Alignment of Language Models – Reward Maximization

Advances in Large Language Models

ELL8299 · AIL861

Gaurav Pandey
Senior Research Scientist, IBM Research

Reward Maximization

- Best-of-N policy
- Reward Maximization Objective Formulation
- Vanilla Policy Gradient
  - The REINFORCE gradient estimator
  - The problem of high variance - Baseline subtraction
  - Final algorithm
- Towards Actor/Critic methods
  - The problem of credit assignment
  - Q-functions, Value functions and advantage
  - Estimating the value function and advantage
  - Actor-Critic Family of algorithms

Reward Maximization

- Best-of-N policy
- Reward Maximization Objective Formulation
- Vanilla Policy Gradient
  - The REINFORCE gradient estimator
  - The problem of high variance - Baseline subtraction
  - Final algorithm
- Towards Actor/Critic methods
  - The problem of credit assignment
  - Q-functions, Value functions and advantage
  - Estimating the value function and advantage
  - Final algorithm

Best-of-N policy

Given

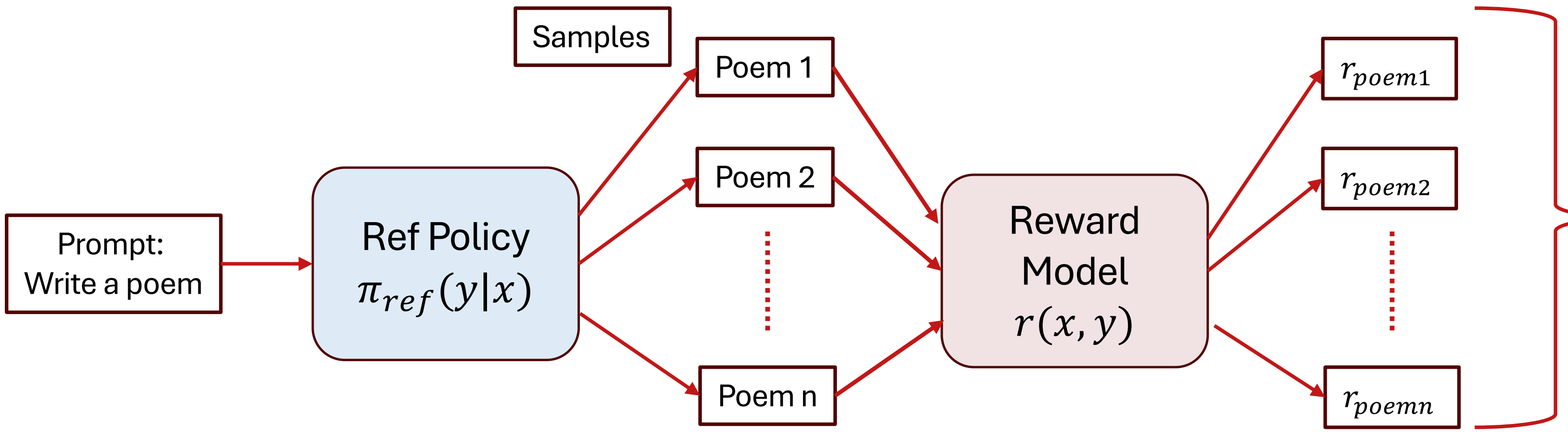
- Base policy or reference policy $\pi_{ref}(y|x)$
  - Often, an instruction tuned LM that serves as the starting point of alignment
- Reward Model $r(x, y)$

Aim – To generate outputs with high reward

Solution

- Sample multiple outputs from the policy π_{ref}
- Score each output using the reward model
- Return the output with the highest reward

Best-of-N policy



Pick the one with the highest reward

Challenge:

- Too expensive during inference
- Can't get better iteratively
- Reward Model may not be available during inference
    - Verifiable rewards

Reward Maximization

- Best-of-N policy
- Reward Maximization Objective Formulation
- Vanilla Policy Gradient
  - The REINFORCE gradient estimator
  - The problem of high variance - Baseline subtraction
  - Final algorithm
- Towards Actor/Critic methods
  - The problem of credit assignment
  - Q-functions, Value functions and advantage
  - Estimating the value function and advantage
  - Final algorithm

The reward-maximization objective

Given

- Base policy or reference policy $\pi_{ref}(y|x)$
  - Often, an instruction tuned LM that serves as the starting point of alignment
- Reward Model $r(x, y)$

Aim

- To find a policy $\pi_{\theta^*}(y|x)$
  - That generated outputs with high reward.
  - That stay close to the reference policy.

