Conservative Offline RL and Reward Learning

CS 224R

Course reminders

Project

- Fill out AWS form for GPU quota
- Proposal due today

  graded fairly lightly- it's for your benefit!

Homework

- Homework 2 due next Friday. (start early!!)

Guest lectures next week!

Archit Sharma

Researcher on Gemini team

Lead author of DPO

Preference optimization for LLMs

Aviral Kumar

Prof at CMU

At forefront of LLM reasoning

RL for LLM reasoning

Both very useful for default project. **Please join lecture in person!**

The plan for today

1. Offline reinforcement learning: recap & conservative methods — **Part of HW3**
2. Reward learning:
   a. Where do rewards come from?
   b. Learning rewards from example goals, behaviors — **Part of default project**
   c. Learning rewards from human preferences

Key learning goals:

- why task specification is hard (& why naïve methods fail)
- methods for learning rewards from human supervision

Recap: Offline RL

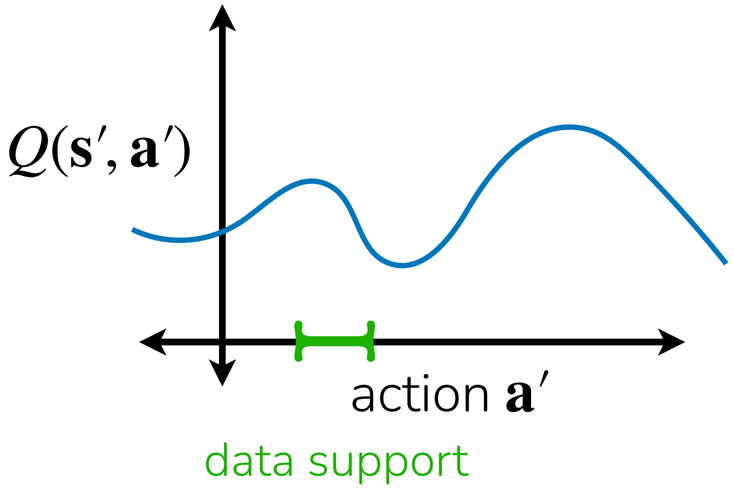
Problem: Can we learn a policy from a batch of existing data, without online data collection?

Batch of data collected by unknown behavior policy π_β, want to maximize rewards under π_θ

Recall: Off-policy critic objective $\min_\phi \sum_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \| \hat{Q}^{\pi_\theta}_\phi(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi_\theta(\cdot|\mathbf{s}')}[\hat{Q}^{\pi_\theta}_\phi(\mathbf{s}',\mathbf{a}')] \right) \|^2$

Querying Q-function on OOD actions causes overestimation.

Randomly init. Q-function for state $\mathbf{s}'$



data support

IQL avoids queries beyond dataset actions altogether.

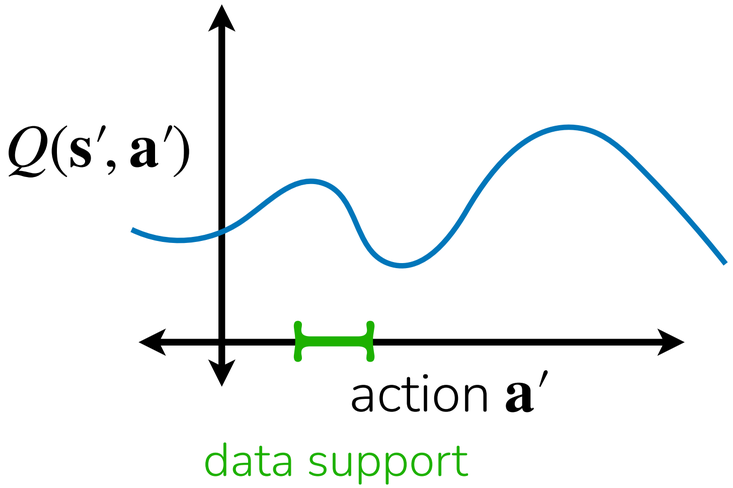
Fit V^π for policy π better than π_β using expectile loss:

$$\hat{V}(\mathbf{s}) \leftarrow \arg\min_V E_{(\mathbf{s},\mathbf{a})\sim D} \left[\ell_2^\lambda \left(V(\mathbf{s}) - \hat{Q}(\mathbf{s},\mathbf{a}) \right) \right]$$

Fit Q using V: $\hat{Q}(\mathbf{s},\mathbf{a}) \leftarrow \arg\min_Q E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D} \left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r + \gamma \hat{V}(\mathbf{s}') \right) \right)^2 \right]$

How to mitigate overestimation in offline RL?

Recall: Randomly init. Q-function for state $\mathbf{s}'$



Can we discourage overestimation?
while still estimating value of the learned policy

What if we just push down on large Q-values?

$$\hat{Q}^\pi = \arg\min_Q \max_\mu E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{a}'\sim\pi}[Q(\mathbf{s}',\mathbf{a}')]\right)\right)^2\right] + \alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]$$

standard critic update — push down on large Q-values

Can show that $\hat{Q}^\pi \leq Q^\pi$ for large enough α

Slide adapted from Sergey Levine

How to mitigate overestimation in offline RL?

