Lecture 8: Imitation Learning and RLHF

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2025

Refresh Your Understanding L8N1

Select all that are true

T

(a) Maximizing a lower bound on the performance gap between a new policy and and old policy can ensure monotonic convergence

F

(b) Behavior cloning requires knowing the dynamics model

F

(c) DAGGER uses demonstrations from experts but no further interactions

(d) Not sure

Refresh Your Understanding L8N1 Solutions

Select all that are true

- (a) Maximizing a lower bound on the performance gap between a new policy and and old policy can ensure monotonic convergence
- (b) Behavior cloning requires knowing the dynamics model
- (c) DAGGER uses demonstrations from experts but no further interactions
- (d) Not sure

How Can RL Enable This?

You

Please write me a program to demonstration 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

```python
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 abo
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, "
    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
```

Class Structure

- Last time: Policy search continued and Imitation Learning
- This time: Imitation Learning and RLHF
- Next time: Author of Direct Preference Optimization (best paper runner up at top ML conference) guest lecture

Today

- Imitation Learning
  - Max entropy inverse RL
- Human feedback
  - RLHF

imitation learning
demonstrations $(s_1, a_1, s_1', a_1', \ldots)$

Reward Learning

Feature Based Reward Function

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

Check Your Understanding L7N3: Feature Based Reward Function

- Given state space, action space, transition model $P(s' \mid s, a)$
- No reward function R
- Set of one or more expert's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from expert's policy π^*)
- 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

T

Check Your Understanding L7N3: Feature Based Reward Function

- Given state space, action space, transition model $P(s' \mid s, a)$
- No reward function R
- Set of one or more expert's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from expert's policy π^*)
- 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 .

Linear Feature Reward Inverse RL

- Linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $\boldsymbol{w} \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^{\pi}(s_0) = \mathbb{E}_{s \sim \pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0]$$

Linear Feature Reward Inverse RL

- Related to linear value function approximation
- Consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi(s_0) = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0] = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{w}^T x(s_t) \mid s_0]$$

$$= \boldsymbol{w}^T \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t x(s_t) \mid s_0]$$

$$= \boldsymbol{w}^T \mu(\pi)$$

- where $\mu(\pi)(s)$ is defined as the discounted weighted frequency of state features under policy π, starting in state s_0.

Relating Frequencies to Optimality

- Assume $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- $V^\pi = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] = \boldsymbol{w}^T \mu(\pi)$ where
  $\mu(\pi)(s) =$ discounted weighted frequency of state s under policy π.

$V^* \geq V^\pi$ because demonstrator was an optimal π^*

$w^T \mu^* \geq w^T \mu^\pi$

Relating Frequencies to Optimality

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi = \boldsymbol{w}^T \mu(\pi)$$

- $\mu(\pi)(s)$ = discounted weighted frequency of state s under policy π.

$$\mathbb{E}_{s\sim\pi^*}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] = V^* \geq V^\pi = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] \quad \forall \pi$$

- Therefore if the expert's demonstrations are from the optimal policy, to identify $\boldsymbol{w}$ it is sufficient to find w^* such that

$$w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi), \forall \pi \neq \pi^*$$

imagine you have dynamics models

Feature Matching

- Want to find a reward function such that the expert policy outperforms other policies.
- For a policy π to be guaranteed to perform as well as the expert policy π^*, sufficient if its discounted summed feature expectations match the expert's policy [Abbeel & Ng, 2004].
- More precisely, if

$$\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon$$

then for all w with $\|w\|_\infty \leq 1$ (uses Holder's inequality):

$$|w^T \mu(\pi) - w^T \mu(\pi^*)| \leq \epsilon$$

Ambiguity

- There is an infinite number of reward functions with the same optimal policy.
- There are infinitely many stochastic policies that can match feature counts
- Which one should be chosen?

Learning from Demonstration / Imitation Learning Pointers

- Many different approaches
- Two of the key papers are:
  - Maximumum Entropy Inverse Reinforcement Learning (Ziebart et al. AAAI 2008)
  - Generative adversarial imitation learning (Ho and Ermon, NeurIPS 2016)

Max Entropy Inverse RL. Ziebart et al. 2008. Note: Much of this presentation follows the slides from Katerina Fragkiadaki's Deep Reinforcement Learning and Control Lecture on Maximum Entropy Inverse RL.

