RLHF

What, Why and How?

Arvind Nagaraj
Dec 2024

Context - SFT step in LLM training

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

Completion GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

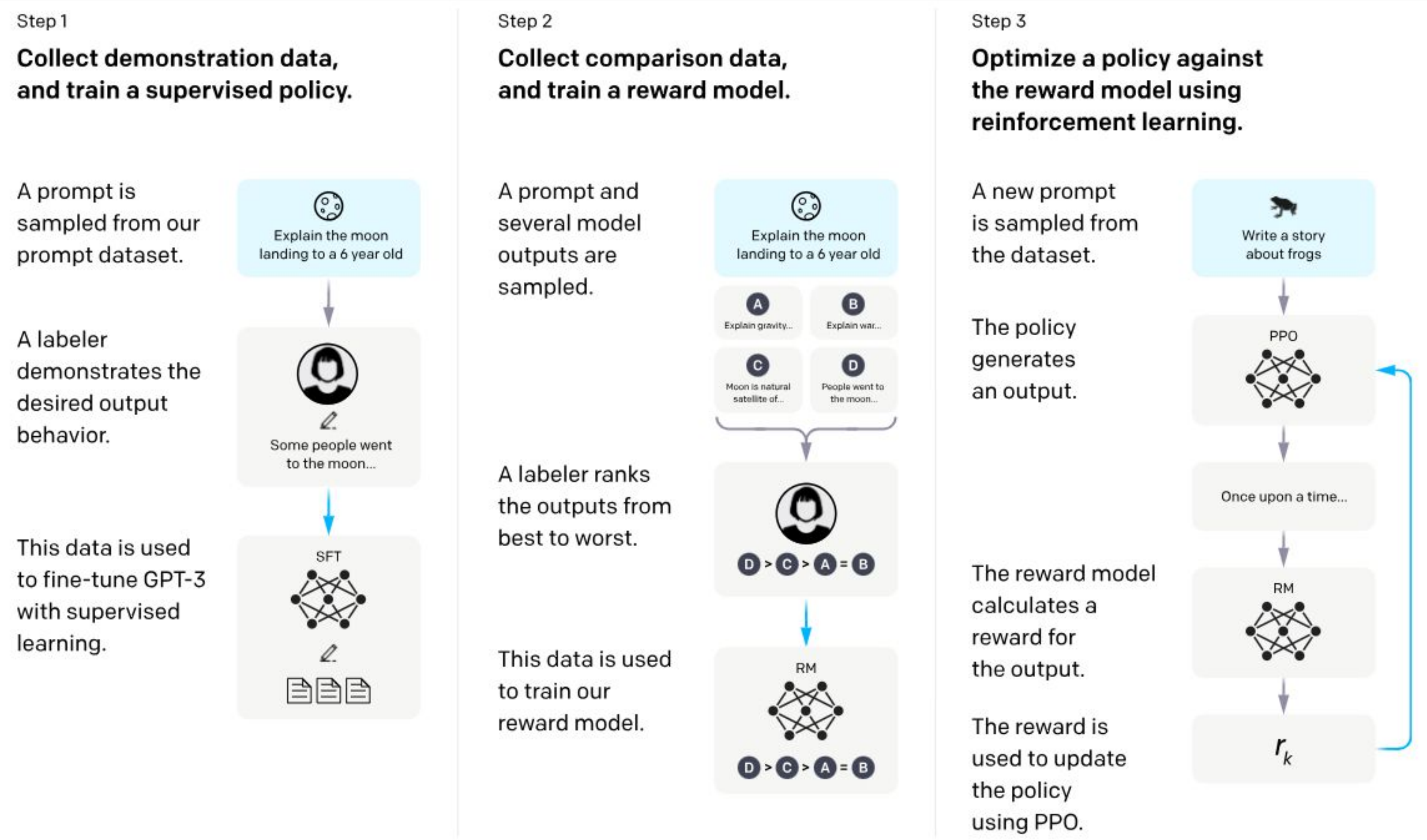
InstructGPT

```
People went to the moon, and they took pictures of what they saw, and
sent them back to the earth so we could all see them.
```

GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM can deploy this model	1-100 GPUs days of training ex: Vicuna-13B can deploy this model	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude can deploy this model

