



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

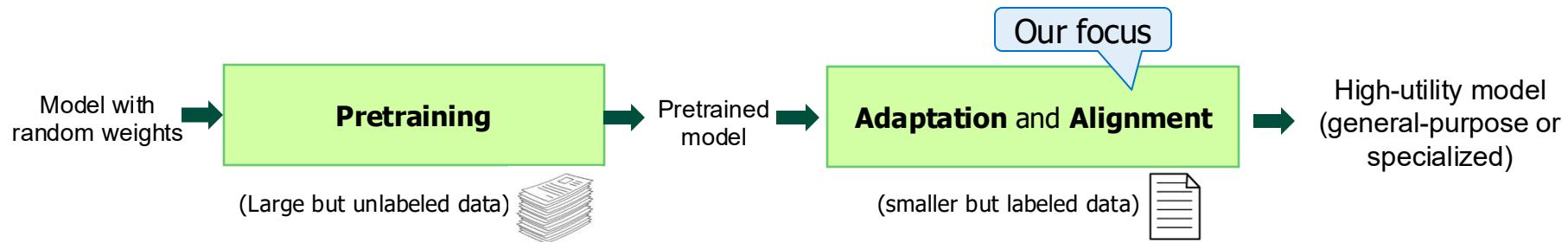
Aligning Self-Supervised Models with Human Intents

CSCI 601-771 (NLP: Self-Supervised Models)

<https://self-supervised.cs.jhu.edu/fa2025/>

[Mis]Alignment in Language Models

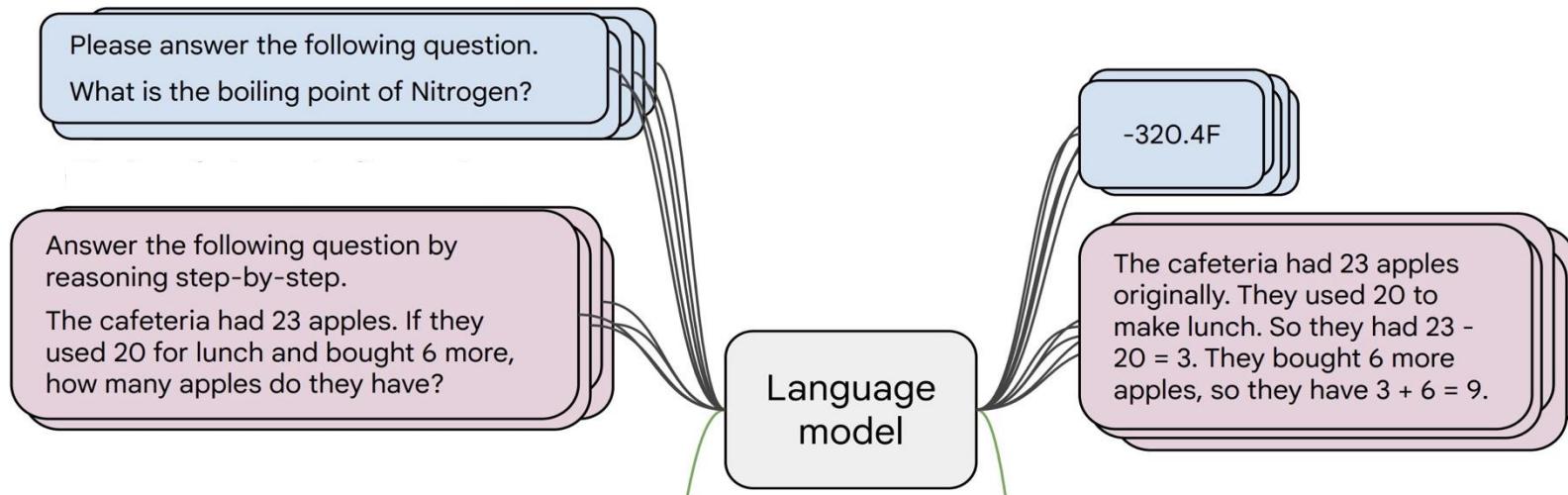
- There is a mismatch between what **pre-trained** models can do and what we want.
- Addressing this gap is the focus of “alignment” research.
- Let’s take a deeper look into what “alignment” is about.



Aligning Language Models: Instruction-tuning

Instruction-tuning

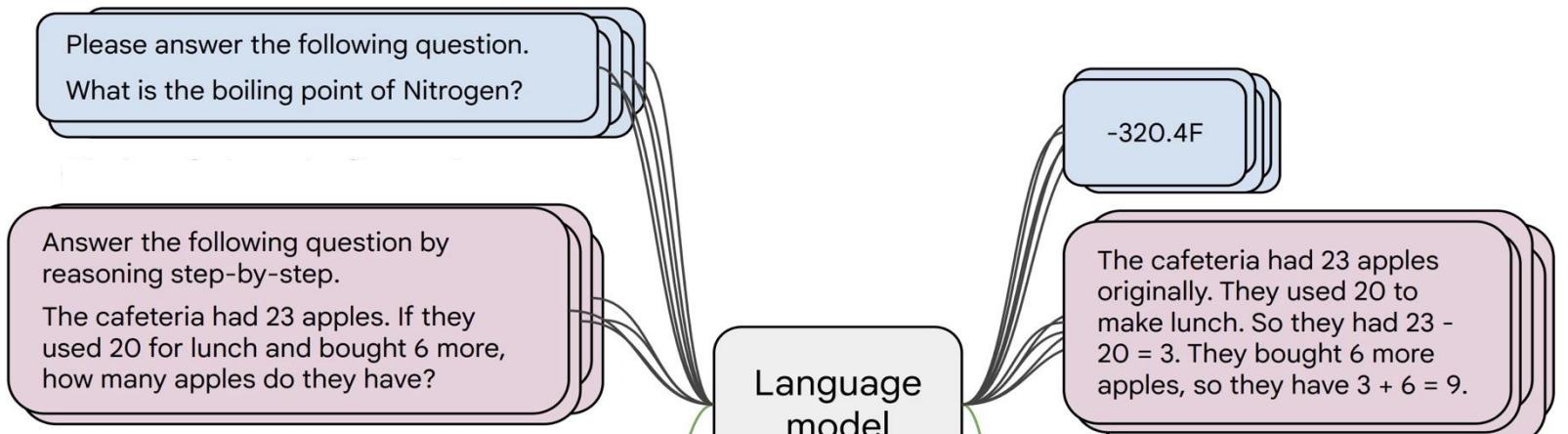
- Finetuning pre-trained LMs to map **instructions** to their corresponding **responses**.



Instruction-tuning

[Weller et al. 2020; Mishra et al. 2021; Wang et al. 2022, Sanh et al. 2022; Wei et al., 2022, Chung et al. 2022, many others]

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



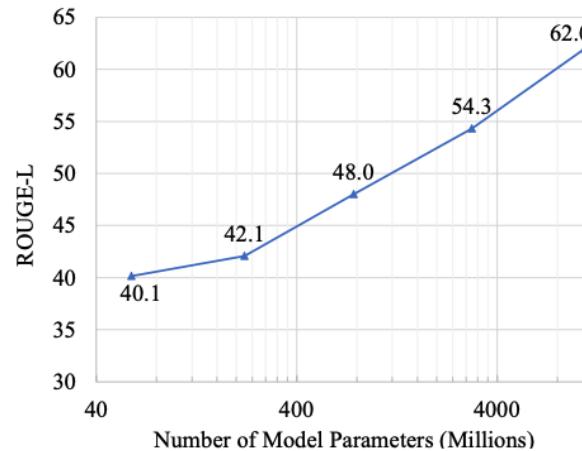
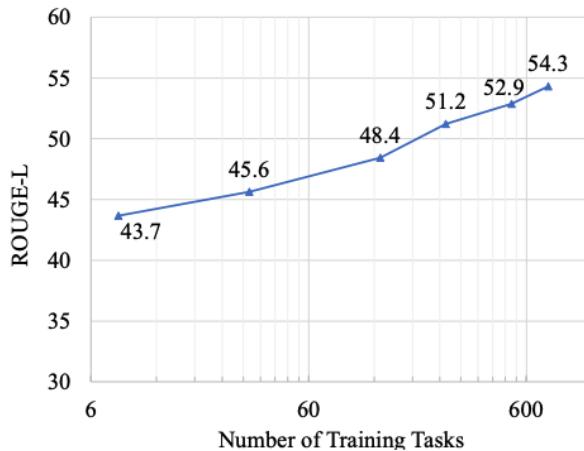
2. Evaluate on **unseen** tasks

Inference: generalization to unseen tasks

Q: Can Geoffrey Hinton have a conversation with George Washington?
Give the rationale before answering.

Geoffrey Hinton is a British-Canadian computer scientist born in 1947. George Washington died in 1799. Thus, they could not have had a conversation together. So the answer is "no".

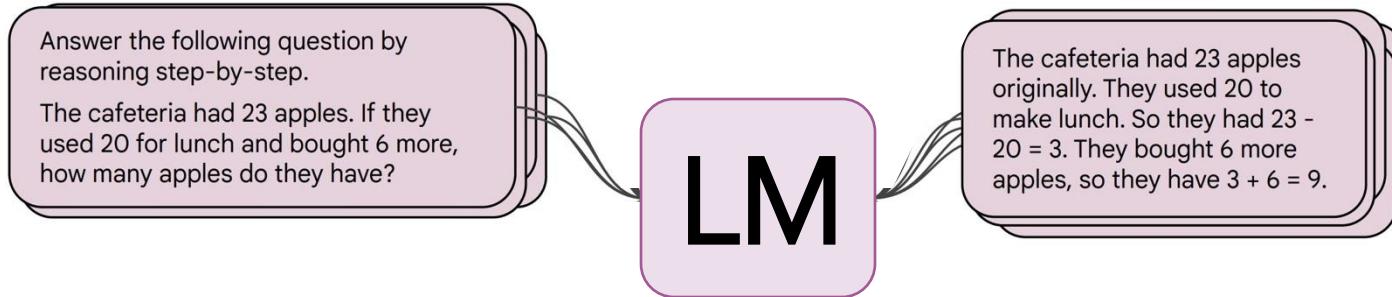
Scaling Instruction-Tuning



Linear growth of model performance
with exponential increase in observed tasks and model size.

Limits of Instruction-Tuning

1. Difficult to collect diverse data.
2. Resulting models may not be good at open-ended generation tasks.
 - o Incentivizes word-by-word rote learning => The resulting LM's **generality/creativity** is bounded by that of **their supervision data**.



Limits of Instruction-Tuning

1. Difficult to collect diverse data.
2. Resulting models may not be good at open-ended generation tasks.
 - Incentivizes word-by-word rote learning => The resulting LM's **generality/creativity** is bounded by that of **their supervision data**.
3. Resulting models may hallucinate more regularly.
 - Labeled data is collected agnostic to the LM's knowledge => there might be a mismatch between labeled data and LM knowledge.
 - Hence, we may be encouraging "hypocritic" behavior => further hallucinations

Aligning Language Models: Reinforcement Learning w/ Human Feedback (RLHF)

Reinforcement Learning: Intuition

Action here: generating responses/token

agent



environment

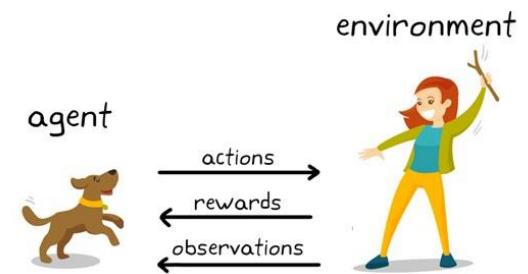
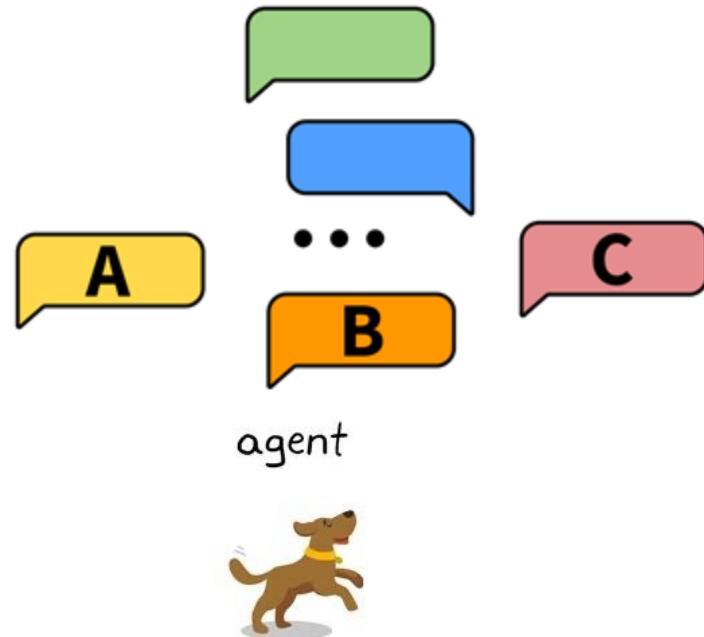


Reward here: whether humans liked the generation (sequence of actions=tokens)

[figure credit]

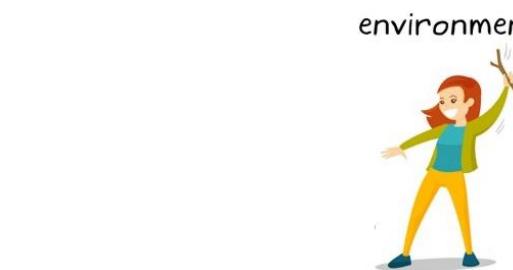
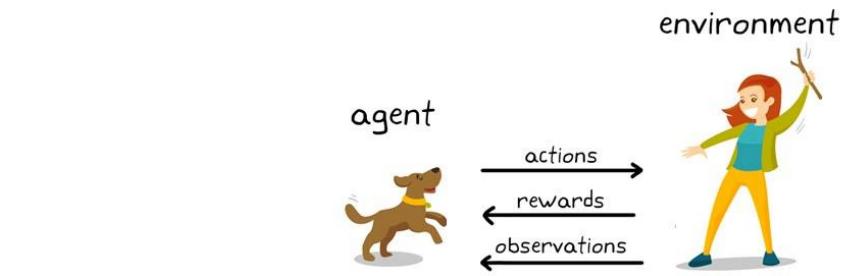
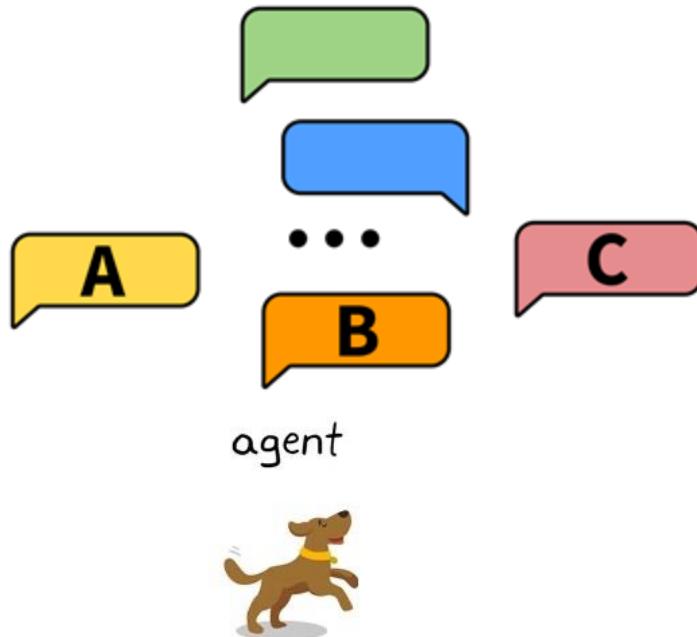
Intuition

Task: choose the better next message in a conversation



Intuition

Scoring interface: Likert scale or rankings



Intuition



Human

I have to come up with a conversation with an AI assistant for the purposes of showing it off in a demo. What sorts of conversation topics would you recommend?



