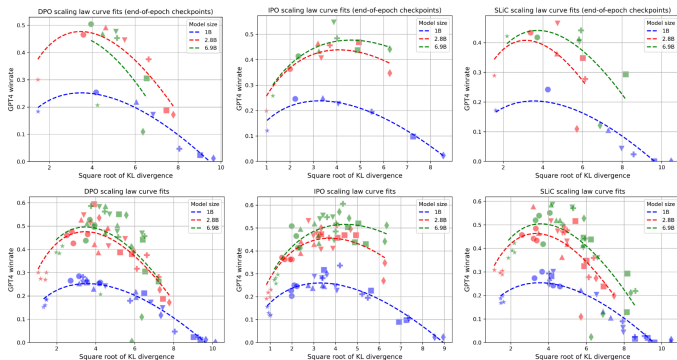


Fig. 32. Results on overoptimization in Direct Alignment Algorithms for DPO, IPO and SLiC [31].

Chosen Implicit Rewards Collapse

Ideally, DPO should increase the implicit rewards for chosen responses while decreasing them for rejected responses. In practice, however, the implicit rewards for both chosen and rejected responses decline, although the margin between them increases [32].

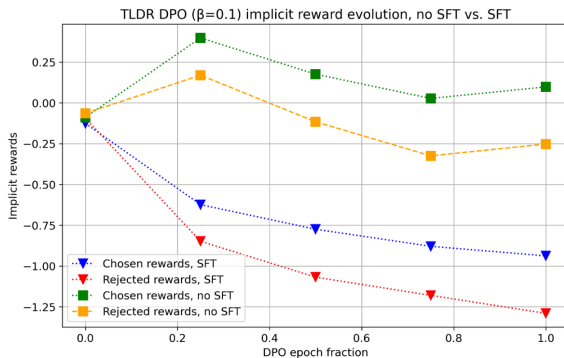


Fig. 33. The evolution of implicit rewards for DPO on TLDR [32].

Chosen Implicit Rewards Collapse

To analyze this problem, we first **reformulate the loss function** [33] of DPO:

$$\begin{aligned}\mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) &= -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right] \\ &= - \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right] \\ &= -\log \left(\frac{x_1^{\beta}}{x_1^{\beta} + x_2^{\beta}} \right),\end{aligned}$$

where β is a hyper-parameter and σ is the sigmoid function. For easing the calculation, we denote $\frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} = x_1$ and $\frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} = x_2$. In this case, **to minimize the loss, we could increase x_1 and decrease x_2 .**

Chosen Implicit Rewards Collapse

The **partial derivatives** of this loss function with respect to x_1 and x_2 are given by:

$$\begin{cases} \frac{\partial \mathcal{L}_{\text{DPO}}(x_1; x_2)}{\partial x_1} = -\frac{\beta x_2^\beta}{x_1(x_1^\beta + x_2^\beta)}, \\ \frac{\partial \mathcal{L}_{\text{DPO}}(x_1; x_2)}{\partial x_2} = \frac{\beta x_2^{\beta-1}}{x_1^\beta + x_2^\beta}. \end{cases}$$

Thus, we have

$$\left| \frac{\partial \mathcal{L}_{\text{DPO}}(x_1; x_2)}{\partial x_1} / \frac{\partial \mathcal{L}_{\text{DPO}}(x_1; x_2)}{\partial x_2} \right| = \frac{x_2}{x_1}.$$

We can further prove, for any pairwise preference data, the update rate $\frac{x_2}{x_1} < 1$ holds [33]. This means that **DPO loss function decreases the probability of dispreferred data at a faster rate** than it **increases the probability of preferred data**.

Chosen Implicit Rewards Collapse

We can see that during DPO training, the gradient varies across different regions, making the performance of the SFT model significantly impact training [33].

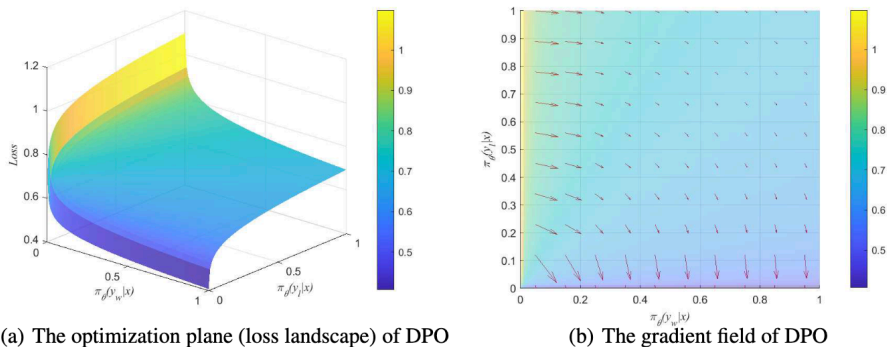


Fig. 34. The optimization plane (loss landscape) and gradient field of DPO. red arrows.

Chosen Implicit Rewards Collapse

An effective approach to address this issue is to **incorporate an additional NLL loss term into the DPO loss function** [34]:

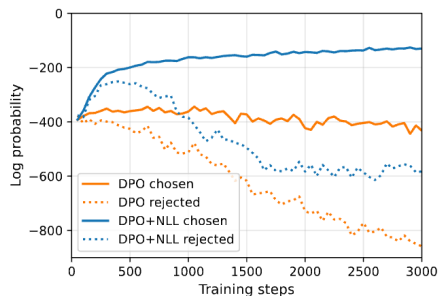
$$\begin{aligned}\mathcal{L}_{\text{DPO+NLL}} &= \mathcal{L}_{\text{DPO}}(\pi_{\theta}, \pi_{\text{ref}}) + \mathcal{L}_{\text{NLL}}(\pi_{\theta}) \\ &= -\mathbb{E}_{(y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) + \alpha \frac{\log \pi_{\theta}(y_w|x)}{|y_w|} \right],\end{aligned}$$

Note that **the NLL term is normalized by the total response length**. The hyperparameter α balances the two loss terms.

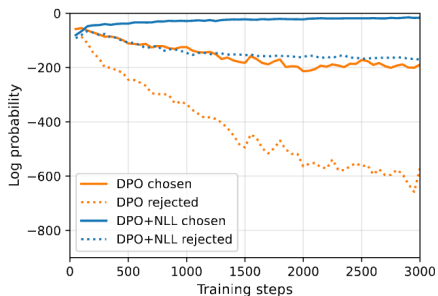
Intuitively, the **additional NLL loss** helps the model **effectively distinguish between the chosen and rejected responses** during training, **encouraging** the model to **maximize the log probability of the chosen responses**.

Chosen Implicit Rewards Collapse

DPO+NLL loss effectively prevents the chosen log probability from decreasing while gives superior test accuracy [34].



(a) Initialized from Llama



(b) Initialized from SFT trained on chosen seqs

Fig. 35. Effect of NLL loss term on DPO training for GSM8K.

Chosen Implicit Rewards Collapse

An alternative approach to address this issue is **DPO-Positive (DPOP)** [35], which incorporates a **regularization term that penalizes the model if it reduces the probability of the preferred completion below that of the reference model**. This encourages DPOP to consistently increase the likelihood of preferred completions.

$$\mathcal{L}_{\text{DPOP}} = -\mathbb{E} \left[\log \sigma \left(\beta \left(\log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) - \lambda \max \left(0, \log \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\theta}(y_w | x)} \right) \right) \right]$$

Should We Update the Reference Policy?

As shown in the Figure [36], generally, it's better to update the reference policy. Updating the reference policy relaxes the constraint paths, facilitating the optimization process. In contrast, relaxing the KL regularization strength only expands the KL ball.

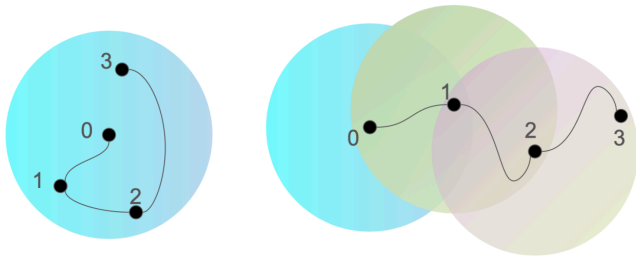


Fig. 36. Illustration of the difference between the two learning objectives. The left-hand figure corresponds to the KL-regularized target where we do not update the reference model. The right-hand figure corresponds to the non-regularized target where we always update the reference model as the last-iteration one [36].

How to Update the Reference Policy?

Generally, there are **two ways** to update the reference policy [37]: a **soft update** (center) and a **hard update** (right). In a **soft update**, the parameters of π_θ are merged into the parameters of π_{ref} with a specified weight α , blending the two gradually. In a **hard update**, the parameters of π_θ are copied entirely into the reference policy at set intervals, after a predetermined number of training steps τ .

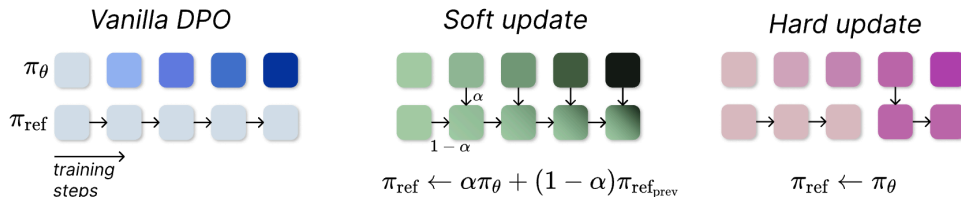
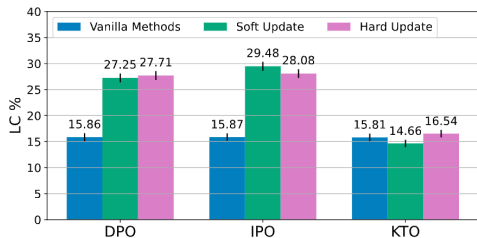


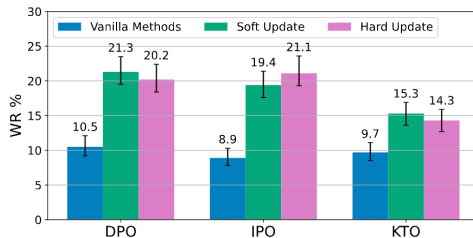
Fig. 37. Two ways to update the reference policy.

How to Update the Reference Policy?

Both update methods **outperform** vanilla DPO [37].



(a) AlpacaEval 2 (LC)



(b) ArenaHard

Fig. 38. Evaluation performance of models trained by different methods, measured on the Alpaca Eval (a) and Arena Hard (b) benchmarks. The Llama-3-Base model was used as the baseline.

Is Direct Alignment Sufficient?

DPO is popular for its stability and efficiency; however, both **empirical and theoretical analyses** show that the **DPO objective is insufficient for correcting even minor ranking errors** in the reference model [38].

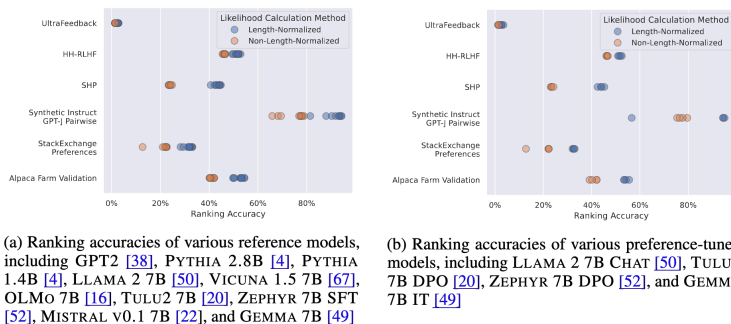


Fig. 39. Both reference and preference-tuned models exhibit low ranking accuracy on most preference datasets [38].

Is Direct Alignment Sufficient?

DPO objective was formulated to ensure that the model learns the preference dataset but does not move too far from the reference model π_{ref} , however, existing reference models rarely have correct rankings.

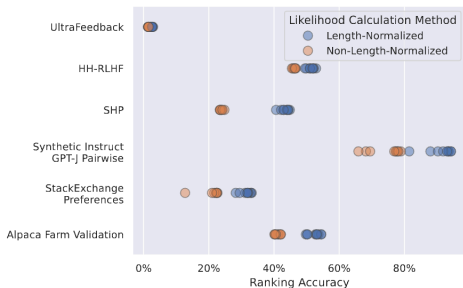


Fig. 40. Various reference models exhibit low ranking accuracy on most preference datasets [38].

Is Direct Alignment Sufficient?

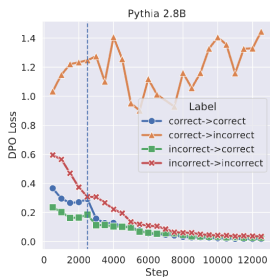
Even under **ideal conditions** (i.e., perfectly optimizing the objective function on true preference data), the **optimal ranking accuracy sometimes falls short of 100%**. This gap varies with the choice of β , indicating that the **limitations of DPO/RLHF are heavily influenced by the dependence on π_{ref}** .

Preference-Tuned Model	Length-Normalized		Non-Length-Normalized	
	$\tilde{\mathcal{R}}$	$\tilde{\mathcal{R}}^*$ (Min./Med./Max.)	\mathcal{R}	\mathcal{R}^* (Min./Med./Max.)
ZEPHYR-7B-DPO	54%	86% / 98% / 100%	42%	90% / 99% / 100%
TULU-2-DPO-7B	53%	87% / 97% / 100%	42%	91% / 99% / 100%
GOOGLE-GEMMA-7B-IT	54%	73% / 73% / 97%	40%	67% / 93% / 100%
LLAMA-2-7B-CHAT-HF	53%	87% / 97% / 100%	40%	91% / 99% / 100%

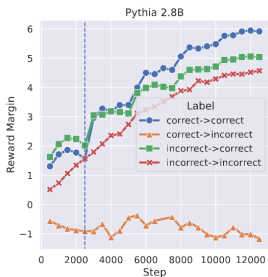
Fig. 41. The idealized ranking accuracy of existing algorithms is not perfect, but preference tuned models exhibit ranking accuracies far even from this idealized case [38].

Is Direct Alignment Sufficient?

