Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 8: Alignment and Reasoning with Reinforcement Learning

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-568 (Spring 2026)

License Information for Reinforcement Learning (EE-568)

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

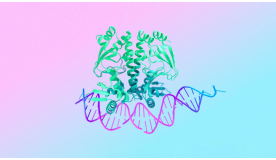
Why next-token prediction is not the final objective

- Pre-training learns to predict the next token well.
- But in deployment, we care about solving tasks, not only continuing text fluently.

Reasoning

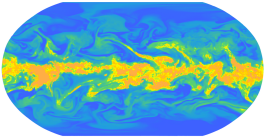
Long, checkable reasoning:
can the model sustain it?
Source: Gemini 3 Deep Think

Biology



Structure and interactions:
can the model predict them?
Source: AlphaFold 3

Climate

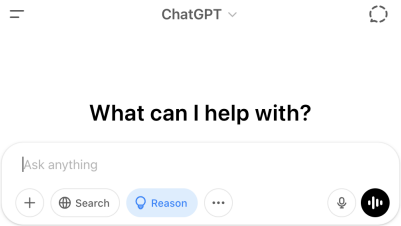


Useful scientific predictions:
can the model make them?
Source: WeatherNext 2

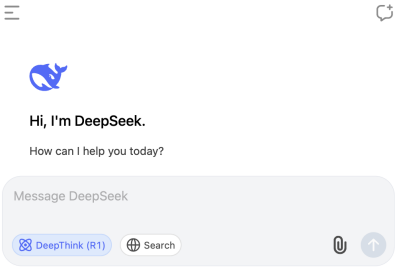
- This creates an objective gap: the final objectives require post-training beyond next-token prediction.

Reasoning chatbots

- More and more LLMs expose a dedicated *reasoning* mode.



https://chatgpt.com/



https://chat.deepseek.com/

- Reasoning mode is one visible outcome of post-training.
- Our question is: how do we turn a next-token predictor into a model that reasons better?

From Chain-of-Thought to reasoning

○ Chain-of-Thought (CoT) [26] tries to elicit reasoning by showing worked examples.

Multiple Q/A examples

CoT examples:

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: $\cdots$ **A:** $\cdots$

User Prompt: s_i

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Answer: a_i

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9.

○ Given a new question, we hope the model imitates the structure of these examples.

Remarks:

○ Here CoT is in-context prompting: no parameter update.

○ Training on CoT traces is learning from demonstrations (*cf.*, Lecture 7).

From Chain-of-Thought to reasoning

- With RL, we try to learn reasoning from rewards, not just elicit it by prompting.

User Prompt: s_i

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Learned reasoning by DeepSeek

Internal CoT:

<THINK> Okay, let's see. The problem says the cafeteria had 23 apples. Then they used 20 to make lunch, and after that, they bought 6 more. We need to find out how many apples they have now.
First, I'll start with the initial number of apples, which is 23. Then, they used 20 for lunch. Using apples means subtracting them from the total. So, I subtract 20 from 23. Let me do that calculation: 23 minus 20. That gives me 3. So after using 20 apples, they have 3 left.

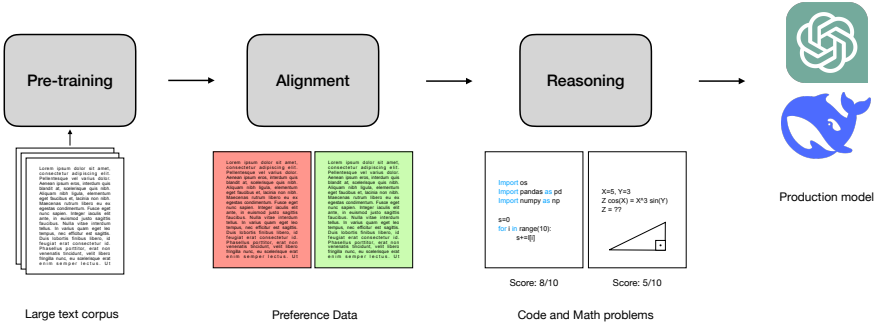
$\cdots$ </THINK>

Answer: a_i

A: The answer is 9.

- The internal CoT is not copied from examples and not directly prompted.
- Instead, it emerges because reasoning that leads to better answers receives higher reward.

This lecture in the LLM pipeline



- Pre-training builds the base next-token predictor.
- This lecture focuses on post-training: alignment and reasoning.

For a broader treatment of LLM training, including pre-training, see EE-628.

Pre-training as next-token prediction

- Let $w_{1:T} = (w_1, \ldots, w_T)$ denote a length-T sequence of tokens from a high-dimensional vocabulary $\mathcal{V}$.

Definition (Language model [11])

A language model assigns probabilities to token sequences as

$$\mathsf{p}(w_{1:T}) = \prod_{t=1}^{T} \mathsf{p}(w_t | w_{1:t-1}).$$

Remarks:

- Pre-training fits these conditional probabilities on a large text corpus.
- At each step, the model predicts the next token from the previous ones.

Example

If $S = w_{1:3} =$ "happy new year", then $\mathsf{p}(S) = \mathsf{p}(\text{happy})\mathsf{p}(\text{new}|\text{happy})\mathsf{p}(\text{year}|\text{happy new})$.

The pre-training objective

- Our goal is to minimize the negative log-likelihood of the training sequences.

Notation:

- Vocabulary $\mathcal{V} = \{v_1, \dots, v_d\}$; target index $y_t \in \{1, \dots, d\}$, so $w_t = v_{y_t}$.
- Model output $\mathbf{u}_t = \mathbf{h}_\theta(w_{1:t-1}) \in \Delta^d$, $\mathbf{u}_t(j) = \mathsf{p}_\theta(v_j | w_{1:t-1})$.
- One-hot label $\hat{\mathbf{u}}_t \in \Delta^d$ with $\hat{\mathbf{u}}_t(y_t) = 1$.

Derivation:

- Apply $-\log$ to the chain rule and read off cross entropy:

$$\boxed{\begin{aligned} -\log \mathsf{p}_\theta(w_{1:T}) &= -\log \prod_{t=1}^{T} \mathsf{p}_\theta(w_t | w_{1:t-1}) = \sum_{t=1}^{T} -\log \mathsf{p}_\theta(w_t | w_{1:t-1}) \\ &= \sum_{t=1}^{T} -\log \mathbf{u}_t(y_t) = -\sum_{t=1}^{T} \sum_{j=1}^{d} \hat{\mathbf{u}}_t(j) \log \mathbf{u}_t(j). \end{aligned}}$$

Remarks:

- Teacher forcing: during training, the model sees the correct history.
- Auto-regressive inference: at test time, the history comes from the model's own predictions.

Beyond next-token prediction

- We now move from pre-training to post-training.
- Post-training uses additional signals to shape model behavior beyond next-token prediction.
- From now on, we use RL notation: s denotes a prompt (state), and a denotes a response (action).
- In this lecture, we discuss three post-training feedback types: demonstrations, preferences, and rewards.
- We start with demonstrations, then move to preferences and verifiable rewards.

Remarks:

- Demonstrations: show me the answer.
- Preferences: tell me which answer is better.
- Rewards: tell me if the answer is correct.

Learning from demonstrations

- Demonstrations provide prompts $\{s_i\}_{i=1}^N$ together with desired answers $\{a_i\}_{i=1}^N$.
- We fit the policy by maximizing the likelihood of these answers.
- In LLM post-training, this is supervised fine-tuning (SFT).

Behavioral cloning from demonstrations

1: The learner receives: (i) A dataset $\mathcal{D}_{\text{demo}} = \{(s_i, a_i)\}_{i=1}^{N_{\text{demo}}}$, (ii) A policy function parameter $\theta \in \Theta$.

2: The learner computes the loss: $\ell_{\text{BC}}(\theta) := \sum_{i=1}^{N_{\text{demo}}} \left[-\log \pi_\theta\left(a_i | s_i\right)\right]$.

3: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta \in \Theta} \ell_{\text{BC}}(\theta)$.

Remarks:

- We covered this in Lecture 7 as behavioral cloning / imitation learning.
- In practice, SFT often produces the reference policy for later post-training.
- Training on CoT traces is a special case: the answer includes reasoning tokens.
- Llama 3, DeepSeek-R1, and Kimi k1.5 all include an SFT stage [8, 7, 25].

Learning from preferences: setup

- A preference dataset contains prompts $\{s_i\}_{i=1}^N$ and response pairs $\left\{(a_i^+, a_i^-)\right\}_{i=1}^N$.
- Here, a_i^+ is preferred to a_i^- for prompt s_i.

Example: user feedback

Prompt s_i: "Write a polite two-sentence email declining a meeting."
a_i^+ follows the requested tone and length, whereas a_i^- is overly long or ignores the instruction.