Principle of Maximum Entropy

$\sum_s p(s) = 1$



- Recall that entropy of a distribution $p(s)$ is $-\sum_{s'} p(s = s') \log p(s = s')$
- Principle of max entropy: The probability distribution which best represents the current state of knowledge is the one with the largest entropy, given the constraints of precisely stated prior data.
- Intuitively: consider all probability distributions consistent with observed data, and select the probability distribution with the maximum entropy.

Principle of Maximum Entropy for IRL

- Recall that entropy of a distribution $p(s)$ is $-\sum_{s'} p(s = s') \log p(s = s')$
- Principle of max entropy: The probability distribution which best represents the current state of knowledge is the one with the largest entropy, given the constraints of precisely stated prior data.
- Intuitively: consider all probability distributions consistent with observed data, and select the probability distribution with the maximum entropy.
- In the linear reward case, this is equivalent to specifying the weights $\boldsymbol{w}$ that yield a policy with the max entropy constrained to matching the feature expectations:

$$\max_P -\sum_\tau P(\tau) \log P(\tau) s.t. \quad \sum_\tau P(\tau)\mu(\tau) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mu(\tau_i) \qquad \sum_\tau P(\tau) = 1 \tag{1}$$

- where $\mu(\tau)$ are the features for trajectory τ and $\mathcal{D}$ is the observed expert data

Ziebart et al., 2008

Matching Rewards

- In the linear reward case, this is equivalent to specifying the weights $\boldsymbol{w}$ that yield a policy with the max entropy constrained to matching the feature expectations:

$$\max_P -\sum_\tau P(\tau)\log P(\tau) s.t. \quad \sum_\tau P(\tau)\mu(\tau) = \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}}\mu(\tau_i) \qquad \sum_\tau P(\tau) = 1 \tag{2}$$

- where $\mu(\tau)$ are the features for trajectory τ and $\mathcal{D}$ is the observed expert data
- In general, would like to find a policy π that induces a distribution over trajectories $p(\tau)$ which has the same expected reward as the expert's demonstrations $\hat{P}(\tau)$ given a reward function r_ϕ ← unknown not computable

$$\max_{p(\tau)} -\sum_\tau p(\tau)\log p(\tau) \quad s.t. \quad \sum_\tau p(\tau) r_\phi(\tau) = \sum_\tau \hat{P}(\tau) r_\phi(\tau) \qquad \sum_\tau p(\tau) = 1 \tag{3}$$

expert's

Ziebart et al., 2008

Matching Rewards to Learning Policies

- In the linear reward case, this is equivalent to specifying the weights $\boldsymbol{w}$ that yield a policy with the max entropy constrained to matching the feature expectations:

$$\max_P -\sum_\tau P(\tau)\log P(\tau) s.t. \quad \sum_\tau P(\tau)\mu(\tau) = \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}}\mu(\tau_i) \qquad \sum_\tau P(\tau) = 1 \tag{4}$$

- where $\mu(\tau)$ are the features for trajectory τ and $\mathcal{D}$ is the observed expert data
- In general, would like to find a policy π that induces a distribution over trajectories $p(\tau)$ which has the same expected reward as the expert's demonstrations $\hat{P}(\tau)$ given a reward function r_ϕ

$\pi \rightarrow p(\tau)$

objective

$$\max_{p(\tau)} -\sum_\tau p(\tau)\log p(\tau) \quad s.t. \quad \sum_\tau p(\tau) r_\phi(\tau) = \sum_\tau \hat{P}(\tau) r_\phi(\tau) \qquad \sum_\tau p(\tau) = 1 \tag{5}$$

- To do so, will alternate between computing a reward function, using that reward function to learn an optimal policy, and then updating the trajectory / state frequencies needed to update the reward function
- Note: in original maximum entropy inverse RL paper, assumed dynamics / reward model is known

Ziebart et al., 2008

From Maximum Entropy to Probability over Trajectories

given constrained obj (aka eqns) if we know r_ϕ what would be the form of $p(\tau)$?

