→ RL-HF and PPO (DPO advance)

- used for alignment

AI- alignment.

To force LLM behave in a specific way eg assistant to behave in a friendly way.

→ Don't use bad or racist words and in particular style.

RL: RL is concerned with how an intelligent agent should take action in an environment to maximize cummulative reward. (select a policy that maximize the return when agent acts accordingly to it)

Policy: rules how the agent select the action to perform given the state it is in (probability to move next given current)

The goal of RL is to select a policy that maximizes the expected return when agent acts according to it (optimize policy)

Connection to LLM.

Agent, LLM State, prompt(n) Action, next token Policy LLM

Reward, rewarding for good and not rewarded for bad response

How to reward / Penalize?

---

we'll build a reward model

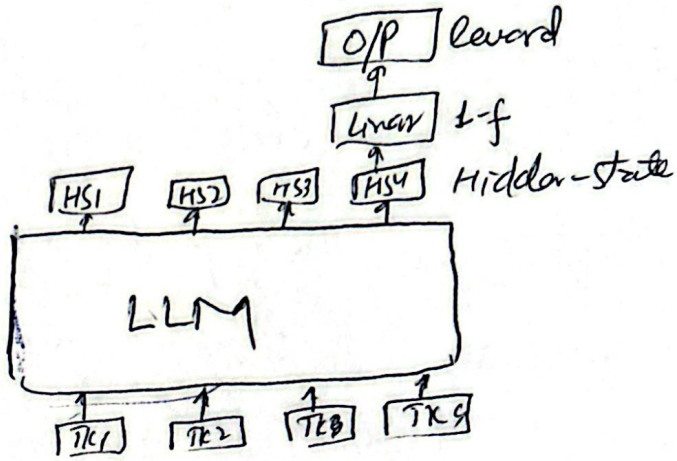
→ Instead of one answer, we generate multiple answers for a prompt (by using high temperature) and then we ask people to choose the answer they prefer. Using the dataset of preference, we create a model that generate numeric reward for each Q and A.

→ we create a NN that gives the highest reward to chosen answer and lower to others

REWARD MODEL ARCHITECTURE.

→ we take a prebained LLM, an input (Q/A) after concatenating them, the from last hidden state passes to linear layer of (1-feature) to generate a single value considered as reward for model

→ we also have to tell the model to give high reward to chossen Q/A.



loss-function.

good ↓ bad ↓

$$Loss = -\log\left(\sigma\left(r(x, y_w) - r(x, y_i)\right)\right)$$

consider we have two answers Q.

two cases,

$$r(x, y_w) > r(x, y_i)$$

→ Sigmoid will return value > 0.5, loss will be small

$$r(x, y_w) < r(x, y_i)$$

→ Sigmoid will retur value < 0.5, loss will be high

Trajectories.

$$\pi^* = \arg\max_{\pi} J(\pi)$$

(* policy)

The expected return over policy is expected return over all trajectories

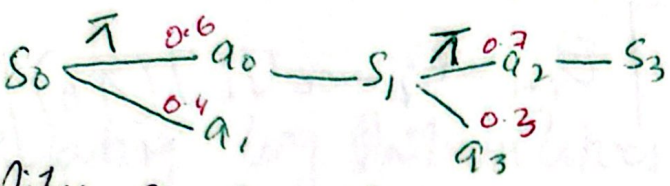
$$J(\pi) = \int P(\tau/\pi) R(\tau) = \underset{\tau \sim \pi}{E}[R(\tau)]$$

trajectory (path) is a series of (action, states), starting from initial state ($T = (s_0, a_0, s_1, a_1)$)

we will model the next state as stochastic (cat doesn't move correctly always)

$$s_{t+1} \sim P(\cdot | s_t, a_t)$$

→ Policy tells the probability of next state to move in all direction with given probability



action depend on the current state and independent. we multiply all to get trajectory

Probability of Trajectory

$$P(T|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

$\rho_0(s_0)$: start; $P(s_{t+1}|s_t, a_t)$: of next state give current; $\pi(a_t|s_t)$: probability of choosing action given state

we will always work with discounted Reward (immediate instead of future without discount. It will take long to reach optimum, because γ^8 → will decay so, intermediately (

$$R(T) = \sum_{t=0}^{\infty} \gamma^t r_t \quad \leftarrow (0 < \gamma < 1)$$

→ Trajectories in LLM.

states = prompt, policy to optimize = "

action = next token,

→ Trajectories are series of prompt or token we have chosen.