Remarks:

- In practice, such comparisons often come from annotators or user feedback.
- Preference comparisons are often cheaper than demonstrations.
- To use such comparisons for learning, we need a model of preferences.

A first preference model: Bradley-Terry

- We model preferences through a probability $\mathsf{p}_{\mathrm{pref}}(a, a'|s)$ that a is preferred to a' in response to s.
- Bradley-Terry [3] assumes that an unknown reward function r induces these probabilities.

Bradley-Terry model

$$\mathsf{p}_{\mathrm{pref}}(a, a'|s) = \sigma(r(s, a) - r(s, a')) = \frac{1}{1 + e^{r(s,a') - r(s,a)}}.$$

Here, σ is the sigmoid function.

Remarks:

- Bradley-Terry reduces pairwise preferences to reward differences.
- Only differences in r matter, not its absolute value.
- Later, we will enrich this model with ties to reduce over-optimization.
- Given such a preference model, we still need a method to train the policy.

RLHF: learning from preferences

- RLHF (reinforcement learning from human feedback) is the standard answer.
- Given a preference model, it trains the policy in two steps.

Step 1:

- Fit a reward model $\widehat{r}$ from preference data.
- Preferred answers should receive higher reward than rejected ones.

Step 2:

- Optimize a policy parameterization $\theta^\star$ against $\widehat{r}$.
- At the same time, keep the policy close to the pretrained model π_{ref}:
  - maximize $\widehat{r}$;
  - control the drift through $\beta D_{\mathrm{KL}}(\pi_{\theta^\star}(\cdot|s)\|\pi_{\mathrm{ref}}(\cdot|s))$.

Remark:

- Here, $D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi'(\cdot|s))$ is the Kullback-Leibler divergence.

RLHF as a bilevel problem

- Notation: ρ is the prompt distribution; RLHF optimizes over prompts $s \sim \rho$.
- The two-step RLHF pipeline leads to the following bilevel problem:

$$\theta^\star \in \arg\max_{\theta \in \Theta} \mathbb{E}_{s\sim\rho, a\sim\pi_\theta(\cdot|s)} \left[\widehat{r}(s,a)\right] - \beta \mathbb{E}_{s\sim\rho}[D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \,\|\, \pi_{\mathrm{ref}}(\cdot|s))]. \tag{1}$$

$$\text{s.t.} \quad \widehat{r} = \arg\max_{r \in \mathcal{R}} \sum_{i=1}^{N} \log \sigma\Big(r(s_i, a_i^+) - r(s_i, a_i^-)\Big). \tag{2}$$

Remarks:

- The upper level trains the policy, and the lower level fits the reward model.
- β controls how far we move from the reference model.
- Larger β gives more conservative alignment updates.

Implementing RLHF: reward fitting

- The first implementation step fits a reward model from preference comparisons.

Reward fitting

Given: a preference data set $\mathcal{D}_{\text{pref}} = \{(s_i, a_i^+, a_i^-)\}_{i=1}^{N_{\text{pref}}}$ and a reward function class $\mathcal{R}$.

$$\widehat{r} = \arg\max_{r \in \mathcal{R}} \sum_{i=1}^{N} \log \left(\sigma \left(r(s_i, a_i^+) - r(s_i, a_i^-) \right) \right). \qquad \text{(Reward Loss)}$$

Implementation: train the reward model with your favorite supervised optimizer.

Remarks:

- This is supervised learning on preference pairs.
- Preferred answers should receive higher reward than rejected ones.
- Often, $\mathcal{R}$ shares the LLM backbone and uses a scalar reward head.
- This step is offline and supervised, but still nontrivial at scale.
- This was the standard RLHF pipeline before direct methods such as DPO [20].

Implementing RLHF: policy optimization

- The second implementation step approximately solves the policy objective.

Approximate policy optimization

Given: a prompt data set $\mathcal{D}_{\text{prompts}} = \{s_i\}_{i=1}^{N_{\text{prompts}}}$, a learned reward $\widehat{r}$, and a policy class Θ.

$$J_\beta^r(\pi) := \mathbb{E}_{s\sim\rho,\, a\sim\pi(\cdot|s)}[r(s,a)] - \beta\mathbb{E}_{s\sim\rho}[D_{\text{KL}}(\pi(\cdot|s)\|\pi_{\text{ref}}(\cdot|s))]\,,$$
$$\theta^\star \in \arg\max_{\theta\in\Theta} \widehat{J_\beta^r}(\pi_\theta). \tag{RLHF}$$

Implementation: approximately solve this step with PPO or REINFORCE.

Remarks:

- π_{ref} denotes the reference policy, typically the supervised fine-tuned model.
- The superscript in J_β^r records which reward function is used.
- In the objective, $a \sim \pi_\theta(\cdot|s)$ means sampling from the current policy.
- This step samples new responses and runs costly RL updates.
- Together, these two stages make RLHF cumbersome.
- Next, we will see a direct alternative.

A direct alternative: DPO

- RLHF uses a two-stage pipeline: reward fitting, then policy optimization.
- Can we optimize preferences directly from the fixed preference data set?
- Direct Preference Optimization (DPO) [20] eliminates explicit reward fitting and online RL.

Direct Preference Optimization

1: The learner receives as input:
   - A preference dataset $\mathcal{D}_{\text{pref}} = \{(s^i, a_i^+, a_i^-)\}_{i=1}^N$.
   - A policy function parameters class Θ.

2: The learner computes the stochastic loss:

$$\ell_{\text{DPO}}(\theta) := -\frac{1}{|\mathcal{D}_{\text{pref}}|} \sum_{s,a^+,a^- \in \mathcal{D}_{\text{pref}}} \left[\log \left(\sigma \left(\beta \log \left(\frac{\pi_\theta(a^+|s)}{\pi_{\text{ref}}(a^+|s)} \right) - \beta \log \left(\frac{\pi_\theta(a^-|s)}{\pi_{\text{ref}}(a^-|s)} \right) \right) \right) \right].$$

3: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta \in \Theta} \ell_{\text{DPO}}(\theta)$.

Remark:
- The goal is to absorb reward fitting into a single preference loss.
- Next, we derive this objective from the RLHF optimum and the Bradley-Terry model.

DPO derivation: the RLHF optimum

- For fixed $r(\cdot,\cdot)$, the policy step in (RLHF) becomes [20]:

$$\pi^\star(\cdot|s) = \arg\max_{\pi(\cdot|s)} \ \mathbb{E}_{a\sim\pi(\cdot|s)}[r(s,a)] - \beta D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi_{\mathrm{ref}}(\cdot|s)).$$

- For each prompt s, the optimizer is a Gibbs reweighting of π_{ref}:

$$\pi^\star(a|s) = \frac{\pi_{\mathrm{ref}}(a|s)\exp(\beta^{-1}r(s,a))}{Z(s,\beta)}. \qquad \text{(Optimal Policy)}$$

Remarks:

- Here, $Z(s,\beta)$ is the partition function that normalizes the policy.
- Maximizing reward minus KL produces an exponential tilting of π_{ref}.

DPO derivation: the obstacle

- Recall the RLHF optimum:

$$\pi^\star(a|s) = \frac{\pi_{\mathrm{ref}}(a|s)\exp(\beta^{-1}r(s,a))}{Z(s,\beta)}.$$

- The partition function is given by

$$Z(s,\beta) := \sum_{a'} \pi_{\mathrm{ref}}(a'|s)\exp(\beta^{-1}r(s,a')). \tag{3}$$

- Therefore, the optimal policy can be written in log form as

$$\log(\pi^\star(a|s)) = \log(\pi_{\mathrm{ref}}(a|s)) + \beta^{-1}r(s,a) - \log(Z(s,\beta)). \tag{4}$$

Remarks:

- Directly using this form is difficult because $Z(s,\beta)$ is intractable.
- This suggests comparing two answers to the same prompt, so that Z may cancel out.

DPO derivation: pairwise cancellation

- Recall that

$$\log(\pi^\star(a|s)) = \log(\pi_{\mathrm{ref}}(a|s)) + \beta^{-1} r(s,a) - \log(Z(s,\beta)).$$

- For two answers a, a' to the same prompt, $Z(s,\beta)$ cancels:

$$\boxed{r(s,a) - r(s,a') = \beta \log\left(\frac{\pi^\star(a|s)}{\pi_{\mathrm{ref}}(a|s)}\right) - \beta \log\left(\frac{\pi^\star(a'|s)}{\pi_{\mathrm{ref}}(a'|s)}\right)} \qquad \text{(implicit reward identification)}$$

- Therefore, the Bradley-Terry model gives

$$\mathsf{p}_{\mathrm{pref}}(a^+, a^-|s) = \sigma\left(\beta \log \frac{\pi^\star(a^+|s)}{\pi_{\mathrm{ref}}(a^+|s)} - \beta \log \frac{\pi^\star(a^-|s)}{\pi_{\mathrm{ref}}(a^-|s)}\right).$$

Remarks:

- The partition function disappears in pairwise differences.
- This is the key observation behind DPO.

