Refresh Your Understanding

Select all that are true

does learn does not

- RLHF and DPO both learn an explicit representation of a reward model from preference data
- Both are constrained to be at most as good as the best examples in the pairwise preference data
- DPO does not use a reference policy
- Not Sure

Refresh Your Understanding Solutions

Select all that are true

- RLHF and DPO both learn an explicit representation of a reward model from preference data
- Both are constrained to be at most as good as the best examples in the pairwise preference data
- DPO does not use a reference policy
- Not Sure

Human Feedback and Reinforcement Learning from Human Preferences

Human Input to Train RL Agents

- There are many ways for humans to help train RL agents
- This is relevant if we want RL agents that can match human performance and/or human values

Training a Robot Through Human and Environmental Feedback

Teachable robots: Understanding human teaching behavior to build more effective robot learners. AL Thomaz, C Breazeal. Artificial Intelligence 2008

Learning a Reward Model from Human Feedback While Training an Agent

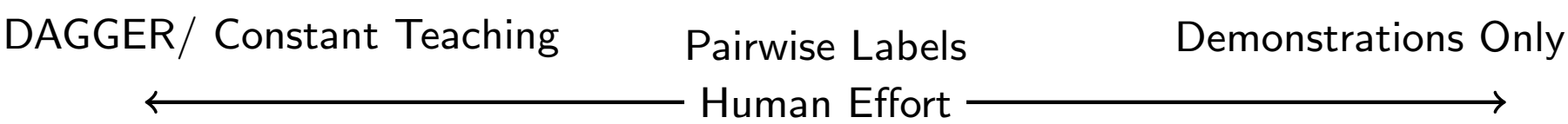
Table 1: Results of various Tetris agents.

Method	Mean Lines Cleared		Games
	at Game 3	at Peak	for Peak
TAMER	**65.89**	**65.89**	**3**
RRL-KBR [15]	5	50	120
Policy Iteration [2]	~ 0 (no learning until game 100)	3183	1500
Genetic Algorithm [5]	~ 0 (no learning until game 500)	586,103	3000
CE+RL [17]	~ 0 (no learning until game 100)	348,895	5000

Interactively shaping agents via human reinforcement: The TAMER framework. W Knox, P Stone. 2008. ICKC

Human Input for Training and Aligning RL Policies

Human Input for Training and Aligning RL Policies: Sweet Spot?



Comparing Recommendation Ranking Systems

Retrieval Function A

CS 159 Purdue University
web.ics.purdue.edu/~cs159/ ▾ Purdue University ▾
Aug 16, 2012 - **CS 159** introduces the tools of software development that have become essential for creative problem solving in Engineering. Educators and ...

CS159: Introduction to Parallel Processing | People | San Jo…
www.sjsu.edu › ... › Chun, Robert K › Courses ▾ San Jose State University ▾
Jan 20, 2015 - Description. A combination hardware architecture and software development class focused on multi-threaded, parallel processing algorithms ...

CS 159: Introduction to Parallel Processing - Info.sjsu.edu
info.sjsu.edu › ... › Courses ▾ San Jose State University ▾
CS 159. Introduction to Parallel Processing. Description Major parallel architectures: shared memory, distributed memory, SIMD, MIMD. Parallel algorithms: ...

Guy falls asleep in CS159 lab Purdue - YouTube


https://www.youtube.com/watch?v=vVciOgZwLag
Mar 24, 2011 - Uploaded by james brand
Guy falls asleep in our 7:30 am lab so we take his phone turn the volume up to full and call him.

CS 159: Advanced Topics in Machine Learning - Yisong Yue
www.yisongyue.com/courses/**cs159**/ ▾
CS 159: Advanced Topics in Machine Learning (Spring 2016). Course Description. This course will cover a mixture of the following topics: Online Learning ...

CS159: Introduction to Computational Complexity
cs.brown.edu/courses/**cs159**/home.html ▾ Brown University ▾
Home | Course Info | Assignments | Syllabus And Lectures | Staff and Hours | LaTeX. An early model of parallel computation... Home Courses.

Retrieval Function B

Guy falls asleep in CS159 lab Purdue - YouTube


https://www.youtube.com/watch?v=vVciOgZwLag
Mar 24, 2011 - Uploaded by james brand
Guy falls asleep in our 7:30 am lab so we take his phone turn the volume up to full and call him.

CS 159 Purdue University
web.ics.purdue.edu/~cs159/ ▾ Purdue University ▾
Aug 16, 2012 - **CS 159** introduces the tools of software development that have become essential for creative problem solving in Engineering. Educators and ...

CS159: Introduction to Parallel Processing | People | San Jo.
www.sjsu.edu › ... › Chun, Robert K › Courses ▾ San Jose State University ▾
Jan 20, 2015 - Description. A combination hardware architecture and software development class focused on multi-threaded, parallel processing algorithms ...

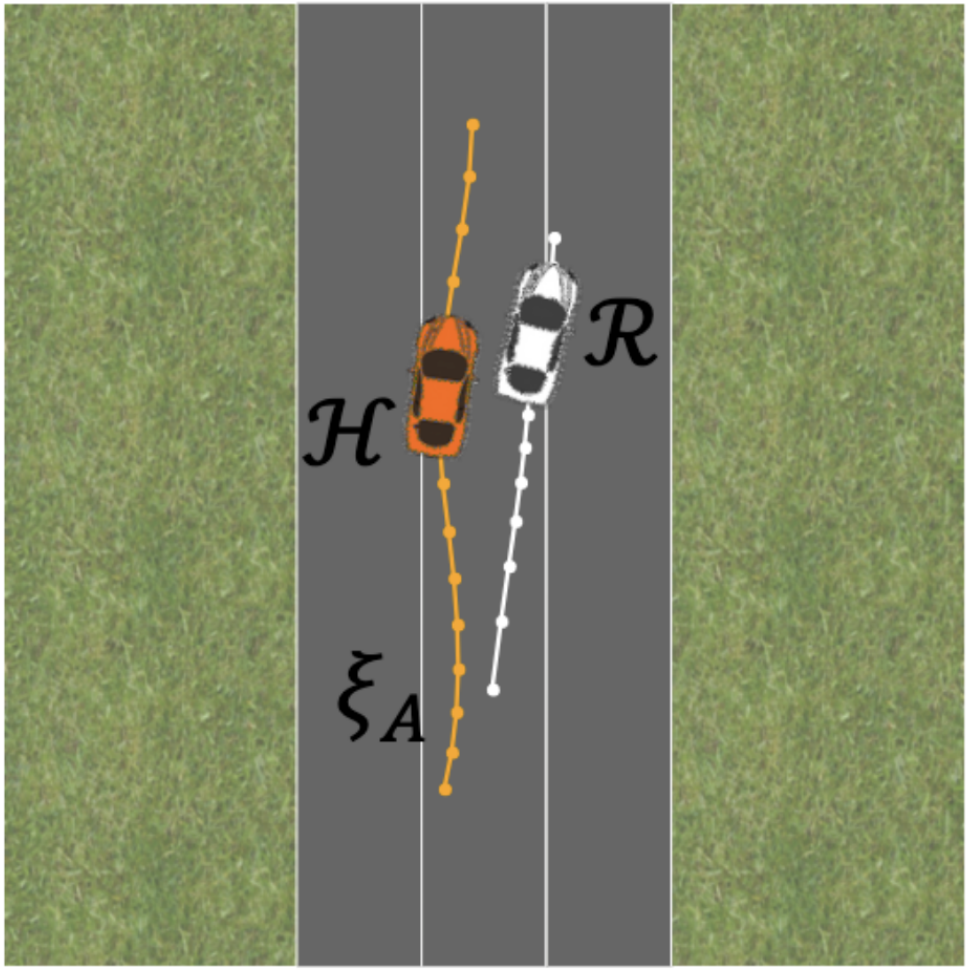
CS 159: Introduction to Parallel Processing - Info.sjsu.edu
info.sjsu.edu › ... › Courses ▾ San Jose State University ▾
CS 159. Introduction to Parallel Processing. Description Major parallel architectures: shared memory, distributed memory, SIMD, MIMD. Parallel algorithms: ...

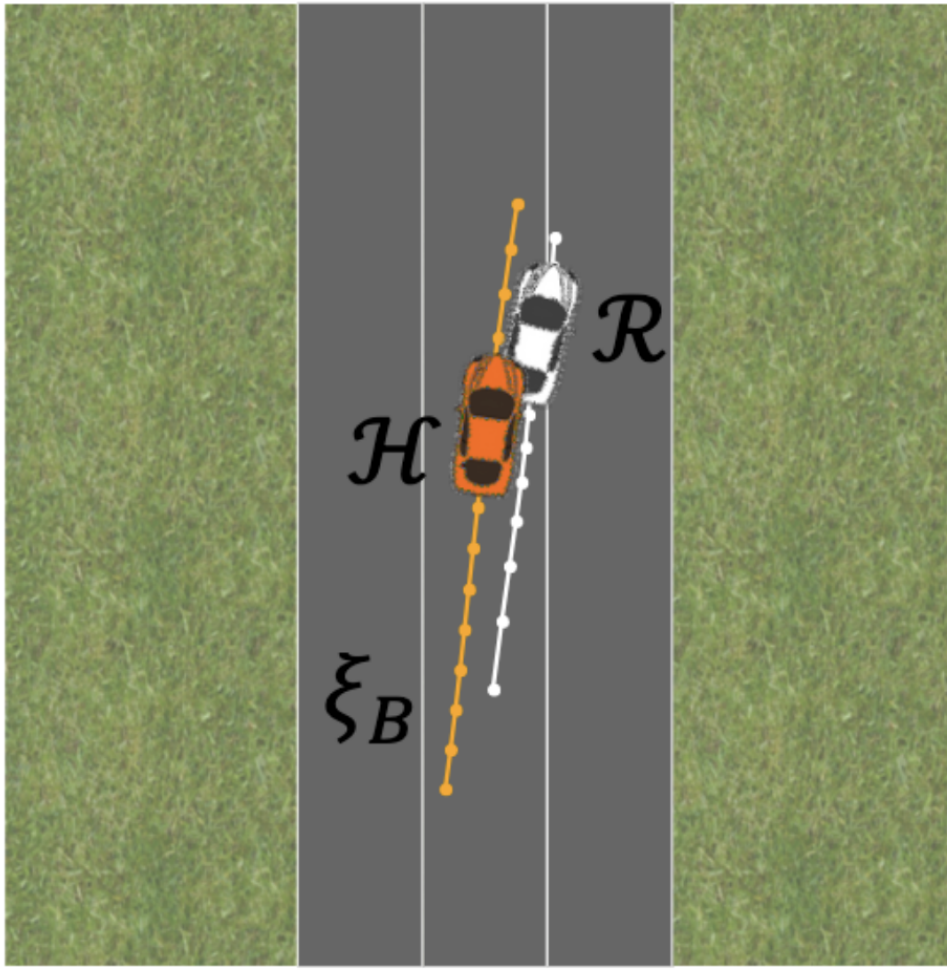
CS 159: Advanced Topics in Machine Learning - Yisong Yue
www.yisongyue.com/courses/**cs159**/ ▾
CS 159: Advanced Topics in Machine Learning (Spring 2016). Course Description. This course will cover a mixture of the following topics: Online Learning ...

