

Recap of offline RL +

^ Reward Learning

CS 224R

Course reminders

- Homework 2 due tonight at 9 pm
- Project proposal due **next Friday** -> graded fairly lightly- it's for your benefit!

Guest lectures next week!

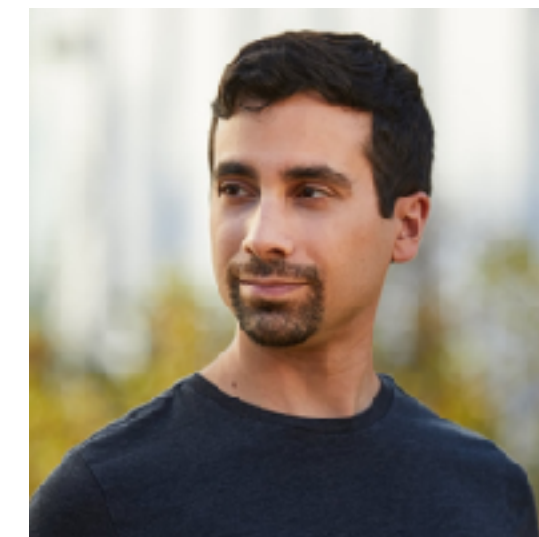


Archit Sharma

Researcher on Gemini team

Lead author of DPO

Preference optimization for LLMs



Noam Brown

Researcher at OpenAI

Frontier reasoning (o1, IMO gold, etc)

RL for LLM reasoning

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

The plan for today

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

Part of HW3

Part of default project
How LLMs are supervised!

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 from online RL:

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

Off-policy policy update $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_i \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_i^{\pi} | \mathbf{s}_i) \hat{Q}^{\pi}(\mathbf{s}_i, \mathbf{a}_i^{\pi})$ where $\mathbf{a}_i^{\pi} \sim \pi_{\theta}(\mathbf{a} | \mathbf{s}_i)$

Train policy using these OOD actions

Querying Q-function on OOD actions causes overestimation.

In online RL, data from new policy can correct errors in future iterations.

If offline RL, there is no additional data! -> need to be more conservative

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 from online RL:

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

Off-policy policy update $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_i \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_i^{\pi} | \mathbf{s}_i) \hat{Q}^{\pi}(\mathbf{s}_i, \mathbf{a}_i^{\pi})$ where $\mathbf{a}_i^{\pi} \sim \pi_{\theta}(\mathbf{a} | \mathbf{s}_i)$

Train policy using these OOD actions

Querying Q-function on OOD actions causes overestimation.

 **Key idea #1:** Only train policy on actions sampled from dataset

$$\theta \leftarrow \arg \max_{\theta} E_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} \left[\log \pi_{\theta}(\mathbf{a} | \mathbf{s}) \exp(\hat{A}^{\pi_{\beta}}(\mathbf{s}, \mathbf{a})) \right]$$

e.g. advantage weighted regression

AWR also avoids OOD value function queries by fitting value for π_{β} instead of for π_{θ}

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 from online RL:

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

Off-policy policy update $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_i \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_i^{\pi} | \mathbf{s}_i) \hat{Q}^{\pi}(\mathbf{s}_i, \mathbf{a}_i^{\pi})$ where $\mathbf{a}_i^{\pi} \sim \pi_{\theta}(\mathbf{a} | \mathbf{s}_i)$

Train policy using these OOD actions

Querying Q-function on OOD actions causes overestimation.

Key idea #2: Use asymmetric **expectile loss** to fit value function for policy π better than π_β .

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

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

Example of Offline RL in Practice



$\pi^*0.6$: a VLA That Learns From Experience. 2025.

Iterated offline RL recipe for robot post-training

1. Collect large batch of data using π (mix of roll-outs and DAgger)
2. Fit V^{π_β} using Monte Carlo
3. Train advantage-conditioned policy

$$\pi \left(\mathbf{a}_t \mid \mathbf{s}_t, \hat{A}(\mathbf{s}_t, \mathbf{a}_t) \right)$$