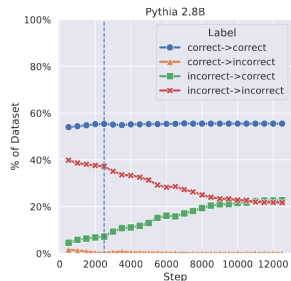
Despite continuously decreasing the loss, **DPO rarely flips the rankings of pairs and instead mostly increases the reward margin of already correctly ranked pairs.**



(a) DPO loss.



(b) DPO reward margin.

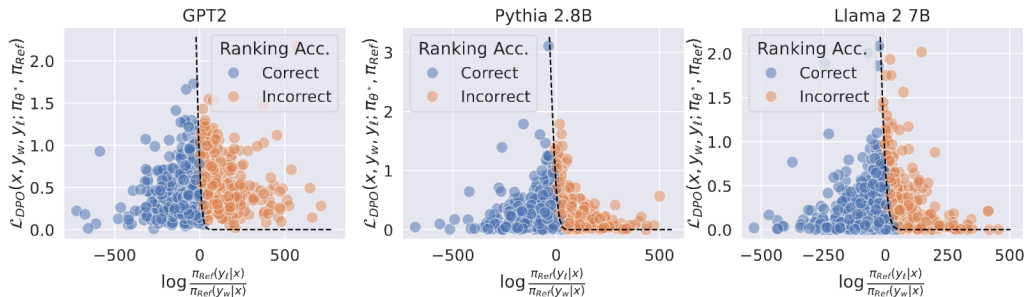


(c) Proportion of the dataset corresponding to each category of data.

Fig. 42. DPO rarely flips the rankings of pairs [38].

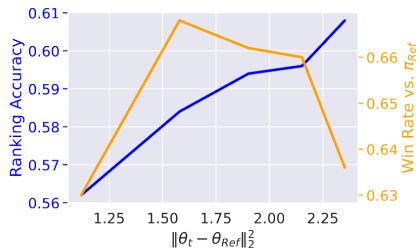
Is Direct Alignment Sufficient?

DPO loss alone does not predict ranking accuracy, due to the influence of the reference model log-ratio in the loss.

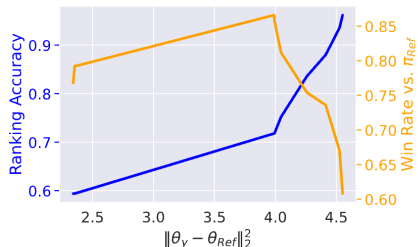


Is Direct Alignment Sufficient?

When the model weights have not travelled far from θ_{ref} , ranking accuracy and win rate increase together.



(a) Ranking accuracy and win rate of various Pythia 2.8B checkpoints during training, versus the distance travelled by the model weights θ_t from the initialization.



(b) Ranking accuracy and win rate of various γ -scaled models, versus the distance travelled by the model weights θ_γ from the initialization.

Is Direct Alignment Sufficient?

Key Takeaways:

- **Impact of Regularization.** When the model diverges significantly from the reference model, regularization toward the reference model can impair its generative capabilities, which are primarily acquired during pretraining.
- **Off-Policy vs. On-Policy Generations.** The model's off-policy behavior can no longer reliably predict its on-policy generations when the reference model in the offline objective differs greatly from the current model.
- **Effectiveness of On-Policy Preference Data.** There is a fundamental tradeoff between fitting the preference data and preserving the generative capabilities learned during pretraining. Adding on-policy preference data can improve the effectiveness of offline learning.

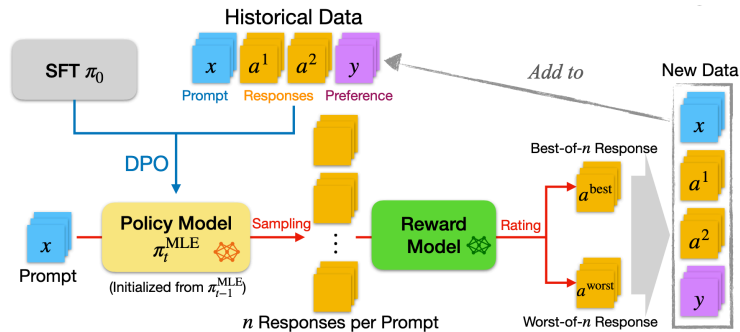
Overview

3 Alignment without Reward Models

- Direct Alignment Algorithms
- Limitations of Direct Alignment Algorithms
- **Online Direct Alignment Algorithms**
- How to Choose: RLHF or DPO?

Standard Online Direct Alignment Pipeline

The **standard online direct alignment algorithm** [39] typically begins with an SFT model as the initial policy. In each iteration, **N responses are generated per prompt** and **evaluated by a reward model (or preference model)** to **identify the best and worst responses**, forming a preference dataset. This dataset is then used by direct alignment algorithms like DPO or IPO to update the policy model iteratively.



Standard Online Direct Alignment Pipeline

This training pipeline has already been applied in the **training process of the Llama 3 series models** [40].

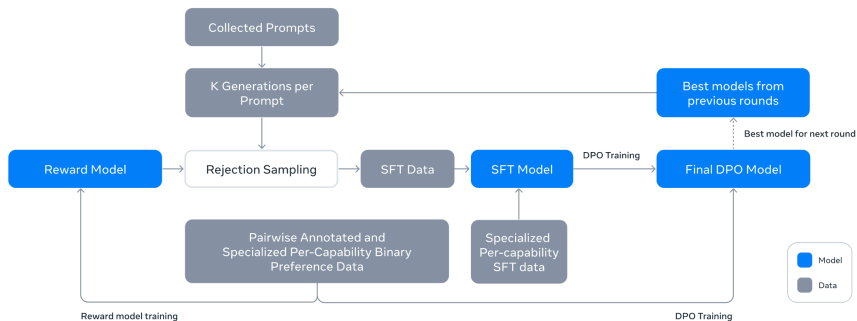


Fig. 44. Illustration of the overall post-training approach for Llama 3. The post-training strategy involves rejection sampling, supervised finetuning, and direct preference optimization [40].

Why Do We Need Online Direct Alignment Algorithms?

Despite the training simplicity of DPO, it is important to note that **DPO is fundamentally an offline algorithm**. Recent studies provide compelling evidence that **online algorithms consistently outperform offline methods** [41, 42].

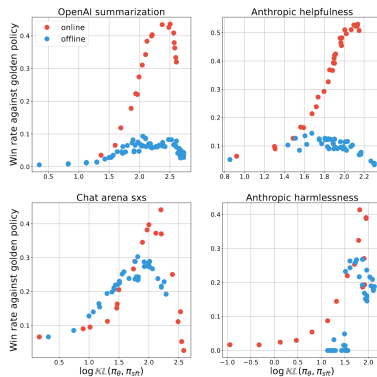


Fig. 45. Online algorithms achieve superior performance-KL divergence trade-offs compared to offline methods [41].

When Does On-policy Sampling Improve over Offline Fine-tuning?

Core Question: When does on-policy sampling improve over offline fine-tuning, even though on-policy samples are annotated by a reward model learned from offline data? Is sample reuse useful or harmful?

Key Takeaways [42]:

① On-policy sampling improves performance

- Consistent performance improvements with on-policy sampling in reward models
- Larger batch sizes B lead to more off-policy updates and lower performance
- Demonstrated on both on-policy RWR and REINFORCE algorithms

② Sample reuse enables leveraging off-policy data

- Moderate reuse improves efficiency (e.g., $T = 2$ outperforms $T = 1$)
- Excessive reuse hurts performance due to increased off-policy nature
- Algorithms with off-policy control (e.g., PPO) perform better with reuse

When Does An Explicit Negative Gradient Help the Discovery of Effective Policies?

Core Question: When and how does an **explicit negative gradient** improve policy discovery in offline preference learning?

Key Takeaways [42]:

1. Negative gradient accelerates convergence

- Aggressively pushes down bad action likelihoods
- Leads to better policies with larger KL values
- IPO (with negative gradient) > Best-of-N/RWR

2. Contrastive training widens margins

- Increases gap between preferred/dispreferred
- DPO > non-contrastive methods (e.g., Pref-FT)
- Effect varies with model capacity & data size

Does On-Policy Sampling Offer Complementary Benefits to Negative Gradient?

Core Question: Does **on-policy sampling** offer **complementary benefits** to negative gradient, resulting in **better performance** with contrastive approaches like DPO?

Empirical Evidence:

- **Online IPO** (on-policy + negative gradient) > **Offline IPO** (negative gradient only) > **RWR** (on-policy only)
- Complementary mechanisms:
 - On-policy sampling: explores new regions of policy space
 - Negative gradient: efficiently suppresses bad actions
- Combined approach achieves superior reward-KL tradeoff

Key Takeaways [42]: On-policy sampling and negative gradients are complementary

Overview

3 Alignment without Reward Models

- Direct Alignment Algorithms
- Limitations of Direct Alignment Algorithms
- Online Direct Alignment Algorithms
- How to Choose: RLHF or DPO?

RLHF vs DPO

RLHF (Two-stage Approach)

① Reward Modeling: Learn $r_{\text{RLHF}} \in \mathcal{F}$

② Policy Optimization:

$$\pi_{\text{RLHF}} = \arg \max_{\pi \in \Pi} V_{r_{\text{RLHF}}}^{\pi}$$

Advantages:

- Explicit reward representation
- Leverages reward structure

DPO (Direct Approach)

- ① Bypasses reward modeling
- ② Directly optimizes policy from preferences
- ③ Uses surrogate reward:

$$\hat{r}_{\theta}(y) = \beta \log \frac{\pi_{\theta}(y)}{\pi_{\text{ref}}(y)}$$

Advantages:

- More stable training
- No need for RL algorithms

Central Question

Under what conditions is DPO equivalent, superior, or inferior to RLHF?

Performance Analysis Framework [43]

Performance Metric:

$$V_{r^*}^\pi := \mathbb{E}_{y \sim \pi}[r^*(y)] - \beta \text{KL}(\pi \| \pi_{\text{ref}})$$

Model Classes:

- Reward class: $\mathcal{F} = \{r_\phi : \phi \in \mathbb{R}^{d_R}\}$
- Policy class: $\Pi = \{\pi_\theta : \theta \in \mathbb{R}^{d_P}\}$
- Surrogate reward class: \mathcal{F}_Π

Key Conditions [43]:

- ① $r^* \in \mathcal{F}, \pi^* \in \Pi$ (No mis-spec)
- ② $r^* \in \mathcal{F}, \pi^* \notin \Pi$ (Policy mis-spec)
- ③ $r^* \notin \mathcal{F}, \pi^* \in \Pi$ (Reward mis-spec)
- ④ $r^* \notin \mathcal{F}, \pi^* \notin \Pi$ (Double mis-spec)

Bradley-Terry Model

Human preferences follow: $p^*(y_1 > y_2) = \sigma(r^*(y_1) - r^*(y_2))$

Results: Model Mis-specification

Scenario	Performance Comparison	Insight
No Mis-specification	$V_{r^*}^{\pi_{\text{RLHF}}} = V_{r^*}^{\pi_{\text{DPO}}}$	Both achieve optimal
Policy Mis-specification	$V_{r^*}^{\pi_{\text{RLHF}}} \geq V_{r^*}^{\pi_{\text{DPO}}}$	RLHF leverages exact reward
Reward Mis-specification	$V_{r^*}^{\pi_{\text{RLHF}}} \leq V_{r^*}^{\pi_{\text{DPO}}}$	DPO avoids reward error
Isomorphic Double	$V_{r^*}^{\pi_{\text{RLHF}}} = V_{r^*}^{\pi_{\text{DPO}}}$	Online DPO can excel

Key Findings [43]:

- **RLHF advantage:** When reward model is realizable but policy is not
- **DPO advantage:** When policy is realizable but reward model is not
- **Online DPO:** Can outperform both under isomorphic mis-specification

Statistical Efficiency Gap: Sparse Recovery

Dual-token Sparse Prediction (DTSP) Task:

- Ground-truth reward: $r^*(y, \omega) = \beta r_{\text{sparse}}^T \psi(y) + \beta e_1^T \psi(y, \omega)$
- Sparsity: $\|r_{\text{sparse}}\|_0 = k \ll d$

Reward Learning (RLHF):

- Can leverage sparsity structure
- Estimation error: $O(\sqrt{k \log d/n})$
- Efficient sparse recovery

Surrogate Reward (DPO):

- Entangles sparse structure
- Estimation error: $\Omega(d/n)$
- Cannot exploit sparsity

Statistical Separation

RLHF requires $O(k \log d)$ samples vs DPO requires $\Omega(d)$ samples

Takeaways

Main Insights [43]

- ① **No universal winner:** Performance depends on model mis-specification type
- ② **RLHF advantages:** Better when reward is simpler/sparse than policy
- ③ **DPO advantages:** Avoids reward modeling errors, more stable training
- ④ **Online methods:** Can bridge gaps in certain scenarios

Practical Guidelines:

- Choose **RLHF** when:
 - Reward structure is simple/sparse
 - High-quality reward model available
 - Policy class is limited
- Choose **DPO** when:
 - Reward modeling is challenging
 - Policy class is expressive
 - Training stability is crucial

Outline

- 1 Alignment for LLMs: Introduction
- 2 Alignment with Reward Models
- 3 Alignment without Reward Models
- 4 Alignment with General Preference Models**
- 5 Alignment with Verifiers

Overview

- 4 Alignment with General Preference Models
 - Revisit Stages in Language Model Training
 - Solution Concept
 - Solving the Minmax Winner

Revisit Three Stages in RLHF

Standard RLHF pipeline [44] typically consists of **three stages**: 1) **supervised fine-tuning (SFT)**; 2) **reward model training** and 3) **RL preference optimization**.

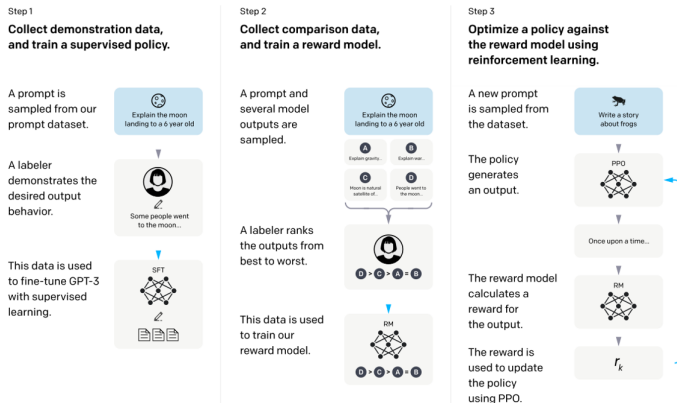


Fig. 46. Three stages of RLHF [44].