Policy - Gradient Optimization.

we want to maximize this function actually

$$J(\pi_\theta) = \underset{\tau \to \pi}{E}[R(\tau)]$$ → to, for max we use gradient ascent

trajectory = series of pairs and next states

objective function

Expression of gradient.

$$\nabla_\theta J(\theta) = \nabla_\theta \underset{\tau \to \pi}{E}[R(\tau)]$$

$$\Rightarrow \nabla_\theta \int_\tau P(\tau|\theta) R(\tau)$$

$$= \int_\tau \nabla_\theta P(\tau|\theta) R(\tau)$$

$$\because \nabla_\theta \log P(\tau|\theta) = \frac{1}{P(\tau|\theta)} \cdot \nabla_\theta P(\tau|\theta)$$

$$= \int_\tau P(\tau|\theta) \nabla_\theta \log P(\tau|\theta) R(\tau)$$

$$= \underset{\tau \to \pi_\theta}{E}[\nabla_\theta \log P(\tau|\theta) R(\tau)]$$

$$\because P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\, \pi_\theta(a_t|s_t)$$

$$\nabla_\theta \log P(\tau|\theta) = \nabla_\theta \log \rho_0(s_0) + \sum \nabla_\theta P(s_{t+1}) + \nabla_\theta \log \pi_\theta(a_t|s_t)$$

(constant)

$$\Rightarrow \underset{\tau \to \pi_\theta}{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi(a_t|s_t) R(\tau)\right]$$

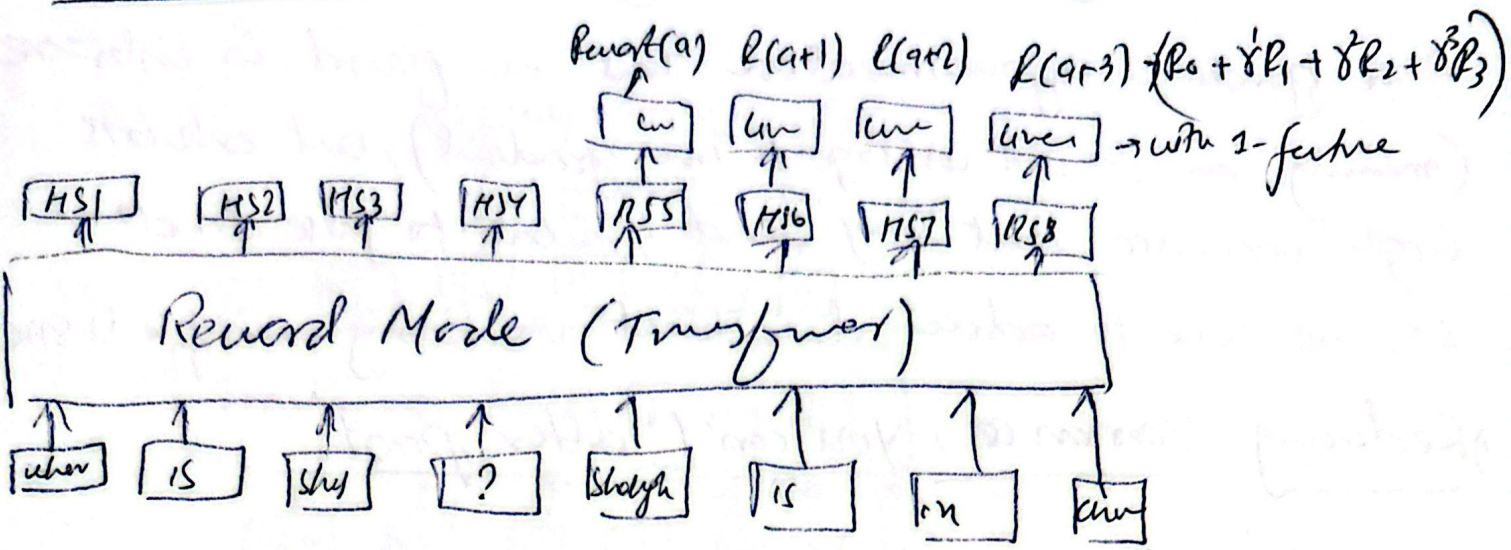
→ This expectation is over all trajectories, in LLM which can be highly costly. The over way to elivate the issue to approximate it using sample mean by collecting a set of D trajectories.