DPO derivation: a direct loss

- Rather than first forming the unknown optimum $\pi^\star$, DPO directly parameterizes the policy as π_θ.
- This defines the preference model

$$\mathsf{p}_\theta(a^+, a^-|s) = \sigma\left(\beta \log \frac{\pi_\theta(a^+|s)}{\pi_{\mathrm{ref}}(a^+|s)} - \beta \log \frac{\pi_\theta(a^-|s)}{\pi_{\mathrm{ref}}(a^-|s)}\right).$$

- We then fit this model by maximum likelihood on the preference data:

$$\ell_{\mathrm{DPO}}(\theta) := -\mathbb{E}_{(s,a^+,a^-)\sim\mathcal{D}_{\mathrm{pref}}}\left[\log\left(\sigma\left(\beta \log\left(\frac{\pi_\theta(a^+|s)}{\pi_{\mathrm{ref}}(a^+|s)}\right) - \beta \log\left(\frac{\pi_\theta(a^-|s)}{\pi_{\mathrm{ref}}(a^-|s)}\right)\right)\right)\right].$$

Remarks:

- The expectation is over tuples (s, a^+, a^-) in $\mathcal{D}_{\mathrm{pref}}$.
- The partition function has already canceled out.
- We minimize $\ell_{\mathrm{DPO}}(\theta)$ over the policy parameters θ.
- All desiderata are achieved: direct loss, fixed data, no policy sampling.

Why DPO is attractive

- DPO turns KL-regularized RLHF into a supervised preference loss.
- It uses a fixed preference dataset: no online policy sampling during training.
- It avoids fitting an explicit reward model.
- This makes DPO simple, stable, and widely used for instruction tuning.

Question: Does this direct loss still recover the RLHF solution from finite data?

- Next, we compare the ideal population picture with the finite-data setting.

When are DPO and RLHF equivalent?

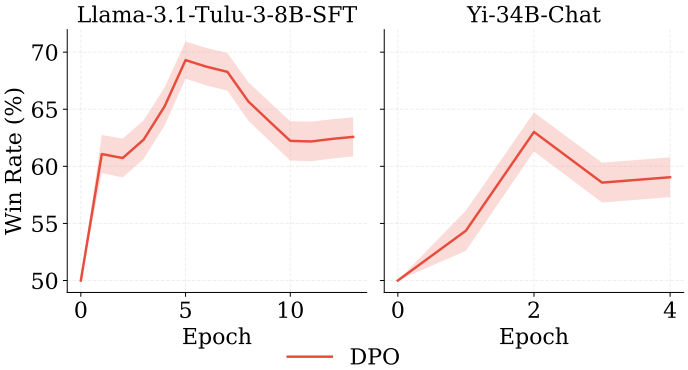
- The exact equivalence holds for unrestricted policies; parameterization introduces a gap.

Unrestricted policy class Π	**Parameterized policy class Π_Θ**
For every reward r, the RLHF optimum satisfies $\pi_r^\star(a\|s) \propto \pi_{\mathrm{ref}}(a\|s)\exp(r(s,a)/\beta)$. This map is a bijection between rewards and optimal policies [20]. $\Rightarrow$ DPO and RLHF are equivalent.	In practice we optimize over π_θ, not all policies. Then $\widehat{r}_\theta := \beta \log \frac{\pi_\theta}{\pi_{\mathrm{ref}}}$ is only constrained on observed preference pairs [1, 28]. $\Rightarrow$ DPO can admit extra solutions not matching RLHF.

Remarks:
- With unrestricted policies, reward differences identify the RLHF optimum.
- With finite preference data, DPO only constrains observed pairs.
- DPO may increase probability on responses that were never reliably compared [1, 28].
- This is the mechanism behind over-optimization.

DPO over-optimizes

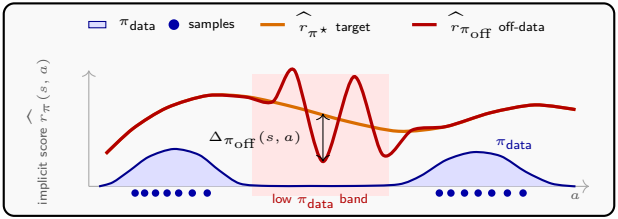
- Empirically, longer DPO training need not improve generation quality.
- Lower DPO training loss need not mean higher win rate.



Problem: ○ DPO is uncertainty-unaware: it can trust comparisons beyond data support.

Finite preference data only constrain covered actions

- Preference data have coverage set by π_{data}: $s_i \sim \rho,\ a_i, a_i' \sim \pi_{\text{data}}(\cdot|s_i)$; p_{pref} only labels those pairs.
- DPO scores a candidate policy by its log-ratio to the reference model: $\widehat{r}_\pi(s,a) := \beta \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)}$.
- The empirical loss only sees observed score gaps: $\ell_i(\pi) = -\log \sigma\Big(\widehat{r}_\pi(s_i, a_i^+) - \widehat{r}_\pi(s_i, a_i^-)\Big)$.



Remarks:

- π_{data} controls which actions enter the empirical DPO loss.
- Covered comparisons constrain score gaps near the samples.
- π_{off} denotes an off-data candidate that can remain wrong with little penalty.

Coverage controls score-gap transfer

- Compare a candidate policy to the target implicit score:

$$\Delta_\pi(s,a) := \widehat{r}_\pi(s,a) - \widehat{r}_{\pi^\star}(s,a).$$

- On the previous slide, $\Delta_{\pi_{\mathrm{off}}}$ is the red-orange gap.
- To control policy-side error, rewrite it under the data distribution:

$$|\mathbb{E}_{\rho,\pi}[\Delta_\pi]| = \left|\mathbb{E}_{\rho,\pi_{\mathsf{data}}}\left[\frac{\pi(a|s)}{\pi_{\mathsf{data}}(a|s)}\Delta_\pi(s,a)\right]\right| \le \underbrace{\left(\mathbb{E}_{\rho,\pi_{\mathsf{data}}}\left[\left(\frac{\pi(a|s)}{\pi_{\mathsf{data}}(a|s)}\right)^2\right]\right)^{1/2}}_{\sqrt{C^\pi}} \left(\mathbb{E}_{\rho,\pi_{\mathsf{data}}}[\Delta_\pi^2]\right)^{1/2}.$$

- The first factor defines concentrability of π against the data policy [2]:

$$C^\pi := \mathbb{E}_{s\sim\rho,\, a\sim\pi_{\mathsf{data}}}\left[\left(\frac{\pi(a|s)}{\pi_{\mathsf{data}}(a|s)}\right)^2\right].$$

Remarks:

- Small data-side score error transfers only when C^π is small.
- C^π is large when π puts mass where π_{data} is tiny.
- π_{off} can look accurate under π_{data}, yet have large $C^{\pi_{\mathrm{off}}}$.

DPO pays for worst-case coverage

- (Recall) Goal: find $\pi^\star$ that does well on the reward-minus-KL objective in (RLHF).
- Offline data can certify $\pi^\star$ only where π_{data} covers its actions.
- DPO selects $\pi_\theta \in \Pi$; its certificate must hold uniformly over Π [2]:

$$\sup_{\pi\in\Pi} \left|\mathbb{E}_{\rho,\pi}[\Delta_\pi]\right| \;\leq\; \sqrt{\max_{\pi\in\Pi} C^\pi}\; \sup_{\pi\in\Pi} \left(\mathbb{E}_{\rho,\pi_{\text{data}}}[\Delta_\pi^2]\right)^{1/2}.$$

Guarantee	Concentrability	Factor	Dominated by
Uniform DPO	all-policy	$\sqrt{\max_{\pi\in\Pi} C^\pi}$	worst $\pi \in \Pi$
Localized target	single-policy	$\sqrt{C^{\pi^\star}}$	comparator $\pi^\star$

Remarks:

- Ideally, certification should pay only for comparator $\pi^\star$: $C^{\pi^\star}$.
- DPO cannot exclude unsupported candidates such as π_{off} from Π.
- Next, we provide recent approaches to suppress off-data candidates such as π_{off}.