$$\mathcal{L}(p, \lambda) = \sum_\tau p(\tau) \log p(\tau) + \lambda_1 \left(\sum_\tau \hat{p}(\tau) r_\phi(\tau) - \sum_\tau p(\tau) r_\phi(\tau) \right) + \lambda_0 \left(\sum_\tau p(\tau) - 1 \right)$$

$$\frac{d\mathcal{L}}{dp(\tau)} = \log p(\tau) + p(\tau) \cdot \frac{1}{p(\tau)} - \lambda_1 r_\phi(\tau) + \lambda_0$$

$$= 0$$

$$\log p(\tau) = \lambda_1 r_\phi(\tau) - 1 - \lambda_0$$

$$p(\tau) \propto \exp(r_\phi(\tau))$$

Maximizing Entropy Over $\tau \equiv$ Maximize Likelihood of Observed Data Under Max Entropy (Exponential Family) Distribution (Jaynes 1957)

$$p(\tau \mid \phi) = \frac{1}{Z(\phi)} \exp(r_\phi(\tau)) \quad Z(\phi) = \sum_\tau \exp(r_\phi(\tau))$$

we don't know what ϕ is

this observation
($p(\tau)$ that max entropy s.t. the constraints looks like a normalized exponential model)
means we can learn r_ϕ by max prob of our observed data

Maximize (over r_ϕ)) Likelihood of Observed Data Under Max Entropy

$$\max_\phi \log \prod_{\tau_i \in D^*} p(\tau_i) \qquad D^* = \text{experts}$$

$$= \max_\phi \sum_{\tau_i \in D^*} \log \left(\frac{\exp r_\phi(\tau_i)}{\sum_\tau e^{r_\phi(\tau_i)}} \right) \qquad |D^*|$$

$$J(\phi) = \max_\phi \sum_{\tau_i \in D^*} r_\phi(\tau_i) - \log \left(\sum_{\tau_i} e^{r_\phi(\tau_i)} \right)$$

$$\nabla_\phi J = \sum_{\tau_i \in D^*} d/d\phi \, r_\phi(\tau_i) - |D^*| \sum_{\tau_i} \frac{e^{r_\phi(\tau_i)}}{\sum_{\tau_i'} e^{r_\phi(\tau_i')}} \frac{d r_\phi(\tau_i)}{d\tau_i}$$

$$p(\tau_i | \phi)$$

$$= \sum_{\tau_i} d/d\phi \, r_\phi(\tau_i) - |D^*| \sum_{\tau_i} p(\tau_i | \phi) \frac{d r_\phi(\tau_i)}{d\tau_i}$$

From Trajectories to States

$$p(\tau) = p(s_i) \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|a_t, s_t)$$

$$e^{-r_\phi(\tau)} = e^{-\sum_{s \in \tau} r_\phi(s)}$$

can expand $J(\phi)$ to be written in terms of states

State Densities

if tabular and dynamics known & π given

$\mu_1(s) = p(s_1)$

post class add

$r_\phi(s) = \phi f(s)$ $f(s)$ = features

$\frac{d r_\phi(s)}{d\phi} = f(s)$

Maximum entropy inverse RL

Note: Assuming known dynamics model and linear rewards

1. Input: expert demonstrations $\mathcal{D}$
2. Initialize r_ϕ
3. Compute optimal $\pi(a|s)$ given r_ϕ e.g. with value iteration)
4. Compute state visitation frequencies $p(s|[\phi, T)$
5. Compute gradient on reward model

$$\nabla J(\phi) = \frac{1}{N} \sum_{\tau_d} f_{\tau_D} - \sum_s p(s|\phi, r) f_s \tag{6}$$

6. Update ϕ with one gradient step
7. Go to step 3

Check Your Understanding: L8N2 Maximum entropy inverse RL

Note: Assuming known dynamics model and linear rewards

1. Input: expert demonstrations $\mathcal{D}$
2. Initialize r_ϕ
3. Compute optimal $\pi(a|s)$ given r_ϕ e.g. with value iteration)
4. Compute state visitation frequencies $p(s(\phi, T)$
5. Compute gradient on reward model

$$\nabla J(\phi) = \tag{7}$$

6. Update ϕ with one gradient step
7. Go to step 3

- What steps in the above algorithm rely on knowing the dynamics model? (select all)
- (1) Computing the optimal policy
- (2) Computing the state visitation frequencies
- (3) Computing the gradient
- (4) No steps required it
- (5) Not sure

Check Your Understanding: L8N2 Maximum entropy inverse RL Solutions

Note: Assuming known dynamics model and linear rewards

1. Input: expert demonstrations $\mathcal{D}$
2. Initialize r_ϕ
3. Compute optimal $\pi(a|s)$ given r_ϕ e.g. with value iteration)
4. Compute state visitation frequencies $p(s(\phi, T)$
5. Compute gradient on reward model

$$\nabla J(\phi) = \tag{8}$$

6. Update ϕ with one gradient step
7. Go to step 3

- What steps in the above algorithm rely on knowing the dynamics model? (select all)
- (1) Computing the optimal policy
- (2) Computing the state visitation frequencies
- (3) Computing the gradient
- (4) No steps required it
- (5) Not sure

