Topics:

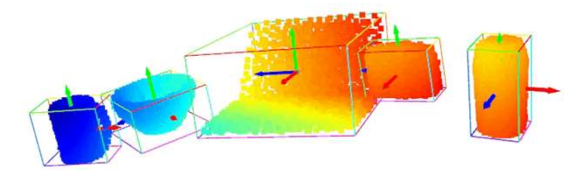
- Reinforcement Learning Part 3
  - **Actor-Critic**
  - **Proximal Policy Optimization (PPO)**
  - **RL from Human Feedback for ChatGPT**

CS 4644-DL / 7643-A

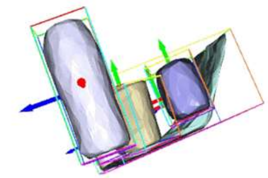
ZSOLT KIRA

Admin

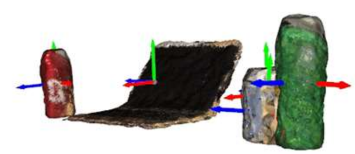
- HW4
  - See OH on Attention/Seq2seq and HW4
- LAST Meta OH on translation & speech - Thursday 2pm ET
- Next time: Guest lecture on implicit 3D models!
- Projects!



Ours - Meshes



Ours - Textures



RL: Sequential decision making in an environment with evaluative feedback.

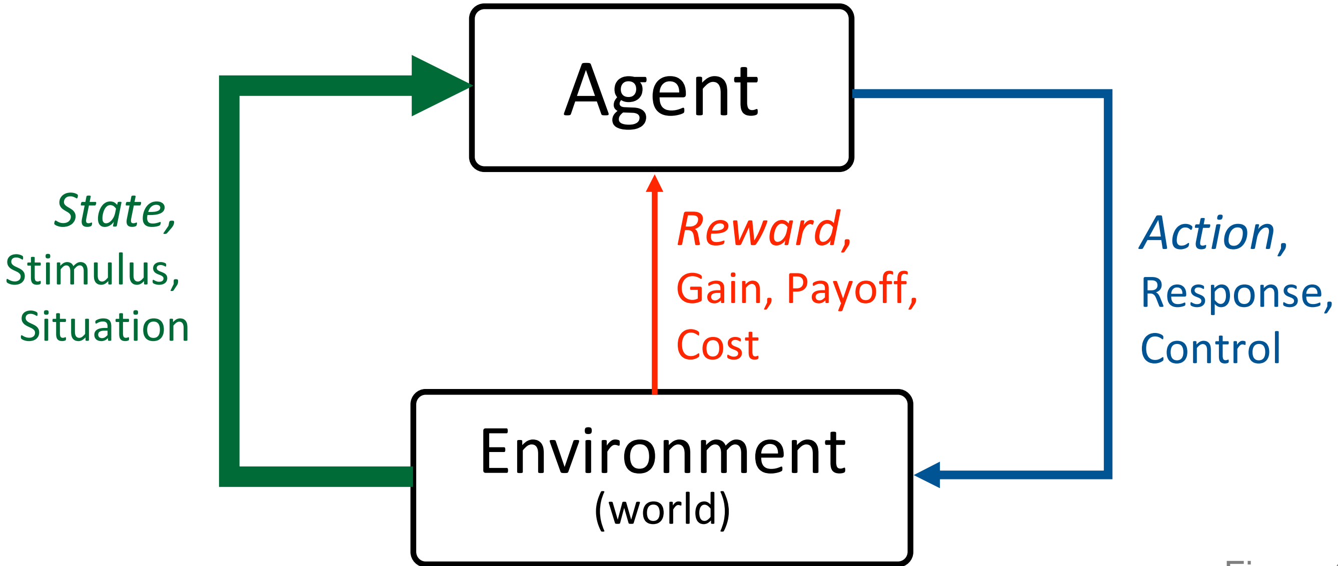


Figure Credit: Rich Sutton

- **Environment** may be unknown, non-linear, stochastic and complex.
- **Agent** learns a **policy** to map states of the environments to actions.
    - Seeking to maximize cumulative reward in the long run.

What is Reinforcement Learning?

- **MDPs**: Theoretical framework underlying RL
- An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$
  - $\mathcal{S}$: Set of possible states
  - $\mathcal{A}$: Set of possible actions
  - $\mathcal{R}(s, a, s')$: Distribution of reward
  - $\mathbb{T}(s, a, s')$: Transition probability distribution, also written as p(s'|s,a)
  - γ : Discount factor
- **Interaction trajectory**: $\ldots s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, s_{t+2}, \ldots$

Markov Decision Processes (MDPs)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Experience Replay

