

## Lecture 8: Policy Gradients and Imitation learning

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2026

Which of the following are true about REINFORCE? In the following options, PG stands for policy gradient.

- a) Adding a baseline term can help to reduce the variance of the PG updates T
- b) It will converge to a global optima F  $V(\pi_\theta)$
- c) It can be initialized with a sub-optimal, deterministic policy and still converge to a local optima, given the appropriate step sizes F
- d) If we take one step of PG, it is possible that the resulting policy gets worse (in terms of achieved returns) than our initial policy T

*↑ part of motivation for PPD*

Which of the following are true about REINFORCE? In the following options, PG stands for policy gradient.

- (a) Adding a baseline term can help to reduce the variance of the PG updates
- (b) It will converge to a global optima
- (c) It can be initialized with a sub-optimal, deterministic policy and still converge to a local optima, given the appropriate step sizes
- (d) If we take one step of PG, it is possible that the resulting policy gets worse (in terms of achieved returns) than our initial policy

a, d are true

- In class
- You are allowed 1 side of 1 8.5" x 11" sheet of notes
- All material through today's lecture is eligible for the exam
- See Ed post for past related practice materials
- Good luck!

- Last time: Policy search continued and Imitation Learning
- This time: Imitation Learning and RLHF and maybe DPO
- Next time: Midterm

# How Can RL Enable Transformative LLM?

PM You

Please write me a program to demonstrate how RLHF works. Please be brief in your explanations and only say 1-2 sentences before you show me the code.

ChatGPT

Reinforcement Learning from Human Feedback (RLHF) trains a model to perform tasks based on human-derived feedback. Here's a simplified Python program that demonstrates RLHF using a scenario where an AI chooses responses in a conversation.

```
python
Copy code

import numpy as np

# Define possible actions (responses) the AI can use
actions = ["I'm doing well, thanks!", "I'm not sure how I feel.", "Let's talk about RLHF"]
q_table = np.zeros(len(actions))  # Initialize Q-values for each action to zero

def human_feedback(action):
    feedback = {"I'm doing well, thanks!": 10, "I'm not sure how I feel.": -5, "Let's talk about RLHF": 5}
    return feedback[action]

# Hyperparameters
learning_rate = 0.1
discount_factor = 0.6

# Training loop
for episode in range(100):
    action_index = np.random.choice(len(actions))  # Randomly choose an action
```

# Imitation Learning

# Table of Contents

recall in IL  
 $T_1, S, a, s', a', \dots -$   
 $T_2$    
} set of traj

## 1 Imitation Learning

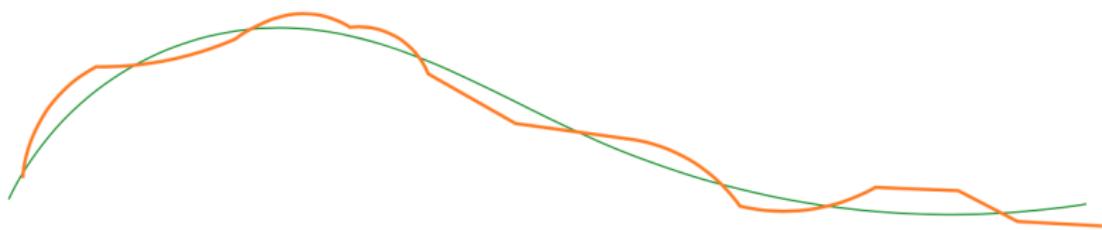
- DAGGER
- Reward Learning
- Human Input to Train RL Agents
- Preference Learning
- From Backflips to ChatGPT.<sup>1</sup>
- From RLHF to Direct Preference Optimization

①  $\hookrightarrow$  learn a  $\pi$  to match expert policy  
②  $\hookrightarrow$  learn the reward func

