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 2025)

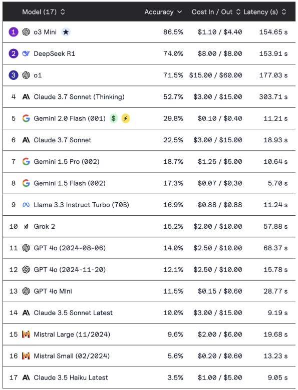
lions@epfl

License Information for Reinforcement Learning (EE-568)

A paradigm shift in Large Language Models (LLMs)

- Next token prediction has accomplished impressive performance in real-world tasks.
- However, there are tasks that requires “planning” beyond next token prediction, such as reasoning.
- An example is mathematics, where next token predictors do not achieve good results.

AIME Benchmark

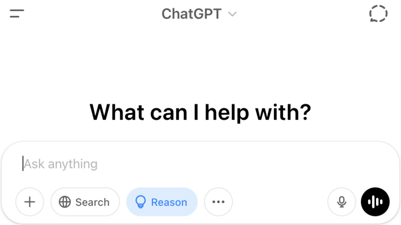


Model (17)	Accuracy	Cost In / Out	Latency (s)
1 o3 Mini ★	86.5%	$1.10 / $4.40	154.65 s
2 DeepSeek R1	74.0%	$8.00 / $8.00	153.91 s
3 o1	71.5%	$15.00 / $60.00	177.03 s
4 Claude 3.7 Sonnet (Thinking)	52.7%	$3.00 / $15.00	303.71 s
5 Gemini 2.0 Flash (001) $ ⚡	29.8%	$0.10 / $0.40	11.21 s
6 Claude 3.7 Sonnet	22.5%	$3.00 / $15.00	18.93 s
7 Gemini 1.5 Pro (002)	18.7%	$1.25 / $5.00	10.64 s
8 Gemini 1.5 Flash (002)	17.3%	$0.07 / $0.30	5.70 s
9 Llama 3.3 Instruct Turbo (70B)	16.9%	$0.88 / $0.88	11.24 s
10 Grok 2	15.2%	$2.00 / $10.00	57.88 s
11 GPT 4o (2024-08-06)	14.0%	$2.50 / $10.00	68.37 s
12 GPT 4o (2024-11-20)	12.1%	$2.50 / $10.00	15.78 s
13 GPT 4o Mini	11.5%	$0.15 / $0.60	28.77 s
14 Claude 3.5 Sonnet Latest	10.0%	$3.00 / $15.00	9.19 s
15 Mistral Large (11/2024)	9.6%	$2.00 / $6.00	19.68 s
16 Mistral Small (02/2024)	5.6%	$0.20 / $0.60	13.23 s
17 Claude 3.5 Haiku Latest	3.5%	$1.00 / $5.00	9.05 s

`https://www.vals.ai/benchmarks/aime-2025-03-11`

Reasoning chatbots

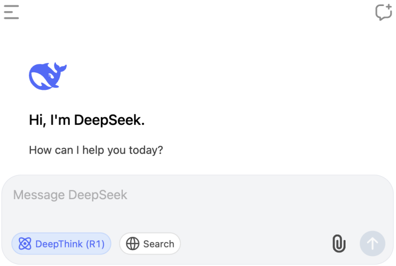
- More and more LLMs include the *reasoning* option.



What can I help with?

Ask anything

https://chatgpt.com/



Hi, I'm DeepSeek.

How can I help you today?

Message DeepSeek

https://chat.deepseek.com/

- Next, we look at possible ways to turn a next token prediction model to a reasoning model.
- **The current "reasoning" paradigm is based on RL!**

From Chain-of-Thought to Reasoning

- Chain-of-Thought (CoT) [18] is an attempt towards reasoning.

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.

- When given a new question, we expect the model to reason similarly to the examples above.
- The model first plans, then it answers.

From Chain-of-Thought to Reasoning

- With reasoning via RL, we aim at teaching the model how to reason from rewards.

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.

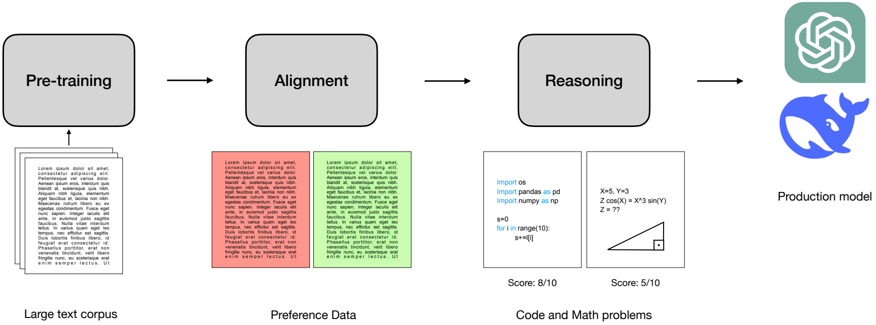
$\cdots$ </THINK>

Answer: a_i

A: The answer is 9.

- The internal CoT is not copied from examples, not prompted, but learned from "reward" maximization.

