



香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP 4901B

Large Language Models

Reinforcement Learning Basics

Junxian He

Oct 15, 2025

Review: Multi-Turn Instruction Tuning

<User> How are you today?\n<Assistant> I'm doing well, thank you! How can I help you? \n\n<User> Can you tell me a joke? \n<Assistant> Sure! Why did the math book look sad? <stop>

Large Language Models



<start> <User> How are you today?\n<Assistant> I'm doing well, thank you! How can I help you? \n\n<User> Can you tell me a joke? \n<Assistant> Sure! Why did the math book look sad?

No different from before, still shift one token left to obtain output

Review: Inference time for ChatBot

I'm doing well, thank you! How can I help you? <stop>

Sure! Why did the math book look sad? <stop>

Large Language Models



<start> <User> How are you today? <user_stop> <Assistant> I'm doing well, thank you! How can I help you? <stop> <User> Can you tell me a joke?
<user_stop> <Assistant>

At inference time, we only ask the model to predict assistant parts

So typically, other parts are all masked when predicting the loss

Review: Inference time for ChatBot

Large Language Models



<start> <User> How are you today?<user_stop><Assistant> I'm doing well, thank you! How can I help you?<stop> <User> Can you tell me a joke?
<user_stop><Assistant>

Or, we only mask <user> <assistant> tags, not learning all contents, then the chatbot can suggest questions each round

Review: Chatbot can ask questions if not masking

Today

How are you today?
4:08 PM ✓

 GPT-5-Chat

I'm doing great, thanks for asking! 😊
How about you? What's your day been like so far?
4:08 PM

 Share 

Compare  @o3-mini →

Compare  @DeepSeek-R1-FW →

Compare  @GPT-5 →

Speak  @ElevenLabs-v2.5-Turbo →

It's been pretty good, just a regular day so far. →

My day is going well, I'm just relaxing at the moment. →

It's been busy, but productive, how can I help you? →

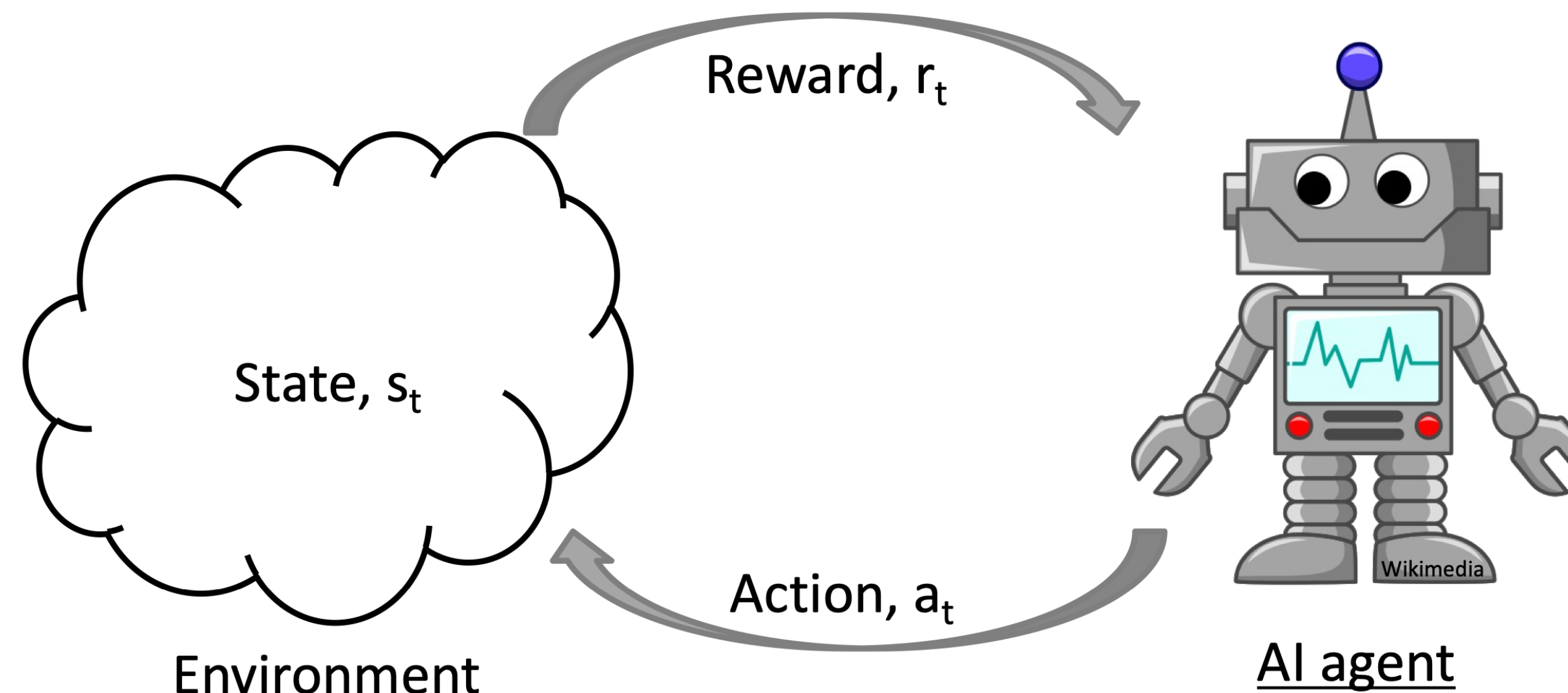
Reinforcement Learning

Learning Tasks

- Supervised learning - $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$
 - Regression - $y^{(i)} \in \mathbb{R}$
 - Classification - $y^{(i)} \in \{1, \dots, C\}$
- Unsupervised learning - $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$
 - Clustering
 - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \{\mathbf{s}^{(t)}, \mathbf{a}^{(t)}, r^{(t)}\}_{t=1}^T$

RL Setup

In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)



