17-740 Spring 2025
Algorithmic Foundations of Interactive Learning

Lecture 21: The Information Geometry of RLHF

Lecturer: Gokul Swamy
Scribe: Zichun Y., Yiming Z., Lintang S., Alfredo G.

21.1 Outline

The lecture addresses three main points that characterizes Reinforcement Learning from Human Feedback (RLHF) for language models (LM):

1. What is the fine-tuning problem?
   A: Regularized maximum likelihood estimation (MLE).
2. End-to-end, what is the process of RLHF doing?
   A: A two-stage process consisting of MLE over reward models followed by MaxEnt (Entropy-Regularized RL) over policies.
3. What are direct alignment algorithms?
   A: Algorithms that directly maximize likelihood over the policy space Π *without explicitly passing through the reward model space* $\mathcal{R}$.

21.2 Motivation: The Era of Fine-Tuning

Pretrained language models such as GPT-3/4 [1, 2] learn a wide range of capabilities and statistical patterns from vast amounts of textual data mainly sourced from unstructured web data. However, their behavior might not align perfectly with human intentions or desired interaction styles (e.g., instruction following, helpfulness, harmlessness).

Fine-tuning, particularly using methods like RLHF [3], aims to "steer" these base models towards desired behaviors. This alignment process engineers the model's outputs to better match human preferences, leading to models like InstructGPT [3] or ChatGPT, which exhibit significantly improved instruction-following and conversational abilities compared to their base counterparts. The core problem we are trying to solve during this fine-tuning or alignment phase can be framed as regularized maximum likelihood estimation. However, it is really hard to write an effective reward function by hand.

21.3 The Standard Three Steps of RLHF

The standard RLHF pipeline typically involves three stages:

1. **Supervised Fine-Tuning (SFT)**: (Also referred to as Imitation Learning - IL). A pre-trained language model is fine-tuned on a dataset of high-quality human demonstrations or instructions. For various prompts, human labelers provide the desired outputs. The model learns to mimic these expert responses (behavior cloning). Let the policy after this stage be π_{SFT} or π_{ref}.

2. **Reward Model (RM) Training**: Reward models are specific models that can asses the quality of a model response or in this case how preferable a response is to a human. While in theory, RMs can be any function that provides a score given a response, in the case of RLHF, it is often initialized from the SFT model. This involves collecting comparison data: for a given prompt, multiple outputs are generated (e.g., from the SFT model), and human labelers rank these outputs or choose the best one (pairwise comparison). The reward model learns a scalar function $r_\phi(\text{prompt}, \text{completion})$ that assigns higher scores to preferred responses. This is essentially a classification or regression problem aiming to model the human preference distribution.

3. **Reinforcement Learning (RL) Optimization**: The language model (initialized from the SFT model) is further optimized using RL. The goal is to maximize the expected reward predicted by the trained reward model r_ϕ. To prevent the policy from deviating too much from the SFT model (which has good generative capabilities and prevents reward hacking), a KL-divergence penalty term is added to the objective. Common RL algorithms used include PPO (Proximal Policy Optimization)[4], but other policy gradient methods (like REINFORCE[5], REBEL[6], GRPO[7]) can also work effectively with proper tuning.

This process iteratively refines the language model to align better with human preferences while retaining its core language capabilities.

21.4 Language Modeling as a Markov Decision Process (MDP)

A language model predicts the probability of the next token w_t given the preceding tokens $w_t \sim P(w_t|w_{t-1}w_{t-2}\cdots w_1)$. We can think of language modeling as a special MDP.

- **Prompts as Initial States (s_0)** We can regard prompts s_0 as an initial state sampled from a distribution $s_0 \sim p_0$. LMs are typically provided instructions as input which initializes the language model to provide a sequence of continuation text.
- **Next Token Predictions as Actions (a_t)**: Provided an initial state s_0, a model predicts the next token from a vocabulary / *action space*. We denote this as $a_t \in \mathcal{A}$.
- **Generated Tokens as State (s_t)**: At each successive step t, the sequence of tokens thus far can be regarded as the current state on which a model is conditioned on to predict the next token. This way, we can write any arbitrary state s_t as a concatenation of the initial state and the sequence of actions (or tokens) predicted up to time step t, $s_t = [s_0, a_1, a_2, \ldots, a_t]$. Thus, our state space is $\mathcal{S} = \mathcal{A}^H$, where H is the horizon.
- **The LM as the Policy (π)**: If we define generated tokens s_{t-1} as state and the next token a_t as action, the language model itself becomes the policy $\pi(a_t|s_{t-1})$ that provides the probability distribution over the next token given the preceding sequence.
- **Transitions (T)**: The transitions are deterministic and known. Given state s and action (token) a, the next state s' is simply the concatenation $s' = [s, a]$.