Epsilon-greedy

Q Update

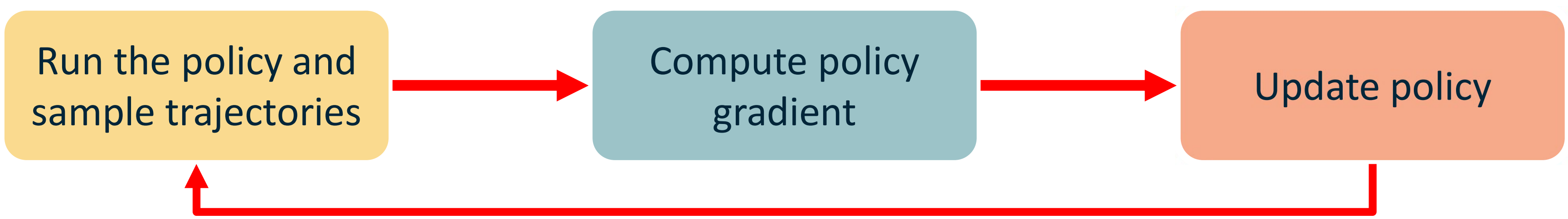
Deep Q-Learning Algorithm

The REINFORCE Algorithm

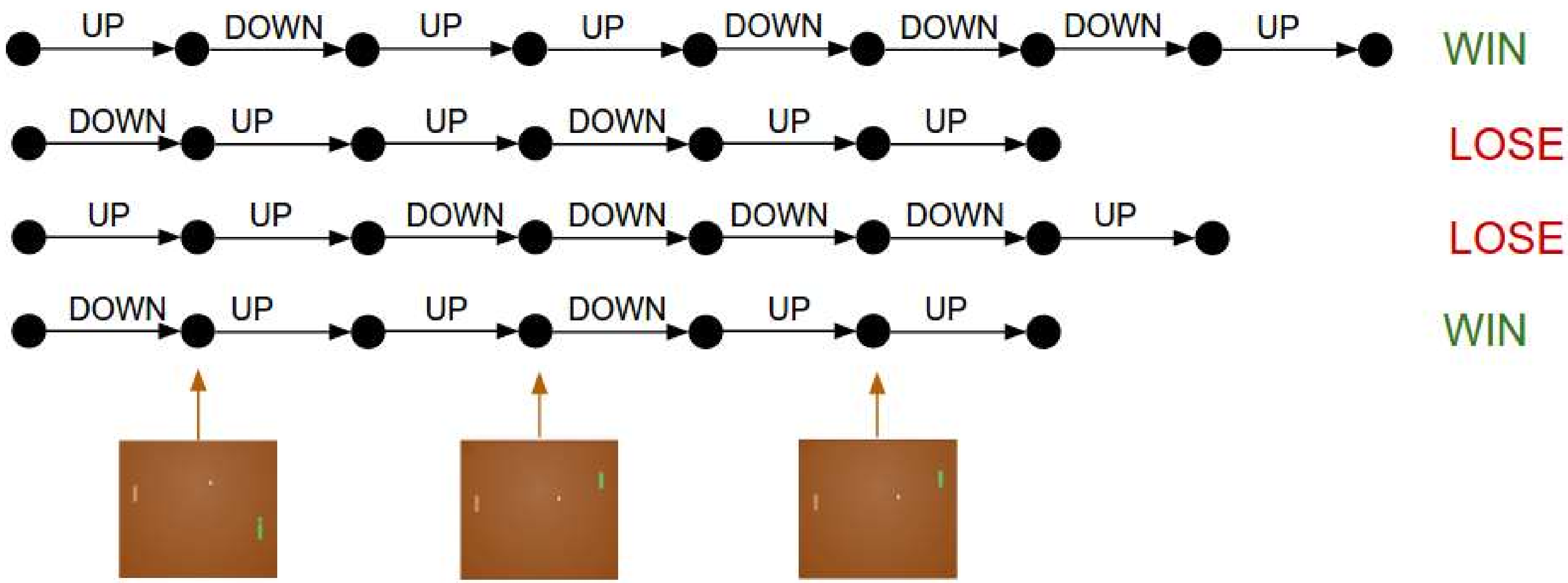
- Sample trajectories $\tau_i = \{s_1, a_1, \ldots s_T, a_T\}_i$ by acting according to π_θ
- Compute policy gradient as

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \left[\sum_{t=1}^T \nabla_\theta \log \pi_\theta \left(a_t^i \mid s_t^i \right) \cdot \sum_{t=1}^T \mathcal{R} \left(s_t^i \mid a_t^i \right) \right]$$

- Update policy parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Slide credit: Sergey Levine



Slide credit: Dhruv Batra

Drawbacks of Policy Gradients

Issues with Policy Gradients

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from high variance → leading to unstable training

Variance reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \left(\sum_{t'\geq t} r_{t'}\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \left(\sum_{t'\geq t} \gamma^{t'-t} r_{t'}\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from **high variance**, leading to unstable training

- How to reduce the variance?
  - Subtract an action independent baseline from the reward

$$\nabla_\theta J(\theta)=\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_\theta \log\pi_\theta\left(a_t \mid s_t\right)\cdot\sum_{t=1}^{T}\left(\mathcal{R}\left(s_t,a_t\right)-b(s_t)\right)\right]$$

  - Why does it work? **Normalization constant (expected value doesn't change)**
  - What is the best choice of b?