① supervised learning  
 $s \rightarrow a$

<sup>1</sup>Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

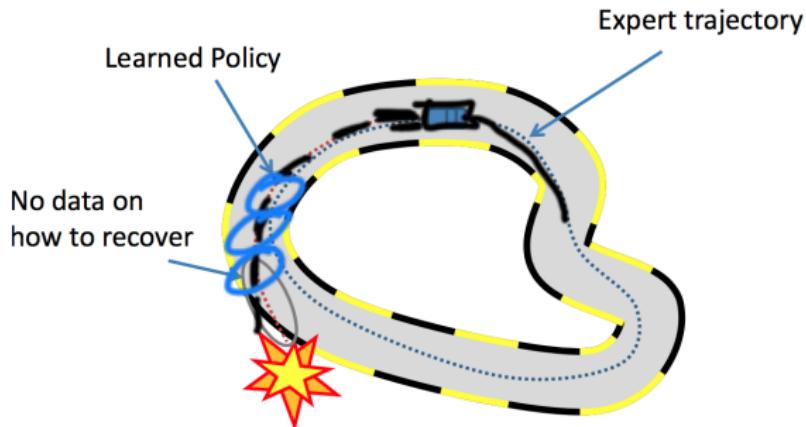
Supervised learning assumes iid.  $(s, a)$  pairs and ignores temporal structure  
Independent in time errors:



Error at time  $t$  with probability  $\leq \epsilon$

$\mathbb{E}[\text{Total errors}] \leq \epsilon T$

## Problem: Compounding Errors

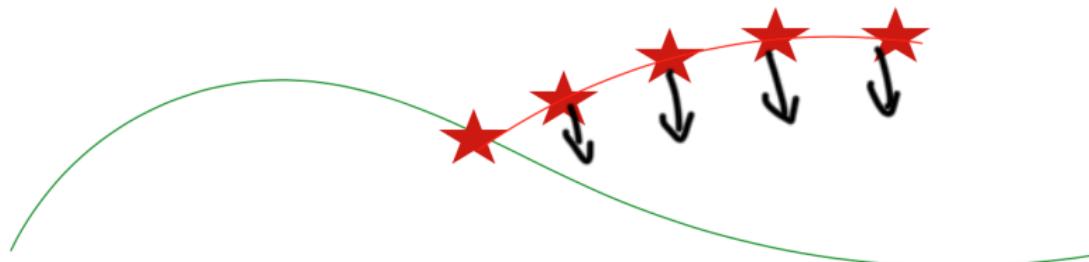


Data distribution mismatch!

In supervised learning,  $(x, y) \sim D$  during train **and** test. In MDPs:

- Train:  $s_t \sim D_{\pi^*}$
- Test:  $s_t \sim D_{\pi_\theta}$

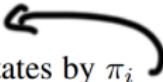
## Problem: Compounding Errors



- Error at time  $t$  with probability  $\epsilon$
- Approximate intuition:  $\mathbb{E}[\text{Total errors}] \leq \epsilon(T + (T - 1) + (T - 2) \dots + 1) \propto \underline{\epsilon T^2}$
- Real result requires more formality. See Theorem 2.1 in <http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats10-paper.pdf> with proof in supplement: <http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats10-sup.pdf>

~ 2020 Imit learning  
fund limit

```
Initialize  $\mathcal{D} \leftarrow \emptyset$ .  
Initialize  $\hat{\pi}_1$  to any policy in  $\Pi$ .  
for  $i = 1$  to  $N$  do  
    Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ .  
     Sample  $T$ -step trajectories using  $\pi_i$ .  
    Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$   
    and actions given by expert.  
    Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ .  
    Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ .  
end for  
Return best  $\hat{\pi}_i$  on validation.
```



- Idea: Get more labels of the expert action along the path taken by the policy computed by behavior cloning
- Obtains a stationary deterministic policy with good performance under its induced state distribution
- Key limitation? *high human effort*

# Table of Contents

## 1 Imitation Learning

- DAGGER
- Reward Learning
- Human Input to Train RL Agents
- Preference Learning
- From Backflips to ChatGPT.<sup>2</sup>
- From RLHF to Direct Preference Optimization

<sup>2</sup>Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

- Given state space, action space, transition model  $P(s' | s, a)$
- No reward function  $R$
- Set of one or more expert's demonstrations  $(s_0, a_0, s_1, s_0, \dots)$   
(actions drawn from expert's policy  $\pi^*$ )
- Goal: infer the reward function  $R$
- Assume that the expert's policy is optimal. What can be inferred about  $R$ ?

- Given state space, action space, transition model  $P(s' | s, a)$
- No reward function  $R$
- Set of one or more expert's demonstrations  $(s_0, a_0, s_1, s_0, \dots)$   
(actions drawn from expert's policy  $\pi^*$ )
- Goal: infer the reward function  $R$
- Assume that the expert's policy is optimal.