  $$T(s' \mid s, a) = \begin{cases} 1 & \text{if } s' = [s, a] \\ 0 & \text{otherwise} \end{cases}$$

  This structure forms a tree rooted at the prompt s_0.
- **Horizon (H)**: A maximum generation length H is typically set which limits the generated sequence to $s_H = [s_0, a_1, a_2, \ldots, a_H]$. Generation can end earlier if an end-of-sequence token is produced.
- **Reward (r)**: In the RLHF context, the reward is typically assigned only at the end of the generation (at horizon H or when an end-of-sequence token is generated). This is because it is often hard to break down the quality of some text into a sum of per-word scores. The reward function $r(s_H)$ or $r(\xi)$ (where $\xi = (a_1, \ldots, a_H)$ is the full completion) is given by the output of the trained reward model from Step 2.

Remark 1 (Special Characteristics of the Language MDP)

1. Dynamics are deterministic, known, and tree-structured.
2. Resets are easy: starting a new generation from any prefix (state) is trivial.
3. The reward function $r(s_H)$ is non-Markovian with respect to the token-level states s_t (it depends on the entire sequence) and doesn't naturally decompose per token.

Remark 2 *Since the reward is only given at the end based on the complete trajectory (completion), this MDP can be analyzed as a contextual bandit problem, where the context is the prompt s_0 and the "action" is the entire generated sequence ξ.*

Remark 3 (Fixed Horizon H) *While language generations vary in length, using a fixed horizon H is often practical. Models can generate a special "$\langle eos \rangle$" token. Tokens generated after "$\langle eos \rangle$" can be considered padding and ignored, effectively handling variable lengths within the fixed horizon framework.*

21.5 Preference Fine-Tuning: The Data and Goal

Setup: The dataset $\mathcal{D}$ consists of tuples (s_0, ξ^+, ξ^-), where:

- $s_0 \sim p_0$ is a prompt.
- ξ^+ and ξ^- are two completions generated for the prompt s_0, often sampled from the SFT policy π_{ref} (i.e., $\xi^+, \xi^- \sim \pi_{\text{ref}}(\cdot|s_0)$).
- A human label indicates that ξ^+ is preferred over ξ^- (denoted $\xi^+ \succ \xi^-$).

Optimization Problem: The ultimate goal of RLHF is to find a policy π that generates completions aligning with human preferences, while staying "close" to the initial SFT policy π_{ref}. This "closeness" is enforced by a (reverse) KL penalty. The fine-tuning problem can be seen as finding a policy $\pi^\star$ that minimizes some loss on the preference data $\mathcal{D}$, regularized by its distance to π_{ref}:

$$\pi^\star = \operatorname*{argmin}_{\pi \in \Pi} \mathbb{D}_{\text{KL}}(\mathcal{D}||\pi) + \beta \mathbb{D}_{\text{KL}}(\pi||\pi_{\text{ref}}) \tag{21.1}$$

where $\mathcal{L}_{\mathcal{D}}(\pi)$ measures how well π explains the preferences in $\mathcal{D}$ (i.e., data likelihood), and β controls the strength of prior regularization. Intuitively, the reason we need regularization to a prior is that $\mathcal{D}$ often doesn't cover all possible generations, which means we don't want our policy to start generating text that we haven't received human feedback on.

Pro vs SFT: Collecting preference data is often easier and cheaper than asking experts to write high-quality demonstrations.

Con vs. SFT: Each data point provides only relative information (e.g., 1 bit for pairwise comparison), which is much less informative than a full demonstration.

21.6 Two-Stage RLHF: Information Geometry View

Let's analyze the standard two-stage process (RM training + RL optimization) using concepts from the field of *information geometry*.

21.6.1 Stage 1: Reward Modeling as MLE (FKL Projection)

We typically model human preferences using the Bradley-Terry (BT) model. It assumes an underlying latent reward function r shared across the entire population such that the probability of preferring ξ_1 over ξ_2 given prompt s_0 is:

$$\mathbb{P}_r(\xi_1 \succ \xi_2 | s_0) = \sigma(r(\xi_1) - r(\xi_2)) \tag{21.2}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function.