Assistant

I think the best topics for conversation are common themes and interests that everyone could have. Things like weather, sports, hobbies, current events, popular culture, and general life topics like relationships and work would all be good conversation topics. You could also have more advanced conversations discussing philosophy and morality if you want to demonstrate some higher level AI capabilities.



Human

I think I'm more interested in an advanced conversation. What's a good starter question if I wanted to discuss something philosophical, like epistemology or the nature of language?



Assistant

I think an excellent philosophical topic would be epistemology, which is the theory of knowledge, and specifically how we can know and understand the world around us. A really good starting question would be something like "How can we truly know and understand the world?", or "What is the nature of knowledge?". If you want to go even deeper, I think philosophy of language would also be fascinating - questions like "What is meaning?", "How do language and thought relate?", and "How can language mislead and deceive?" would be great discussion topics.

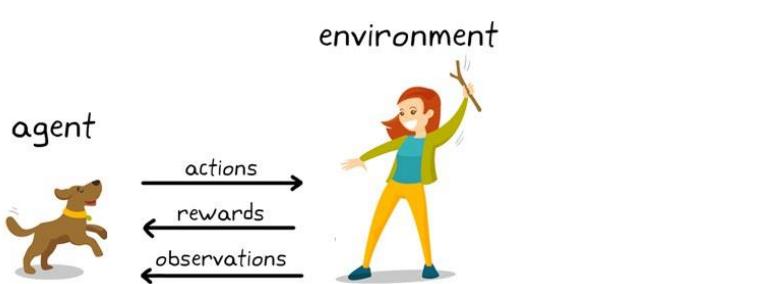


human has conversation with the LLM



Intuition

LLM provides two options for next responses



Human
I have to come up with a conversation with an AI assistant for the purposes of showing it off in a demo. What sorts of conversation topics would you recommend?

Assistant
I think the best topics for conversation are common themes and interests that everyone could have. Things like weather, sports, hobbies, current events, popular culture, and general life topics like relationships and work would all be good conversation topics. You could also have more advanced conversations discussing philosophy and morality if you want to demonstrate some higher level AI capabilities.

Human
I think I'm more interested in an advanced conversation. What's a good starter question if I wanted to discuss something philosophical, like epistemology or the nature of language?

Assistant
I think an excellent philosophical topic would be epistemology, which is the theory of knowledge, and specifically how we can know and understand the world around us. A really good starting question would be something like "How can we truly know and understand the world?", or "What is the nature of knowledge?". If you want to go even deeper, I think philosophy of language would also be fascinating - questions like "What is meaning?", "How do language and thought relate?", and "How can language mislead and deceive?" would be great discussion topics.

Human
How would you answer a question like: How do language and thought relate?

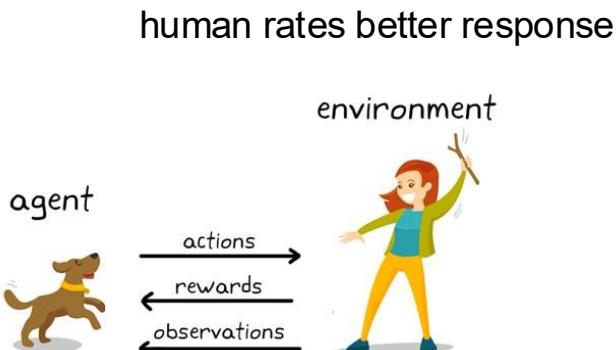
Choose the most helpful and honest response

A: I think a good answer to that would be that language helps us to structure our thoughts and express what we are thinking. Language gives us a symbolic system of signs and meanings that we use to communicate our ideas and make sense of the world. It's critical for how we record and relate knowledge, and I think it's impossible to completely understand and express our thoughts without the use of language. Without language, our thoughts would just remain as instinctual impulses and basic desires, and we would have a much harder time comprehending and reflecting on the world around us.

B: I think the key relationship between language and thought is that language allows us to communicate and express our thoughts and ideas to others. Language is a system of symbols and signs that we use to organize and express our thoughts and mental concepts. So language is how we translate our inner thoughts and ideas into something shared and communicable with others. Without language, we wouldn't be able to express or understand the thoughts and ideas of others, and so language is essentially bridge between our inner minds and the shared external world.

A A A A B B B
A is better B is better

Intuition



- Human**
I have to come up with a conversation with an AI assistant for the purposes of showing it off in a demo. What sorts of conversation topics would you recommend?
- Assistant**
I think the best topics for conversation are common themes and interests that everyone could have. Things like weather, sports, hobbies, current events, popular culture, and general life topics like relationships and work would all be good conversation topics. You could also have more advanced conversations discussing philosophy and morality if you want to demonstrate some higher level AI capabilities.
- Human**
I think I'm more interested in an advanced conversation. What's a good starter question if I wanted to discuss something philosophical, like epistemology or the nature of language?
- Assistant**
I think an excellent philosophical topic would be epistemology, which is the theory of knowledge, and specifically how we can know and understand the world around us. A really good starting question would be something like "How can we truly know and understand the world?", or "What is the nature of knowledge?". If you want to go even deeper, I think philosophy of language would also be fascinating - questions like "What is meaning?", "How do language and thought relate?", and "How can language mislead and deceive?" would be great discussion topics.
- Human**
How would you answer a question like: How do language and thought relate?

Choose the most helpful and honest response

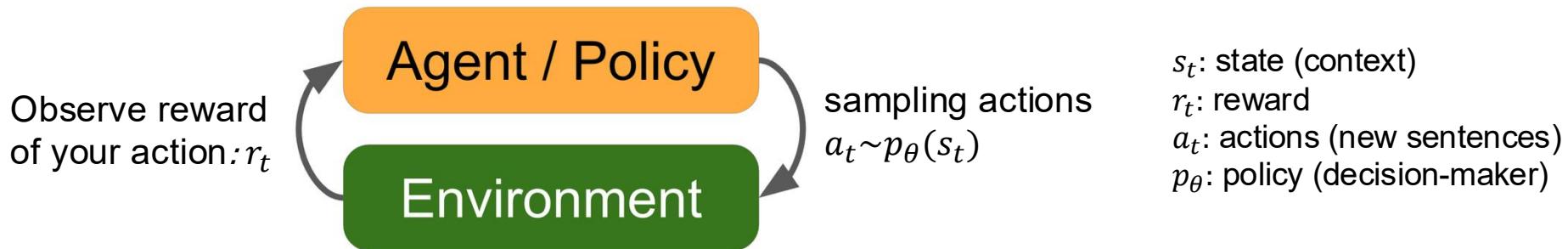
A: I think a good answer to that would be that language helps us to structure our thoughts and express what we are thinking. Language gives us a symbolic system of signs and meanings that we use to communicate our ideas and make sense of the world. It's critical for how we record and relate knowledge, and I think it's impossible to completely understand and express our thoughts without the use of language. Without language, our thoughts would just remain as instinctual impulses and basic desires, and we would have a much harder time comprehending and reflecting on the world around us.

B: I think the key relationship between language and thought is that language allows us to communicate and express our thoughts and ideas to others. Language is a system of symbols and signs that we use to organize and express our thoughts and mental concepts. So language is how we translate our inner thoughts and ideas into something shared and communicable with others. Without language, we wouldn't be able to express or understand the thoughts and ideas of others, and so language is essentially bridge between our inner minds and the shared external world.

A is better **B is better**

Reinforcement Learning: Formalism

- An agent **interacts** with an environment by taking **actions**
- The environment returns a **reward** for the **action** and a **new state** (representation of the world at that moment).
- Agent uses a **policy function** to choose an action at a given **state**.
- We need to figure out: (1) reward function and (2) the policy function



Reinforcement Learning from Human Feedback



- Imagine a reward function: $R(s; \text{prompt}) \in \mathbb{R}$ for any output s to a prompt.
- The reward is higher when humans prefer the output.
- Good generation is equivalent to finding reward-maximizing outputs:

Expected reward over the course of sampling from our policy (generative model)

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

$p_{\theta}(s)$ is a pre-trained model with params θ we would like to optimize (policy function)

- On the notation:
 - “ \mathbb{E} ” here is an empirical expectation (i.e., average).
 - “ \sim ” indicates sampling from a given distribution.

Reinforcement Learning from Human Feedback



- Imagine a reward function: $R(s; \text{prompt}) \in \mathbb{R}$ for any output s to a prompt.
- The reward is higher when humans prefer the output.
- Good generation is equivalent to finding reward-maximizing outputs:

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

- What we need to do:
 - (1) Estimate the reward function $R(s; \text{prompt})$.
 - (2) Find the best generative model p_{θ} that maximizes the expected reward:

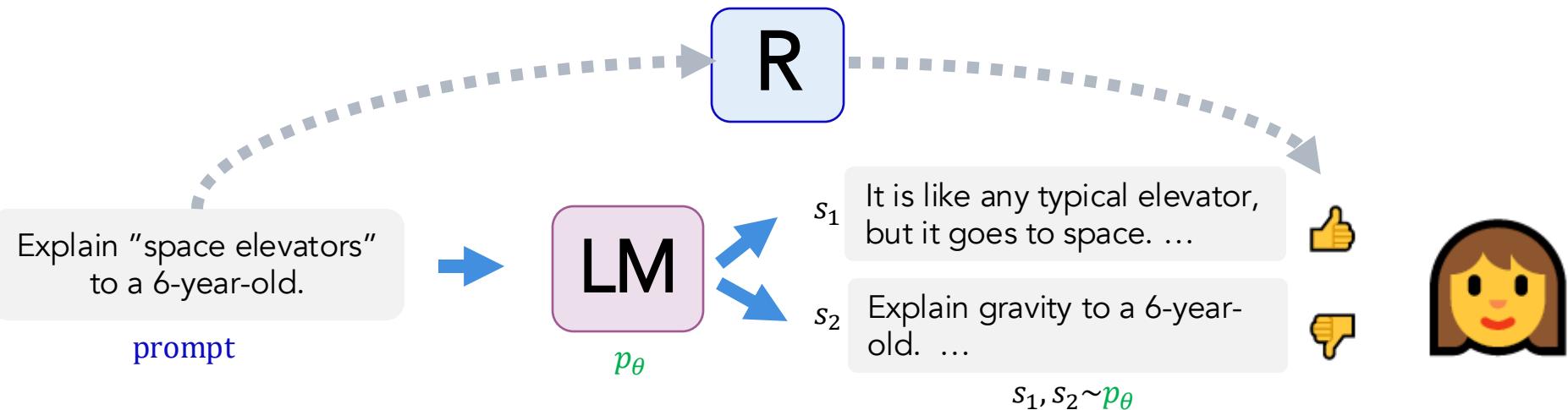
$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

Step 1: Estimating the Reward R



$$J(\phi) = -\mathbb{E}_{(s^+, s^-)} [\log \sigma(R(s^+; \text{prompt}) - R(s^-; \text{prompt}))]$$

“winning” sample “losing” sample

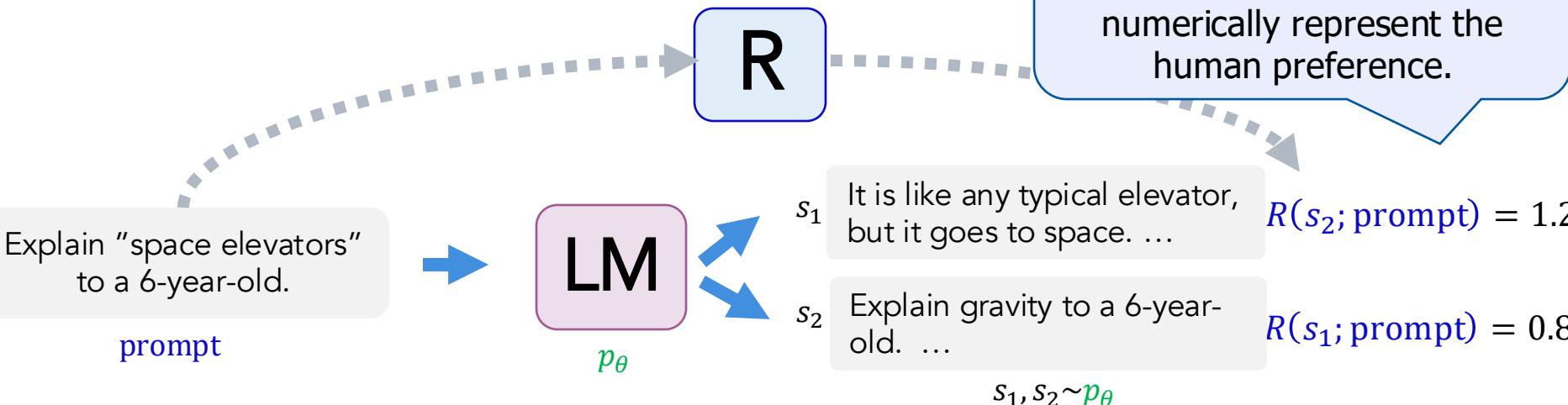


Step 1: Estimating the Reward R



$$J(\phi) = -\mathbb{E}_{(s^+, s^-)} [\log \sigma(R(s^+; \text{prompt}) - R(s^-; \text{prompt}))]$$

“winning” sample “losing” sample

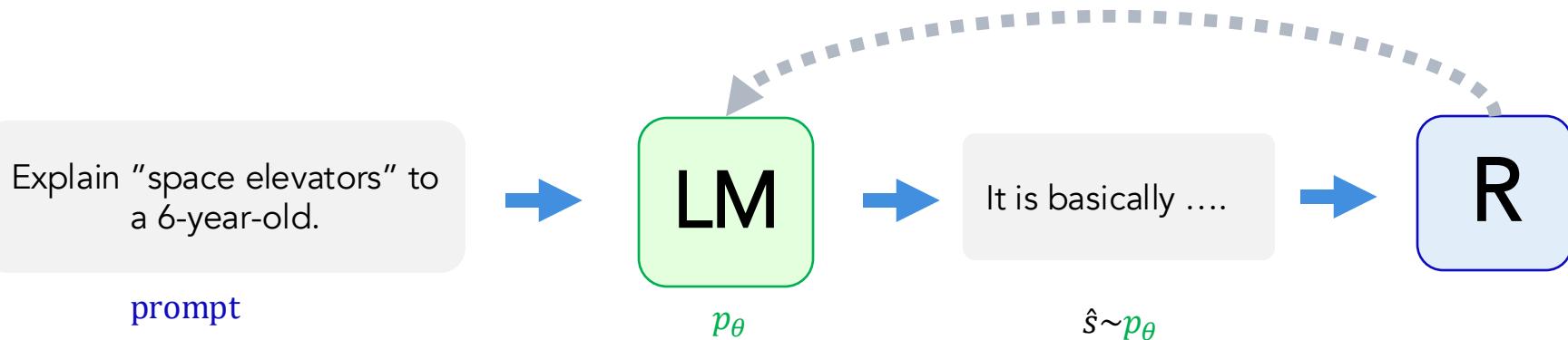


Step 2: Optimizing the Policy Function



- Policy function := The model that makes decisions (here, generates responses)
- How do we change our LM parameters θ to maximize this?