Source: Andrej Karpathy's talk at Microsoft Build 2023


Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
RM
D > C > A = B
Step 3
Optimize a policy against the reward model using reinforcement learning.
A new prompt is sampled from the dataset.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

src

PPO

Proximal

Policy

Optimization

PPO (RL)

RL

HF

REINFORCEMENT LEARNING

Supervised learning = Perception

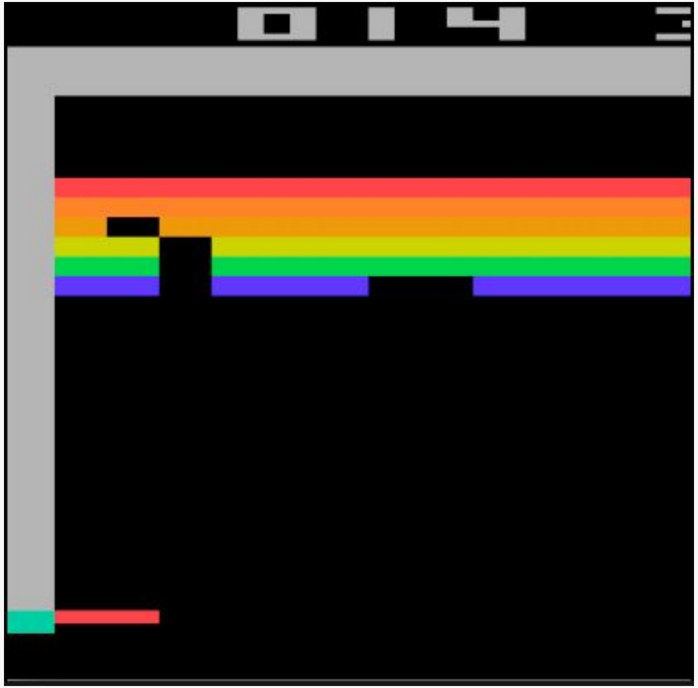
Reinforcement learning = Judgement

Action and Choices

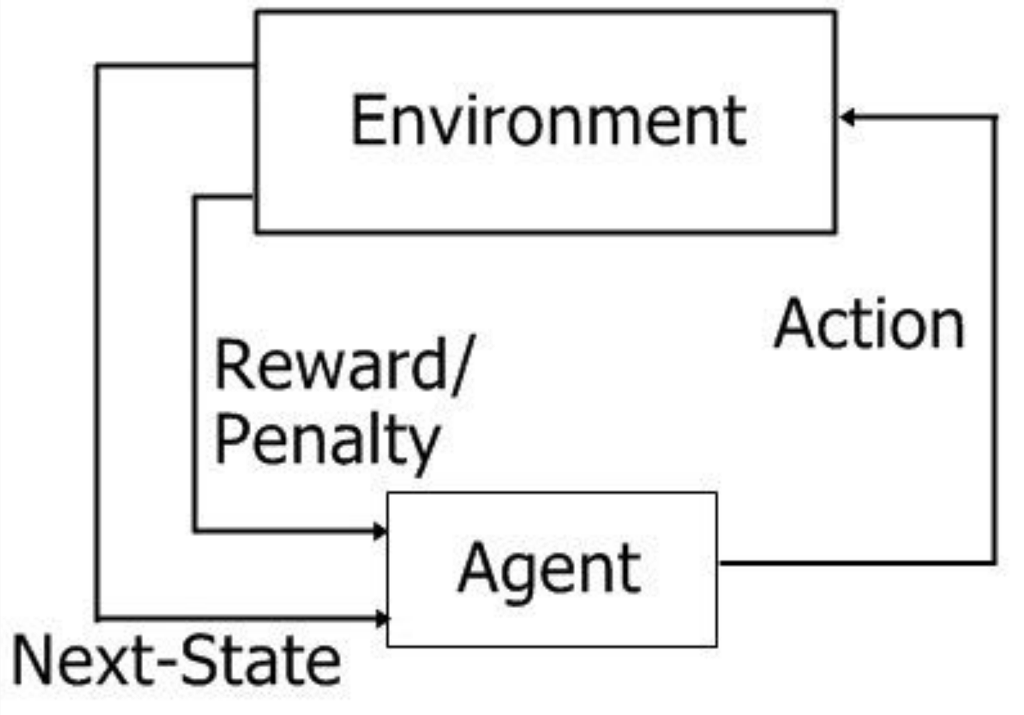
Robotics and Games

Real time decisions with Mitra

Atari games



Rewards and Penalties



Driving from A to B

from HSR Layout, Bengaluru, Karnataka
to Kempegowda International Airport Bengaluru

1 hr 29 min (53.9 km)

via Outer Ring Rd/Srinagar - Kanyakumari Hwy and NH 44

Best route now, avoids road closure

⚠ This route has tolls.

