Lecture 8: Imitation Learning and RLHF

Emma Brunskill

CS234 Reinforcement Learning.

Spring 2024

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
b. Behavior cloning requires knowing the dynamics model F
c. DAGGER uses demonstrations from experts but no further interactions F
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?

EM **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, "I
    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

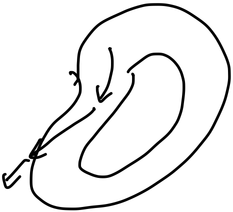
Today

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

Learning from Demonstrations

- Expert provides a set of **demonstration trajectories**: sequences of states and actions
- Imitation learning is useful when it is easier for the expert to demonstrate the desired behavior rather than:
  - Specifying a reward that would generate such behavior,
  - Specifying the desired policy directly

Problem Setup



- Input:
  - State space, action space
  - Transition model $P(s' \mid s, a)$ (sometimes)
  - 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^*)
- Behavioral Cloning:
  - Can we directly learn the expert's policy using supervised learning?

DAGGER

- Inverse RL:
  - Can we recover R?
- Apprenticeship learning via Inverse RL:
  - Can we use R to generate a good policy?

Recall: Feature Matching

$\pi \longrightarrow \tau \longrightarrow (s,a)$ distrib

- 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

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

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

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

- where here μ is used to represent the features experienced under π

Recall: 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

$\sum_s p(s) = 1$ prob distrib

- 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:

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

- 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

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

expert

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

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

- 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. if we knew the costs, what would be the form of the distrib over τ?

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

$$\frac{d\mathcal{L}}{dp(\tau)} = \log p(\tau) + p(\tau)\cdot\frac{1}{p(\tau)} - \lambda_1 r_\phi(\tau) + \lambda_0 = 0$$

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

$$p(\tau) = e^{-1+\lambda_1 r_\phi(\tau) - \lambda_0}$$

$$p(\tau) \propto e^{r_\phi(\tau)}$$

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

$$p(\tau_i \mid \phi) = \frac{1}{Z(\phi)} e^{r_\phi(\tau_i)}$$

$$Z(\phi) = \sum_\tau e^{r_\phi(\tau_i)}$$

$$p(\tau_i \mid \phi) = \frac{1}{Z(\phi)} e^{\sum_{s_i \in \tau_i} r_\phi(s_i)}$$

we don't know what r_ϕ is

constrained

this observation (prob over τ that max entropy looks like a normalized expon)

means we can now estimate/learn r_ϕ by max prob of our observed data

$$\max_\phi \log \underbrace{\prod_{\tau^* \in D}}_{\text{expert}} p(\tau^*)$$

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

$$= \max \sum_{\tau^* \in D} r_\phi(\tau^*) - \underbrace{\log \sum_\tau e^{r_\phi(\tau)}}$$

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

$$\nabla_\phi J(\phi) = \sum_{\tau^* \in D} \frac{d r_\phi(\tau^*)}{d\phi} - |D| \underbrace{\frac{1}{\sum_\tau e^{r_\phi(\tau)}} \sum_\tau e^{r_\phi(\tau)}} \frac{d r_\phi(\tau)}{d\phi}$$

Note $\Rightarrow p(\tau|\phi) = \dfrac{e^{r_\phi(\tau)}}{Z(\phi)}$

$$= \sum_{\tau^* \in D} \frac{d r_\phi(\tau^*)}{d\phi} - |D| \sum_\tau p(\tau|\phi) \frac{d r_\phi(\tau)}{d\phi}$$

$$p(\tau) = p(s_1) \prod_{t=1}^{T} \underbrace{p(a|s)}_{\pi} \, p(s_{t+1}|s_t, a_t)$$

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

$$\nabla_\theta J(\theta) = \sum_{s \in \tau^* \in D} \frac{d r_\phi(s)}{d\theta} - |D| \sum_s p(s|\theta, \tau) \frac{d r_\phi(s)}{d\theta}$$

State Densities

if tabular & dynamics known & π is given

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

for $t = 1 \ldots T$ (time indexed)

