Lecture 8: Imitation Learning and RLHF

Emma Brunskill

CS234 Reinforcement Learning.

Spring 2024

Select all that are true

1. Maximizing a lower bound on the performance gap between a new policy and and old policy can ensure monotonic convergence
2. Behavior cloning requires knowing the dynamics model
3. DAGGER uses demonstrations from experts but no further interactions
4. 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?
- Inverse RL:
  - Can we recover R?
- Apprenticeship learning via Inverse RL:
  - Can we use R to generate a good policy?

Recall: 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):

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

- 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

- 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

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

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

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

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

From Trajectories to States

State Densities

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

$$\nabla J(\phi) = \tag{6}$$

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

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

DAGGER/ Constant Teaching

Demonstrations Only

⟵ Human Effort ⟶

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 ...

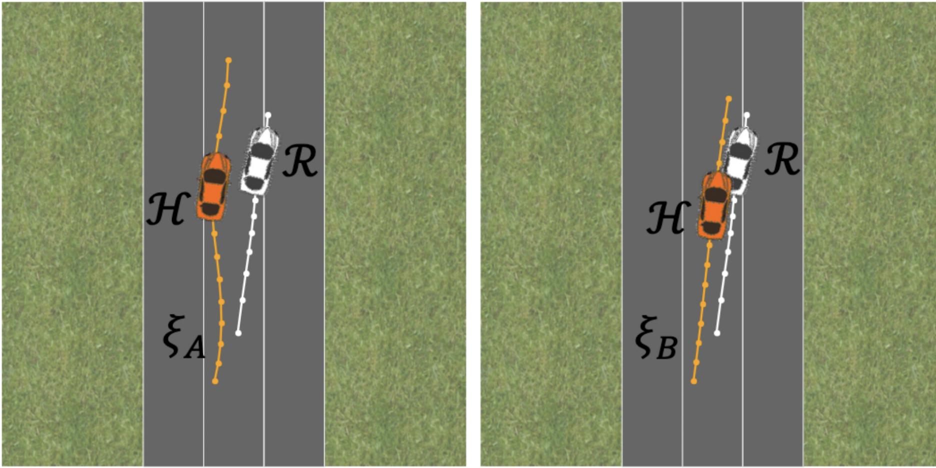
CS 159: Introduction to Parallel Processing - Info.sjsu.edu
info.sjsu.edu › ... › Courses ▾ San Jose State University ▾
CS 159. Introduction to Parallel Processing. Description Major parallel architectures: shared memory, distributed memory, SIMD, MIMD. Parallel algorithms: ...

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. Ar 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 $b_1, b_2, \dots b_k$. 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 $b_1, b_2, \ldots b_k$. 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, $\tau^1(s_0, a_7, s_{14}, \dots)$ 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

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

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

- 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

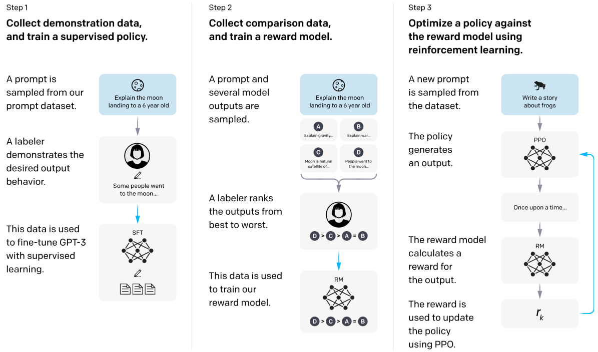
From Backflips to ChatGPT. [4]

[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

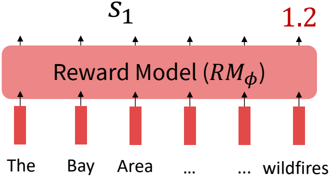


- First step: instruction tuning!
- Second + third steps: maximize reward (but how??)

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_1		s_3		s_2



Bradley-Terry [1952] paired comparison model

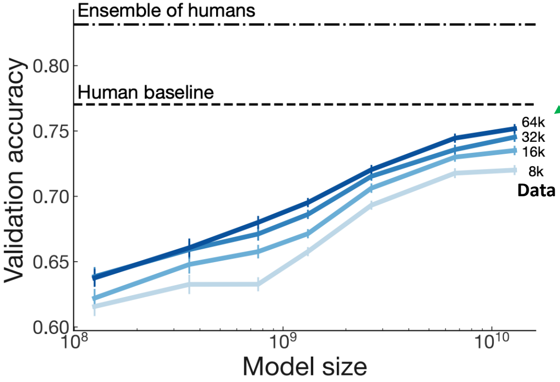
$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l) \sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

"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]