From J^r_β to the preference link

- Recall the RLHF policy objective in (RLHF):

$$J^r_\beta(\pi) := \mathbb{E}_{s\sim\rho,\, a\sim\pi(\cdot|s)}[r(s,a)] - \beta\mathbb{E}_{s\sim\rho}[D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi_{\mathsf{ref}}(\cdot|s))]\,.$$

- For a given prompt s, define the ratio $z_a := \pi(a|s)/\pi_{\mathsf{ref}}(a|s)$, then $D_{\mathrm{KL}}(\pi\|\pi_{\mathsf{ref}}) = \mathbb{E}_{a\sim\pi_{\mathsf{ref}}(\cdot|s)}[z_a \log z_a]$.
- More generally, replace $z\log z$ by an f-regularizer:

$$J^{r,f}_{\beta,s}(\pi) := \sum_a \pi(a|s)r(s,a) - \beta\sum_a \pi_{\mathsf{ref}}(a|s)f(z_a), \quad \text{s.t.} \quad \sum_a \pi(a|s) = 1.$$

- KKT optimality of $J^{r,f}_{\beta,s}$ w.r.t. $\pi(\cdot|s)$ (w/ dual variable λ_s) gives: $\quad r(s,a) - \beta f'(z_a) - \lambda_s = 0.$
- Pairwise comparisons remove the state-only dual variable λ_s:

$$r(s,a) - r(s,a') = \beta\Big(f'(z_a) - f'(z_{a'})\Big), \qquad \phi(z) = f'(z) \text{ up to constants.}$$

Remarks:
- The link is the map from π/π_{ref} ratios to implicit reward scores.
- The preference logit then uses $\beta(\phi(z_a) - \phi(z_{a'}))$.

χ^2PO: changing the DPO link

- Apply the previous slide to KL: $f_{\mathrm{KL}}(z) = z \log z \Rightarrow f'_{\mathrm{KL}}(z) = 1 + \log z$.
- Pairwise constants cancel, so DPO/KL uses the log-ratio link:

$$\phi_{\mathrm{KL}}(z) = \log z.$$

- χ^2PO [10] keeps KL but adds a χ^2 penalty:

$$f_{\chi\mathrm{mix}}(z) = \frac{1}{2}(z-1)^2 + z \log z, \qquad \phi_{\chi\mathrm{mix}}(z) = f'_{\chi\mathrm{mix}}(z) = z + \log z.$$

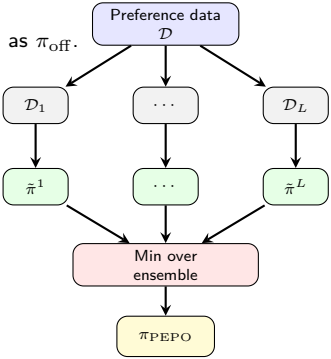
Remarks:

- Critical assumption: π_{data} is known; in the paper, $\pi_{\mathsf{data}} = \pi_{\mathsf{ref}}$.
- KL is logarithmic: score c allows $z = e^c$, so ratios can spike.
- χ^2PO adds z, making such spikes costly and damping π_{off}.

PEPO: pessimistic ensemble preference optimization [2]

- χ^2PO changes the link; PEPO instead changes the estimator.
- Use data-split disagreement to suppress off-data candidates such as π_{off}.
- Split preference data into L disjoint subsets.
- Train DPO-like policies $\tilde{\pi}^1, \ldots, \tilde{\pi}^L$.
- Favor responses on which all members agree.
- Aggregate by a worst-case score:

$$\boxed{\pi_{\text{PEPO}}(a|s) \propto \min_{\ell \in [L]} \tilde{\pi}^\ell(a|s)}$$

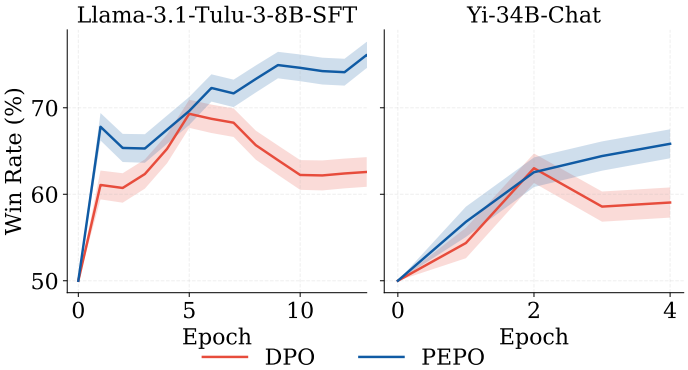


Remarks:

- Agreement across splits acts as a proxy for data support.
- If one split assigns low probability, the minimum vetoes the action.
- Thus π_{off} is suppressed when it relies on unstable off-data gains.

PEPO reduces over-optimization [2]

- PEPO makes preference optimization uncertainty-aware through ensemble pessimism.
- Empirically, pessimism keeps useful gains while damping late-stage overfit.



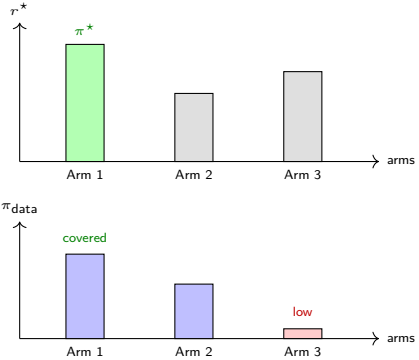
Remarks:

- DPO can keep increasing empirical preference fit after generalization degrades.
- PEPO is conservative where ensemble members disagree.

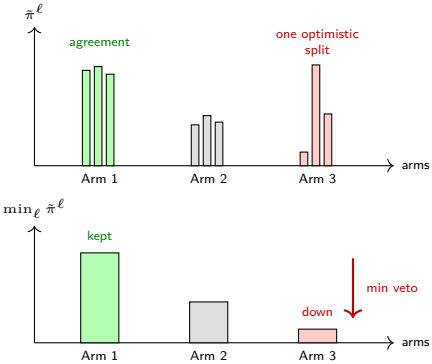
Three arms: pessimism vetoes unsupported candidates

- Stylized view of one prompt s: three candidate responses/actions compete.
- Arm 1 mirrors $\pi^\star$; Arm 3 mirrors π_{off} with weak π_{data} coverage.

Data and reward



What pessimism does



Remarks:

- Here $\pi^\star$ is covered, but π_{off} can make $\max_{\pi\in\Pi} C^\pi$ large.
- The minimum is not averaging: one pessimistic split can veto an unsupported arm.

PEPO guarantee: single-policy coverage

- In the tabular setting, PEPO gets a single-policy coverage guarantee.
- Let π_{PEPO} denote the ensemble-min output policy.
- We measure true reward-minus-KL performance: $J_\beta^{r^\star}$, not empirical preference loss.

Theorem (PEPO sample complexity [2])

For finite prompt and response sets $\mathcal{X}, \mathcal{A}$, data $\mathcal{D}_{\mathrm{pref}} = \{(s_i, a_i^+, a_i^-)\}_{i=1}^N$ with $s_i \in \mathcal{X}$ and $a_i^\pm \in \mathcal{A}$, reward bound $R_{\max}$, and logarithmic ensemble size L, with probability at least $1-\delta$, it holds that

$$J_\beta^{r^\star}(\pi^\star) - J_\beta^{r^\star}(\pi_{\mathrm{PEPO}}) \leq \widetilde{\mathcal{O}}\left(\sqrt{\frac{C^{\pi^\star} e^{R_{\max}} |\mathcal{X}||\mathcal{A}|^2}{N}}\right).$$

Remarks:

- The comparator $\pi^\star$ appears only through its coverage $C^{\pi^\star}$.
- Unsupported candidates like π_{off} do not enter via $\max_{\pi\in\Pi} C^\pi$.
- The rate is meaningful when N is large relative to $|\mathcal{X}||\mathcal{A}|^2$.

Proof sketch: why only $C^{\pi^\star}$ appears

- Pairwise reward gaps drive preferences:

$$\Delta_{r^\star}(s,a,a') := r^\star(s,a) - r^\star(s,a'), \qquad \widehat{\Delta} := \text{ensemble-min estimate.}$$

- Pessimism makes $\widehat{\Delta}$ a lower confidence estimate:

$$\mathbb{E}_{a'\sim\pi_{\mathsf{data}}(\cdot|s)}[\widehat{\Delta}(s,a,a')] \leq \mathbb{E}_{a'\sim\pi_{\mathsf{data}}(\cdot|s)}[\Delta_{r^\star}(s,a,a')].$$

- Let $\mathrm{Err}(s,a) := \mathbb{E}_{a'\sim\pi_{\mathsf{data}}(\cdot|s)}[\Delta_{r^\star}(s,a,a') - \widehat{\Delta}(s,a,a')]$:

$$J_\beta^{r^\star}(\pi^\star) - J_\beta^{r^\star}(\pi_{\mathrm{PEPO}}) \leq \sqrt{C^{\pi^\star}}\left(\mathbb{E}_{\rho,\pi_{\mathsf{data}}}[\mathrm{Err}(s,a)^2]\right)^{1/2}.$$

Remarks:

- Concentration controls the last factor from finite preference data.
- The π_{PEPO} term is absent because pessimism gives it the right sign.

PEPO takeaway: pessimism controls off-data extrapolation

- DPO is RLHF at the population level.
- With finite offline data, DPO can extrapolate off-data and over-optimize.
- PEPO uses a minimum over ensembles to make disagreement pessimistic.
- The key gain is the coverage constant: $C^{\pi^\star}$ instead of $\max_{\pi\in\Pi} C^{\pi}$.