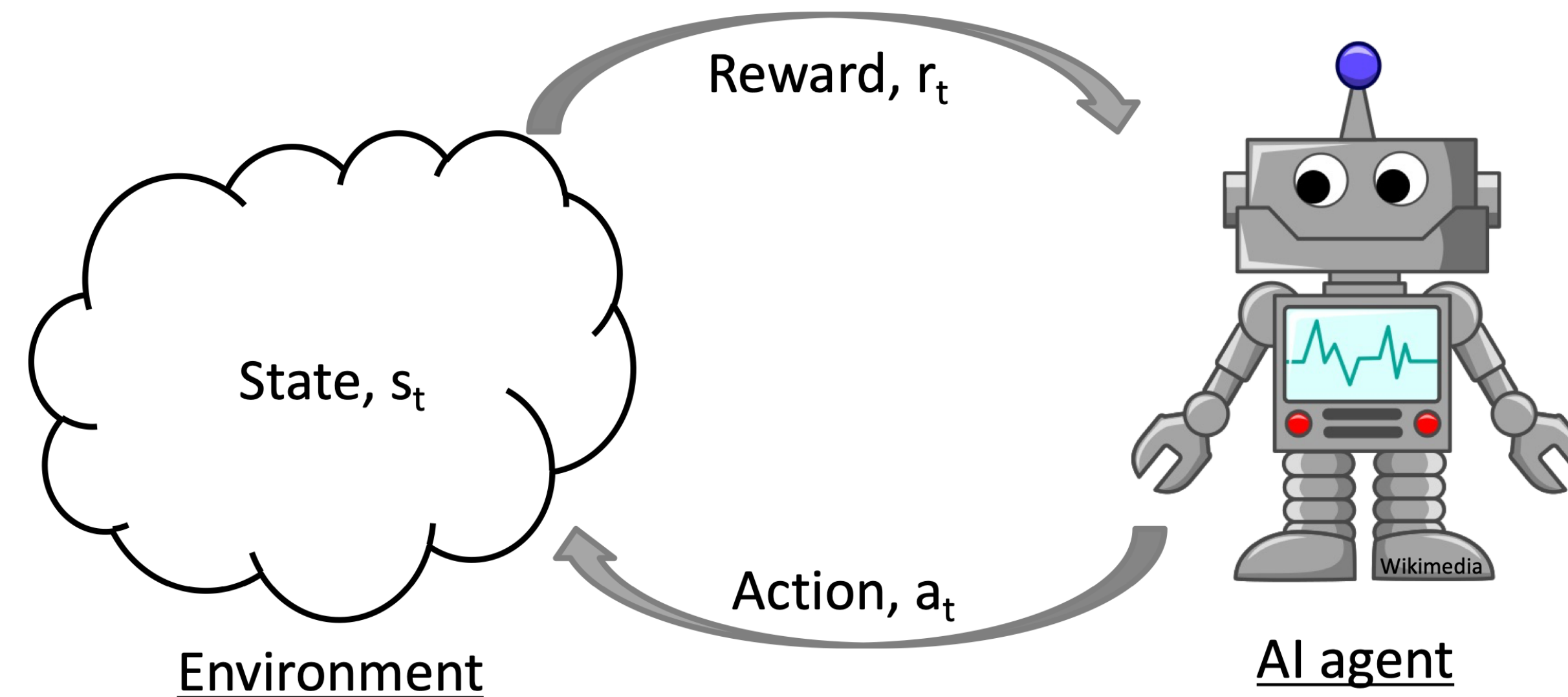
Agent chooses **actions** which can depend on past