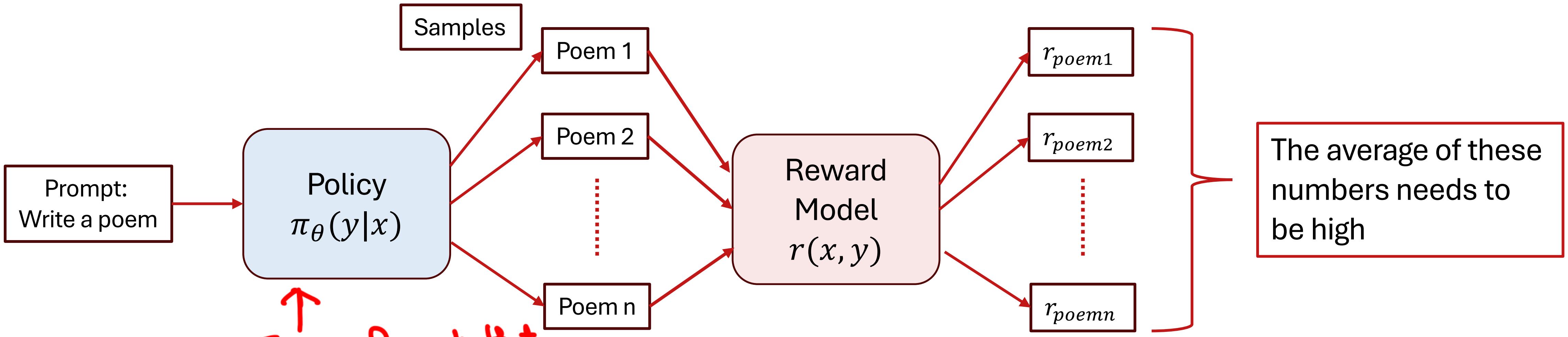
Why care about closeness to π_{ref}?

Reward Models are not perfect.

- They have been trained to score only selected natural language outputs.
- The policy can hack the reward model – generate outputs with high reward but meaningless
- An input can have multiple correct outputs (Write a poem?)
  - Reward maximization can collapse the probability to 1 outputs
  - Staying close to π_{ref} can preserve diversity.

Formulating the objective – Reward Maximization

- What does it mean for a policy to have high reward?



Train θ such that

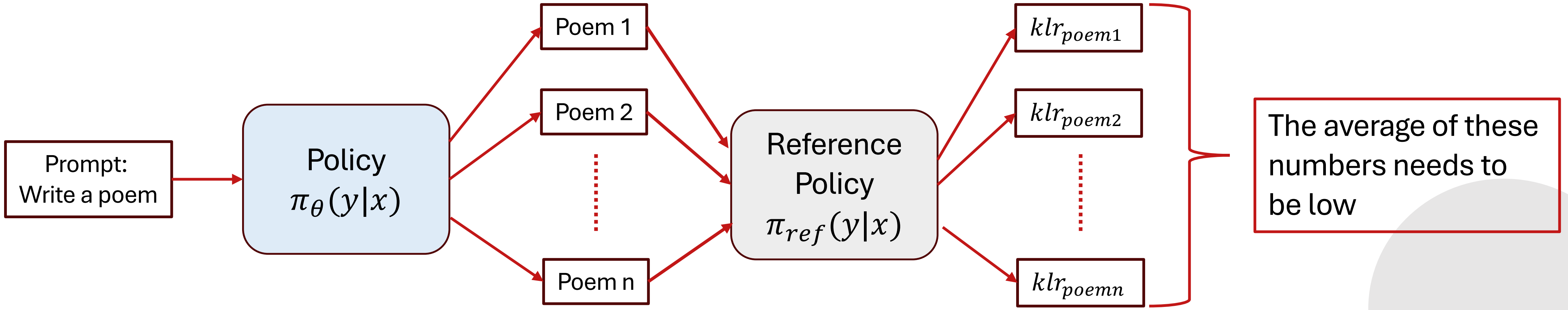
$$\mathbb{E}_{y\sim\pi_\theta(y|x)}\, r(x,y) \uparrow \approx \frac{1}{n}\sum_{i=1}^{n} r(x,y_i)$$