Method	**Pessimism**	**Coverage in bound**	**Needs π_{data}?**
DPO	no	$\max_{\pi\in\Pi} C^{\pi}$	no
χ^2PO	yes	$C^{\pi^\star}$	yes
PEPO	ensemble min	$C^{\pi^\star}$	no

Limitation 2: Bradley-Terry assumes transitivity

- PEPO addresses coverage, but still inherits the Bradley-Terry model.
- Bradley-Terry represents preferences by one scalar reward $r(s, a)$.
- Therefore, it cannot represent preference cycles such as:

$$a_1 \succ a_2, \qquad a_2 \succ a_3, \qquad a_3 \succ a_1.$$

Human	Transitive ranking
h_1	$a_3 \succ a_1 \succ a_2$
h_2	$a_2 \succ a_3 \succ a_1$
h_3	$a_1 \succ a_2 \succ a_3$

- Majority vote gives $a_1 \succ a_2$, $a_2 \succ a_3$, and $a_3 \succ a_1$.
- Thus each human can be transitive while the aggregate preference is not.

Takeaway: ○ Non-transitive preferences require a game view, not a single reward.

NLHF as a policy game

- Bradley-Terry gave us a scalar reward; cyclic preferences need a game.
- First make the decision variable the policy itself: $\pi \in \Pi$, with Π living on the simplex.
- NLHF seeks a policy that no opponent policy can systematically beat [17]:

$$\pi^\star \in \arg\min_{\pi \in \Pi} \max_{\pi' \in \Pi} \ell_{\mathrm{NLHF}}(\pi, \pi').$$

- π proposes answers; π' searches for answers that beat them.

Remarks:
- A Nash policy is a mixed strategy over responses [18].
- Neural parameters come later; first define the policy-space game.

NLHF objective: three ingredients

- Recipe: learn a judge, sample policy matchups, and regularize the Nash loss.
- Ingredient 1 (data gives the judge): fit a Bernoulli model from comparisons:

$$\phi^\star \in \arg\min_{\phi\in\Phi} -\sum_{i=1}^{N} \log \mathsf{p}_{\mathrm{pref}}(a_i^+ \succ a_i^- | s_i; \phi).$$

- Ingredient 2 (matchup distribution): under $s \sim \rho$, compare $a \sim \pi(\cdot|s)$ with $a' \sim \pi'(\cdot|s)$.
- Ingredient 3 (loss defines the Nash game): take expectation over this matchup and penalize KL drift:

$$\ell_{\mathrm{NLHF}}(\pi, \pi') := -\mathbb{E}_{s,a,a'}[\mathsf{p}_{\mathrm{pref}}(a \succ a' | s; \phi^\star)] + \beta D_{\mathrm{KL},\rho}(\pi \| \pi_{\mathsf{ref}}) - \beta D_{\mathrm{KL},\rho}(\pi' \| \pi_{\mathsf{ref}}).$$

Remarks:

- The minimizer wants high preference against the opponent.
- The opponent tries to expose weaknesses of the current policy.

Policy space gives a strongly convex-concave game

- In policy space, NLHF is a saddle-point problem with linear coupling.
- The preference term from the previous loss expands as:

$$\begin{aligned}\mathcal{P}(\pi \succ \pi') &:= \mathbb{E}_{s,a,a'}[\mathsf{p}_{\mathrm{pref}}(a \succ a'|s;\phi^\star)] \\ &= \mathbb{E}_{s\sim\rho}\sum_{a,a'}\pi(a|s)\pi'(a'|s)\mathsf{p}_{\mathrm{pref}}(a \succ a'|s;\phi^\star).\end{aligned}$$

- The KL terms bring the "strong" geometry:

$$\ell_{\mathrm{NLHF}}(\pi,\pi') = -\underbrace{\mathcal{P}(\pi \succ \pi')}_{\text{bilinear coupling}} + \underbrace{\beta D_{\mathrm{KL},\rho}(\pi\|\pi_{\mathsf{ref}})}_{\text{strong convexity}} - \underbrace{\beta D_{\mathrm{KL},\rho}(\pi'\|\pi_{\mathsf{ref}})}_{\text{strong concavity}}.$$

Remarks:

- Since $\mathsf{p}_{\mathrm{pref}}$ is fixed, $\mathcal{P}$ is linear in each policy separately.
- Thus the policy-space game is strongly convex-strongly concave.
- This is the ideal theory model: policies live on probability simplexes.
- Nash-MD exploits exactly this KL geometry.

Nash-MD: simultaneous mirror descent-ascent

- Since ℓ_{NLHF} is strongly convex-concave, use descent-ascent for $\min_{\pi\in\Pi}\max_{\pi'\in\Pi}\ell_{\mathrm{NLHF}}(\pi,\pi')$.
- Policies live on simplexes, so use KL geometry.
- Suppress s; the opponent induces $g_{\pi'_t}(a) := \sum_{a'\in\mathcal{A}}\pi'_t(a')\mathsf{p}_{\mathrm{pref}}(a\succ a';\phi^\star)$.
- The minimizer gradient is then:

$$\nabla_\pi \ell_{\mathrm{NLHF}}(\pi_t,\pi'_t) = -g_{\pi'_t} + \beta\left(\log\frac{\pi_t}{\pi_{\mathsf{ref}}} + \mathbf{1}\right).$$

- The KL mirror descent step for the minimizer is:

$$\pi_{t+1} = \arg\min_{\pi\in\Pi}\left\{\left\langle \pi, \nabla_\pi \ell_{\mathrm{NLHF}}(\pi_t,\pi'_t)\right\rangle + \frac{1}{\eta}D_{\mathrm{KL}}(\pi\|\pi_t)\right\}.$$

- The opponent takes the corresponding KL mirror ascent step.

Remarks:
- This is simultaneous: both gradients are evaluated at (π_t,π'_t).
- Next we derive the KL-reference part of this update.

Nash-MD: deriving the reference pull

- First apply mirror descent to the KL term alone:

$$\bar{\pi}_t = \arg\min_{\pi \in \Pi} \left\{ \eta\beta \left\langle \pi, \log \frac{\pi_t}{\pi_{\text{ref}}} + \mathbf{1} \right\rangle + D_{\text{KL}}(\pi \| \pi_t) \right\}.$$

- The KKT condition, subject to $\sum_a \pi(a) = 1$ (dual λ'), yields the reference pull:

$$\log \frac{\bar{\pi}_t(a)}{\pi_t(a)} + \eta\beta \log \frac{\pi_t(a)}{\pi_{\text{ref}}(a)} + \lambda' = 0,$$

$$\bar{\pi}_t(a) = \frac{\pi_t(a)^{1-\beta\eta} \pi_{\text{ref}}(a)^{\beta\eta}}{\sum_{b \in \mathcal{A}} \pi_t(b)^{1-\beta\eta} \pi_{\text{ref}}(b)^{\beta\eta}}.$$

- The preference mirror step then uses the win-score against $\bar{\pi}_t$, not π_t:

$$\pi_{t+1} = \arg\min_{\pi \in \Pi} \left\{ -\langle \pi, g_{\bar{\pi}_t} \rangle + \frac{1}{\eta} D_{\text{KL}}(\pi \| \bar{\pi}_t) \right\}.$$

Remarks:

- The exponent $\beta\eta$ controls the pull toward π_{ref}.
- MTPO extends this MD view to multi-turn feedback via preference Q-functions [22].

Nash-MD vs mirror GDA: the red difference

- Both updates first apply the same KL reference pull:

$$\bar{\pi}_t(a) = \frac{\pi_t(a)^{1-\beta\eta}\pi_{\mathsf{ref}}(a)^{\beta\eta}}{\sum_b \pi_t(b)^{1-\beta\eta}\pi_{\mathsf{ref}}(b)^{\beta\eta}}.$$

Nash-MD [17]

$$g_{\bar{\pi}_t}(a) = \sum_b \bar{\pi}_t(b)\mathsf{p}_{\mathrm{pref}}(a \succ b)$$

$$\pi_{t+1} = \arg\min_{\pi\in\Pi}\left\{-\langle \pi, g_{\bar{\pi}_t}\rangle + \frac{1}{\eta}D_{\mathrm{KL}}(\pi\|\bar{\pi}_t)\right\}.$$

Mirror GDA

$$g_{\pi_t}(a) = \sum_b \pi_t(b)\mathsf{p}_{\mathrm{pref}}(a \succ b)$$

$$\pi_{t+1} = \arg\min_{\pi\in\Pi}\left\{-\langle \pi, g_{\pi_t}\rangle + \frac{1}{\eta}D_{\mathrm{KL}}(\pi\|\bar{\pi}_t)\right\}.$$

Remarks:

- Nash-MD forms the preference gradient after the reference pull.
- Mirror GDA keeps the old opponent in the preference gradient.