Environment can change **state** with each action

Reward (Output) depends on (Inputs) action and state of environment

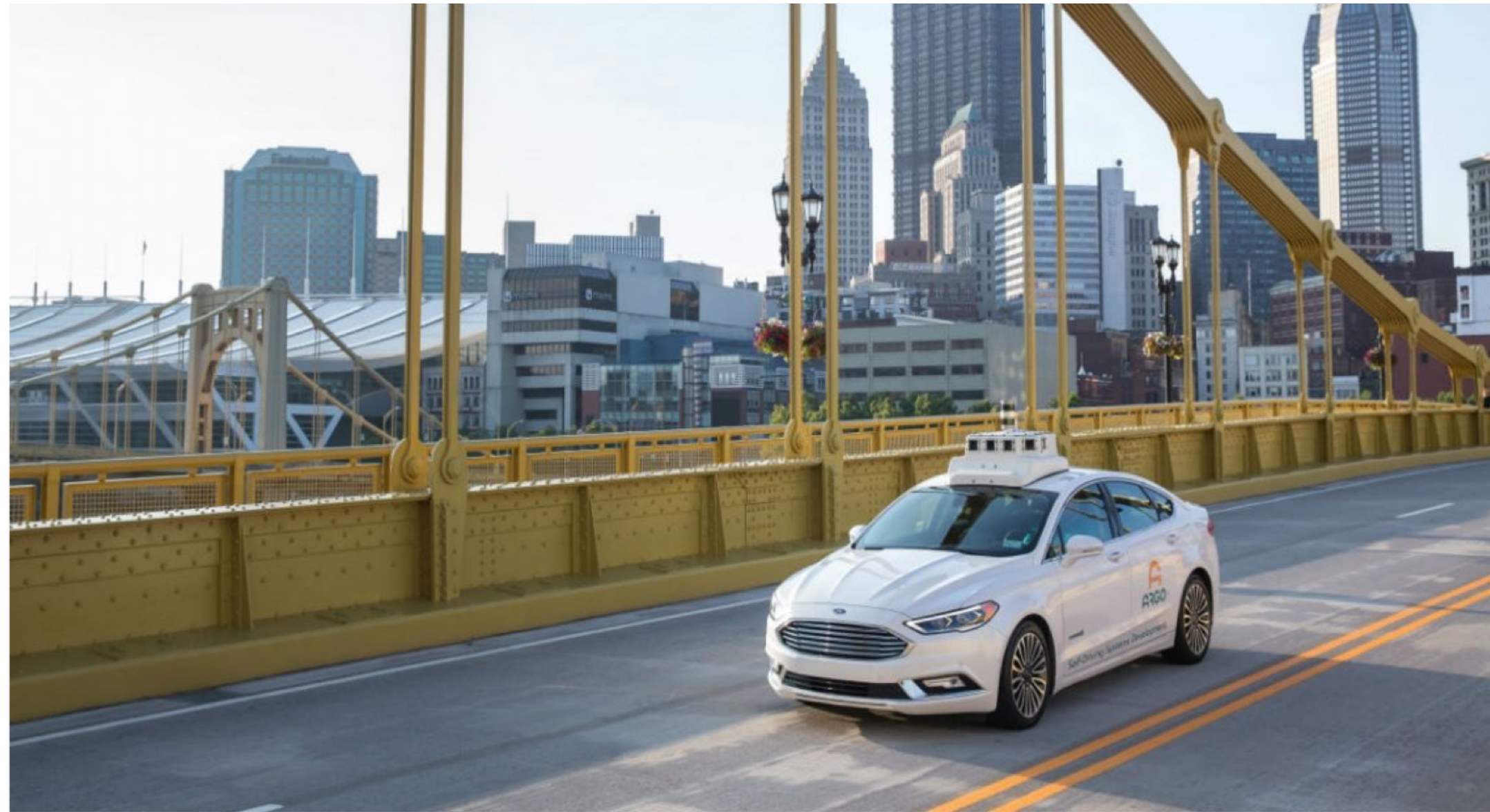
Goal: maximize the total reward

Differences from Supervised Learning



- Maximize reward (rather than learn reward) **Supervised training is like imitation**
- Inputs are not iid – state & action depends on past

RL Examples



RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
 - Stochastic, $p(r \mid s, a)$
 - Deterministic, $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
- Transition function
 - Stochastic, $p(s' \mid s, a)$
 - Deterministic, $\delta: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$

In this lecture, we assume they are known

- Reward and transition functions can be known or unknown

RL Setup

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$
 - Specifies an action to take in *every* state
- Value function, $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$
 - Measures the expected total reward of starting in some state s and executing policy π , i.e., in every state, taking the action that π returns

RL Example - gridworld

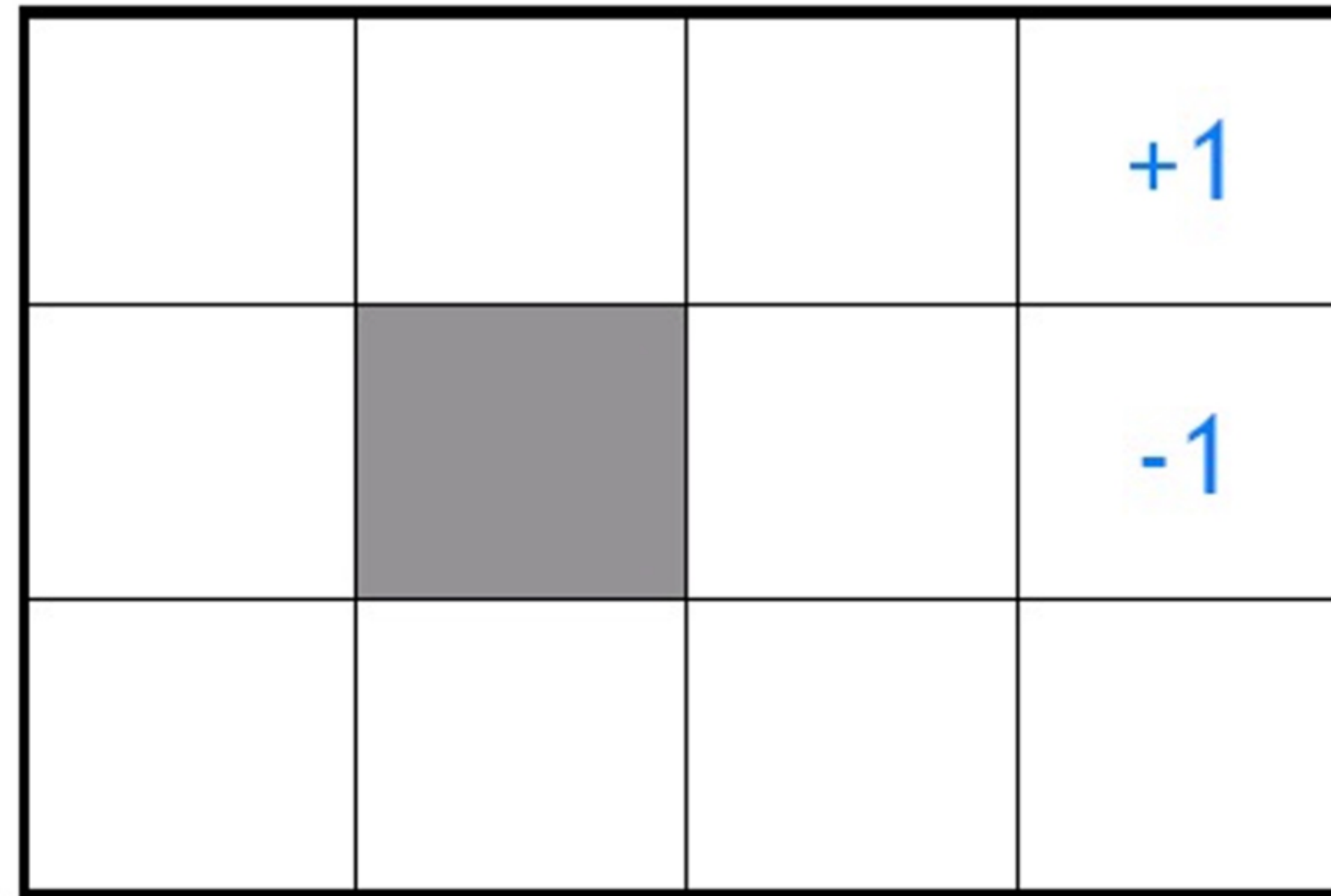
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