Outline



```
Import os
Import pandas as pd
Import numpy as np

s=0
for i in range(10):
      s+=l[i]
```

X=5, Y=3
Z cos(X) = X^3 sin(Y)
Z = ??

- This lecture
  1. Basics of language models
  2. Fundamentals of pre-training
  3. Reinforcement learning with human feedback (RLHF)
  4. Reasoning with reinforcement learning

More on training LLMs: Visit EE-628 at EPFL

A motivation for language models (LMs)

Example

Predict the next word w given the following source sentence S_{source}?

S_{source} : *"On January 1 people usually say happy new [w]."*

A motivation for language models (LMs)

Example

Predict the next word w given the following source sentence S_{source}?

S_{source} : *"On January 1 people usually say happy new [w]."*

Question: ○ Why is this important?

- spelling & grammar correction $\quad \mathrm{p}(\text{year}|S_{\text{source}}) > \mathrm{p}(\text{years}|S_{\text{source}})$
- machine translation $\quad \mathrm{p}(S_{\text{translation 1}}|S_{\text{source}}) > \mathrm{p}(S_{\text{translation 2}}|S_{\text{source}})$
- sentence classification $\quad \mathrm{p}(S_{\text{class 1}}|S_{\text{source}}) > \mathrm{p}(S_{\text{class 2}}|S_{\text{source}})$
- speech recognition $\quad \mathrm{p}(w|S_{\text{source}})$
- chatbot $\quad \mathrm{p}(w|S_{\text{source}})$
- (more generally) labeling, automated decisions,...

Basics for language models (LMs) – I

Definition (Language model [7])

Models that assign probabilities to sequences of words are called language models.

Remarks:

- Given a sentence with T words: $S = w_{1:T} = (w_1, \dots, w_T)$, by the chain rule of probability:

$$\mathsf{p}(S) = \mathsf{p}(w_{1:T}) = \mathsf{p}(w_1)\mathsf{p}(w_2|w_1)\mathsf{p}(w_3|w_{1:2}) \cdots \mathsf{p}(w_T|w_{1:T-1}) = \prod_{t=1}^{T} \mathsf{p}(w_t|w_{1:t-1})$$

- Implicitly, we are enforcing a graphical model that takes “time” into account.

Example

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

Basics for language models (LMs) – II

Question:

- ◦ How can we compute $\mathsf{p}(w_t|w_{1:t-1})$?

Remarks:

- ◦ A trivial solution: Just count the frequency on a large corpus, e.g.,

$$\mathsf{p}(\text{year}|S_{\text{source}}) = \frac{\mathsf{p}(S_{\text{source}} + \text{year})}{\mathsf{p}(S_{\text{source}})} \approx \frac{\#(\text{On January 1 people usually say happy new year})}{\#(\text{On January 1 people usually say happy new})}$$

- ◦ But the language is creative, there are several ways to express the same meaning.
- ◦ The sentence above might even not appear on the corpus.
- ◦ We need better ways to estimate such probabilities!

N-gram LMs

Figure: Markov in 1913 [9] used “Markov chains” to predict whether the upcoming letter would be a vowel or a consonant.

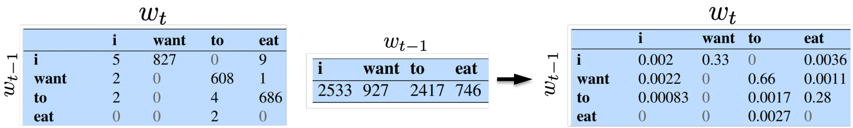
Markov assumption [9]

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

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

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.



	i	want	to	eat
i	5	827	0	9
want	2	0	608	1
to	2	0	4	686
eat	0	0	2	0

i	want	to	eat
2533	927	2417	746

	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 [7].

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 $\mathsf{p}(w_t|w_{t-1})$ over the corpus.