Optimistic mirror descent for Nash games

- Extragradient and optimistic methods stabilize games by anticipating rotation.
- For NLHF, $x = (\pi, \pi')$ and $F(x) := (\nabla_\pi \ell, -\nabla_{\pi'} \ell)$ is the joint game gradient.
- An optimistic KL step reuses the previous game gradient:

$$x_{t+1} = \arg\min_{x \in \Delta} \left\{ \langle x, 2F(x_t) - F(x_{t-1}) \rangle + \frac{1}{\eta} D_{\mathrm{KL}}(x \| x_t) \right\}.$$

- This is the principle behind OGDA and extragradient methods [12, 21].
- OMPO casts multi-step alignment as a Markov game and adds optimism [27].

Takeaway:
- Policy-space Nash tells us what to solve.
- Optimism/extragradient tells us how to stabilize game training.

Neural implementation reopens the game

- In practice, we parameterize policies by neural networks: $\pi = \pi_\theta$.
- The implemented game becomes:

$$\min_{\theta \in \Theta} \max_{\theta' \in \Theta} \ell_{\mathrm{NLHF}}(\pi_\theta, \pi_{\theta'}).$$

- The nonlinear map $\theta \mapsto \pi_\theta$ can destroy convexity-concavity.
- Gradient descent-ascent can then cycle or approach spurious sets [9, 15].

Remarks:
- Policy-space convexity does not transfer to θ-space.
- Two-time-scale/extragradient-type schemes can escape such limit cycles [19].

Before training: best-of-k at inference

- Some tasks have a programmatic checker $r_{\mathrm{ver}}(s, a)$ (math answer, code tests, format).
- Without training, sample k completions and keep the best [5]:

$$a_1, \ldots, a_k \sim \pi_\theta(\cdot|s), \qquad \widehat{a} \in \operatorname*{arg\,max}_{1 \le i \le k} r_{\mathrm{ver}}(s, a_i).$$

- For binary r_{ver}, $p_\theta(s) := \mathbb{P}_{a \sim \pi_\theta(\cdot|s)}[r_{\mathrm{ver}}(s, a) = 1]$ gives:

$$\text{pass@1} = p_\theta(s), \qquad \text{pass@}k = 1 - (1 - p_\theta(s))^k.$$

- Best-of-k spends inference compute; θ does not change.

Bridge:

- RLVR trains θ to raise $p_\theta(s)$, amortizing inference cost.
- Match the training objective to the intended inference budget [24].

RLVR: learning from executable rewards

- RLVR means reinforcement learning with verifiable rewards.
- Instead of learning a reward model from preferences, we are given an executable reward map

$$r_{\mathrm{ver}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}.$$

- This reward is computed by a checker that encodes success:

$$\begin{array}{rcl} \text{math} & : & \mathbb{1}\{\text{final answer is correct}\}, \\ \text{code} & : & \text{unit-test pass rate}, \\ \text{proofs} & : & \mathbb{1}\{\text{proof checker accepts}\}, \\ \text{format} & : & \mathbb{1}\{\text{constraint is satisfied}\}. \end{array}$$

- Once this reward exists, train the policy with KL-regularized RL:

$$\pi^\star \in \arg\max_{\pi \in \Pi} J_\beta^{r_{\mathrm{ver}}}(\pi),$$

$$J_\beta^{r_{\mathrm{ver}}}(\pi) := \mathbb{E}_{s\sim\rho,\, a\sim\pi(\cdot|s)}[r_{\mathrm{ver}}(s,a)] - \beta\, \mathbb{E}_{s\sim\rho}[D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi_{\mathsf{ref}}(\cdot|s))]\,.$$

Remarks:

- RLVR gives the optimization formulation; REINFORCE or PPO are optimizers.
- The key change from RLHF is the reward source, not the KL regularizer.
- Verifiable means checkable by a program, not guaranteed to match intent.

RLVR loop: sample, verify, update

- At iteration k, prompts are drawn from the task distribution:

$$s_j \sim \rho, \qquad j = 1, \ldots, B.$$

- The current policy, parameterized by θ_k, generates completions:

$$a_j \sim \pi_{\theta_k}(\cdot|s_j).$$

- The verifier converts each completion into a reward:

$$r_j := r_{\mathrm{ver}}(s_j, a_j), \qquad \mathcal{D}_k = \{(s_j, a_j, r_j)\}_{j=1}^{B}.$$

- An RL optimizer updates π_{θ_k} using $\mathcal{D}_k$ and the KL anchor.
- The next batch is generated by the updated policy $\pi_{\theta_{k+1}}$.

Remarks:

- Like PPO-style RLHF, this is an online policy-improvement loop.
- Unlike DPO or reward-model fitting, the rewards are not fixed preference labels.
- The reward source is executable: r_{ver} scores completions after sampling.

Why RLVR needs adapted policy gradients

- The action is a long completion: token probabilities are updated, but reward arrives at the end.
- Rewards are sparse and often binary; many completions can tie for the same prompt.
- A learned critic over prompts and completions is another large sequence model.
- On-policy sampling is expensive; every update changes the completion distribution.
- The KL anchor is structural: it prevents drift from pretrained language and instruction skills.

Remarks:

- We will first derive the gradient, then replace critics with prompt-level baselines.
- Avoiding a learned critic will save compute, memory, and engineering complexity.

RLVR gradient: score function and baselines

- Fix a prompt s and write $r_{\mathrm{ver}}(a) := r_{\mathrm{ver}}(s, a)$.
- The score-function identity gives the basic RLVR gradient:

$$\begin{aligned}\nabla_\theta \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[r_{\mathrm{ver}}(a)] &= \nabla_\theta \sum_a \pi_\theta(a|s) r_{\mathrm{ver}}(a) \\ &= \sum_a \pi_\theta(a|s) r_{\mathrm{ver}}(a) \nabla_\theta \log \pi_\theta(a|s) \\ &= \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[r_{\mathrm{ver}}(a) \nabla_\theta \log \pi_\theta(a|s)] \,.\end{aligned}$$

- Any prompt-dependent baseline $b(s)$ can be subtracted without bias:

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[b(s) \nabla_\theta \log \pi_\theta(a|s)] = b(s) \nabla_\theta \sum_a \pi_\theta(a|s) = 0.$$

Remarks:
- This is REINFORCE applied to completions rather than one-step actions.
- The estimator can use $r_{\mathrm{ver}}(s, a) - b(s)$: same gradient, lower variance.

Group relative policy optimization (GRPO)

- GRPO is the DeepSeekMath critic-free variant of PPO for LLM reinforcement learning [23].
- For a prompt s, sample a group from the old policy and verify:

$$a_i \sim \pi_{\theta_{\text{old}}}(\cdot|s), \qquad r_i := r_{\text{ver}}(s, a_i), \qquad i = 1, \ldots, G.$$

- Estimate a same-prompt baseline and scale:

$$\bar{r}_s = \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \widehat{\sigma}_s^2 = \frac{1}{G}\sum_{i=1}^{G}(r_i - \bar{r}_s)^2.$$

- Define the normalized, group-relative advantage:

$$\widehat{A}_i = \frac{r_i - \bar{r}_s}{\widehat{\sigma}_s + \varepsilon_{\text{std}}}, \qquad \varepsilon_{\text{std}} > 0.$$

Remarks:

- The group mean replaces a learned value baseline for the same prompt.
- The group scale normalizes rewards; its objective effect appears later.

GRPO objective

- Recall PPO: clip the likelihood ratio with radius $\epsilon_{\text{clip}} > 0$.
- For completion $a_i = (a_{i,1}, \ldots, a_{i,T_i})$, define the token ratio

$$\omega_{i,t}(\theta) := \frac{\pi_\theta(a_{i,t}|s, a_{i,<t})}{\pi_{\theta_{\text{old}}}(a_{i,t}|s, a_{i,<t})}.$$

- GRPO uses PPO clipping with the group advantage from the previous slide:

$$\max_\theta \quad \mathbb{E}_{s,\, a_{1:G} \sim \pi_{\theta_{\text{old}}}} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{t=1}^{T_i} \min\Big\{ \omega_{i,t}(\theta) \widehat{A}_i,\, \text{clip}(\omega_{i,t}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}) \widehat{A}_i \Big\} - \beta\, \widehat{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Bigg].$$

Meaning: - Increase tokens from above-average completions; suppress below-average ones.