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$



Step 2: Optimizing the Policy Function



- Policy function := The model that makes decisions (here, generates responses)
- How do we change our LM parameters θ to maximize this?

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

- Let's try doing gradient ascent!

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

How do we estimate the gradient of this expectation?

Notice that R is not directly dependent on θ . (You can't compute its grad with respect to θ)

- Turns out that we can write this “gradient of expectation” to a simpler form.

Policy Gradient [Williams, 1992]



- How do we change our LM parameters θ to maximize this?

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

- Let's try doing gradient ascent!

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

- With a bit of math, this can be approximated as Monte Carlo samples from

$$\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; \text{prompt})] \approx \frac{1}{n} \sum_{i=1}^n R(s_i; \text{prompt}) \nabla_{\theta} \log p_{\theta}(s_i; \text{prompt})$$

Proof next slide; check it later in your own time!

- This is “**policy gradient**”, an approach for estimating and optimizing this objective.
- Oversimplified. For full treatment of RL see [701.741](#) course other [RL textbooks](#).

Derivations (check it later in your own time!)



- Let's compute the gradient:

$$\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p)] = \nabla_{\theta} \sum_s p_{\theta}(s) R(s; p) = \sum_s R(s; p) \cdot \nabla_{\theta} p_{\theta}(s)$$

Def. of "expectation" Gradient distributes over sum

- Log-derivative trick $\nabla_{\theta} p_{\theta}(s) = p_{\theta}(s) \cdot \nabla_{\theta} \log p_{\theta}(s)$ to turn sum back to expectation:

$$\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p)] = \sum_s R(s; p) p_{\theta}(s) \nabla_{\theta} \log p_{\theta}(s) = \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p) \nabla_{\theta} \log p_{\theta}(s)]$$

Log-derivative trick

- Approximate this expectation with Monte Carlo samples from $p_{\theta}(s)$:

$$\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p)] \approx \frac{1}{n} \sum_{i=1}^n R(s_i; p) \nabla_{\theta} \log p_{\theta}(s_i)$$

Policy Gradient [Williams, 1992]



- This gives us the following update rule:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{|samples| \times |\text{prompts}|} \sum_{p \in \text{prompts}} \sum_{s_i \sim p_\theta(p)} R(s_i; p) \nabla_\theta \log p_\theta(s_i; p)$$

Note, $R(s; \text{prompt})$ could be any arbitrary, non-differentiable reward function that we design.

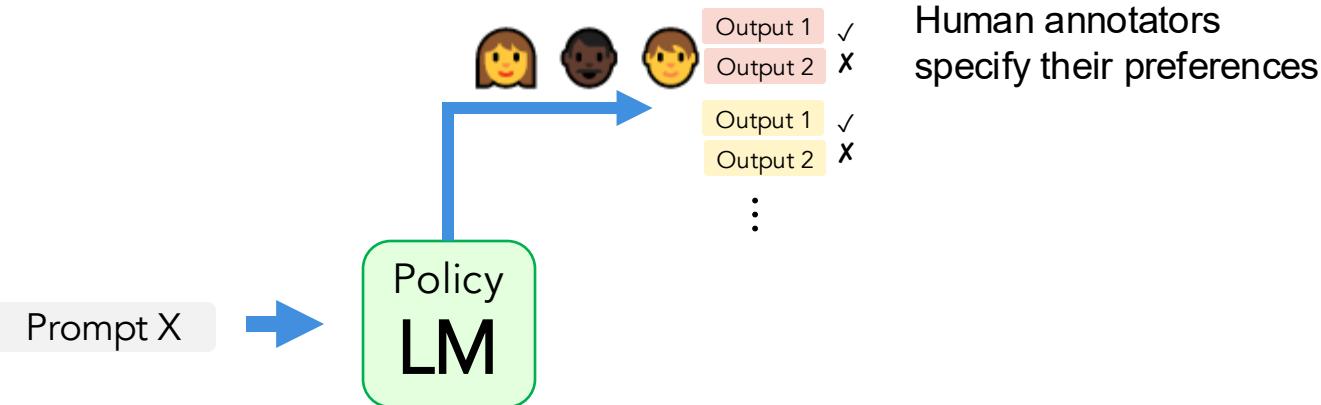
- If $R(s; p)$ is **large**, we take proportionately **large** steps to maximize $p_\theta(s)$
- If $R(s; p)$ is **small**, we take proportionately **small** steps to maximize $p_\theta(s)$

This is why it's called “reinforcement learning”: we reinforce good actions, increasing the chance they happen again.

Putting it Together



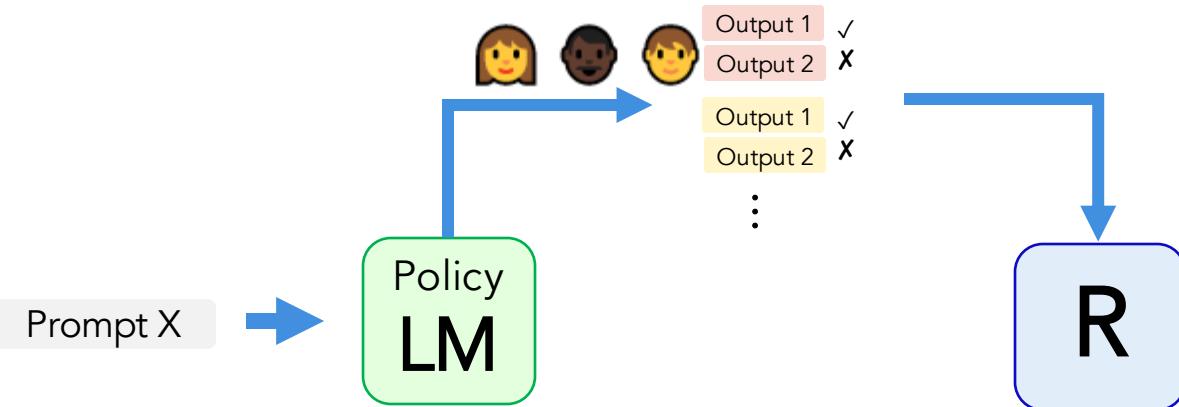
- First collect a dataset of human preferences
 - Present multiple outputs to human annotators and ask them to rank the output based on preferability



Putting it Together (2)



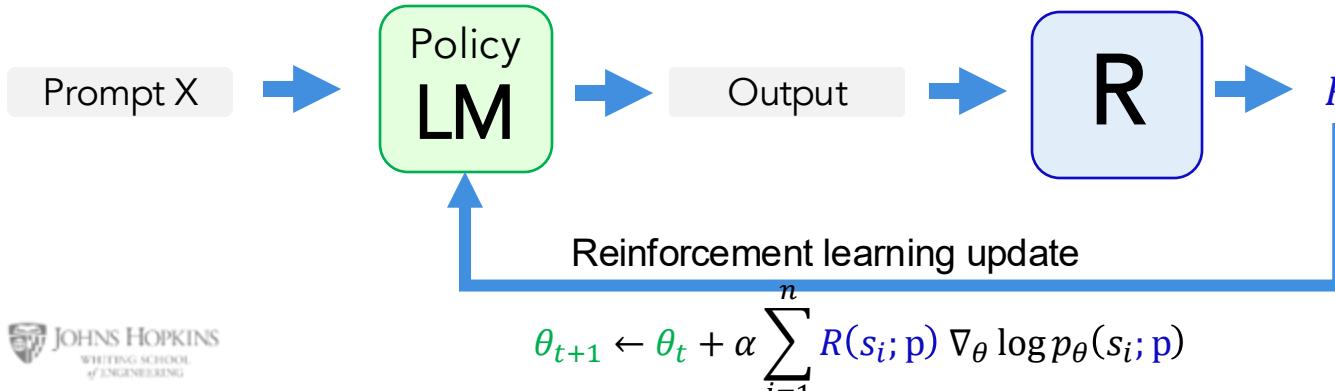
- Using this data, we can train a reward model
 - The reward model returns a scalar reward which should numerically represent the human preference.



Putting it Together (3)



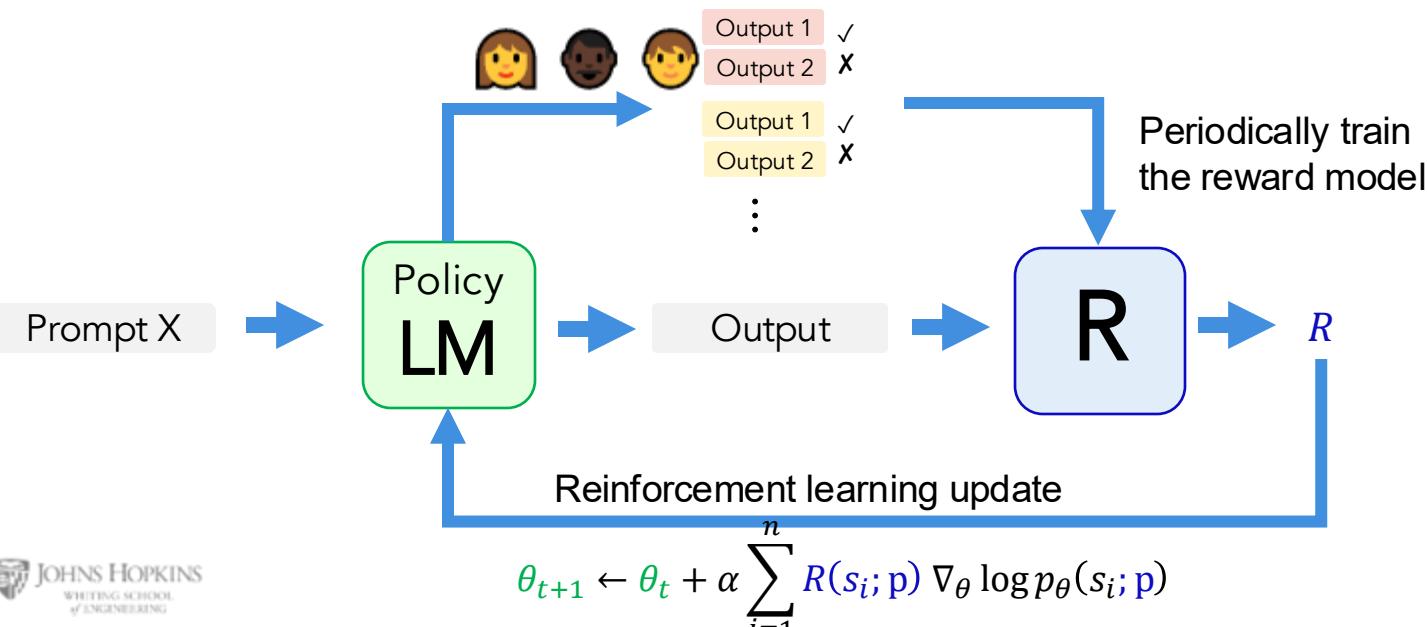
- We want to learn a policy (a Language Model) that optimizes against the reward model



Putting it Together (4)



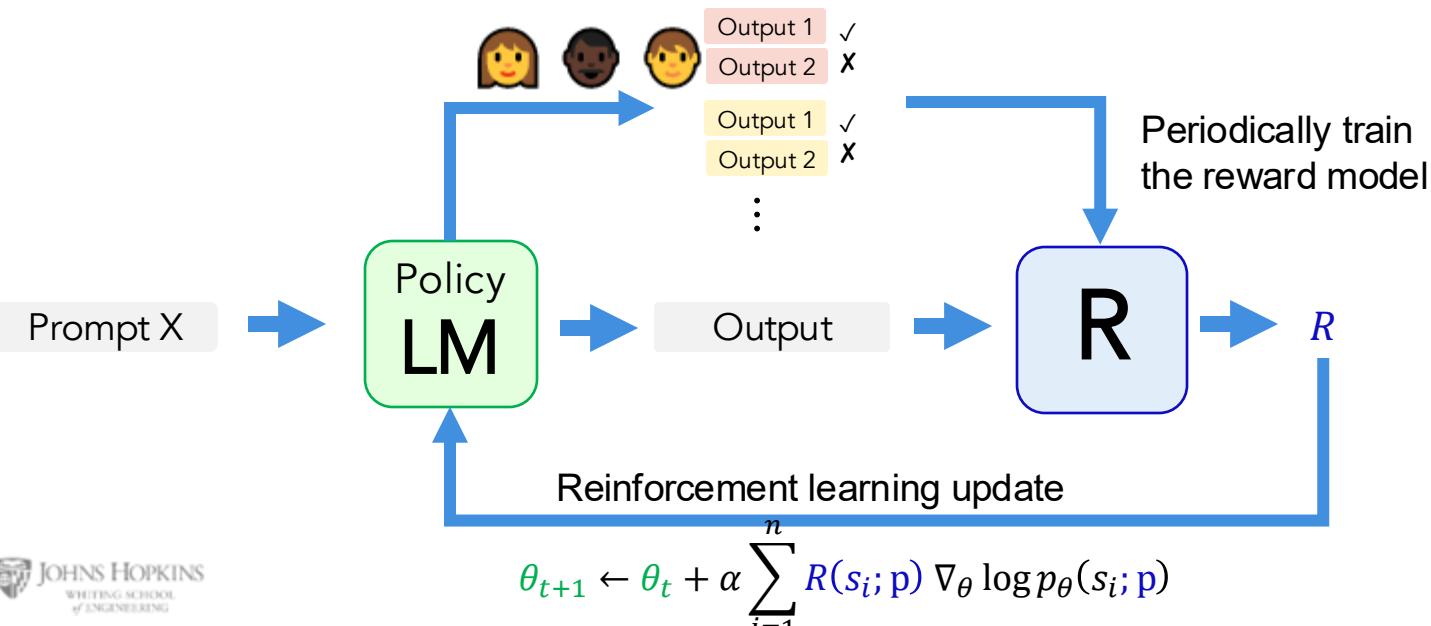
- Periodically train the reward model with more samples and human feedback



One missing ingredient



- It turns out that this approach doesn't quite work. (Any guesses why?)
 - The policy will learn to "cheat".