1 and 2

Max Entropy IRL Summary

- Max entropy approach has been hugely influential
- Initial formulation (Ziebart et al) using linear rewards and assumed dynamics model is known
  - Check your understanding: was this needed in behavioral cloning?
- Finn et al. 2016 (Guided cost learning: Deep inverse optimal control via policy optimization) showed how to use general reward/cost functions and removed the need to know the dynamics model

Imitation Learning Summary

- Imitation learning can greatly reduce the amount of data need to learn a good policy
- Challenges remain and one exciting area is combining inverse RL / learning from demonstration and online reinforcement learning
- For a look into some of the theory between imitation learning and RL, see Sun, Venkatraman, Gordon, Boots, Bagnell (ICML 2017)

Imitation learning: What You Should Know

- Define behavior cloning and how it differs from reinforcement learning
- Understand principle of maximum entropy, the resulting distribution over trajectories, and how this can be used to learn a reward function and fit a policy

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.

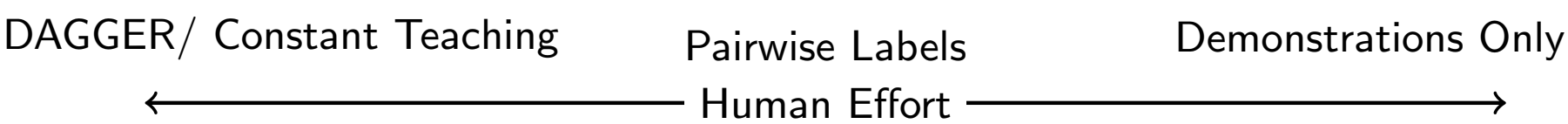
Method	Mean Lines Cleared		Games
	at Game 3	at Peak	for Peak
TAMER	**65.89**	**65.89**	**3**
RRL-KBR [15]	5	50	120
Policy Iteration [2]	~ 0 (no learning until game 100)	3183	1500
Genetic Algorithm [5]	~ 0 (no learning until game 500)	586,103	3000
CE+RL [17]	~ 0 (no learning until game 100)	348,895	5000

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 ...

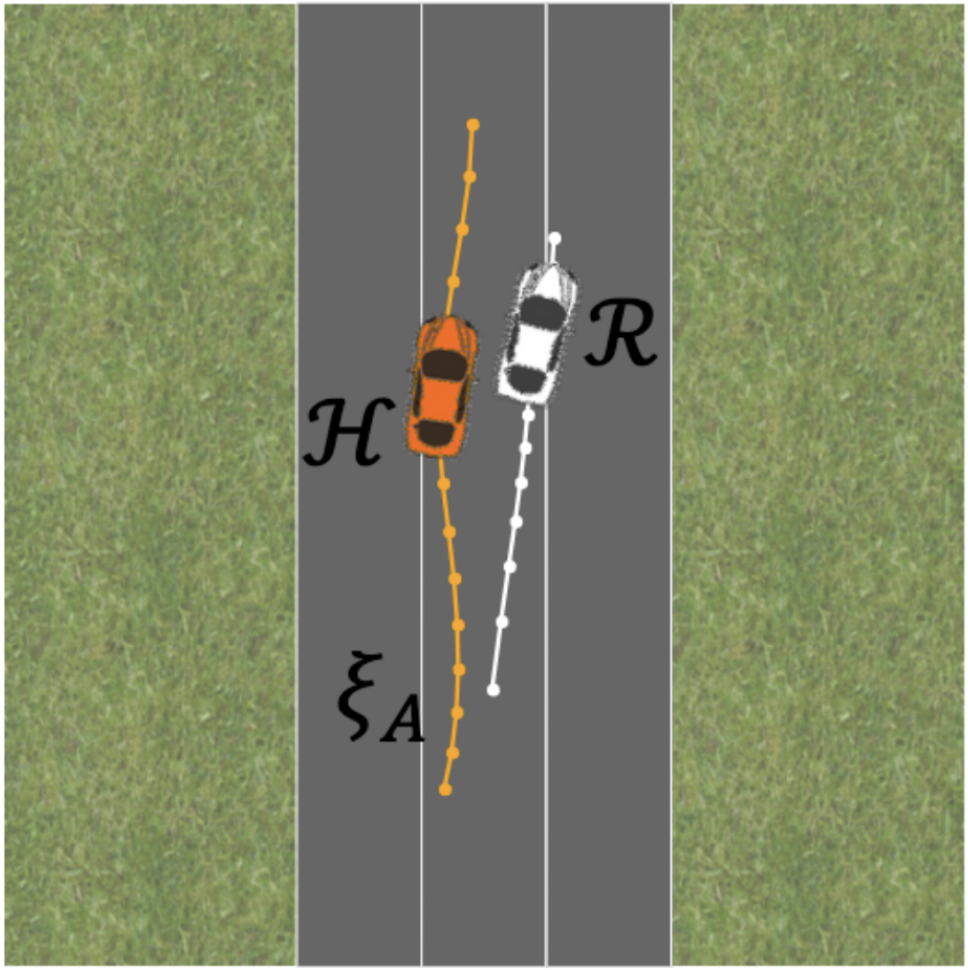
CS 159: Introduction to Parallel Processing - Info.sjsu.edu
info.sjsu.edu › ... › Courses ▾ San Jose State University ▾
CS 159. Introduction to Parallel Processing. Description Major parallel architectures: shared memory, distributed memory, SIMD, MIMD. Parallel algorithms: ...