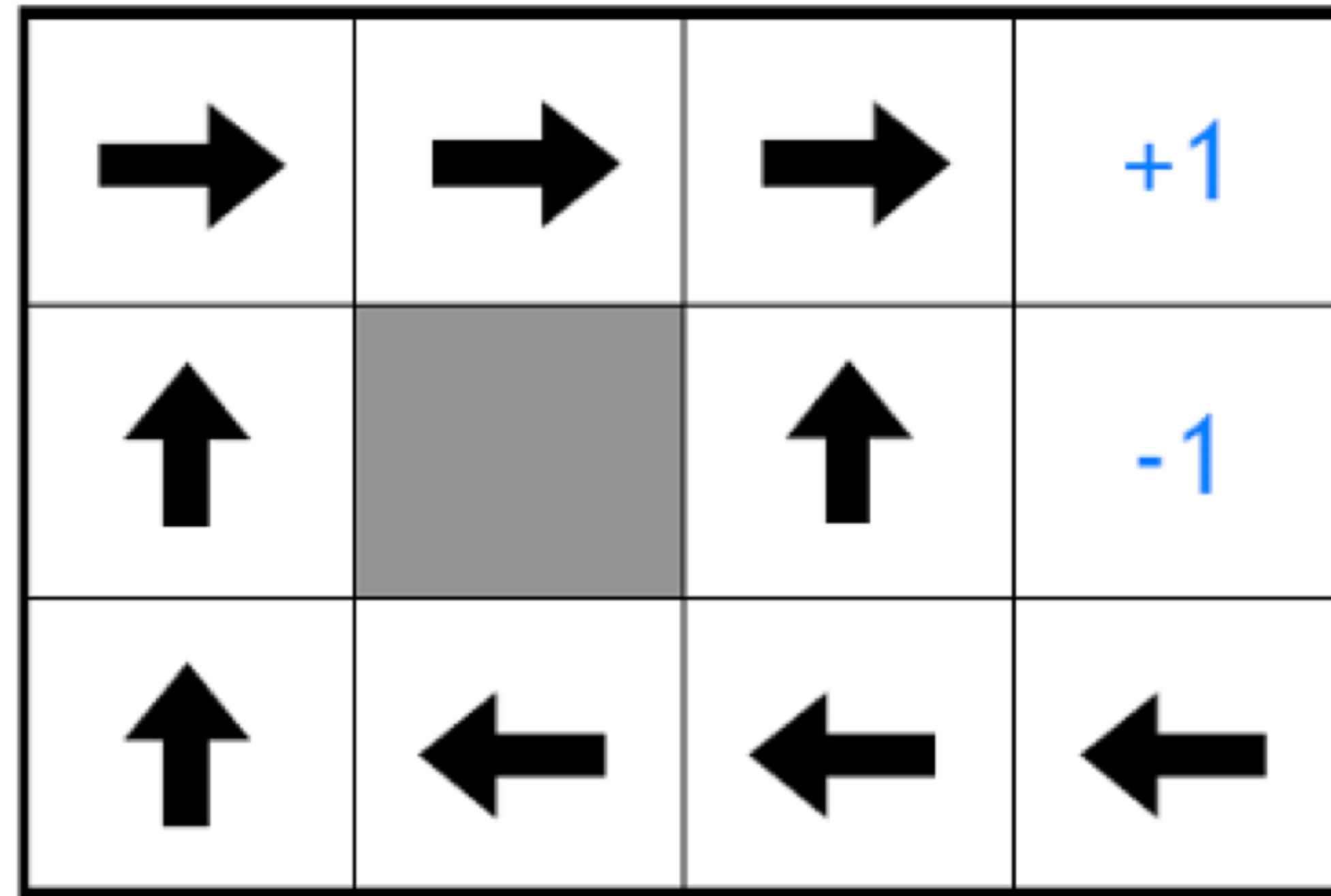
Deterministic transitions

Rewards of +1 and -1 for entering the labelled squares

Terminate after receiving either reward



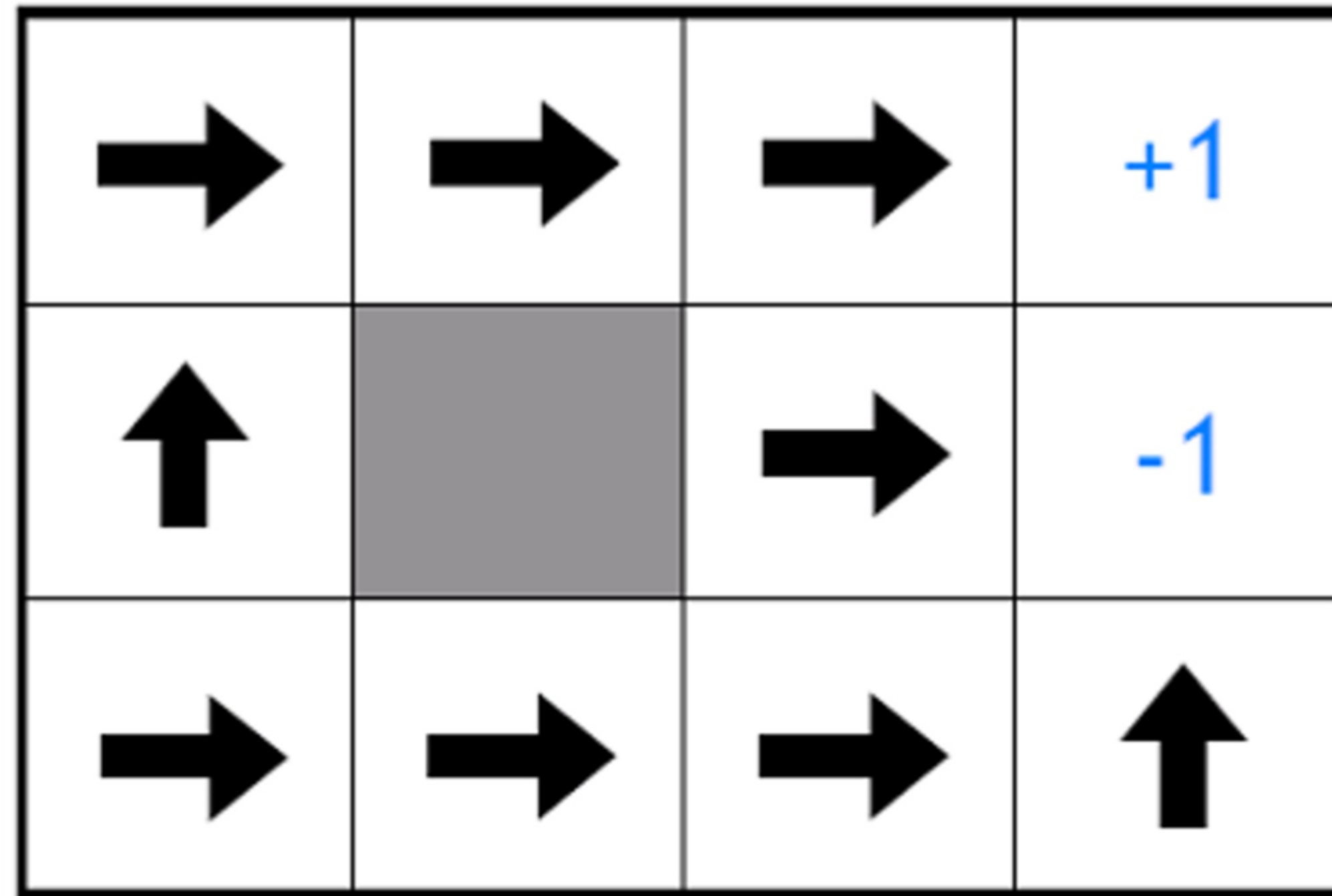
RL Example - gridworld



Is this policy optimal?

RL Example - gridworld

Optimal policy given a reward of -2 per step



Reinforcement Learning in LLMs

Step 1

Collect demonstration data and train a supervised policy.

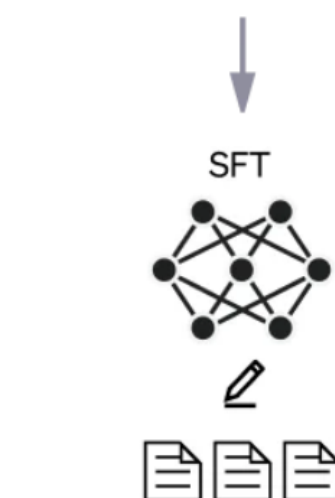
A prompt is sample from our prompt dataset.



A labeler demonstrates the desired output behavior.



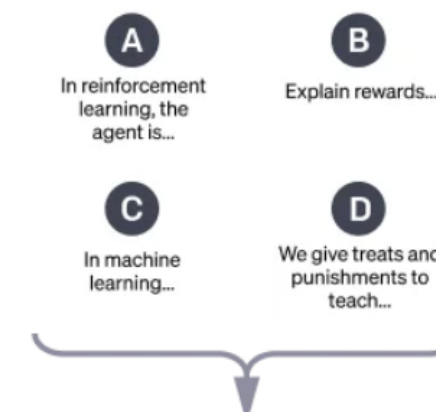
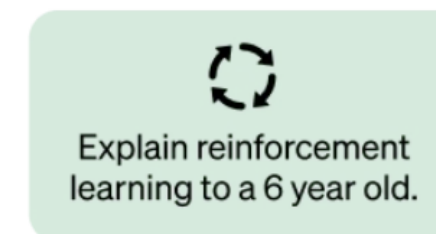
This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

Collect comparison data and train a reward model.

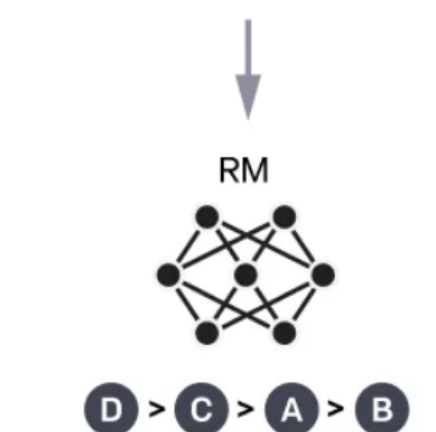
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



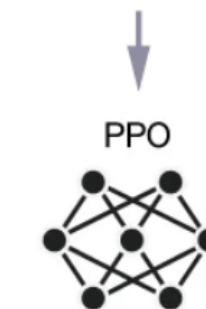
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



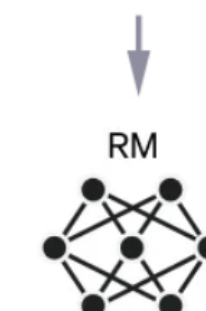
The PPO model is initialized from the supervised policy.



The policy generates an output.

Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

Reinforcement Learning in LLMs

1. There is no external environment
2. Each predicted token is an action, and the context is the environment states

From Imitation to Optimization

Imitation (SFT)

Fit $\hat{p}(y|x) \approx p^*(y|x)$ for some reference distribution $p^*(y|x)$

- Pure generative modeling perspective
- Requires samples from reference policy

Optimization (RLHF)

Find $\hat{p}(y|x)$ such that $\max_p E_p[R(y, x)]$ for a reward $R(y, x)$

- Maximize some reward function that we can measure
 - LMs are policies, not a model of some distribution

Reward Optimization in Language Models

Objective:

$$\theta = \arg \max_{\theta} \mathbb{E}_{x \sim p_{\theta}(x)} R(x)$$

X is the language sequence, R is the reward function, scores the generated response (we can consider R as a human or a model)

How to do gradients over this objective?