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') p(a|s') p(s|s', a)$$

$$p(s|\phi, T) = \sum_t \mu_t(s)$$

$$\nabla_\phi J(\phi) = \sum_{s_t \in \tau^* \in D} \frac{d r_\phi(s)}{d\phi} - |D| \sum_s p(s|\phi, T) \frac{d r_\phi(s)}{d\phi}$$

if $r_\phi(s) = \phi f_s$ (linear)

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

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 iterations
4. Compute state visitation frequencies $p(s|\xi\phi, T)$ (linear)
5. Compute gradient on reward model

$$\nabla J(\phi) = \frac{1}{M} \sum_{\tau_d \in \mathcal{D}} f_{\tau_d} - \sum_s p(s|\phi, T) f_s \quad (6)$$

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

$$\nabla J(\phi) = \frac{1}{M} \sum_{\tau_d \in \mathcal{D}} f_s - \sum_s p(s|\phi, \tau) f_s \tag{7}$$

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 T
- (2) Computing the state visitation frequencies T
- (3) Computing the gradient NO
- (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

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

DAGGER/ Constant Teaching

Demonstrations Only

$\longleftarrow$ Human Effort $\longrightarrow$

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

DAGGER/ Constant Teaching | Pairwise Labels | Demonstrations Only

$\longleftarrow$ Human Effort $\longrightarrow$

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 ...

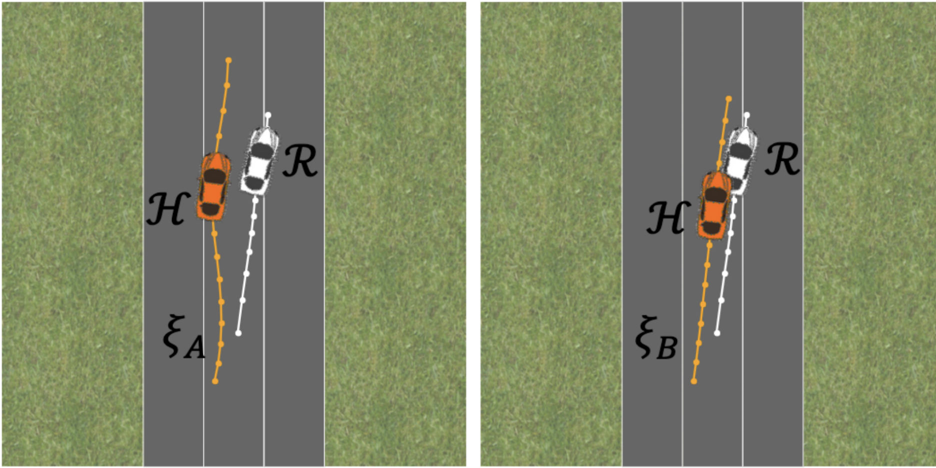
CS159: Introduction to Parallel Processing | People | San Jo.
www.sjsu.edu › ... › Chun, Robert K › Courses ▾ San Jose State University ▾
Jan 20, 2015 - Description. A combination hardware architecture and software development class focused on multi-threaded, parallel processing algorithms ...

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. An early model of parallel computation... Home Courses.

Slide from Yisong Yue
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

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

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

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

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

[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 $b_1, b_2, \dots b_k$. No state/context.
- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

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

- 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, $\tau^1(s_0, a_7, s_{14}, \dots)$ 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 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

- 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

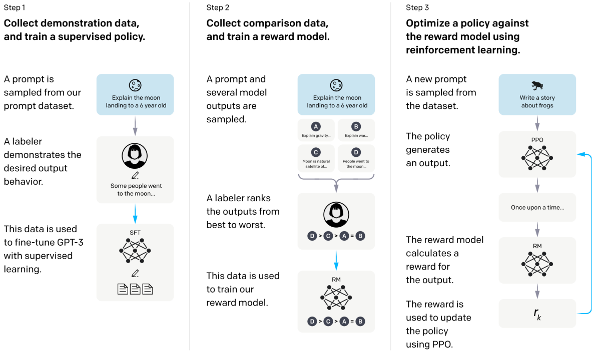
Christiano et al. 2017. Deep RL from Human Preferences https://arxiv.org/pdf/1706.03741.pdf

From Backflips to ChatGPT. [4]

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

- 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??)