Drawbacks of Policy Gradients

Georgia Tech

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Actor-Critic

- Learn both policy and Q function
  - Use the “actor” to sample trajectories
  - Use the Q function to “evaluate” or “critic” the policy

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy

- REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \mathcal{R}(s,a) \right]$

- Actor-critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \right]$

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy
- REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \mathcal{R}(s,a) \right]$
- Actor-critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \right]$
- Q function is unknown too! Update using $\mathcal{R}(s,a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$

Actor-Critic

- Initialize s,θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

  - Update "critic": $\text{MSE Loss} := \Big(Q_{new}(s,a) - (r + \max_a Q_{old}(s',a))\Big)^2$
    - Recall Q-learning

Actor-Critic

Georgia Tech

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s, a)$ and next state $s' \sim p(s'|s, a)$
  - evaluate "actor" using "critic" $Q_\beta(s, a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s, a)$$

  - Update "critic":
    - Recall Q-learning $\text{MSE Loss} := \Big(Q_{new}(s, a) - (r + \max_a Q_{old}(s', a))\Big)^2$

$$a \leftarrow a', s \leftarrow s'$$

    - Update β Accordingly

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Intuitively, we are happy with an action a_t in a state s_t if $Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t | s_t)$

Actor-critic

- In general, replacing the policy evaluation or the "critic" leads to different flavors of the actor-critic
  - REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\mathcal{R}(s,a)\right]$
  - Q – Actor Critic $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)Q^{\pi_\theta}(s,a)\right]$
  - Advantage Actor Critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)A^{\pi_\theta}(s,a)\right]$

$$= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

"how much better is an action than expected?

Advanced policy gradient methods

- **Trust Region Policy Gradient** (TRPO, Schulman 2017)
- Issue with vanilla actor critic: policy may receive huge update!
  - Big parameter update -> drastic change in behavior -> may stuck in low-reward region!
- Idea: Anchor policy updates to past!

$$J(\theta) = \mathbb{E}_{s\sim\rho^{\pi_{\theta_{\text{old}}}}, a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\hat{A}_{\theta_{\text{old}}}(s,a)\right]$$

- Idea: constrain the update to a *trust region* using off-policy policy gradient
- Subject to:

$$\mathbb{E}_{s\sim\rho^{\pi_{\theta_{\text{old}}}}}[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(.\,|s)\|\pi_\theta(.\,|s)] \leq \delta$$

importance sampling

$$\begin{aligned} E_{x\sim p(x)}[f(x)] &= \int p(x)f(x)dx \\ &= \int \frac{q(x)}{q(x)}p(x)f(x)dx \\ &= \int q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right] \end{aligned}$$

Optimizing this objective requires calculating Hessian (second-order optimization)!

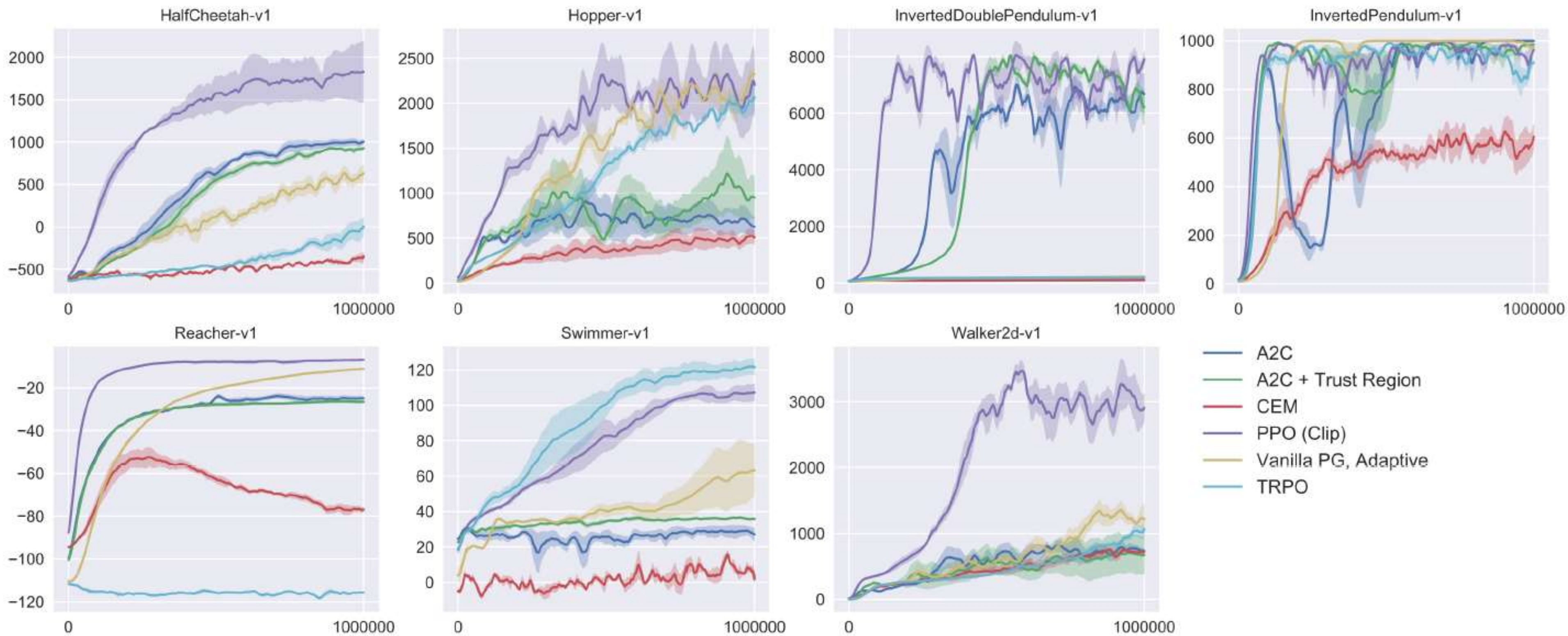
Advanced policy gradient methods

- **Proximal Policy Optimization** (PPO, Schulman 2017)
- Issue with TRPO: objective too complicated! Requires second-order optimization (calculating Hessian).