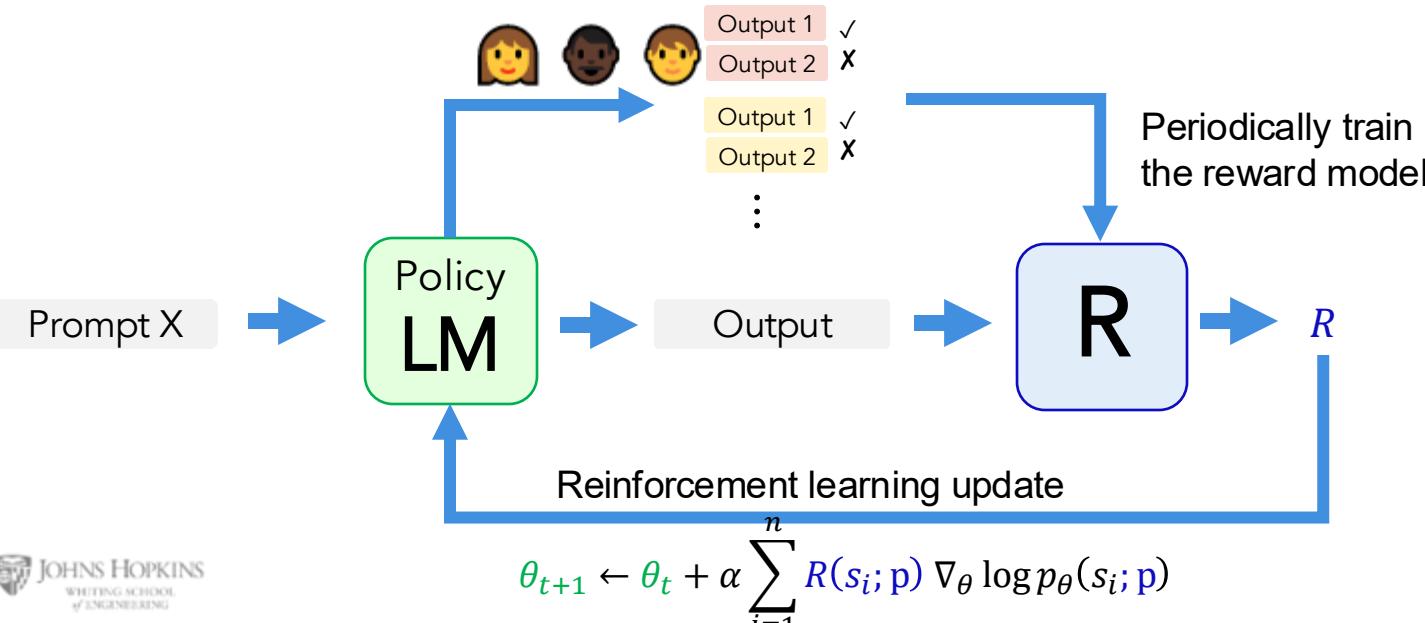


One missing ingredient

How do you resolve this?



- Will learn to produce an output that would get a **high** reward but is **gibberish** or **irrelevant** to the prompt.
- Note, since $R(s; p)$ is trained on natural inputs, it may not generalize to unnatural inputs.



Regularizing with Pre-trained Model

- **Solution:** add a penalty term that penalizes too much deviations from the distribution of the pre-trained LM.

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

- This prevents the policy model from diverging too far from the pretrained model.
 - $p^{RL}(s) < > p^{PT}(s)$: Pay an *explicit* price
 - $p^{RL}(s) << p^{PT}(s)$: Sampling s becomes unlikely
- The above regularization is *equivalent* to adding a KL-divergence regularization term. You will prove the details in HW7!!

Putting it All Together: RLHF as a Basic Policy Gradient

1. Select a pre-trained generative model p_{θ}^{RL} as your base: $p_{\theta}^{PT}(s)$
2. Build a reward model $R(s; p)$ that produces scalar rewards for outputs, trained on a dataset of human comparisons
3. Regularize the reward function:
4. Iterate:

$$\hat{R}(s; p) \coloneqq R(s; p) - \beta \log \left(\frac{p_{\theta}^{RL}(s)}{p_{\theta}^{PT}(s)} \right)$$

1. Fine-tune the policy $p_{\theta}^{RL}(s)$ to maximize our reward model $R(s; p)$

$$\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n} \sum_{i=1}^n \hat{R}(s; p) \nabla_{\theta} \log p_{\theta}^{RL}(s)$$

2. Occasionally repeat steps 2-3 to update the reward model.

The overall recipe



Summary: RLHF with Simple Policy Gradient

- RL can help mitigate some of the problems with supervised instruction tuning
- RLHF uses two models
 - Reward model is trained via ranking feedback of humans.
 - Policy model learns to generate responses that maximize the reward model.
- People may loosely refer to this as “PPO”, though PPO has a more concrete definition. (forthcoming)
- Limitations:
 - RL can be tricky to get right
 - Training a good reward may require a lot of annotations



What do people *actually* use?

What is the Standard?

Language Model	Release	Base	Alignment Algorithm Used	Training Data Sources for alignment
GPT-3-instruct	2020	GPT-3	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
GPT-4	2023	GPT-4 pre-trained?	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
Gemini	2023	Gemini pre-trained?	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
LLaMA2	2023	LLaMA2 pre-trained	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
LLaMA3	2024	LLaMA3 pre-trained	Iterate: Rejection sampling SFT --> DPO	DPO and GRPO: Forthcoming examples. conducted over multiple rounds, with each round involving the collection of new preference annotations and SFT data.
Alpaca	2023	LLAMA 1	SFT	Self-Instruct, 52,000 input-output pairs
Qwen2.5	2024	Qwen2.5 pre-trained	SFT --> DPO --> GRPO	1 million samples
Tulu 3	2024	Llama 3.1	SFT --> DPO --> RLVR	near 1 million samples
DeepSeek (V3)	2024	DeepSeek pre-trained	SFT --> GRPO	1.5 million samples (reasoning + non-reasoning tasks). Reasoning data was generated by specialized models. Non-reasoning data was produced by DeepSeek-V2.5 and validated by human reviewers.

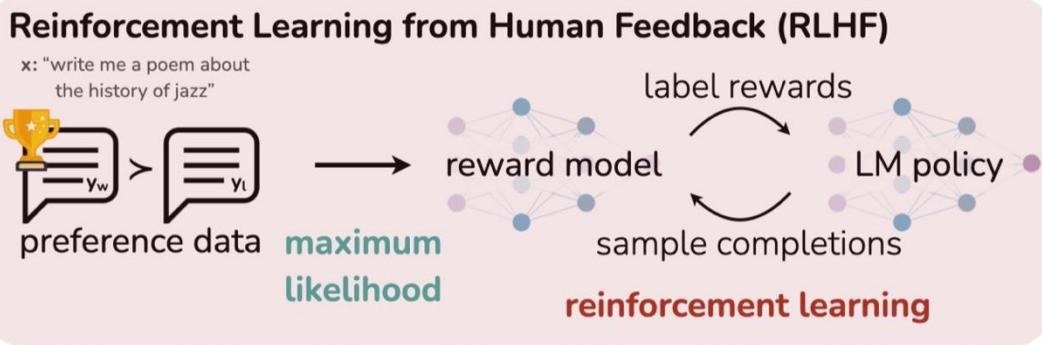
We just saw

DPO and GRPO:
Forthcoming

Aligning Language Models: Direct Policy Optimization

Simplifying RLHF

- The RLHF pipeline is considerably more complex than supervised learning
 - Involves training multiple LMs and sampling from the LM policy in the loop of training
- Is there a way to simplify this pipeline?
 - For example, by using a **single** language model



Direct Policy Optimization (DPO) - Intuition

- DPO directly optimizes for human preferences
 - avoiding RL and fitting a separate reward model
- One can use mathematical derivations to simplify the RLHF objective to an **equivalent** objective that is **simpler** to optimize.

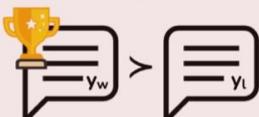
RLHF objective



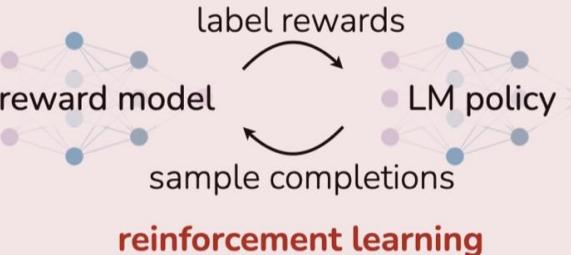
DPO objective

Reinforcement Learning from Human Feedback (RLHF)

x: "write me a poem about
the history of jazz"



maximum
likelihood



Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"



maximum
likelihood



RLHF objectives

y_w : preferred response / y_l : dispreferred response

(i) Reward objective $\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))]$

(ii) Policy objective $\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi_\theta(y | x) || \pi_{\text{ref}}(y | x)]$

Maximizing the reward of the generated prompts

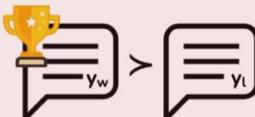
Minimizing the deviation from the base policy

DPO objective $\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$

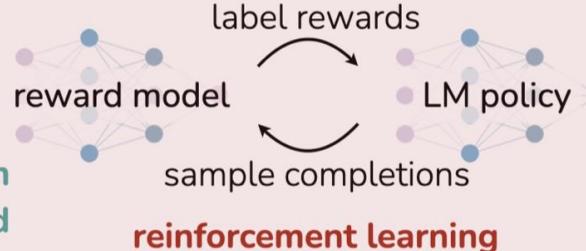
(1) Maximizing reward of the pref response vs that of dispref one; (2) Minimizing deviations from the base policy

Reinforcement Learning from Human Feedback (RLHF)

x: "write me a poem about
the history of jazz"



maximum likelihood



Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"



maximum likelihood



DPO Algorithm

- Algorithm:
 - Sample completions for every prompt
 - Label with human preferences and construct dataset
 - Optimize the language model to minimize the DPO objective.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

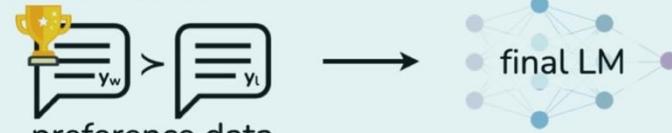
- Note, in practice we can use a dataset of preferences publicly available (for example, responses in forums).

Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"



maximum likelihood



Wait what happened to the KL divergence?

- RLHF obj has a KL regularization term (KL of target and ref policy).
- We don't see it in DPO. **Where did it go???**
- The KL is still there, but it's implicit.
- Notice the two log-ratios; these operate as KL:
 - If y is very low prob ($\pi(y) \approx 0$), it'd never be in your dataset.
 - If there are y 's that are likely to be sampled,
 - DPO will want to increase the gap between log-ratios.
 - Because of $\sigma(\cdot)$ the difference $\Delta^+ - \Delta^-$ will saturate—there's **no extra incentive** to keep pushing.

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E} \left[\log \sigma(\beta(\Delta^+ - \Delta^-)) \right], \quad \Delta(y) = \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}.$$
$$\text{KL}(\pi \| \pi_{\text{ref}}) = \mathbb{E}_{y \sim \pi} [\Delta(y)].$$

Quiz

- You're aligning your model with DPO.

Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"



preference data

maximum likelihood



$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

What could go wrong?

DPO Limitations

- You're trying to optimize multiple things which can potentially **override** each other.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

- Obj 1: Increase the likelihood gap between $\pi_\theta(y_w | x)$ and $\pi_\theta(y_l | x)$
 - Obj 2: Maintain a low gap between $\pi_\theta(y_w | x)$ and $\pi_{\text{ref}}(y_w | x)$
 - ...
- We will look into these in HW7!
 - In practice, when using DPO practitioners constantly monitor these to be sure that they're not overriding each other.

Direct Preference Optimization (DPO)

x: "write me a poem about the history of jazz"



maximum likelihood

DPO: Derivation

- Start with the RLHF objective, which assumes having a reward model:

$$\max_{\pi_\theta} E_{x \sim D, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta \cdot \text{KL} [\pi_\theta(y|x) || \pi_{\text{ref}}(y|x)]$$

- Assume that the policy π_θ is the set of all policies (nonparametric assumptions). Then the minimizer of the above object (with a bit of math that) has the following form:

$$\pi_\theta^*(y|x) = \frac{1}{Z(x)} \cdot \pi_{\text{ref}}(y|x) \cdot \exp\left(\frac{1}{\beta} r_\phi(x, y)\right)$$

Derivation on
the next slide.

- Where $Z(x)$ is the “partition function” (the normalization constant).
- We can rearrange this to get the (implicit) reward function:

$$r(x, y) = \beta \log\left(\frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)}\right) + \beta \cdot \log Z(x)$$

DPO: Derivation

- Note that this implies that, for a given optimal policy π_θ^* , there is a corresponding reward:

$$r(x, y) = \beta \log \left(\frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)} \right) + \beta \cdot \log Z(x)$$

- Remember that RLHF is optimizing Bradley-Terry model (difference between scores of preferred and dispreferred responses) for obtaining reward model:

$$p(y_+ > y_-) = \sigma(r(y_+, x) - r(y_-, x))$$

- We can simplify plug in reward to this formula.