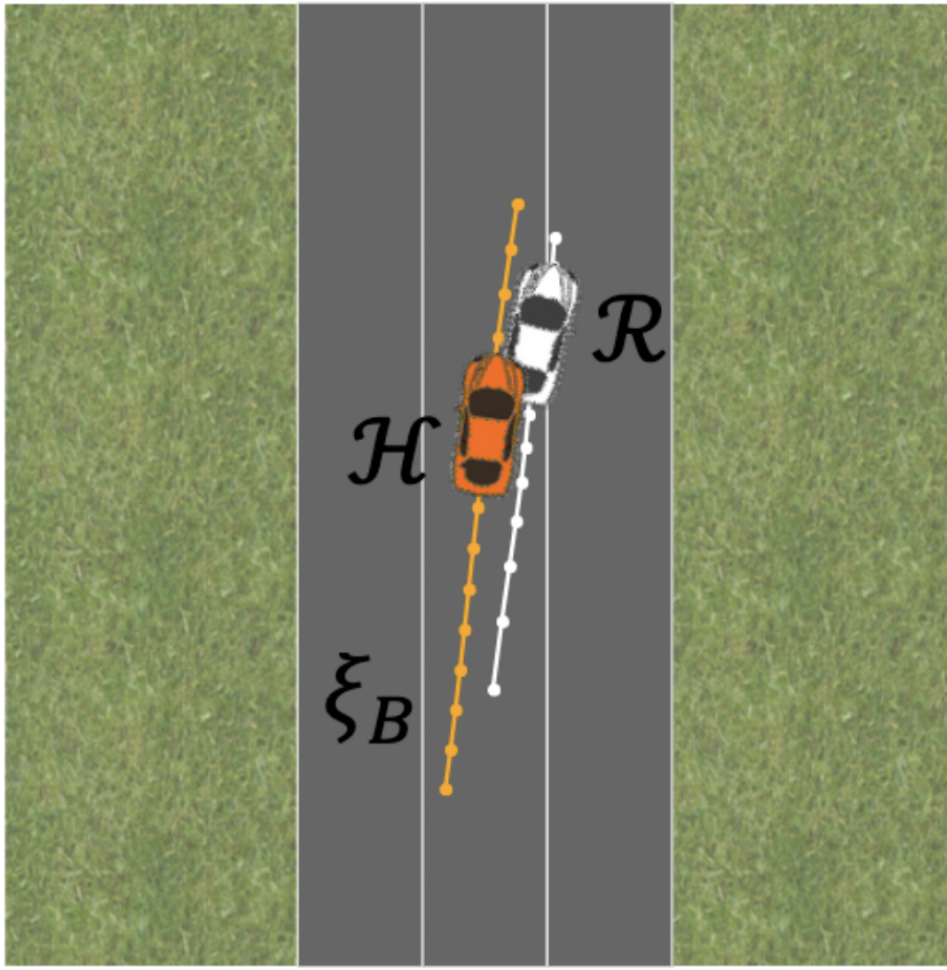
CS 159: Advanced Topics in Machine Learning - Yisong Yue
www.yisongyue.com/courses/**cs159**/ ▾
CS 159: Advanced Topics in Machine Learning (Spring 2016). Course Description. This course will cover a mixture of the following topics: Online Learning ...