Remark 4 *This model assumes consistent preferences across the population, which frequently does not hold in practice. In such situations, intransitivity might occur from preference aggregation, which means that* ***no*** *reward function can explain the observed preferences. We will discuss how to deal with this issue in a later lecture.*

Also, let $\mathbb{P}_{\mathcal{D}}(\xi_1 \succ \xi_2 | s_0)$ denote the empirical probability (frequency) in the dataset $\mathcal{D}$ that ξ_1 was preferred over ξ_2 for prompt s_0 by our pool of raters..

A reward model is nothing but a classifier. Training the reward model r_ϕ (parameterized by ϕ, often within a space $\mathcal{R}$) is performed by maximizing the likelihood of the observed human preferences in $\mathcal{D}$. This is equivalent to minimizing the forward KL divergence from the empirical preference distribution $\mathbb{P}_{\mathcal{D}}$ to the model's predicted preference distribution $\mathbb{P}_{r_\phi}$:

$$\hat{r}_{\text{mle}} = r^*_\phi = \underset{r_\phi \in \mathcal{R}}{\text{argmin}} \mathbb{E}_{s_0 \sim \mathcal{D}} \left[\mathbb{D}_{\text{KL}}(\mathbb{P}_{\mathcal{D}}(\cdot \succ \cdot | s_0) || \mathbb{P}_{r_\phi}(\cdot \succ \cdot | s_0)) \right] \tag{21.3}$$

$$= \underset{r_\phi \in \mathcal{R}}{\text{argmax}} \mathbb{E}_{(s_0, \xi^+, \xi^-) \sim \mathcal{D}} \left[\log \mathbb{P}_{r_\phi}(\xi^+ \succ \xi^- | s_0) \right] \tag{21.4}$$

$$= \underset{r_\phi \in \mathcal{R}}{\text{argmax}} \mathbb{E}_{(s_0, \xi^+, \xi^-) \sim \mathcal{D}} \left[\log \sigma(r_\phi(\xi^+) - r_\phi(\xi^-)) \right] \tag{21.5}$$

Observe that this is precisely the objective function for training a classifier using logistic regression on the preference pairs. From an information geometry perspective, this MLE step is performing a **Forward KL projection** (FKL) of the empirical preference distribution $\mathbb{P}_{\mathcal{D}}$ onto the space of realizable preference distributions induced by reward models in $\mathcal{R}$.

21.6.2 Stage 2: RL Optimization as MaxEnt RL (RKL Projection)

In the second stage, we use the learned reward model $\hat{r}_{\text{mle}}$ to optimize the policy π (parameterized by θ, within a space Π). The objective is to maximize the expected reward under the policy π, while regularizing with the reverse KL divergence to the reference policy π_{ref} (usually the SFT policy) to prevent *reward hacking* caused by limited $\mathcal{D}$ coverage:

$$\hat{\pi}_{\text{rlhf}} = \underset{\pi \in \Pi}{\text{argmax}} \mathbb{E}_{\xi \sim \pi(\cdot|s_0)} \left[\hat{r}_{\text{mle}}(\xi)\right] - \beta \mathbb{D}_{\text{KL}}(\pi(\cdot|s_0)||\pi_{\text{ref}}(\cdot|s_0)) \tag{21.6}$$

Here, β is a hyperparameter controlling the regularization strength. The KL divergence term is calculated over entire sequences ξ:

$$\mathbb{D}_{\text{KL}}(\pi||\pi_{\text{ref}}) = \mathbb{E}_{\xi \sim \pi}\left[\log \frac{\pi(\xi|s_0)}{\pi_{\text{ref}}(\xi|s_0)}\right] = \mathbb{E}_{\xi \sim \pi}\left[\sum_{h=1}^{H} \log \frac{\pi(a_h|s_{h-1})}{\pi_{\text{ref}}(a_h|s_{h-1})}\right] \tag{21.7}$$