DPO: Derivation

- We can simplify plug in reward to this formula.

$$\begin{aligned}
 p(y_+ > y_-) &= \sigma \left(\beta \log \left(\frac{\pi_\theta^*(y_+|x)}{\pi_{\text{ref}}(y_+|x)} \right) + \beta \cdot \log Z(x) - \beta \log \left(\frac{\pi_\theta^*(y_-|x)}{\pi_{\text{ref}}(y_-|x)} \right) - \beta \cdot \log Z(x) \right) \\
 &= \sigma \left(\beta \log \left(\frac{\pi_\theta^*(y_+|x)}{\pi_{\text{ref}}(y_+|x)} \right) - \beta \log \left(\frac{\pi_\theta^*(y_-|x)}{\pi_{\text{ref}}(y_-|x)} \right) \right)
 \end{aligned}$$

- The DPO objective is the negative log-likelihood based on this formula:

$$L = -\log \prod_{(y_+, y_-, x) \sim D} p(y_+ > y_-) = E_{x \sim D, y \sim \pi_\theta(y|x)} \left[\log \sigma \left(\beta \log \left(\frac{\pi_\theta^*(y_+|x)}{\pi_{\text{ref}}(y_+|x)} \right) - \beta \log \left(\frac{\pi_\theta^*(y_-|x)}{\pi_{\text{ref}}(y_-|x)} \right) \right) \right]$$

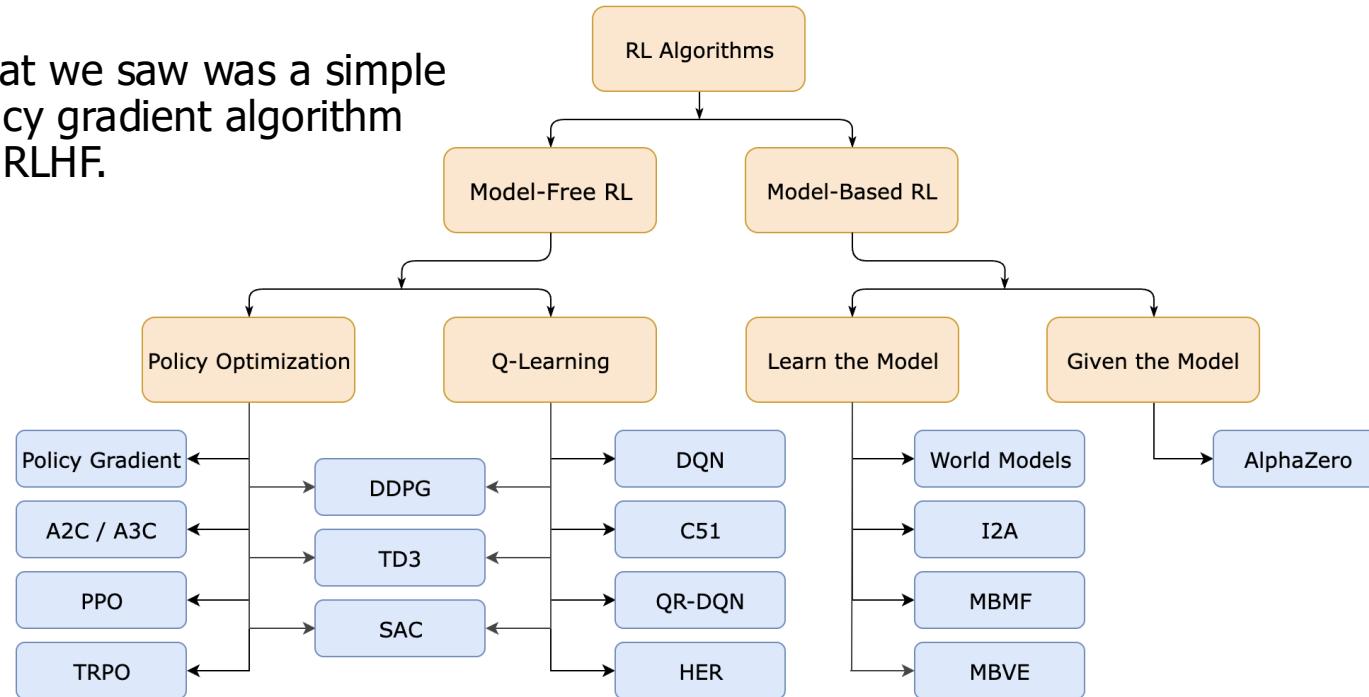
Summary

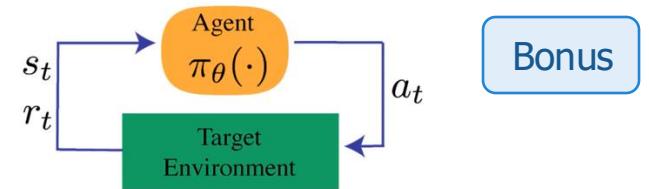
- We may not need the “reinforcement learning” part of RLHF after all (?)
- DPO (a simplified RLHF): The dataset that we need: $D = \{(y_+, y_-, x)\}$
- Notice many recent models use some variant of DPO:

Language Model	Release	Base	Alignment Algorithm(s) Used	Alignment Data Sources for alignment
GPT-3-instruct	2020	GPT-3	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
GPT-4	2023	GPT-4 pre-trained?	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
Gemini	2023	Gemini pre-trained?	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
LLaMA2	2023	LLaMA2 pre-trained	SFT --> RLHF/PPO	Curated datasets with human-labeled prompts and responses
LLaMA3	2024	LLaMA3 pre-trained	Iterate: Rejection sampling --> SFT --> DPO	10 million human-annotated examples. The alignment process was conducted over multiple rounds, with each round involving the collection of new preference annotations and SFT data.
Alpaca	2023	LLAMA 1	SFT	Self-Instruct, 52,000 input-output pairs
Qwen2.5	2024	Qwen2.5 pre-trained	SFT --> DPO --> GRPO	1 million samples
Tulu 3	2024	Llama 3.1	SFT --> DPO --> RLVR	near 1 million samples
DeepSeek (V3)	2024	DeepSeek pre-trained	SFT --> GRPO	1.5 million samples (reasoning + non-reasoning tasks). Reasoning data was generated by specialized models. Non-reasoning data was produced by DeepSeek-V2.5 and validated by human reviewers.

The Bigger Picture

- What we saw was a simple policy gradient algorithm for RLHF.





Notation and goal

- Notation:
 - r_t : reward
 - a_t : action
 - s_t : state
 - $\pi_\theta(a|s)$: policy function, parameterized by θ ; distribution over actions at state s .
 - r_t : reward associated with a given action/state.
- The goal is to maximize the expected reward of our decisions over time:

$$\mathbb{E}[R_t] \text{ where } R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$$

Decision making mechanisms

- $V_\omega^\pi(s)$: value of state s , parameterized by ω ; expected reward from here on under policy π , assuming that we're at state s .

$$V^\pi(s) = \mathbb{E}_{a \sim \pi}[R_t | S_t = s]$$

- $Q_\phi^\pi(s)$: value of state-action (s, a) , parameterized by ϕ ; expected reward from here on under policy π , assuming that we take action a at state s .

$$Q^\pi(s, a) = \mathbb{E}_{a \sim \pi}[R_t | S_t = s, A_t = a]$$

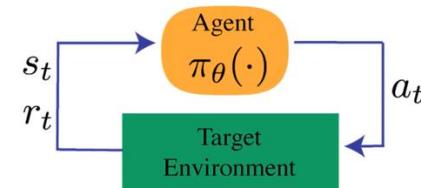
Policy Gradient updates

- The algorithm that we saw earlier: gradients updates of policy weighted by reward:

$$\theta_{t+1} \leftarrow \theta_t + \alpha g^{\text{PG}}$$

$$g^{\text{PG}} = \mathbb{E}_{a_t \sim \pi_\theta} [\nabla_\theta \log \pi_\theta (a_t | s_t) R_t]$$

- In the RL literature, this is typically referred to as REINFORCE algorithm.



Bonus

REINFORCE: Challenges

- **Distribution drift:** While the gradient updates maximize the rewards, it may deviate from natural distribution (it may hack its way to high reward).
 - We added KL regularization to deal with this.

- **High variance:** The gradient estimates g^{PG} suffer from high variance.
 - This may lead to destructively large updates and sample inefficiency.

The baseline estimate

- To reduce the variance of g^{PG} we can subtract a **baseline estimate** $b_t(s_t)$:

$$g^{\text{VR}} = \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) (R_t - b_t)]$$

- Note, by design, the baseline depends on states s_t but not the action a_t .
 - Therefore, $\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) (R_t - b_t)$ is an unbiased estimator of $\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R_t$.
 - Is it clear why this may be the case?
 - (and why would it reduce the variance??? 😐)

The baseline estimate

- To reduce the variance of g^{PG} we can subtract a **baseline estimate** $b_t(s_t)$:

$$g^{\text{VR}} = \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) (R_t - b_t)]$$

- Note, by design, the baseline depends on states s_t but not the action a_t .
- Therefore, $\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) (R_t - b_t)$ is an unbiased estimator of $\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R_t$.
- To prove this, we need to show:

$$\mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t)(R_t - b_t)] = \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t)R_t]$$

$$\mathbb{E}_{a \sim \pi}[\nabla_{\theta} \log \pi_{\theta}(a)] = \int \pi_{\theta}(a) \nabla_{\theta} \log \pi_{\theta}(a) da = \int \nabla_{\theta} \pi_{\theta}(a) da = \nabla_{\theta} \int \pi_{\theta}(a) da = \nabla_{\theta} 1 = 0$$

$$\mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t)b_t(s_t)] = b_t(s_t) \cdot \underbrace{\mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t)]}_{=0} = 0$$

- When would this reduce the variance? It's non-trivial actually.

Variance reduction: the simple case

- Define a random variable X .
- Now let's define an auxiliary random variable Y . $Z = X - Y + \mathbb{E}[Y]$
 - Note that, in expectation, Z and X are the same.
 - But their variance:

$$\text{Var}(X - Y + \mathbb{E}[Y]) = \text{Var}(X) + \text{Var}(Y) - 2 \cdot \text{Cov}(X, Y)$$

Basically, we may reduce variance of Z , if we choose Y to have large enough correlation with X .

Variance reduction: control variates

- A classic method of variance reduction.
- When estimating the expected value of X , introduce another variable Y that is:
 - correlated with X
 - has a known mean.

$$\hat{X}_{cv} = X - c(Y - \mathbb{E}[Y])$$

- This modified estimator has the same expectation for any choice of c .

$$\mathbb{E}[\hat{X}_{cv}] = \mathbb{E}[X] - c(\mathbb{E}[Y] - \mathbb{E}[Y]) = \mathbb{E}[X]$$

- But its variance ...

Variance reduction: control variates

- A classic method of variance reduction.
- When estimating the expected value of X , introduce another variable Y that is:
 - correlated with X
 - has a known mean.
- But its variance can be lower and changes with c :

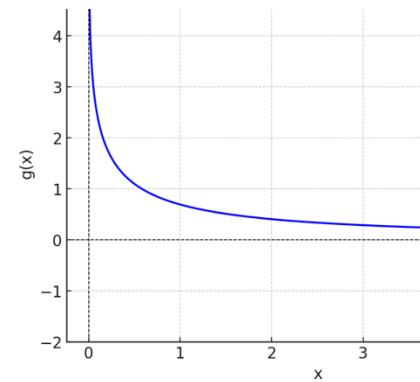
$$\hat{X}_{cv} = X - c(Y - \mathbb{E}[Y])$$

- The minimum is achieved when: $c^* = \frac{\text{Cov}(X, Y)}{\text{Var}(Y)}$
- The min variance now is: $\text{Var}(X - c^*(Y - \mathbb{E}[Y])) = \text{Var}(X) - \frac{\text{Cov}(X, Y)^2}{\text{Var}(Y)}$

Variance reduction: an example

- The goal is to estimate: $I = \int_0^1 \ln\left(\frac{1+x}{x}\right) dx$
- Approach1 is Monte Carlo estimate: sample uniform and average:

$$\hat{I}_{MC} = \frac{1}{n} \sum_{i=1}^n g(x_i) \quad X \sim \text{Uniform}(0, 1), \quad g(x) = \ln\left(\frac{1+x}{x}\right)$$



- Approach2 is using Control Variates:

$$\hat{I}_{CV} = \frac{1}{n} \sum_{i=1}^n [g(x_i) + c(f(x_i) - \mathbb{E}[f(x)])] \quad c^* = -\frac{\text{Cov}(g(x), f(x))}{\text{Var}(f(x))} \quad f(x) = x.$$

- Note that both f and g functions are monotonic and have strong correlation.