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.

show a_1, a_3 ask $r(a_1) > r(a_3)$

- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

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

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

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

$r(b_j) = r(b_i)$
then $p_{ij} = .5$

[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.

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

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

learning $r(b)$ parameters

[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

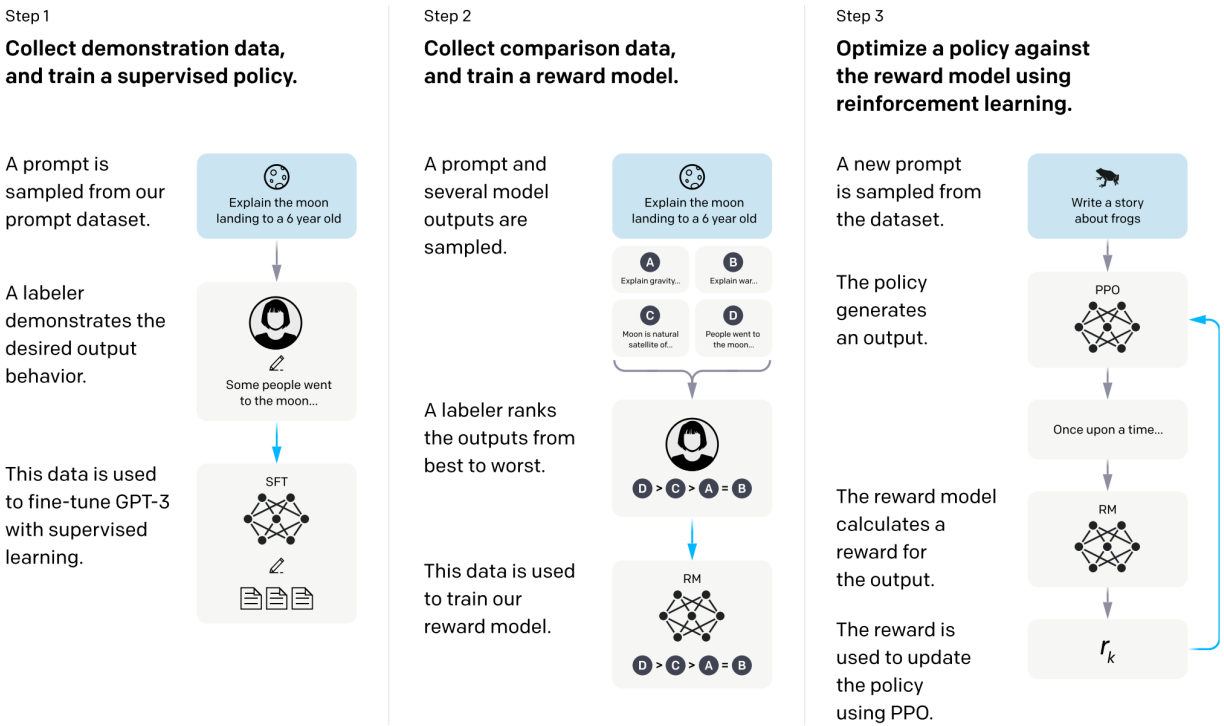
From Backflips to ChatGPT. [4]

[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

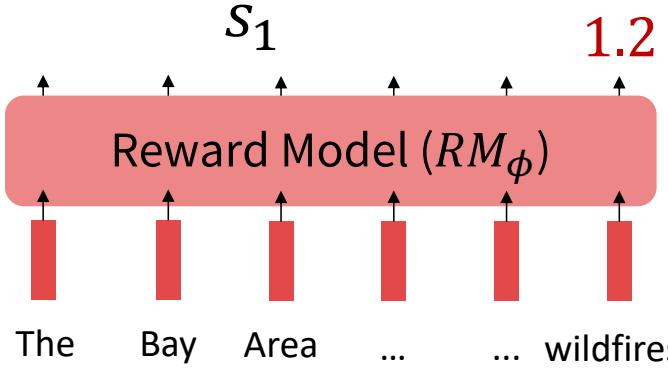


- First step: instruction tuning!
- Second + third steps: maximize reward (but how??)