Can we discourage overestimation?

while still estimating value of the learned policy

standard critic update

push down on large Q-values

$$\hat{Q}^\pi = \arg\min_Q \max_\mu E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{a}'\sim\pi}[Q(\mathbf{s}',\mathbf{a}')]\right)\right)^2\right] + \alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})]$$

push up on Q-values for $(\mathbf{s},\mathbf{a})$ in the data

No longer guaranteed that $\hat{Q}^\pi \leq Q^\pi$ for all $(\mathbf{s},\mathbf{a})$.

BUT, guaranteed that $E_{\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})] \leq E_{\pi(\mathbf{a}|\mathbf{s})}[Q^\pi(\mathbf{s},\mathbf{a})]$ for all $\mathbf{s} \in D$.

Conservative Q-learning (CQL)

Slide adapted from Sergey Levine

How to mitigate overestimation in offline RL?

Conservative Q-learning (CQL)

Full algorithm

1. Update $\hat{Q}^\pi$ using L_{CQL} using D
2. Update policy π

If actions are discrete: $\pi(\mathbf{a}|\mathbf{s}) = \begin{cases} 1 \text{ if } \mathbf{a} = \arg\max_{\bar{\mathbf{a}}} \hat{Q}^\pi(\mathbf{s}, \bar{\mathbf{a}}) \\ 0 \text{ otherwise} \end{cases}$

If actions are continuous: $\theta \leftarrow \theta + \eta \nabla_\theta E_{\mathbf{s}\sim D, \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})} \left[\hat{Q}^\pi(\mathbf{s}, \mathbf{a}) \right]$

Slide adapted from Sergey Levine

How to mitigate overestimation in offline RL?

Conservative Q-learning (CQL)

1. Update $\hat{Q}^\pi$ using L_{CQL} using D
2. Update policy π

How compute objective L_{CQL}?

$$\hat{Q}^\pi = \arg\min_Q \max_\mu E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma E_\pi[Q(\mathbf{s}',\mathbf{a}')]\right)\right)^2\right] + \alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})] + R(\mu)$$

regularizer

Common choice: $R(\mu) = E_{\mathbf{s}\sim D}[\mathcal{H}(\mu(\cdot|\mathbf{s})]$

With max entropy regularizer R, optimal $\mu(\mathbf{a}\,|\,\mathbf{s}) \propto \exp\left(Q(\mathbf{s},\mathbf{a})\right)$

Then: $E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] = \log \sum_{\mathbf{a}} \exp\left(Q(\mathbf{s},\mathbf{a})\right)$

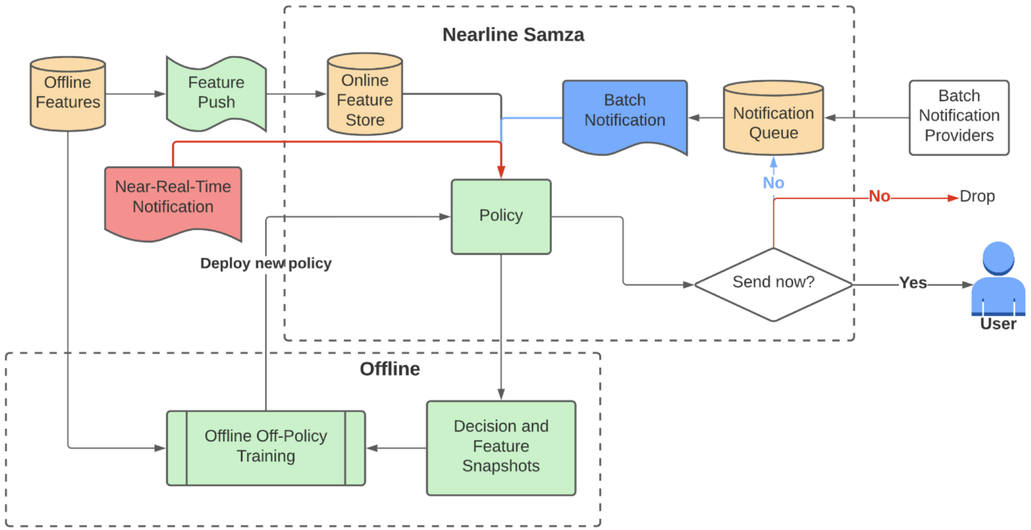
Don't need to construct μ directly.

You will implement CQL in homework 3!

Slide adapted from Sergey Levine

An example application

Optimizing policy for sending notifications to users on LinkedIn



WAU: weekly active users
Volume: total # of notifications

CTR: click-through-rate of notifications

Metric	DDQN vs. Baseline	DDQN + CQL vs. Baseline
Sessions	not stat sig	+ 0.24%
WAU	-0.69%	+ 0.18%
Volume	+7.72%	-1.73%
CTR	-7.79%	+2.26%

Table 1: Online A/B test results for DDQN with and without CQL

Prabhakar, Yuan, Yang, Sun, Muralidharan. Multi-Objective Optimization of Notifications Using Offline RL. ‘22

Summary

Why offline RL? Online data is expensive. *Reusing offline data* is good!