Gradient estimation (policy gradient):

$$\hat{g} = \mathbb{E}_{x \sim p_{\theta}(x)} R(x) \nabla_{\theta} \log p_{\theta}(x)$$

$\hat{g}$ is the gradient (not objective)

REINFORCE / Policy Gradient

$$\hat{g} = \mathbb{E}_{x \sim p_{\theta}(x)} R(x) \nabla_{\theta} \log p_{\theta}(x)$$

How to implement?

$$\text{Objective} = \sum_{i=1}^n \frac{1}{n} R(x^{(i)}) \log p_{\theta}(x^{(i)}) \quad x^{(1)}, \dots, x^{(n)} \sim p_{\theta}(x)$$

This objective looks kinda like weighted log likelihood maximization?

What is different?

1. Have a weight of $R(x)$
2. The data x is sampled from the model itself, not from a static dataset

REINFORCE / Policy Gradient

$$\text{Objective} = \sum_{i=1}^n \frac{1}{n} R(x^{(i)}) \log p_{\theta}(x^{(i)}) \quad x^{(1)}, \dots, x^{(n)} \sim p_{\theta}(x)$$

1. Have a weight of $R(x)$
2. The data x is sampled from the model itself, not from a static dataset

This equation is not that complex, just view it as a weighted likelihood maximization

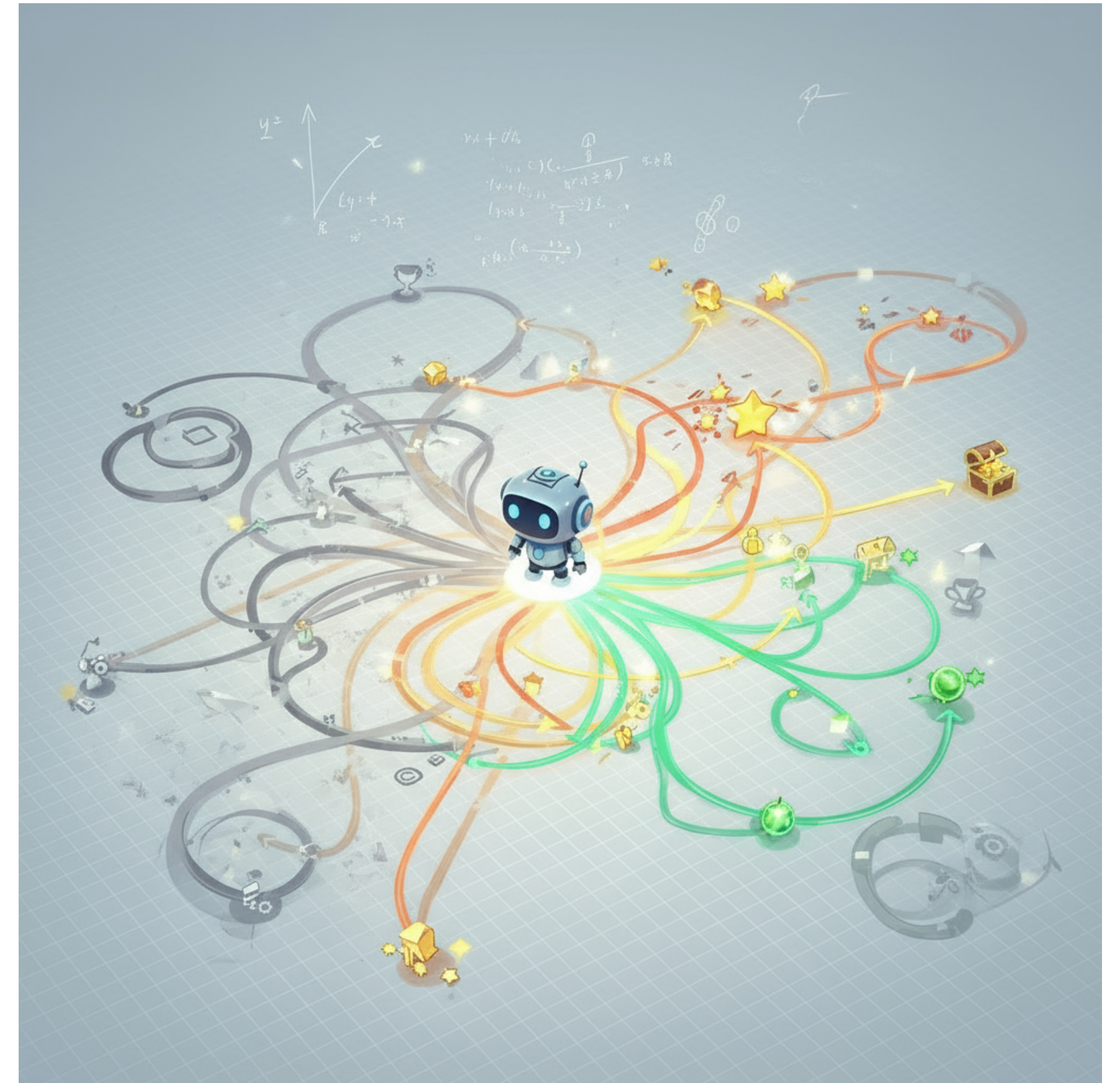
This is the simplest form of RL, many other RL algorithms (PPO, GRPO) are more like variants of this simple equation with the same spirit

We can see why RL is called “self-improving”, and it is trained by “synthetic data”

REINFORCE / Policy Gradient

$$\text{Objective} = \sum_{i=1}^n \frac{1}{n} R(x^{(i)}) \log p_{\theta}(x^{(i)}) \quad x^{(1)}, \dots, x^{(n)} \sim p_{\theta}(x)$$

1. Sample data from the model (or we call policy) itself (exploration)
2. A reward function judges whether the explored data is good or bad