Inference

```
1. Set $w_1$ as $\langle \mathrm{BOS} \rangle$, $t = 1$.
2. While True:
    ▸ $w_{t+1} = \arg\max_{w \in \mathcal{V}} \mathsf{p}(w|w_t)$
    ▸ 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.

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

The optimization objective for next-token prediction

- A (vector-output) neural network $\mathbf{h}_\theta \in \Delta^{|\mathcal{V}|-1}$ can be used to model such probability.

$$-\log \mathrm{p}_\theta(\mathbf{b}_{1:T}) = -\log \left(\prod_{t=1}^{T} \mathrm{p}_\theta(\mathbf{b}_t | \mathbf{b}_{1:t-1}) \right) = \sum_{t=1}^{T} \left(-\log \underbrace{\mathrm{p}_\theta(\mathbf{b}_t | \mathbf{b}_{1:t-1})}_{\mathbf{h}_\theta(\mathbf{b}_{1:t-1})^{[\text{``}\mathbf{b}_t\text{''}]}} \right)$$

$$= \sum_{t=1}^{T} \left(-\log \mathbf{h}_\theta(\mathbf{b}_{1:t-1})^{[\text{``}\mathbf{b}_t\text{''}]} \right) = \sum_{t=1}^{T} \left(-\sum_{i=1}^{|\mathcal{V}|} \hat{\mathbf{u}}_t^{[i]} \log \mathbf{u}_t^{[i]} \right) = \text{cross entropy loss}$$

  - $\mathbf{u}_t := \mathbf{h}_\theta(\mathbf{b}_{1:t-1}) \in \mathbb{R}^{|\mathcal{V}|}$ is the probability distribution of the next word given previous $t-1$ words.
  - $\hat{\mathbf{u}}_t \in \mathbb{R}^{|\mathcal{V}|}$ is the correct distribution (one-hot) at t step.

Remarks:

- Teacher forcing training: We always give the model the correct history sequence.
- Auto-regressive inference: The history sequence comes from its prediction result.
- Notation: We will use s for prompts and a for an answer sampled from $\pi := \mathbf{h}_\theta$ in the sequel.

Alignment: going beyond next token predictions

◦ After the training, we need methods to enforce certain behaviours of the LLM.

Examples: ◦ Impose that the LLM follows instructions.

◦ Avoid bad words in the LLM responses.

◦ Make it likely that the LLM outputs sentences that match human taste.

◦ Alignment can be performed considering different inputs.

- Demonstrations
- Demonstrations + online access to the environment
- Preferences
- Preferences + online access to the environment
- Reward function

Alignment from demonstrations

- We are given a dataset $\mathcal{D}_{\text{demo}}$ of prompts $\{s_i\}_{i=1}^{N}$ and desired answers $\{a_i\}_{i=1}^{N}$.
- We aim at maximizing the likelihood of the answers in the dataset.
- This can be done via imitation learning (see Lecture 7).

Alignment via Behavioral Cloning

```
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: $\widehat{\ell}_{\text{BC}}(\theta) := \sum_{i=1}^{N_{\text{demo}}} \left[ -\log \pi_\theta \left( a_i | s_i \right) \right]$.
3: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta \in \Theta} \widehat{\ell}_{\text{BC}}(\theta)$.
```

Alignment from demonstrations + online access

- A common method for this setting is SPIN (Self-Play fIne-tuNing) [3], which considers a two player game:
  - One player tries to generate answers as similar as possible as the observed ones in $\mathcal{D}_{\text{demo}}$.
  - The other player aims at distinguish answers of the first player from the the ones in $\mathcal{D}_{\text{demo}}$.
- At iteration t, both "players" aim at solving the following bilevel formulation:

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

$$\text{s.t.} \quad r_{t+1} = \arg\min_{r\in\mathcal{R}} \mathbb{E}_{s,a\sim\mathcal{D}_{\text{demo}}} \mathbb{E}_{a'\sim\pi(\cdot|s;\theta_t)} \left[\ell(r(s,a) - r(s,a')) \right], \quad (2)$$

  - 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 $\ell(x) = -x$ corresponding to minimize an integral probability metric.
- A common choice in practice is the logistic loss function $\ell(s) = \log\big(1 + \exp(-s)\big)$.

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 $\theta_{t+1} \in \Theta$, the both problems reduce to a single optimization problem:

$$\theta_{t+1} = \arg\max_{\theta\in\Theta} \mathbb{E}_{s,a\sim\mathcal{D}_{\text{demo}}} \mathbb{E}_{a'\sim\pi(\cdot|s;\theta_t)} \left[-\ell\left(\eta^{-1}\log\left(\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_t)}\right) - \eta^{-1}\log\left(\frac{\pi(a'|s;\theta)}{\pi(a'|s;\theta_t)}\right)\right)\right]. \quad (3)$$

- **Self-study:** Compare it with DPO to be introduced in few slides.

Alignment from preferences: Setup

- A preference dataset is formed by
  - Prompts $\{s_i\}_{i=1}^N$.
  - Response pairs $\left\{a_i^+, a_i^-\right\}_{i=1}^N$.
- a_i^+ denote the preferred response to the prompt s_i according to the preference model p_{pref}.
- The preference model is a mapping as follows $\mathsf{p}_{\text{pref}} : \mathcal{A} \times \mathcal{A} \to [0, 1]$.
- $\mathsf{p}_{\text{pref}}(a, a'|s)$ is the probability that a is preferred to a' in response to s.
- We start focusing on a specific preference model known as Bradley-Terry model [2].

Bradley-Terry model

For some unknown reward function r, the preferences are generated as follows

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

where σ is the sigmoid function.

Alignment from preferences via RLHF

- The most popular learning from preferences paradigm is RLHF (RL from Human Feedback).
- RLHF follows a two step procedure.
- First, we fit a reward model $\widehat{r}$ on the preference dataset.
- Second, we learn a policy parameterization $\theta^\star$ that
  - maximizes $\widehat{r}$;
  - does not drift excessively from the pretrained model π_{ref} as measured by $\beta D_{\mathrm{KL}}(\pi(\cdot|s;\theta^\star)||\pi_{\mathrm{ref}}(\cdot|s))$.
- All in all, we aim at solving the following bilevel problem

$$\theta^\star = \underset{\theta\in\Theta}{\arg\max}\, \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)}\left[\widehat{r}(s,a;\phi)\right] - \beta\mathbb{E}_{s\sim\rho}[D_{\mathrm{KL}}(\pi\left(\cdot|s;\theta\right)\|\pi_{\mathrm{ref}}\left(\cdot|s\right))]. \tag{4}$$

$$\text{s.t.}\quad \widehat{r} = \underset{r\in\mathcal{R}}{\arg\max} \sum_{i=1}^{N} \log\sigma(r(s_i,a_i^+) - r(s_i,a_i^-)). \tag{5}$$

- β is called the "alignment parameter."
- For larger β, the solution will be more *aligned* to the reference model π_{ref}.

The RLHF algorithm

- The two steps in pseudocode look as follows.

RLHF (Reinforcement Learning from Human Feedback)

