Maximize expected reward

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

keep reward model fixed

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

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

↳ REINFORCE (Williams, 1992)

↳ "log trick", multiply / divide by $p_\theta(y|x)$

$$= \sum_y r(x,y)\, p_\theta(y|x) \boxed{\frac{\frac{d}{d\theta} p_\theta(y|x)}{p_\theta(y|x)}}$$

↳ $\frac{d}{d\theta} \log p_\theta(y|x)$

↳ remember $\frac{d}{d\theta} \log v = \frac{1}{v} \cdot \frac{dv}{d\theta}$

$$= \frac{dJ}{d\theta} = \sum_y r(x,y)\, P_\theta(y|x)\, \frac{d}{d\theta} \log P_\theta(y|x)$$

↳ this itself is an expectation!

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

1. Given a prompt x, we sample a response y from our current model P_θ
2. Compute reward $r(x,y)$ from our frozen reward model
3. Adjust θ in direction $\frac{d}{d\theta} \log P_\theta(y|x)$ scaled by our reward

↳ usually, we sample 1 to 16/32 y's per prompt x to compute the update

↳ don't generally use the raw reward $r(x,y)$, we usually subtract a baseline first, which reduces variance

$$r(x,y) - b$$ } ↳ advantage

↳ baseline

↳ raw reward

↳ if positive, increase the prob of y

↳ if negative, decrease prob. of y

advantage $A(x,y) = r(x,y) - b(x)$

↳ if $A(x,y) > 0$, better than usual

↳ if $A(x,y) < 0$, worse than usual

batch:

$x_1, y_1, r(x_1, y_1)$

$x_2, y_2, r(x_2, y_2)$

$\vdots$

$x_n, y_n, r(x_n, y_n)$

} compute b as mean batch reward

↳ this strategy doesn't consider prompt x

↳ to get a better estimate of $b(x)$

↳ sample multiple y_i for that x

↳ use $b(x)$ = mean reward over y_i

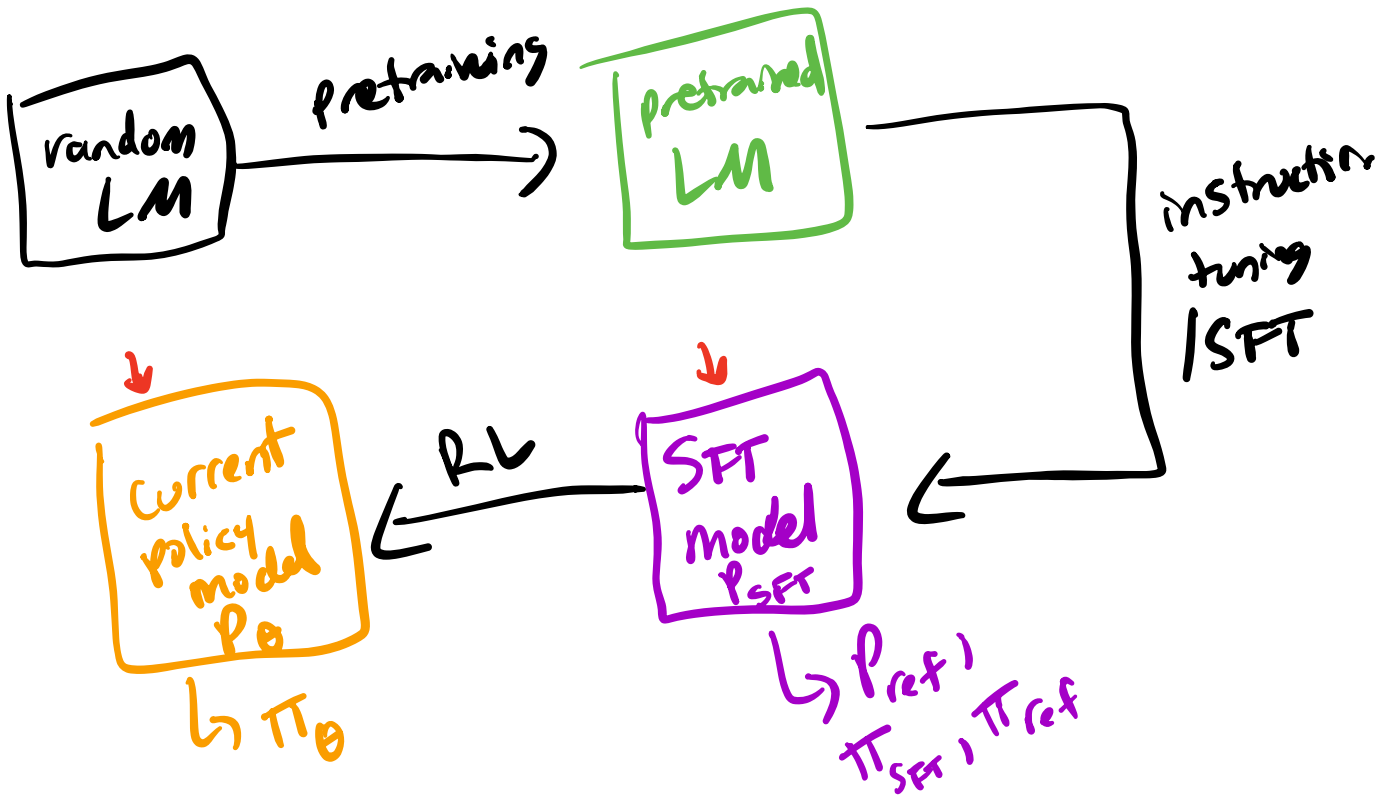
↳ RLOO, GRPO

Reward hacking:

↳ model finds unintended ways to maximize the reward

↳ solutions

1. train a better reward model
   ↳ get more / higher quality pref. data
   ↳ start w/ a bigger LM
2. penalize large deviations from SFT model



↳ use KL divergence to measure difference between P_θ and P_{SFT}

$$D_{KL}(P \| Q) = \mathop{E}_{Y \sim P(\cdot|x)} \left[\log P(y|x) - \log Q(y|x) \right]$$

↳ if P and Q are equal, $D_{KL}(P \| Q) = 0$

↳ the more P places probability on responses y where Q doesn't place prob, the higher the KL penalty

$$A(x,y) - \beta D_{KL}(P_\theta \| P_{SFT})$$

↳ hyperparameter that controls strength of penalty

Key differences between RL vs. SFT

↳ RL's updates are scaled by the prob. of the model P_θ producing y, it penalizes low-reward responses that the model actually produces

↳ SFT has no such scaling, pushing model towards y it wouldn't currently produce