Key challenge: Overestimating Q-values because of shift between π_β and π_θ

Approaches:

1. filtered or weighted imitation learning is a simple baseline
2. **implicitly constrain** the policy to π_β by only supervising on actions in data
3. construct conservative objective by **penalizing Q-values**

Trajectory stitching allows offline RL methods to improve over imitation.

The plan for today

1. Offline reinforcement learning: recap & conservative methods
2. **Reward learning**:
   a. Where do rewards come from?
   b. Learning rewards from example goals, behaviors
   c. Learning rewards from human preferences

Where does the reward come from?

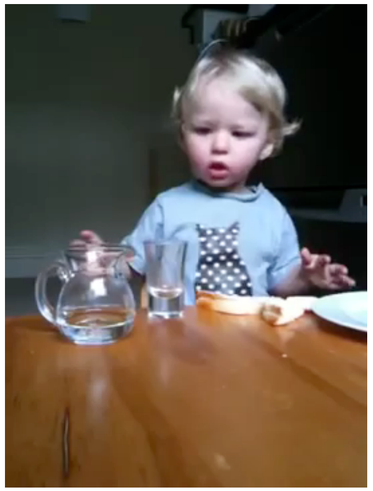
Computer Games

reward

Mnih et al. ‘15

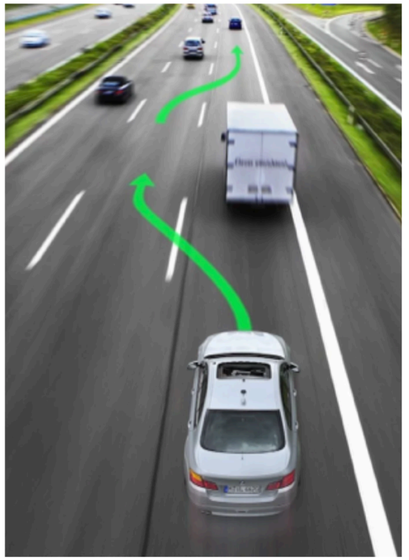
Real World Scenarios

robotics



dialog

autonomous driving



what is the reward?

often use a proxy

Are there other easier ways of providing task supervision?

We’ve seen one alternative approach so far.

Where does the reward come from?

Direct imitation learning: Mimic actions of expert

- but no reasoning about outcomes or dynamics
- the expert might have different degrees of freedom
- might not be possible to provide demonstrations

Can we reason about what
an expert is trying to achieve?

The plan for today

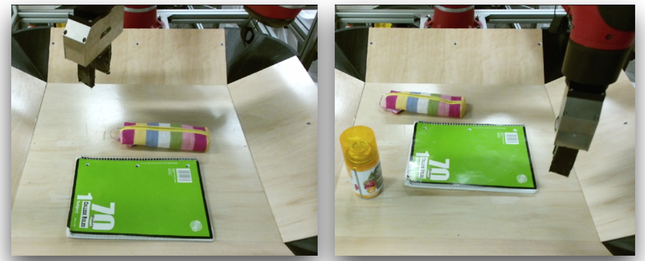
1. Offline reinforcement learning: recap & conservative methods
2. Reward learning:
   a. Where do rewards come from?
   b. Learning rewards from example goals, behaviors
   c. Learning rewards from human preferences

Goal classifiers

Key idea: Learn to discern goal states from other states

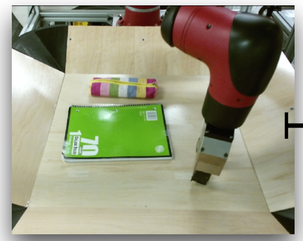
Example task: put pencil case behind notebook

Positive examples



Negative examples

Trained binary classifier



$\mapsto 1$

$\mapsto 0$

Use output as reward signal.

1. Collect examples of successful & unsuccessful states
   (states inside and outside of goal set G)
2. Train binary classifier
   (with inputs s_i and labels $1(s_i \in G)$)
3. Run RL with classifier as reward.

Goal classifiers

Key idea: Learn to discern goal states from other states

1. Collect examples of successful & unsuccessful states
   (states inside and outside of goal set G)
2. Train binary classifier
   (with inputs s_i and labels $1(s_i \in G)$)
3. Run RL with classifier as reward.

What can go wrong?

The RL algorithm will seek out states that the classifier thinks are good.

It may simply find states that the classifier wasn't trained on!

—> exploiting the classifiers weaknesses

Goal classifiers

Can we prevent the RL algorithm from exploiting the classifier's weaknesses?

A proposition: Add states that RL visits as negative examples for the classifier

Specifically:

1. Collect initial set of successful states D_+ and unsuccessful states D_-.
2. Update classifier using D_+ and D_- (balance dataset 50/50)
3. Collect experience $s_t, a_t, \ldots$ using policy π
4. Update policy π using classifier-based reward.
5. Add visited states to negatives: $D_- \leftarrow D_- \cup \{s_t\}$

Why might this work or not work?

Will it learn an accurate classifier?

Do we expect the policy to work?