```
1: The learner receives: $(i)$ A preference dataset $\mathcal{D}_{\text{pref}} = \{(s_i, a_i^+, a_i^-)\}_{i=1}^{N_{\text{pre}}}$, $(ii)$ A prompt dataset $\mathcal{D}_{\text{prompts}} = \{s_i\}_{i=1}^{N_{\text{prompts}}}$, $(iii)$ A reward function class $\mathcal{R}$, $(iv)$ A policy function parameter class $\Theta$.
2: The learner estimates the reward as $r(\cdot, \cdot; \phi^\star)$ (typically an NN with parameters $\phi$) via
       $\widehat{r} = \arg\max_{r \in \mathcal{R}} \sum_{i=1}^{N} \log\Big(\sigma\Big(r(s_i, a_i^+; \phi) - r(s_i, a_i^-; \phi)\Big)\Big).$   (Reward Loss)
3: The learner then approximately solves the following problem:
       $\theta^\star \in \arg\max_{\theta \in \Theta} \mathbb{E}_{s \sim \rho, a \sim \pi(\cdot|s)} \left[\widehat{r}(s, a; \phi)\right] - \beta \mathbb{E}_{s \sim \rho}[D_{\mathrm{KL}}(\pi(\cdot|s; \theta) \| \pi_{\text{ref}}(\cdot|s))],$   (RLHF)
   using PPO or REINFORCE. Note that we do not have all the ingredients to solve the problem.
```

- Thereoretical improvements can be obtained replacing the KL with the sum of KL and χ^2 divergences [4].

Learning from preferences without online access

- RLHF requires to generate new responses to the prompt dataset.
- It is more computationally efficient to leverage only the preference dataset.
- DPO (Direct Preference Optimization) [14] was introduced towards this goal.

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.

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

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

DPO derivation (Part 1)

- For a fixed $r(\cdot, \cdot; \phi)$, the solution to (RLHF) (for expressive enough policy classes!) is given by

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

  where we define the partition function as $Z(s,\phi,\beta) := \sum_{a'} \pi_{\text{ref}}(a'|s)\exp(\beta^{-1} r(s,a';\phi))$.

- By rearranging, it holds that for all s, a that $\log(\pi_\phi^\star(a|s)) = \log(\pi_{\text{ref}}(a|s)) + \beta^{-1} r(s,a;\phi) - \log(Z(s,\phi,\beta))$.
- Such a quantity cannot be computed in closed form because computing $Z(s,\phi,\beta)$ is intractable.
- For two possible answers a, a' to the *same* question s, it holds that

$$\log(\pi_\phi^\star(a|s)) - \log(\pi_\phi^\star(a'|s)) = \log(\pi_{\text{ref}}(a|s)) - \log(\pi_{\text{ref}}(a'|s)) + \beta^{-1} r(s,a;\phi) - \beta^{-1} r(s,a';\phi).$$

DPO derivation (Part 2)

- Therefore, the reward functions difference can be computed as

$$r(s,a;\phi) - r(s,a';\phi) = \beta \log\left(\frac{\pi^\star_\phi(a|s)}{\pi_{\mathrm{ref}}(a|s)}\right) - \beta \log\left(\frac{\pi^\star_\phi(a'|s)}{\pi_{\mathrm{ref}}(a'|s)}\right). \tag{6}$$

- The computation is efficient because the normalization constant $Z(s,\phi,\beta)$ does not appear.
- It follows that we can plug in the above analytical solution into (Reward Loss)

$$\begin{aligned} \min_{\phi\in\Phi}\ \ell(\phi) &:= -\mathbb{E}_{s\sim\rho,(a^-\prec a^+)\sim \mathsf{p}_{\mathrm{pref}}(\cdot|s)}\left[\log\left(\sigma\left(\beta\log\left(\frac{\pi^\star_\phi(a^+|s)}{\pi_{\mathrm{ref}}(a^+|s)}\right) - \beta\log\left(\frac{\pi^\star_\phi(a^-|s)}{\pi_{\mathrm{ref}}(a^-|s)}\right)\right)\right)\right] \\ \text{s.t.}\quad \pi^\star_\phi &:= \arg\max_{\pi\in\Pi}\ \mathbb{E}_{s\sim\rho,a\sim\pi(\cdot|s)}\left[r(s,a;\phi)\right] - \beta\mathbb{E}_{s\sim\rho}[D_{\mathrm{KL}}(\pi(\cdot|s)\,\|\pi_{\mathrm{ref}}(\cdot|s))]. \end{aligned}$$

- The bilevel problem is still too complicated, DPO [14] is derived to ignore the lower level problem.

$$\arg\min_{\theta\in\Theta}\ \ell_{\mathrm{DPO}}(\theta) := -\mathbb{E}_{s\sim\rho,(a^-\prec a^+)\sim \mathsf{p}_{\mathrm{pref}}(\cdot|s)}\left[\log\left(\sigma\left(\beta\log\left(\frac{\pi(a^+|s;\theta)}{\pi_{\mathrm{ref}}(a^+|s)}\right) - \beta\log\left(\frac{\pi(a^-|s;\theta)}{\pi_{\mathrm{ref}}(a^-|s)}\right)\right)\right)\right]. \tag{7}$$

DPO vs RLHF

- DPO is easier computationally. It uses only one neural network.
- DPO and RLHF are not equivalent because we dropped the constraint on the policy in the last step.
- Therefore, the solution set of the DPO optimization problem includes the RLHF solution set [20].
- Some DPO solutions assign high probability to answers unseen in the preference dataset.
- To fix the above issue one can constrain the probability mass moved [1].
- For the above reason, DPO requires using early stopping in practice.
- DPO can be extended in the multi stage setting [13].

Some limitations of the Bradley-Terry model

- The Bradley-Terry model can only capture transitive preferences.
- Averaging across humans might not give a dataset where transitivity holds.
- As an example, let us consider 3 humans h_1, h_2, h_3 and 3 possible answers y_1, y_2, y_3.
- Let us denote $\mathsf{p}^h_{\mathrm{pref}}$ the preference model of human h defined as follows

$$\begin{array}{lll} \mathsf{p}^{h_1}_{\mathrm{pref}}(y_1, y_2) = 1 & \mathsf{p}^{h_1}_{\mathrm{pref}}(y_2, y_3) = 0 & \mathsf{p}^{h_1}_{\mathrm{pref}}(y_3, y_1) = 1 \\ \mathsf{p}^{h_2}_{\mathrm{pref}}(y_1, y_2) = 0 & \mathsf{p}^{h_2}_{\mathrm{pref}}(y_2, y_3) = 1 & \mathsf{p}^{h_2}_{\mathrm{pref}}(y_3, y_1) = 1 \\ \mathsf{p}^{h_3}_{\mathrm{pref}}(y_1, y_2) = 1 & \mathsf{p}^{h_3}_{\mathrm{pref}}(y_2, y_3) = 1 & \mathsf{p}^{h_3}_{\mathrm{pref}}(y_3, y_1) = 0. \end{array}$$

- Each of these models is transitive.
- However, the average model defined as $\mathsf{p}_{\mathrm{pref}}(y, y') = \frac{1}{3} \sum_{h \in \{h_1, h_2, h_3\}} \mathsf{p}^h_{\mathrm{pref}}(y, y')$ satisfies

$$\mathsf{p}_{\mathrm{pref}}(y_1, y_2) = \mathsf{p}_{\mathrm{pref}}(y_2, y_3) = \mathsf{p}_{\mathrm{pref}}(y_3, y_1) = 2/3.$$

- That is, the average model is non transitive and can not be modeled by the BT assumption.

Nash learning from human feedback (NLHF)

○ NLHF allows to use general (possibly non transitive) preference models.

Nash Learning from Human Feedback [11]

