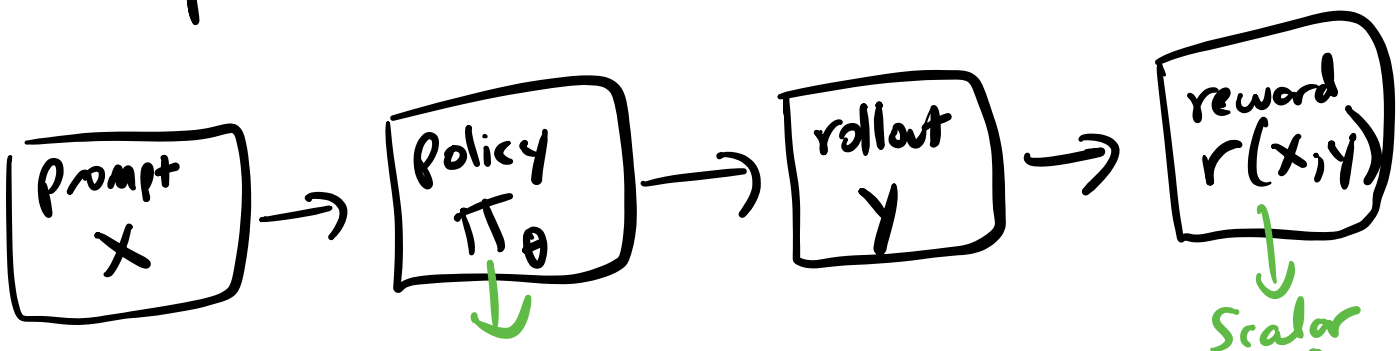
we left off:



$$\pi_\theta(y|x) = \pi_\theta(y_1|x) \cdot \pi_\theta(y_2|x,y_1) \cdot \pi_\theta(y_3|x,y_1,y_2)$$

goal: maximize expected reward

$$J(\theta) = \mathop{E}_{y \sim \pi_\theta(\cdot|x)} \left[r(x,y) \right]$$

$$= \sum_y \pi_\theta(y|x)\, r(x,y)$$

we need $\frac{dJ}{d\theta}$, $J =$

$$\frac{dJ}{d\theta} = \sum_Y r(x,y) \frac{d\pi_\theta(y|x)}{d\theta}$$

↑
what happens if we try to approximate this via sampling.

we can do this for

$$J(\theta) = \sum_Y \pi_\theta(y|x) r(x,y) \approx \frac{1}{N} \sum_{i=1}^{N} r(x,y)$$

but if we do it for $\frac{dJ}{d\theta}$

$$\frac{dJ}{d\theta} = \sum_Y r(x,y) \frac{d\pi_\theta(y|x)}{d\theta}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \pi_\theta(y|x) r(x,y) \frac{d\pi_\theta(y|x)}{d\theta}$$

↳ extra term, so w/o this term we are not approximating $\frac{dJ}{d\theta}$

multiply by $\frac{\pi_\theta(y|x)}{\pi_\theta(y|x)}$

$$\frac{dJ}{d\theta} = \sum_y \pi_\theta(y|x) r(x,y) \frac{d\pi_\theta(y|x)}{d\theta} \cdot \frac{1}{\pi_\theta(y|x)}$$

$$= E_{y \sim \pi_\theta(\cdot|x)} \left[r(x,y) \frac{1}{\pi_\theta(y|x)} \frac{d\pi_\theta(y|x)}{d\theta} \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} r(x,y_i) \frac{1}{\pi_\theta(y_i|x)} \frac{d\pi_\theta(y_i|x)}{d\theta}$$

"log trick"

↳ via chain rule, we know

$$\frac{d \log \pi_\theta(y|x)}{d\theta} = \frac{d \log \pi_\theta(y|x)}{d\pi_\theta(y|x)} \cdot \frac{d\pi_\theta(y|x)}{d\theta}$$

$$\frac{d \log x}{dx} = \frac{1}{x}$$

$$= \frac{1}{\pi_\theta(y|x)} \cdot \frac{d\pi_\theta(y|x)}{d\theta}$$

$$\frac{dJ}{d\theta} = \mathop{E}_{y \sim \pi_\theta(\cdot|x)} \left[r(x,y) \frac{d \log \pi_\theta(y|x)}{d\theta} \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} r(x,y_i) \frac{d \log \pi_\theta(y_i|x)}{d\theta}$$

↳ we want to maximize $J(\theta)$, so for gradient descent we minimize $-J(\theta)$

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} r(x,y_i) \log \pi_\theta(y_i|x)$$

↳ this loss fn has exactly the gradient we want, $-\frac{dJ}{d\theta}$

↳ PyTorch: loss = -(rewards * logprobs).mean()

how does this compare to SFT?

↳ $\log \pi_\theta(y|x) = \sum_t \log \pi_\theta(y_t | x, y_{<t})$]

↳ loss for SFT, $-\log \pi_\theta(y|x)$

SFT: $L = -\frac{1}{N} \sum_i \log \pi_\theta(y_i|x)$

↳ y_i come from data

REINFORCE: $L = -\frac{1}{N} \sum_i r(x, y_i) \log \pi_\theta(y_i|x)$

↳ y_i comes from the policy

↳ main diff: in REINFORCE, we use the same loss as SFT but on samples from policy, weighted by reward

↳ so what happens if $r(x,y) = \begin{cases} 1 & \text{if correct} \\ 0 & \text{if incorrect} \end{cases}$

↳ incorrect rollouts y_i get ignored

↳ for correct rollouts, just normal SFT

---

reward scaling

↳ if all rewards are positive, every rollout's probability is increased

↳ fix: subtract "baseline" b from every reward

$$\frac{dJ}{d\theta} \approx \frac{1}{N} \sum_{i=1}^{N} \left(r(x,y_i) - b\right) \frac{d \log \pi_\theta(y_i|x)}{d\theta}$$

↳ advantage $A_i = r(x,y_i) - b$

GRPO: introduced DeepSeek R1

↳ given a prompt x, we sample G different rollouts, a "group"

↳ compare the rollouts to each other

$$A_i = \frac{r(x, y_i) - \text{mean}(r_1 \dots r_G)}{\text{std}(r_1 \dots r_G)}$$

example: x is a math problem

$G = 4$

↳ rewards: 1, 0, 0, 1

rewards - mean: 0.5, -0.5, -0.5, 0.5

↳ rewards: 1, 1, 1, 1

rewards - mean: 0, 0, 0, 0

↳ no updates

↳ rewards: 0, 0, 0, 0

↳ no updates