**1** There is a single unique  $R$  that makes expert's policy optimal

**2** There are many possible  $R$  that makes expert's policy optimal

**3** It depends on the MDP

**4** Not sure

$$R \rightarrow \gamma R + c$$

$\gamma > 0$     $c$  anything

- Given state space, action space, transition model  $P(s' | s, a)$
- No reward function  $R$
- Set of one or more expert's demonstrations  $(s_0, a_0, s_1, s_0, \dots)$   
(actions drawn from expert's policy  $\pi^*$ )
- Goal: infer the reward function  $R$
- Assume that the expert's policy is optimal.

① There is a single unique  $R$  that makes expert's policy optimal

② There are many possible  $R$  that makes expert's policy optimal

③ It depends on the MDP

④ Not sure

Answer: There are an infinite set of  $R$ .

- Very powerful
- A lot of extensions
- Max entropy reward learning is an important idea

## Human Feedback and Reinforcement Learning from Human Preferences

# Table of Contents

- DAGGER
- Reward Learning

## 2 Human Feedback and Reinforcement Learning from Human Preferences

- Human Input to Train RL Agents
- Preference Learning
- From Backflips to ChatGPT.<sup>3</sup>
- From RLHF to Direct Preference Optimization

---

<sup>3</sup>Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

- 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

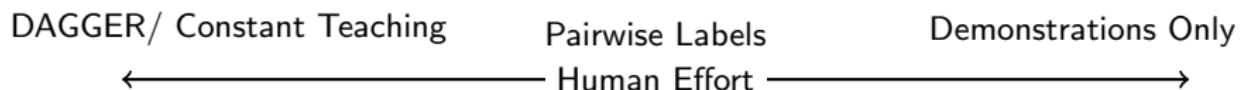


Teachable robots: Understanding human teaching behavior to build more effective robot learners. AL  
Thomaz, C Breazeal. Artificial Intelligence 2008

# 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/](http://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](http://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](http://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/](http://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](http://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/](http://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](http://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](http://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/](http://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

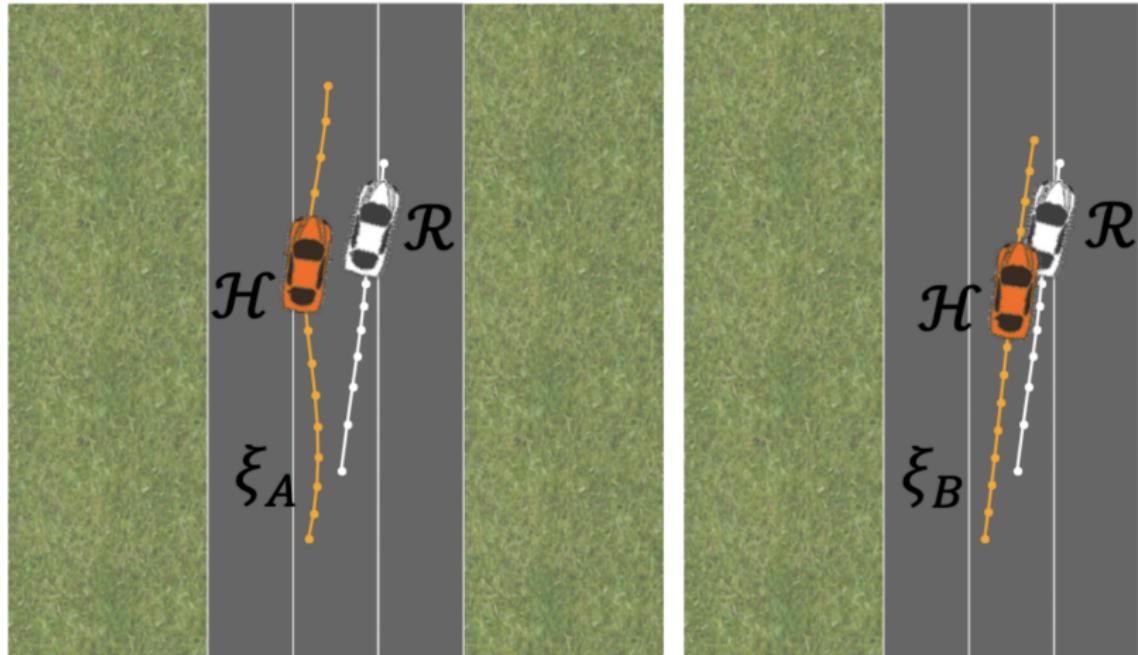
[cs.brown.edu/courses/cs159/home.html](http://cs.brown.edu/courses/cs159/home.html) ▾ Brown University ▾