GRPO KL estimator

- The KL term keeps the updated policy close to the reference model token by token.
- For prefix $(s, a_{i,<t})$, define the reference-to-policy ratio

$$u_{i,t}(\theta) := \frac{\pi_{\mathsf{ref}}(a_{i,t}|s, a_{i,<t})}{\pi_\theta(a_{i,t}|s, a_{i,<t})}.$$

- If $a_{i,t} \sim \pi_\theta(\cdot|s, a_{i,<t})$, then

$$\mathbb{E}[u_{i,t}(\theta) - \log u_{i,t}(\theta) - 1] = D_{\mathrm{KL}}(\pi_\theta(\cdot|s, a_{i,<t}) \| \pi_{\mathsf{ref}}(\cdot|s, a_{i,<t}))\,.$$

- GRPO averages this sampled-token estimate:

$$\widehat{D}_{\mathrm{KL}} = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}[u_{i,t}(\theta) - \log u_{i,t}(\theta) - 1]\,.$$

Remarks:
- The identity explains the estimator; optimization still uses sampled completions.
- This anchor is to π_{ref}, not to the old policy.

DeepSeek-R1 in this notation

- DeepSeek-R1-Zero applies GRPO directly to a base model on reasoning prompts [7].
- For each prompt, it samples a group of completions and scores them by rule rewards:

$$r_i = r_{\mathrm{acc}}(s, a_i) + \lambda r_{\mathrm{fmt}}(s, a_i).$$

- r_{acc} checks final answers or code tests; r_{fmt} checks required output tags.
- DeepSeek-R1 adds cold-start SFT before RL, then additional alignment stages.

Remarks:

- R1-Zero is the clean RLVR example; R1 is the production pipeline.
- Rule rewards remove reward-model extrapolation, not specification errors.

What GRPO standardization changes

- In a binary-verifier setting, define the success probability

$$p_\theta(s) := \mathbb{P}_{a\sim\pi_\theta(\cdot|s)}[r_{\mathrm{ver}}(s,a)=1]\,.$$

- REINFORCE ascends $p_\theta(s)$; subtracting a prompt baseline leaves this objective unchanged.
- Dividing by the group scale rescales each prompt gradient:

$$\nabla_\theta f_{\mathrm{GRPO}}(p_\theta(s)) = \frac{1}{\pi\sqrt{p_\theta(s)(1-p_\theta(s))}}\nabla_\theta p_\theta(s).$$

- Therefore, in the Davis and Recht idealization [6],

$$\max_\theta \mathbb{E}_{s\sim\rho}[f_{\mathrm{GRPO}}(p_\theta(s))]\,, \qquad f_{\mathrm{GRPO}}(p) = \frac{2}{\pi}\arcsin\sqrt{p}.$$

Remarks:
- Mean subtraction is baseline-like; scale division changes prompt weighting.
- This is an idealized objective view, not the full neural training objective.

RLVR fixes reward learning, not reward design

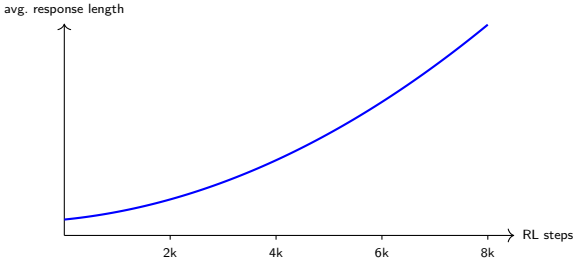
- Verifier hacking: pass public tests, exploit regexes, or guess finite answers.
- Sparse groups: if all G rewards match, then $\widehat{A}_i \approx 0$ and GRPO has little signal.
- Collapse and length incentives: diversity can decrease and CoT can become unnecessarily long.

Remarks:

- Hidden tests, held-out verifiers, KL, entropy, and diversity should be monitored separately.
- Pass@k from the opening slide of this section reappears as a diversity check.
- DAPO responds to these issues with sampling, token losses, and length control [29].

R1: reasoning behavior emerges during RL [7]

- In R1-Zero, the average response length grows during RL training.



Schematic of [7], Fig. 3.

Aha-moment sample [7]:

> "Wait, wait. Wait. That's an aha moment I can flag here. Let's reevaluate this step-by-step. . . "

Remarks:

- Reflective patterns can emerge from outcome rewards, without supervised CoT traces.
- This is evidence from a training trajectory, not a proof that RL is the only mechanism.

Post-training pipelines combine signals

○ Frontier post-training stacks SFT, preference methods, and verifiable rewards.

Model	Post-training stages	Reference
Llama 3	SFT $\rightarrow$ iterative DPO $\rightarrow$ PPO	[8]
Tulu 3	SFT $\rightarrow$ DPO $\rightarrow$ RLVR	[13]
DeepSeek-R1	cold-start SFT $\rightarrow$ RLVR (GRPO) $\rightarrow$ alignment	[7]
Kimi K1.5	SFT $\rightarrow$ RLVR with ℓ_2 regularization	[25]

Remark: ○ These stages are complementary: preferences shape behavior; verifiers train checkable correctness.

Reasoning is not only RL

- Reasoning-style traces can also be induced at inference time [16].

How many r in raspberry? **Question**

Let's break down the process of counting the letter 'r' in the word "raspberry" ...
* First letter: 'r' - This is an 'r', count = 1.
* Second letter: 'a' - Not an 'r', count remains 1 ...
* Sixth letter: 'e' - Not an 'r', count remains 1.
* Seventh letter: 'r' - This is an 'r', count = 2.
* Eighth letter: 'y' - Not an 'r', count remains 2 ...
The number of 'r's in "raspberry" is 2.
Wait, let's re-read the question carefully. It asks "How many r in raspberry?" ... * r - a - s - p - b - e - r - r - y ... * First 'r' ... * Second 'r' ... * Third 'r' ... Count = 3 ... **Reasoning trace**

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is $\boxed{3}$ **Response**

Figure: Source: `https://arxiv.org/pdf/2501.19393`

- For example, forcing the token "Wait" can extend the model's generation.
- This improves some reasoning benchmarks without changing the model parameters.

Remark: ○ Observing long CoT is not enough to identify the training mechanism that produced it.

Summary of RL for language models

- RLHF aligns models with human preferences through reward modeling and policy optimization.
- DPO is a direct offline alternative, but it can over-optimize under poor data coverage.
- PEPO uses pessimism to make preference optimization more reliable.
- NLHF treats non-transitive preferences as a policy game and uses Nash-style updates.
- RLVR replaces learned rewards with verifiers when correctness is checkable.
- Reasoning gains often combine verifiable rewards with scalable RL optimizers such as GRPO.

Thank you!

- Let's start with the project!

References I

[1] 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.
24

[2] Adam Barla, Emanuele Nevali, Luca Viano, and Volkan Cevher.
Provably avoiding over-optimization in direct preference optimization without knowing the data distribution.
arXiv preprint arXiv:2602.06239, 2026.
27, 28, 31, 32, 34

[3] 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.
13

[4] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu.
Self-play fine-tuning converts weak language models to strong language models.
arXiv preprint arXiv:2401.01335, 2024.
73

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman.
Training verifiers to solve math word problems, 2021.
46

References II

[6] Damek Davis and Benjamin Recht.
What is the objective of reasoning with reinforcement learning?
arXiv preprint arXiv:2510.13651, 2025.
55

[7] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, and Ziyi Gao.
Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
11, 54, 57, 58

[8] 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, 2024.
11, 58

[9] Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher.
The limits of min-max optimization algorithms: Convergence to spurious non-critical sets.
arXiv preprint arXiv:2006.09065, 2020.
45

[10] Audrey Huang, Wenhao Zhan, Tengyang Xie, Jason D Lee, Wen Sun, Akshay Krishnamurthy, and Dylan J Foster.
Correcting the mythos of kl-regularization: Direct alignment without overoptimization via chi-squared preference optimization.
arXiv preprint arXiv:2407.13399, 2024.
30

References III

[11] Dan Jurafsky and James H. Martin.
Speech and Language Processing (3rd ed. draft).
draft, third edition, 2023.
8, 70

[12] Galina M Korpelevich.
The extragradient method for finding saddle points and other problems.
Matecon, 12:747–756, 1976.
44

[13] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi.
Tulu 3: Pushing frontiers in open language model post-training, 2024.
58

[14] Andrey Andreyevich Markov.
Essai d'une recherche statistique sur le texte du roman.
Eugene Onegin" illustrant la liaison des epreuve en chain ('Example of a statistical investigation of the text of "Eugene Onegin" illustrating the dependence between samples in chain')". In: Izvistia Imperatorskoi Akademii Nauk (Bulletin de l'Académie Impériale des Sciences de St.-Pétersbourg). 6th ser, 7:153–162, 1913.
70

References IV

[15] Panayotis Mertikopoulos, Ya-Ping Hsieh, and Volkan Cevher.
A unified stochastic approximation framework for learning in games.
Mathematical Programming, 203:559–609, 2024.
45

[16] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto.
s1: Simple test-time scaling, 2025.
59

[17] 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.
38, 43

[18] John F Nash Jr.
Equilibrium points in n-person games.
Proceedings of the national academy of sciences, 36(1):48–49, 1950.
38

[19] Thomas Michaelsen Pethick, Puya Latafat, Panagiotis Patrinos, Olivier Fercoq, and Volkan Cevher.
Escaping limit cycles: Global convergence for constrained nonconvex-nonconcave minimax problems.
In *10th International Conference on Learning Representations (ICLR 2022)*, 2022.
45

References V

[20] 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, 36:53728–53741, 2023.
16, 18, 19, 24

[21] Alexander Rakhlin and Karthik Sridharan.
Optimization, learning, and games with predictable sequences.
arXiv preprint arXiv:1311.1869, 2013.
44

[22] Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al.
Multi-turn reinforcement learning from preference human feedback.
arXiv preprint arXiv:2405.14655, 2024.
42

[23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo.
Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.
51

[24] Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette.
Maximum likelihood reinforcement learning.
arXiv preprint arXiv:2602.02710, 2026.
46

References VI

[25] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang.