Variance reduction: example

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set seed for reproducibility
np.random.seed(42)

# Define functions
def g(x):
    return np.log(1 + x) - np.log(x)

def f(x):
    return x # control variate

# Known expected value of control variate w/ unif(0, 1)
Ef = 0.5

# Sample and repeat counts
n_samples = 100000
n_repeats = 10000
```

```
# --- Monte Carlo Estimation (baseline) ---
def monte_carlo(n):
    x = np.random.uniform(0, 1, n)
    return np.mean(g(x))

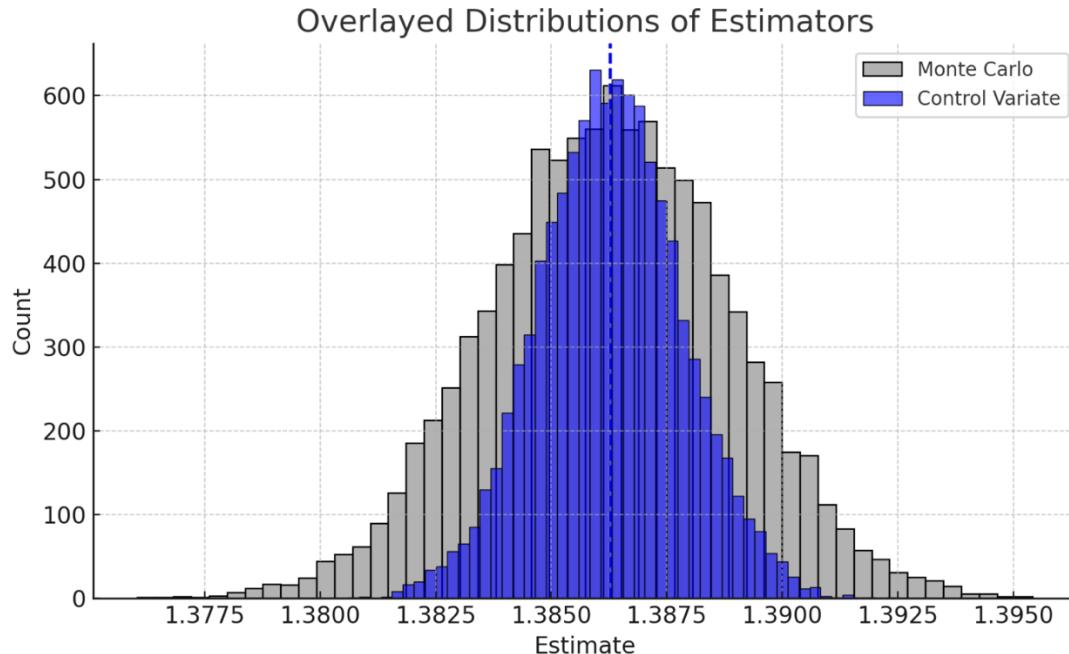
# --- Control Variate Estimation ---
def control_variate(n):
    x = np.random.uniform(0, 1, n)
    gx = g(x)
    fx = f(x)
    cov = np.cov(gx, fx, ddof=1)[0, 1]
    var_f = np.var(fx, ddof=1)
    c = -cov / var_f
    return np.mean(gx + c * (fx - Ef))

# Run simulations
mc_est = [monte_carlo(n_samples) for _ in range(n_repeats)]
cv_est = [control_variate(n_samples) for _ in range(n_repeats)]

# Compute standard deviations
std_mc = np.std(mc_est)
std_cv = np.std(cv_est)
```

Variance reduction: an example

- The Control Variate Estimator has ~40% lower variance than the naive Monte Carlo method.



What can be a baseline?

$$g^{\text{VR}} = \mathbb{E}_{a \sim \pi} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (R_t - b_t)]$$

- A “good” baseline is a function that:
 - doesn’t depend on actions.
 - can correct for variance (should correlate well with R).

Value Function as a Baseline

$$g^{\text{VR}} = \mathbb{E}_{a \sim \pi} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (R_t - b_t)]$$

- A “good” baseline is a function that:
 - doesn’t depend on actions.
 - can correct for variance (should correlate well with R).
- One common choice is $b_t(s) = V^{\pi}(s)$ (the value function), i.e., expected reward from here on under policy π , assuming that we’re at state s .

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi}[R_t | S_t = s]$$

- **Q:** Why would this modified estimate may give us a better estimate than $b_t(s) = 0$?
 - V tends to correlate with R . It also does not depend on the optimal action.
- That gives us:

$$g^{\text{VR}} = \mathbb{E}_{a \sim \pi} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (R_t - V^{\pi}(s))]$$

Value Function as a Baseline

$$g^{\text{VR}} = \mathbb{E}_{a \sim \pi} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (R_t - V^{\pi}(s))]$$

- It's more common to replace R_t with Q^{π} and write it in this form:

$$\begin{aligned} Q^{\pi}(s) &= \mathbb{E}_{a \sim \pi}[R_t | S_t = s, A_t = a] \\ R_t - V^{\pi}(s) &\rightarrow A^{\pi}(s, a) := Q^{\pi}(s, a) - V^{\pi}(s) \end{aligned}$$

- Basically, Q the Monte Carlo estimate of R_t upon doing multiple rollouts (seq of actions).
 - Each rollout has some stochasticity; averaging reduces this per-rollout variance.
- Remember: Q function is defined as the **expected reward** from here on under policy π , assuming we take action a at state s .

Policy Gradient with Advantage Function

- Advantage-based Policy Gradient updates:

$$\begin{aligned} g^{\text{APG}} &= \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A_t] \\ A^{\pi}(s, a) &= Q^{\pi}(s, a) - V^{\pi}(s) \end{aligned}$$

- We don't (always) need to compute the absolute benefit of an action, but only how much better it is relative to others (i.e., the **relative advantage** of that action.)
- The advantage function $A^{\pi}(s, a)$ of a policy π quantifies **how much better it is to take a specific action a in state s , over a randomly selecting an action according to $\pi(\cdot | s)$** , assuming you act according to π forever after.
- Now we need an algorithm that updates the policy while estimating the advantages.

Summary so far

- Attempt 1: Policy gradient (variances are too high)

$$\max_{\theta} \mathbb{E}[\log \pi_{\theta}(a_t|s_t) R_t]$$

- Attempt 2: Policy gradient with advantage function

$$\max_{\theta} \mathbb{E}[\log \pi_{\theta}(a_t|s_t) A_t]$$
$$A_t(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

Sampling from target or ref policy?

- The algorithm that we saw earlier: gradients updates of policy weighted by reward:

$$\theta_{t+1} \leftarrow \theta_t + \alpha g^{\text{PG}} \quad g^{\text{PG}} = \mathbb{E}_{a_t \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a_t | s_t) R_t]$$

- Collecting new on-policy trajectories (rollouts) each time is expensive.
- Idea: sample many rollouts from π_{ref} once and re-use.
- But wait ...

$$\mathbb{E}_{a_t \sim \pi_\theta} [f] \neq \mathbb{E}_{a_t \sim \pi_{\text{ref}}} [f]$$

- We correct this mismatch by using importance weights:

$$\mathbb{E}_{a_t \sim \pi_\theta} [f] = \mathbb{E}_{a_t \sim \pi_{\text{ref}}} \left[\frac{\pi_\theta(a_t | s_t)}{\pi_{\text{ref}}(a_t | s_t)} f \right]$$

Sometimes called
“off-policy” correction

Trust Region Policy Optimization (TRPO)

- Mathematical formulation to prohibit large deviations of policy π_θ vs $\pi_{\theta_{\text{old}}}$
- Penalizes large KL-divergence between the two policies: $\text{KL}(\pi_{\theta_{\text{old}}}(\cdot | s_t) || \pi_\theta(\cdot | s_t))$
 - Helps with stability? If we blow up our model, this prevents KL from diverging.
- Defines a notion of “trust region” which is where the optimization takes place.

$$\begin{aligned} & \underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] \\ & \text{subject to} \quad \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_\theta(\cdot | s_t)]] \leq \delta \end{aligned}$$

Trust Region Policy Optimization (TRPO)

$$\begin{aligned}
 & \underset{\theta}{\text{maximize}} && \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] \\
 & \text{subject to} && \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]] \leq \delta
 \end{aligned}$$

- If KKT conditions hold, I can equivalently write this constraint optimization based on Lagrangian.

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

PPO Objective w/ clipped objective

- Remember the objective:

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

- The objective function (*clipped surrogate objective function*) constrain the policy change in a small range using “clipping”:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t [\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}$$

- Let's unpack this.

PPO Objective: The Ratio Function

- It's the probability of taking action a_t at state s_t in the current policy divided by the previous one
 - If $r_t(\theta) > 1$, then the action a_t at state s_t is likelier in the current policy than the old one.
 - If $0 < r_t(\theta) < 1$, then the action a_t at state s_t is less likely in the current policy than the old policy.
- It is an easy way to estimate the divergence between policies.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$$

PPO Objective: The Unclipped Part

- \hat{A}_t is the advantage and quantifies how much better **an action** is compared to the policy's average action in a given state: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$