CS159: Introduction to Computational Complexity
cs.brown.edu/courses/**cs159**/home.html ▾ Brown University ▾
Home | Course Info | Assignments | Syllabus And Lectures | Staff and Hours | LaTeX. A early model of parallel computation... Home Courses.

Slide from Yisong Yue
http://www.yisongyue.com/courses/cs159/lectures/dueling_bandits_lecture.pdf

Active Learning of Preferences for Human Robot Interaction





Active preference-based learning of reward functions. D Sadigh, AD Dragan, S Sastry, SA Seshia. RSS 2017

Pairwise Comparisons

- Often easier for people to make than hand writing a reward function
- Often easier than providing scalar reward (how much do you like this ad?)

Bradley-Terry Model (1952)

- Already saw with no other assumptions, the latent reward model is not unique
- Now focus on a particular structural model
- First consider simpler setting of k-armed bandits[1]: K actions $b_1, b_2, \ldots b_k$. No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

$$P(b_i \succ b_j) = \frac{\exp\left(r(b_i)\right)}{\exp\left(r(b_i)\right) + \exp\left(r(b_j)\right)} = p_{ij} \tag{9}$$

- Transitive: p_{ik} is determined from p_{ij} and p_{jk}

[1] We will see more on bandits later in the course
See: The K -armed dueling bandits problem. Y Yue, J Broder, R Kleinberg and T. Joachims. Journal of Computer and System Sciences. 2012.

Definitions

Condorcet Winner

An item b_i is a Condorcet winner if for every other item b_j, $P(b_i \succ b_j) > 0.5$.

Copeland Winner

An item b_i is a Copeland winner if it has the highest number of pairwise victories against all other items. The score for an item is calculated as the number of items it beats minus the number of items it loses to.

Borda Winner

An item b_i is a Borda winner if it maximizes the expected score, where the score against item b_j is 1 if $b_i \succ b_j$, $(P(b_i \succ b_j) > 0.5)$ 0.5 if $b_i = b_j$, and 0 if $b_i \prec B_j$.

- Historically algorithms for k-armed or dueling (k=2) bandits focused on finding a copeland winner.

Preference learning

Fitting the Parameters of a Bradley-Terry Model

- First consider k-armed bandits[2]: K actions $b_1, b_2, \ldots b_k$. No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

$$P(b_i \succ b_j) = \frac{\exp\left(r(b_i)\right)}{\exp\left(r(b_i)\right) + \exp\left(r(b_j)\right)} = p_{ij} \tag{10}$$

[2]We will see more on bandits later in the course

Fitting the Parameters of a Bradley-Terry Model

- First consider k-armed bandits[3]: K actions $b_1, b_2, \ldots b_k$. No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

$$P(b_i \succ b_j) = \frac{\exp(r(b_i))}{\exp(r(b_i)) + \exp(r(b_j))} = p_{ij} \tag{11}$$

- Assume have N tuples of form (b_i, b_j, μ) where $\mu(1) = 1$ if the human marked $b_i \succ b_j$, $\mu(1) = 0.5$ if the human marked $b_i = b_j$, else 0 if $b_j \succ b_i$
- Maximize likelihood with cross entropy

$$loss = -\sum_{(b_i, b_j, \mu) \in \mathcal{D}} \mu(1) \log P(b_i \succ b_j) + \mu(2) \log P(b_j \succ b_j) \tag{12}$$

[3] We will see more on bandits later in the course

Preference to Reward Modeling for RL

- Can also do this for trajectories
- Consider two trajectories, $\tau^1(s_0, a_7, s_{14}, \ldots)$ and $\tau^2(s_0, a_6, s_{12}, \ldots)$
- Let $R^1 = \sum_{i=0}^{T-1} r_i^1$ be the (latent, unobserved) sum of rewards for trajectory τ^1 and similarly for R^2.
- Define the probability that a human prefers $\tau^1 \succ \tau^2$ as

$$\hat{P}\left[\tau^1 \succ \tau^2\right] = \frac{\exp \sum_{i=0}^{t-1} r_i^1}{\exp \sum_{i=0}^{t-1} r_i^1 + \exp \sum_{i=0}^{t-1} r_i^2}, \tag{13}$$

Preference to Reward Modeling for RL

- Can also do this for trajectories
- Consider two trajectories, $\tau^1(s_0, a_7, s_{14}, \ldots)$ and $\tau^2(s_0, a_6, s_{12}, \ldots)$
- Let $R^1 = \sum_{i=0}^{T-1} r_i^1$ be the (latent, unobserved) sum of rewards for trajectory τ^1 and similarly for R^2.
- Define the probability that a human prefers $\tau^1 \succ \tau^2$ as

$$\hat{P}\left[\tau^1 \succ \tau^2\right] = \frac{\exp \sum_{i=0}^{t-1} r_i^1}{\exp \sum_{i=0}^{t-1} r_i^1 + \exp \sum_{i=0}^{t-1} r_i^2}, \tag{14}$$

- Use learned reward model, and do PPO with this model

Reinforcement Learning from Human Feedback

- Learning to backflip
- "needed 900 bits of feedback from a human evaluator to learn to backflip"
- https://player.vimeo.com/video/754042470?h=e64a40690d&badge=0&autopause=0&player_id=0&app_id=58479

Christiano et al. 2017. Deep RL from Human Preferences https://arxiv.org/pdf/1706.03741.pdf

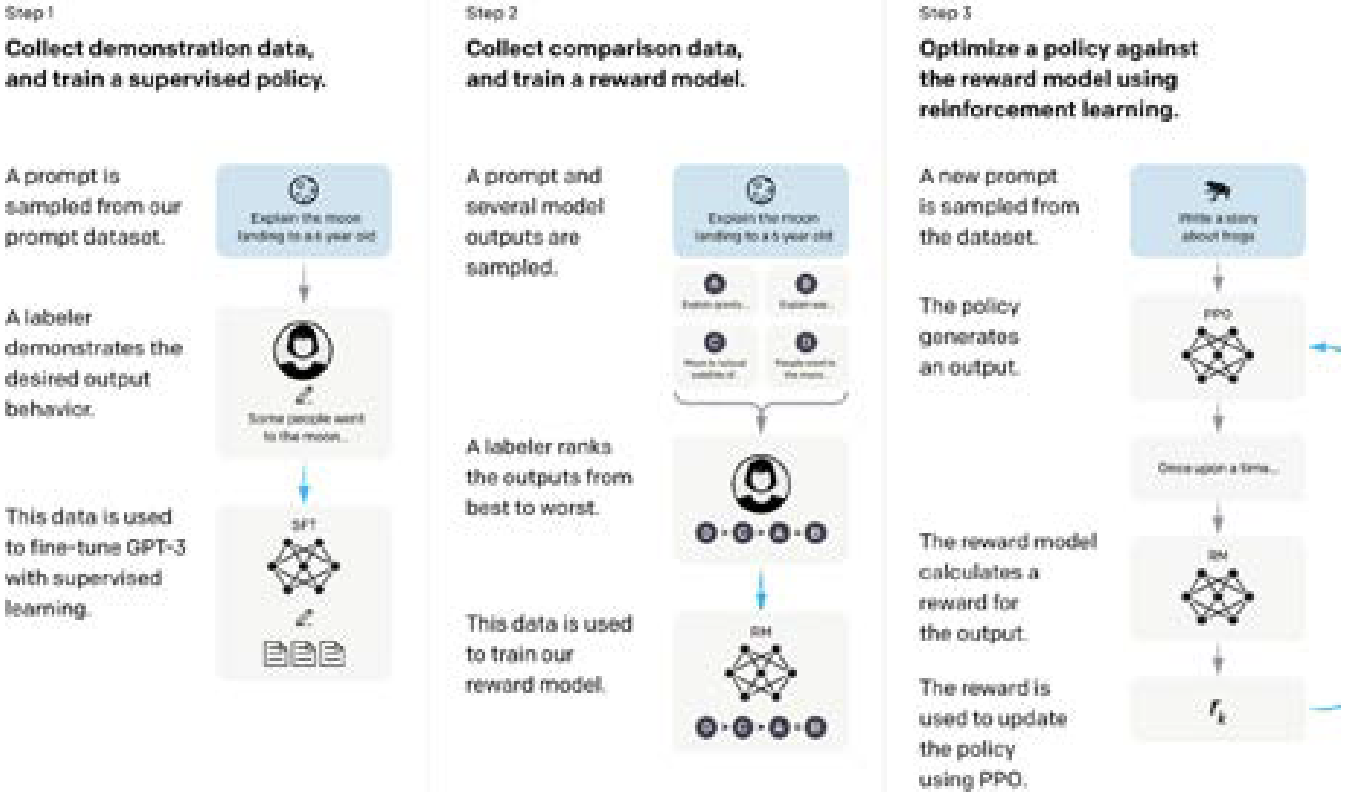
From Backflips to ChatGPT. [4]

[4] Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

From Backflips to ChatGPT

- Next set of slides are from part of Tatsu Hashimoto's Lecture 11 in CS224N

High-level instantiation: ‘RLHF’ pipeline



- First step: instruction tuning!
- Second + third steps: maximize reward (but how??)

Lecture 9: RLHF and Guest Lecture on DPO

Emma Brunskill

CS234 Reinforcement Learning.

Refresh Your Understanding L9N1

Select all that are true

(a) The Bradley Terry model expresses the probability that someone will select option b_i over b_j

(b) Using preference tuples and the Bradley Terry model, one can learn a model of the reward function

(c) The resulting reward function can be shifted by any constant and will not change the resulting preferences

(d) The resulting reward function can be multiplied by any constant and will not change the resulting preferences

(e) In RLHF we update the reward model after each PPO roll out

(f) Not sure

Refresh Your Understanding L9N1 Solutions

Select all that are true

(a) The Bradley Terry model expresses the probability that someone will select option b_i over b_j

(b) Using preference tuples and the Bradley Terry model, one can learn a model of the reward function

(c) The resulting reward function can be shifted by any constant and will not change the resulting preferences

(d) The resulting reward function can be multiplied by any constant and will not change the resulting preferences

(e) In RLHF we update the reward model after each PPO roll out

(f) Not sure

1,2,3 are true. 4 is false: for example, we cannot multiply the rewards by -1 and preserve the ordering.

Class Structure

- Last time: Imitation Learning (Max Entropy IRL) and RLHF
- This time: RLHF and Direct Preference Optimization (best paper runner up at top ML conference) guest lecture
-

Today

- RLHF for LLM
- Direct Preference Optimization

Pairwise Comparisons

- Often easier for people to make than hand writing a reward function
- Often easier than providing scalar reward (how much do you like this ad?)