```
1: The learner estimates the preference model as $\mathsf{p}_{\mathrm{pref}}(\cdot, \cdot | \cdot; \phi^\star)$ with
       $$\phi^\star = \arg\min_{\phi \in \Phi} \widehat{\ell}_{\mathrm{PM}}(\phi) := -\sum_{i=1}^{N} \log \left( \mathsf{p}_{\mathrm{pref}}(a_i^+, a_i^- | s^i; \phi) \right).$$
2: Sample $a^i \sim \pi(\cdot | s^i; \theta)$ and $a'^{,i} \sim \pi(\cdot | s^i; \theta')$ for all $i \in [N]$.
3: The learner computes the stochastic objective
       $$\widehat{\ell}_{\mathrm{NLHF}}(\theta, \theta') := -\frac{1}{|\mathcal{D}_{\mathrm{prompts}}|} \sum_{i=1}^{N} \mathsf{p}_{\mathrm{pref}}(a^i, a'^{,i} | s^i; \phi^\star) + \beta D_{\mathrm{KL}}(\pi\left(\cdot | s^i; \theta\right) \| \pi_{\mathrm{ref}}\left(\cdot | s^i\right)) - \beta D_{\mathrm{KL}}(\pi\left(\cdot | s^i; \theta'\right) \| \pi_{\mathrm{ref}}\left(\cdot | s^i\right)),$$
4: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta \in \Theta} \max_{\theta' \in \Theta} \widehat{\ell}_{\mathrm{NLHF}}(\theta, \theta')$.
```

Finding a saddle point of $\widehat{\ell}_{\mathrm{NLHF}}(\theta, \theta')$

- From a computational perspective general preferences are harder.
- The reason is that we now need to solve a minmax problem.
- Notice that under the Bradley-Terry assumption a minimization was enough.
- Simple solving with gradient descent ascent has two problem:
  - No last iterate guarantees.
  - Slow $\mathcal{O}(1/\sqrt{T})$ rate.
- In the next slides we see how to overcome these two challenges.

Strongly convex, strongly concave case: Nash MD

- For $\beta > 0$ the problem is strongly convex-strongly concave.
- Assuming above and a **single state**, [11] showed that Nash-MD has $\mathcal{O}(1/T)$ last iterate convergence.

Nash-MD