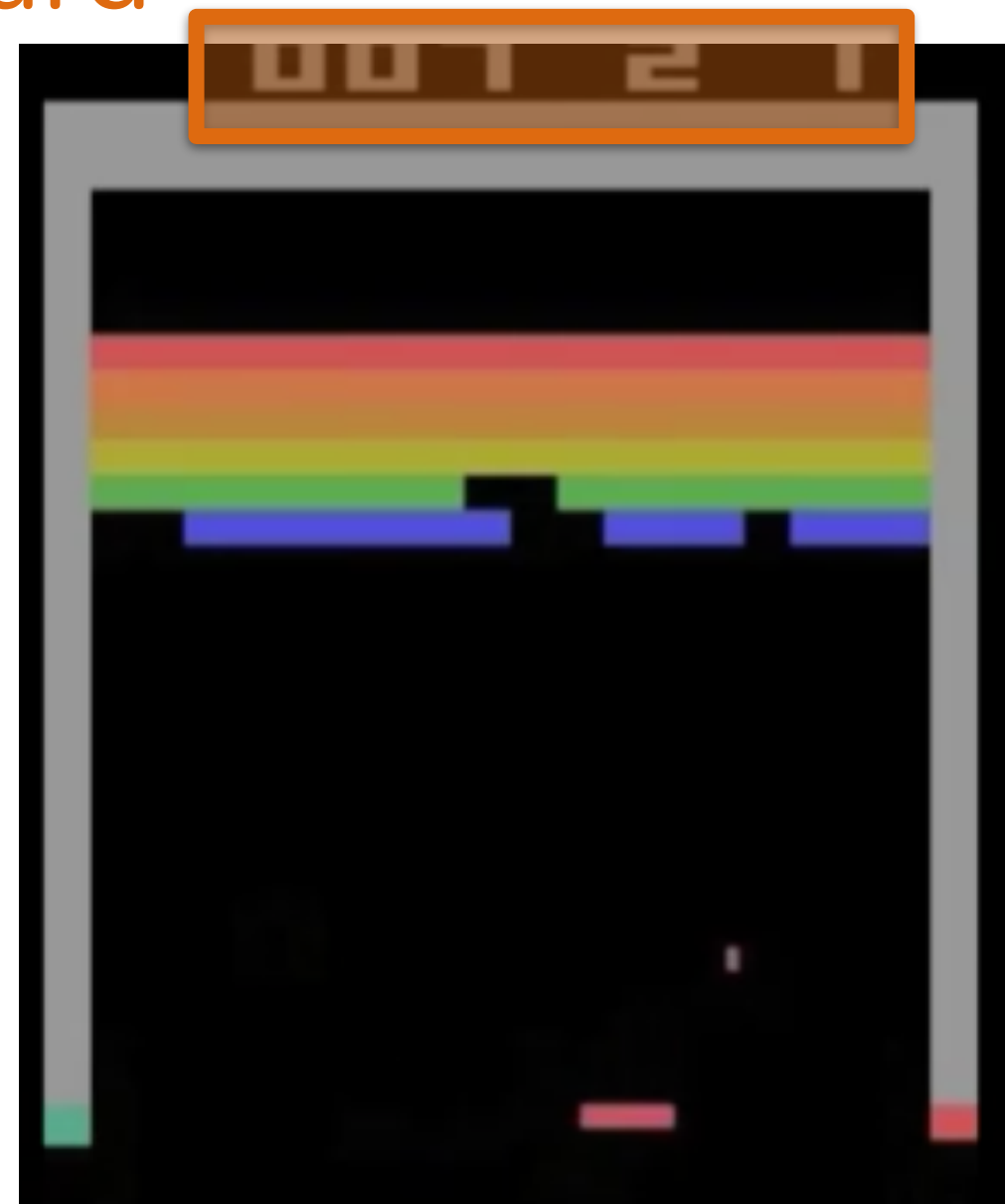
The plan for today

1. Offline reinforcement learning: recap & example
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



what is the reward?
often use a proxy

autonomous driving



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

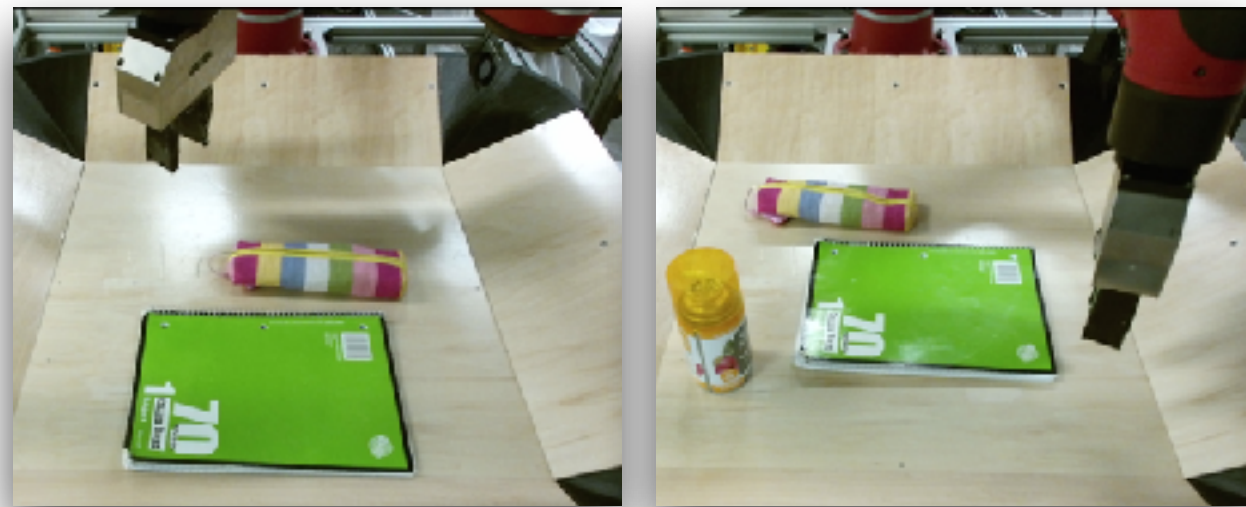
1. Offline reinforcement learning: recap & example
2. Reward learning:
 - a. Where do rewards come from?
 - b. Learning rewards from example goals, behaviors**
 - c. Learning rewards from human preferences

Goal classifiers

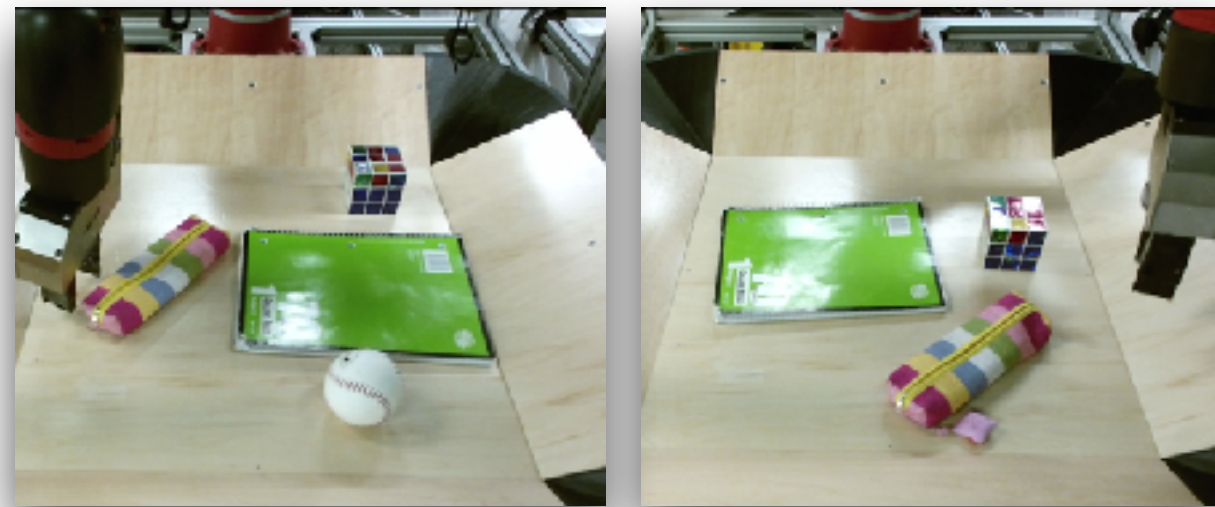
Key idea: Learn to discern goal states from other states