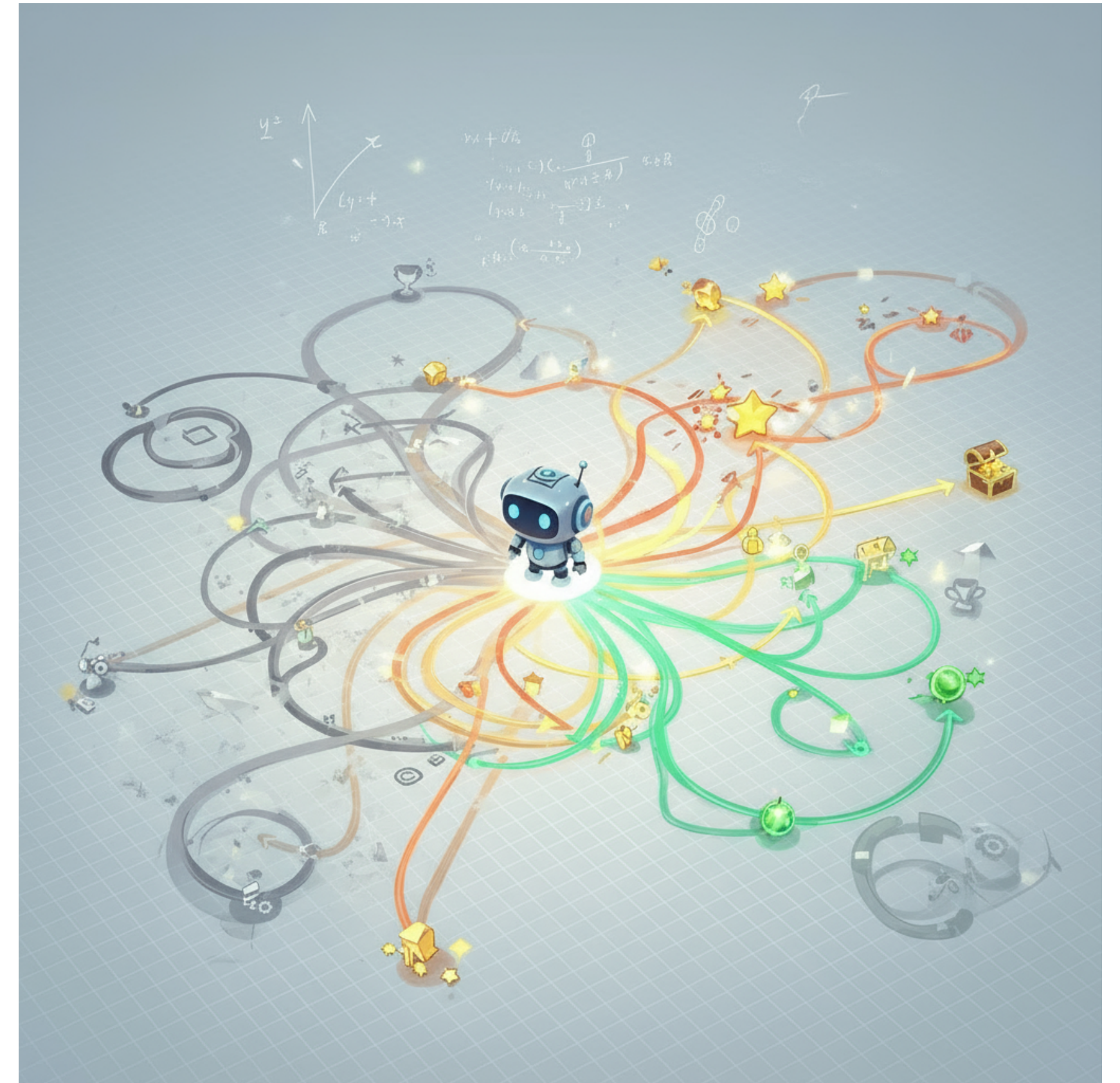


REINFORCE / Policy Gradient

$$\text{Objective} = \sum_{i=1}^n \frac{1}{n} R(x^{(i)}) \log p_{\theta}(x^{(i)}) \quad x^{(1)}, \dots, x^{(n)} \sim p_{\theta}(x)$$

Reinforcement learning is a mixed art of both training and inference during training time

Why?



REINFORCE / Policy Gradient

$$\text{Objective} = \sum_{i=1}^n \frac{1}{n} R(x^{(i)}) \log p_{\theta}(x^{(i)}) \quad x^{(1)}, \dots, x^{(n)} \sim p_{\theta}(x)$$

Training Inference

Each training step, the algorithm needs to run inference again

Is this efficient?

REINFORCE / Policy Gradient for Language Models

$$\text{Objective} = \sum_{i=1}^n \frac{1}{n} R(\{x_1^{(i)}, x_2^{(i)}, x_t^{(i)}\}) \sum_{j=1}^t \log p_{\theta}(x_j^{(i)} | x_{<j}^{(i)})$$

Training a Reward Model in RLHF

Step 1

Collect demonstration data and train a supervised policy.

A prompt is sample from our prompt dataset.



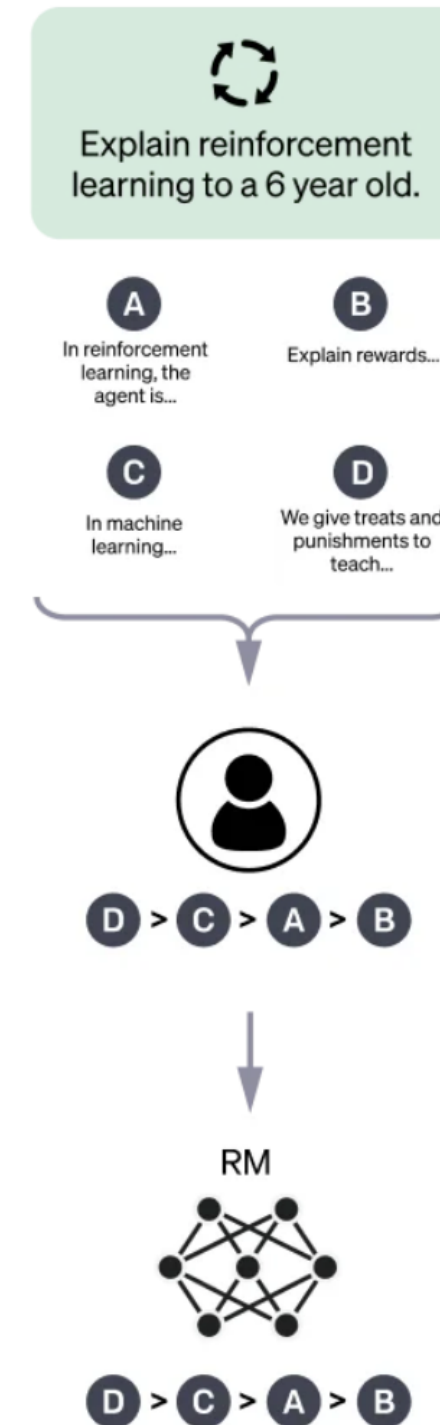
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