HSR Layout

Bengaluru, Karnataka

Take 17th Cross Rd, 14th Main Rd and Service Rd to 100 Feet Ring Rd/Outer Ring Rd/Srinagar - Kanyakumari Hwy in Agara Village

4 min (1.5 km)

↑ Head west on 17th Cross Rd toward 17th Main Rd

ⓘ Pass by HDFC Bank (on the right)

700 m

↱ Turn right at 14th Main Rd

130 m

↑ Continue onto 14th Main Rd

ⓘ Pass by Mantri Ceramics (on the right)

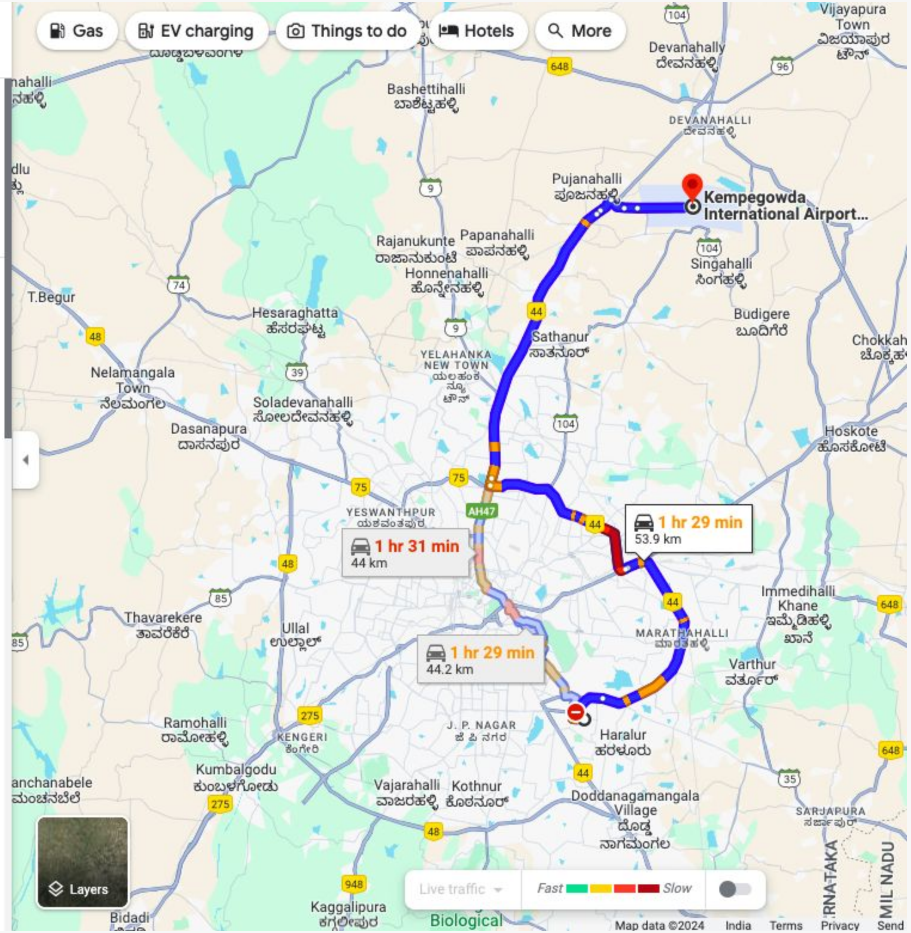
240 m

↑ Continue straight onto 14th Main Rd/Jakkasandra Cross Rd

66 m

↱ Turn right onto Service Rd

350 m



Sparse rewards

Rewards are sparse but when you get them, it's usually insightful.

RL is still early

- Explore or exploit?
- Sparse rewards
- Credit assignment?