Home | Course Info | Assignments | Syllabus And Lectures | Staff and Hours | LaTeX. An early model of parallel computation... Home Courses.

Slide from Yisong Yue

[http://www.yisongyue.com/courses/cs159/lectures/dueling\\_bandits\\_lecture.pdf](http://www.yisongyue.com/courses/cs159/lectures/dueling_bandits_lecture.pdf)





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

# Table of Contents

- DAGGER
- Reward Learning

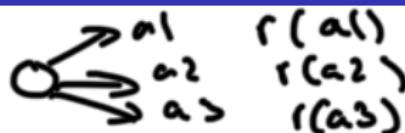
## 2 Human Feedback and Reinforcement Learning from Human Preferences

- Human Input to Train RL Agents
- **Preference Learning**
- From Backflips to ChatGPT.<sup>4</sup>
- From RLHF to Direct Preference Optimization

---

<sup>4</sup>Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

- 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?)



- 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<sup>5</sup>:  $K$  actions  $b_1, b_2, \dots, b_k$ . No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers  $b_i \succ b_j$  is

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

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

$$r(b_j) = r(b_i) = \frac{1}{1 + \exp(r(b_j) - r(b_i))}$$

<sup>5</sup>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.

## 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  $b_i \succ b_j$ ,  $0.5$  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

goal: learn rewards for each option  
 $r(b_1) \dots r(b_K)$

- First consider  $k$ -armed bandits<sup>6</sup>:  $K$  actions  $b_1, b_2, \dots, b_k$ . No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers  $b_i \succ b_j$  is

$$P(b_i \succ b_j) = \frac{\exp(r(b_i))}{\exp(r(b_i)) + \exp(r(b_j))} = p_{ij} \quad (2)$$

---

<sup>6</sup>We will see more on bandits later in the course

# Fitting the Parameters of a Bradley-Terry Model

- First consider  $k$ -armed bandits<sup>7</sup>:  $K$  actions  $b_1, b_2, \dots, b_k$ . No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers  $b_i \succ b_j$  is

$$P(b_i \succ b_j) = \frac{\exp(r(b_i))}{\exp(r(b_i)) + \exp(r(b_j))} = p_{ij} \quad (3)$$

- Assume have  $N$  tuples of form  $(b_i, b_j, \mu)$  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 *parameters are the rewards*

$$\text{loss} = - \sum_{(b_i, b_j, \mu) \in \mathcal{D}} \mu(1) \log P(b_i \succ b_j) + (1 - \mu(1)) \log P(b_j \succ b_i) \quad (4)$$

<sup>7</sup>We will see more on bandits later in the course

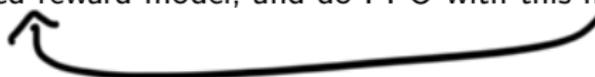
- Can also do this for trajectories
- Consider two trajectories,  $\tau^1(s_0, a_7, s_{14}, \dots)$  and  $\tau^2(s_0, a_6, s_{12}, \dots)$
- Let  $R^1 = \sum_{i=0}^{T-1} r_i^1$  be the (latent, unobserved) sum of rewards for trajectory  $\tau^1$  and similarly for  $R^2$ . *rewards per train step*
-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

$$\hat{P} [\tau^1 \succ \tau^2] = \frac{\exp \sum_{i=0}^{t-1} r_i^1}{\exp \sum_{i=0}^{t-1} r_i^1 + \exp \sum_{i=0}^{t-1} r_i^2}, \quad (5)$$

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

$$\hat{P} [\tau^1 \succ \tau^2] = \frac{\exp \sum_{i=0}^{t-1} r_i^1}{\exp \sum_{i=0}^{t-1} r_i^1 + \exp \sum_{i=0}^{t-1} r_i^2}, \quad (6)$$