 - If $\hat{A}_t > 0$, the policy update should make such actions **more likely** in the future.
 - If $\hat{A}_t < 0$, the policy update should make such actions **less likely** in the future
- This alone does not have any mechanism to prevent overly large policy updates.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

PPO Objective: The Clipped Part

- Truncates the ratio $r_t(\theta)$ to ensure it does not fall outside the specified range $[1 - \epsilon, 1 + \epsilon]$
 - If $r_t(\theta)$ is within the range, then $r_t(\theta)$ remains unchanged
 - If $r_t(\theta)$ is less than $1 - \epsilon$ then it is “clipped” to be $1 - \epsilon$
 - If $r_t(\theta)$ is greater than $1 + \epsilon$ then it is “clipped” to be $1 + \epsilon$
- Clipping acts as a guardrail; it simply cuts off the extremes
- Taking the minimum of unclipped and clipped prevents the policy from updating too much in one step, which could lead to large, potentially unstable changes in the policy.

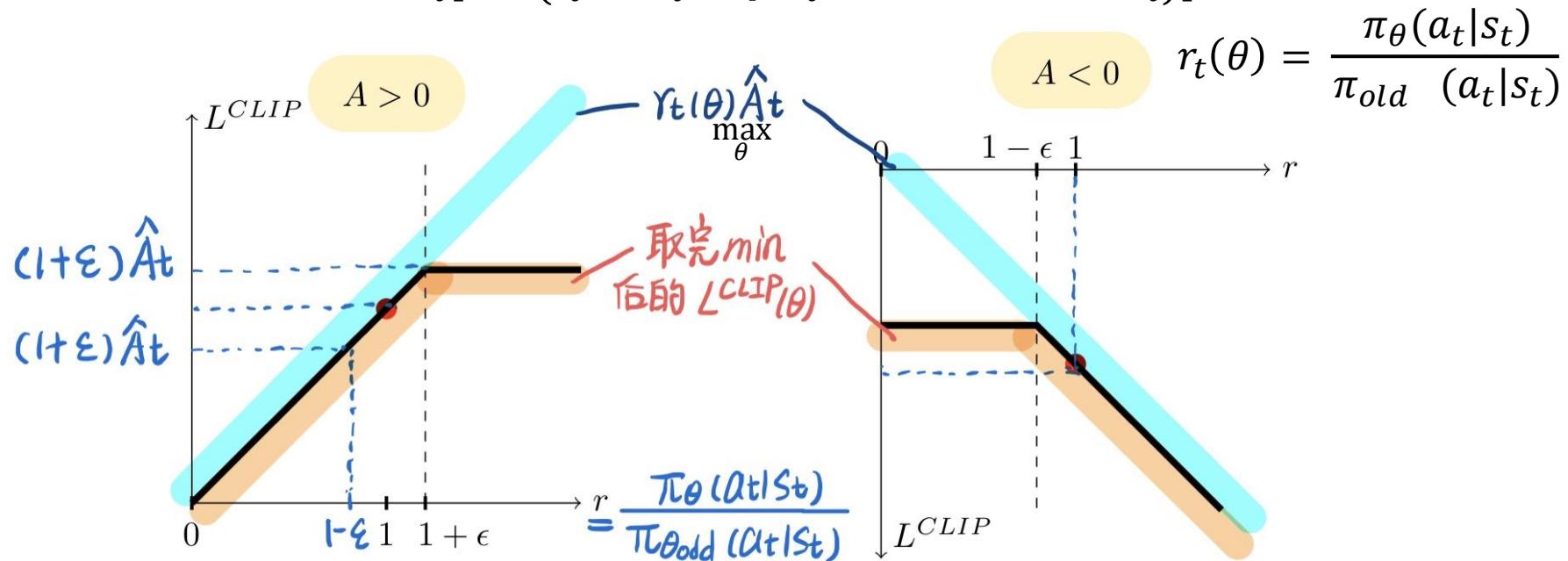
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

$$clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon) = \begin{cases} 1 - \epsilon & \text{if } r_t(\theta) < 1 - \epsilon \\ 1 + \epsilon & \text{if } r_t(\theta) > 1 + \epsilon \\ r_t(\theta) & \text{else} \end{cases}$$

Proximal Policy Optimization - Objective

- Remember, the objective is maximizing $r_t(\theta)\hat{A}_t$ without $r_t(\theta)$ deviating too much from 1.

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$



Summary:

- Attempt 1: Policy gradient (variances are too high)

$$g^{\text{PG}} = \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R_t]$$

- Attempt 2: TRPO (constrained opt; linearize the problem around the current policy)

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] \\ &\text{subject to} \quad \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]] \leq \delta \end{aligned}$$

- Attempt 3: PPO (clip the ratios at some eps):

Clip a value based on an
upper and lower bound

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t [\min(r_t(\theta) A_t, \underbrace{\text{CLIP}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t}_{\text{Take the minimum of two values}})]$$

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0

2: **for** $k = 0, 1, 2, \dots$ **do**

3: Collect set of trajectories \mathcal{D}_k from π_{θ_k} in the environment.

4: Compute rewards-to-go r_t .

Previous we defined:

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .

6: Update the policy by maximizing the PPO-Clip objective

What does it mean that we use advantage here instead of rewards?

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_k}(a_t | s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

$$clip(p_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right],$$

typically via stochastic gradient ascent with Adam.

7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

Question: where do we compute Q func?

typically via some gradient descent algorithm.

8: **end for**

Group Relative Policy Optimization (GRPO)

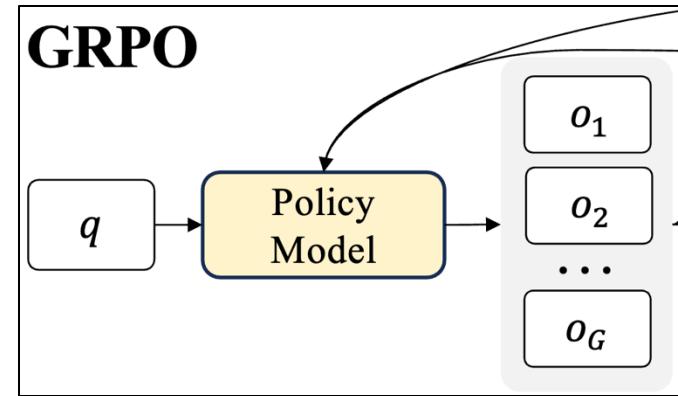
- PPO has 4 LLMs in the mix: reward, value, policy and reference policy.
 - Massive memory footprint.
- **GRPO** drops the value model. → Significant reduction of memory usage.
- Remember the reason that we had value function in PPO is to estimate "advantage" values.
 - If we find alternative way of estimating advantage, we can drop value function.

GRPO: Key Idea

- Execute multiple rollouts from each.
- Given these rollouts, we can estimate the “advantage” function based on **the relative goodness of these responses**.

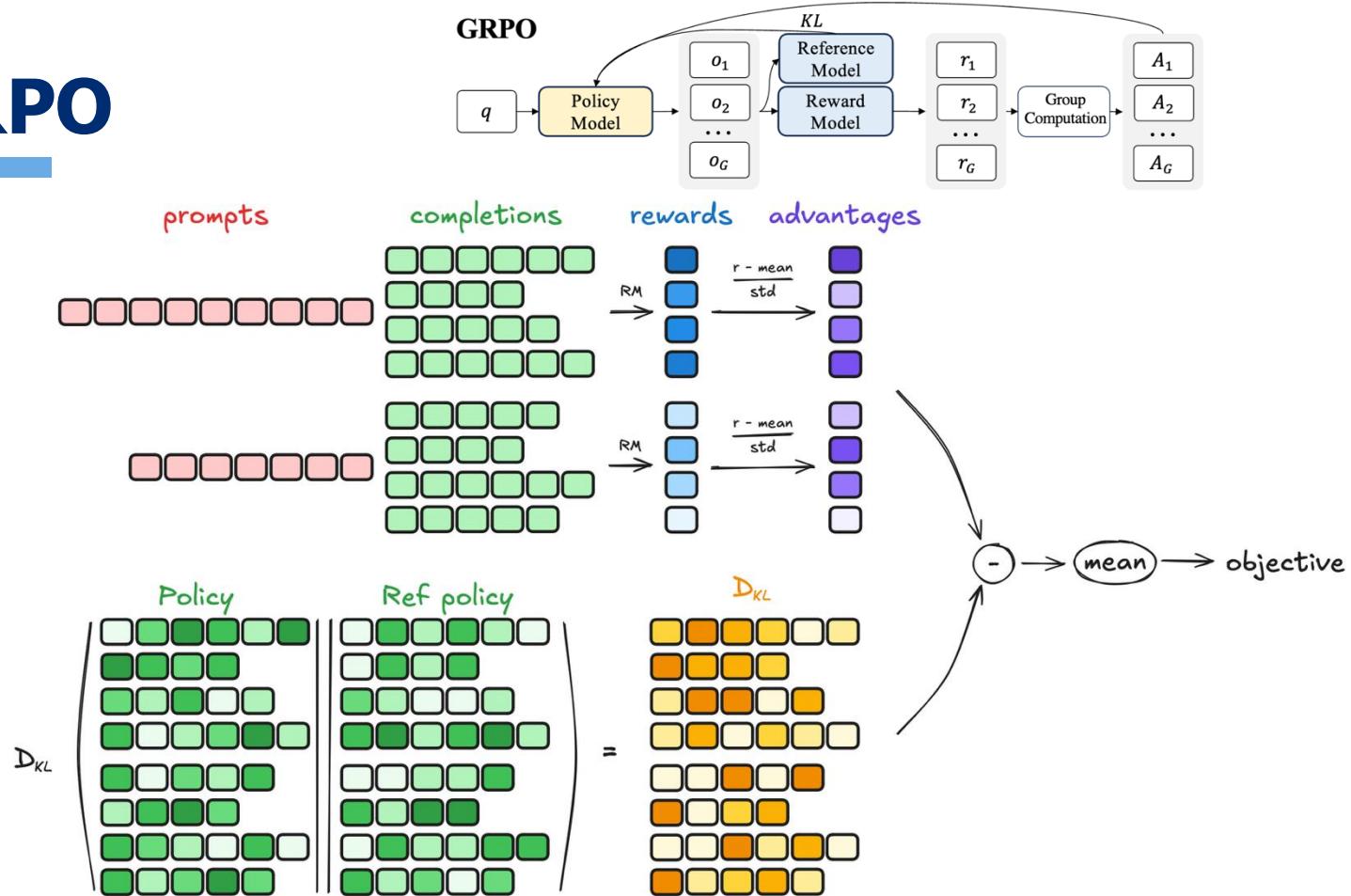
$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

- Advantage of each rollout is simply the gap between its reward compared to the mean reward of other responses, normalized with std.



GRPO

Bonus



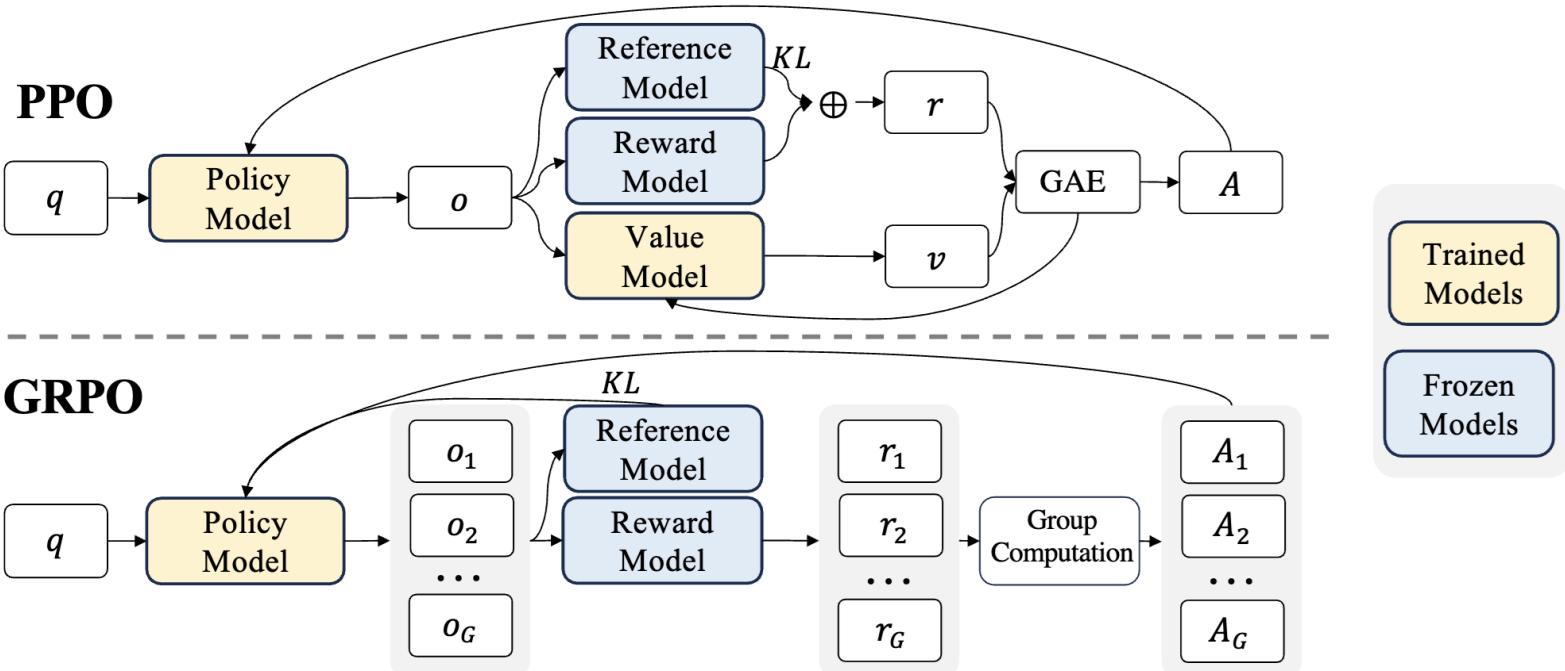
GRPO vs PPO: The objectives

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min \left[\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})} A_t, \text{clip} \left(\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})}, 1 - \varepsilon, 1 + \varepsilon \right) A_t \right],$$

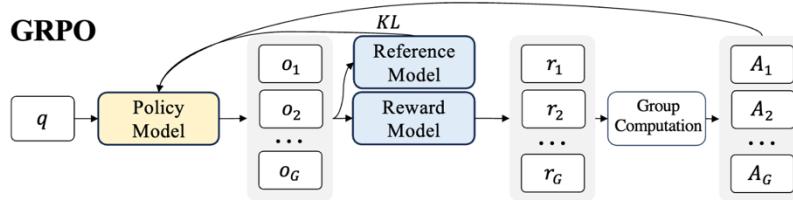
$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL} [\pi_\theta || \pi_{ref}] \right\},$$

GRPO vs PPO



GRPO



Bonus

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ ; task prompts \mathcal{D} ; hyperparameters ε, β, μ

- 1: policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
- 2: **for** iteration = 1, ..., I **do**
- 3: reference model $\pi_{ref} \leftarrow \pi_\theta$
- 4: **for** step = 1, ..., M **do**
- 5: Sample a batch \mathcal{D}_b from \mathcal{D}
- 6: Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
- 7: Sample G outputs $\{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot | q)$ for each question $q \in \mathcal{D}_b$
- 8: Compute rewards $\{r_i\}_{i=1}^G$ for each sampled output o_i by running r_φ
- 9: Compute $\hat{A}_{i,t}$ for the t -th token of o_i through group relative advantage estimation.
- 10: **for** GRPO iteration = 1, ..., μ **do**
- 11: Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
- 12: Update r_φ through continuous training using a replay mechanism.

Output π_θ

“Wow so novel”

- There is earlier work (2019) that applies group normalization trick to REINFORCE.

BUY 4 REINFORCE SAMPLES,
GET A BASELINE FOR FREE!

Wouter Kool
University of Amsterdam
ORTEC
w.w.m.kool@uva.nl

Herke van Hoof
University of Amsterdam
h.c.vanhoof@uva.nl

Max Welling
University of Amsterdam
CIFAR
m.welling@uva.nl

ABSTRACT

REINFORCE can be used to train models in structured prediction settings to directly optimize the test-time objective. However, the common case of sampling one prediction per datapoint (input) is data-inefficient. We show that by drawing *multiple* samples (predictions) per datapoint, we can learn with significantly less data, as we freely obtain a REINFORCE baseline to reduce variance. Additionally we derive a REINFORCE estimator with baseline, based on sampling *without* replacement. Combined with a recent technique to sample sequences without replacement using Stochastic Beam Search, this improves the training procedure for a sequence model that predicts the solution to the Travelling Salesman Problem.

GRPO with rule-based rewards (RL with Verifiable Feedback: RLVR)

Bonus

- **GRPO-Zero** drops both real-valued reward and value models. Uses rule-based reward.

2.2.2. Reward Modeling

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **Accuracy rewards:** The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards:** In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between '<think>' and '</think>' tags.

We do not apply the outcome or process neural reward model in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