John Schulman's lecture at Berkeley

Youtube link

Prof.Yoav Goldberg's arguments on : Why RL for LLMs ?

yoavg/rl-for-llms.md

PPO "algorithm"

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

PPO's Clipped surrogate objective function

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

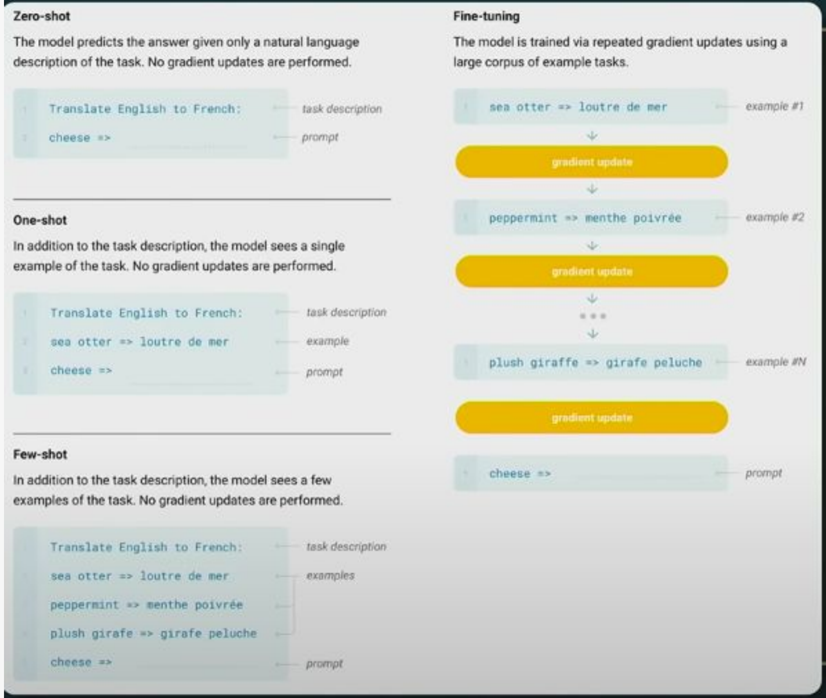
An extra precaution - KL Divergence

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

KL Divergence

$$KL(P||Q) = \sum p_i(x) log(\frac{p_i(x)}{q_i(x)})$$

Finetuning



It is becoming a lot more accessible to finetune LLMs:

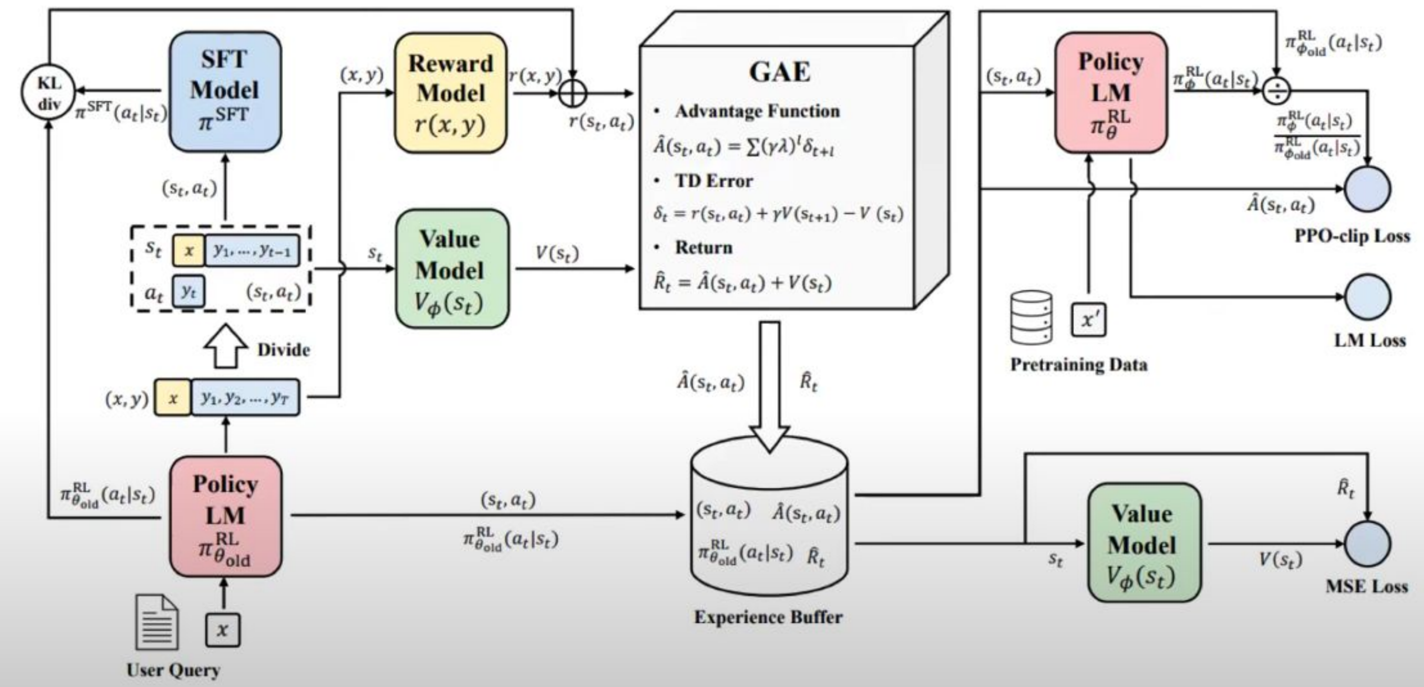
- Parameter Efficient FineTuning (PEFT), e.g. LoRA
- Low-precision inference, e.g. bitsandbytes
- Open-sourced high quality base models, e.g. LLaMA