$$y_1,\dots,y_n \sim \pi_\theta(y|x)$$

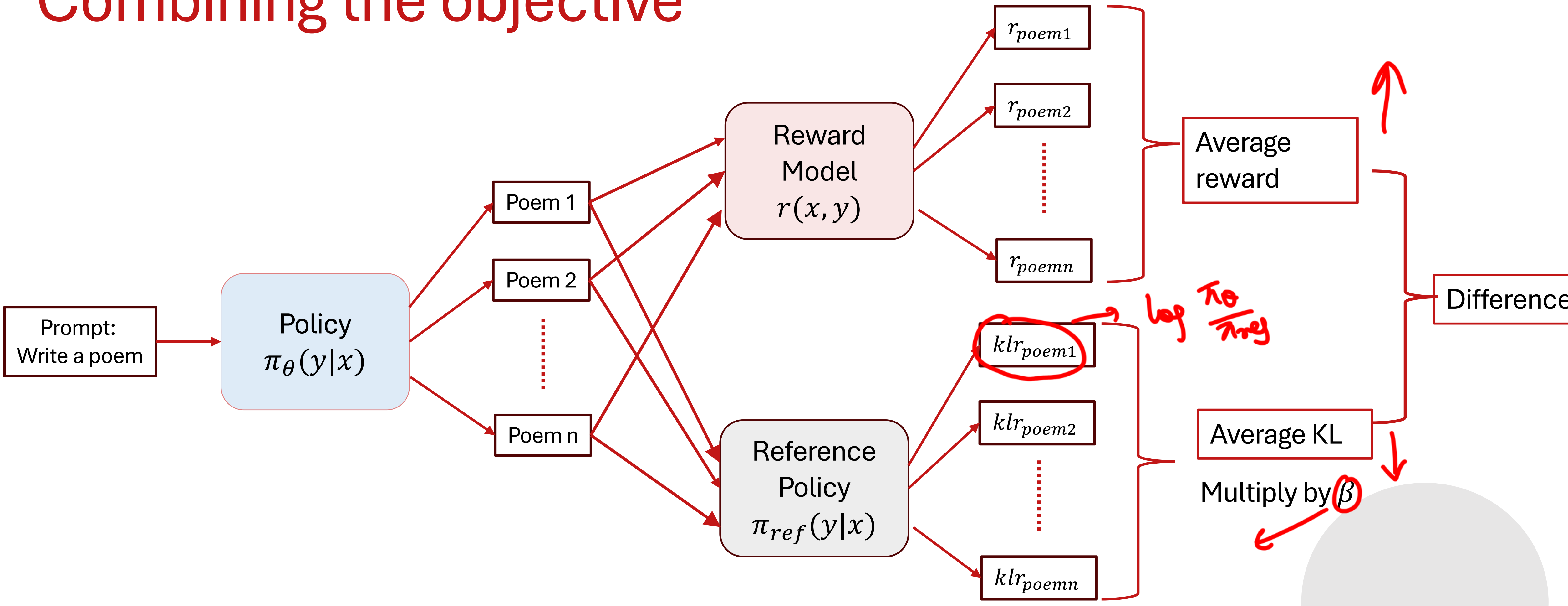
Formulating the objective – closeness to π_{ref}

- How do we capture closeness to π_{ref}?

$$KL\left(\pi_\theta(y|x)\ ||\ \pi_{ref}(y|x)\right) = \mathbb{E}_{y\sim\pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right] \downarrow$$