This is a standard objective in entropy-regularized RL or "Soft RL". Recall from a past lecture that the optimal policy $\pi^\star$ for this objective has the form:

$$\mathbb{P}^\star_{\hat{r}_{\text{mle}}}(\xi|s_0) = \frac{1}{Z(s_0)} \mathbb{P}_{\text{ref}}(\xi|s_0) \exp\left(\frac{1}{\beta}\hat{r}_{\text{mle}}(\xi)\right) = \prod_h^H \pi^\star_{\hat{r}_{\text{mle}}}(a_h|s_h), \tag{21.8}$$

where $Z(s_0)$ is the partition function ensuring the distribution sums to 1 over all possible sequences ξ starting from s_0, and the RHS equality uses the fact that the dynamics are deterministic. We don't discuss the proof in lecture but it can be shown that solving this soft RL problem over some policy class Π is equivalent to projecting $\mathbb{P}^\star_{\hat{r}_{\text{mle}}}$ onto the space of trajectory distributions $\mathbb{P}_\pi$ induced by $\pi \in \Pi$ under the reverse KL metric:

$$\hat{\pi}_{\text{rlhf}} = \underset{\pi \in \Pi}{\text{argmin}}\, \mathbb{D}_{\text{KL}}(\mathbb{P}_\pi||\mathbb{P}^\star_{\hat{r}_{\text{mle}}}) \tag{21.9}$$

End-to-End Summary: The standard two-stage RLHF process first performs an FKL projection from the data $\mathcal{D}$ to the reward space $\mathcal{R}$ (MLE for $\hat{r}_{\text{mle}}$), and then an RKL projection from the reward-induced target distribution $\mathbb{P}^\star_{\hat{r}_{\text{mle}}}$ to the policy space Π (MaxEnt RL for $\hat{\pi}_{\text{rlhf}}$). We visualize this process in Figure 21.1.

21.7 Direct Alignment Algorithms (e.g., DPO)

Direct Alignment Algorithms aim to bypass the explicit reward modeling step (Stage 1) and directly optimize the policy π using the preference data $\mathcal{D}$ via offline maximum likelihood estimation. Direct Preference Optimization (DPO) is a prominent example.

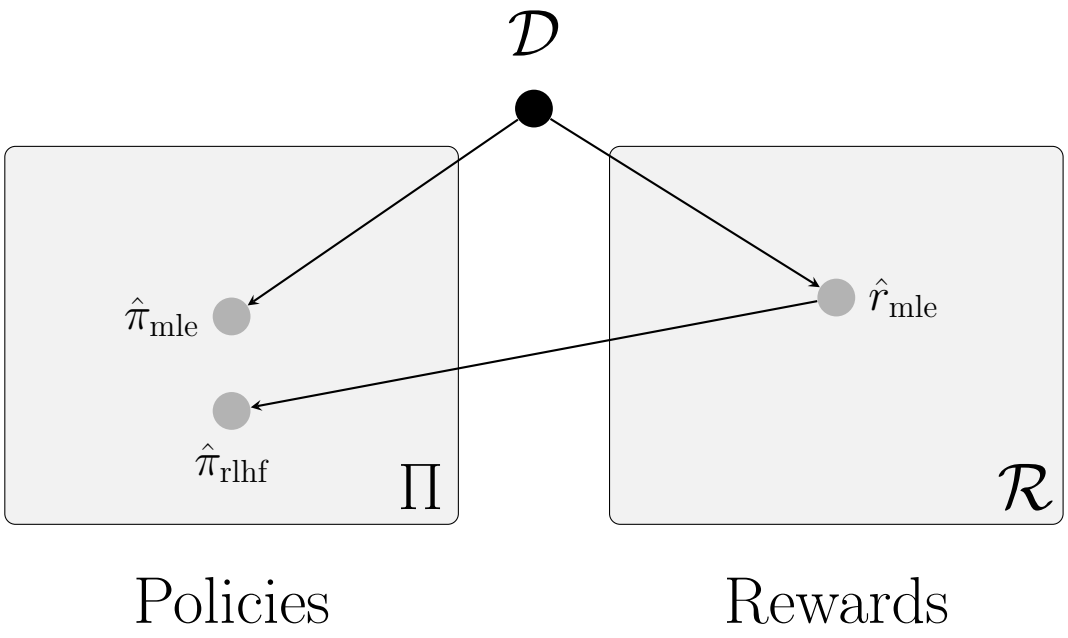

Figure 21.1: Geometric Interpretation of RLHF and DPO (Adapted from [8]).