Example task: put pencil case behind notebook

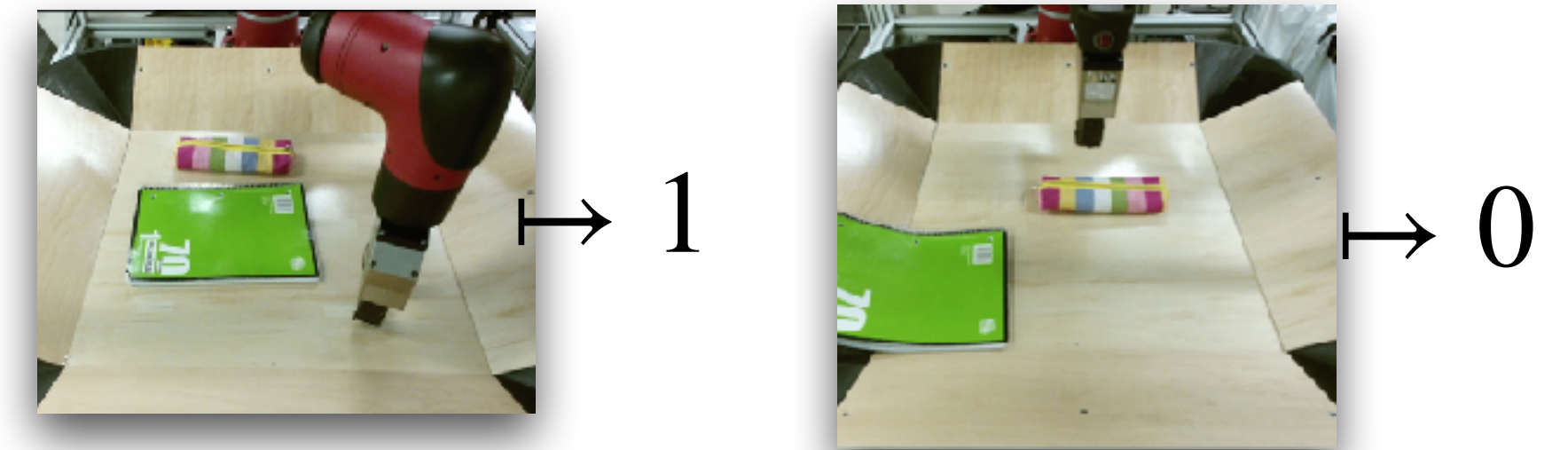
Positive examples



Negative examples



Trained binary classifier



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, \dots 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, \dots 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 $p \geq 0.5$ for successful states. 😄

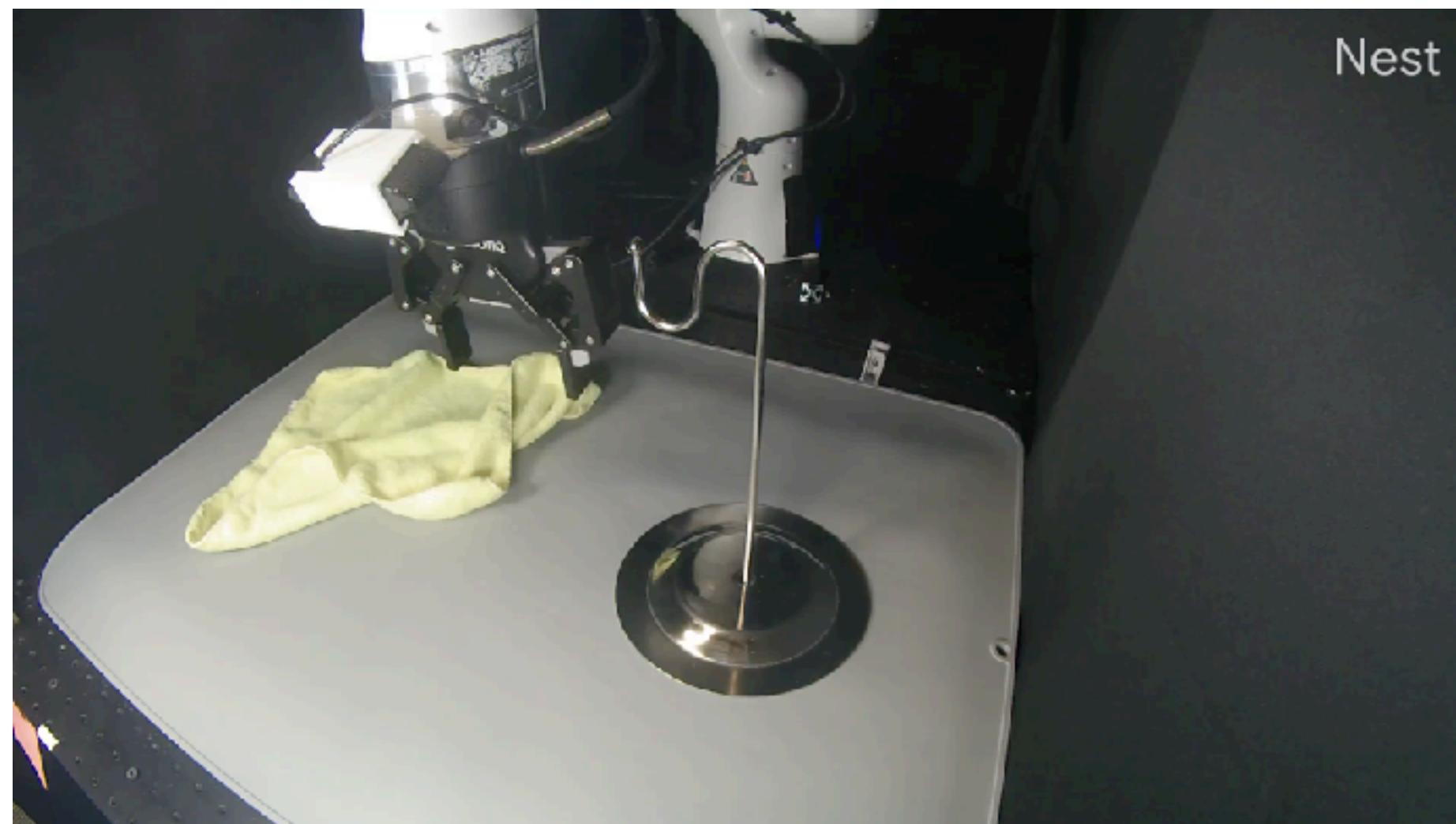
Goal classifiers for robotic reinforcement learning