Combining the objective



Regularized reward maximization

- Maximize the reward

$$\mathbb{E}_{y \sim \pi_\theta(y|x)}\, r(x,y)$$

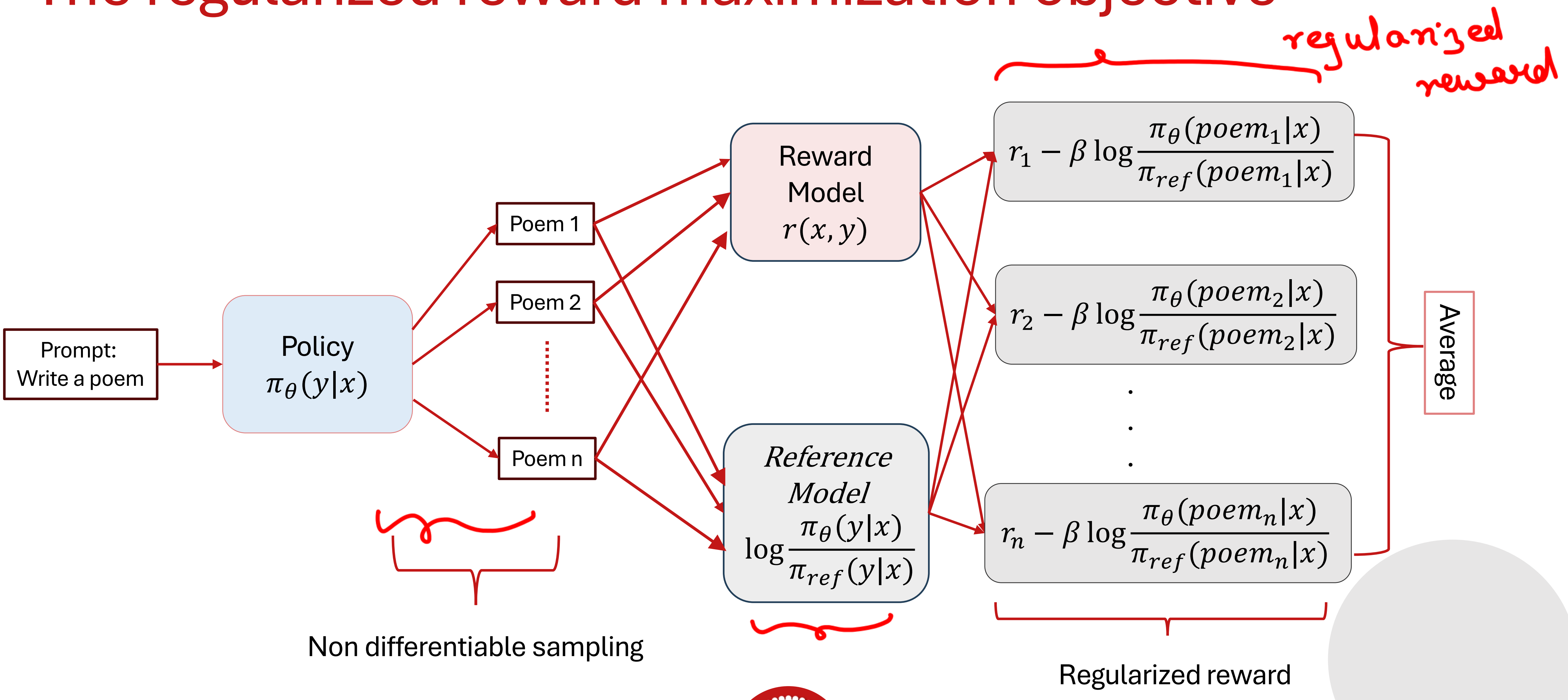
- Minimize the KL divergence

$$\mathbb{E}_{y \sim \pi_\theta(y|x)} \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}$$

- Add a scaling factor β & combine

$$\mathbb{E}_{y \sim \pi_\theta(y|x)} \left[r(x,y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)} \right]$$

The regularized reward maximization objective



Reward Maximization

- Best-of-N policy
- Reward Maximization Objective Formulation
- Towards Vanilla Policy Gradient
  - The REINFORCE gradient estimator
  - The problem of high variance - Baseline subtraction
  - Final algorithm
- Towards Actor/Critic methods
  - The problem of credit assignment
  - Q-functions, Value functions and advantage
  - Estimating the value function and advantage
  - Final algorithm

How to maximize the objective?

- Derive the gradient of the objective.
- Estimate the gradient - the REINFORCE gradient estimator.
- Train using Adam/Adagrad optimization algorithms

$$\nabla_\theta E_{\pi_\theta(y|x)} r(x, y) = \nabla_\theta \sum_{y \in \mathcal{Y}} \pi_\theta(y|x)\, r(x, y)$$

$$= \underbrace{\sum_{y \in \mathcal{Y}} \nabla_\theta \pi_\theta(y|x)\, r(x, y)}$$

How to estimate this using samples?