The core idea of DPO is to re-express the reward function in terms of the optimal policy and the reference policy. Recall the form of the soft optimal policy $\pi^\star$ for a given reward r:

$$\mathbb{P}_r^\star(\xi|s_0) = \frac{1}{Z(s_0)}\mathbb{P}_{\text{ref}}(\xi|s_0)\exp\left(\frac{1}{\beta}r(\xi)\right) \tag{21.10}$$

We can take a log on both sides and and rearrange to solve for $r(\xi)$:

$$\log \mathbb{P}_r^\star(\xi|s_0) = \log \mathbb{P}_{\text{ref}}(\xi|s_0) + \frac{1}{\beta}r(\xi) - \log Z(s_0) \tag{21.11}$$

$$r(\xi) = \beta\left(\log \frac{\mathbb{P}_r^\star(\xi|s_0)}{\mathbb{P}_{\text{ref}}(\xi|s_0)}\right) + \beta \log Z(s_0) \tag{21.12}$$

Since $\mathbb{P}_r^\star(\xi|s_0) = \prod_h \pi^\star(a_h|s_{h-1})$ and $\mathbb{P}_{\text{ref}}(\xi|s_0) = \prod_h \pi_{\text{ref}}(a_h|s_{h-1})$ due to the deterministic dynamics, we can expand to write terms using token-level probabilities:

$$r(\xi) = \beta\sum_{h=1}^{H}(\log \pi^\star(a_h|s_{h-1}) - \log \pi_{\text{ref}}(a_h|s_{h-1})) + \beta \log Z(s_0) \triangleq r_\pi(\xi). \tag{21.13}$$

Thus, we can express the reward function that makes a policy soft optimal in terms of said policy by inverting the MaxEnt RL equations. So, while soft value iteration lets us go from r to $\pi^\star$, DPO tells us that logistic regression lets us go in the reverse direction.

Remark 5 (Your Language Model is Secretly a Reward Model) *One can show that the optimal policy for the entropy-regularized RL objective using this implicit reward r_π is precisely the policy π itself.*

$$\mathbb{P}^\star_{r_\pi}(\xi) \propto \exp(r_\pi(\xi)) \tag{21.14}$$

$$\propto \exp\left(\sum_h^H \log \pi(a_h|s_h) + \log Z(s_0)\right) \tag{21.15}$$

$$\propto \exp\left(\sum_h^H \log \pi(a_h|s_h)\right) \tag{21.16}$$

$$\propto \prod_h^H \pi(a_h|s_h). \tag{21.17}$$

Thus, if we optimize over r_π, we get the corresponding soft-optimal policy π "for-free".

The DPO Objective: DPO substitutes this implicit reward function r_π directly into the Bradley-Terry likelihood objective used for reward modeling (i.e. Stage 1), cancelling out the partition function as both trajectories share the same prompt:

$$\begin{aligned}\hat{\pi}_{\text{dpo}} &= \operatorname*{argmax}_{\pi\in\Pi} \mathbb{E}_{(s_0,\xi^+,\xi^-)\sim\mathcal{D}}\left[\log \sigma(r_\pi(\xi^+) - r_\pi(\xi^-))\right] \\ &= \operatorname*{argmax}_{\pi\in\Pi} \mathbb{E}_{(s_0,\xi^+,\xi^-)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi(\xi^+|s_0)}{\pi_{\text{ref}}(\xi^+|s_0)} - \beta \log \frac{\pi(\xi^-|s_0)}{\pi_{\text{ref}}(\xi^-|s_0)}\right)\right]\end{aligned}$$

This results in a single-stage optimization problem where we directly maximize the likelihood of the preference data under the policy π without any RL / on-policy sampling.

Information Geometry View of DPO: DPO performs a forward KL projection (FKL) directly from the empirical preference distribution $\mathbb{P}_\mathcal{D}$ onto the policy space Π. It bypasses the intermediate reward model space $\mathcal{R}$.

- Standard RLHF: $\mathcal{D} \xrightarrow{\text{FKL (MLE)}} \mathcal{R} \xrightarrow{\text{RKL (MaxEnt RL)}} \Pi$
- DPO: $\mathcal{D} \xrightarrow{\text{FKL (MLE)}} \Pi$

Key Point: While both methods aim to align the policy with preferences, the resulting policies $\hat{\pi}_{\text{rlhf}}$ and $\hat{\pi}_{\text{dpo}}$ are not necessarily identical, as we discuss in subsequent lectures.

21.8 *

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub

Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[3] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[5] Junzi Zhang, Jongho Kim, Brendan O'Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce, 2020.

[6] Zhaolin Gao, Jonathan D. Chang, Wenhao Zhan, Owen Oertell, Gokul Swamy, Kianté Brantley, Thorsten Joachims, J. Andrew Bagnell, Jason D. Lee, and Wen Sun. Rebel: Reinforcement learning via regressing relative rewards, 2024.

[7] 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.

[8] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.