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

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

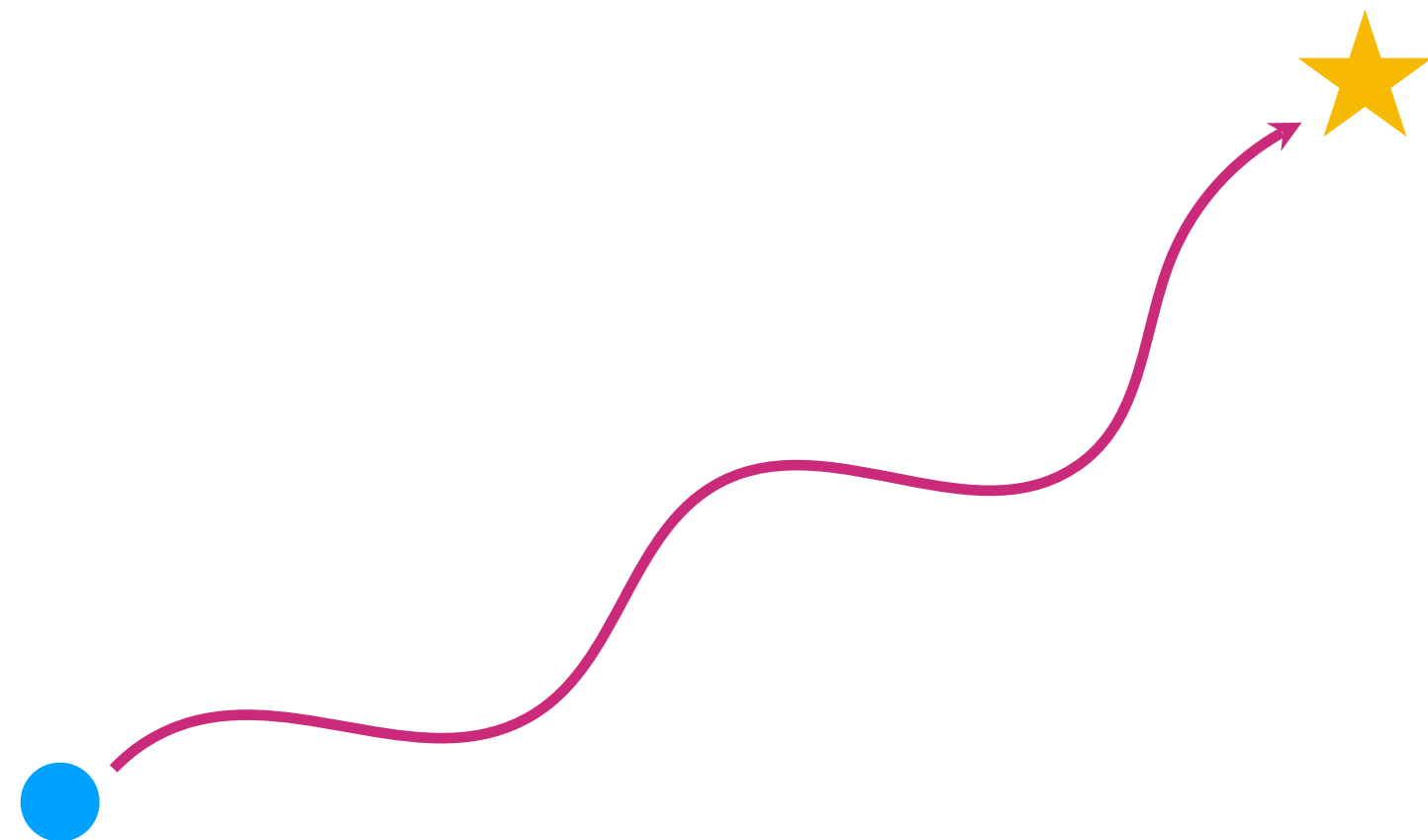
1. Offline reinforcement learning: recap & example
2. Reward learning:
 - a. Where do rewards come from?
 - b. Learning rewards from example goals, behaviors
 - c. Learning rewards from human preferences**

Can humans provide feedback on policy roll-outs?