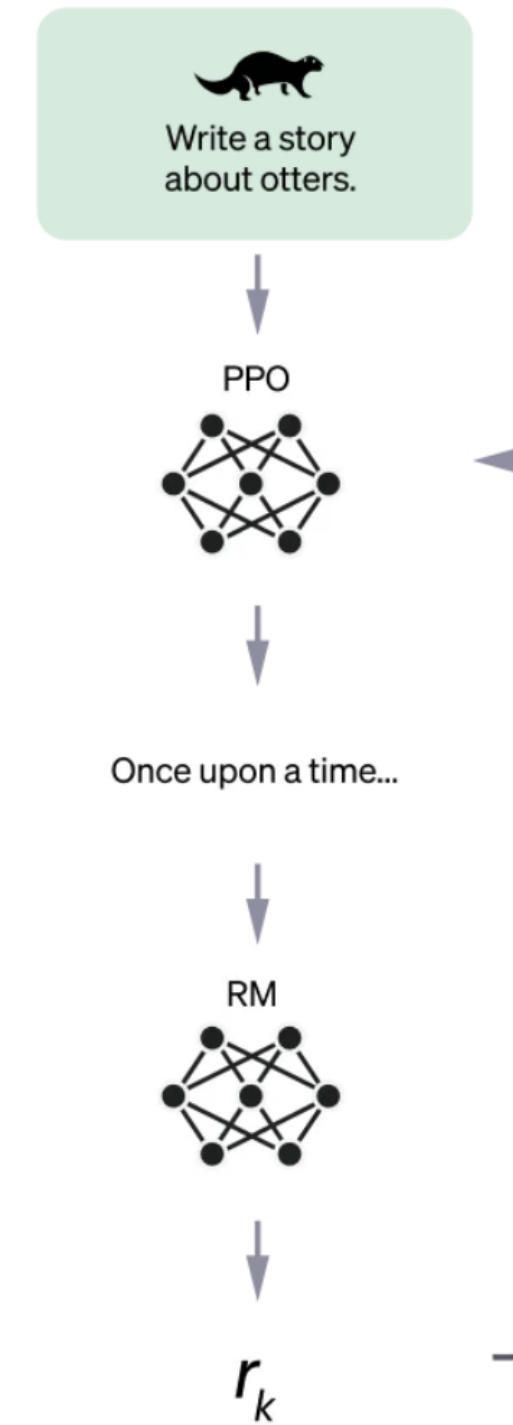
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Training a Reward Model

Suppose we have K responses and have them ranked by humans, then for all possible pairs of responses, y_w is the preferred one, y_l is the less preferred one, the objective of reward model $r_\theta(x, y)$ is:

$$\text{Objective} = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log (\sigma (r_\theta (x, y_w) - r_\theta (x, y_l)))]$$

In practice, we can just sample a subset of pairs, rather than enumerating all pairs

Reinforcement Learning Relaxes Human Annotations

	Base model	Supervised learning (one time)	Pairwise feedback	RL	Evaluation
Compute cost	\$300k	\$100	\$100	\$100	\$0
Annotation cost	\$0	\$25k	\$4k	\$0	\$50

1. SFT data can be expensive
2. There may be tasks that are much easier for humans to verify than solve, particularly when AI today is outperforming humans

Reinforcement Learning from Human Feedback Data

Basic Setup

Hi test22. [Not your MTurk id?](#) [Logout](#)

Your annotation progress for this HIT: 0/6 completed.

Sample 1 of 6.

Lets you skip forward (all samples still need to be marked):

Returns you back to the next un-marked sample to mark:

► **Annotation Guidelines**

Instruction

Tell me about self driving cars

Input

AI Response 1

Self driving cars are vehicles that are capable of detecting their surroundings and navigating without any input from a human driver. They use a variety of sensors, such as cameras, radar, and lidar, to detect objects in their environment and use advanced algorithms to make decisions about how to maneuver around them. Self driving cars are becoming increasingly popular and are being used in a variety of applications, such as ride-hailing services and autonomous delivery services.

AI Response 2

Self driving cars are cars that are equipped with sensors, cameras, and software, which allows them to navigate without the need for a driver. Self driving cars are able to monitor their environment and navigate without the assistance of a driver. Self driving cars use a combination of technologies such as GPS, radar, lidar, and computer vision to detect their environment and avoid obstacles. They are also able to update their maps and adjust their routes to account for variables such as traffic and construction.

The main benefit of self driving cars is their ability to reduce crashes and save lives. Autonomous vehicles have the potential to save up to 30,000 lives annually. They are also more energy efficient than traditional vehicles, as they do not require a driver who has to maintain a minimum speed. Autonomous vehicles also create new opportunities for increased mobility, allowing those who are unable to drive to get around.

Rating

☐ Response 1 is better.

☐ Response 1 is only slightly better. (only pick this if it's truly close)

☐ Response 2 is only slightly better. (only pick this if it's truly close)

☐ Response 2 is better.

Easier than annotating the responses directly

Thank You!