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



2017

- 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&autoplay=0&player\\_id=0&app\\_id=58479](https://player.vimeo.com/video/754042470?h=e64a40690d&badge=0&autoplay=0&player_id=0&app_id=58479)

# Table of Contents

- DAGGER
- Reward Learning
- Human Input to Train RL Agents
- Preference Learning

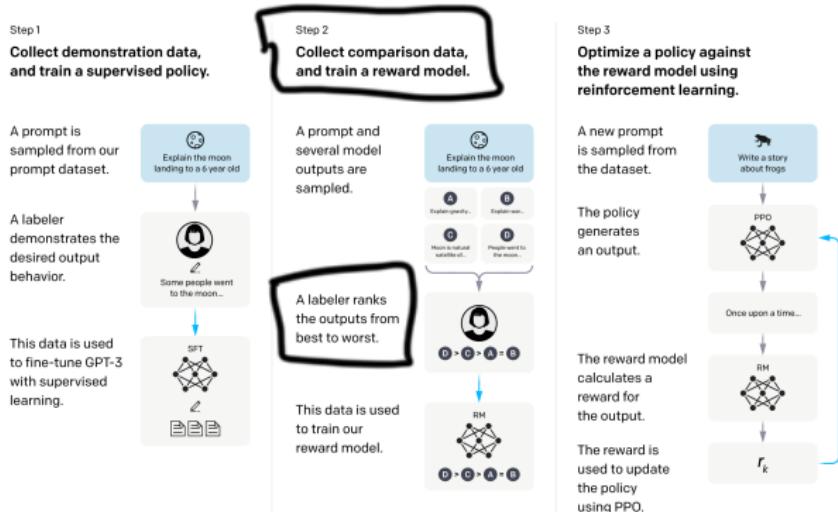
## 3 Preference learning

- From Backflips to ChatGPT.<sup>8</sup>
- From RLHF to Direct Preference Optimization

<sup>8</sup>Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

- 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.

$s_1$

1.2

$s_3$

$s_2$

Bradley-Terry [1952] paired comparison model

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

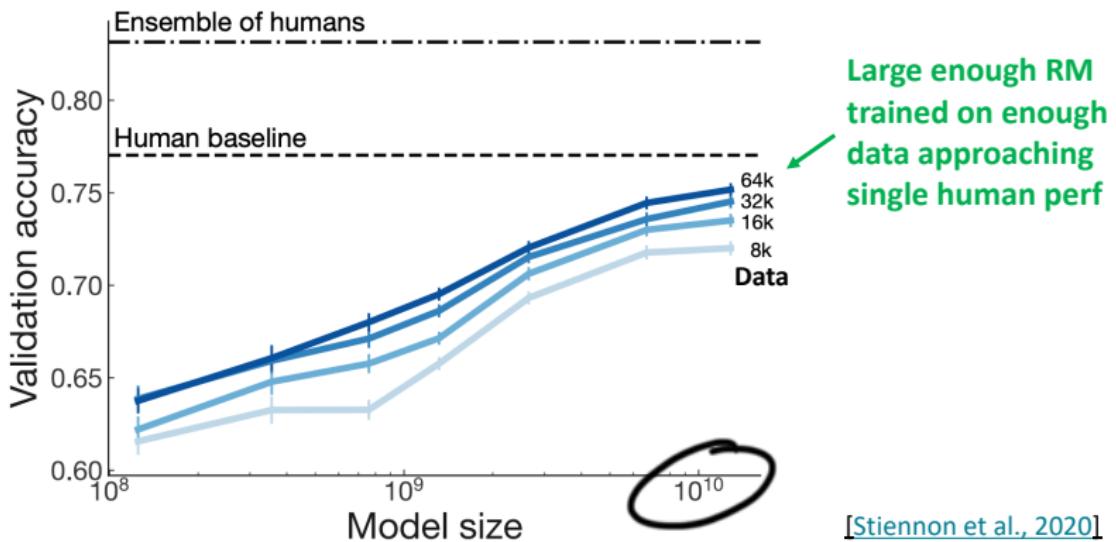
“winning”  
sample

“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



# RL from human feedback

## 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  $\theta$  we would like to optimize
  - Optimize the following reward with RL:

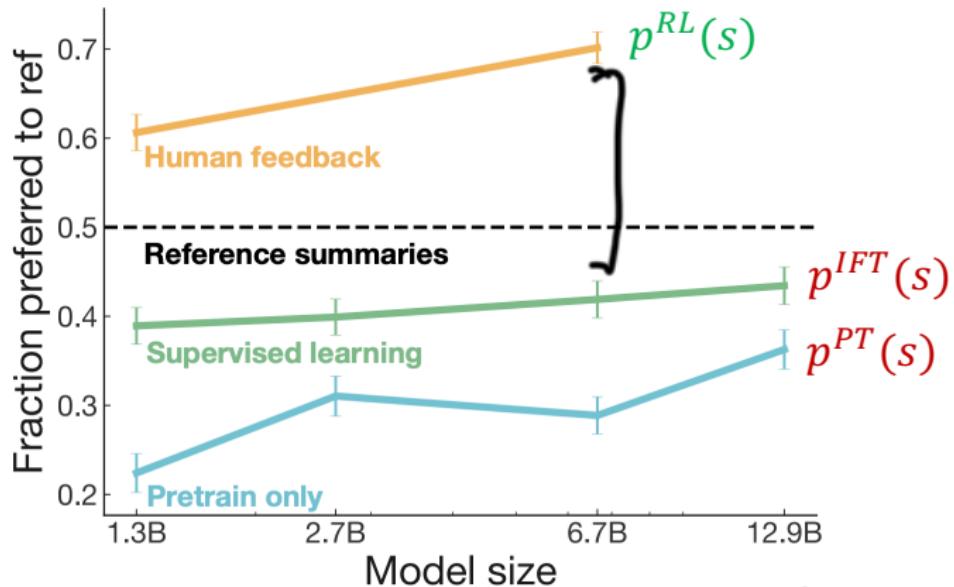
RLHF  
PPD

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

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

Step 1

**Collect demonstration data, and train a supervised policy.**

A prompt is sampled from our prompt dataset.

Explain the moon landing to a 6 year old

A labeler demonstrates the desired output behavior.



Some people went to the moon...

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.

Explain the moon landing to a 6 year old

A  
Explain gravity...  
B  
Explain wet...  
C  
Moon is natural satellite of...  
D  
People went to the moon...

A labeler ranks the outputs from best to worst.



D > C > A = B

This data is used to train our reward model.



D > C > A = B

Step 3

**Optimize a policy against the reward model using reinforcement learning.**

Write a story about frogs



Once upon a time...

RM

$r_k$

## Controlled comparisons of “RLHF” style algorithms

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

- 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]

# Table of Contents

- DAGGER
- Reward Learning
- Human Input to Train RL Agents
- Preference Learning

## 3 Preference learning

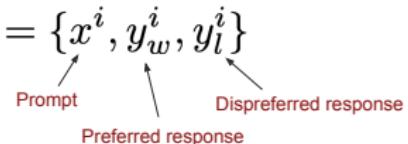
- From Backflips to ChatGPT.<sup>9</sup>
- From RLHF to Direct Preference Optimization

<sup>9</sup>Slides from part of Tatsu Hashimoto's Lecture 11 in CS224N

pref  $\rightarrow$  reward  $\rightarrow$  policy

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

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



**Bradley-Terry Model** connects rewards to preferences:

$$p(y_w \succ y_l \mid x) = \sigma(r(x, \overset{\swarrow}{y_w}) - r(x, \overset{\searrow}{y_l}))$$

Reward assigned to preferred and dispreferred responses

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

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

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

Now we have a **reward model**  $r_\phi$  that represents\* **goodness according to humans**

Now, learn a policy  $\pi_\theta$  achieving **high reward** while **staying close** to original model  $\pi_{\text{ref}}$