$$\hat{g} = \sum_{\tau \in D} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)$$

Policy Gradient Algorithm

$$\theta_{k+1} = \theta_k + \alpha \nabla_k J(\pi_\theta)\big|_{\theta_k}$$

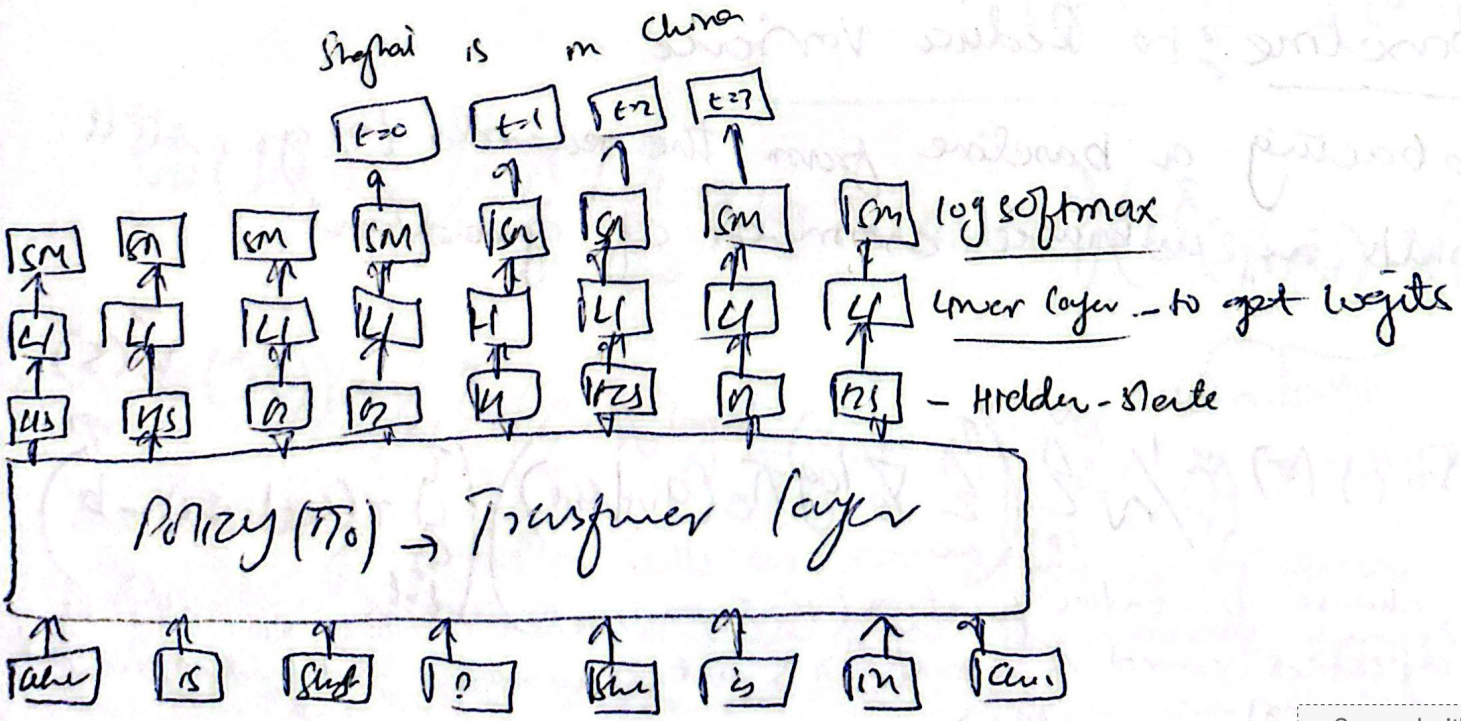
Calculating reward for each trajectory.



Calculating log Probabilities of our Policy.

after calculating policy and reward. we run gradient ascent. (to optimize LLM)

$$\hat{g} = \frac{1}{|D|} \sum_{\tau \in D} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)$$



Problems with Gradient Policy Optimization

1) the gradient approximization that we found is unbiased (meaning as it will converge to True gradient), but exhibits high variance that may lead gradient to false direction - so, we have to reduce it without increasing sample D size.

