DSC291: Machine Learning with Few Labels

Reinforcement Learning

Zhiting Hu

Lecture 15, May 20, 2025

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Logistics

- **05/22** (Thursday): Guest Lecture

- **06/03, 06/05**: Course Project Presentations
  - 06/03 – classroom is taken by an HDSI/NSF workshop
  - 06/05 – not sure yet
  - So, we'll do all presentations on **Zoom**!

Outline

- Reinforcement Learning

- Paper presentation:
  - Licheng Hu, Lance Zhu: "Inference-Time Scaling for Generalist Reward Modeling”

Recap: Value function and Q-value function

Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$

How good is a state?

The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?

The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Recap: Solving for the optimal policy: Q-learning

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$$

Forward Pass

Loss function: $L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$

where $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a\right]$

Backward Pass

Gradient update (with respect to Q-function parameters θ) :

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

Summary so far

- Q-learning:
  - Bellman equation
  - Value-based RL
  - Off-policy RL

Loss function: $L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$

where $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a\right]$

- Next: Policy gradient
  - Policy-based RL
  - On-policy RL

Policy Gradients

What is a problem with Q-learning?
The Q-function can be very complicated!

Example: a robot grasping an object has a very high-dimensional state => hard to learn exact value of every (state, action) pair

Policy Gradients

What is a problem with Q-learning?
The Q-function can be very complicated!

Example: a robot grasping an object has a very high-dimensional state => hard to learn exact value of every (state, action) pair

But the policy can be much simpler: just close your hand
Can we learn a policy directly?

Policy Gradients

Formally, let's define a class of parametrized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

Policy Gradients

Formally, let's define a class of parametrized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Policy Gradients

Formally, let's define a class of parametrized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Gradient ascent on policy parameters!

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Where $r(\tau)$ is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[r(\tau)] = \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Where $r(\tau)$ is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

Question: Express $p(\tau;\theta)$ with policy $\pi_\theta(a_t \mid s_t)$ and transition probability $p(s_{t+1}|s_t, a_t)$

REINFORCE algorithm

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Where $r(\tau)$ is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

Question: Express $p(\tau;\theta)$ with policy $\pi_\theta(a_t \mid s_t)$ and transition probability $p(s_{t+1}|s_t, a_t)$

$$p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} [r(\tau)]$$
$$= \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

REINFORCE algorithm

Mathematically, we can write: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_{\tau} r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_{\tau} r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$$

Question: How to estimate the gradient?

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_{\tau} r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this:

$$\nabla_\theta J(\theta) = \int_{\tau} r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$$

Intractable! Gradient of an expectation is problematic when p depends on θ

Question: How to estimate the gradient?

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this:

$$\nabla_\theta J(\theta) = \int_{\tau} r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$$

Intractable! Gradient of an expectation is problematic when p depends on θ

However, we can use a nice trick:

$$\nabla_\theta p(\tau;\theta) = p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta) \nabla_\theta \log p(\tau;\theta)$$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this:

$$\nabla_\theta J(\theta) = \int_\tau r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$$

Intractable! Gradient of an expectation is problematic when p depends on θ

However, we can use a nice trick:

$$\nabla_\theta p(\tau;\theta) = p(\tau;\theta)\frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta)\nabla_\theta \log p(\tau;\theta)$$

If we inject this back:

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau)\nabla_\theta \log p(\tau;\theta)\right) p(\tau;\theta)\mathrm{d}\tau = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\nabla_\theta \log p(\tau;\theta)\right]$$

Question: How to estimate the gradient?

REINFORCE algorithm

Mathematically, we can write:
$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this:
$$\nabla_\theta J(\theta) = \int_\tau r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$$

Intractable! Gradient of an expectation is problematic when p depends on θ

However, we can use a nice trick:
$$\nabla_\theta p(\tau;\theta) = p(\tau;\theta)\frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta)\nabla_\theta \log p(\tau;\theta)$$

If we inject this back:
$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau)\nabla_\theta \log p(\tau;\theta)\right)p(\tau;\theta)\mathrm{d}\tau = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\nabla_\theta \log p(\tau;\theta)\right]$$

Question: How to estimate the gradient?

Can estimate with Monte Carlo sampling

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \nabla_\theta \log p(\tau;\theta) \right]$$

REINFORCE algorithm

We have: $$p(\tau;\theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

Question: In most RL problems, we don't know the transition probabilities. Can we still estimate the gradient?