Fitting the Parameters of a Bradley-Terry Model

- Consider k-armed bandits[1]: K actions $b_1, b_2, \ldots b_k$. No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

$$P(b_i \succ b_j) = \frac{\exp(r(b_i))}{\exp(r(b_i)) + \exp(r(b_j))} = p_{ij} \tag{1}$$

- Assume have N tuples of form (b_i, b_j, μ) where $\mu(1) = 1$ if the human marked $b_i \succ b_j$, $\mu(1) = 0.5$ if the human marked $b_i = b_j$, else 0 if $b_j \succ b_i$
- Maximize likelihood with cross entropy

$$loss = -\sum_{(b_i, b_j, \mu) \in \mathcal{D}} \mu(1) \log P(b_i \succ b_j) + \mu(2) \log P(b_j \succ b_j) \tag{2}$$

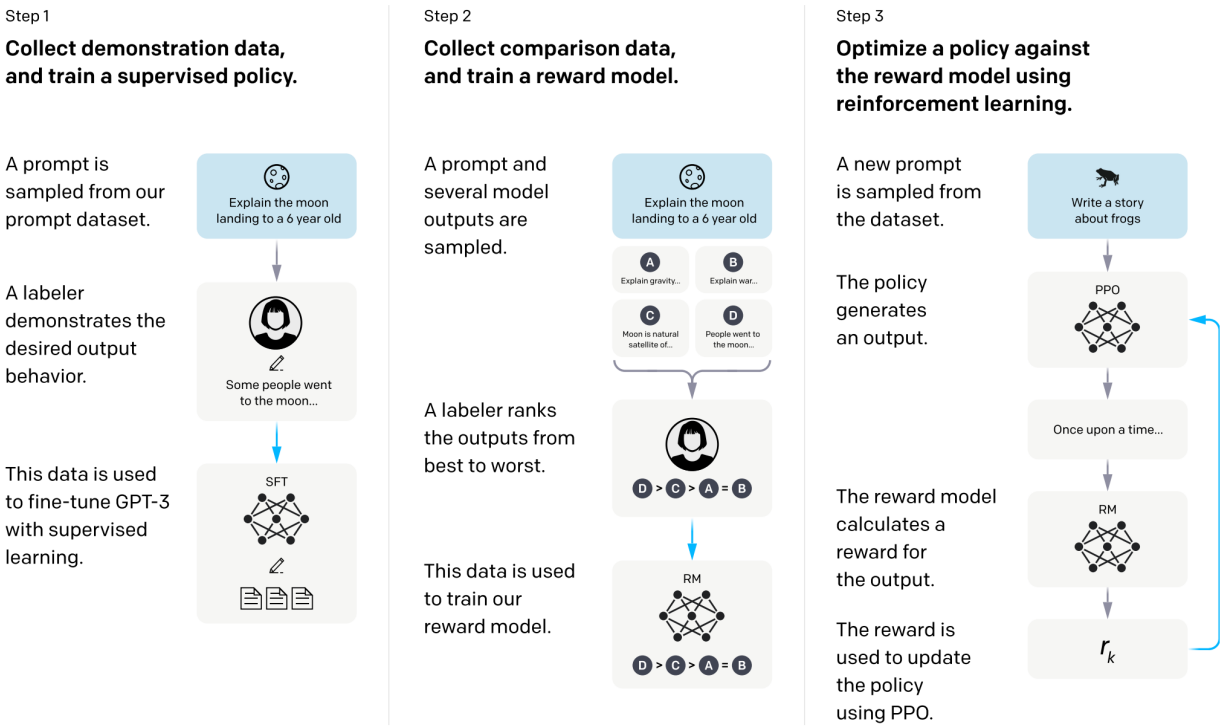
- Use learned reward model, and do PPO with this model
- See prior lecture for notes on doing this over trajectories

[1] We will see more on bandits later in the course

From Backflips to ChatGPT

- How is this used in ChatGPT?
- Next set of slides are from part of Tatsu Hashimoto's Lecture 11 in CS224N

High-level instantiation: ‘RLHF’ pipeline

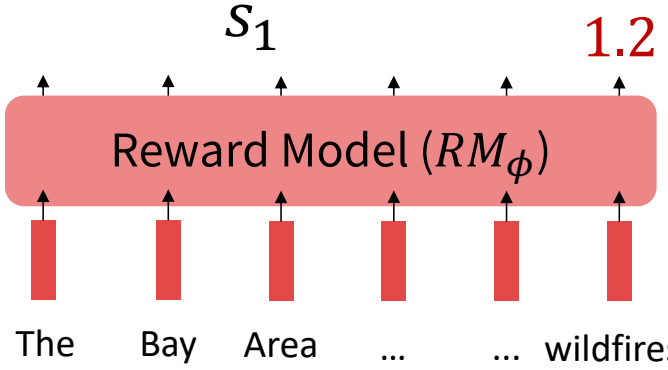


- First step: instruction tuning!
- Second + third steps: maximize reward (but how??)

How do we model human preferences?

- **Problem 2:** human judgments are noisy and miscalibrated!
- **Solution:** instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable [Phelps et al., 2015; Clark et al., 2018]

`An earthquake hit San Francisco. There was minor property damage, but no injuries.` > `A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.` > `The Bay Area has good weather but is prone to earthquakes and wildfires.`



s_3 s_2

Bradley-Terry [1952] paired comparison model

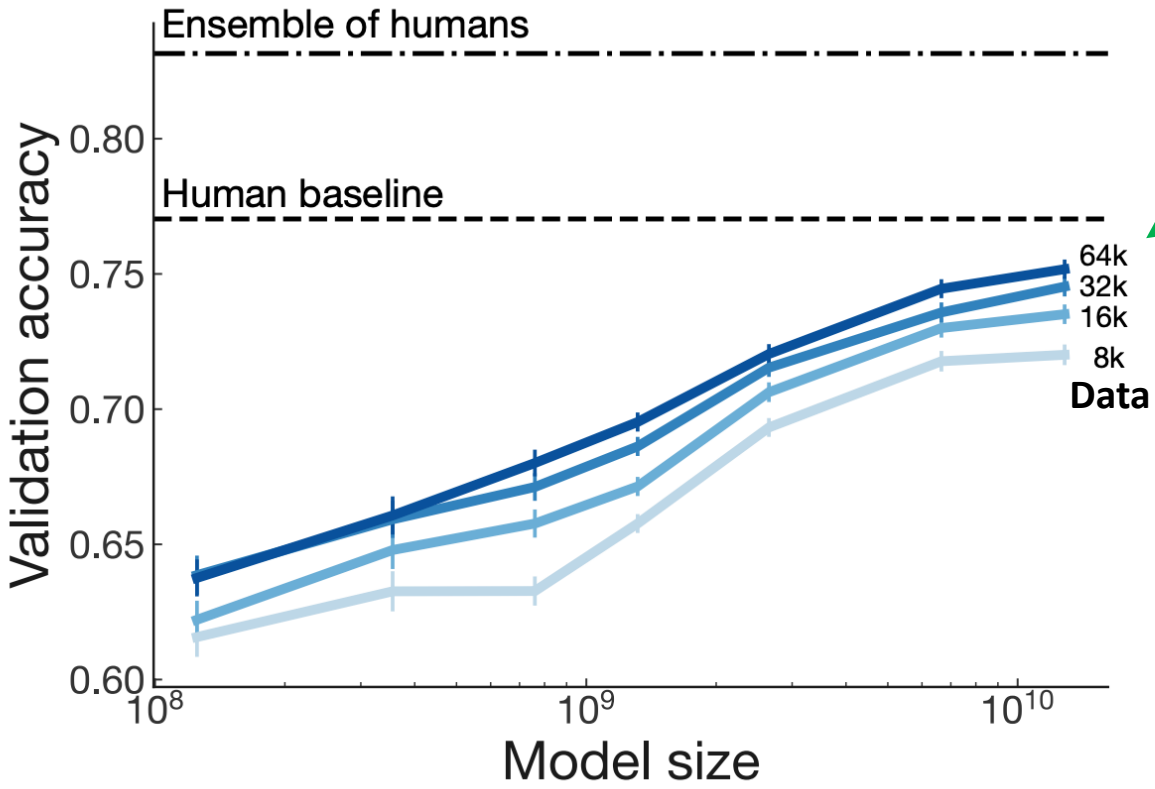
$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

"winning" sample — "losing" sample

s^w should score higher than s^l

Make sure your reward model works first!

Evaluate RM on predicting outcome of held-out human judgments



Large enough RM trained on enough data approaching single human perf

[Stiennon et al., 2020]

RLHF: Putting it all together [Christiano et al., 2017; Stiennon et al., 2020]

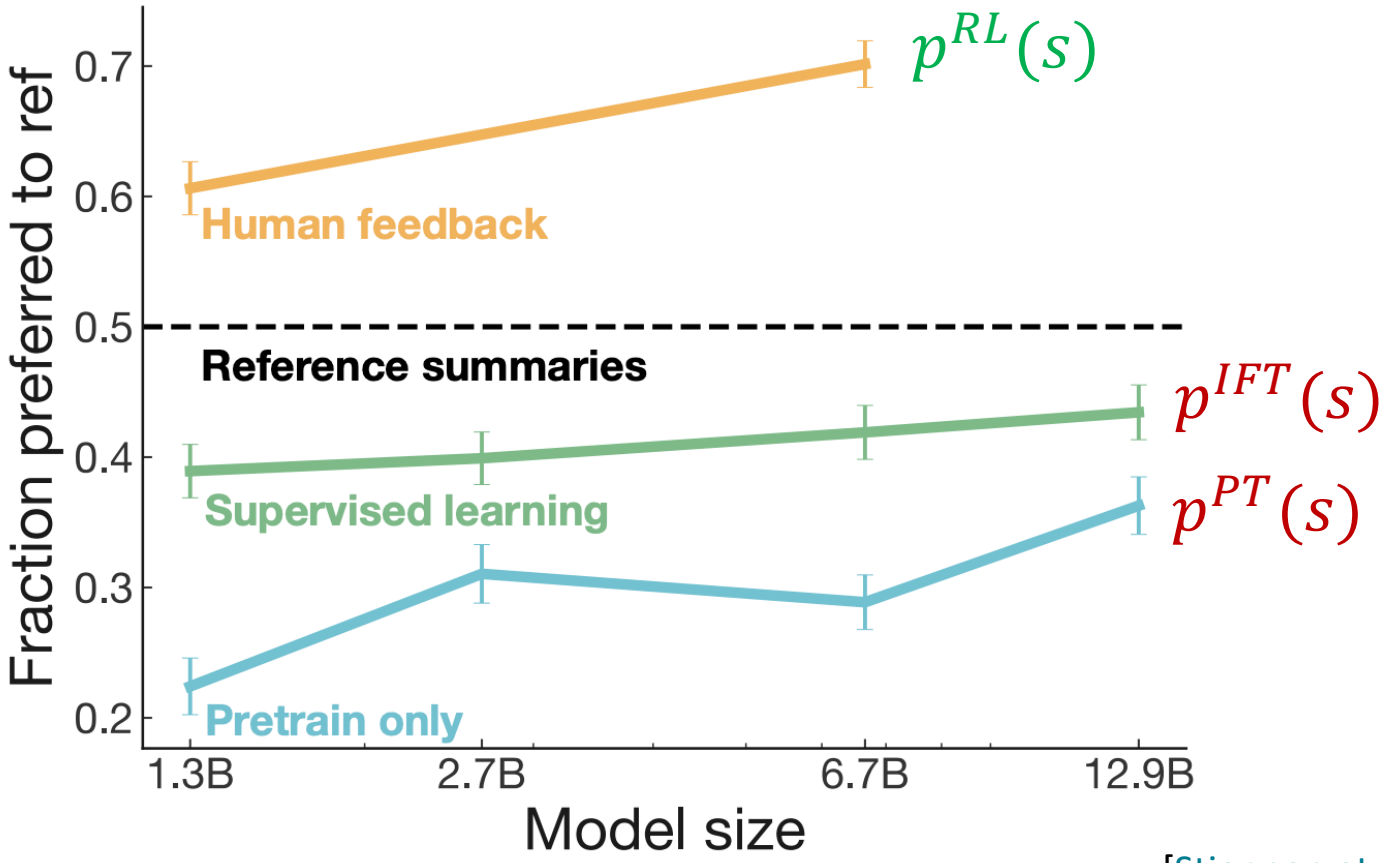
- Finally, we have everything we need:
  - A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
  - A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
  - A method for optimizing LM parameters towards an arbitrary reward function.
- Now to do RLHF:
  - Initialize a copy of the model $p_\theta^{RL}(s)$, with parameters θ we would like to optimize
  - Optimize the following reward with RL:

$$R(s) = RM_\phi(s) - \underbrace{\beta \log \left(\frac{p_\theta^{RL}(s)}{p^{PT}(s)} \right)}$$

Pay a price when $p_\theta^{RL}(s) > p^{PT}(s)$

This is a penalty which prevents us from diverging too far from the pretrained model. In expectation, it is known as the **Kullback-Leibler** (**KL**) divergence between $p_\theta^{RL}(s)$ and $p^{PT}(s)$.

RLHF provides gains over pretraining + finetuning



[Stiennon et al., 2020]

InstructGPT: scaling up RLHF to tens of thousands of tasks

30k tasks!

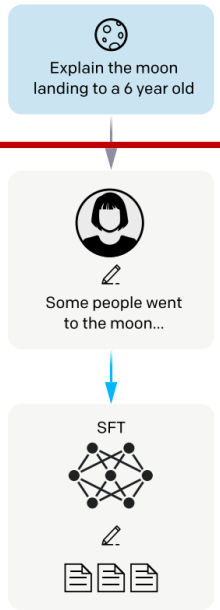
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



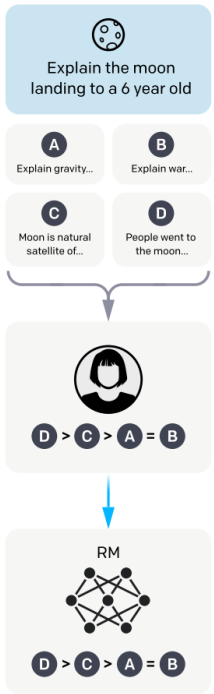
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

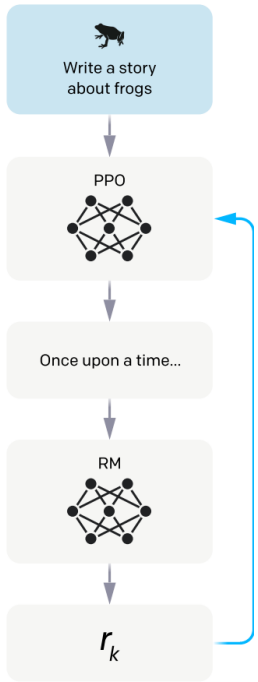
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



[Ouyang et al., 2022]

Controlled comparisons of “RLHF” style algorithms

Method	Simulated win-rate (%)	Human win-rate (%)
GPT-4	79.0 ± 1.4	69.8 ± 1.6
ChatGPT	61.4 ± 1.7	52.9 ± 1.7
PPO	46.8 ± 1.8	55.1 ± 1.7
Best-of-n	45.0 ± 1.7	50.7 ± 1.8
Expert Iteration	41.9 ± 1.7	45.7 ± 1.7
SFT 52k (Alpaca 7B)	39.2 ± 1.7	40.7 ± 1.7
SFT 10k	36.7 ± 1.7	44.3 ± 1.7
Binary FeedME	36.6 ± 1.7	37.9 ± 1.7
Quark	35.6 ± 1.7	-
Binary Reward Conditioning	32.4 ± 1.6	-
Davinci001	24.4 ± 1.5	32.5 ± 1.6
LLaMA 7B	11.3 ± 1.1	6.5 ± 0.9

- Many works study RLHF behaviors using GPT-4 feedback (**Simulated**) as a surrogate for **Human** feedback.
- PPO (method in InstructGPT) does work
- Simple baselines (Best-of-n, Training on ‘good’ outputs) works well too

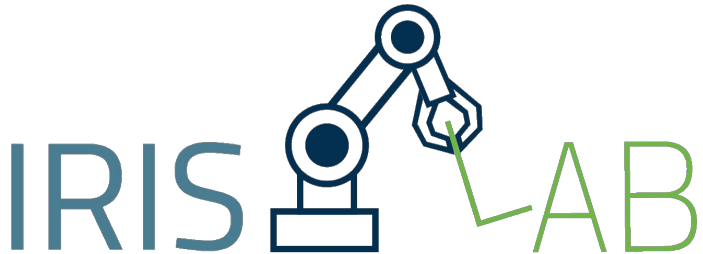
[Dubois et al 2023]

Today

- RLHF for LLM
- **Direct Preference Optimization**

Direct Preference Optimization: A New RLHF Approach

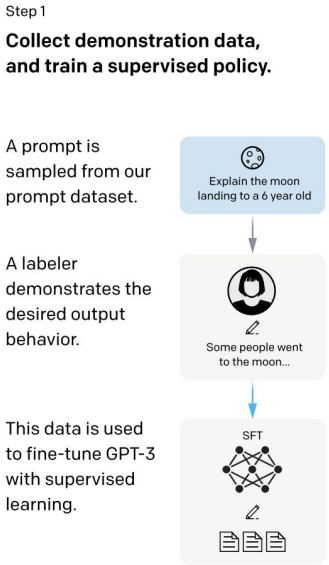
Rafael Rafailov Archit Sharma Eric Mitchell



Stanford University

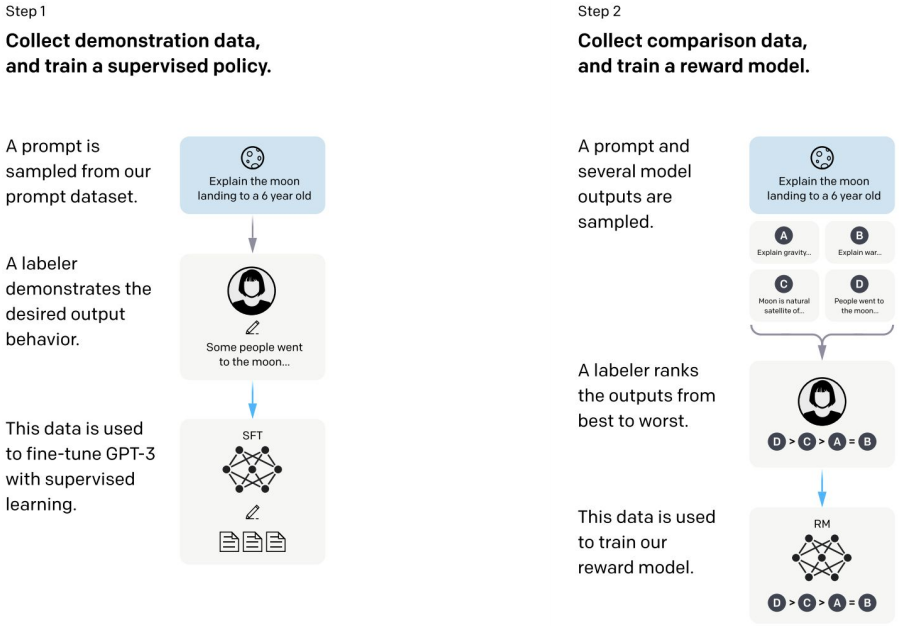
RLHF: Reinforcement Learning From Human Feedback

RLHF: Reinforcement Learning From Human Feedback



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Reinforcement Learning From Human Feedback



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Reinforcement Learning From Human Feedback

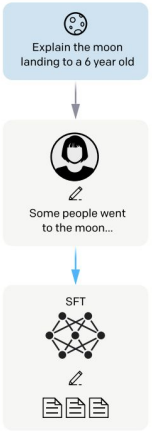
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



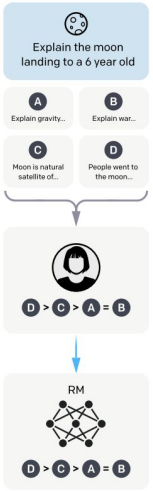
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

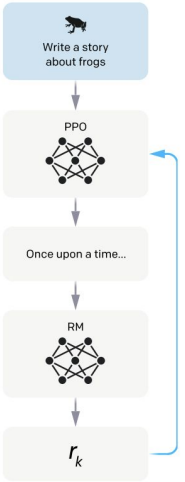
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Learning a **reward model** from human feedback

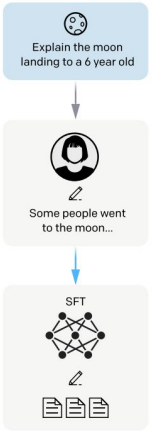
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



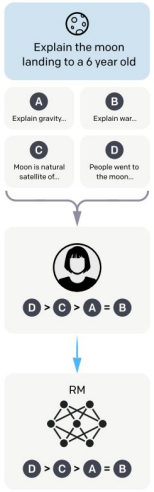
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

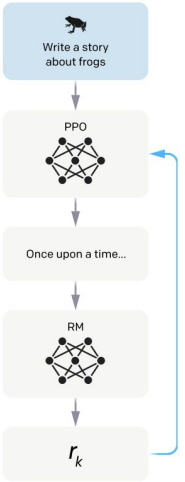
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Learning a **reward model** from human feedback

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt

Preferred response

Dispreferred response

Bradley-Terry Model connects rewards to preferences:

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt

Preferred response

Dispreferred response

Bradley-Terry Model connects rewards to preferences:

$$p(y_w \succ y_l \mid x) = \sigma(r(x, y_w) - r(x, y_l))$$

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt

Preferred response

Dispreferred response

Bradley-Terry Model connects rewards to preferences:

Reward assigned to **preferred** and **dispreferred** responses

$$p(y_w \succ y_l \mid x) = \sigma(r(x, y_w) - r(x, y_l))$$

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt

Preferred response

Dispreferred response

Bradley-Terry Model connects rewards to preferences:

Reward assigned to **preferred** and **dispreferred** responses

$$p(y_w \succ y_l \mid x) = \sigma(r(x, y_w) - r(x, y_l))$$

Train the reward model by **minimizing negative log likelihood:**

RLHF: Learning a **reward model** from human feedback

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt

Preferred response

Dispreferred response

Bradley-Terry Model connects rewards to preferences:

Reward assigned to **preferred** and **dispreferred** responses

$$p(y_w \succ y_l \mid x) = \sigma(r(x, y_w) - r(x, y_l))$$

Train the reward model by **minimizing negative log likelihood:**

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

RLHF: Reinforcement Learning From Human Feedback

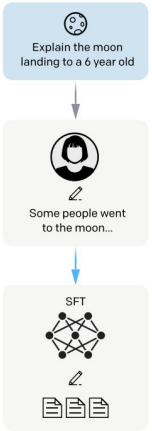
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



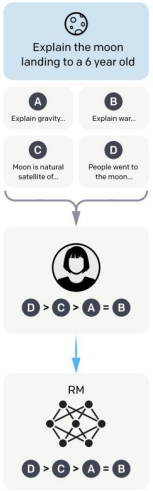
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

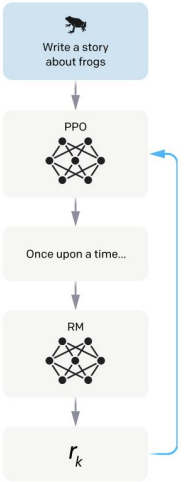
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Learning a **policy** that optimizes the **reward**

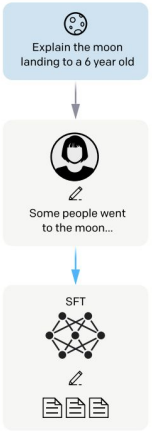
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



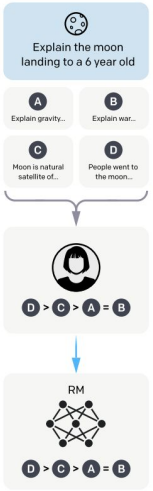
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

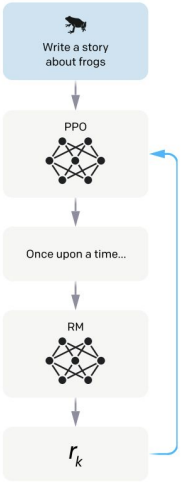
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Learning a **policy** that optimizes the **reward**

Now we have a **reward model** r_ϕ that represents* **goodness according to humans**

RLHF: Learning a **policy** that optimizes the **reward**

Now we have a **reward model** r_ϕ that represents* **goodness according to humans**

Now, learn a policy π_θ achieving **high reward**

RLHF: Learning a **policy** that optimizes the **reward**

Now we have a **reward model** r_ϕ that represents* **goodness according to humans**

Now, learn a policy π_θ achieving **high reward**

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r_\phi(x, y) \right]$$

RLHF: Learning a **policy** that optimizes the **reward**

Now we have a **reward model** r_ϕ that represents* **goodness according to humans**

Now, learn a policy π_θ achieving **high reward**

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r_\phi(x, y) \right]$$

Sample from policy

Want high reward…

RLHF: Learning a **policy** that optimizes the **reward**

Now we have a **reward model** r_ϕ that represents* **goodness according to humans**

Now, learn a policy π_θ achieving **high reward** while **staying close** to original model π_{ref}

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r_\phi(x, y) \right]$$

Sample from policy

Want high reward…

RLHF: Learning a **policy** that optimizes the **reward**

Now we have a **reward model** r_ϕ that represents* **goodness according to humans**

Now, learn a policy π_θ achieving **high reward** while **staying close** to original model π_{ref}

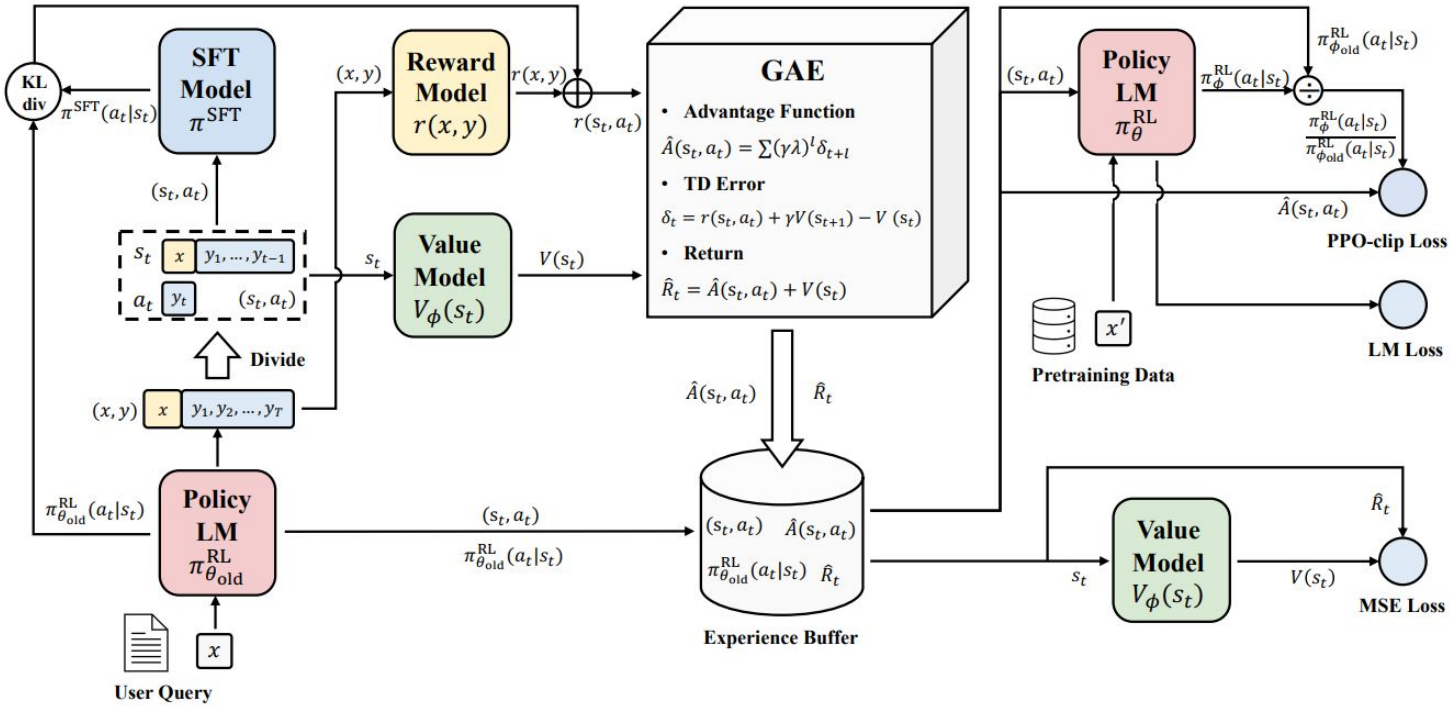
$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r_\phi(x,y) \right] - \beta \mathbb{D}_{\text{KL}} \left[\pi_\theta(y|x) || \pi_{\text{ref}}(y|x) \right]$$

Sample from policy

Want high reward…

…but keep KL to original model small!

RLHF: Learning a **policy** that optimizes the **reward**



Secrets of RLHF in Large Language Models Part I: PPO, Zheng et.al. 2023

Direct Preference Optimization

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Direct Preference Optimization

any reward function

RLHF Objective

(get **high reward**, stay **close** to reference model)

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$

Note **intractable sum** over possible responses; can't immediately use this

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$

Note **intractable sum** over possible responses; can't immediately use this

Rearrange

(write **any reward function** as function of **optimal policy**)

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi(y|x)} \left[r(x,y)\right] - \beta \mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x) \| \pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x,y)\right)$

Note **intractable sum** over possible responses; can't immediately use this

Rearrange

(write **any reward function** as function of **optimal policy**)

$$r(x,y) = \underbrace{\beta \log \frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta \log Z(x)}_{\text{some parameterization of a reward function}}$$

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi(y|x)} \left[r(x,y)\right] - \beta \mathbb{D}_{\mathrm{KL}}\left(\pi(\cdot \mid x) \| \pi_{\mathrm{ref}}(\cdot \mid x)\right)$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x,y)\right)$

Note **intractable sum** over possible responses; can't immediately use this

Rearrange

(write **any reward function** as function of **optimal policy**)

Ratio is **positive** if policy likes response more than reference model, **negative** if policy likes response less than ref. model

$$r(x,y) = \underbrace{\beta \log \frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta \log Z(x)}_{\text{some parameterization of a reward function}}$$

Direct Preference Optimization: Putting it together

Direct Preference Optimization: Putting it together

A loss function on
reward functions

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

A loss function on policies

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

A loss function on policies

Derived from the Bradley-Terry model of human preferences:

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

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

A loss function on policies

Derived from the Bradley-Terry model of human preferences:

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

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

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

A loss function on policies

Derived from the Bradley-Terry model of human preferences:

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

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

Reward of **preferred** response

Reward of **dispreferred** response

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

Closed-Form Optimal Policy

$$\max_{\pi} E_{x\sim D,\, y\sim\pi}\left[r(x,y) - \beta D_{KL}\left(\pi(y|x)\,\|\,\pi_{ref}(y|x)\right)\right]$$

$$= \text{''} \quad \left[r(x,y) - \beta \log \pi(y|x)/\pi_{ref}(y|x)\right]$$

$$= \text{''} \quad -\frac{1}{\beta}\left[\log \pi(y|x)/\pi_{ref}(y|x) - \frac{1}{\beta} r(x,y)\right]$$

$$= \min_{\pi} E_{x\sim D,\, y\sim\pi}\left[\log \pi(y|x)/\pi_{ref}(y|x) - \frac{1}{\beta} r(x,y)\right]$$

$$= \text{''} \quad \left[\log \pi(y|x)/\pi_{ref}(y|x) - \log \exp \frac{1}{\beta} r(x,y)\right]$$

$$= \text{''} \quad \left[\log \frac{\pi(y|x)}{\pi_{ref}(y|x)\exp\left(\frac{1}{\beta} r(x,y)\right)}\right]$$

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

$$= \text{''} \quad \left[\log\left[\quad \text{''} \quad \frac{Z(x)}{Z(x)}\right]\right]$$

$$= \min_{\pi} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{ref}(y|x)\exp\left(\frac{1}{\beta} r(x,y)\right)} - \log Z(x)\right]$$