- Idea: Approximate trust-region constraint with a penalty term

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] - \beta\hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

Advanced policy gradient methods



Schulman 2017

Applications to Language Modeling

- One of the key benefits of RL is ability to tune to **evaluative feedback**
- Used to turn *unaligned* language models to ones that do what we want: Answer our questions
- Reinforcement Learning from Human Feedback (RLHF)

From Language Models to Assistants

1. **Zero-Shot (ZS) and Few-Shot (FS) In-Context Learning**
2. **Instruction finetuning**
3. **Reinforcement Learning from Human Feedback (RLHF)**

4. **What's next?**

RL to Improve Language modeling ≠ assisting users

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.
```

Language models are not *aligned* with user intent [Ouyang et al., 2022].

Language modeling ≠ assisting users

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

COMPLETION **Human**
A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

Language models are not *aligned* with user intent [Ouyang et al., 2022].
Finetuning to the rescue!

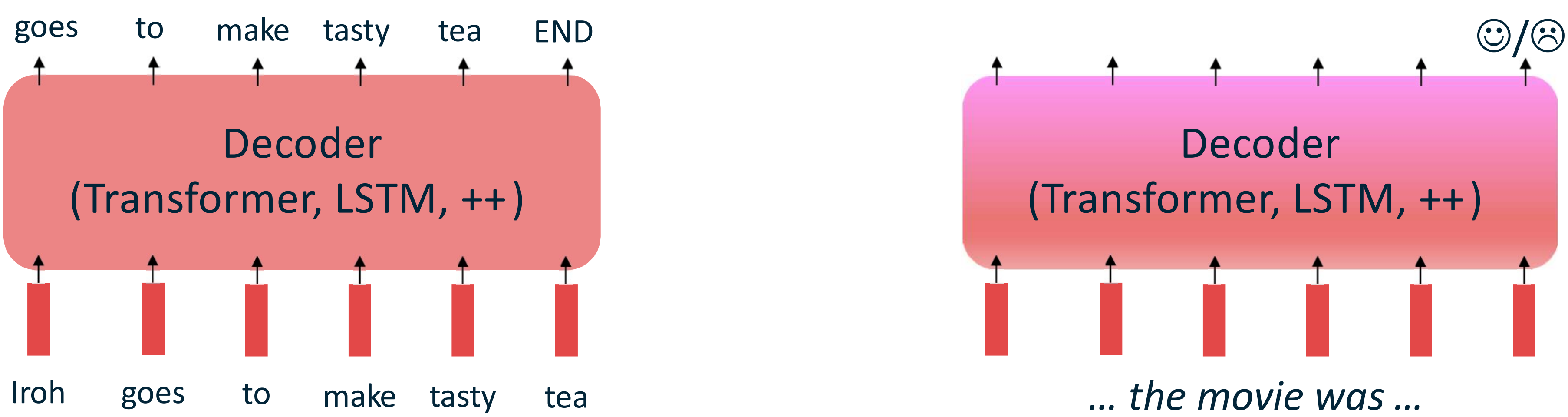
Pretraining can improve NLP applications by serving as parameter initialization.

Step 1: Pretrain (on language modeling)

Lots of text; learn general things!

Step 2: Finetune (on your task)

Not many labels; adapt to the task!



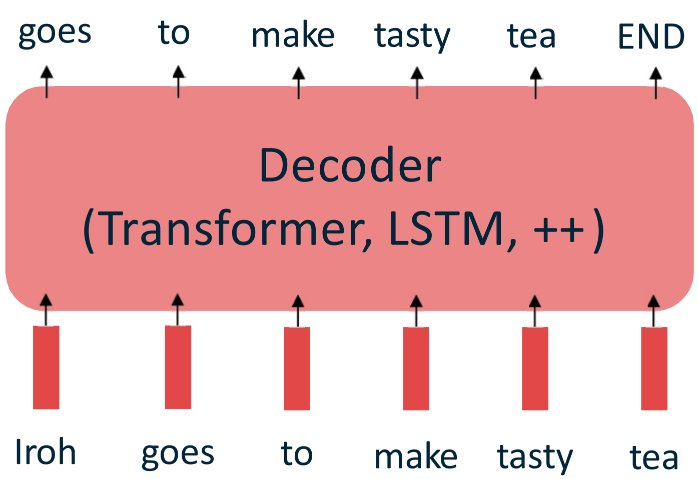
Slide Credit: Jesse Mu

Scaling up finetuning

Pretraining can improve NLP applications by serving as parameter initialization.

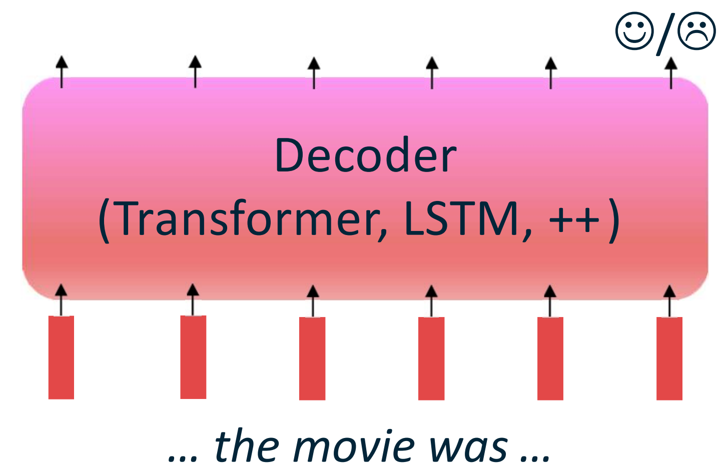