Revisit Reward model (RM) training

In the **reward model training stage**, the SFT model is prompted with prompts x to produce pairs of answers $(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$. **These answer pairs are then presented to human labelers who express preferences for one answer**, denoted as:

$$y_w \succ y_l \mid x,$$

where y_w and y_l denotes the preferred and dispreferred completion amongst (y_1, y_2) respectively. The preferences are assumed to be generated by some latent reward model $r^*(y, x)$, which we do not have access to. **The Bradley-Terry [7] model stipulates that the human preference distribution p^* can be written as:**

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}.$$

Revisit Reward Model (RM) Training

Assuming access to a static dataset of comparisons:

$$\mathcal{D} = \left\{ x^{(i)}, y_w^{(i)}, y_l^{(i)} \right\}_{i=1}^N$$

sampled from p^* , we can parametrize a reward model $r_\phi(x, y)$ and estimate the parameters via maximum likelihood. The negative log-likelihood loss:

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

where σ is the logistic function. In the context of LMs, the network $r_\phi(x, y)$ is often initialized from the SFT model $\pi^{\text{SFT}}(y | x)$. To ensure a reward function with lower variance, prior works normalize the rewards, such that:

$$\mathbb{E}_{x, y \sim \mathcal{D}} [r_\phi(x, y)] = 0$$

for all x .

Bradley-Terry Model Assumption

Standard RLHF pipeline **relies on the Bradley-Terry (BT) model** [7] assumption to train a reward model. However, this assumption **oversimplifies** the complex nature of human preferences and **fails** to capture several critical aspects [14] of real-world human preferences:

- **Transitivity** The BT model **enforces strict transitivity in preferences**, meaning if a human prefers response A to B and B to C, they must prefer A to C.
- **Independence** The BT model assumes independence between preference judgments, **treating each comparison as an isolated event**.
- **Completeness** The BT model presumes completeness in human judgments, **suggesting that humans can always make clear preference decisions between any two responses**.

Overview

- 4 Alignment with General Preference Models
 - Revisit Stages in Language Model Training
 - **Solution Concept**
 - Solving the Minmax Winner

Background

Given choices from a population of raters that are represented as a preference function \mathcal{P} , social choice theory [45] studies the question of how best to select options that satisfy the diversity of preferences inherent in the said population.

	a	b	c	d
a	0	+1	+1	-1
b	-1	0	+1	-1
c	-1	-1	0	+1
d	+1	+1	-1	0

Fig. 47. An intransitive preference function \mathcal{P} over (a, b, c, d) . $\mathcal{P}(x, y) = 1$ if $P(xy) = 1$, 1 if $P(xy) = 0$, and 0 if $P(xy) = 0.5$ [46].

Copeland Winner

Given this preference function, the solution concept of **Copeland Winner** [46] means to **pick the option that beats the largest number of other options**. In the above matrix, this would be either option *a* or *d* as they have the largest row sums.

$$CW(\mathcal{P}) \triangleq \arg \max_{\pi \in \Pi} \sum_{\pi' \in \Pi} \mathcal{P}(\pi, \pi').$$

While intuitively appealing, **Copeland Winners** are often **not unique** and **can not handle intransitivity** [46].

Minmax Winner

In contrast to Copeland Winner, **Minmax Winner** [47], also known as a von Neumann Winner, is defined as the following pair of strategies:

$$\text{MW}(\mathcal{P}) \triangleq \left(\begin{array}{l} \operatorname{argmax}_{p \in \Delta(\Pi)} \min_{q \in \Delta(\Pi)} \mathbb{E}_{\pi_1 \sim p, \pi_2 \sim q} [\mathcal{P}(\pi_1, \pi_2)], \\ \operatorname{argmin}_{q \in \Delta(\Pi)} \max_{p \in \Delta(\Pi)} \mathbb{E}_{\pi_1 \sim p, \pi_2 \sim q} [\mathcal{P}(\pi_1, \pi_2)] \end{array} \right).$$

The Minmax Winner is essentially the **Nash equilibrium** of the preference-based payoff matrix, thus **it is unique and always exists**. Intuitively, this means that **we never pick a solution that makes a significant portion of the population consistently unhappy**.

Minmax Winner

We can make following observations about a **Minmax Winner** [46]:

- **We don't need to assume the existence of an underlying reward function** when we define a Minmax Winner.
- **When there actually does exist an underlying reward function** that explains the observed preferences, **the Minmax Winners coincide with the optimal policy for that reward**.
- **Minmax Winners satisfy a variety of desirable consistency properties** (e.g. merging populations that agree on a Minmax Winner cannot change the outcome), which deterministic options like the Copeland Winner cannot satisfy simultaneously.

Overview

- 4 Alignment with General Preference Models
 - Revisit Stages in Language Model Training
 - Solution Concept
 - Solving the Minmax Winner

Mirror Descent

Mirror Descent (MD) [48, 49] is a **first-order optimization algorithm**. The update rule for MD applied to player i is given by:

$$\pi^{k+1} = \arg \min_{\pi \in \Pi} \left\{ \langle F(\pi^k), \pi \rangle + \frac{1}{\eta} B_{\psi}(\pi, \pi^k) \right\}, \quad (5)$$

where $\eta > 0$ is the learning rate, and $B_{\psi}(\pi, \pi') = \psi(\pi) - \psi(\pi') - \langle \nabla \psi(\pi'), \pi - \pi' \rangle$ is the **Bregman divergence associated with a strongly convex function ψ** .

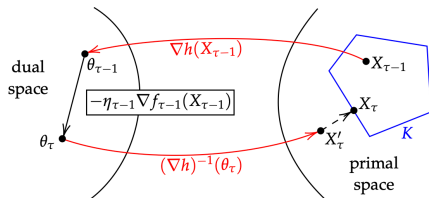


Fig. 48. Visualization of a step of Mirror Descent.

Self-Play Preference Optimization

Self-Play Preference Optimization (SPO) [46] proposes leveraging MD-based deep RL algorithms to find the Nash equilibrium.

Algorithm 2 SPO (Practical Version)

```

1: Input: Iterations  $T$ , Preference fn.  $\mathcal{P}$ , Queue size  $B$ ,
   Reinforcement learning algo.  $\text{RL} : \Pi \times \mathcal{D} \rightarrow \Pi$ .
2: Output: Trained policy  $\pi$ .
3: Initialize  $\pi_1 \in \Pi$ , Queue  $\mathcal{Q} \leftarrow [\xi_{1:B} \sim \pi_1]$ .
4: for  $t$  in  $1 \dots T$  do
5:   Sample  $\xi_t \sim \pi_t$ .
6:   // Win-rate over queue as reward
7:   Compute  $r_t(\xi_t) = \frac{1}{|\mathcal{Q}|} \sum_{q=1}^B \mathcal{P}(\xi_t, \xi_q)$ .
8:   Set  $r_t^h = r_t(\xi_t)/H, \forall h \in [H]$ .
9:   // use PPO, TRPO, SAC ...
10:   $\pi_{t+1} \leftarrow \text{RL}(\pi_t, \mathcal{D} = \{(s_t^h, a_t^h, r_t^h)\}_{h \in [H]})$ .
11:   $\mathcal{Q} \leftarrow [\xi_2, \dots, \xi_B, \xi_t]$ .
12: end for
13: Return best of  $\pi_{1:T}$  on validation data.
```

Fig. 49. Pseudocode of SPO [46].

Direct Nash Optimization

Inspired by DPO [50], [Direct Nash Optimization](#) [51] **bypasses the RL step** and **directly optimizes policies** to find the Nash equilibrium.

Algorithm 1 Direct Nash Optimization (DNO)

input: General preference function \mathcal{P} , learning rate η , number of iterations T , prompt distribution ρ .

- 1: Initialize $\pi_1 \leftarrow \text{unif}(\mathcal{A})$.
- 2: **for** iteration $t = 1, 2, \dots, T$ **do**
- 3: Compute $r_t(x, y) \leftarrow \mathbb{E}_{y' \sim \pi_t(\cdot | x)} [\mathcal{P}(y \succ y' | x)], \forall (x, y) \in \mathcal{X} \times \mathcal{Y}$.
- 4: Obtain π_{t+1} by,

$$\pi_{t+1} \leftarrow \operatorname{argmax}_{\pi \in \Pi} \mathbb{E}_{(x, y_1, y_2) \sim \mathcal{D}_t} \left\{ \sigma(r_t(x, y_1) - r_t(x, y_2)) \log \left[\sigma \left(\eta \log \frac{\pi(y_1 | x)}{\pi_t(y_1 | x)} - \eta \log \frac{\pi(y_2 | x)}{\pi_t(y_2 | x)} \right) \right] \right. \\ \left. + \sigma(r_t(x, y_2) - r_t(x, y_1)) \log \left[\sigma \left(\eta \log \frac{\pi(y_2 | x)}{\pi_t(y_2 | x)} - \eta \log \frac{\pi(y_1 | x)}{\pi_t(y_1 | x)} \right) \right] \right\}, \quad (8)$$

where \mathcal{D}_t is generated by $x \sim \rho, y_1 \sim \mu_{1,t}(\cdot | x), y_2 \sim \mu_{2,t}(\cdot | x)$; $\mu_{1,t}$ and $\mu_{2,t}$ can be either off-policy (e.g., pre-defined) or on-policy (based on π_t).

- 5: **end for**
 - 6: **return** $\bar{\pi} = \text{unif}(\pi_{1:T})$.
-

Fig. 50. Pseudocode of DNO [51].

Self-Play Preference Optimization

Self-Play Preference Optimization (SPPO) [52] follows the same idea, but adopts a different approach to derive the closed-form solution. The **closed-form solution of MD** can be written as:

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) = \frac{\pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}))}{Z_{\pi_t}(\mathbf{x})}, \quad (6)$$

where $Z_{\pi_t}(\mathbf{x}) = \sum_{\mathbf{y}} \pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}))$ is the **normalizing factor**. For any fixed \mathbf{x} and \mathbf{y} , π_{t+1} should satisfy the following equation:

$$\log \left(\frac{\pi_{t+1}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})} \right) = \eta \cdot \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x}). \quad (7)$$

Self-Play Preference Optimization

Unlike DPO, SPPO **direct approximate eq. (7) using L_2 loss**:

$$\pi_{t+1} = \arg \min_{\pi} \mathbb{E}_{x \sim \mathcal{X}, y \sim \pi_t(\cdot|x)} \left(\log \left(\frac{\pi(y|x)}{\pi_t(y|x)} \right) - (\eta \mathbb{P}(y \succ \pi_t|x) - \log Z_{\pi_t}(x)) \right)^2.$$

SPPO chooses to **sample K finite samples** and **denote the empirical distribution by $\hat{\pi}_t^K$** and replaces $\log Z_{\hat{\pi}_t^K}(x)$ with $\eta/2$. The **finite-sample optimization problem** can be then approximated as

$$\pi_{t+1} = \arg \min_{\pi} \mathbb{E}_{x \sim \mathcal{X}, y \sim \pi_t(\cdot|x)} \left(\log \left(\frac{\pi(y|x)}{\pi_t(y|x)} \right) - (\eta \mathbb{P}(y \succ \hat{\pi}_t^K|x) - \frac{1}{2}) \right)^2.$$

Limitations of Mirror Descent

Average-iterate convergence can be achieved by either:

- Maintaining multiple policies for averaging, or
- Learning and updating an averaged policy.

Both approaches introduce considerable computational overhead, particularly in the context of LLM alignment.

Limitations of Mirror Descent

To avoid this computational burden, many studies simply adopt the last-iterate policy of MD. However, recent research has revealed that the last-iterate policy of MD exhibits Poincaré recurrence behavior [53, 54].

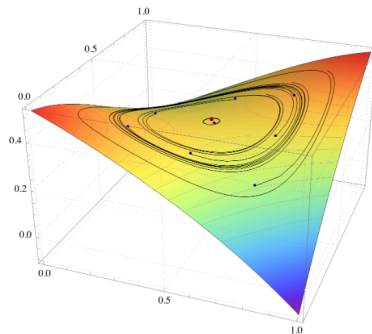


Fig. 51. MD in the saddle-point problem $f(x_1, x_2) = (x_1 \frac{1}{2})(x_2 \frac{1}{2}) + \frac{1}{3} \exp((x_1 \frac{1}{4})^2 (x_2^2 \frac{3}{4})^2)$.

Last-iterate Convergence

In the context of LLM alignment, this implies that approaches that based on MD such as SPO [46] and SPPO [52] may not be well-suited. This raises a critical question: can we design an algorithm that achieves last-iterate convergence? Formally, we define last-iterate convergence as:

Definition (Last-iterate Convergence)

Consider nonempty set of equilibria $\Pi^* \subset \Pi$, we say that a sequence $\{\pi^k\}_{k \geq 1}$ exhibits last-iterate convergence if π^k converges to $\pi^* \in \Pi^*$ as $k \rightarrow \infty$.

Magnetic Mirror Descent

Compared to MD, Magnetic Mirror Descent (MMD) [55] introduces an **additional magnetic term**. Formally, the MMD update rule can be expressed as

$$\pi^{k+1} \in \arg \min_{\pi \in \Pi} \left\{ \langle F(\pi^k), \pi \rangle + \alpha B_\psi(\pi; \pi_{\text{ref}}) + \frac{1}{\eta} B_\psi(\pi; \pi^k) \right\}, \quad (8)$$

where π_{ref} is the magnet, which means π^{k+1} is attracted to either $\min_{\pi \in \Pi} \psi(\pi)$ or π_{ref} , α is the regularization temperature, η is the learning rate. **In contrast to MD, MMD solves the regularized game**

$$\min_{\pi_1 \in \Pi_1} \max_{\pi_2 \in \Pi_2} \alpha g(\pi_1) + f(\pi_1, \pi_2) - \alpha g(\pi_2), \quad (9)$$

where f and g are both convex functions and g can be taken either ψ or $B_\psi(\cdot; \pi_{\text{ref}})$ for some π_{ref} .

Magnetic Mirror Descent

The **last-iterate** policy of MMD performs **much better** than that of MD.