What will the classifier output for successful states?

Goal classifiers

A proposition: Add states that RL visits as negative examples for the classifier

Specifically:

1. Collect initial set of successful states D_+ and unsuccessful states D_-.
2. Update classifier using D_+ and D_- (balance dataset 50/50)
3. Collect experience $s_t, a_t, \ldots$ using policy π
4. Update policy π using classifier-based reward.
5. Add visited states to negatives: $D_- \leftarrow D_- \cup \{s_t\}$

Why might this work or not work?

Classifier can't be exploited. 😌

But, what if some of the visited states *are* successful? 😳

As long as batches are balanced, classifier will output p≥0.5 for successful states. 😅

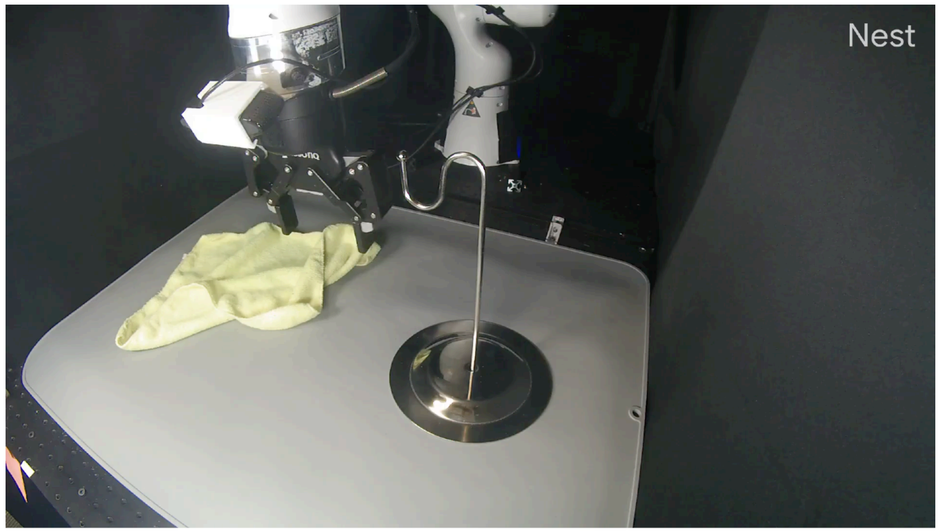
Goal classifiers for robotic reinforcement learning

Collect 50 demonstrations:

- use final states as success state examples
- initialize RL replay buffer with demos

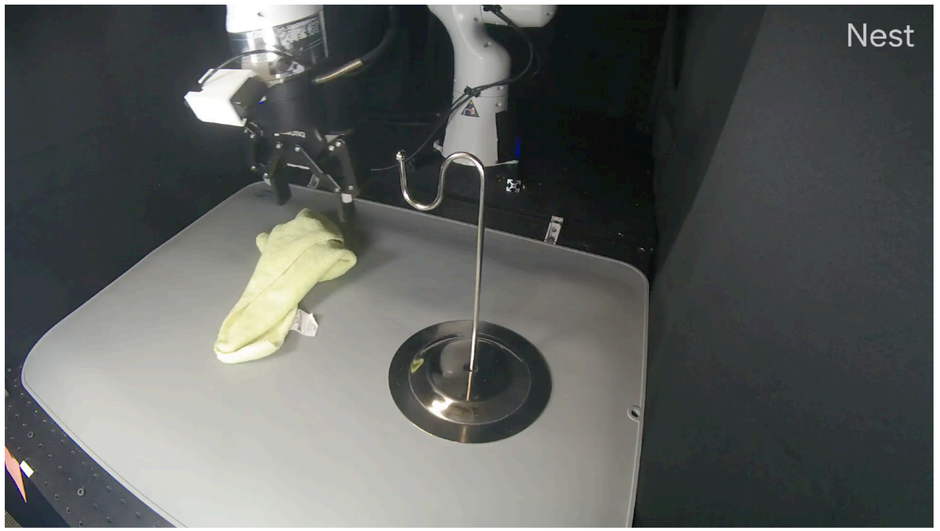
Important to regularize the classifier!

Directly imitating demos:



26% success rate

RL policy trained with learned classifier:



62% success rate

Sharma, Ahmed, Ahmad, Finn. Self-Improving Robots: End-to-End Autonomous Visuomotor Reinforcement Learning. ‘23

Aside: Generative adversarial networks

This is also how GANs work!

1. Train classifier to discriminate between real data and generated data
2. Train generator to generate data that the classifier thinks is real.

"Yorkshire terrier" generated by VQ-GAN

Yu et al. ViT-VQGAN. ICLR '22.

Phenaki video generation (uses GAN-loss)

Side view of an astronaut is walking through a puddle on mars.
The astronaut is dancing on mars.
The astronaut walks his dog on mars.
The astronaut and his dog watch fireworks.

Villegas et al. Phenaki. ICLR '23.

At convergence: generator will match data distribution p(x)

Recap of reward classifiers

- Pre-trained classifiers can be exploited when optimized against.
- Solution: Update the classifier *during RL*, using policy data as negatives
- Can learn goal classifier with success examples, full reward with demos.