```
π_1 = π_ref, learning rate η.
for t = 1, ..., T do
    Compute mixture between initial policy and π_t
        $$\bar{\pi}_t(a) = \frac{\pi_t(a)^{1-\beta\eta}\pi_{\mathrm{ref}}(a)^{\beta\eta}}{\sum_{b\in\mathcal{A}}\pi_t(b)^{1-\beta\eta}\pi_{\mathrm{ref}}(b)^{\beta\eta}}.$$
    Compute the averaged preference model
        $$\mathsf{p}_{\mathrm{pref}}^{\bar{\pi}_t}(a) = \sum_{b\in\mathcal{A}}\bar{\pi}_t(b)\mathsf{p}_{\mathrm{pref}}(a,b).$$
    Mirror descent step with gradient evaluated in $\bar{\pi}_t$: $\pi_{t+1}(a) = \arg\min_{\pi\in\Pi}\left[\langle \pi, \mathsf{p}_{\mathrm{pref}}^{\bar{\pi}_t}\rangle + \frac{1}{\eta}D_{\mathrm{KL}}(\pi,\bar{\pi}_t)\right].$
end for
```

- Is Nash MD applying gradient descent ascent on $\widehat{\ell}_{\mathrm{NLHF}}(\theta,\theta')$?

Gradient descent ascent

- Almost! Find the differences in red.

Gradient Descent Ascent

```
$\pi_1 = \pi_{\text{ref}}$, learning rate $\eta$.
for $t = 1, \dots, T$ do
    Compute mixture between initial policy and $\pi_t$
        $$\bar{\pi}_t(a) = \frac{\pi_t(a)^{1-\beta\eta}\pi_{\text{ref}}(a)^{\beta\eta}}{\sum_{b\in\mathcal{A}} \pi_t(b)^{1-\beta\eta}\pi_{\text{ref}}(b)^{\beta\eta}}.$$
    Compute the averaged preference model
        $$\mathsf{p}_{\text{pref}}^{\pi_t}(a) = \sum_{b\in\mathcal{A}} \pi_t(b)\mathsf{p}_{\text{pref}}(a,b).$$
    Mirror descent step with gradient evaluated in $\pi_t$: $\pi_{t+1}(a) = \arg\min_{\pi\in\Pi}\left[\langle \pi, \mathsf{p}_{\text{pref}}^{\pi_t}\rangle + \frac{1}{\eta}D_{\text{KL}}(\pi, \bar{\pi}_t)\right]$.
end for
```

- Since the game is antisymmetric the two players produces exactly the same iterates.

- That's why we talk about gradient descent ascent but it is enough to keep only one sequence of iterates.

Multi Turn Preference Optimization (MTPO)

◦ [15] uses Gradient Descent Ascent in the multi state setting deriving MTPO.

MTPO

```
$\pi_1 = \pi_{\text{ref}}$, learning rate $\eta$.
for $t = 1, \ldots, T$ do
    Let $Q^{\pi,\pi'}$ denote the state action value functions associated to th preference model $\mathrm{p}_{\text{pref}}$.
    Apply the gradient descent ascent at each state $s$ using $Q^{\pi,\pi'}$:
        ▸ $\bar{\pi}_t(a|s) = \frac{\pi_t(a)^{1-\beta\eta}\pi_{\text{ref}}(a)^{\beta\eta}}{\sum_{b\in\mathcal{A}}\pi_t(b)^{1-\beta\eta}\pi_{\text{ref}}(b)^{\beta\eta}}$.
        ▸ $\underline{Q}^{\pi_t,\pi_t}(s,a) = \sum_{b\in\mathcal{A}}\pi_t(b)Q^{\pi_t,\pi_t}(s,a,b)$.
        ▸ $\pi_{t+1}(\cdot|s) = \arg\min_{\pi\in\Pi}\left[\langle\pi(\cdot|s), \underline{Q}^{\pi_t,\pi_t}(s,\cdot)\rangle + \frac{1}{\eta}D_{\text{KL}}(\pi(\cdot|s), \bar{\pi}_t(\cdot|s))\right]$.
end for
```

◦ MTPO can be seen to converge at a $\mathcal{O}(1/T)$ rate in the strongly convex concave setting.

◦ In the convex concave setting instead gradient descent ascent and therefore MTPO can divergence.

Optimism to fix it

- Let us revisit optimistic gradient descent.
- Let Δ denote the probability simplex.

Optimistic gradient descent ascent (OGDA)