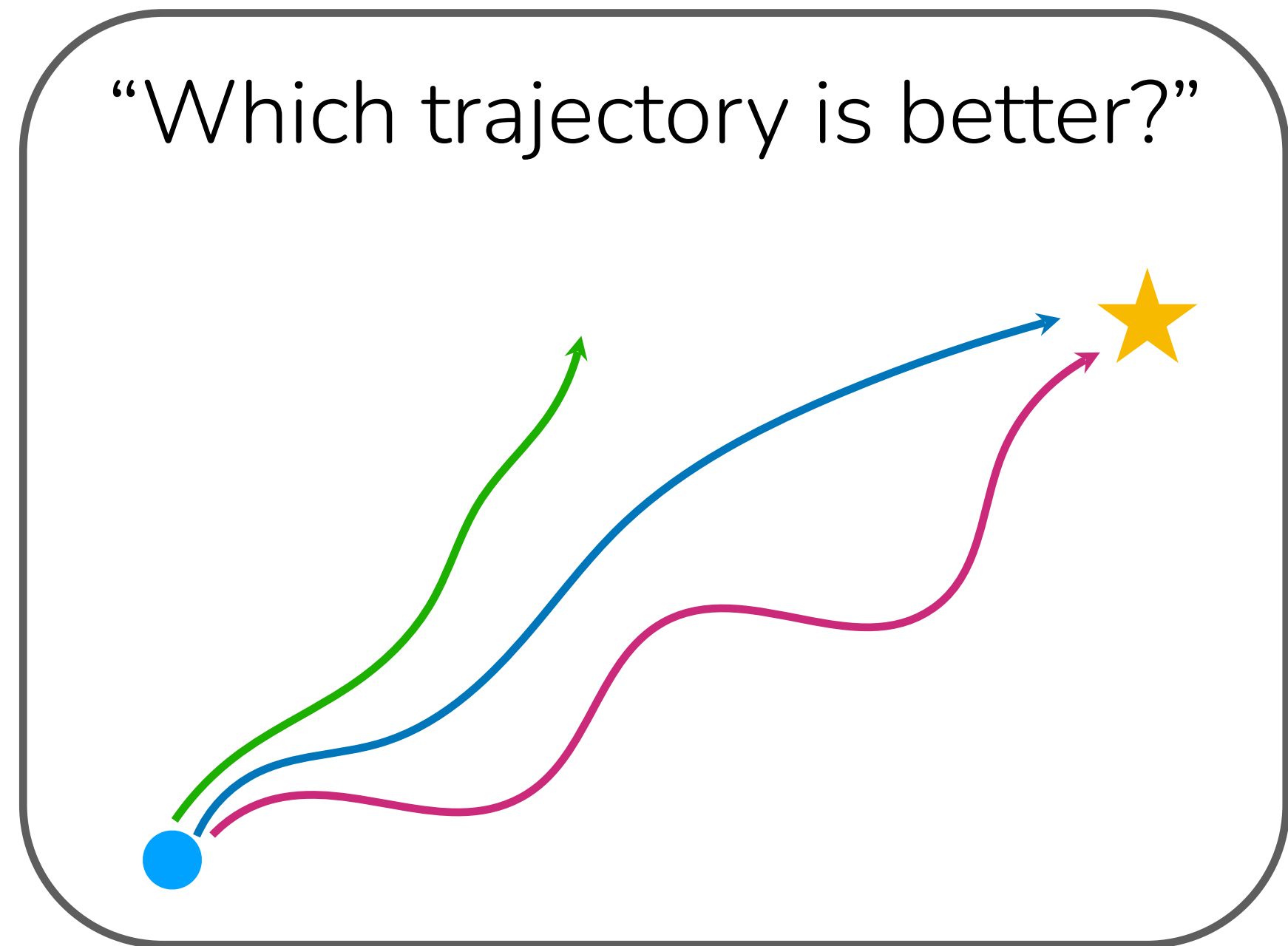
instead of requiring demos or example goals
(or in addition to!)

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 $\sum_{(s,a) \in \tau_w} r_\theta(s,a) > \sum_{(s,a) \in \tau_l} r_\theta(s,a)$.


shorthand $r_\theta(\tau_w)$

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 $\sum_{(s,a) \in \tau_w} r_\theta(s,a) > \sum_{(s,a) \in \tau_l} r_\theta(s,a)$.

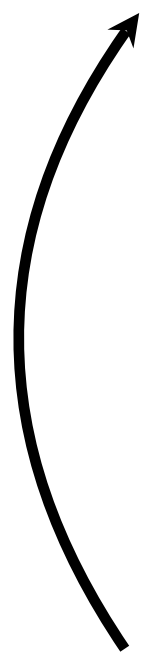

shorthand $r_\theta(\tau_w)$

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

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

How to learn a reward function from human preferences?