1) Reducing Variance: you can't alter Past

* each action can't alter the reward, we look in past

$$\nabla_\theta J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, R(\tau)\right]$$

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t'=t}^{T} r(s_{i,t}, a_{i,t})\right)$$

rewards to go

For each action, we are multiplying rewards that come before the action was taken. It is proved that Past terms cancel out in expectation, we removed the

2) Baseline: to Reduce Variance.

→ Subtracting a baseline from the rewards to go, still results in unbiased estimator of gradient.

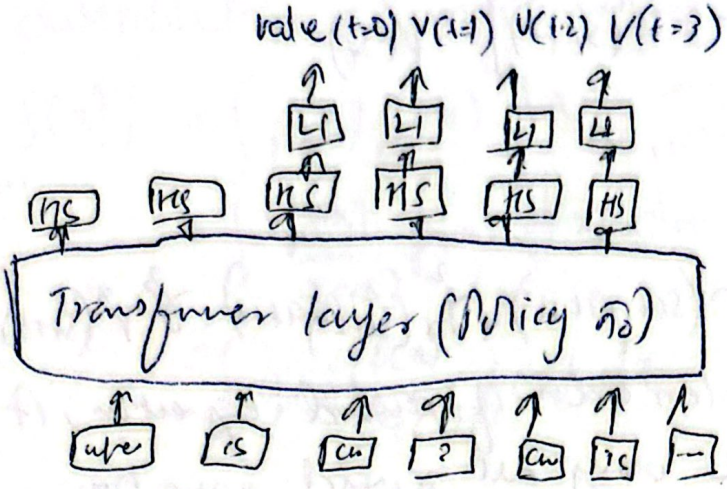
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t'=t}^{T} r(s_{i,t}|a_{i,t}) - b\right)$$

$\hat{V}(s)$

→ we choose b = value function (that indicates the expected reward of agent in s act (a) according to policy (π)) $\Rightarrow V^{\pi}(s)$

can also be dependent on State

How do we Estimate $\rightarrow V^{\pi}(s)$:



3) Reducing Variance - introducing Q and V; (advantage function)

Q = Reward to go fcn

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left(\sum_{t=t}^{T} \gamma(s_{i,t} | a_{i,t}) \right)$$

$$\hat{Q}(s,a) = \underset{s'}{E}[r(s,a)] + \gamma \underset{a'}{E}[\hat{Q}(s',a')]$$

So,

$$\nabla_\theta (J(\theta)) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left(\hat{Q}(s_{i,t}, a_{i,t}) - \hat{V}^{\pi}(s) \right)$$

$$\nabla_\theta (J(\theta)) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left(\hat{A}(s,a) \right)$$ advantage function

→ The advantage function tells us how better is to choose a particular action in a state over the average expectation we get by choosing randomly an action in some state.

→ how good is particular than the average action in st

Estimating the advantage term.

$$\hat{A}^{\pi}(s_t, a_t) = [r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})] - V^{\pi}(s_t)$$

or

for three step

$$Q^{\pi}(s_t, a_t)$$

$$\hat{A}^{\pi}(s_t, a_t) = [r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \gamma^3 V^{\pi}(s_{t+3})] - V^{\pi}(s_t)$$

→ If we stop early, use less order of reward equation, it will have high bias, while using all reward equation and doing no approximation with value function we get high variance. In order to solve bias-variance, we take weighted sum of term & get Generalized Advage Estimate

$$\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

$$\hat{A}_t = \delta(t) + \gamma\lambda \hat{A}(t+1)$$

↳ decaying parameter

This is recursive formulae that for last terms equals the first expansion. For the second last term equals to second expansion weighted by $(1)e^{t}$

Advantage for LLM's

So, we want to maximize the reward for action that are good for a state (s)

Problem #2 : Gradient - Policy Optimization.

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi)|_{\theta_k}$$

→ As we are sampling per model, while applying gradient ascent, we have to do alot of calculation V, R, Φ etc and we update the parameter for all trajectories, hence many times by taking small step (which make is complicated)

Importance Sampling & Off-policy learning

Importance sampling allow us to evaluate an expectation over a distribution x using sample taken from distribution Y.