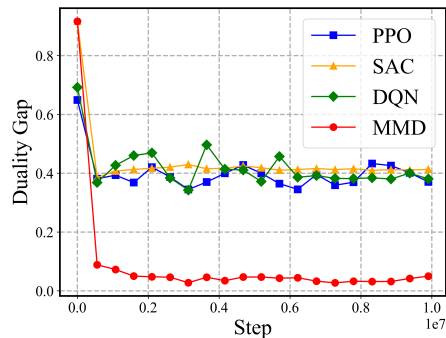


Fig. 52. Performance of PPO, SAC, DQN, MMD on Kuhn Poker.

Magnetic Mirror Descent

Theorem 1 (Last-iterate Convergence) [55]

Consider the MMD update rule. Assume $\pi^{k+1} \in \text{dom}\psi$ and Π is bounded, F is monotone and L -smooth with respect to $\|\cdot\|$, g is 1-strongly convex relative to ψ over Π with g differentiable over the interior of $\text{dom}\psi$. Then the sequence $\{\pi^k\}_{k \geq 1}$ generated by MMD exhibits linear last-iterate convergence to the solution π_r^* if $\eta \leq \frac{\alpha}{L^2}$. Specifically,

$$B_\psi(\pi_r^*; \pi^{k+1}) \leq B_\psi(\pi_r^*; \pi^1) \left(\frac{1}{1 + \eta\alpha} \right)^k,$$

where $\alpha > 0$ is the regularization temperature and $\eta > 0$ is the learning rate.

Magnetic Mirror Descent

Another interesting aspect of MMD is that **MMD can be seen as a special case of MD with an adjusted gradient and stepsize.**

Theorem 2 (Connections to Mirror Descent) [55]

The update rule of MMD in (8) is equivalent to the following rule:

$$\pi^{k+1} \in \arg \min_{\pi \in \Pi} \left\{ \langle F(\pi^k) + \alpha \nabla_{\pi^k} B_{\psi}(\pi^k; \pi_{\text{ref}}), \pi \rangle + \frac{1}{\bar{\eta}} B_{\psi}(\pi; \pi^k) \right\},$$

where the stepsize is defined as $\bar{\eta} = \frac{\eta}{1+\eta\alpha}$.

Nash Learning from Human Feedback

Nash-MD [56] aims to find the **Nash equilibrium** of the **KL-regularized game** via MD.

$$\mathcal{P}_\tau(\pi \succ \pi') \stackrel{\text{def}}{=} \mathcal{P}(\pi \succ \pi') - \tau \text{KL}_\rho(\pi, \mu) + \tau \text{KL}_\rho(\pi', \mu) \quad (2)$$

The **reference policy** of Nash-MD is defined as a **geometric mixture between the current policy π_t and the reference policy μ** :

$$\pi_t^\mu(y) \stackrel{\text{def}}{=} \frac{\pi_t(y)^{1-\eta_t\tau} \mu(y)^{\eta_t\tau}}{\sum_{y'} \pi_t(y')^{1-\eta_t\tau} \mu(y')^{\eta_t\tau}}, \quad (10)$$

A **step of mirror descent** relative to the regularized policy π_t^μ is:

$$\pi_{t+1} \stackrel{\text{def}}{=} \arg \max_{\pi} [\eta_t \mathcal{P}(\pi \succ \pi_t^\mu) - \text{KL}(\pi, \pi_t^\mu)]. \quad (11)$$

Nash Learning from Human Feedback

Nash-MD produces a sequence of policies $(\pi_t)_{1 \leq t \leq T}$ with **last-iterate convergence to the regularized Nash equilibrium π_τ^* at a speed $\mathcal{O}(1/T)$** .

Theorem 3 (Last-iterate Convergence of Nash-MD) [56]

Let π_τ^* be the Nash equilibrium of the regularized preference model. At every iteration t we have that

$$\text{KL}(\pi_\tau^*, \pi_{t+1}) \leq (1 - \eta_t \tau) \text{KL}(\pi_\tau^*, \pi_t) + 2\eta_t^2. \quad (6)$$

For the choice $\eta_t = 2/(\tau(t+2))$ we have

$$\text{KL}(\pi_\tau^*, \pi_T) \leq \frac{8}{\tau^2(T+1)}. \quad (12)$$

Iterative Nash Policy Optimization

Similar to DNO [51] and SPPO [52], [Iterative Nash Policy Optimization](#) (INPO) [57] aims to [directly optimize policies](#) to find Nash equilibrium while [achieving last-iterate convergence](#). Given the preference oracle \mathbb{P} , the loss function for any $\pi \in \Pi$ is defined as:

$$\ell_t(\pi) := -\mathbb{E}_{y \sim \pi, y' \sim \pi_t} [\mathbb{P}(y \succ y')] + \tau \text{KL}(\pi \| \pi_{\text{ref}}). \quad (13)$$

Minimize this loss with MD:

$$\pi_{t+1} = \arg \min_{\pi \in \Pi} \langle \nabla \ell_t(\pi_t), \pi \rangle + \eta \text{KL}(\pi \| \pi_t), \quad (14)$$

Iterative Nash Policy Optimization

Following [the derivation of DPO](#) [50], we have that:

$$\begin{aligned}\pi_{t+1}(y) &\propto \pi_t(y) \exp\left(-\frac{1}{\eta} \nabla_y \ell_t(\pi_t)\right) \\ &\propto \exp\left(\frac{\mathbb{P}(y \succ \pi_t)}{\eta}\right) \pi_{\text{ref}}(y)^{\frac{\tau}{\eta}} \pi_t(y)^{1-\frac{\tau}{\eta}},\end{aligned}$$

where $\mathbb{P}(y \succ \pi_t)$ represents $\mathbb{E}_{y' \sim \pi_t}[\mathbb{P}(y \succ y')]$. To **avoid the normalization factor**, we define $h_t(\pi, y, y')$ as:

$$h_t(\pi, y, y') = \log \frac{\pi(y)}{\pi(y')} - \frac{\tau}{\eta} \log \frac{\pi_{\text{ref}}(y)}{\pi_{\text{ref}}(y')} - \frac{\eta - \tau}{\eta} \log \frac{\pi_t(y)}{\pi_t(y')}. \quad (15)$$

Iterative Nash Policy Optimization

Following the derivation of IPO [58], we have the population loss:

$$\mathbb{E}_{y, y' \sim \pi_t, y_w, y_l \sim \lambda_p(y, y')} \left[\left(h_t(\pi, y_w, y_l) - \frac{1}{2\eta} \right)^2 \right]. \quad (16)$$

Iterative Nash Policy Optimization

INPO enjoys the last-iterate convergence to Nash policy π^* at the speed $\mathcal{O}(1/T)$.

Theorem 4 (Last-iterate Convergence of INPO)

let $C = \max(B\tau, 1)$, at each iteration t we have

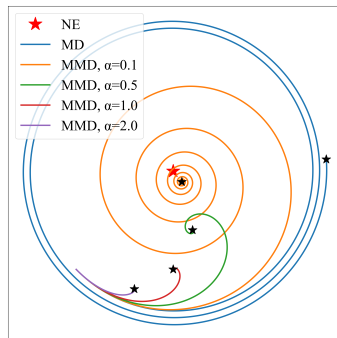
$$\text{KL}(\pi^*, \pi_{t+1}) \leq \left(1 - \frac{\tau}{\eta}\right) \text{KL}(\pi^*, \pi_t) + \frac{8C^2}{\eta^2}. \quad (17)$$

Suppose we use a time-varying parameter $\eta_t = \frac{\tau(t+2)}{2}$, we obtain

$$\text{KL}(\pi^*, \pi_T) \leq \frac{32C^2}{\tau^2(T+1)}. \quad (18)$$

Convergence to the NE of the Original Game

Although Nash-MD, INPO enjoy last-iterate convergence, **MMD alone only achieves last-iterate convergence to the regularized game**. **Increasing the regularization accelerates MMD convergence**, but simultaneously **causes the learned NE to deviate further from the NE of the original game**.



Convergence to the NE of the Original Game

To achieve last-iterate convergence to the NE of the original game, we first define the n -th regularized game as

$$J_n(\pi_1, \pi_2) = \min_{\pi_1 \in \Pi_1} \max_{\pi_2 \in \Pi_2} \mathcal{P}(\pi_1 \succ \pi_2) + \alpha D_{\text{KL}}(\pi_1 \| \pi_r^{*,n-1}) - \alpha D_{\text{KL}}(\pi_2 \| \pi_r^{*,n-1}), \quad (19)$$

where $\pi_r^{*,n-1}$ is the NE of the $(n-1)$ -th regularized game.

Convergence to the NE of the Original Game

Then, we can prove the following Lemma.

Lemma 1

Let $\{\pi_r^{*,n}\}_{n \geq 1}$ be the sequence of NEs of the regularized games generated by iteratively solving (19), where $\pi_r^{*,1}$ is an arbitrary initial reference policy in the interior of Π . For any $n \geq 1$, if $\pi_r^{*,n} \in \Pi \not\subseteq \Pi^*$, we have

$$\min_{\pi^* \in \Pi^*} D_{\text{KL}}(\pi^* \parallel \pi_r^{*,n+1}) < \min_{\pi^* \in \Pi^*} D_{\text{KL}}(\pi^* \parallel \pi_r^{*,n}). \quad (20)$$

Otherwise, if $\pi_r^{*,n} \in \Pi^*$, then $\pi_r^{*,n+1} = \pi_r^{*,n} \in \Pi^*$.

Convergence to the NE of the Original Game

Based on Lemma 1, we have the following theorem.

Theorem 5 (Convergence to the NE of the Original Game)

If Lemma 1 holds, the sequence $\{\pi_r^{*,n}\}_{n \geq 1}$ converges to the NE $\pi^* \in \Pi^*$ of the original game as $n \rightarrow \infty$.

Convergence to the NE of the Original Game

Theorem 2 suggests a **two-stage convergence process** for MMD to reach the NE of the original game.

- First, as established in Theorem 1, **MMD achieves linear last-iterate convergence to the NE** of each **regularized game**.
- Then, by **iteratively updating the magnet policy to the most recent regularized NE**, we guide the sequence of regularized NEs $\{\pi_r^{*,n}\}_{n \geq 1}$ towards the NE π^* of the original game.

Magnetic Preference Optimization

Based on above analysis, we propose **Magnetic Preference Optimization (MPO)**, which achieves last-iterate convergence to the NE of the original game [59].

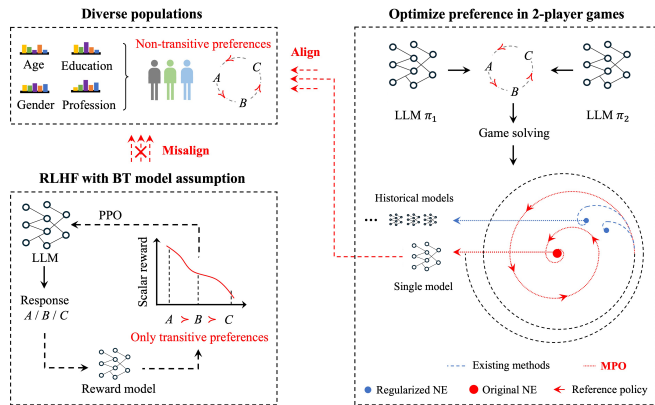


Fig. 53. Overview of MPO.

Magnetic Preference Optimization

MPO significantly enhances model safety across three self-play iterations and consistently boosts the win rate across eight safety-related categories.

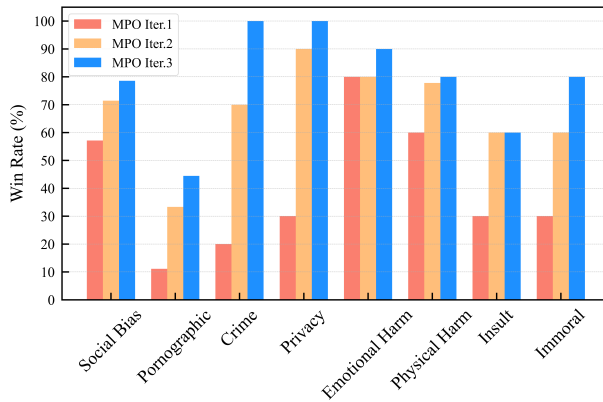


Fig. 54. Performance across each safety-related category for three self-play iterations of MPO.

Magnetic Preference Optimization

MPO demonstrates a steady improvement in win rates across three iterations. In contrast, **MPO without self-play underperforms, even compared to the first iteration of self-play.** This reveals the fact that while RLHF with BT models runs the risk of overfitting to the reward model, **RLHF with general preference models faces the risk of overfitting to the opponent.**

Settings	GPT-4o-Evaluation		
	Win ↑	Lose ↓	Tie ↔
MPO Iter.1	51.8%	21.7%	26.5%
MPO Iter.2	69.9%	10.8 %	19.3%
MPO Iter.3	79.5%	9.6 %	10.9%
MPO wo.SP	30.1%	15.7%	54.2%

Fig. 55. Performance across each safety-related category for three self-play iterations of MPO.

Outline

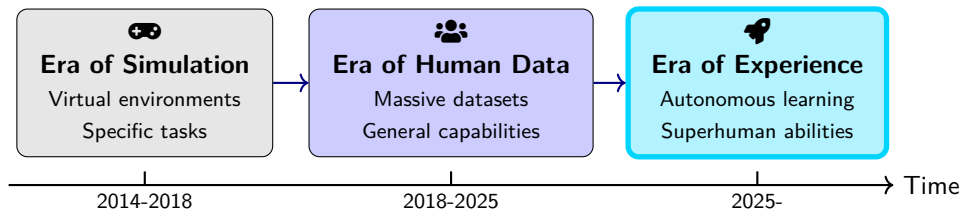
- 1 Alignment for LLMs: Introduction
- 2 Alignment with Reward Models
- 3 Alignment without Reward Models
- 4 Alignment with General Preference Models
- 5 Alignment with Verifiers**

Overview

5 Alignment with Verifiers

- Era of Experience
- Test-time Scaling Law
- Verifiable Rewards
- Process Rewards

Evolution of AI Paradigms



Key Milestones [60]

- **Era of Simulation:** AlphaGo, Atari, StarCraft II, Dota 2
- **Era of Human Data:** GPT-3, ChatGPT, Large Language Models
- **Era of Experience:** AlphaProof, Autonomous scientific discovery

Why Do We Need the Era of Experience?