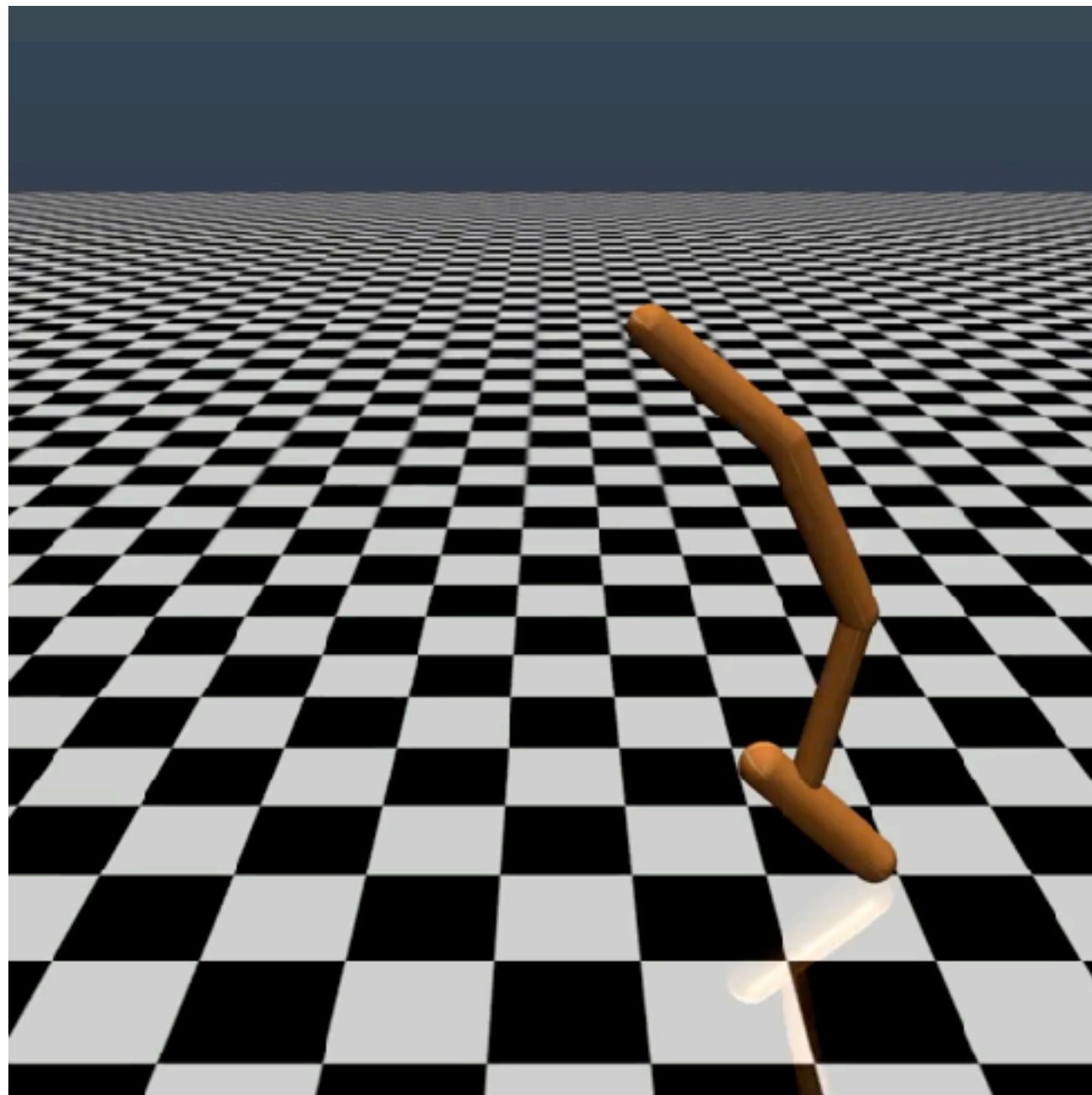
Complete reward learning algorithm

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 $r_\theta(\tau_1), \dots, r_\theta(\tau_k)$ under current reward model r_θ
 3. For all $\binom{k}{2}$ pairs per batch, compute $\nabla_\theta \mathbb{E}_{\tau_w, \tau_l} [\log \sigma(r_\theta(\tau_w) - r_\theta(\tau_l))]$ where $\tau_w \succ \tau_l$
 4. Update θ using computed gradient.
- 

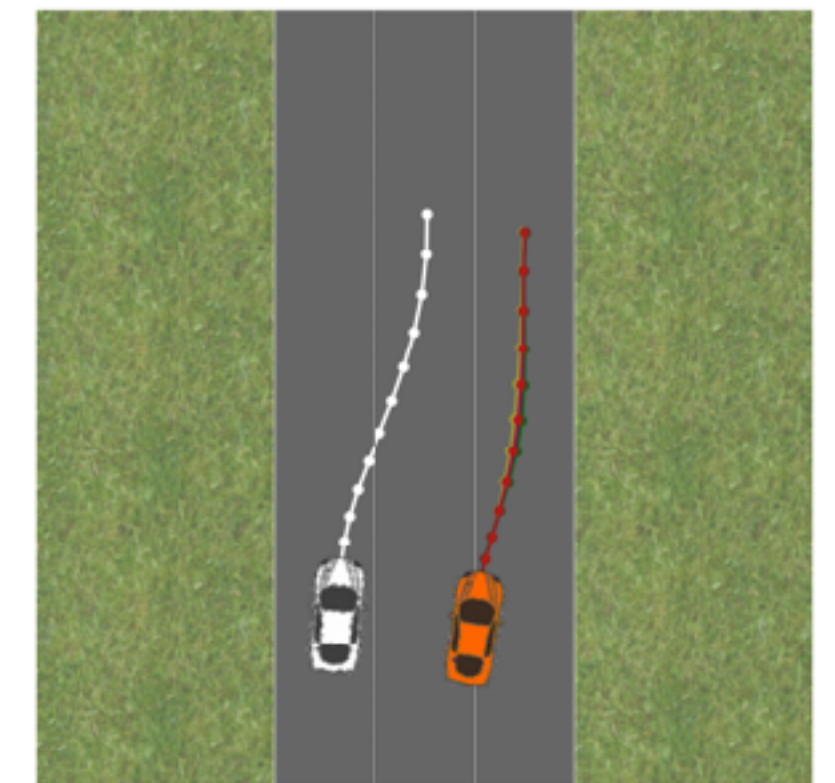
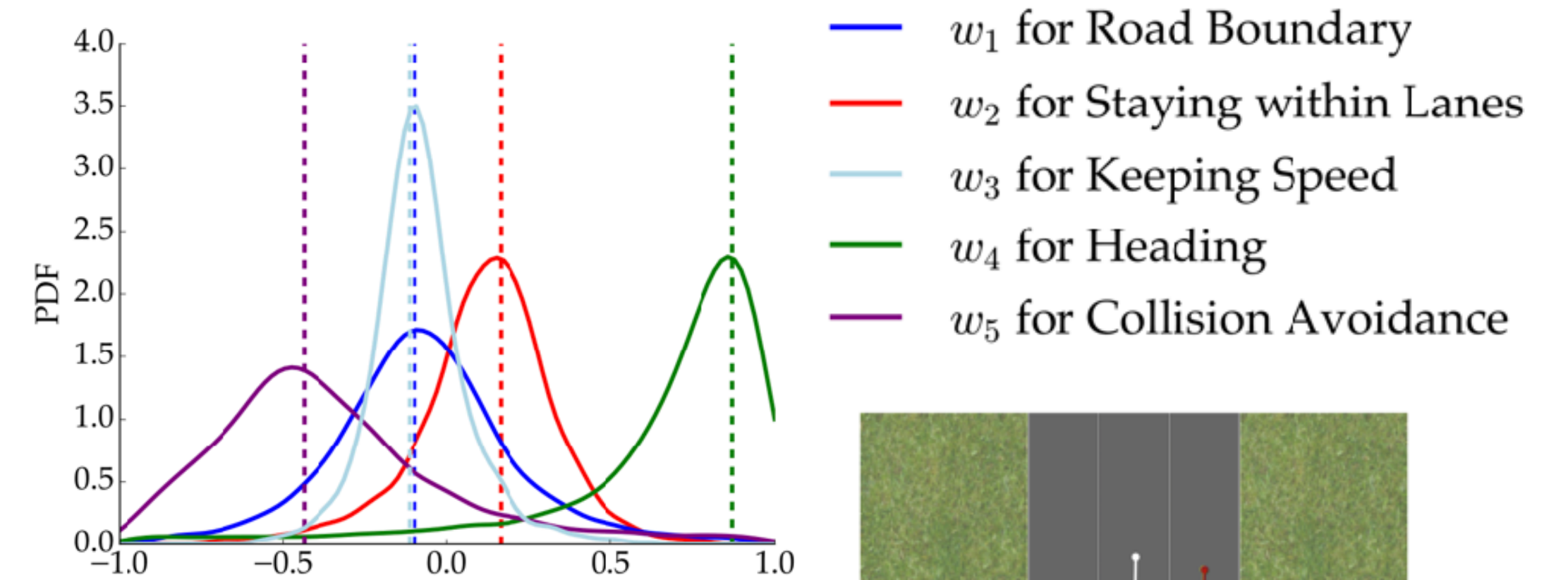
Note: This can be done in the loop of online RL.