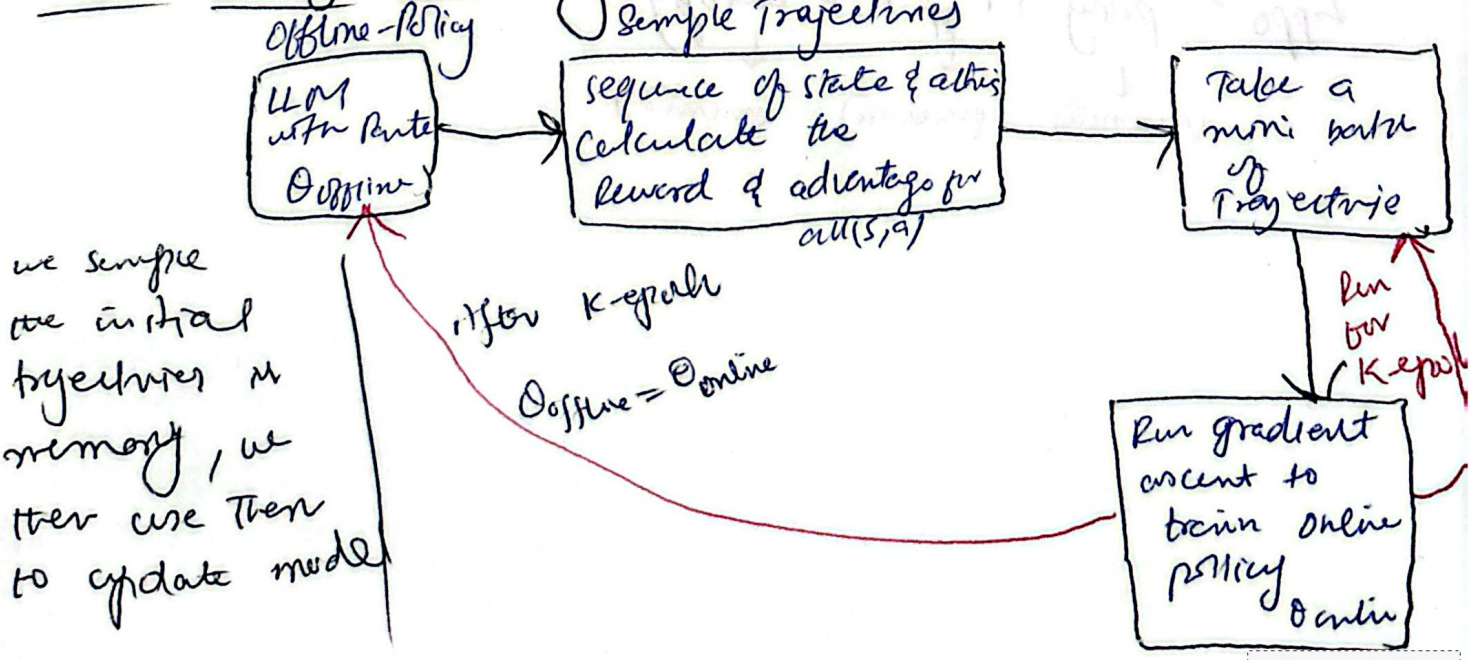
$$\begin{aligned}
\underset{x\sim p(x)}{E}[f(x)] &= \int p(x) f(x)\, dx \\
&= \int \frac{q(x)}{q(x)} p(x) \cdot f(x)\, dx \\
&= \int q(x) \frac{p(x)}{q(x)} f(x)\, dx \\
&= \underset{x\sim q(x)}{E}\left[\frac{p(x)}{q(x)} f(x)\right]
\end{aligned}$$

So,

$$\nabla_\theta U(\theta) \approx \frac{1}{N}\sum_{i=1}^{\tilde{N}}\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right) A^{\pi}(s,a) \quad \text{becomes.}$$

$$\nabla_{\theta_{online}} J(\theta_{online}, \theta_{offline}) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T} \nabla_{\theta_{online}} \frac{\log \pi_{online}(a_{i,t}, s_{i,t})}{\log \pi_{offline}(a_{i,t}, s_{i,t})}\right) A^{\pi}(s,a)$$

→ Off-policy learning.