[20] Slides with Red Text Font from Rafailov, Sharma, Mitchell 2024 guest lecture[21]

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

A loss function on policies

Derived from the Bradley-Terry model of human preferences:

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

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

When substituting, the **log Z term cancels**, because the loss only cares about **difference** in rewards

Reward of **preferred** response

Reward of **dispreferred** response

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

Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

Derived from the Bradley-Terry model of human preferences:

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

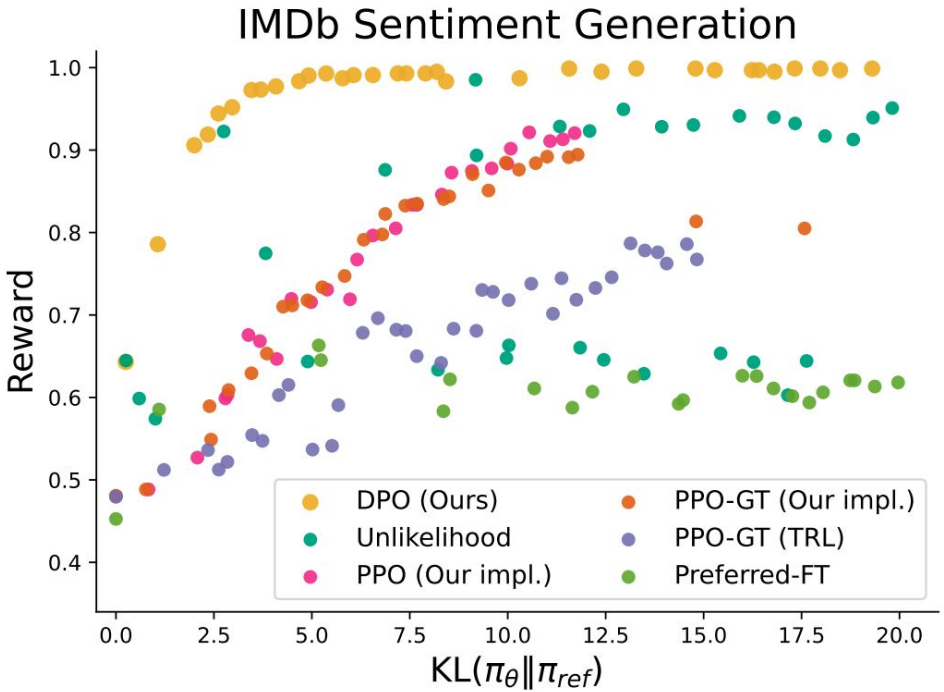
$$r_{\pi_\theta}(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

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

Reward of **preferred** response Reward of **dispreferred** response

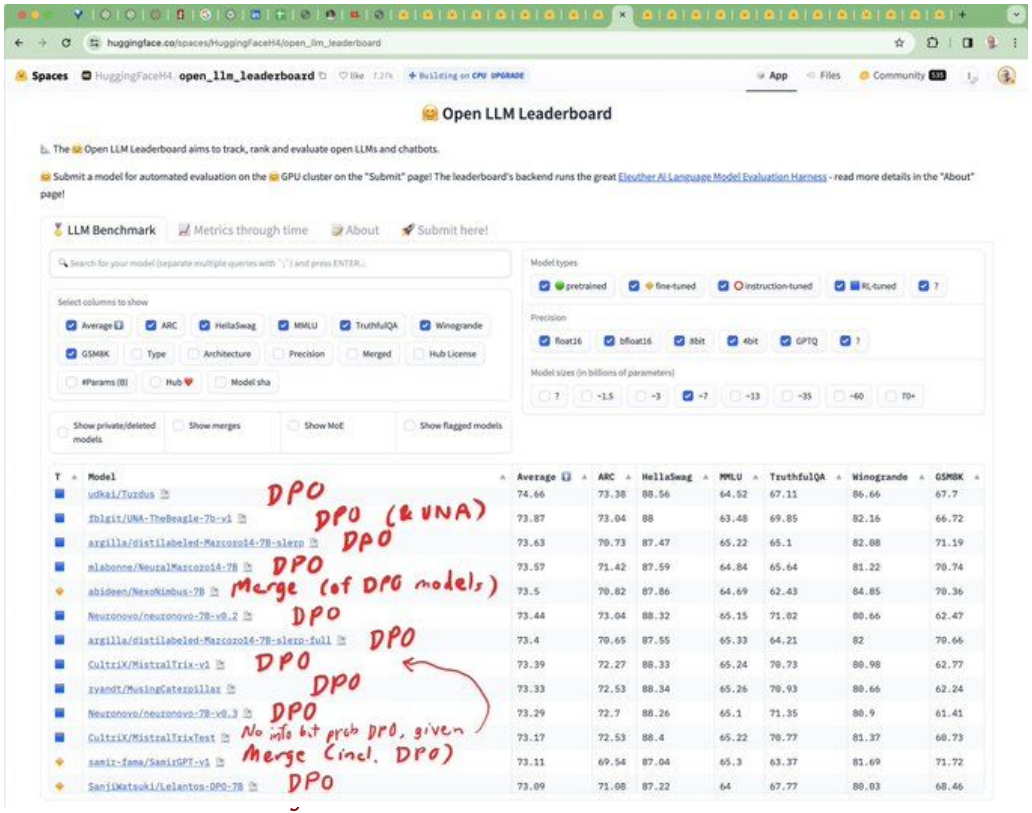
Results

How Efficiently does DPO Trade off Reward & KL?



1. Generate positive IMDB reviews from GPT2-XL
2. Use pre-trained sentiment classifier as Gold RM
3. Create preferences based on Gold RM
4. Optimize with PPO and DPO

Models Trained With DPO



Large-Scale DPO Training

Large-Scale DPO Training

Mistral

4 Instruction Fine-tuning

We train Mixtral – Instruct using supervised fine-tuning (SFT) on an instruction dataset followed by Direct Preference Optimization (DPO) [25] on a paired feedback dataset. Mixtral – Instruct reaches a score of 8.30 on MT-Bench [33] (see Table 2), making it the best open-weights model as of December 2023. Independent human evaluation conducted by LMSys is reported in Figure 6[3] and shows that Mixtral – Instruct outperforms GPT-3.5-Turbo, Gemini Pro, Claude-2.1, and Llama 2 70B chat.

Model	Arena Elo rating	MT-bench (score)	License
GPT-4-Turbo	1243	9.32	Proprietary
GPT-4-0314	1192	8.96	Proprietary
GPT-4-0613	1158	9.18	Proprietary
Claude-1	1149	7.9	Proprietary
Claude-2.0	1131	8.06	Proprietary
Mixtral-8x7b-Instruct-v0.1	1121	8.3	Apache 2.0
Claude-2.1	1117	8.18	Proprietary
GPT-3.5-Turbo-0613	1117	8.39	Proprietary
Gemini Pro	1111		Proprietary
Claude-Instant-1	1110	7.85	Proprietary
Tulu-2-DPO-70B	1110	7.89	AI2 ImpACT Low-risk
Yi-34B-Chat	1110		Yi License
GPT-3.5-Turbo-0314	1105	7.94	Proprietary
Llama-2-70b-chat	1077	6.86	Llama 2 Community

Figure 6: LMSys Leaderboard. (Screenshot from Dec 22, 2023) Mixtral 8x7B Instruct v0.1 achieves an Arena Elo rating of 1121 outperforming Claude-2.1 (1117), all versions of GPT-3.5-Turbo (1117 best), Gemini Pro (1111), and Llama-2-70b-chat (1077). Mixtral is currently the best open-weights model by a large margin.

Large-Scale DPO Training

Mistral

4 Instruction Fine-tuning

We train Mixtral – Instruct using supervised fine-tuning (SFT) on an instruction dataset followed by Direct Preference Optimization (DPO) [25] on a paired feedback dataset. Mixtral – Instruct reaches a score of 8.30 on MT-Bench [33] (see Table 2), making it the best open-weights model as of December 2023. Independent human evaluation conducted by LMSys is reported in Figure 6[3] and shows that Mixtral – Instruct outperforms GPT-3.5-Turbo, Gemini Pro, Claude-2.1, and Llama 2 70B chat.

Model	Arena Elo rating	MT-bench (score)	License
GPT-4-Turbo	1243	9.32	Proprietary
GPT-4-0314	1192	8.96	Proprietary
GPT-4-0613	1158	9.18	Proprietary
Claude-1	1149	7.9	Proprietary
Claude-2.0	1131	8.06	Proprietary
Mixtral-8x7b-Instruct-v0.1	1121	8.3	Apache 2.0
Claude-2.1	1117	8.18	Proprietary
GPT-3.5-Turbo-0613	1117	8.39	Proprietary
Gemini Pro	1111		Proprietary
Claude-Instant-1	1110	7.85	Proprietary
Tulu-2-DPO-70B	1110	7.89	AI2 ImpACT Low-risk
Yi-34B-Chat	1110		Yi License
GPT-3.5-Turbo-0314	1105	7.94	Proprietary
Llama-2-70b-chat	1077	6.86	Llama 2 Community

Figure 6: LMSys Leaderboard. (Screenshot from Dec 22, 2023) Mixtral 8x7B Instruct v0.1 achieves an Arena Elo rating of 1121 outperforming Claude-2.1 (1117), all versions of GPT-3.5-Turbo (1117 best), Gemini Pro (1111), and Llama-2-70b-chat (1077). Mixtral is currently the best open-weights model by a large margin.

LLaMa3

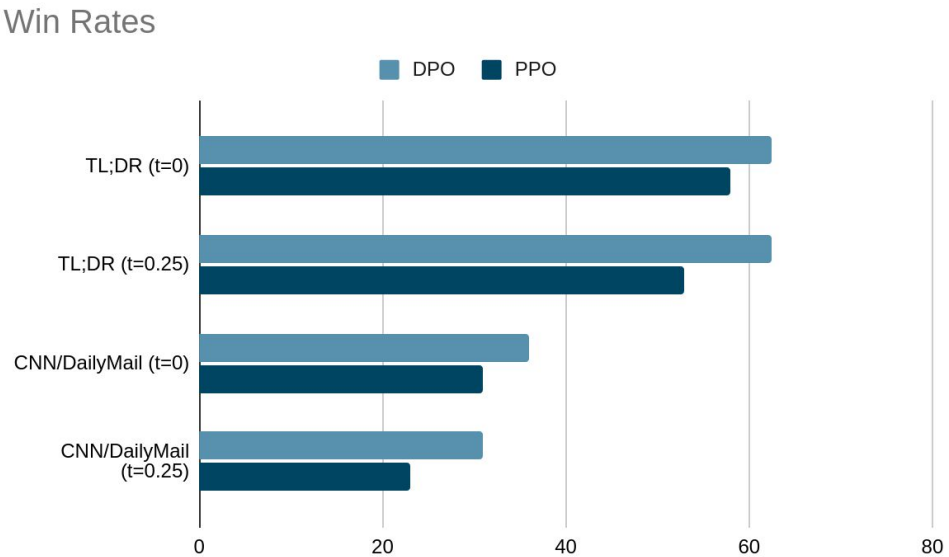
Instruction fine-tuning

To fully unlock the potential of our pretrained models in chat use cases, we innovated on our approach to instruction-tuning as well. Our approach to post-training is a combination of supervised fine-tuning (SFT), rejection sampling, proximal policy optimization (PPO), and direct preference optimization (DPO). The quality of the prompts that are used in SFT and the preference rankings that are used in PPO and DPO has an outsized influence on the performance of aligned models. Some of our biggest improvements in model quality came from carefully curating this data and performing multiple rounds of quality assurance on annotations provided by human annotators.