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.

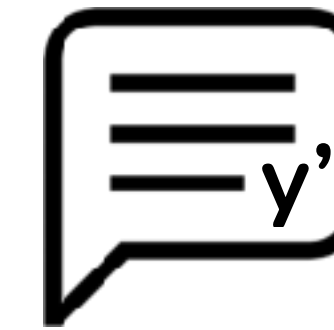
Learning rewards from human feedback for LLMs

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 non-stick 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 letting 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

$$y > y'$$

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

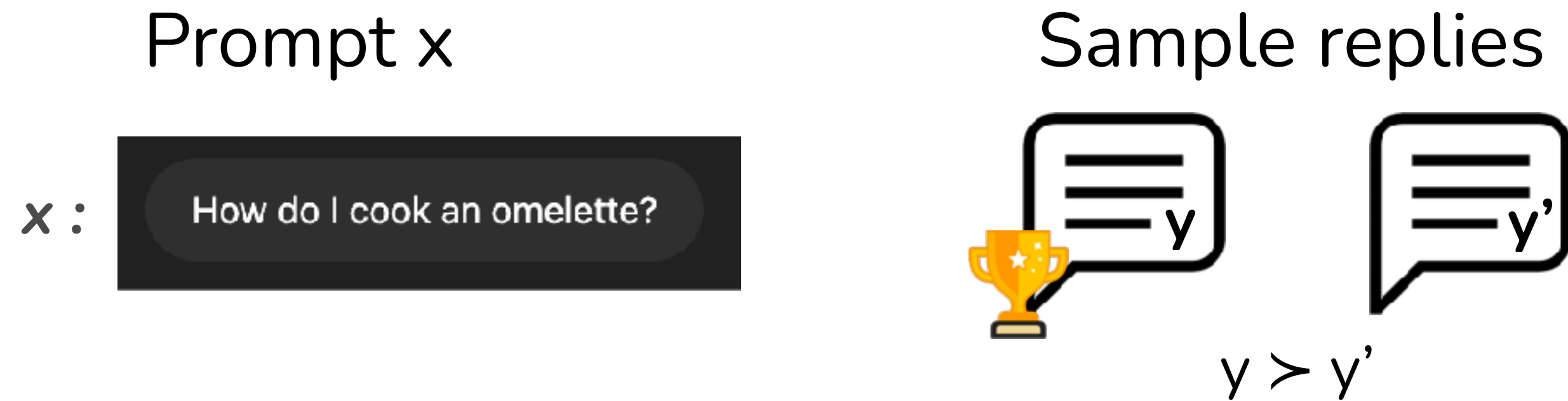
Learning rewards from human feedback for LLMs

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.