Keep in mind:

- Requires a lot more technical expertise
- Requires contractors and/or synthetic data pipelines
- A lot slower iteration cycle
- SFT is achievable
- RLHF is research territory

[Language Models are Few-Shot Learners, Brown et al. 2020]

RL is Hard!



[Secrets of RLHF in Large Language Models Part I: PPO, Zheng, et al. 2023]

HF TRL library

Step 1: `SFTTrainer`
Train your model on your favorite dataset

```
from trl import SFTTrainer

trainer = SFTTrainer(
    "facebook/opt-350m",
    train_dataset=dataset,
    dataset_text_field="text",
    max_seq_length=512,
)

trainer.train()
```

Step 2: `RewardTrainer`
Train a preference model on a comparison data to rank generations from the supervised fine-tuned (SFT) model

```
from trl import RewardTrainer

trainer = RewardTrainer(
    model=model,
    args=training_args,
    tokenizer=tokenizer,
    train_dataset=dataset,
)

trainer.train()
```

Step 3: `PPOTrainer`
Further optimize the SFT model using the rewards from the reward model and PPO algorithm

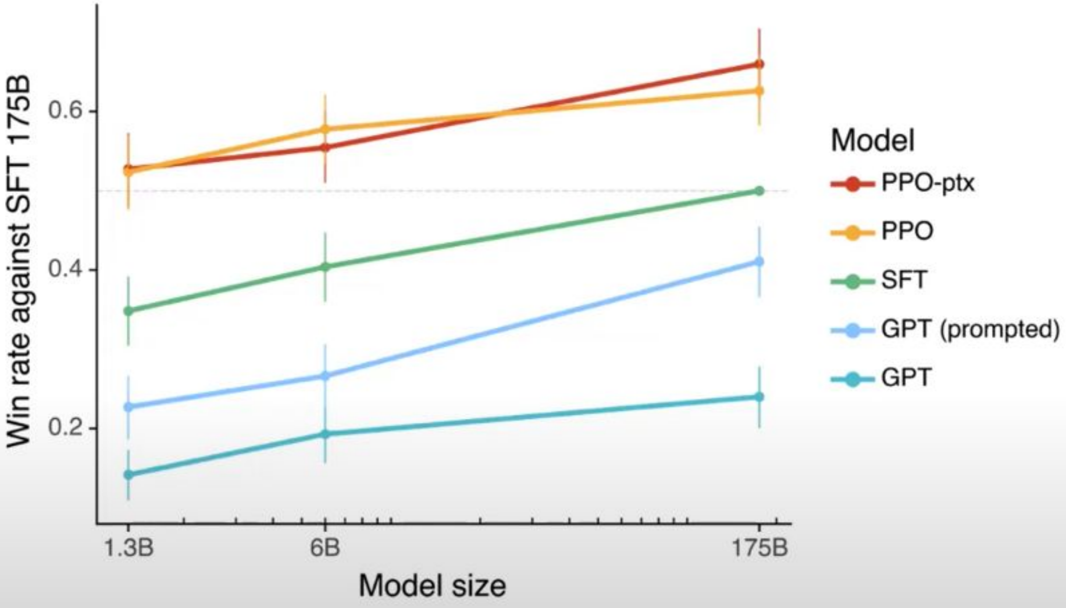
```
from trl import PPOConfig, PPOTrainer

trainer = PPOTrainer(
    config,
    model,
    tokenizer=tokenizer,
)

for query in dataloader:
  response = model.generate(query)
  reward = reward_model(response)
  trainer.step(query, response, reward)
```

We still want RL's benefits though!