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



Sample from policy

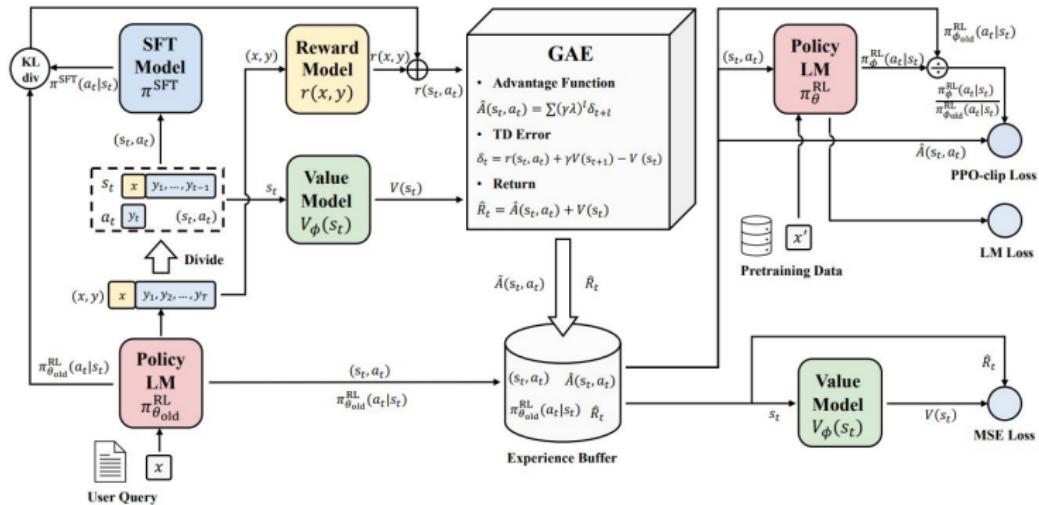


Want high reward...



...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

Stanford University

goal:  $\mu \rightarrow \pi$

DPO

## Direct Preference Optimization

### RLHF Objective

(get high reward, stay close  
to reference model)

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

any reward function

# Direct Preference Optimization

## RLHF Objective

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

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

any reward function

## Closed-form Optimal Policy

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

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

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

## Closed-Form Optimal Policy

$$\begin{aligned}
 & \text{arg max}_{\pi} \mathbb{E}_{x \sim D, y \sim \pi} [r(x, y) - \beta D_{KL}(\pi(y|x) || \pi_{\text{ref}}(y|x))] \\
 &= \text{arg} " \left[ r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \\
 &= \max_{\pi} \mathbb{E}_{x \sim D, y \sim \pi} \left[ -\frac{1}{\beta} \left[ \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \right] \quad \beta > 0 \\
 &= \max_{\pi} \mathbb{E}_{x \sim D, y \sim \pi} \left[ \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \log \exp \left( \frac{1}{\beta} r(x, y) \right) \right] \\
 &= " \left[ \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x) \exp \left( \frac{1}{\beta} r(x, y) \right)} \right] \\
 & \left( \begin{array}{l} z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp \left( \frac{1}{\beta} r(x, y) \right) \\ = \left[ \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x) \exp \left( \frac{1}{\beta} r(x, y) \right)} \cdot \frac{z(x)}{z(x)} \right] \end{array} \right) \\
 &= \underset{\pi}{\text{arg min}} \left[ \log \frac{\pi(y|x)}{\frac{1}{z(x)} \pi_{\text{ref}}(y|x) \exp \left( \frac{1}{\beta} r(x, y) \right)} - \log z(x) \right]
 \end{aligned}$$

# Direct Preference Optimization

## RLHF Objective

(get high reward, stay close to reference model)

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

any reward function

## Closed-form Optimal Policy

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

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

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

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

$$g_{\text{rel}} : \text{pref} \rightarrow \pi$$

## Direct Preference Optimization

### RLHF Objective

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

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

any reward function

### Closed-form Optimal Policy

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

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

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

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

### Rearrange

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

Stanford University

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

solve for  $r(x,y)$  in terms of  $\pi_r^*$ :

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

$$\log \frac{\pi_r^*(y|x)}{\pi_{ref}(y|x)} + \log Z(x) = \frac{1}{\beta} r(x,y) \quad \text{take log}$$

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

now plug into Bradley-Terry pref func  
so we can derive pref directly in terms  
of  $\pi_{ref}$  and  $\pi^*$