+ practical framework for task specification

~ adversarial training can be unstable

(though variety of regularization tricks from GAN literature)

- requires examples of desired behavior or outcomes

The plan for today

1. Offline reinforcement learning: recap & conservative methods
2. Reward learning:
   a. Where do rewards come from?
   b. Learning rewards from example goals, behaviors
   c. Learning rewards from human preferences

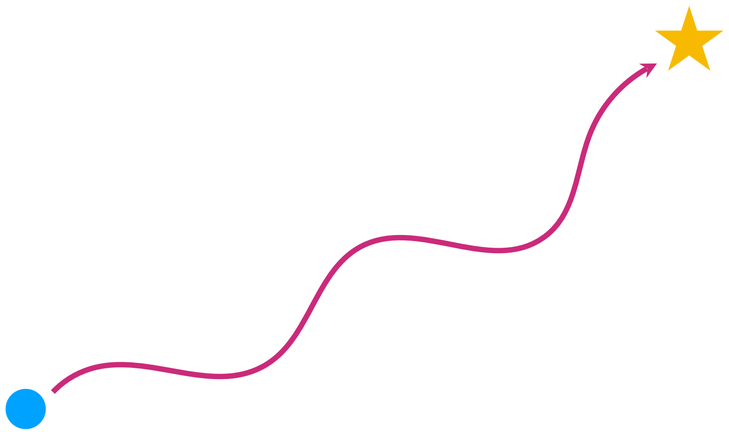
Can humans provide feedback on policy roll-outs?

instead of requiring demos or example goals

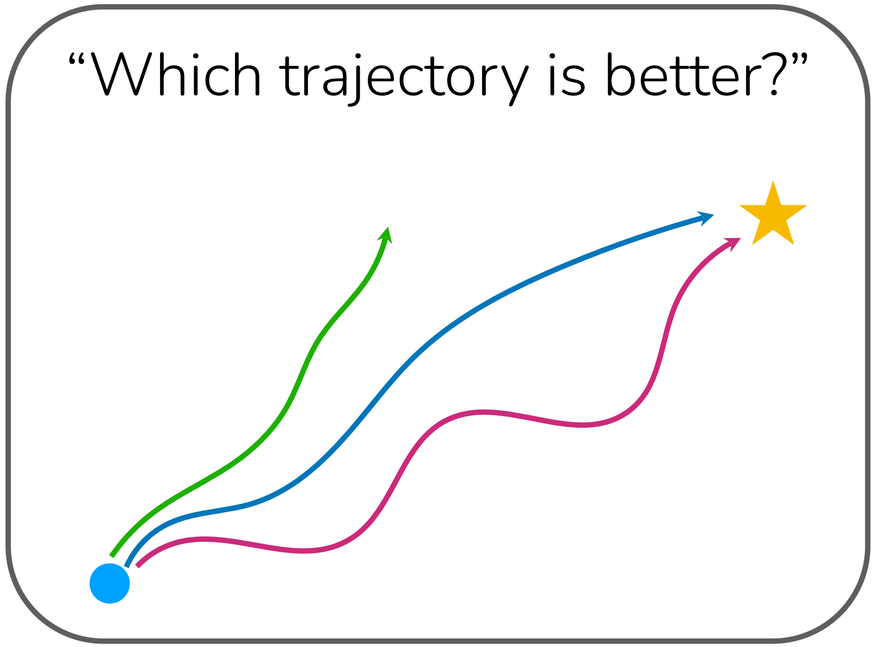
(or in addition to!)

A couple options

“How good is this trajectory?”



“Which trajectory is better?”



Relative **preferences** are easier to provide!

How to learn a reward function from human preferences?

Human says τ_w is better than τ_l. (can write it as $\tau_w \succ \tau_l$)

Note: τ could be a *full* or *partial* roll-out!

We want a reward r_θ such that $\underbrace{\sum_{(s,a)\in\tau_w} r_\theta(s,a)}_{\text{shorthand } r_\theta(\tau_w)} > \sum_{(s,a)\in\tau_l} r_\theta(s,a).$

Humans are classifying which trajectory is better.

Reward should be discriminative as well.

How to learn a reward function from human preferences?

Human says τ_w is better than τ_l. (can write it as $\tau_w \succ \tau_l$)

Note: τ could be a *full* or *partial* roll-out!

We want a reward r_θ such that $\underbrace{\sum_{(s,a)\in\tau_w} r_\theta(s,a)}_{\text{shorthand } r_\theta(\tau_w)} > \sum_{(s,a)\in\tau_l} r_\theta(s,a).$

We can define $\sigma\big(r_\theta(\tau_a) - r_\theta(\tau_b)\big)$ to be the estimated probability that $\tau_a \succ \tau_b$.

Then, we can maximize log probability: $\max_\theta \mathbb{E}_{\tau_w,\tau_l}\Big[\log\sigma\big(r_\theta(\tau_w) - r_\theta(\tau_l)\big)\Big]$

How to learn a reward function from human preferences?