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



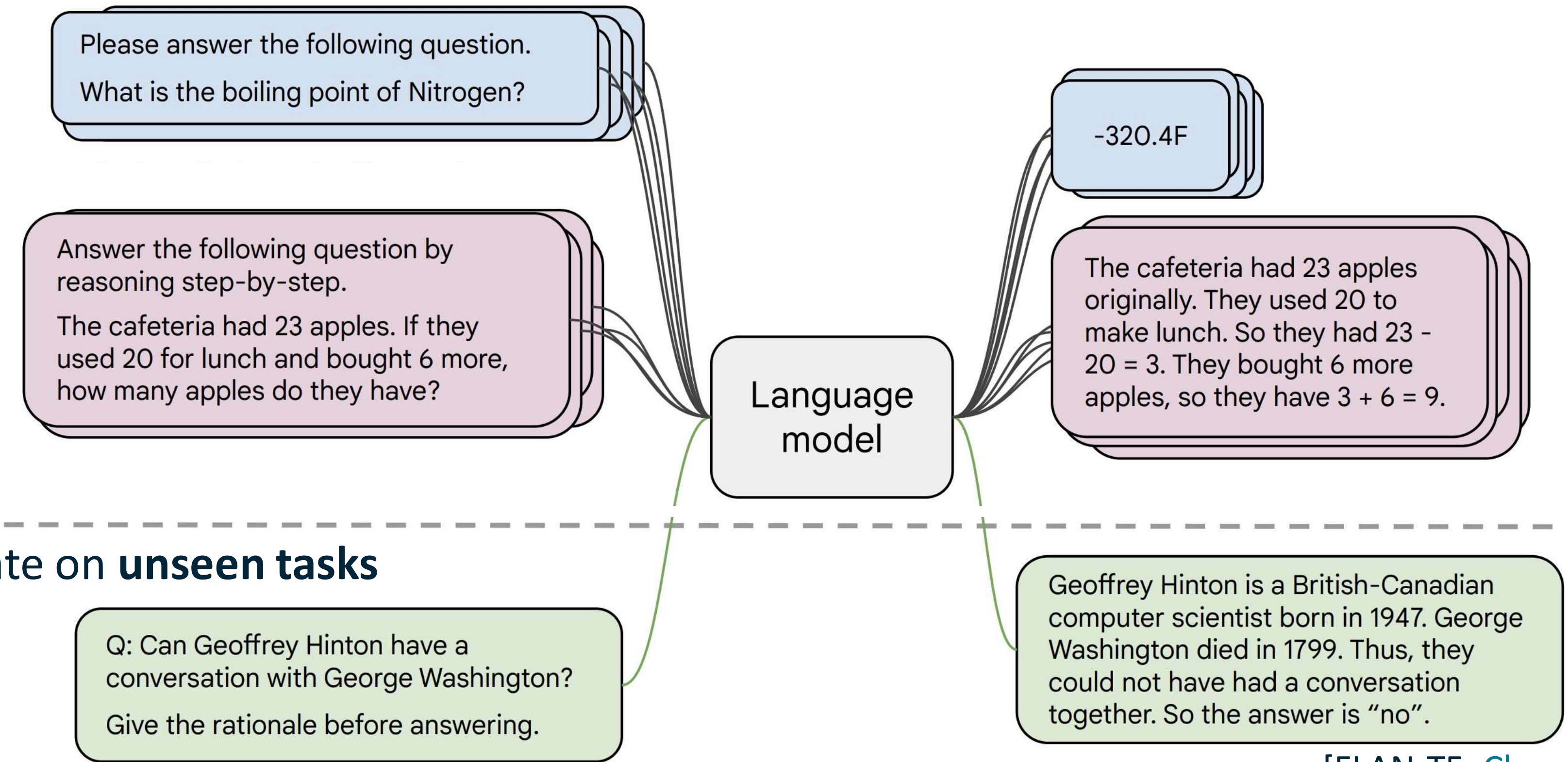
Step 2: Finetune (on many tasks)

~~Not~~ many labels; adapt to the tasks!



Instruction finetuning

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM



- Evaluate on **unseen tasks**

[FLAN-T5; Chung et al., 2022]

Instruction finetuning

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

Before instruction finetuning

The reporter and the chef will discuss their favorite dishes.
The reporter and the chef will discuss the reporter's favorite dishes.
The reporter and the chef will discuss the chef's favorite dishes.
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✖ (doesn't answer question)

Highly recommend trying FLAN-T5 out to get a sense of its capabilities:
https://huggingface.co/google/flan-t5-xxl

[Chung et al., 2022]

Instruction finetuning

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

After instruction finetuning

The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C).

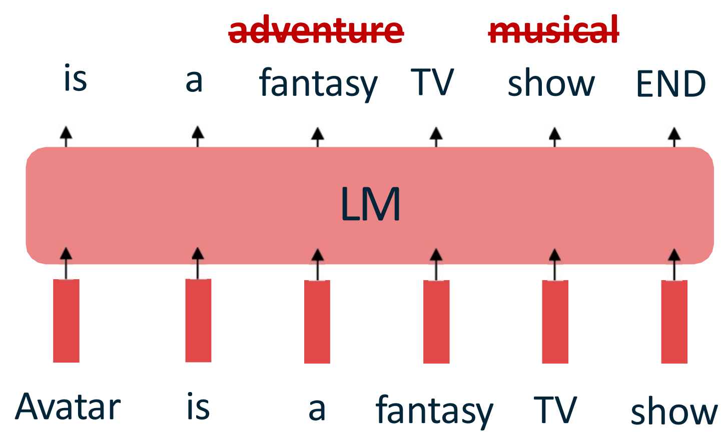
Highly recommend trying FLAN-T5 out to get a sense of its capabilities:
https://huggingface.co/google/flan-t5-xxl

[Chung et al., 2022]

Slide Credit: Jesse Mu

Limitations of instruction finetuning?

- One limitation of instruction finetuning is obvious: it's **expensive** to collect ground-truth data for tasks.
- But there are other, subtler limitations too. Can you think of any?
- **Problem 1:** tasks like open-ended creative generation have no right answer.
  - *Write me a story about a dog and her pet grasshopper.*
- **Problem 2:** language modeling penalizes all token-level mistakes equally, but some errors are worse than others.