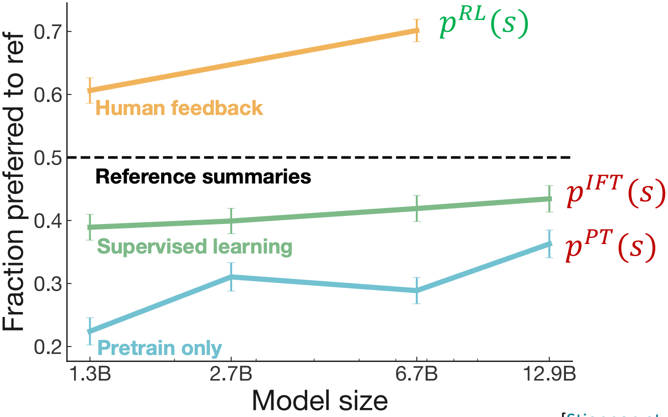
- Finally, we have everything we need:
  - A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
  - A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
  - A method for optimizing LM parameters towards an arbitrary reward function.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

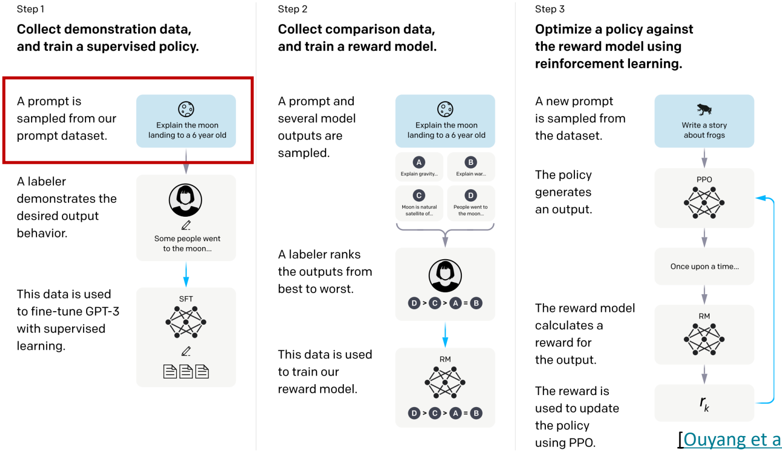
RLHF provides gains over pretraining + finetuning



[Stiennon et al., 2020]

InstructGPT: scaling up RLHF to tens of thousands of tasks

30k tasks!



[Ouyang et al., 2022]

InstructGPT: scaling up RLHF to tens of thousands of tasks

Tasks collected from labelers:

- **Plain:** We simply ask the labelers to come up with an arbitrary task, while ensuring the tasks had sufficient diversity.
- **Few-shot:** We ask the labelers to come up with an instruction, and multiple query/response pairs for that instruction.
- **User-based:** We had a number of use-cases stated in waitlist applications to the OpenAI API. We asked labelers to come up with prompts corresponding to these use cases.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.

[Ouyang et al., 2022]

InstructGPT

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and
sent them back to the earth so we could all see them.
```

InstructGPT

PROMPT *Write a short poem about a wise frog.*

COMPLETION

GPT-3

```
Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you
have lost.