Complete reward learning algorithm

1. Given dataset $\{\tau_i\}$, sample batches of k trajectories and ask humans to rank.

   (for LLMs, these k trajectories all have the same prompt)

2. Compute $r_\theta(\tau_1), \ldots, r_\theta(\tau_k)$ under current reward model r_θ

3. For all $\binom{k}{2}$ pairs per batch, compute $\nabla_\theta \mathbb{E}_{\tau_w, \tau_l}\left[\log \sigma\left(r_\theta(\tau_w) - r_\theta(\tau_l)\right)\right]$ where $\tau_w \succ \tau_l$

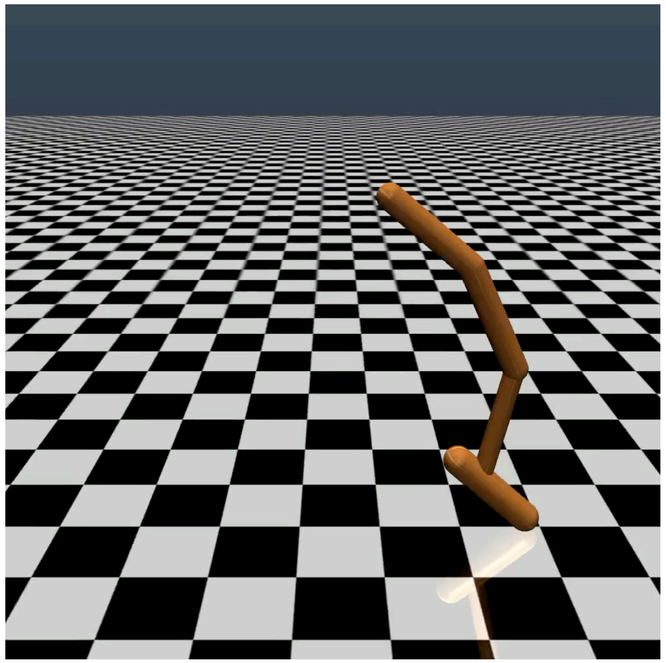
4. Update θ using computed gradient.

Note: This can be done in the loop of online RL.

Learning rewards from human feedback

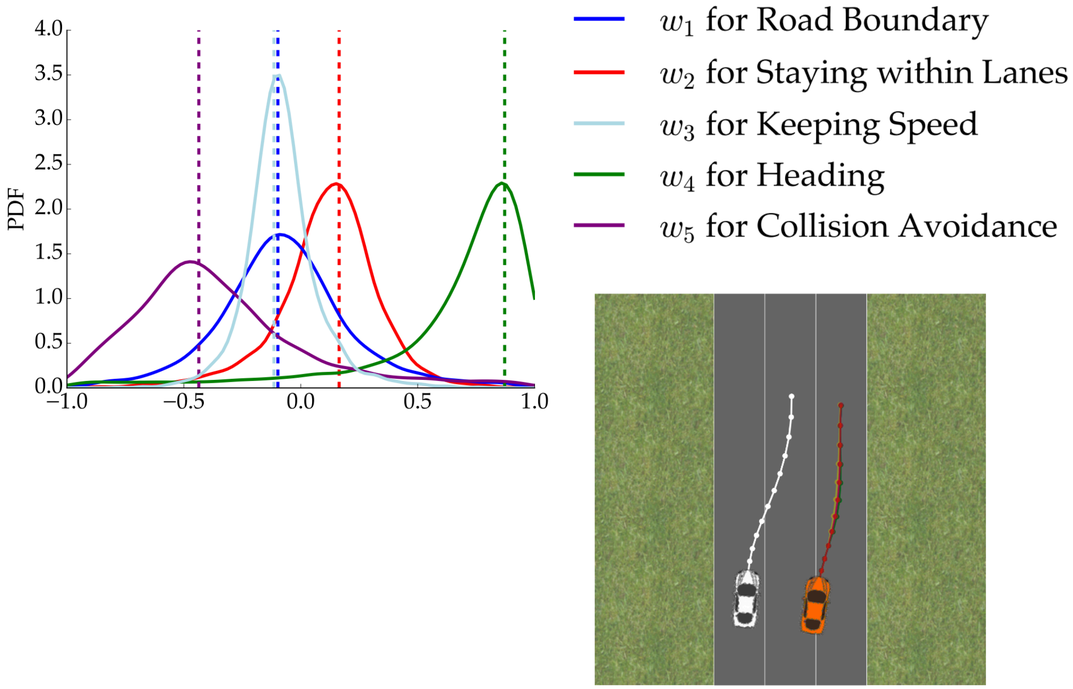
Learning rewards in the loop of online RL

uses 900 human preference queries



Learning rewards for driving

to weight different factors



Christiano et al. Deep Reinforcement Learning from Human Preferences. NeurIPS 2017.

Sadigh et al. Active Preference-Based Learning of Reward Functions. RSS 2017.

Learning rewards from human feedback for LLMs

Prompt x

x :

How do I cook an omelette?

Sample replies

Cooking an omelette is quick and easy! Here's a simple step-by-step guide for a classic 2-egg omelette:

Ingredients:

- 2 eggs
- Salt and pepper (to taste)
- 1–2 tbsp milk or water (optional, for fluffiness)
- 1 tsp butter or oil
- Optional fillings: cheese, chopped veggies, ham, herbs, etc.