Limitations of Human Data

- High-quality data sources nearly exhausted
- Cannot exceed human knowledge boundaries
- Progress demonstrably slowing down

Advantages of Experiential Learning

- **Self-improvement:** Data quality improves with agent capability
- **Breaking boundaries:** Discover new theorems, technologies, breakthroughs
- **Scale advantage:** Experience will dwarf human data volume

Example: AlphaProof generated 100 million formal proofs through RL, achieving medal-level performance at the IMO.

Four Pillars of the Era of Experience

☰ Streams of Experience

- Continuous lifelong learning
- Adaptation over months/years
- Self-correction and improvement

⚡ Grounded Rewards

- Environmental feedback
- Not human prejudgment
- Discover novel strategies

🎯 Grounded Actions

- Rich sensorimotor interaction
- Autonomous exploration
- Beyond human interfaces

🧠 Experience-Based Reasoning

- World model construction
- Planning from consequences
- Beyond human thought patterns

Verification as the Key

Sutton's Insight

"The insight that I would claim to have is that the key to a successful AI is that it can tell for itself whether or not it is working correctly."

— Richard Sutton, *Self-Verification, The Key to AI*

From Human Feedback to Self-Verification

- **Traditional RLHF:** Limited by human annotation capacity and quality
- **Self-Verification:** Models can autonomously evaluate their outputs
- **Key Advantage:** Enables infinite learning cycles without human bottlenecks

Verification enables the transition from human-limited to experience-unlimited learning

Deliberative Alignment: A New Paradigm

"I propose to consider the question, '*Can machines think?*'"

— Alan M. Turing, 1950,
Computing Machinery and Intelligence



Deliberative Alignment [61]

Directly teach models safety specifications and train them to **explicitly reason** over these specifications before answering

Traditional Approach:

- Learn from labeled examples
- Implicit pattern recognition
- Fixed compute per response

Deliberative Alignment:

- Learn actual safety policies
- Explicit reasoning via CoT
- Variable inference compute

Deliberative Alignment vs. Traditional RLHF

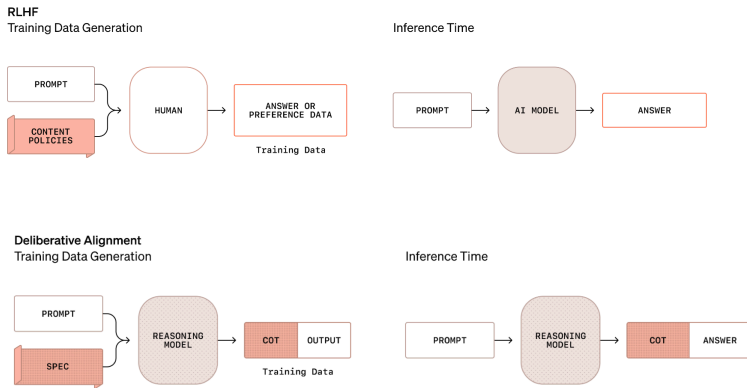


Fig. 56. Comparison of deliberative alignment and traditional RLHF [61]. In RLHF, there is no reasoning during inference time.

Two-Stage Training Process

1. Supervised Fine-Tuning (SFT) Stage

Goal: Teach model to directly reference safety specifications in its CoT.

Method: Collect *(prompt, CoT, output)* tuples for supervised fine-tuning.

- CoT references safety specs.
- Context Distillation generates data.

Result: SFT provides strong prior.

2. Reinforcement Learning (RL) Stage

Goal: Train model to reason more effectively.

Method: Use high-compute RL with judge LLM (G_{RM}) providing reward signal.

- G_{RM} has access to safety specs.

Result: Optimized reasoning process, more aligned with specs.

Deliberative Alignment vs. Traditional RLHF

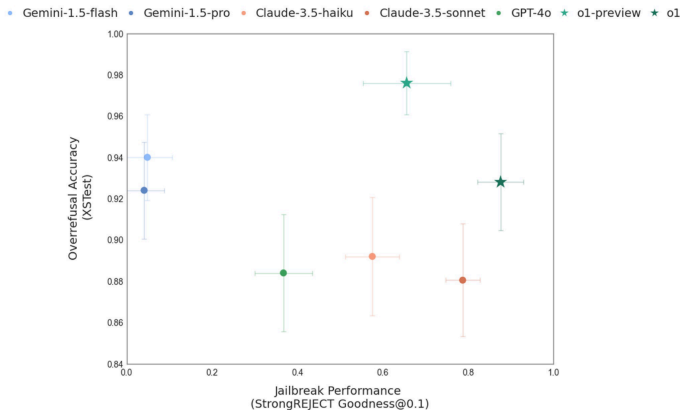


Fig. 57. Main safety results. The o1 models advance the Pareto frontier of refusing to answer malicious jailbreak prompts and not over-refusing benign prompts, compared to GPT-4o and other state-of-the-art LLMs [61].

Impact of Test-Time Compute

Findings:

- More compute → better safety
- Especially for complex tasks
- Validates test-time reasoning

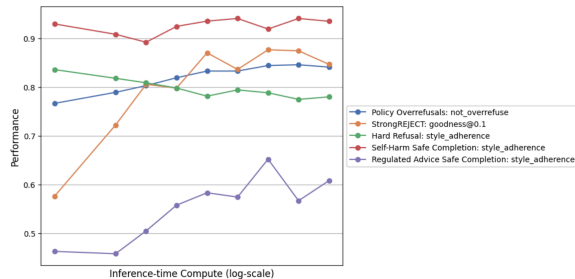


Fig. 58. Impact of inference-time compute on model performance. The o1 model has stronger performance on challenging tasks. More compute to spend on reasoning [61].

Overview

5 Alignment with Verifiers

- Era of Experience
- Test-time Scaling Law
- Verifiable Rewards
- Process Rewards

Human Thinking vs AI Reasoning



Human Thinking

Difficult problems Longer thinking
Deep reasoning Better decisions



AI Reasoning

More compute Multiple attempts
Search & verify Optimized output

How can AI "think harder" like humans?

Question: Given a fixed inference-time compute budget, how much can an LLM improve its performance on challenging prompts?

Scaling Law

Test-time Scaling Law [62]: Model performance improves logarithmically with increased inference-time computation, allowing the model to think longer.



Mechanisms for Scaling Test-time Compute [62]

1. Search Against Verifiers

- Process Reward Models (PRM)
- Beam search
- Lookahead search
- Best-of-N sampling

2. Refining Proposal Distribution

- Sequential revisions
- Self-critique & improvement
- Iterative refinement
- Learning from mistakes

💡 **Key Insight:** Different approaches work better for different problem difficulties

Test-time vs. Pretraining Compute

FLOPs-Matched Evaluation

Compare smaller model + test-time compute vs. 14x larger pretrained model

Test-time compute wins when:

- Easy/medium questions
- Low inference requirements
- $R = \frac{D_{\text{inference}}}{D_{\text{pretrain}}} \ll 1$

Pretraining wins when:

- Hard questions
- High inference load
- $R = \frac{D_{\text{inference}}}{D_{\text{pretrain}}} \gg 1$

Note: Test-time and pretraining compute are NOT 1-to-1 exchangeable

Overview

5 Alignment with Verifiers

- Era of Experience
- Test-time Scaling Law
- **Verifiable Rewards**
- Process Rewards

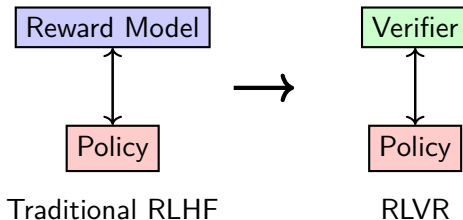
RLVR [63]: Beyond Traditional RLHF

Traditional RLHF Limitations

- Requires training reward models
- Susceptible to reward hacking
- Complex preference data needed
- High computational overhead

RLVR Innovation

- Uses **verifiable** rewards
- Binary correctness signals
- No reward model needed
- Applied to math, coding, IF tasks



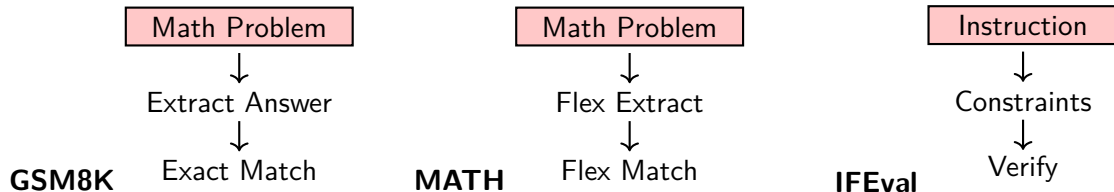
RLVR Objective and Verification

RLVR Optimization Objective

$$\max_{\pi_{\theta}} \mathbb{E}_{y \sim \pi_{\theta}(x)} [v(x, y) - \beta \text{KL}[\pi_{\theta}(y|x) || \pi_{\text{ref}}(y|x)]]$$

where verifiable reward function:

$$v(x, y) = \begin{cases} \alpha & \text{if correct} \\ 0 & \text{otherwise} \end{cases}$$



RLVR vs Traditional Approaches

Best Practices

- Initialize value from general RM
- Use verifiable rewards only (not RM scores)
- Start from stronger base models
- Monitor for overoptimization
- Careful hyperparameter tuning

When to Use RLVR

- Tasks with clear correctness criteria
- Domains where verification is feasible
- When reward hacking is a concern
- As part of comprehensive training pipeline

DeepSeek R1 Series

Challenges

- LLMs still struggle with complex reasoning
- OpenAI o1 showed potential of test-time scaling
- Lack of open-source reasoning models
- Limited understanding of RL's role

Objectives

- Explore pure RL for reasoning
- Develop open-source alternative to o1
- Distill capabilities to smaller models
- Advance AI alignment research

Key Innovation: First to validate that pure RL can incentivize LLM reasoning without SFT

GRPO: RL Algorithm behind DeepSeek R1

Group Relative Policy Optimization (GRPO) [64] also leverages multiple samples, but **keeps most features of PPO**.

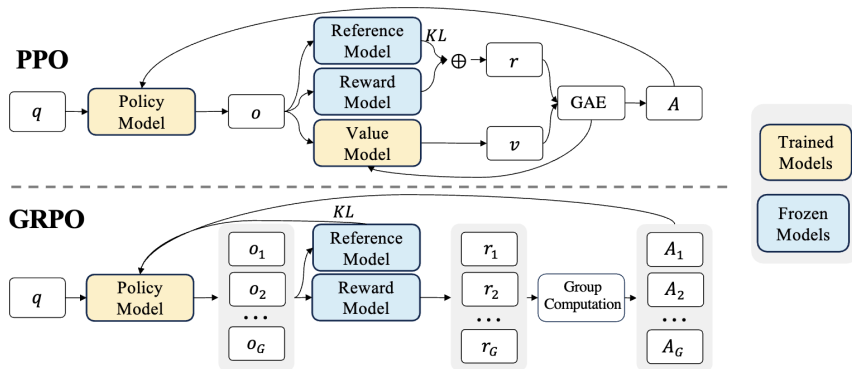


Fig. 59. GRPO foregoes the value model, instead estimating the baseline from group scores [64].

GRPO: RL Algorithm behind DeepSeek R1

Specifically, GRPO optimizes the following objective:

$$\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{old}} \left\{ \min \left[\frac{\pi_{\theta}}{\pi_{old}} A^{GRPO}, \text{clip} \left(\frac{\pi_{\theta}}{\pi_{old}}, 1 - \varepsilon, 1 + \varepsilon \right) A^{GRPO} \right] - \beta \mathbb{D}_{KL}[\pi_{\theta} || \pi_{ref}] \right\},$$

where A^{GRPO} is estimated as:

$$A^{GRPO}(y, x) = \hat{r}(y, x) = \frac{r(y, x) - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}.$$

GRPO estimates the KL divergence with the following unbiased estimator, which is **guaranteed to be positive**:

$$\mathbb{D}_{KL}[\pi_{\theta} || \pi_{ref}] = \frac{\pi_{ref}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta}(o_{i,t} | q, o_{i,<t})} - \log \frac{\pi_{ref}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta}(o_{i,t} | q, o_{i,<t})} - 1,$$

GRPO as Adaptive Contrastive Loss [65]

Standard GRPO Objective

$$\max_{\theta} \mathbb{E}_{q \sim \rho_Q} \mathbb{E}_{o \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \frac{\pi_{\theta}(o|q)}{\pi_{\theta_{\text{old}}}(o|q)} A(q, o) - \beta \text{KL}(\pi_{\theta} || \pi_{\text{ref}})$$

Advantage Function with Verifiable Rewards

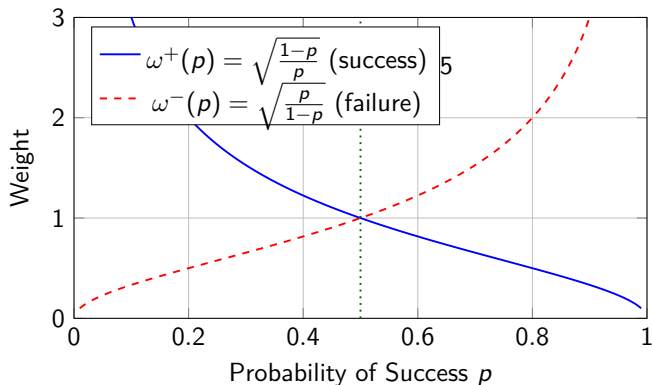
For binary reward $r(q, o) \in \{0, 1\}$:

$$A(q, o) = \begin{cases} \sqrt{\frac{1-p}{p}} & \text{if } r(q, o) = 1 \\ -\sqrt{\frac{p}{1-p}} & \text{if } r(q, o) = 0 \end{cases}$$

where $p = \mathbb{P}_{o \sim \pi_{\text{old}}}(\cdot|q)(r(q, o) = 1)$

Key Insight: This creates an **adaptive weighting scheme** based on success probability!