## DPO Derivation Cont.

$$p(y_1 > y_2 | x) = \frac{1}{1 + \exp(r^*(x, y_1) - r(x, y_2))}$$
$$= \frac{1}{1 + \exp \left[ \beta \log \frac{\pi^*(y_1 | x)}{\pi_{ref}(y_1 | x)} - \beta \log \frac{\pi^*(y_2 | x)}{\pi_{ref}(y_2 | x)} \right]}$$

directly solve for  $\pi^*$   $\nabla$

no reward

no  $Z(x)$

# Direct Preference Optimization

## RLHF Objective

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

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

any reward function

## Closed-form Optimal Policy

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

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

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

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

## Rearrange

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

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

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

# Direct Preference Optimization: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

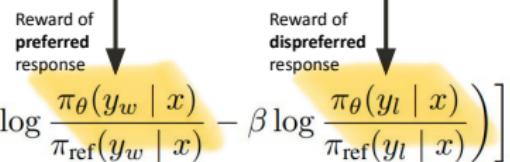
A loss function on policies

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

Derived from the Bradley-Terry model of human preferences:

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

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


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

# Direct Preference Optimization: Putting it together

**A loss function on reward functions**

Derived from the Bradley-Terry model of human preferences:

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

+

**A transformation between reward functions and policies**

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

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

Reward of preferred response

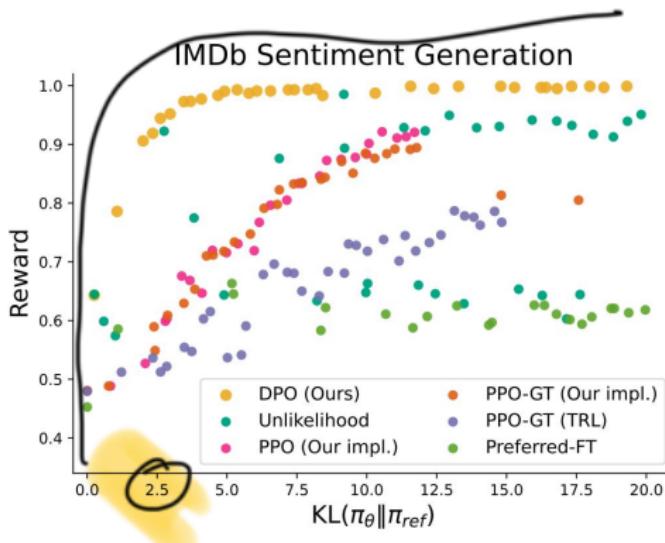
Reward of dispreferred response

Stanford University

# Results

Stanford University

# 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

Stanford University

# 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 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-4-934	1192	8.96	Proprietary
GPT-4-963	1158	9.18	Proprietary
Claude-1	1149	7.9	Proprietary
Claude-2.0	1131	8.06	Proprietary
<b>Mixtral-8x7B-Instruct-v0.1</b>	<b>1121</b>	<b>8.3</b>	Apache 2.0
Claude-2.1	1117	8.18	Proprietary
GPT-3.5-Turbo-963	1117	8.39	Proprietary
Gemini Pro	1111		Proprietary
Claude-Instant-1	1110	7.85	Proprietary
Julia-2-DPO-70B	1110	7.89	AII ImpACT Low-risk
Yi-34B-Chat	1110		Yi License
GPT-3.5-Turbo-834	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.

Stanford University

# 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 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-4-9334	1192	8.96	Proprietary
GPT-4-9633	1158	9.18	Proprietary
Claude-1	1149	7.9	Proprietary
Claude-2.0	1131	8.06	Proprietary
<b>Mixtral-8x7B-Instruct-v0.1</b>	<b>1121</b>	<b>8.3</b>	Apache 2.0
Claude-2.1	1117	8.18	Proprietary
GPT-3.5-Turbo-9633	1117	8.39	Proprietary
Gemini Pro	1111		Proprietary
Claude-Instant-1	1110	7.85	Proprietary
Tulu-2-DPO-70B	1110	7.89	AII ImpACT Low-risk
Yi-34B-Chat	1110		Yi License
GPT-3.5-Turbo-9334	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.

Stanford University

## 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.





- Learning and making decisions from human preferences is a rich area intersecting social choice, computational economics and AI
- New course at Stanford on this topic: Koyejo's CS329H: Machine Learning from Human Preferences

**Table 1:** Results of various Tetris agents.

Method	Mean Lines Cleared		Games for Peak
	at Game 3	at Peak	
TAMER	<b>65.89</b>	<b>65.89</b>	<b>3</b>
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