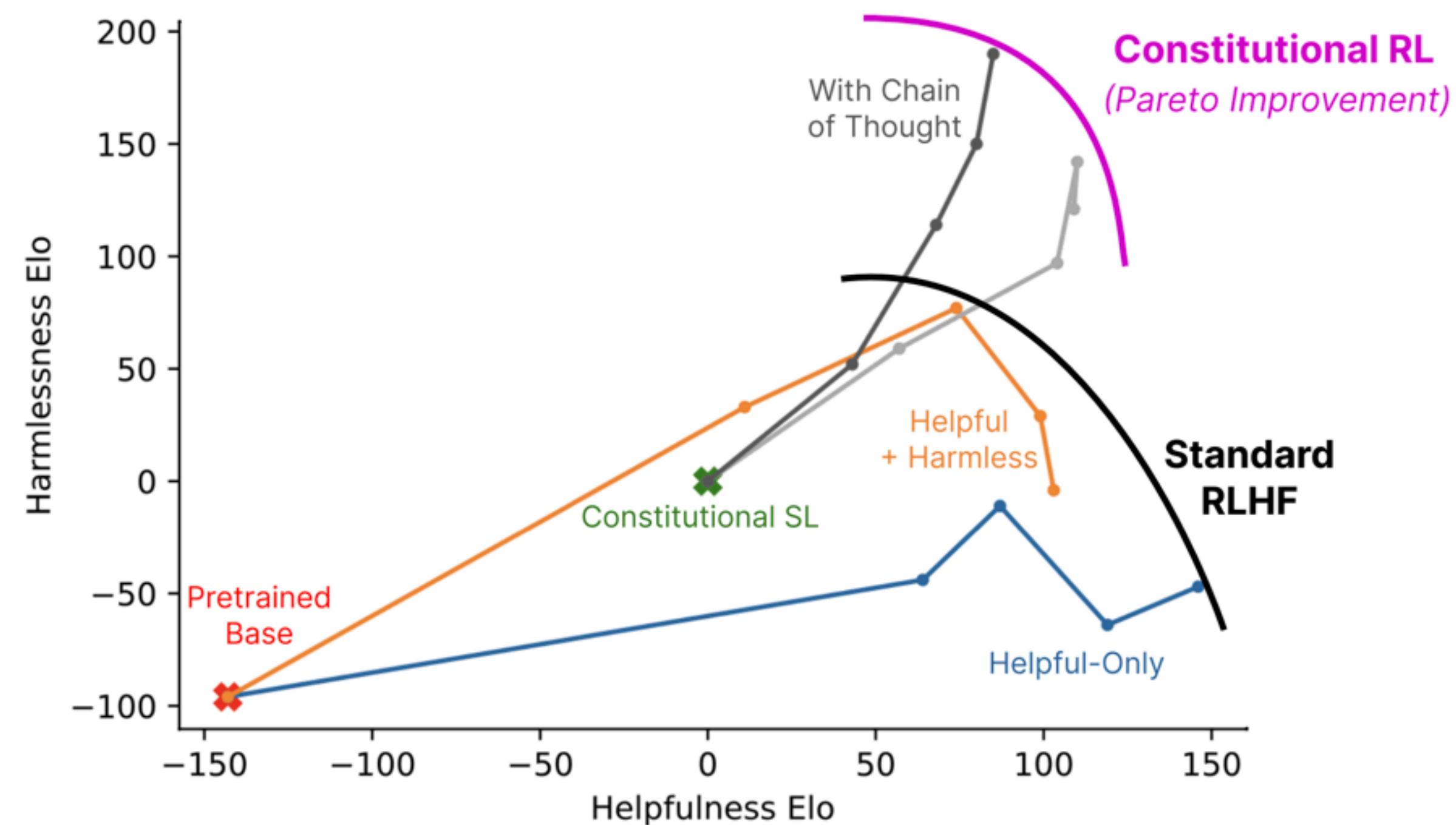
Finetune model to predict y for prompt x with high reward

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!



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

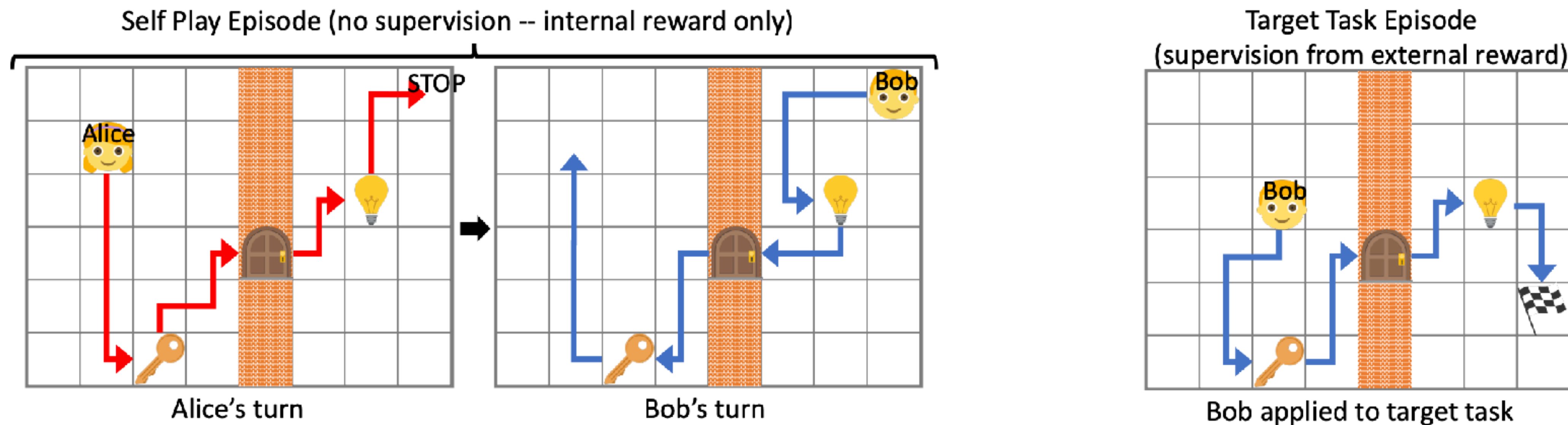
(usually requires more human time)

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

- Homework 2 due tonight at 9 pm
- Project proposal due **next Friday** -> graded fairly lightly- it's for your benefit!

Guest lectures next week!

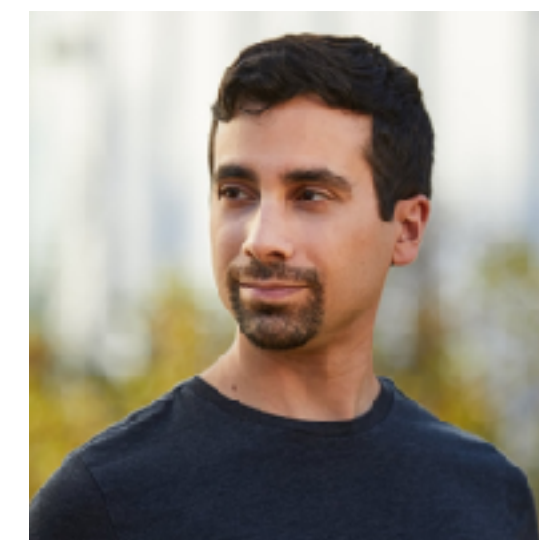


Archit Sharma

Researcher on Gemini team

Lead author of DPO

Preference optimization for LLMs



Noam Brown

Researcher at OpenAI

Frontier reasoning (o1, IMO gold, etc)

RL for LLM reasoning

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