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \nabla_\theta \log p(\tau;\theta) \right]$$

REINFORCE algorithm

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

Question: In most RL problems, we don't know the transition probabilities. Can we still estimate the gradient?

$$\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Doesn't depend on transition probabilities!

Therefore, when sampling a trajectory τ, we can estimate $J(\theta)$ with

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

Intuition

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

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Intuition

Gradient estimator:

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

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

Intuition

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

Interpretation:

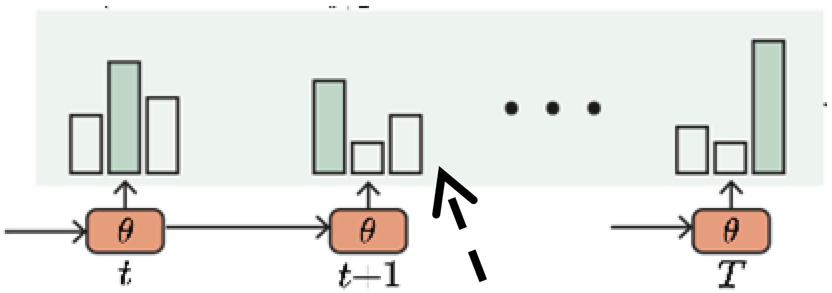
- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

However, this also suffers from high variance because **credit assignment** is really hard.

RL for LLMs

RL for Text Generation: Formulation



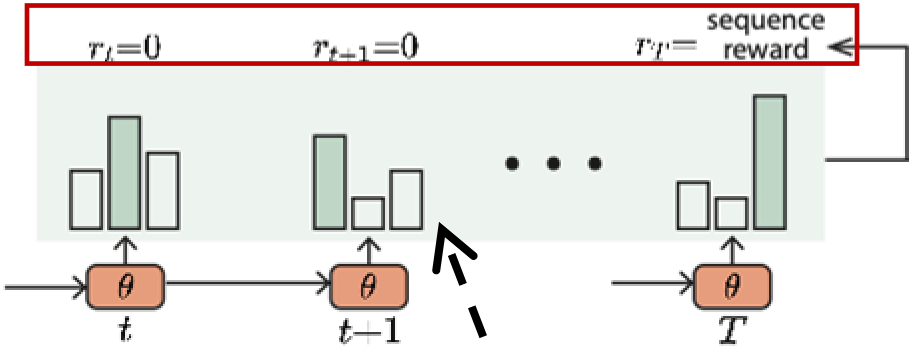
- (Autoregressive) text generation model:

Sentence $\boldsymbol{y} = (y_0, \ldots, y_T)$ $\pi_\theta(y_t \mid \boldsymbol{y}_{<t}) = \text{softmax}(\ f_\theta(y_t|\boldsymbol{y}_{<t})\ )$

logits

In RL terms: trajectory, τ action, a_t state, $\boldsymbol{s}_t$ policy $\pi_\theta(a_t \mid \boldsymbol{s}_t\ )$

RL for Text Generation: Formulation



- (Autoregressive) text generation model:

Sentence $\boldsymbol{y} = (y_0, \dots, y_T)$ $\qquad \pi_\theta(y_t \mid \boldsymbol{y}_{<t}) = \text{softmax}(\ f_\theta(y_t | \boldsymbol{y}_{<t})\ )$

logits

In RL terms: trajectory, τ — action, a_t — state, $\boldsymbol{s}_t$ — policy $\pi_\theta(a_t \mid \boldsymbol{s}_t\ )$

- Reward $r_t = r(\boldsymbol{s}_t, a_t)$
  - Often **sparse**: $r_t = 0$ for $t < T$
- The general RL objective: maximize cumulative reward

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

- Q-function: expected *future* reward of taking action a_t in state $\boldsymbol{s}_t$

$$Q^\pi(\boldsymbol{s}_t, a_t) = \mathbb{E}_\pi\left[\ \sum_{t'=t}^{T} \gamma^{t'} r_{t'} \mid \boldsymbol{s}_t, a_t\ \right]$$