```
Initialize $x_1$ uniformely.
for $t = 1, \dots, T$ do
    Update the decision avariable $x_{t+1}$ as follows:
        $$x_{t+1} = \arg\min_{x \in \Delta} \left[ \langle x, 2\nabla f(x_t) - \nabla f(x_{t-1}) \rangle + \frac{1}{\eta} D_{\mathrm{KL}}(x, x_t) \right]$$
    .
end for
```

- The average iterate of ODGA converges at $\mathcal{O}(1/T)$ for convex concave games.
- Asymptotically the last iterate converges strictly faster.
- The rate becomes $o(1/t)$ for all t large enough.
- [19] applies this technique to obtain faster convergence rate in NLHF.

Learning to reason with RL

User Prompt: s_i

"A rectangle's length is three times its width. If the perimeter of the rectangle is 64 units, what are its dimensions?"

Internal Chain of Though (CoT): b_i

<THINK>
Step 1: Let the width be s. Then the length is $3x$.

Step 2: The perimeter is given by $P = 2(s + 3x) = 8x$. So, $8x = 64$ and $s = 8$.

Step 3: Calculate the length: $3x = 3 \times 8 = 24$.

</THINK>

Final Answer: a_i

<ANSWER> The rectangle has a width of 8 units and a length of 24 units. </ANSWER> r_i

- **Question:** How can we learn the internal CoT from questions s_i and rewards r_i?

Group relative policy optimization (GRPO)

- **Problem:** The PPO algorithm requires estimating a value function, which is as big as our LLM.

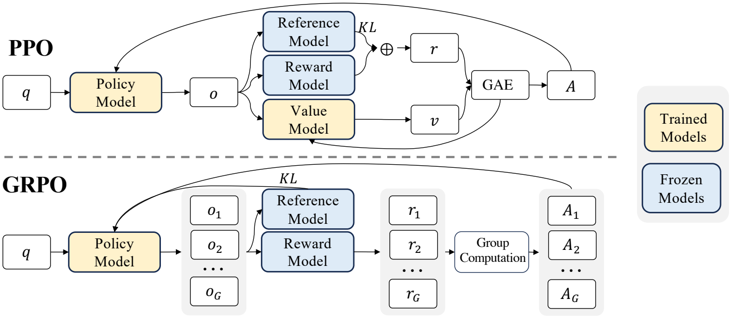


Figure: Source: DeepSeekMath `https://arxiv.org/abs/2402.03300`

- **Solution:** Estimate A by sampling several answers and computing relative rewards within the sample.
- Previous work had similar ideas for removing the value model dependency [8].

Group relative policy optimization (GRPO)

GRPO (optimization formulation) [16]

$$\max_{\theta} \quad \mathbb{E}_{s' \sim \lambda_{\mu}^{\pi_{\theta_t}}, \{a_i\}_{i=1}^{G} \sim \pi_{\theta_t}(\cdot|s)} \frac{1}{G} \sum_{i=1}^{G} \min \left\{ \frac{\pi_{\theta}(a_i|s)}{\pi_{\theta_t}(a_i|s)} A_i^{\pi_{\theta_t}}(s, a_i), \mathrm{clip}\left(\frac{\pi_{\theta}(a_i|s)}{\pi_{\theta_t}(a_i|s)}; 1-\epsilon; 1+\epsilon \right) A_i^{\pi_{\theta_t}}(s, a) \right\}$$
$$- \beta \mathrm{KL}(\pi_{\theta}(\cdot|s) || \pi_{\text{ref}}(\cdot|s))$$

Remarks:

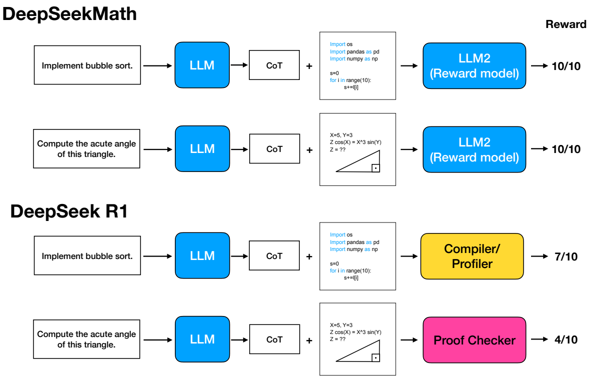
- The advantages are estimated as $A_i^{\pi_{\theta_t}}(s, a_i) = \frac{r(s,a_i) - G^{-1}\sum_{i=1}^{G} r(s,a_i)}{\sqrt{G^{-1}\sum_{i=1}^{G}\left(r(s,a_i) - G^{-1}\sum_{i=1}^{G} r(s,a_i)\right)^2}}$.
- Here G represents the number of actions sampled as a group at each state, i.e., its group size.
- GRPO uses the following unbiased estimator [6] to estimate the KL divergence:

$$\mathrm{KL}(\pi_{\theta}(\cdot|s) || \pi_{\text{ref}}(\cdot|s)) = \frac{\pi_{\text{ref}}(a_i|s)}{\pi_{\theta}(a_i|s)} - \log \frac{\pi_{\text{ref}}(a_i|s)}{\pi_{\theta}(a_i|s)} - 1$$

- By computing group relative rewards, GRPO reduces update variance and ensures stable learning.
- Moreover, GRPO avoids the use of critics. This way it is more memory efficient.

DeepSeek R1: training on raw rewards

- In the DeepSeekMath paper, the rewards still come from a reward model.
- **Problem:** The reward model is not perfect and overoptimizing can lead to *reward hacking*.



DeepSeekMath

Reward

Implement bubble sort. → LLM → CoT +