GRPO as Adaptive Contrastive Loss



When $p < 0.5$:

- High weight on successes
- Low weight on failures

When $p > 0.5$:

- Low weight on successes
- High weight on failures

GRPO Policy Evolution and Fixed Point Iteration [65]

Theorem (GRPO Policy Dynamics)

The optimal GRPO policy at iteration n is:

$$\pi_n(o|q) = \frac{1}{Z_{n-1}(q)} \pi_{\text{ref}}(o|q) \exp \left(\frac{1}{\beta} [\omega_{\varepsilon}^{+}(p_{n-1}) \mathbf{1}_{r=1} - \omega_{\varepsilon}^{-}(p_{n-1}) \mathbf{1}_{r=0}] \right)$$

Theorem (Probability of Success Fixed Point)

The probability of success satisfies:

$$p_n = h_{\varepsilon, p_{\text{ref}}}(p_{n-1})$$

where

$$h_{\varepsilon, p_{\text{ref}}}(p) = \frac{1}{1 + \frac{1-p_{\text{ref}}}{p_{\text{ref}}} \exp \left(-\frac{1}{\beta \sqrt{p(1-p)+\varepsilon}} \right)}$$

GRPO Success Amplification Theorem [65]

Theorem (GRPO Amplifies Success)

Let $0 < p_{\text{ref}} < 1$. Any fixed point p^* of $h_{\epsilon, p_{\text{ref}}}$ satisfies $p^* > p_{\text{ref}}$ if:

- ① $p_{\text{ref}} \leq \frac{1}{2}$ for all $\beta > 0$
- ② $p_{\text{ref}} > \frac{1}{2}$

$$\beta \cosh^2 \left(\frac{1}{2\beta \sqrt{\frac{1}{4} + \epsilon}} \right) \geq \frac{p_{\text{ref}}(1 - p_{\text{ref}})(2p_{\text{ref}} - 1)}{2[p_{\text{ref}}(1 - p_{\text{ref}}) + \epsilon]^{3/2}}$$

Convergence Condition

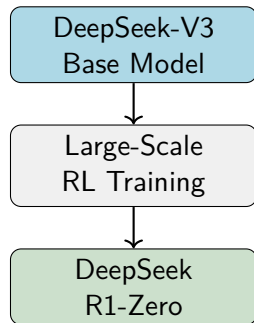
Local convergence to fixed point p^* occurs when $|h_{\epsilon, p_{\text{ref}}}| < 1$, i.e.,

$$\beta > \frac{p^*(1 - p^*)|2p^* - 1|}{2[p^*(1 - p^*) + \epsilon]^{3/2}}$$

DeepSeek-R1-Zero [66]: Pure Reinforcement Learning

Core Components

- **GRPO**: Group Relative Policy Optimization
- **Rule-based Rewards**:
 - Accuracy rewards (math verification)
 - Format rewards (thinking process)
- **Simple Template**: Guide reasoning
- **No SFT**: Direct RL on base model



Emergent Behaviors

The model spontaneously learned to reflect, self-verify, and explore alternative approaches!

The "Aha Moment" Discovery

Fascinating Observation

During RL training, the model spontaneously learned to re-evaluate its approach:

Wait, wait. Wait. That's an aha moment I can flag here.
Let's reevaluate this step-by-step to identify if the
correct sum can be...

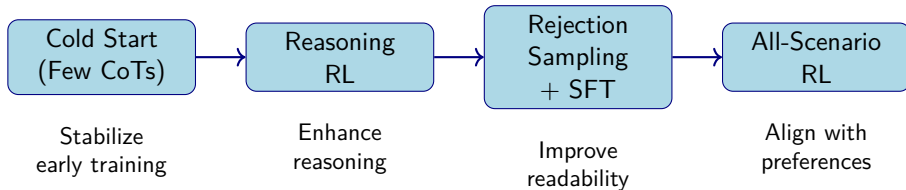
Significance

- Self-directed learning
- No explicit instruction
- Emergent meta-cognition

Implications

- RL discovers reasoning patterns
- Models can self-improve
- Path to more autonomous AI

DeepSeek-R1: Multi-Stage Training Pipeline



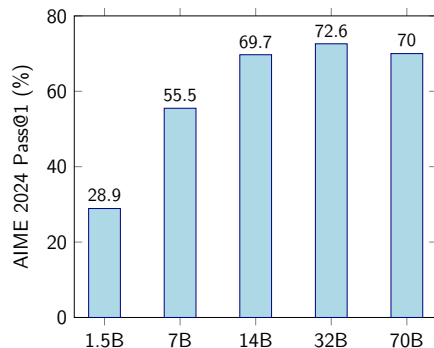
Key Improvements over R1-Zero

- Language consistency rewards to reduce mixing
- Combined reasoning and non-reasoning data
- Helpfulness and harmlessness reward models

Distillation Results

Key Findings

- **7B model** outperforms QwQ-32B
- **14B model** surpasses all comparable models
- **32B/70B models** approach o1-mini
- Distillation > Direct RL on small models



Important Discovery

Reasoning patterns from larger models transfer effectively to smaller ones through distillation!

Logic-RL [67]: Motivation

Research Questions

- Can reasoning abilities emerge in smaller models?
- What is the optimal training data structure?
- How to reliably replicate DeepSeek-R1's results?

Key Challenge

Current datasets (GSM8K, Omni-MATH) have uncontrolled variance in problem complexity

Knights and Knaves: Problem Structure

Problem Example

Setup: Island with Knights (truth) and Knaves (lies)

Statements:

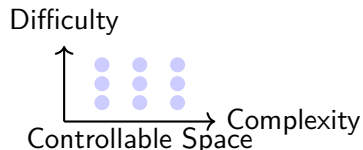
- ① Zoey: "Oliver is not a knight"
- ② Oliver: "Oliver is a knight iff Zoey is a knave"

Solution:

- Zoey is a knave
- Oliver is a knight

Why K&K Puzzles?

- **Procedural Generation** - Infinite variants
- **Controlled Difficulty** - 2-8 people, 1-4 operators
- **Unambiguous Solutions** - Binary verification
- **Pure Logic** - No domain knowledge needed



Rule-Based Reward Design

Format Reward Criteria

- Tags appear exactly once
- Correct sequential order
- Genuine reasoning in `<think>`
- Extractable answer format

Reward Hacking Prevention

Observed hacking behaviors:

- Skipping thinking process
- Reasoning in answer tag
- Repeated guessing
- Nonsense padding
- Revisiting after answer

Reward Function

$$s_{\text{format}} = \begin{cases} 1 & \text{if correct} \\ -1 & \text{if incorrect} \end{cases} \quad (21)$$

$$s_{\text{answer}} = \begin{cases} 2 & \text{if fully correct} \\ -1.5 & \text{if partial} \\ -2 & \text{if missing} \end{cases} \quad (22)$$

Reinforce++ [68]

Key Modifications

① Token-level KL Penalty

$$r(s_t, a_t) = \mathbb{I}(s_t = [\text{EOS}])r(x, y) - \beta \text{KL}(t)$$

② Unbiased KL Estimator

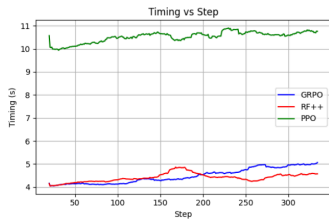
$$D_{\text{KL}} = \frac{\pi_{\text{ref}}}{\pi_{\theta}} - \log \frac{\pi_{\text{ref}}}{\pi_{\theta}} - 1$$

Always **non-negative!**

③ Global Batch Mean Reward as Baseline

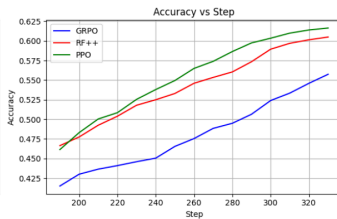
$$A_{q,o_t}^{\text{norm}} = \frac{A_{q,o_t} - \text{mean}(A_{q,o_t})}{\text{std}(A_{q,o_t})}$$

RL Algorithm Comparison



GRPO

Weakest performance
Less stable convergence



REINFORCE++

Best balance
Stable & efficient



PPO

Highest accuracy
138% slower

SFT Memorizes, RL Generalizes [67]

Key Insights

- **RL:** Higher test accuracy with **minimal memorization**
- **RFT:** Slight improvement but **heavy memorization**
- Within same LiMem range, RL vastly outperforms, suggesting better generalization ability

Memorization Score

$$\text{LiMem}(f; D) = \text{Acc}(f; D) \cdot (1 - \text{CR}(f; D))$$

where CR = consistency ratio on perturbations

Key Findings

What Works

- ✓ Thinking tokens ("verify", "yet") improve reasoning
- ✓ RL generalizes better than SFT
- ✓ REINFORCE++ outperforms GRPO
- ✓ Curriculum learning provides marginal benefits

What Doesn't

- ✗ Longer responses \neq better reasoning
- ✗ Language mixing hinders performance
- ✗ Some tokens ("recheck") decrease accuracy
- ✗ No sudden "aha moment" observed

Key Insight: SFT memorizes, RL generalizes
RL develops abstract problem-solving schemata

Overview

- 5 Alignment with Verifiers
 - Era of Experience
 - Test-time Scaling Law
 - Verifiable Rewards
 - Process Rewards

Process Reward Model

- Traditional reward models use **outcome-based supervision**, i.e., **Outcome Supervised Reward Models (ORMs)** [69], where only the final result is evaluated and receives feedback. However, **in reasoning tasks**, ORM targets is not perfectly reliable: **false positives solutions that reach the correct answer with incorrect reasoning will be misgraded**.
- In contrast, **process supervised reward models (PRMs)** [69] **provide feedback for each step in the chain-of-thought (COT)**. It is easier for humans to interpret and it more directly rewards models for following a human-endorsed COT.

Process Reward Model

The solution on the left is correct while the solution on the right is incorrect. A green background indicates a high PRM score, and a red background indicates a low score. The PRM correctly identifies the mistake in the incorrect solution.

<p>Let</p> $x^5 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$ <p>where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.</p> <p>I notice that the given polynomial has even degree and only even powers of x, so I can try to make a substitution to simplify it.</p> <p>Let $y = x^2$, then the polynomial becomes $y^2 + 3y - 4$, which is a quadratic equation.</p> <p>I can factor this quadratic equation as $(y + 4)(y - 1)$, so the original polynomial is $(x^2 + 4)(x^2 - 1)$.</p> <p>Now I need to factor each of these factors further, if possible, over the integers.</p> <p>For the first factor, $x^2 + 4$, I recall the Sophie Germain identity, which says that $a^4 + 4b^4 = (a^2 + 2b^2 + 2ab)(a^2 + 2b^2 - 2ab)$.</p> <p>Applying this identity with $a = x$ and $b = 1$, I get $x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2)$.</p> <p>For the second factor, $x^2 - 1$, I can use the difference of squares formula, which says that $a^2 - b^2 = (a + b)(a - b)$.</p> <p>Applying this formula with $a = x^2$ and $b = 1$, I get $x^4 - 1 = (x^2 + 1)(x^2 - 1)$.</p> <p>Then I can apply the difference of squares formula again with $a = x$ and $b = 1$, to get $x^2 - 1 = (x + 1)(x - 1)$.</p> <p>So the original polynomial is $(x^2 + 4)(x^2 - 1) = (x^2 + 2x + 2)(x^2 - 2x + 2)(x^2 + 1)(x + 1)(x - 1)$.</p> <p>These are all the irreducible factors over the integers, so $p_1(x) = x^2 + 2x + 2$, $p_2(x) = x^2 - 2x + 2$, $p_3(x) = x^2 + 1$, $p_4(x) = x + 1$, and $p_5(x) = x - 1$.</p> <p>To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.</p> <p>I get $p_1(1) = 1^2 + 2 \cdot 1 + 2 = 5$, $p_2(1) = 1^2 - 2 \cdot 1 + 2 = 1$, $p_3(1) = 1^2 + 1 = 2$, $p_4(1) = 1 + 1 = 2$, and $p_5(1) = 1 - 1 = 0$.</p> <p>So the final answer is $5 + 1 + 2 + 2 + 0 = 10$.</p> <p>Answer: 10</p>	<p>Let</p> $x^5 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$ <p>where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.</p> <p>This problem is about factoring a polynomial over the integers, which can be a tricky task. One possible strategy is to look for rational roots of the polynomial, using the rational root theorem.</p> <p>However, that would involve checking a lot of possible factors of -4, and there is no guarantee that the polynomial has any rational roots at all.</p> <p>Another possible strategy is to use some algebraic tricks to simplify the polynomial or reduce its degree.</p> <p>I notice that the polynomial has only even powers of x, so I can try to substitute $y = x^2$ and see what happens.</p> <p>Then the polynomial becomes $y^2 + 3y^2 - 4$, which is a quadratic equation in y.</p> <p>I can try to factor this quadratic equation by looking for two numbers that multiply to -4 and add to 3.</p> <p>Those numbers are 4 and -1, so I can write $y^2 + 3y^2 - 4 = (y^2 + 4)(y^2 - 1)$.</p> <p>Now I can substitute back $x^2 = y$ and get $x^4 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1)$.</p> <p>This looks promising, because I have reduced the degree of the factors from 5 to 4.</p> <p>I can try to factor each of these factors further, using the difference of squares formula.</p> <p>I get $x^4 + 3x^4 - 4 = (x^2 + 2)(x^2 - 2)(x^2 + 1)(x^2 - 1)$.</p> <p>I can apply the difference of squares formula again to the last factor and get $x^4 + 3x^4 - 4 = (x^2 + 2)(x^2 - 2)(x^2 + 1)(x + 1)(x - 1)$.</p> <p>Now I have factored the polynomial completely into monic linear and quadratic factors with integer coefficients.</p> <p>These are the $p_i(x)$'s that the problem is asking for.</p> <p>To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.</p> <p>I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (1^2 + 2)(1^2 - 2)(1^2 + 1)(1 + 1)(1 - 1)$.</p> <p>Simplifying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (3)(-1)(2)(2)(0)$.</p> <p>Multiplying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = 0$.</p> <p>Answer: 0</p>
---	--

Fig. 60. Two solutions to the same problem, graded by the PRM [69].

rStar-Math [70]: Overview

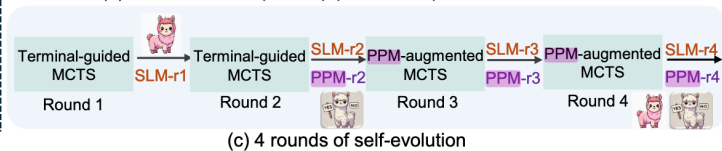
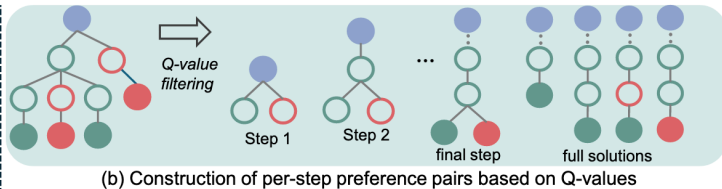
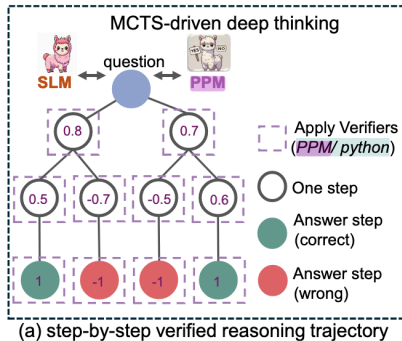


Fig. 61. Overview of rStar-Math, which is a self-evolvable System 2-style reasoning approach that achieves the state-of-the-art math reasoning.