- Even with instruction finetuning, there a mismatch between the LM objective and the objective of "satisfy human preferences"!
- Can we **explicitly attempt to satisfy human preferences?**



Slide Credit: Jesse Mu

From Language Models to Assistants

1. **Zero-Shot (ZS) and Few-Shot (FS) In-Context Learning**
   - \+ No finetuning needed, prompt engineering (e.g. CoT) can improve performance
   - – Limits to what you can fit in context
   - – Complex tasks will probably need gradient steps
2. **Instruction finetuning**
   - \+ Simple and straightforward, generalize to unseen tasks
   - – Collecting demonstrations for so many tasks is expensive
   - – Mismatch between LM objective and human preferences
3. **Reinforcement Learning from Human Feedback (RLHF)**

4. **What's next?**

From Language Models to Assistants

1. **Zero-Shot (ZS) and Few-Shot (FS) In-Context Learning**
   - \+ No finetuning needed, prompt engineering (e.g. CoT) can improve performance
   - – Limits to what you can fit in context
   - – Complex tasks will probably need gradient steps
2. **Instruction finetuning**
   - \+ Simple and straightforward, generalize to unseen tasks
   - – Collecting demonstrations for so many tasks is expensive
   - – Mismatch between LM objective and human preferences
3. **Reinforcement Learning from Human Feedback (RLHF)**

4. **What's next?**

Optimizing for human preferences

- Let's say we were training a language model on some task (e.g. summarization).
- For each LM sample s, imagine we had a way to obtain a *human reward* of that summary: $R(s) \in \mathbb{R}$, higher is better.