The PPO loss: Proximal Policy Optimization

$$L^{clip}_{policy} = \min_{i}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t,\ clip\left(\frac{\pi_\theta(a_t,s_t)}{\pi_{\theta_{old}}(a_t,s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right)$$

(if high or if low) (clip to this) (clip to this)

→ Because we want the model to not be very sure or unsure to select this word or not at all we want a little surety (too sure will make it select token all time)

$$L^{VF}_{t} = \frac{1}{2}\left\| V_{\theta(s)} - \left(\sum_{t=0}^{T} \gamma^t r_t | s_0 = s\right)\right\|^2_2$$

output value state

actual value based on trajectories we sampled

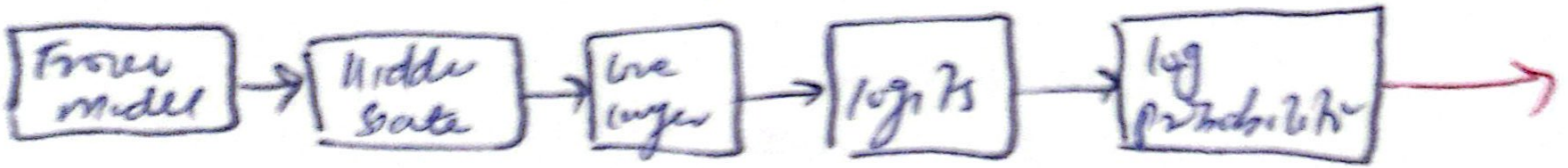
$$L_{entropy} = -\sum_{x} p(x)\log p(x) \rightarrow$$ to force our model to explore more options, makes model right (select token to explore other options)

$$L_{PPO} = L_{policy} + c_1 L_{VF} + c_2 L_{entropy}$$

(maximize) (minimize) (maximize)

Reward hacking:

→ we also want model to given answer that only gives a good reward but also makes sense and one similar to unaligned model.



we calculate KL divergence to penalize the reward by

$r_0 - KL(LP_F \| LP_A)$

irect-Preference Optimization (DPO) & (Bradely Terry)

Braley-Terry Model.

temperature is a parameter that controls the level of randomness in how deterministic (or random) a model's output is
(0.0 = deterministic)
(0.3 - 0.7 = Balanced)
(1 = high randomness)

our goal is to train a model that gives a numeric score (reward) to pair of Q/A that resembles to the preference chosen by people.

$$P(y_w > y_l) = \frac{e^{r^*(x, y_w)}}{e^{r^*(x, y_w)} + e^{r^*(x, y_l)}}$$

One way to maximize our preference correctly is use Maximum Likelihood Estimation. MLE can estimate the parameters of reward model such that the probability of ranking preference is correctly maximized.

Derivation of loss that maximize P_r:

$$\frac{e^A}{e^A + e^B} = \frac{e^A/e^A}{e^A/e^A + e^B/e^A} = \frac{1}{e^{A-A} + e^{B-A} + 1 - 1}$$

$$\Rightarrow \frac{1}{1 + \left(\frac{e^A}{e^A + e^B} - 1\right)} \Rightarrow \frac{1}{1 + (e^B/e^A)}$$

$\because \frac{e^A}{e^A + e^B} = \sigma(A - B)$

$\because f(x) = \frac{1}{1 + e^{-x}}$

$$\Rightarrow 1/1 + e^{B-A} \Rightarrow 1/1 + e^{-(A-B)} \Rightarrow \boxed{\sigma(A - \beta)}$$

Hence,

$$L_{DPO} = -E_{(x, y_w, y_l) \sim D}\left[\log \sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right]$$

RLHF-objective

$$J_{RLHF} = \max_{\pi_\theta} E_{x \sim D, y \sim \pi_\theta(y|x)}\left[r_\theta(x, y) - \beta D_{KL}[\pi_\theta(y|x) \| \pi_{ref}(y|x)]\right]$$

reward — constrict the model to be not so different from original one

What does it mean to maximize an objective fn?

The Optimization Problem

- means to find values of one or more variables such that the fn is maximized. (for unconstrained problem)
(we find derivative of fn, set to zero, get value)

constraint optimization: (J_{PLHf})
these are not differentiable so, we can not run gradient descent and forced to used algorithms like PPO.