- Eval: **win rate** against the 175B parameter SFT model (% of responses humans like better)



[Training language models to follow instructions with human feedback, Ouyang, et al., 2022]

DIRECT PREFERENCE OPTIMIZATION

What do we need?

Bradley-Terry Model

$$\text{reward\_diff} = \log(\text{sigmoid}((\text{reward}_{winner}) - (\text{reward}_{loser})))$$

LOSS = maximize (reward_diff)

LOSS = minimize (reward_diff X -1)

Bradley-Terry Model

reward_diff = log (sigmoid (($\text{reward}_{\text{winner}}$) - ($\text{reward}_{\text{loser}}$)))

LOSS = maximize (reward_diff)

LOSS = minimize (reward_diff X -1)

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

Direct Preference Optimization: Simplifying RLHF

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed Form solution

Basically, An ideal mathematical solution (*may not be computable!*)

$$\text{Optimal Policy} = \frac{\text{Reference Policy X assigned reward}}{\text{Normalization Value}}$$

- Normalization treats all sequence lengths with equal fairness

Direct Preference Optimization: Simplifying RLHF

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(y|x)} [r(x, y)] - \beta \mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x) \| \pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

Slides by : Eric Mitchell's(Stanford University) lecture at Berkeley

Z - the problem!

SOFTMAX FUNCTION

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

with $Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

Direct Preference Optimization: Simplifying RLHF

any reward function

RLHF Objective

(get **high reward**, stay **close** to reference model)

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

$$\pi^*(y \mid x) = \frac{1}{Z(x)}\pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$

Note **intractable sum** over possible responses; can't immediately use this

Rearrange

(write **any reward function** as function of **optimal policy**)

Ratio is **positive** if policy likes response more than reference model, **negative** if policy likes response less than ref. model

$$r(x,y) = \underbrace{\beta\log\frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta\log Z(x)}_{\text{Some parameterization of a reward function}}$$

Direct Preference Optimization: Putting it together

A loss function on
<u>reward functions</u>

+

A transformation
between <u>reward</u>
<u>functions</u> and <u>policies</u>

=

A loss function
on <u>policies</u>

Direct Preference Optimization: Putting it together

A loss function on <u>reward functions</u>

Derived from the Bradley-Terry model of human preferences

$$\mathcal{L}_R(r, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r(x,y_w) - r(x,y_l))\right]$$

+

A transformation between <u>reward</u> <u>functions</u> and <u>policies</u>

Reward function for which that policy is optimal

Any policy (w/ mild assumptions)!

$$r_{\pi_\theta}(x,y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

When substituting, the **log Z term cancels**, because the loss only uses **difference** in rewards

=

Reward of **preferred** response

Reward of **dispreferred** response

A loss function on <u>policies</u>

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

Since π_θ is normalized, **we've lost a degree of freedom, but not expressiveness** (see paper)

Gradient mechanics of DPO

What does the gradient of the loss look like? That is,

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = ?$$

$$-\beta \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \Big[\sigma\big(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w)\big) \big[\nabla_\theta \log \pi(y_w \mid x) - \nabla_\theta \log \pi(y_l \mid x) \big] \Big]$$