Instructions:

1. **Crack the eggs** into a bowl. Add a pinch of salt and pepper. Whisk them well with a fork or whisk until the yolks and whites are fully combined. Add milk or water if using.
2. **Heat a non-stick skillet** over medium heat. Add the butter or oil and swirl it around to coat the pan.
3. **Pour in the eggs** and let them sit for a few seconds. Then gently stir with a spatula, pulling the cooked edges toward the center while letting the uncooked egg flow to the sides.
4. **Let it set** once the eggs are mostly cooked but still slightly soft on top. Stop stirring and let it cook undisturbed for 30 seconds to 1 minute.
5. **Add your fillings** to one half of the omelette.
6. **Fold it** in half using a spatula, covering the fillings. Cook for another 30 seconds if you want the cheese to melt more.
7. **Slide it onto a plate** and serve hot!

Sure! To cook an omelet, start by whisking a couple of eggs with a pinch of salt and pepper. Heat a nonstick skillet over medium heat and add a bit of butter or oil. Pour in the eggs and let them sit undisturbed for a few seconds, then gently stir with a spatula while tilting the pan to let uncooked egg flow underneath. Once the eggs are mostly set but still slightly soft on top, add your desired fillings like cheese, vegetables, or ham. Fold the omelet in half, cook for another 30 seconds, then slide it onto a plate and serve hot.

Ask human which response is better

$y \succ y'$

Train reward model $r(x, y)$ that judges how good a response y is for a prompt x.

Learning rewards from human feedback for LLMs

1. Large-scale pre-training

next token prediction, mixed quality

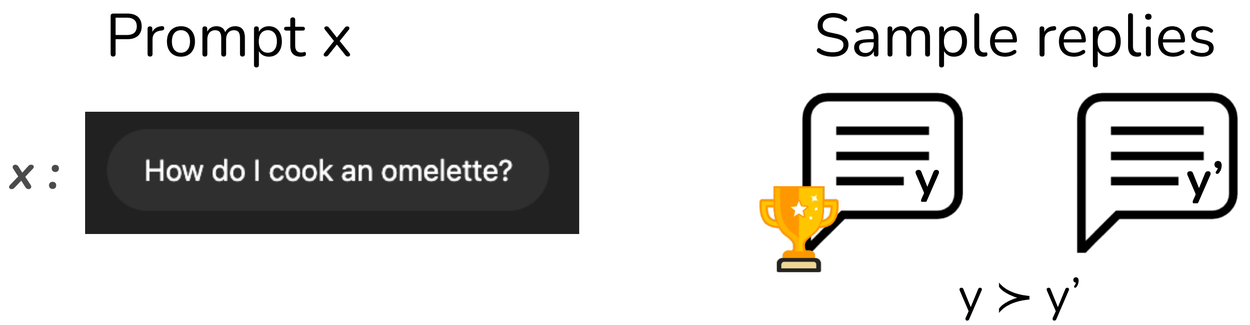
2. Supervised fine-tuning

on higher-quality (prompt, response)

3. RL from human feedback

on higher-quality (prompt, response)

3a. Gather preference data.



3b. Train reward model.

Train reward model $r(x, y)$ that judges how good a response y is for a prompt x.

3c. Reinforcement learning to maximize reward.

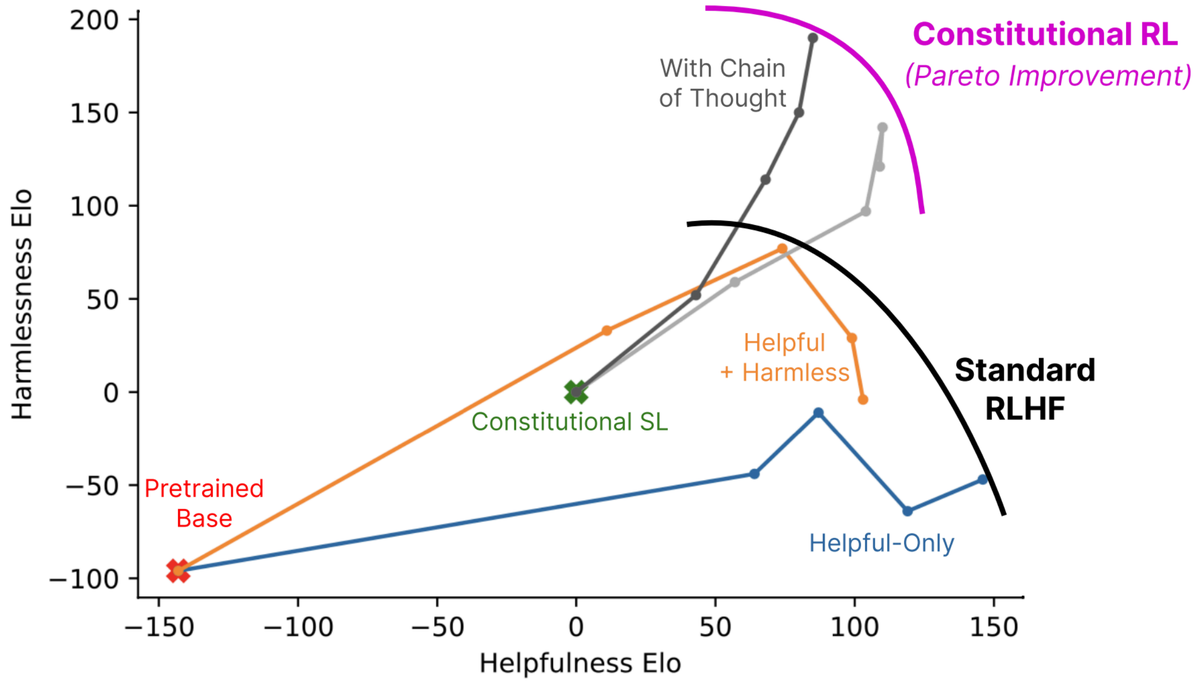
Finetune model to predict y for prompt x with high reward