Learning from preference rankings via PPO and DPO also greatly improved the performance of Llama 3 on reasoning and coding tasks. We found that if you ask a model a reasoning question that it struggles to answer, the model will sometimes produce the right reasoning trace: The model knows how to produce the right answer, but it does not know how to select it. Training on preference rankings enables the model to learn how to select it.

The DPO vs PPO Debate

DPO vs PPO: Empirics



1. DPO is trained only on the Reddit TL;DR feedback data.
2. PPO uses a trained reward function and additional prompts for RL training.
3. We evaluate the trained policies on OOD CNN/DailyMail news summarization task.

DPO vs PPO:

DPO vs PPO:

DPO fits an implicit reward function:

DPO vs PPO:

DPO fits an implicit reward function:

1. Is the DPO implicit reward as good as the explicit one?

DPO vs PPO:

DPO fits an implicit reward function:

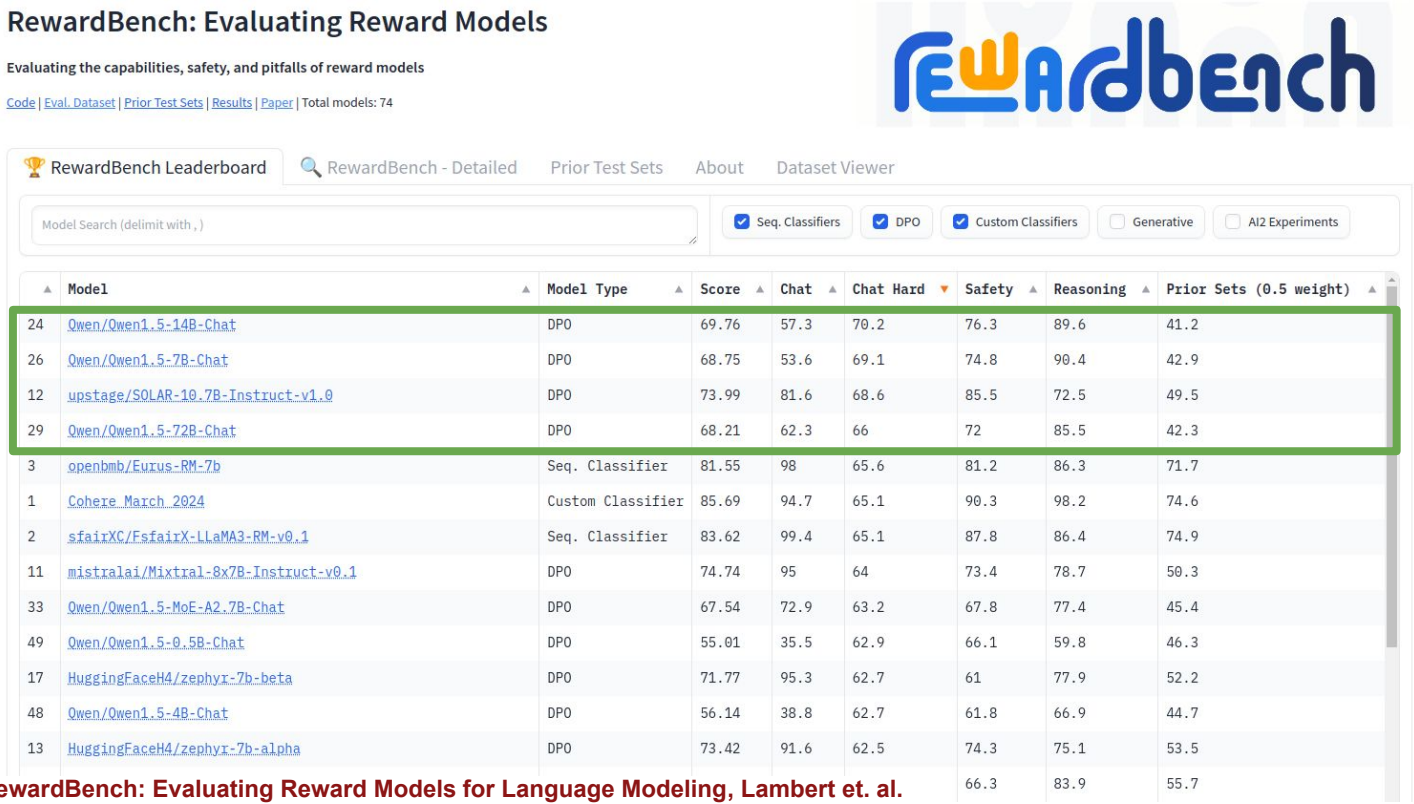
1. Is the DPO implicit reward as good as the explicit one?
2. Does using a weaker optimizer, such as PPO provide a better solution (regularization).

DPO vs PPO:

DPO fits an implicit reward function:

1. **Is the DPO implicit reward as good as the explicit one?**
2. Does using a weaker optimizer, such as PPO provide a better solution (regularization).

DPO vs PPO: Reward Function Quality - Chat



	Model	Model Type	Score	Chat	Chat Hard	Safety	Reasoning	Prior Sets (0.5 weight)
24	Qwen/Qwen1.5-14B-Chat	DPO	69.76	57.3	70.2	76.3	89.6	41.2
26	Qwen/Qwen1.5-7B-Chat	DPO	68.75	53.6	69.1	74.8	90.4	42.9
12	upstage/SOLAR-10.7B-Instruct-v1.0	DPO	73.99	81.6	68.6	85.5	72.5	49.5
29	Qwen/Qwen1.5-72B-Chat	DPO	68.21	62.3	66	72	85.5	42.3
3	openbmb/Eurus-RM-7b	Seq. Classifier	81.55	98	65.6	81.2	86.3	71.7
1	Cohere March 2024	Custom Classifier	85.69	94.7	65.1	90.3	98.2	74.6
2	sfairXC/FsfairX-LLaMA3-RM-v0.1	Seq. Classifier	83.62	99.4	65.1	87.8	86.4	74.9
11	mistralai/Mixtral-8x7B-Instruct-v0.1	DPO	74.74	95	64	73.4	78.7	50.3
33	Qwen/Qwen1.5-MoE-A2.7B-Chat	DPO	67.54	72.9	63.2	67.8	77.4	45.4
49	Qwen/Qwen1.5-0.5B-Chat	DPO	55.01	35.5	62.9	66.1	59.8	46.3
17	HuggingFaceH4/zephyr-7b-beta	DPO	71.77	95.3	62.7	61	77.9	52.2
48	Qwen/Qwen1.5-4B-Chat	DPO	56.14	38.8	62.7	61.8	66.9	44.7
13	HuggingFaceH4/zephyr-7b-alpha	DPO	73.42	91.6	62.5	74.3	75.1	53.5
						66.3	83.9	55.7

RewardBench: Evaluating Reward Models for Language Modeling, Lambert et. al.

DPO vs PPO: Reward Function Quality - Reasoning

RewardBench: Evaluating Reward Models

Evaluating the capabilities, safety, and pitfalls of reward models

Code | Eval. Dataset | Prior Test Sets | Results | Paper | Total models: 74

RewardBench Leaderboard | RewardBench - Detailed | Prior Test Sets | About | Dataset Viewer

Model Search (delimit with ,)

Seq. Classifiers | DPO | Custom Classifiers | Generative | AI2 Experiments

	Model	Model Type	Score	Chat	Chat Hard	Safety	Reasoning	Prior Sets (0.5 weight)
1	Cohere March 2024	Custom Classifier	85.69	94.7	65.1	90.3	98.2	74.6
26	Qwen/Qwen1.5-7B-Chat	DPO	68.75	53.6	69.1	74.8	90.4	42.9
24	Qwen/Qwen1.5-14B-Chat	DPO	69.76	57.3	70.2	76.3	89.6	41.2
7	stabilityai/stablelm-2-12b-chat	DPO	77.42	96.6	55.5	82.6	89.4	48.4
19	jondurbin/bagel-dpo-34b-v0.5	DPO	71.5	93.9	55	61.5	88.9	44.9
22	0-hero/Matter-0.1-7B-DPO-preview	DPO	71.19	89.4	57.7	58	88.5	53.5
4	Nexusflow/Starling-RM-34B	Seq. Classifier	81.44	96.9	57.2	88.2	88.5	71.4
2	sfairXC/FsfairX-LLaMA3-RM-v0.1	Seq. Classifier	83.62	99.4	65.1	87.8	86.4	74.9
3	openbmb/Eurus-RM-7b	Seq. Classifier	81.55	98	65.6	81.2	86.3	71.7
29	Qwen/Qwen1.5-72B-Chat	DPO	68.21	62.3	66	72	85.5	42.3
15	0-hero/Matter-0.1-7B-boost-DPO-preview	DPO	73.35	91.1	61	66.3	83.9	55.7
36	openbmb/MiniCPM-2B-dpo-fp32	DPO	66.25	89.1	49.3	52.5	82.3	49.6
16	HuggingFaceH4/starchat2-15b-v0.1	DPO	72.08	93.9	55.5	65.8	81.6	55.2
11	[illegible]	DPO	74.74	95	64	73.4	78.7	50.3

RewardBench: Evaluating Reward Models for Language Modeling, Lambert et. al.