Monte Carlo Tree Search for Math Reasoning

MCTS Process

- 1 **Selection:** UCT-based node selection
- 2 **Expansion:** Generate code-augmented steps
- 3 **Verification:** Execute Python code
- 4 **Back-propagation:** Update Q-values

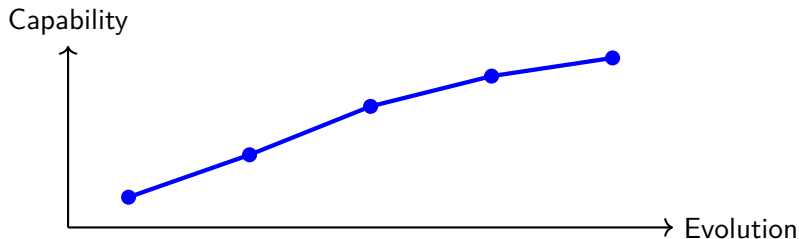
Key Advantage

- Breaks complex problems into simple steps
- Automatic Q-value annotation
- Eliminates erroneous intermediate steps

$$\text{UCT}(s) = Q(s) + c \sqrt{\frac{\ln N_{\text{parent}}(s)}{N(s)}}$$

Four Rounds of Self-Evolution

Round	Policy Model	Reward Model	Method	Problems Solved	Key Achievement
1	DeepSeek-236B	None	Terminal-guided	60.17%	Bootstrap SLM
2	SLM-r1 (7B)	PPM-r1	Terminal-guided	66.60%	Reliable PPM
3	SLM-r2 (7B)	PPM-r2	PPM-augmented	77.86%	Quality boost
4	SLM-r3 (7B)	PPM-r3	PPM-augmented	90.25%	Olympiad-level



Emergent Capabilities

Self-Reflection

- Model recognizes mistakes
- Backtracks to find better solutions
- No explicit self-reflection training
- Emerges from deep thinking

PPM Insights

- Identifies theorem applications
- Fermat's Little Theorem
- Vieta's formulas
- AM-GM inequality
- Guides toward correct paths

Key Insight

PPM shapes the reasoning boundary - Once policy model is reasonably strong, PPM becomes the key determinant of upper performance limit.

Challenge of Process Reward Models

- **Outcome Reward Models (ORMs):**
Evaluate entire responses
- **Process Reward Models (PRMs):**
Score each reasoning step
- PRMs provide:
 - Denser, fine-grained rewards
 - Better transparency & interpretability
 - Superior performance in best-of-N sampling

The Problem

Training PRMs requires **step-level annotations**, which are:

- Expensive to collect (38.8x more FLOPs)
- Prone to annotation noise

Implicit PRMs [71] from ORMs

Get PRM for Free from ORM

An implicit PRM can be obtained **at no additional cost** by training an ORM with a specific reward parameterization.

Reward Parameterization:

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

Process Reward:

$$r_{\theta}^t = q_{\theta}^t - q_{\theta}^{t-1} = \sum_{i=t-1}^t \beta \log \frac{\pi_{\theta}(y_i | y_{<i})}{\pi_{\text{ref}}(y_i | y_{<i})}$$

Key Properties:

- Works with various objectives (DPO, KTO, NCA, CE)
- No step-level annotations needed
- Learns Q-function implicitly
- More accurate than MCTS-based approaches

Theoretical Foundation

Theorem (Implicit Q-function Learning)

When parameterizing the outcome reward as $r_\theta(y) = \beta \log \frac{\pi_\theta(y)}{\pi_{\text{ref}}(y)}$, the model implicitly learns:

$$q_\theta^t(y_{<t}, y_t) = \beta \log \mathbb{E}_{\pi_{\text{ref}}(y|y_{\leq t})} e^{\frac{1}{\beta} r_\theta(y)}$$

Advantages over MCTS

- **Bounded accuracy:** $q_{\theta_s}^t \leq q_\theta^t \leq q_{\theta_h}^t$
- Mitigates overestimation (hard) and underestimation (soft) issues
- No sampling noise

Compatible Objectives

- Direct Preference Optimization (DPO)
- Kahneman-Tversky Optimization (KTO)
- Noise Contrastive Alignment (NCA)
- Cross-Entropy (CE) Loss

PRIME [72]: Overview

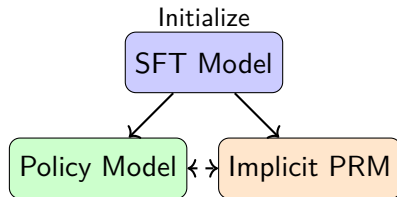
Process Reinforcement through Implicit Rewards

Key Innovation:

- Use **implicit process rewards** from outcome labels only
- Enable **online PRM updates** without step-level annotations
- **Scalable** and **efficient** dense reward framework

Core Formula:

$$r_{\phi}(y_t) = \beta \log \frac{\pi_{\phi}(y_t|y_{<t})}{\pi_{ref}(y_t|y_{<t})}$$



Online Update

How PRIME Works

Algorithm Flow:

- 1 Initialize policy and PRM from SFT
- 2 Sample responses from policy
- 3 Get outcome rewards from verifier
- 4 Calculate implicit process rewards
- 5 Update PRM with outcome labels only
- 6 Compute advantages with dense rewards
- 7 Update policy with PPO

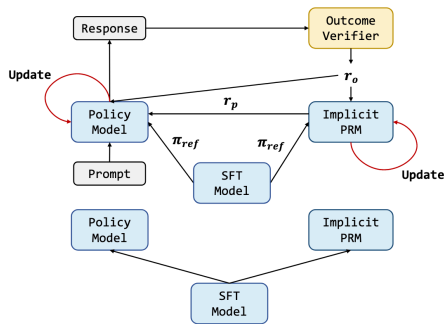


Fig. 62. Illustration of PRIME.

Efficiency and Performance

Sample Efficiency:

- 2.5 more efficient than outcome-only rewards
- Faster convergence to high performance
- Lower variance during training

Online PRM Update Benefits:

- Prevents reward hacking
- Maintains high PRM accuracy
- No need for step-level annotations

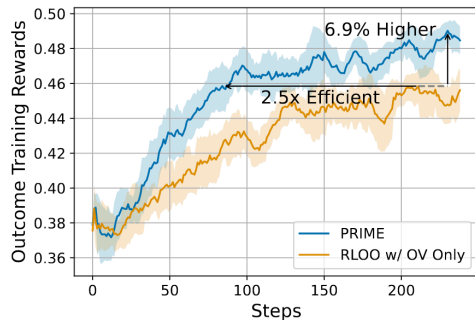


Fig. 63. 2.5 efficiency gain and 6.9% higher final performance

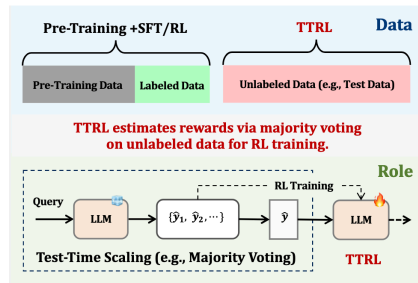
TTRL [73]: Test-Time Reinforcement Learning

Current Limitations:

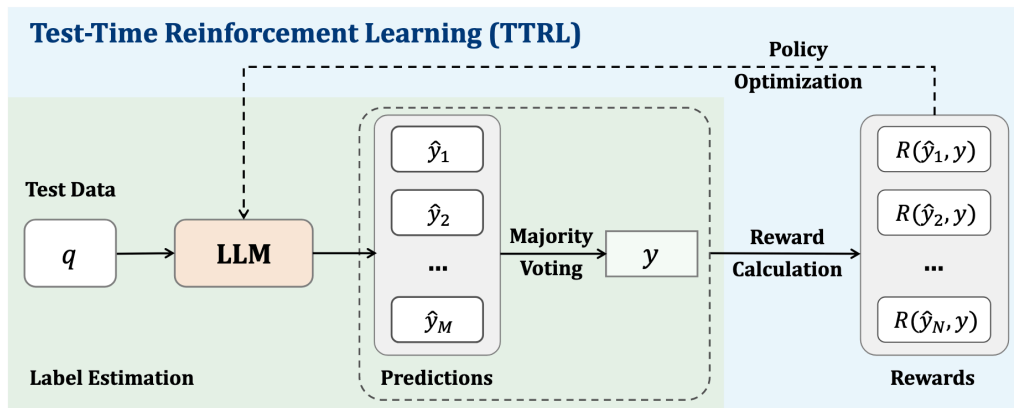
- Large Reasoning Models (e.g., o1, DeepSeek-R1) require expensive labeled data
- Complex unlabeled questions continuously emerge
- o3 achieves 75.7% on ARC-AGI-1 but only 4% on ARC-AGI-2

The Vision:

- Enable AI systems to **self-evolve** through experience
- Learn from unlabeled test data
- Push the boundaries of AI capabilities



TTRL: Test-Time Reinforcement Learning



Key Innovation: Use majority voting to estimate labels and compute rewards without ground truth

TTRL Methodology

Algorithm:

- ① Generate multiple outputs $\{y_1, \dots, y_N\}$ from $\pi_\theta(y|x)$
- ② Derive consensus output y^* via **majority voting**
- ③ Compute rewards:

$$r(y_i, y^*) = \begin{cases} 1, & \text{if } y_i = y^* \\ 0, & \text{otherwise} \end{cases}$$

- ④ Update policy:

$$\theta \leftarrow \theta + \eta \nabla_\theta \mathbb{E}_{y \sim \pi_\theta} [r(y, y^*)]$$

Why Does It Work?

- **Label Estimation:** Majority voting provides reliable pseudo-labels
- **Reward Robustness:** "Lucky Hit" phenomenon ensures high reward accuracy
- **Online Learning:** Dynamic improvement of supervision quality

ProRL [74]: The Central Question

Core Research Question

Does reinforcement learning truly expand a model's reasoning capabilities, or does it merely amplify high-reward outputs already latent in the base model's distribution?

Previous Claims:

- RL doesn't acquire new reasoning capabilities
- RL converges toward dominant output distributions
- Limited by short training periods

ProRL Hypothesis:

- Prolonged RL training can discover novel reasoning strategies
- Extended training explores new solution spaces
- Diverse tasks enable better generalization

Limitations of Previous Work

Methodological Constraints

- ➊ **Narrow Domain Focus:** Overreliance on mathematics where models are often overtrained
- ➋ **Premature Termination:** RL training limited to hundreds of steps
- ➌ **Limited Task Diversity:** Restricted evaluation across reasoning domains

ProRL's Approach

- **Extended Training:** More than 2,000 training steps
- **Diverse Tasks:** Math, coding, STEM, logic puzzles, instruction following
- **Stable Training:** KL divergence control + reference policy resetting

ProRL: Implementation Details

Key Components

- 1 **Base Algorithm:** Group Relative Policy Optimization (GRPO)
- 2 **Entropy Preservation:** Decoupled clipping + dynamic sampling
- 3 **Stability Control:** KL divergence penalty + reference policy reset

$$L_{KL-RL}(\theta) = L_{GRPO}(\theta) - \beta D_{KL}(\pi_{\theta} || \pi_{ref})$$

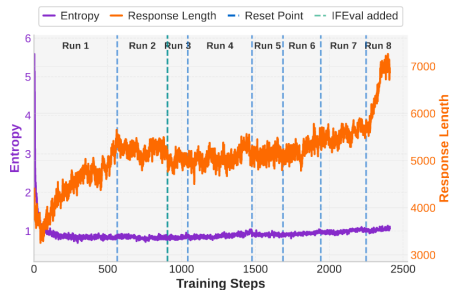
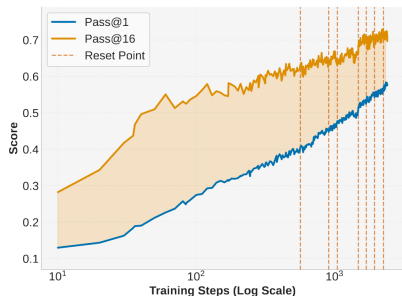
Training Data (136K examples):

- Math: 40K problems
- Code: 24K problems
- STEM: 25K problems
- Logic Puzzles: 37K problems
- Instruction Following: 10K problems

Training Setup:

- 16k GPU hours on H100s
- Periodic reference resets
- Context: 8K 16K tokens
- Temperature: 1.2 (rollout)

Training Dynamics



- **Sustained Improvement:** Both Pass@1 and Pass@16 continue scaling
- **Entropy Preservation:** Avoids collapse through multiple techniques
- **Strategic Resets:** 8 training runs with periodic reference policy resets

Does ProRL Discover New Reasoning Patterns?