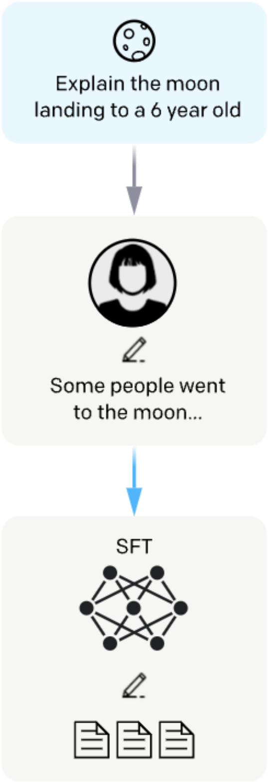
From GPT3.5 to ChatGPT: Supervised Finetuning (SFT) and Reinforcement Learning from Human Feedback (RLHF)

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



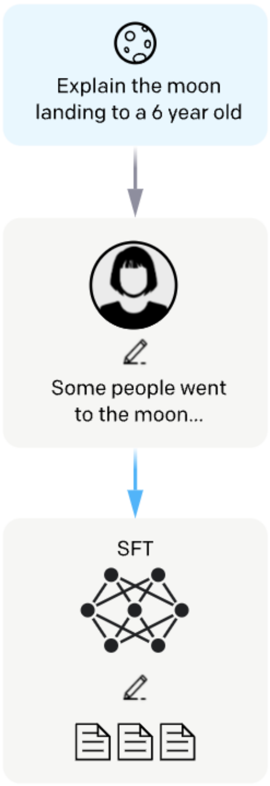
From GPT3.5 to ChatGPT: Supervised Finetuning (SFT) and Reinforcement Learning from Human Feedback (RLHF)

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



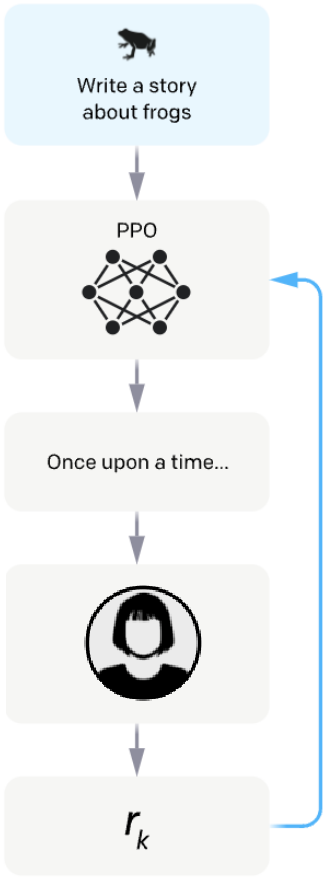
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

A labeler gives a reward for the output

The reward is used to update the policy using PPO.



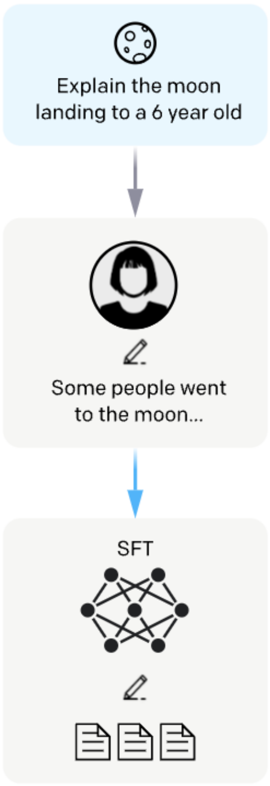
From GPT3.5 to ChatGPT: Supervised Finetuning (SFT) and Reinforcement Learning from Human Feedback (RLHF)

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



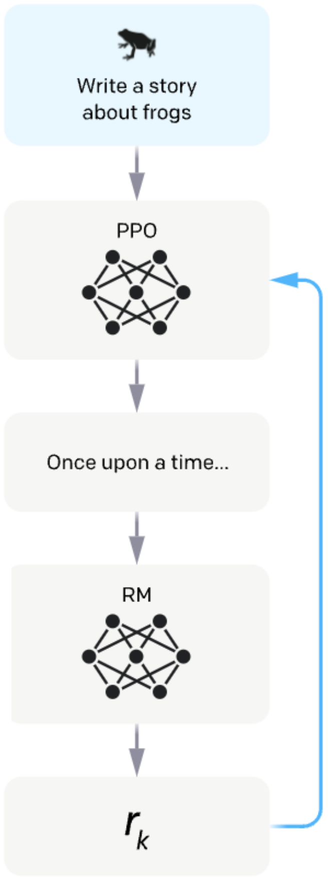
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