DPO vs PPO:

DPO fits an implicit reward function:

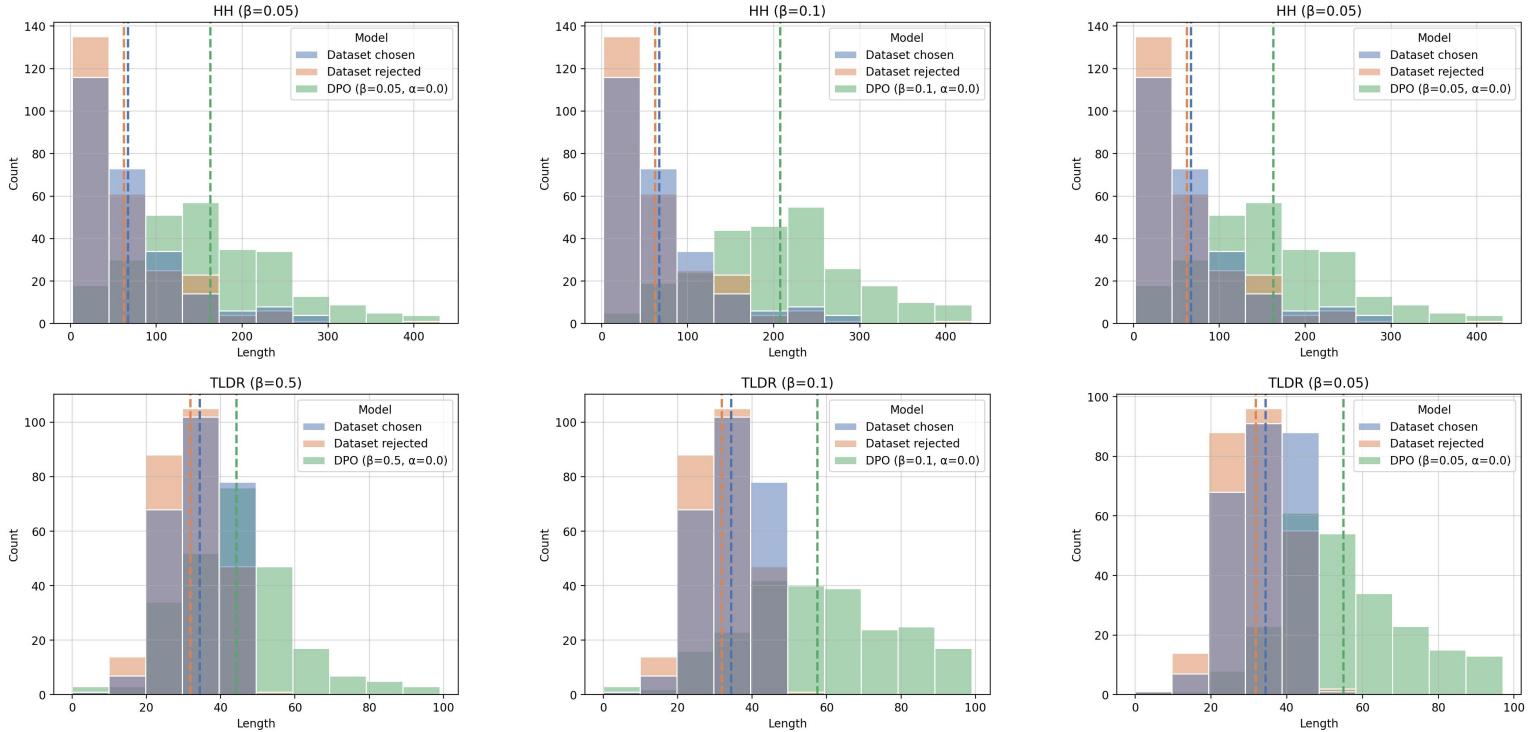
1. **Is the DPO implicit reward as good as the explicit one?**
2. Does using a weaker optimizer, such as PPO provide a better solution (regularization).

DPO vs PPO:

DPO fits an implicit reward function:

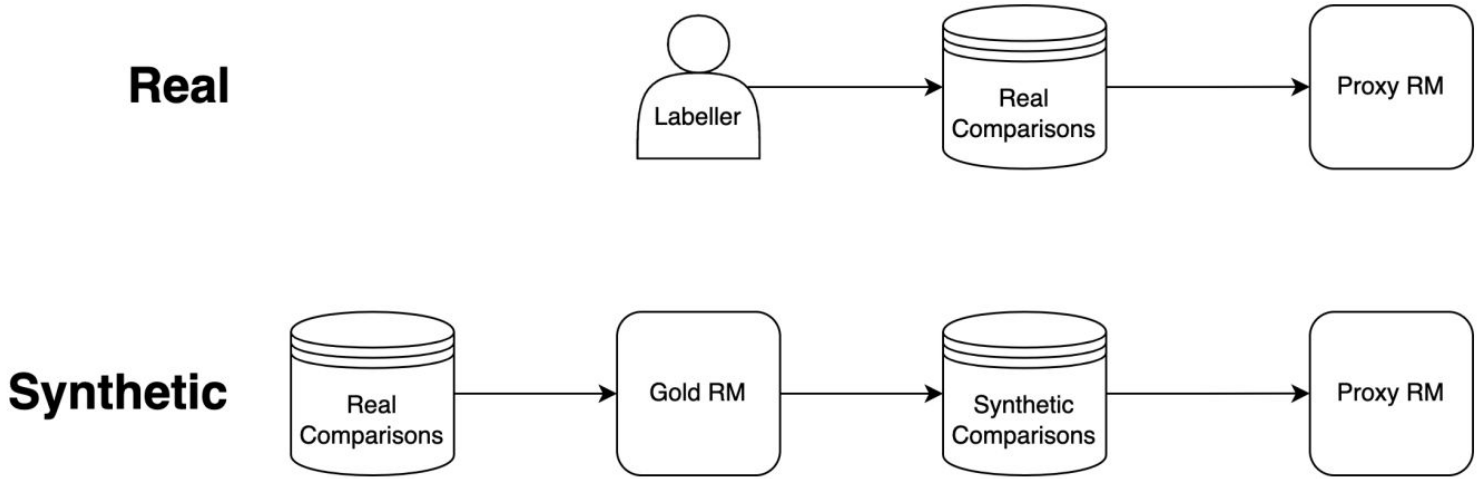
1. Is the DPO implicit reward as good as the explicit one?
2. **Does using a weaker optimizer, such as PPO provide a better solution (regularization).**

DPO vs PPO: Reward Hacking



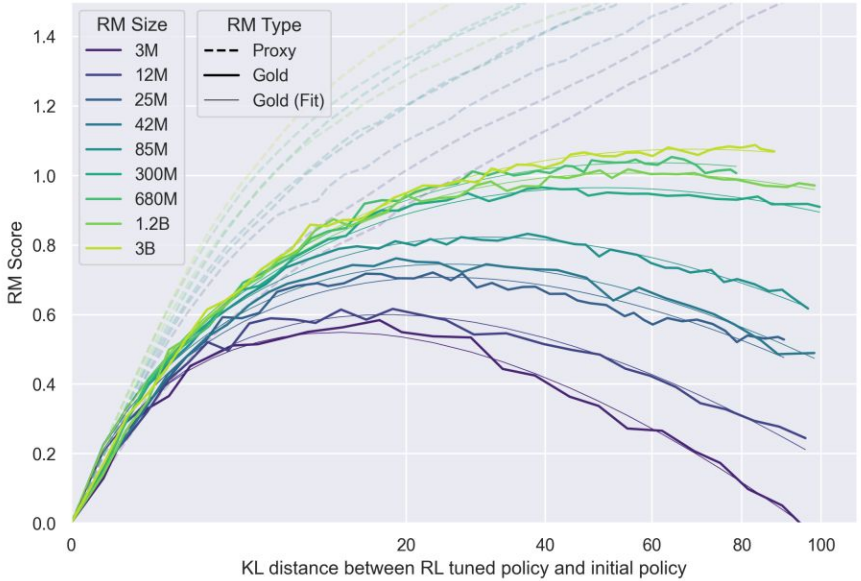
Disentangling Length from Quality in Direct Preference Optimization, Park et. al.

DPO vs PPO: Reward Hacking



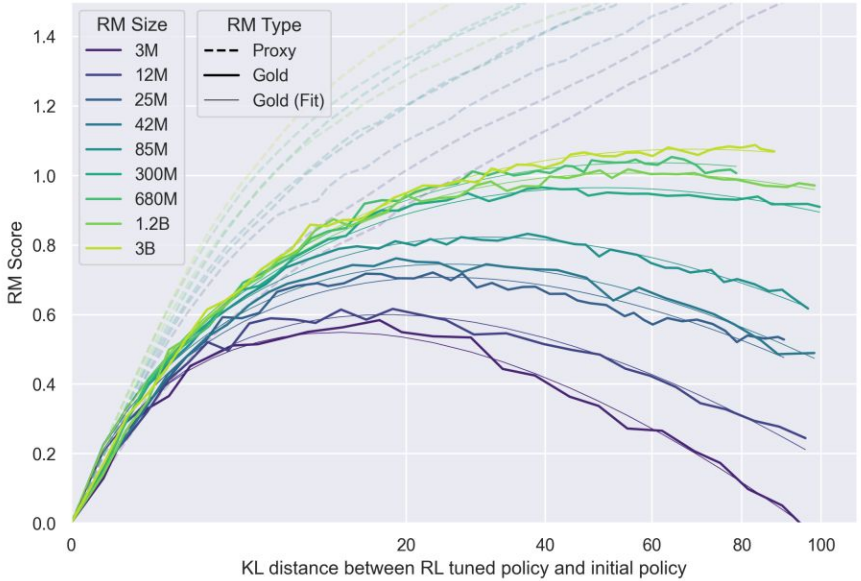
Scaling Laws for Reward Model Overoptimization, Gao et. al.

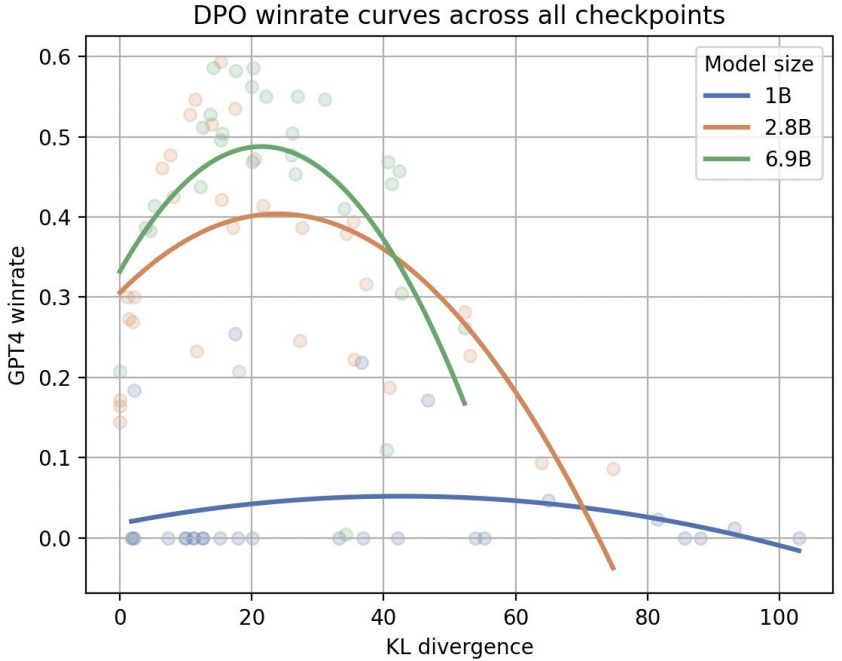
DPO vs PPO: Reward Hacking



Scaling Laws for Reward Model Overoptimization, Gao et. al.

DPO vs PPO: Reward Hacking





Scaling Laws for Reward Model Overoptimization, Gao et. al.