→ Optimal policy.
It is possible mathematically but very computationally heavy.

$$\pi_r(y|x) = \frac{1}{Z(x)} \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right)$$

$\log Z(x) \sum_y \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right)$

But suppose, magically we have optimal policy $\pi^*(y|x)$, what would be reward fn for optimal policy?
By leveraging:

$$r(x,y) = \beta \log \frac{\pi^*(y|x)}{\pi_{ref}(y|x)} + \beta \log Z(x)$$

∵ Bt-model → $P(y_w > y_l) = \frac{e^{r(x,y_w)}}{e^{r(x,y_w)} + e^{r(x,y_l)}} = \sigma(A - B)$

we plug the reward and get.

$$P(y_w > y_l) = \sigma\left(\beta \log \frac{\pi^*_\theta(y_w|x)}{\pi_{ref}(y_w|x)} + \beta \log(Z) - \beta \log \frac{\pi^*_\theta(y_l|x)}{\pi_{ref}(y_l|x)} - \beta \log(Z)\right)$$

(maximize this

if in actual we don't have optimal policy & reward we pretend we have policy.

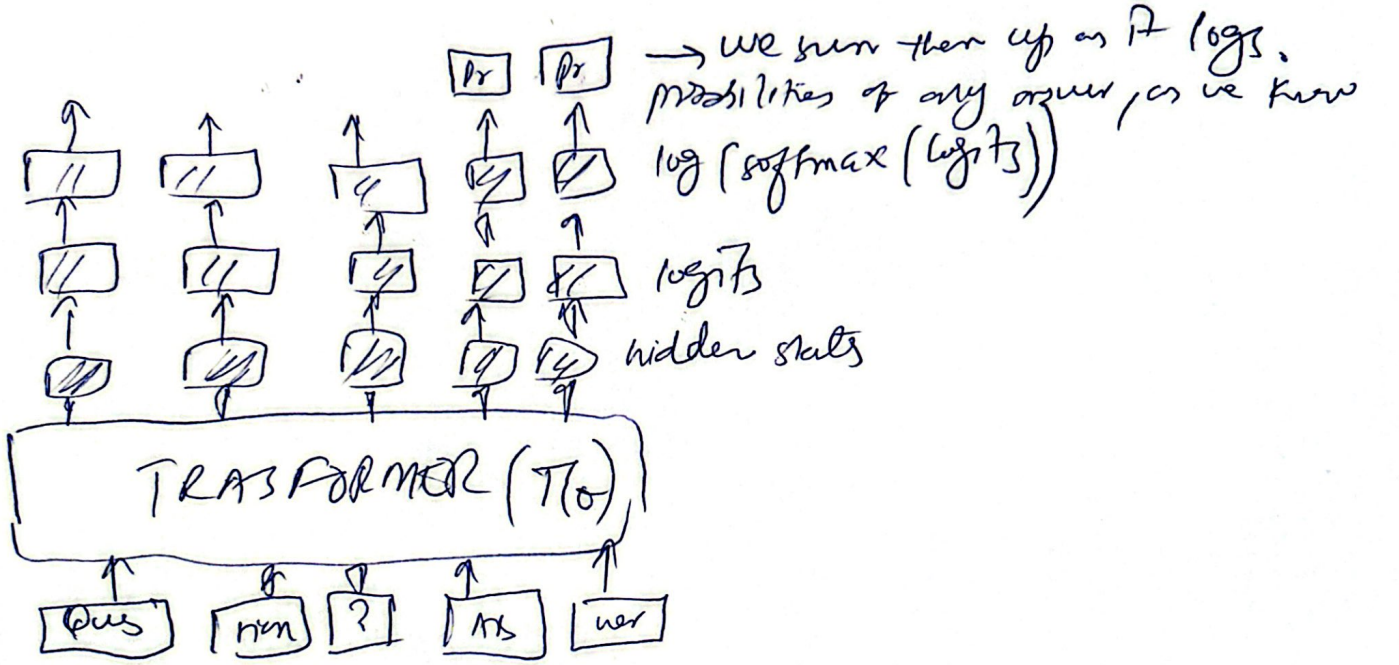
$$L_{DPO}(\pi_\theta ; \pi_{ref}) = -\hat{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)} \right) \right]$$

So, instead of optimizing reward function, we are optimizing policy (which depends on optimal reward fun).

Computing log-Probabilities:-

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

y_w: chosen answer, y_l: failed answer



combine question and chosen answer