```
Import os
Import pandas as pd
Import numpy as np

s=0
for i in range(10):
    s+=l[i]
```

→ LLM2 (Reward model) → 10/10

Compute the acute angle of this triangle. → LLM → CoT +

X=5, Y=3
Z cos(X) = X^3 sin(Y)
Z = ??

→ LLM2 (Reward model) → 10/10

DeepSeek R1

Implement bubble sort. → LLM → CoT +

```
Import os
Import pandas as pd
Import numpy as np

s=0
for i in range(10):
    s+=l[i]
```

→ Compiler/ Profiler → 7/10

Compute the acute angle of this triangle. → LLM → CoT +

X=5, Y=3
Z cos(X) = X^3 sin(Y)
Z = ??

→ Proof Checker → 4/10

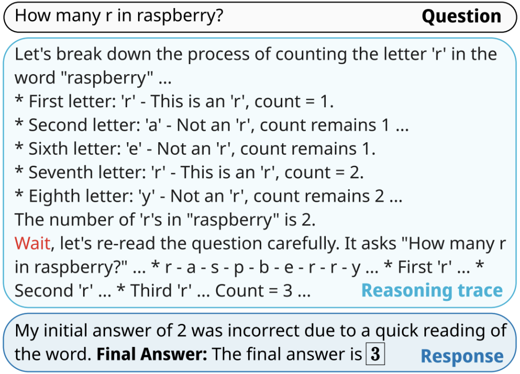
- **Solution:** We can query code compiler/interpreters and proof checkers to get high quality rewards.

Other approaches

- OpenAI were the first to use RL for reasoning, with no details disclosed [12].
- Kimi 1.5 uses an off-policy ℓ_2 regularized policy gradient method with the same baseline as GRPO [17].
- Many details about the reward model of DeepSeekR1 remain undisclosed.
- HuggingFace is organizing an open source reproduction of DeepSeekR1: OpenR1 [5].
- **Question:** is RL really needed?

Simple reasoning without RL

- Through simple sampling tricks, we can replicate the reasoning behavior of reasoning LLMs [10]



How many r in raspberry?

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 ...

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

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

- Instead of letting the model answer "2," we can force the generation for the token "Wait" and continue.
- This replicates the reasoning behavior obtained with RL and leads to improve performance without RL.

Summary of RL for language models

Remarks:
- RLHF is needed to align the model with human preferences.
- RLHF is a two-step process: reward modeling and policy optimization.
- DPO is a more efficient alternative to RLHF.
- Reasoning is critical for improving the performance of production models.

Thank you!

- Let's start with the project!

References I

[1] Kavosh Asadi, Julien Han, Xingzi Xu, Dominique Perrault-Joncas, Shoham Sabach, Karim Bouyarmane, and Mohammad Ghavamzadeh.
C-3dpo: Constrained controlled classification for direct preference optimization.
arXiv preprint arXiv:2502.17507, 2025.
26

[2] 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.
20

[3] 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.
18

[4] 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.
22

[5] Huggingface.
Open r1.
38

References II

[6] Joschu.
Kl approximation.
`http://joschu.net/blog/kl-approx.html`, 2017.
Accessed: March 09, 2025.
36

[7] Dan Jurafsky and James H. Martin.
Speech and Language Processing (3rd ed. draft).
draft, third edition, 2023.
10, 12

[8] 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*, 2024.
35

[9] 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.
12

References III

[10] 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.

39

[11] 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.

28, 30

[12] OpenAI.

Learning to reason with llms.

38

[13] Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn.

From r to $q^{\star}$: Your language model is secretly a q-function.

arXiv preprint arXiv:2404.12358, 2024.

26

[14] 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.

23, 25

References IV

[15] 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.

32

[16] 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.

36

[17] 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.

38

[18] 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

References V

[19] Yongtao Wu, Luca Viano, Yihang Chen, Zhenyu Zhu, Kimon Antonakopoulos, Quanquan Gu, and Volkan Cevher.
Multi-step alignment as markov games: An optimistic online gradient descent approach with convergence guarantees.
arXiv preprint arXiv:2502.12678, 2025.
33, 48

[20] 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.
26

Supplementary material

OMPO: Using OGDA in Nash learning from Human Feedback

- The previous slides highlights results in convex concave setting.
- However, in the multi turn setting ℓ_{NLHF} is clearly non-convex and non-concave.
- [19] bypasses the problem resorting to LP techniques.
- Indeed, [19] 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

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

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