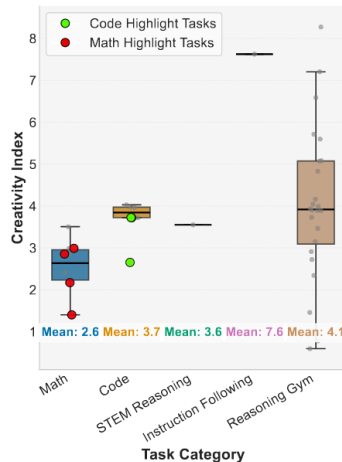
Evidence for Novel Reasoning:

- **Creativity Index:** Higher novelty in reasoning trajectories
- **Low-to-High Performance:** Dramatic improvements on initially challenging problems
- **OOD Generalization:** Strong performance on unseen tasks

Key Pattern:

The Weaker the Start, the Stronger the Gain

RL expands reasoning boundaries most effectively where base models initially struggle



Three Training Regimes Identified

- ① **Diminished:** Reduced diversity in high-performance domains
 - Pass@1 improves, Pass@128 decreases
 - Model becomes more confident but less exploratory
- ② **Plateau:** Early saturation of RL benefits
 - Both Pass@1 and Pass@128 improve initially
 - Gains plateau with continued training
- ③ **Sustained:** Continued boundary expansion
 - Consistent improvements with prolonged training
 - Most evident in complex coding and novel reasoning tasks

Out-of-Distribution Performance

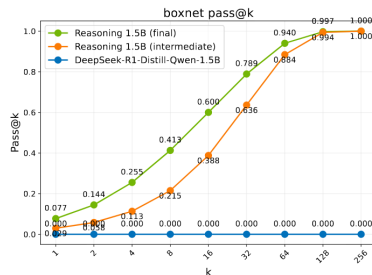
Boxnet Task Example:

- Base model: 0% success rate
- RL model: 7.7% 58.6% (OOD)
- Demonstrates genuine capability expansion

Graph Coloring Scalability:

- Trained on 10-node graphs
- Tested on larger graphs (15-20 nodes)
- Maintains superior performance

Pass@k Improvements



Significance

ProRL enables models to internalize **abstract reasoning patterns** that generalize beyond training distribution

Distribution Shifts in Reasoning

Codeforces Problems:

- Initial: Concentrated near zero
- Post-RL: Broader, higher success rates
- Sustained exploration benefits

Family Relationships:

- Base: Predominantly zero accuracy
- RL: Peak at perfect accuracy
- Novel reasoning challenge solved

Mathematical Upper Bound Analysis

$$E_{x,y \sim D}[\text{pass@k}] \leq 1 - [(1 - E_{x,y \sim D}[\rho_x])^2 + \text{Var}(\rho_x)]^{k/2} \quad (23)$$

ProRL generates sufficient ρ_x improvement to overcome variance increases, unlike previous observations of declining pass@k during training.

Key Takeaways

Main Findings

- 1 **ProRL discovers novel reasoning strategies** beyond base model capabilities
- 2 **Extended stable training** is crucial for reasoning boundary expansion
- 3 **Task diversity** enables robust generalization across domains
- 4 **Strategic training techniques** (KL control, reference resets) enable prolonged optimization

Empirical Evidence:

- 2,000+ training steps
- 136K diverse problems
- State-of-the-art 1.5B model
- Strong OOD performance

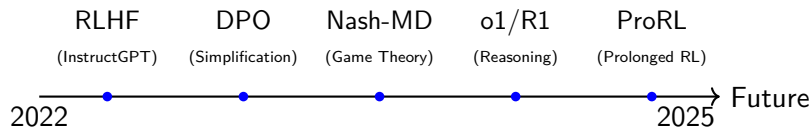
Broader Impact:

- Reaffirms RL value for reasoning
- Enables smaller, capable models
- Opens new research directions
- Challenges current limitations

Conclusion: The Evolution of Language Model Alignment

Key Takeaways

- **RLHF**: Pioneered human preference learning but faces scalability challenges
- **Direct Alignment**: DPO/IPO simplify training while maintaining effectiveness
- **General Preferences**: Nash equilibrium approaches handle diverse human values
- **Reasoning Models**: Test-time scaling and process rewards unlock new capabilities



The future of AI alignment lies in models that can learn, reason, and improve autonomously

Thanks!

References I



Wikipedia.

AI alignment — Wikipedia, the free encyclopedia.

<http://en.wikipedia.org/w/index.php?title=AI%20alignment&oldid=1164585294>, 2023.

[Online; accessed 15-July-2023].



Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al.

A general language assistant as a laboratory for alignment.

arXiv preprint arXiv:2112.00861, 2021.



Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al.

Ai alignment: A comprehensive survey.

arXiv preprint arXiv:2310.19852, 2023.



Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.

Training language models to follow instructions with human feedback.

Advances in Neural Information Processing Systems, 35:27730–27744, 2022.



Nathan Lambert, Thomas Krendl Gilbert, and Tom Zick.

The history and risks of reinforcement learning and human feedback.

arXiv preprint arXiv:2310.13595, 2023.



David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton.

Reward is enough.

Artificial Intelligence, 299:103535, 2021.

References II



Ralph Allan Bradley and Milton E Terry.

Rank analysis of incomplete block designs: I. the method of paired comparisons.
Biometrika, 39(3/4):324–345, 1952.



John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.

Proximal policy optimization algorithms.
arXiv preprint arXiv:1707.06347, 2017.



Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash.

Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback, 2024.



Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al.

Constitutional ai: Harmlessness from ai feedback.
arXiv preprint arXiv:2212.08073, 2022.



Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al.

Open problems and fundamental limitations of reinforcement learning from human feedback.
arXiv preprint arXiv:2307.15217, 2023.



David Manheim and Scott Garrabrant.

Categorizing variants of goodhart's law.
arXiv preprint arXiv:1803.04585, 2018.

References III



Leo Gao, John Schulman, and Jacob Hilton.

Scaling laws for reward model overoptimization.

In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.



Manuela Cattelan.

Models for paired comparison data: A review with emphasis on dependent data.

Statistical Science, 27(3), August 2012.



Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett.

A long way to go: Investigating length correlations in rlhf.

arXiv preprint arXiv:2310.03716, 2023.



Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You.

Colossal-ai: A unified deep learning system for large-scale parallel training.

In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 766–775, 2023.



Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo.

Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models.

In *Forty-first International Conference on Machine Learning*, 2023.



Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour.

Policy gradient methods for reinforcement learning with function approximation.

Advances in neural information processing systems, 12, 1999.

References IV



Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker.

Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms.

arXiv preprint arXiv:2402.14740, 2024.



Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh.

Mirror descent policy optimization.

arXiv preprint arXiv:2005.09814, 2020.



Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al.

Apple intelligence foundation language models.

arXiv preprint arXiv:2407.21075, 2024.



OpenAI.

Gpt-4 technical report.

arXiv preprint arXiv:2303.08774, 2023.



Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang.

Safe rlhf: Safe reinforcement learning from human feedback.

arXiv preprint arXiv:2310.12773, 2023.



Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, et al.

Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales.

arXiv preprint arXiv:2308.01320, 2023.

References V



Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn.
Direct preference optimization: Your language model is secretly a reward model.
arXiv preprint arXiv:2305.18290, 2023.



Chaoqi Wang, Yibo Jiang, Chenghao Yang, Han Liu, and Yuxin Chen.
Beyond reverse kl: Generalizing direct preference optimization with diverse divergence constraints.
arXiv preprint arXiv:2309.16240, 2023.



Amos Tversky and Daniel Kahneman.
Advances in prospect theory: Cumulative representation of uncertainty.
Journal of Risk and uncertainty, 5:297–323, 1992.



Saddique Ansari.
Prospect theory.
<https://www.economicsonline.co.uk/definitions/prospect-theory.html>, 2024.
Economics Online, accessed July 10, 2025.



Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela.
Kto: Model alignment as prospect theoretic optimization.
arXiv preprint arXiv:2402.01306, 2024.



Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello.
A general theoretical paradigm to understand learning from human preferences.
In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 2024.

References VI



Rafael Rafailov, Yaswanth Chittepudi, Ryan Park, Harshit Sikchi, Joey Hejna, Bradley Knox, Chelsea Finn, and Scott Niekum.
Scaling laws for reward model overoptimization in direct alignment algorithms.
arXiv preprint arXiv:2406.02900, 2024.



Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn.
From r to q^* : Your language model is secretly a q-function, 2024.



Duanyu Feng, Bowen Qin, Chen Huang, Zheng Zhang, and Wenqiang Lei.
Towards analyzing and understanding the limitations of dpo: A theoretical perspective.
arXiv preprint arXiv:2404.04626, 2024.



Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston.
Iterative reasoning preference optimization.
arXiv preprint arXiv:2404.19733, 2024.



Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddhartha Naidu, and Colin White.
Smaug: Fixing failure modes of preference optimisation with dpo-positive.
arXiv preprint arXiv:2402.13228, 2024.



Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al.
Building math agents with multi-turn iterative preference learning.
arXiv preprint arXiv:2409.02392, 2024.

References VII



Alexey Gorbатовski, Boris Shaposhnikov, Alexey Malakhov, Nikita Surnachev, Yaroslav Aksenov, Ian Maksimov, Nikita Balagansky, and Daniil Gavrilov.
Learn your reference model for real good alignment.
arXiv preprint arXiv:2404.09656, 2024.



Angelica Chen, Sadhika Malladi, Lily H Zhang, Xinyi Chen, Qiuyi Zhang, Rajesh Ranganath, and Kyunghyun Cho.
Preference learning algorithms do not learn preference rankings.
arXiv preprint arXiv:2405.19534, 2024.



Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang.
Rlhf workflow: From reward modeling to online rlhf.
arXiv preprint arXiv:2405.07863, 2024.



Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al.
The llama 3 herd of models.
arXiv preprint arXiv:2407.21783, 2024.



Yunhao Tang, Daniel Zhaoan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, et al.
Understanding the performance gap between online and offline alignment algorithms.
arXiv preprint arXiv:2405.08448, 2024.



Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar.
Preference fine-tuning of llms should leverage suboptimal, on-policy data.
arXiv preprint arXiv:2404.14367, 2024.

References VIII



Ruizhe Shi, Minhak Song, Runlong Zhou, Zihan Zhang, Maryam Fazel, and Simon S Du.
Understanding the performance gap in preference learning: A dichotomy of rlhf and dpo.
arXiv preprint arXiv:2505.19770, 2025.



Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving.
Fine-tuning language models from human preferences.
arXiv preprint arXiv:1909.08593, 2019.



Amartya Sen.
Social choice theory.
In *Chapter 22 Social choice theory*, volume 3 of *Handbook of Mathematical Economics*, pages 1073–1181. Elsevier, 1986.



Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal.
A minimaximalist approach to reinforcement learning from human feedback.
arXiv preprint arXiv:2401.04056, 2024.



Paul B Simpson.
On defining areas of voter choice: Professor tullock on stable voting.
The Quarterly Journal of Economics, 83(3):478–490, 1969.



Amir Beck and Marc Teboulle.
Mirror descent and nonlinear projected subgradient methods for convex optimization.
Operations Research Letters, 31(3):167–175, 2003.

References IX



Amir Beck.

First-order methods in optimization.
SIAM, 2017.



Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn.

Direct preference optimization: Your language model is secretly a reward model.
Advances in Neural Information Processing Systems, 36, 2024.



Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacrose, Ahmed Awadallah, and Tengyang Xie.

Direct nash optimization: Teaching language models to self-improve with general preferences.
arXiv preprint arXiv:2404.03715, 2024.



Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu.

Self-play preference optimization for language model alignment.
arXiv preprint arXiv:2405.00675, 2024.



Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, et al.

From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization.
In *International Conference on Machine Learning*, pages 8525–8535. PMLR, 2021.



Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras.

Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile.
arXiv preprint arXiv:1807.02629, 2018.

References X



Samuel Sokota, Ryan D'Orazio, J Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, Noam Brown, and Christian Kroer.
A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games.
arXiv preprint arXiv:2206.05825, 2022.



Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, et al.
Nash learning from human feedback.
arXiv preprint arXiv:2312.00886, 18, 2023.



Yuheng Zhang, Dian Yu, Baolin Peng, Linfeng Song, Ye Tian, Mingyue Huo, Nan Jiang, Haitao Mi, and Dong Yu.
Iterative nash policy optimization: Aligning llms with general preferences via no-regret learning.
arXiv preprint arXiv:2407.00617, 2024.



Daniele Calandriello, Daniel Guo, Remi Munos, Mark Rowland, Yunhao Tang, Bernardo Avila Pires, Pierre Harvey Richemond, Charline Le Lan, Michal Valko, Tianqi Liu, et al.
Human alignment of large language models through online preference optimisation.
arXiv preprint arXiv:2403.08635, 2024.



Mingzhi Wang, Chengdong Ma, Qizhi Chen, Linjian Meng, Yang Han, Jiancong Xiao, Zhaowei Zhang, Jing Huo, Weijie J Su, and Yaodong Yang.
Magnetic preference optimization: Achieving last-iterate convergence for language models alignment.
arXiv preprint arXiv:2410.16714, 2024.



David Silver and Richard S Sutton.
Welcome to the era of experience.
Google AI, 1, 2025.

References XI



Melody Y Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Helyar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, et al.
Deliberative alignment: Reasoning enables safer language models.
arXiv preprint arXiv:2412.16339, 2024.



Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar.
Scaling llm test-time compute optimally can be more effective than scaling model parameters.
arXiv preprint arXiv:2408.03314, 2024.



Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al.
Tulu 3: Pushing frontiers in open language model post-training.
arXiv preprint arXiv:2411.15124, 2024.



Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al.
Deepseekmath: Pushing the limits of mathematical reasoning in open language models.
arXiv preprint arXiv:2402.03300, 2024.



Youssef Mroueh.
Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification.
arXiv preprint arXiv:2503.06639, 2025.



Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al.
Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.
arXiv preprint arXiv:2501.12948, 2025.

References XII



Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo.

Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning.

arXiv preprint arXiv:2502.14768, 2025.



Jian Hu.

Reinforce++: A simple and efficient approach for aligning large language models.

arXiv preprint arXiv:2501.03262, 2025.



Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe.

Let's verify step by step.

arXiv preprint arXiv:2305.20050, 2023.



Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang.

rstar-math: Small llms can master math reasoning with self-evolved deep thinking.

arXiv preprint arXiv:2501.04519, 2025.



Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng.

Free process rewards without process labels.

arXiv preprint arXiv:2412.01981, 2024.



Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al.

Process reinforcement through implicit rewards.

arXiv preprint arXiv:2502.01456, 2025.

References XIII



Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al.

Ttrl: Test-time reinforcement learning.

arXiv preprint arXiv:2504.16084, 2025.



Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong.

Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models.

arXiv preprint arXiv:2505.24864, 2025.