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

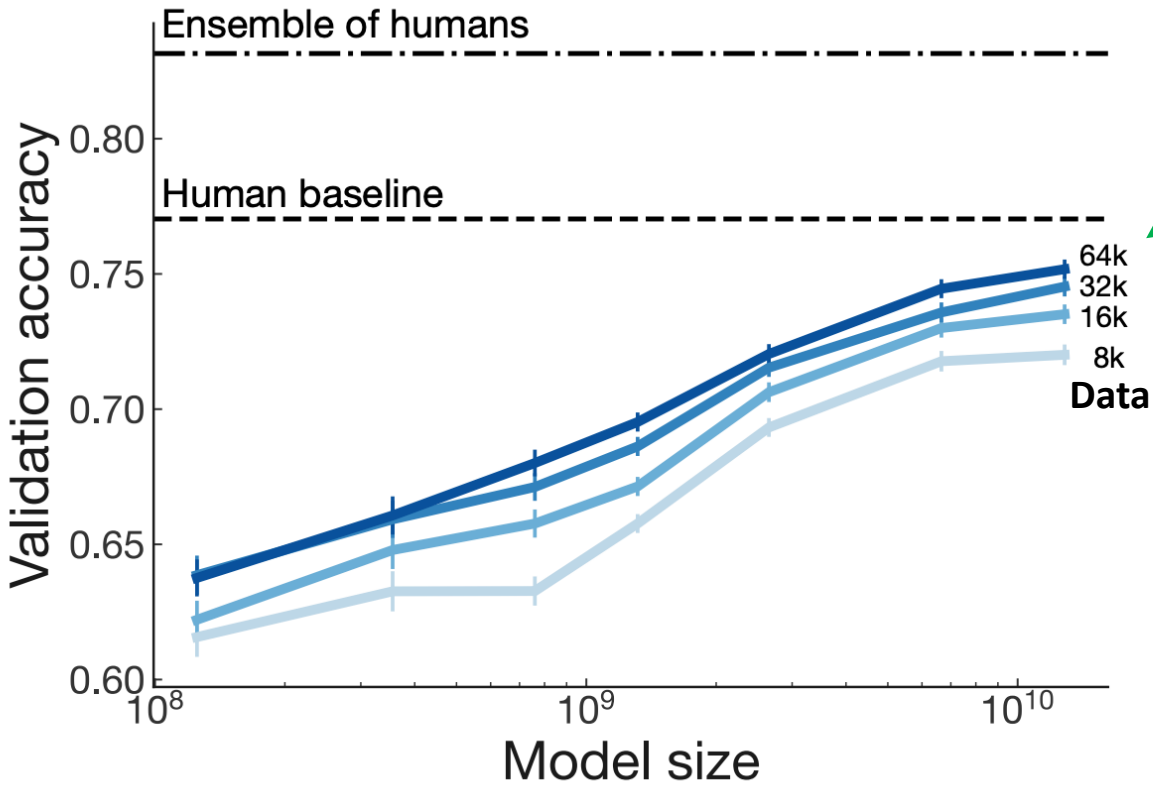
$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

"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]