Per-example weight: **Higher weight when the reward model is wrong**

Increase the likelihood of the **preferred completions**

Decrease the likelihood of the **dispreferred completions**

$$\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

TRL implementation

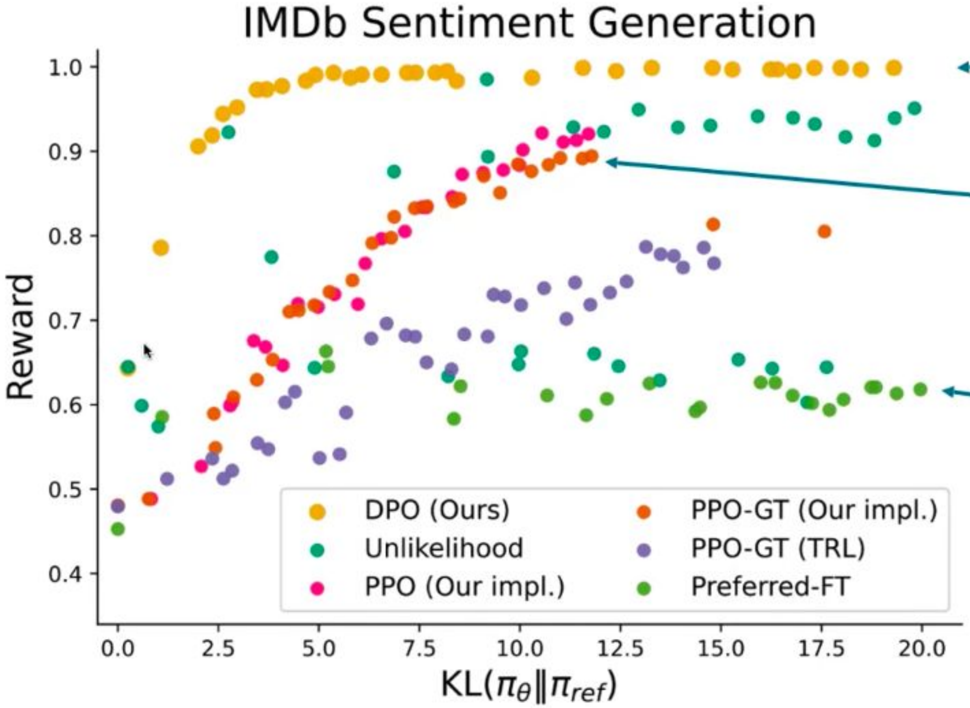
Blame 1246 lines (1068 loc) · 60.3 KB

```python
def dpo_loss(
    pi_logratios = pi_logratios.to(self.accelerator.device)
    ref_logratios = ref_logratios.to(self.accelerator.device)
    logits = pi_logratios - ref_logratios

    # The beta is a temperature parameter for the DPO loss, typically something in the range of 0.1 to 0.5.
    # We ignore the reference model as beta -> 0. The label_smoothing parameter encodes our uncertainty abo
    # calculates a conservative DPO loss.
    if self.loss_type == "sigmoid":
        losses = (
            -F.logsigmoid(self.beta * logits) * (1 - self.label_smoothing)
            - F.logsigmoid(-self.beta * logits) * self.label_smoothing
        )
    elif self.loss_type == "hinge":
        losses = torch.relu(1 - self.beta * logits)
    elif self.loss_type == "ipo":
        # eqn (17) of the paper where beta is the regularization parameter for the IPO loss, denoted by tau
        losses = (logits - 1 / (2 * self.beta)) ** 2
    elif self.loss_type == "kto_pair":
        # eqn (7) of the HALOs paper
        chosen_KL = (policy_chosen_logps - reference_chosen_logps).mean().clamp(min=0)
        rejected_KL = (policy_rejected_logps - reference_rejected_logps).mean().clamp(min=0)

        chosen_logratios = policy_chosen_logps - reference_chosen_logps
        rejected_logratios = policy_rejected_logps - reference_rejected_logps
        # As described in the KTO report, the KL term for chosen (rejected) is estimated using the rejected
        losses = torch.cat(
            (
                1 - F.sigmoid(self.beta * (chosen_logratios - rejected_KL)),
```

Reward-KL trade-off



DPO provides the strongest reward-KL frontier

PPO doesn't provide optimal reward, even when provided with ground truth

DPO provides a substantial improvement over Preferred-FT

[Direct Preference Optimization, Rafailov, Sharma, Mitchell, Ermon, Manning, & Finn., NeurIPS 2023]

Direct Preference Optimization: Simplifying RLHF

High-level punchline

If we parameterize our reward model correctly…

…we can extract the optimal policy for our learned reward model **in closed form, with no additional training**

The trick: use a direct correspondence between optimal policy and reward model!

$$\pi(y|x) \Leftrightarrow r(x, y)$$