Learning rewards from ~~human~~ AI feedback

Reinforcement learning with AI feedback (RLAIF)

Ask another language model “which of these responses is less harmful?”

Key insight: critique is easier than generation!



Bai et al. Constitutional AI: Harmlessness from AI Feedback. 2022.

Summary of Reward Learning

#1 Takeaway: Rewards can't be taken for granted!

Learning rewards from goals, demos

+ practical framework for task specification

~ adversarial training can be unstable
(though variety of regularization tricks from GAN literature)

- requires examples of desired behavior or outcomes

Learning rewards from human preferences

+ pairwise preferences easy to provide
(doesn't require example goals, demos!)

+ has been deployed at scale!

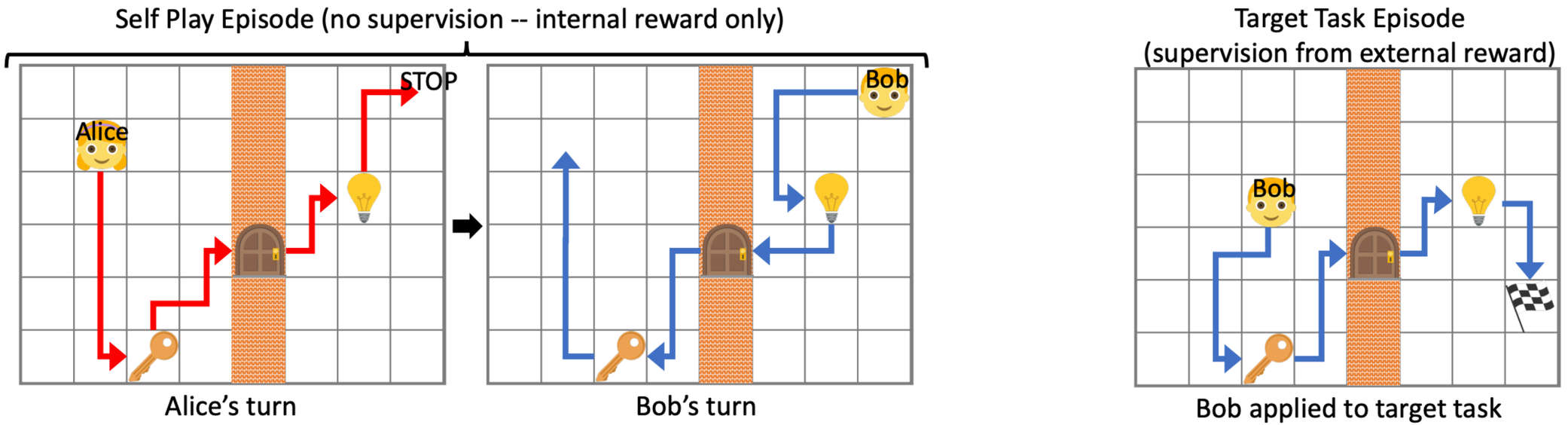
- may require supervision in the loop of RL
(usually requires more human time)

Thought exercise: Are there other forms of feedback, supervision that might be helpful?

Can RL agents propose their own goals?

Entire area of “unsupervised” RL

One example: Formulate two-player game, with a goal-setter and a goal reacher



Sukhbaatar et al. Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. ICLR 2018

The plan for today

1. Offline reinforcement learning: recap & conservative methods
2. Reward learning:
   a. Where do rewards come from?
   b. Learning rewards from example goals, behaviors
   c. Learning rewards from human preferences

Key learning goals:

- why task specification is hard (& why naïve methods fail)
- methods for learning rewards from human supervision

Course reminders

Project

- Fill out AWS form for GPU quota
- Proposal due today

  graded fairly lightly- it's for your benefit!

Homework

- Homework 2 due next Friday. (start early!!)

Guest lectures next week!

Archit Sharma

Researcher on Gemini team

Lead author of DPO

Preference optimization for LLMs

Aviral Kumar

Prof at CMU

At forefront of LLM reasoning

RL for LLM reasoning

Both very useful for default project. **Please join lecture in person!**