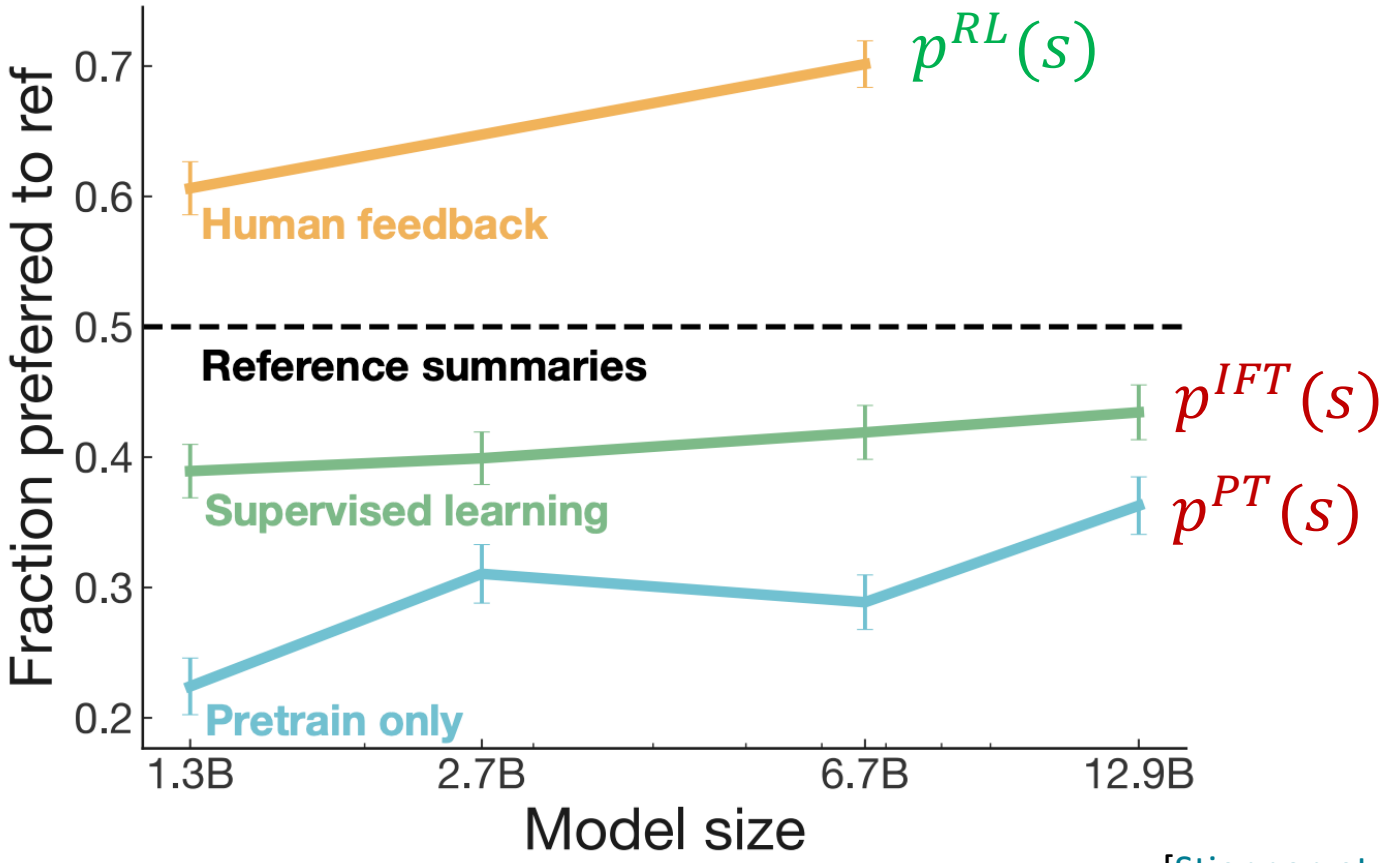
- Finally, we have everything we need:
  - A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
  - A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
  - A method for optimizing LM parameters towards an arbitrary reward function.
- 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!

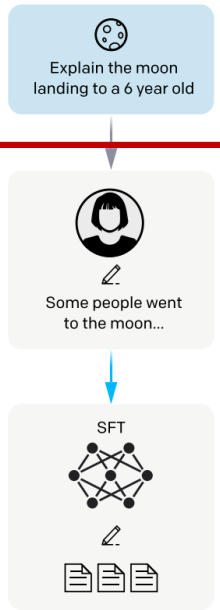
Step 1

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.



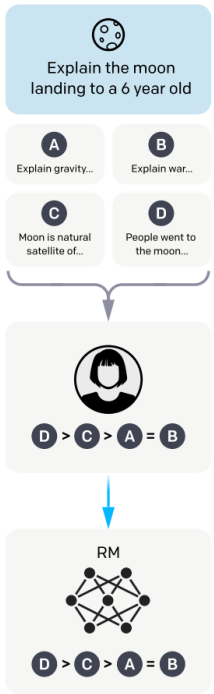
Step 2

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

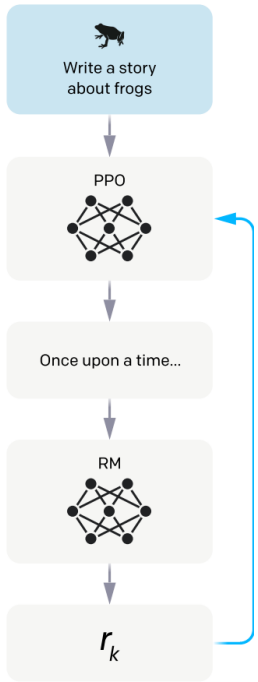
Optimize a policy against the reward model using reinforcement learning.

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]

Learning More

- 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

Class Structure

- Last time: Policy search continued and Imitation Learning
- This time: Imitation Learning and RLHF
- Next time: Direct Preference Optimization