Kimi k1.5: Scaling reinforcement learning with llms, 2025.

11, 58

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al.

Chain-of-thought prompting elicits reasoning in large language models.

Advances in neural information processing systems, 35:24824–24837, 2022.

5

[27] Yongtao Wu, Luca Viano, Kimon Antonakopoulos, Yihang Chen, Zhenyu Zhu, Quanquan Gu, and Volkan Cevher.

Multi-step alignment as markov games: An optimistic online mirror descent approach with convergence guarantees.

Transactions on Machine Learning Research, 2025.

44, 75

[28] Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu.

Is dpo superior to ppo for llm alignment? a comprehensive study.

arXiv preprint arXiv:2404.10719, 2024.

24

References VII

[29] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang.

Dapo: An open-source llm reinforcement learning system at scale, 2025.

56

Supplementary material

N-gram LMs

Markov assumption [14]

The probability of a word only depends on the last $N-1$ words as

$$\mathsf{p}(w_t|w_{1:t-1}) = \mathsf{p}(w_t|w_{t-N:t-1}) \approx \frac{\#(w_{t-N:t})}{\#(w_{t-N:t-1})}.$$

Figure: Markov in 1913 [14] used "Markov chains" to predict whether the upcoming letter would be a vowel or a consonant.

Example

In the bigram LM ($N=2$), we only need to estimate $\mathsf{p}(w_t|w_{t-1}) \approx \frac{\#(w_{t-1:t})}{\#(w_{t-1})}$ to generate text.

w_t

w_{t-1}	i	want	to	eat
i	5	827	0	9
want	2	0	608	1
to	2	0	4	686
eat	0	0	2	0

w_{t-1}

i	want	to	eat
2533	927	2417	746

w_t

w_{t-1}	i	want	to	eat
i	0.002	0.33	0	0.0036
want	0.0022	0	0.66	0.0011
to	0.00083	0	0.0017	0.28
eat	0	0	0.0027	0

Figure: Count (Left) and probability $\mathsf{p}(w_t|w_{1:t-1})$ (Right) from the Berkeley Restaurant Project corpus of 9332 sentences [11].

Towards pre-training an N-gram LM

- In natural language processing (NLP), we use tokens to represent words coming from a vocabulary $\mathcal{V}$.

Terminologies:

- A *token* is the smallest unit that can be assigned a meaning to be processed.
  - In English, a token often corresponds to a word.
  - However, a single token can also encode compound words like *New York*.
  - In Chinese or Japanese, there is no space between words.
  - In these languages, sentence segmentation is required before we tokenize.
- We indicate the beginning and the end of sentences with tokens $\langle \mathrm{BOS} \rangle$ and $\langle \mathrm{EOS} \rangle$.
  - S_{source} "$\langle \mathrm{BOS} \rangle$ Happy new year $\langle \mathrm{EOS} \rangle$" has $T = 5$ tokens.
- The size of our vocabulary is denoted as $|\mathcal{V}|$.
- *Pre-training*: building a LM based on a large corpus in a (often) self-supervised manner.
- *Inference*: Using a trained LM to do next word prediction.

N-gram LMs: "Pre-training" & Inference

- The following simplified examples show the difficulty of pre-training and inference with 2-gram LMs.

"Pre-training"
1. Count $\#(w_{t-1})$ and $\#(w_{t-1:t})$ over the corpus.
2. Obtain probability $\mathsf{p}(w_t \mid w_{t-1})$ over the corpus.

Inference
1. Set w_1 as $\langle \mathrm{BOS} \rangle$, $t = 1$.
2. **While True**:
▸ $w_{t+1} = \arg\max_{w \in \mathcal{V}} \mathsf{p}(w \mid w_t)$
▸ **If** w_{t+1} is $\langle \mathrm{EOS} \rangle$: **break**
▸ $t = t + 1$
3. Output: $[w_1, \cdots, w_{t+1}]$.

Remarks:

- Need to store the probability for all N-gram pairs.
- Language is creative, some new N-gram pairs might not even appear on the corpus.
- Cannot incorporate earlier words than N due to the Markov assumption.

$$\mathsf{p}(\text{two} \mid \text{one plus one equals}) = \mathsf{p}(\text{two} \mid \text{it is wrong that one plus one equals})?$$

Alignment from demonstrations + online access

- A common method for this setting is SPIN (Self-Play fIne-tuNing) [4], which considers a two-player game:
  - One player tries to generate answers as similar as possible to the observed ones in $\mathcal{D}_{\text{demo}}$.
  - The other player aims to distinguish answers of the first player from the ones in $\mathcal{D}_{\text{demo}}$.
- At iteration t, both "players" solve the following bilevel formulation:

$$\theta_{t+1} = \arg\max_{\theta \in \Theta} \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s;\theta_t)} \left[r_{t+1}(s,a) - \eta^{-1} D_{\text{KL}}(\pi(\cdot|s;\theta)||\pi(\cdot|s;\theta_t)) \right], \tag{5}$$

$$\text{s.t.} \quad r_{t+1} = \arg\min_{r \in \mathcal{R}} \mathbb{E}_{s,a\sim\mathcal{D}_{\text{demo}}} \mathbb{E}_{a'\sim\pi(\cdot|s;\theta_t)} \left[\ell(r(s,a) - r(s,a')) \right], \tag{6}$$

  - where $\ell(\cdot)$ is a monotonically decreasing, non negative, smooth and convex function,
  - ρ is the prompt distribution,
  - η is the step size.

- The choice $\ell(x) = -x$ corresponds to minimizing an integral probability metric.
- A common choice in practice is the logistic loss function $\ell(s) = \log\left(1 + \exp(-s)\right)$.

Alignment from demonstrations + online access (continued)

- The upper level has an explicit solution $\pi(a|s;\theta_{t+1}) \propto \pi(a|s;\theta_t)\exp\left(\eta r_{t+1}(s,a)\right)$.
  - Question: What if $\theta_{t+1} \notin \Theta$?
- Assuming $\theta_{t+1} \in \Theta$, both problems reduce to a single optimization problem:

$$\theta_{t+1} = \arg\max_{\theta\in\Theta} \mathbb{E}_{s,a\sim\mathcal{D}_{\text{demo}}} \mathbb{E}_{a'\sim\pi(\cdot|s;\theta_t)} \left[-\ell\left(\eta^{-1}\log\left(\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_t)}\right) - \eta^{-1}\log\left(\frac{\pi(a'|s;\theta)}{\pi(a'|s;\theta_t)}\right)\right)\right]. \tag{7}$$

- **Self-study:** Compare it with DPO, introduced a few slides later.

OMPO: Using OGDA in Nash learning from Human Feedback

- The previous slides highlight results in the convex-concave setting.
- However, in the multi turn setting ℓ_{NLHF} is clearly non-convex and non-concave.
- [27] bypasses the problem resorting to LP techniques.
- Indeed, [27] recast the problem as a bilinear program over the space of occupancy measures.

$$(d^\star, d^\star) = \arg\max_{d \in \tilde{\mathcal{F}}} \min_{d' \in \tilde{\mathcal{F}}} \mathbb{E}_{s_1 \sim \rho} \sum_{h=1}^{H} \sum_{s,a,s',a'} d_h(s,a|s_1) r(s,a,s',a') d'_h(s',a'|s_1)\,,$$

- $\tilde{\mathcal{F}}$ is the product set of the Bellman flow constraints for a particular initial state, i.e.

$$\tilde{\mathcal{F}} = \times_{s_1 \in \mathrm{supp}(\rho)} \mathcal{F}_{s_1}\,.$$

- The Bellman flow constraints for a specific initial state are

$$\mathcal{F}_{s_1} = \left\{ d = (d_1, \ldots, d_H) : \sum\nolimits_a d_{h+1}(s,a) = \sum\nolimits_{s',a'} f(s|s',a') d_h(s',a'), d_1(s) = \mathbb{1}\{s = s_1\} \right\}.$$

- OMPO applies optimistic gradient descent ascent over the occupancy measure space.