Write a short story in which a character has a bad dream.
```

InstructGPT

```
The frog is a symbol of wisdom
He knows all the secrets of the world
He is a master of disguise
And a great teacher of life
He is a symbol of transformation
And the bringer of change
He is the frog who has seen it all
And knows the meaning of it all
```

ChatGPT: Instruction Finetuning + RLHF for dialog agents

ChatGPT: Optimizing Language Models for Dialogue

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge…

Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

(Instruction finetuning!)

https://openai.com/blog/chatgpt/

ChatGPT: Instruction Finetuning + RLHF for dialog agents

ChatGPT: Optimizing Language Models for Dialogue

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge…

Methods

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using Proximal Policy Optimization. We performed several iterations of this process.

(RLHF!)

https://openai.com/blog/chatgpt/

ChatGPT: Instruction Finetuning + RLHF for dialog agents

MU

Explain in a few sentences what methods were used in training ChatGPT to the audience of CS224n, Stanford's Deep Learning for Natural Language processing course. A lot of the students are very Gen Z, so be sure to use a ton of emojis to seem hip and cool.

Hey CS224n peeps! 😎 ChatGPT here to spill the tea on how I was trained to be a savvy language model! 🤓

So basically, I was pre-trained using a combination of unsupervised learning techniques, specifically a transformer-based language model architecture, that allowed me to learn patterns and relationships within large amounts of text data. 🤖📚

The training data came from various sources such as books, articles, and websites, and I was fine-tuned on a variety of downstream NLP tasks, which involved supervised learning using labeled datasets to further improve my performance. 🤯

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]

RLHF behaviors – clear stylistic changes

Instruction: What are the five most common causes of stress among people?

Alpaca (Finetuned w/ 10k data): The five most common causes of stress among people are work, money, relationships, health, and family

Alpaca + PPO: The five most common causes of stress among people are:

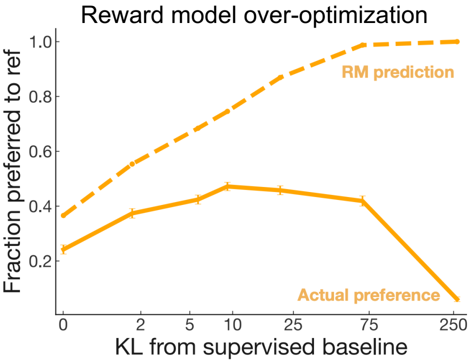
1. Pressure at work: Whether it is due to tight deadlines, high expectations, or an overwhelming workload, pressure at work can be a major source of stress
2. Money problems: Financial issues can be a significant source of stress, from not being able to make ends meet to struggling with debt.
3. Unhealthy lifestyle: Poor diet, lack of exercise, and sleep deprivation can all contribute to stress.
4. Unhappiness at home: Problems in relationships, conflicts with family members, and other domestic issues can be a major source of stress.
5. Uncertainty: Excessive worry and anxiety about the future, health issues, or other unknowns can be a major cause of stress

- Significantly more detailed, nicer/clearer list like formatting

[Dubois et al 2023]

Limitations of RL + Reward Modeling

- Human preferences are unreliable!
  - ”Reward hacking” is a common problem in RL
  - Chatbots are rewarded to produce responses that *seem* authoritative and helpful, *regardless of truth*
  - This can result in making up facts + hallucinations
- **Models** of human preferences are *even more* unreliable!



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

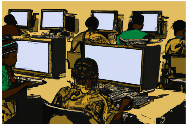
[Stiennon et al., 2020]

Where does the labels come from?

BUSINESS • TECHNOLOGY

Exclusive: OpenAI Used Kenyan Workers on Less Than $2 Per Hour to Make ChatGPT Less Toxic

15 MINUTE READ



NIAMH ROWE BUSINESS 15.10.2023 08:00 AM

Millions of Workers Are Training AI Models for Pennies

From the Philippines to Colombia, low-paid workers label training data for AI models used by the likes of Amazon, Facebook, Google, and Microsoft.

Oskarina Vero Fuentes with her dog. COURTESY OF OSKARINA VERO FUENTES

Behind the AI boom, an army of overseas workers in 'digital sweatshops'

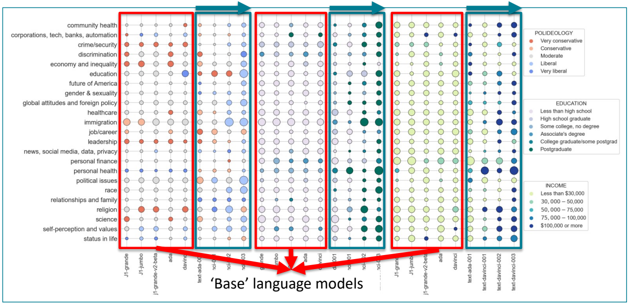
By Rebecca Tan and Regine Cabato
August 28, 2023 at 2:00 a.m. EDT

- RLHF labels are often obtained from overseas, low-wage workers

Where does the label come from?

Table 12: Labeler demographic data

What gender do you identify as?	
Male	50.0%
Female	44.4%
Nonbinary / other	5.6%
What ethnicities do you identify as?	
White / Caucasian	31.6%
Southeast Asian	52.6%
Indigenous / Native American / Alaskan Native	0.0%
East Asian	5.3%
Middle Eastern	0.0%
Latinx	15.8%
Black / of African descent	10.5%
What is your nationality?	
Filipino	22%
Bangladeshi	22%
American	17%
Albanian	5%
Brazilian	5%
Canadian	5%
Colombian	5%
Indian	5%
Uruguayan	5%
Zimbabwean	5%
What is your age?	
18-24	26.3%
25-34	47.4%
35-44	10.5%
45-54	10.5%
55-64	5.3%
65+	0%
What is your highest attained level of education?	
Less than high school degree	0%
High school degree	10.5%
Undergraduate degree	52.6%
Master's degree	36.8%
Doctorate degree	0%



[Santurkar+ 2023, OpinionQA]

- We also need to be quite careful about how annotator biases might creep into LMs

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: Author of Direct Preference Optimization (best paper runner up at top ML conference) guest lecture