```
SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.
```

```
An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.
```

s_1

$R(s_1) = 8.0$

```
The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.
```

s_2

$R(s_2) = 1.2$

- Now we want to maximize the expected reward of samples from our LM:

$$\mathbb{E}_{\hat{s} \sim p_\theta(s)} [R(\hat{s})]$$

Note: for mathematical simplicity we're assuming only one "prompt"

Slide Credit: Jesse Mu

How do we model human preferences?

- Awesome: now for any **arbitrary, non-differentiable reward function** $R\ (s)$, we can train our language model to maximize expected reward.
- Not so fast! (Why not?)
- **Problem 1:** human-in-the-loop is expensive!
- **Solution:** instead of directly asking humans for preferences, **model their preferences** as a separate (NLP) problem! [Knox and Stone, 2009]

```
An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.
```

s_1

$R(s_1) = 8.0$

```
The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.
```

s_2

$R(s_2) = 1.2$

Train an LM $RM_\phi(s)$ to predict human preferences from an annotated dataset, then optimize for RM_ϕ instead.

Slide Credit: Jesse Mu

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]

```
A 4.2 magnitude
earthquake hit
San Francisco,
resulting in
massive damage.
```

s_3

$$R(s_3) = \quad 4.1? \quad 6.6? \quad 3.2?$$

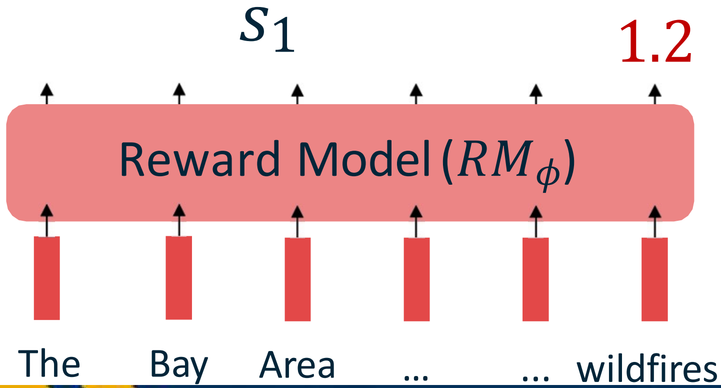
?

Slide Credit: Jesse Mu

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

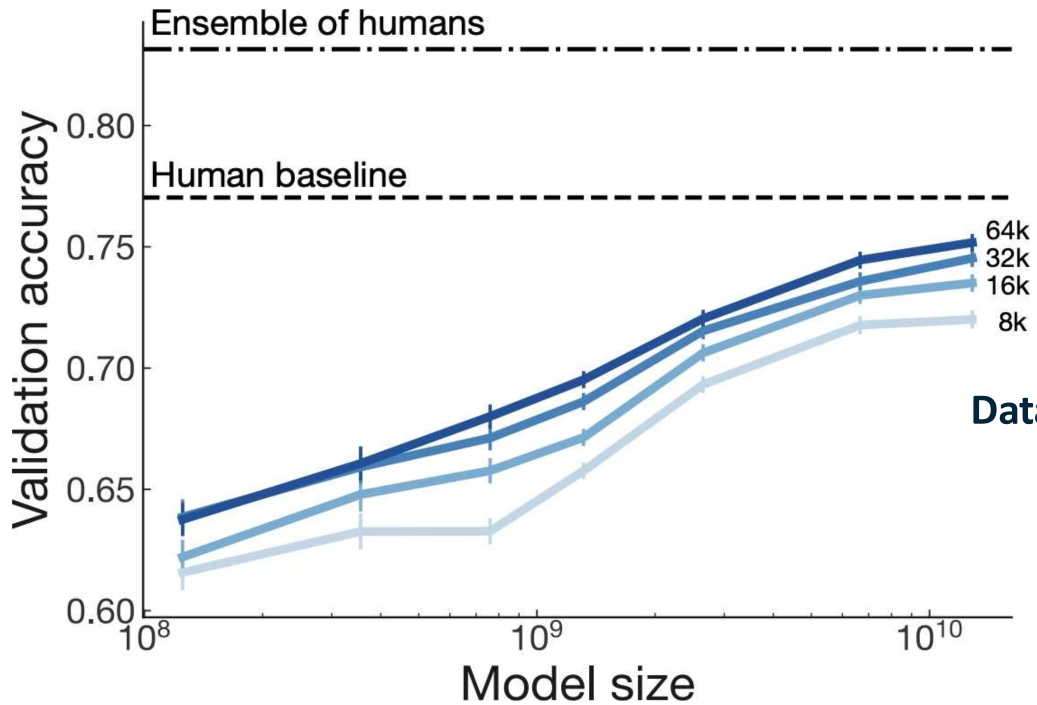
s^w should score higher than s^l

Slide Credit: Jesse Mu

Georgia Tech

Make sure your reward model works first!

Evaluate RM on predicting outcome of held-out human judgments



Data

Large enough RM trained on enough data approaching single human perf

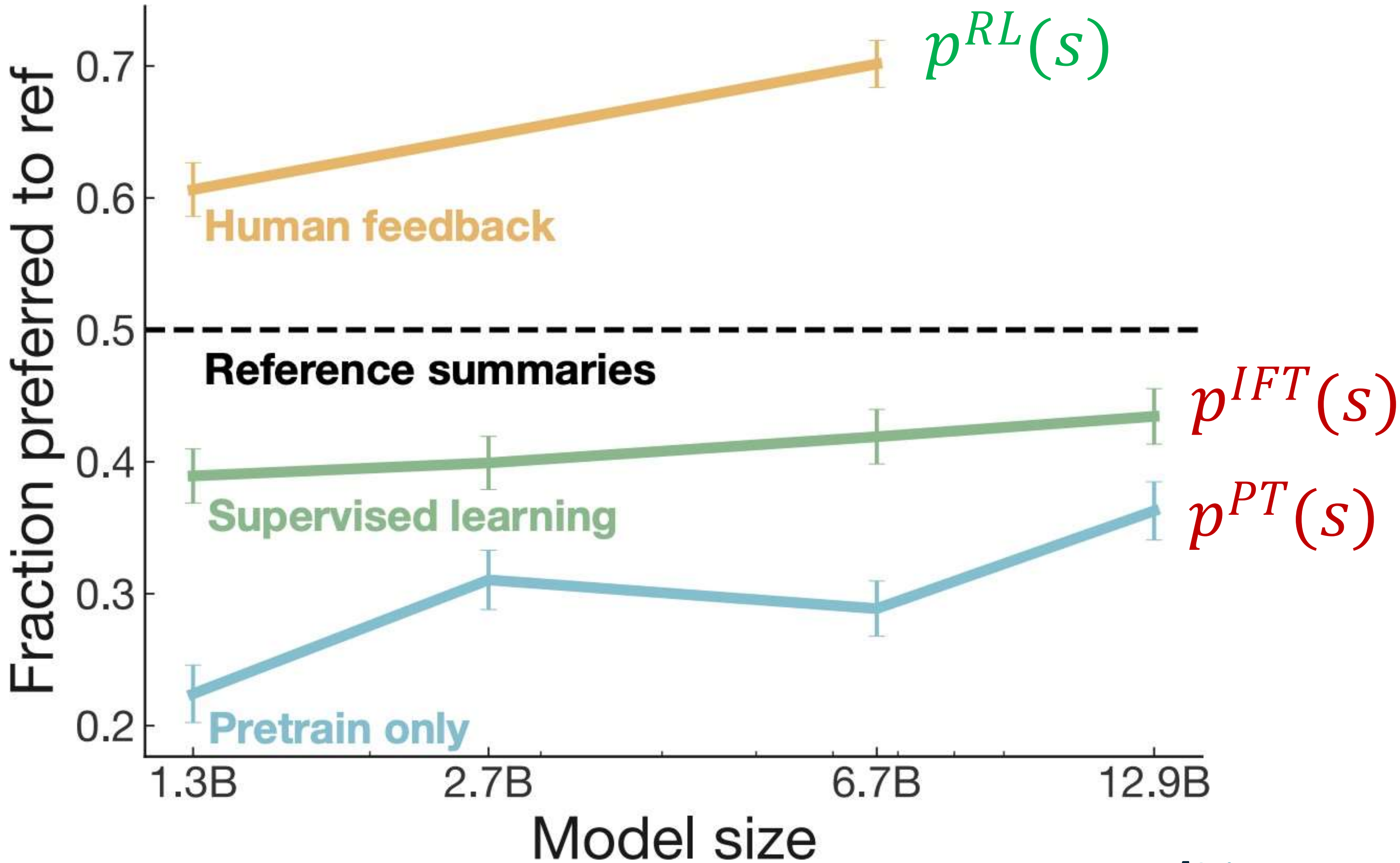
Slide Credit: Jesse Mu

[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 reward with RL

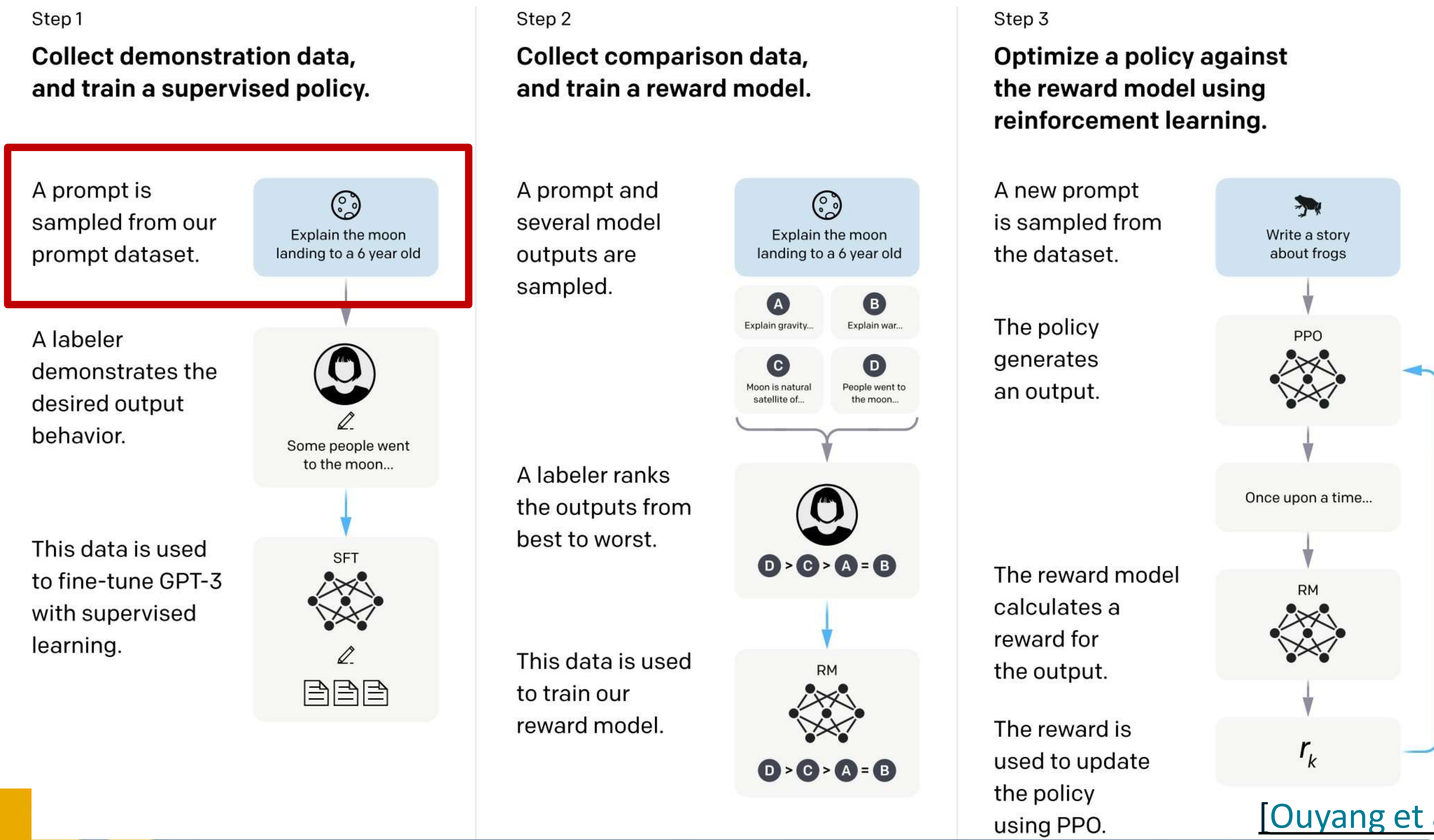
RLHF provides gains over pretraining + finetuning



[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.
```


Slide Credit: Jesse Mu

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!)

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/

Slide Credit: Jesse Mu