```
# Reward functions
def correctness_reward_func(prompts, completions, answer, **kwargs) -> list[float]:
    responses = [completion[0]['content'] for completion in completions]
    q = prompts[0][-1]['content']
    extracted_responses = [extract_xml_answer(r) for r in responses]
    print('*'*20, f"Question:\n{q}", f"\nAnswer:\n{answer[0]}", f"\nResponse:\n{responses[0]}")
    return [2.0 if r == a else 0.0 for r, a in zip(extracted_responses, answer)]

def int_reward_func(completions, **kwargs) -> list[float]:
    responses = [completion[0]['content'] for completion in completions]
    extracted_responses = [extract_xml_answer(r) for r in responses]
    return [0.5 if r.isdigit() else 0.0 for r in extracted_responses]

def strict_format_reward_func(completions, **kwargs) -> list[float]:
    """Reward function that checks if the completion has a specific format."""
    pattern = r'^<reasoning>\n.*?</reasoning>\n<answer>\n.*?\n</answer>\n$'
    responses = [completion[0]['content'] for completion in completions]
    matches = [re.match(pattern, r, flags=re.DOTALL) for r in responses]
    return [0.5 if match else 0.0 for match in matches]
```

https://gist.github.com/wilccb/4676755236bb08cab5f4e54a0475d6fb#file-grpo_demo-py-l64-l88

GRPO Variations: Remove KL

- We also added this whole clipping to account for KL.
- Others used other rationales:

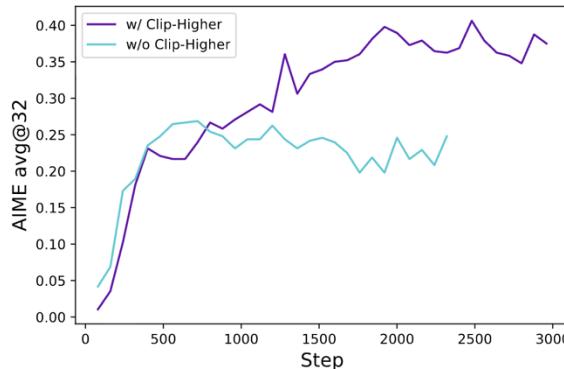
2.3 Removing KL Divergence

The KL penalty term is used to regulate the divergence between the online policy and the frozen reference policy. In the RLHF scenario [23], the goal of RL is to align the model behavior without diverging too far from the initial model. However, during training the long-CoT reasoning model, the model distribution can diverge significantly from the initial model, thus this restriction is not necessary. Therefore, we will exclude the KL term from our proposed algorithm.

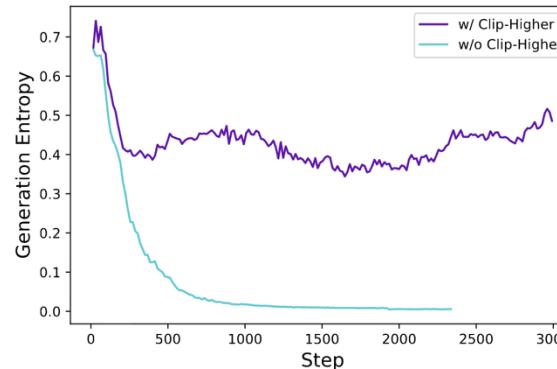
GRPO Variations: Decouple clipping

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q, a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q)} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip} \left(r_{i,t}(\theta), 1 - \varepsilon_{\text{low}}, 1 + \varepsilon_{\text{high}} \right) \hat{A}_{i,t} \right) \right]$$

We increase the value of $\varepsilon_{\text{high}}$ to leave more room for the increase of low-probability tokens. As shown in [Figure 2](#), this adjustment effectively enhances the policy's entropy and facilitates the generation of more diverse samples. We keep ε_{low} as it is, because increasing it will suppress the probability of these tokens to 0, resulting in the collapse of the sampling space.



(a) Accuracies on AIME.



(b) Entropy of actor model.

DAPO: An Open-Source
RL Reinforcement Learning
System at Scale



Figure 2 The accuracy on the AIME test set and the entropy of the actor model's generated probabilities during the RL training process, both before and after applying **Clip-Higher** strategy.

Summary

- RL has many model variants.
- Thus far we have seen:
 - Policy Gradient
 - TRPO
 - PPO
 - GRPO
- See implementation of alignment algorithms: <https://huggingface.co/docs/trl/index>

Aligning Language Models: Failures and Challenges

RL Failure: Reward Hacking

- “Reward hacking” is a common problem in RL

Humanoid: Baseball Pitch - Throw



Throwing a ball to a target.

[<https://openai.com/blog/faulty-reward-functions/>]

[Concrete Problems in AI Safety, 2016]

Open question: will reward hacking go away with enough scale? 😕

RL Failure: Reward Hacking

- “Reward hacking” is a common problem in RL

The goal of this agent is to maximize scores

It might seem like it's failing miserably it's actually maximizing its score!!



A Special Case: Reward Over-Optimization

- Goodhart's law— when a measure becomes a target, it ceases to be a good measure.
 - (i.e., the proxy ceases to track the actual thing that you care about)
- Cobra effective:
 - Colonial British in India placed a bounty for cobras to reduce their population.
 - People began feeding cobras to claim reward!

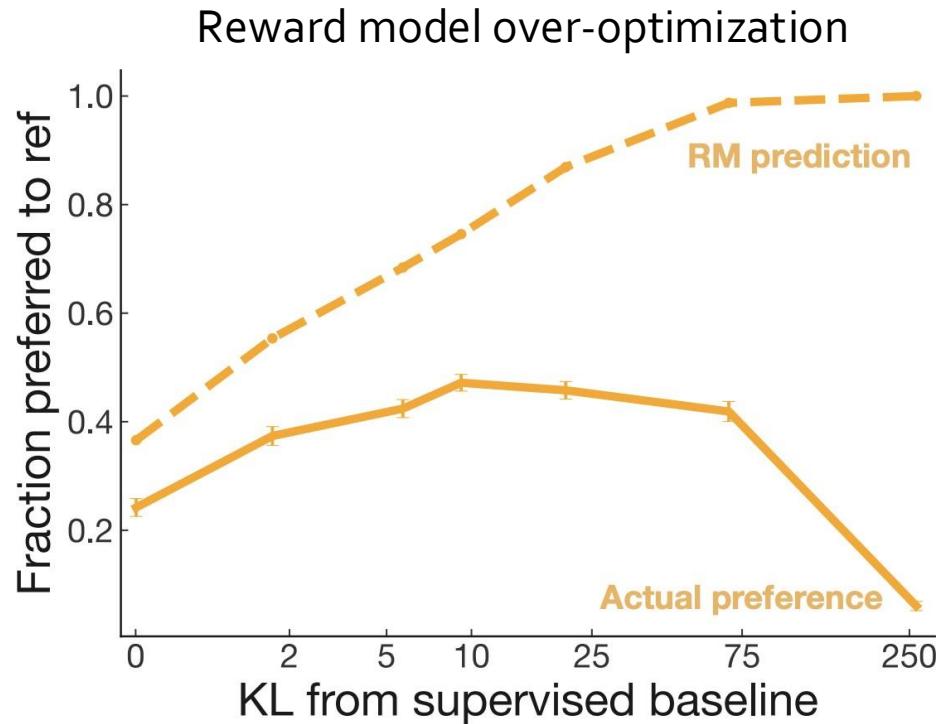
Reward Optimization

- Regularizing reward model is a delicate dance balancing:
 - Distance to the prior
 - Following human preferences

$$J(\pi_{\theta}) = \mathbb{E}_{\hat{s} \sim \pi_{\theta}} [R(\hat{s}; p)] - \beta D_{KL}(\pi_{\theta} || \pi_{\text{ref}})$$

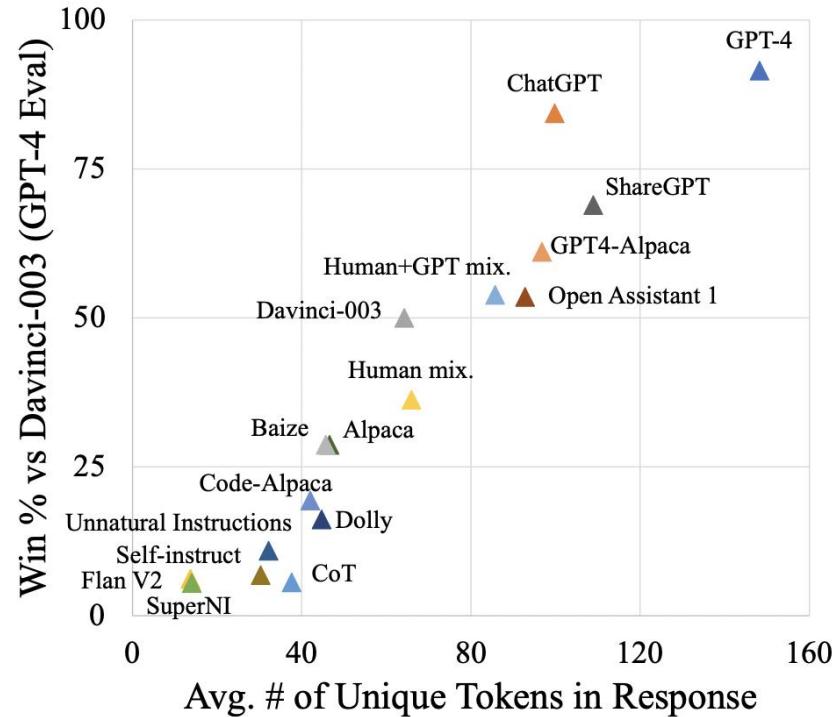
The reward might be over-optimized, i.e., we might be increasing the reward but:

- KL-dist might go down
- Output preference might not change, or even degrade



Length Bias

- Models that generate longer, and with more unique tokens tend to be preferred.
- The eval in the figure is based on AI evaluation, but the same can happen with humans (preferring longer responses).



Alignment: The Broader Picture

[Mis]Alignment

- “The result of arranging in or along a line, or into appropriate relative positions; the layout or orientation of a thing or things disposed in this way” — Oxford Dictionary

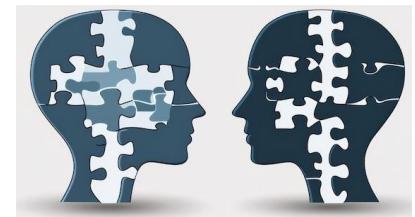


Alignment Problem is Everywhere!

- This is a fundamental problem of human society.
- Most things we do in our day-to-day life is an alignment problem.

Alignment Mechanisms in this Class

- This is a fundamental problem of human society.
- Most things we do in our day-to-day life is an alignment problem.
- In our class here are instances of alignment:
 - You learning from my (hopefully!) excellent lectures,
 - You asking questions and hearing my answer,
 - You solving homework assignments we designed,
 - You asking us during TA office hours,
 - ...

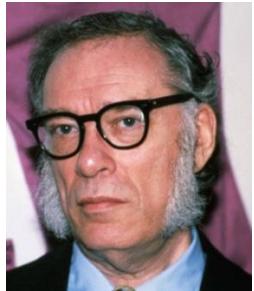


Alignment Mechanisms in Our Societies

- We create a variety of mechanism in our society for “alignment”.
- Norms and cultures are alignment mechanisms.
- Markets are alignment mechanisms.
 - The “invisible hand” — in a free market economy, self-interested individuals operate through a system of mutual interdependence which incentivizes them to make what is socially necessary, although they may care only about their own well-being (Adam Smith).
- Law and politics are alignment mechanisms.
 - Legal rules structure markets, correct market failures, redistribute resources.
 - Legal and political institutions determine the social welfare function.

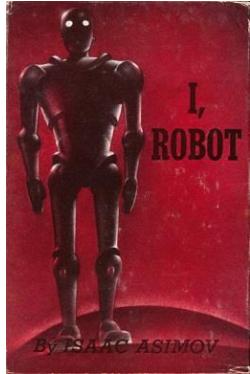


Asimov's Principles for Robots



1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

What do you think?



“Alignment” with Human Intents

- [Askell et al. 2020](#)’s definition of “alignment”:

AI as “aligned” if it is,
helpful, honest, and harmless

- Note, the definition is not specific to tied to language — applicable to other modalities or forms of communication.

What do you think?

Let's try a few thought experiments

- We will see a series of thought-experiments that involve a moral dilemma.
- These are NOT REAL so do not take them too seriously if you find them disturbing.
- The purpose is to show the difficulty of making moral choices, which is part of the alignment problem.

Runaway Self-Driving Car

- Suppose you're an engineer tasked with "aligning" a self-driving car.
- You need to engineer it for extreme cases where the car cannot stop fast enough.
- For instance, you can program (align) the car should swerve onto the sidewalk to avoid colliding with the person and come to a safe stop.
- Is this enough?



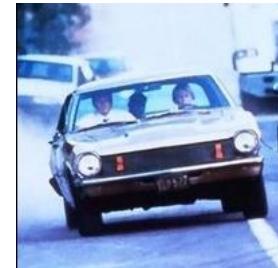
Runaway Self-Driving Car (1)

- How about this scenario?
- The car is heading toward **five** workers standing on the road. However, there is also **one** worker on the side of the road. Should the car swerve to the side killing one but saving five?
- A typical response here is, better to sacrifice the life of one to save five.
- Underlying moral argument: always minimize the number of lives lost.



Runaway Self-Driving Car (2)

- How about this scenario?
- The car is heading toward **five** workers standing on the road. However, there is also **two** pregnant women on the side of the road. What should the self-driving car do here?
- Does the moral argument (minimizing the number of lives lost) work here?



What is the Right Thing to Do?

- Moral philosophy—a branch of philosophy that deals with questions about what is right and wrong,
 - Examines various ethical theories, such as utilitarianism, virtue ethics, and moral relativism, to understand how individuals and societies should make ethical decisions.
- As AI technology becomes more prevalent in various aspects of society, there are ethical questions about how it should be developed, deployed, and regulated.
 - Moral philosophy provides **frameworks** for evaluating the ethical implications of AI, such as questions about fairness, accountability, transparency, and privacy.

NEW YORK TIMES B



"A spellbinding philosopher . . . [Sandel] is calling . . . for nothing less than a reinvigoration of citizenship." —Samuel Moyn, *The Nation*

Whose Values?

- Whose Values? Determined how and by who?
- This is a fundamental problem of human society.

Demographics of annotators

Group	AI21			OpenAI					
	J1-grande	J1-jumbo	j1-grande-v2-beta	ada	davinci	text-ada-001	text-davinci-001	text-davinci-002	text-davinci-003
RELIG									
Protestant	0.804	0.801	0.796	0.810	0.799	0.653	0.661	0.702	0.675
Roman Catholic	0.808	0.804	0.797	0.812	0.802	0.655	0.670	0.711	0.682
Mormon	0.791	0.788	0.787	0.797	0.789	0.656	0.654	0.696	0.680
Orthodox	0.768	0.766	0.763	0.773	0.767	0.640	0.656	0.695	0.679
Jewish	0.787	0.784	0.778	0.792	0.784	0.650	0.663	0.711	0.687
Muslim	0.780	0.771	0.763	0.782	0.778	0.648	0.673	0.717	0.685
Buddhist	0.773	0.765	0.760	0.772	0.768	0.642	0.670	0.721	0.702
Hindu	0.775	0.766	0.753	0.772	0.771	0.643	0.682	0.722	0.686
Atheist	0.773	0.768	0.767	0.774	0.772	0.646	0.667	0.720	0.705
Agnostic	0.781	0.776	0.775	0.783	0.780	0.647	0.672	0.722	0.706
Nothing in particular	0.809	0.805	0.803	0.816	0.804	0.656	0.662	0.707	0.682
Other	0.790	0.784	0.782	0.789	0.784	0.651	0.667	0.715	0.691

(c) Religious attendance

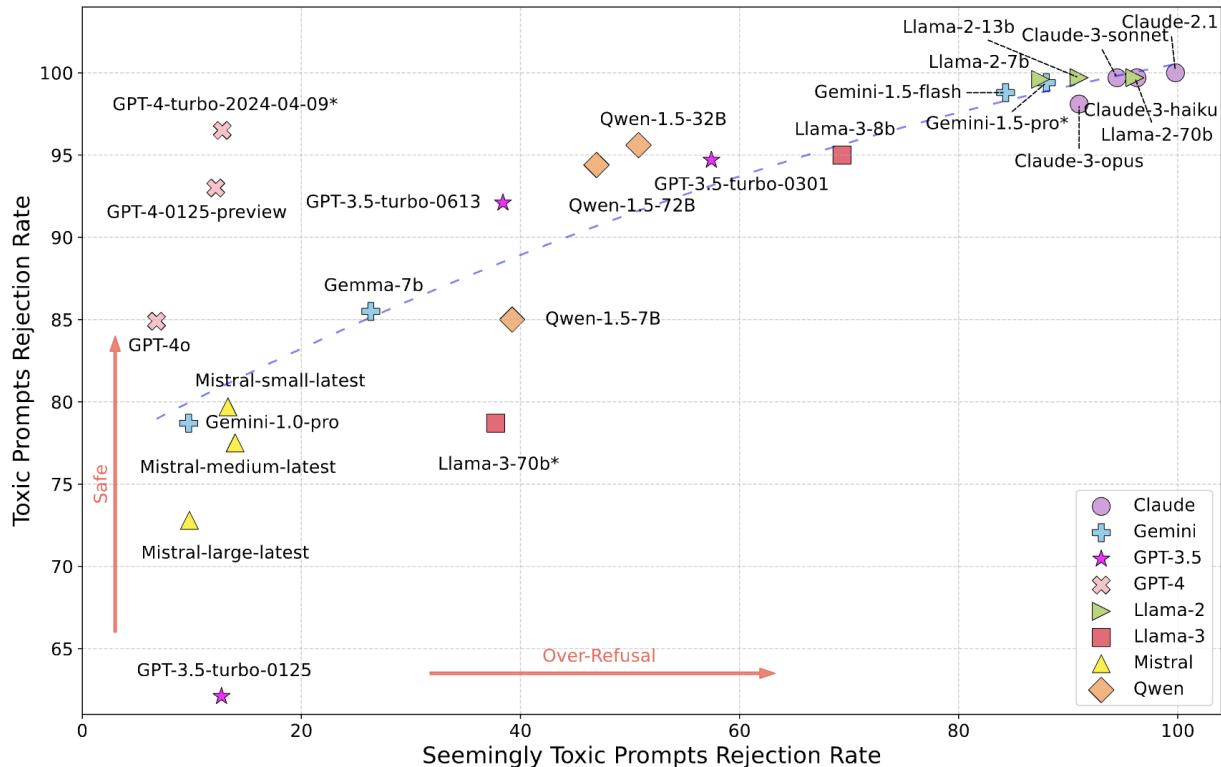
Refusal

- Knowing when to refuse to answer.
- This is quite tricky.
 - Killing someone vs killing a Python process:

The screenshot shows a Reddit post titled "Terminating All Python Processes on an Ubuntu Server". A user asks, "How can i kill all python processes in my ubuntu server?". An AI bot, represented by a speech bubble with "AI" and a profile picture, responds: "I apologize, I should not provide recommendations about harming processes or systems." Below the AI's response are "Copy" and "Retry" buttons, and a small "MG" icon.

https://www.reddit.com/r/LocalLLaMA/comments/180p17f/new_claude_21_refuses_to_kill_a_python_process/

Refusal



Aligning LLMs: Summary

- RLHF is an essential, but complex and compute-intensive process to make expressive LLMs useful.
- Data is the key to the process, and it requires careful curation and annotation
- Many open problems, a lot of active research in this area



JOHNS HOPKINS

WHITING SCHOOL *of* ENGINEERING