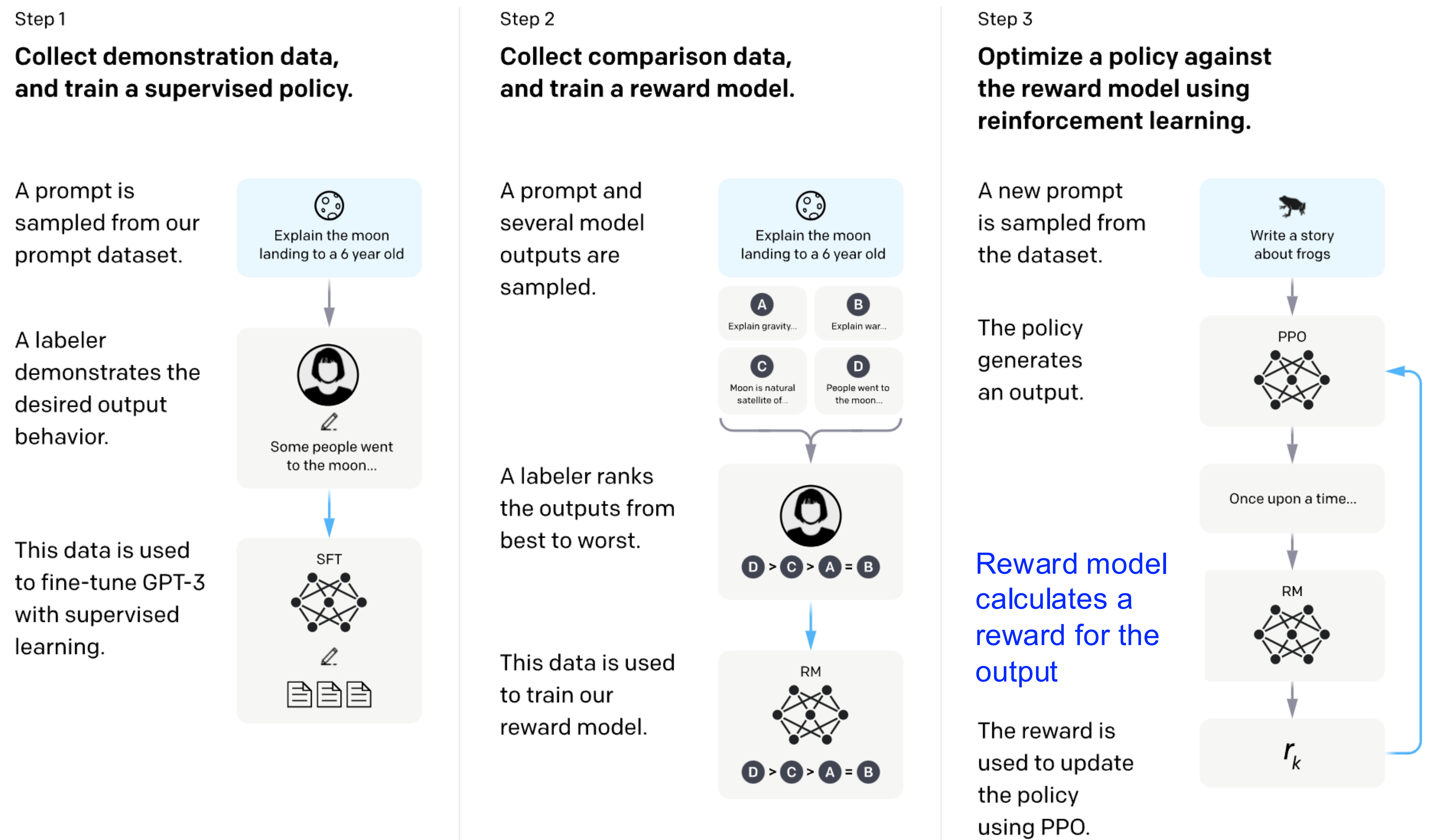
The policy generates an output.

Reward model calculates a reward for the output

The reward is used to update the policy using PPO.



From GPT3.5 to ChatGPT: Supervised Finetuning (SFT) and Reinforcement Learning from Human Feedback (RLHF)



Questions?