Computing the gradient

$$\sum_{y \in Y} \nabla_\theta \pi_\theta(y|x)\, r(x,y)$$

→ Not an expectation

- Exact computation of the gradient is intractable
  - Output space is too large
- Can we approximate it using samples?
- To be able to do that, we need an expression of the form

$$\boxed{E_{\pi_\theta(y|x)}[\dots] = \sum_{y \in Y} \pi_\theta(y|x)\, [\dots]}$$

- How to transform the derivative to this desired form?

The log-derivative trick

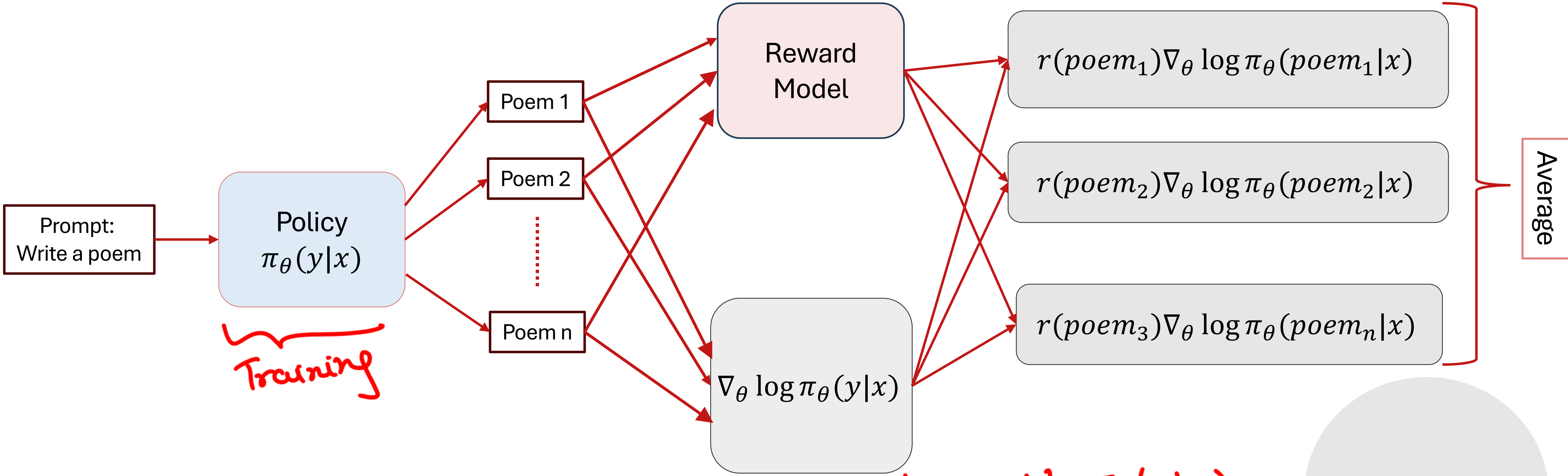
$$\nabla_\theta \log \pi_\theta(y|x) = \frac{1}{\pi_\theta(y|x)} \underbrace{\nabla_\theta \pi_\theta(y|x)} \iff \nabla_\theta \pi_\theta(y|x) = \underline{\pi_\theta(y|x)\, \nabla_\theta \log \pi_\theta(y|x)}$$

- Replacing it in the gradient, we get

$$\sum_{y \in Y} \nabla_\theta \pi_\theta(y|x)\, r(x,y) = \sum_{y \in Y} \pi_\theta(y|x) \left[\nabla_\theta \log \pi_\theta(y|x)\, r(x,y) \right]$$

$$= \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[r(x,y)\, \nabla_\theta \log \pi_\theta(y|x) \right]$$

Estimating the gradient – REINFORCE



Backward pass of $\log \pi_\theta(y|x)$ computation

Estimating the gradient – REINFORCE

$\nabla_\theta E_{\pi_\theta(y|x)} r(x, y) \approx \frac{1}{n} \sum_{i=1}^{n} r(x, y_i) \nabla_\theta \log \pi_\theta (y_i|x)$

Where $y_1, \ldots, y_n$ are samples from the policy $\pi_\theta(y|x)$.

- The estimator is very noisy – different samples give different gradient estimates.

Expanding the gradient

- Let $y = (a_1, \ldots, a_m)$ be the tokens of y.
- $r(x,y)\nabla_\theta \underbrace{\log \pi_\theta}(y|x) =$ $r(x,y)\,\nabla_\theta \left[\sum_{j=1}^{m} \log \pi_\theta (a_j \mid a_1 \ldots a_{j-1}, x) \right]$

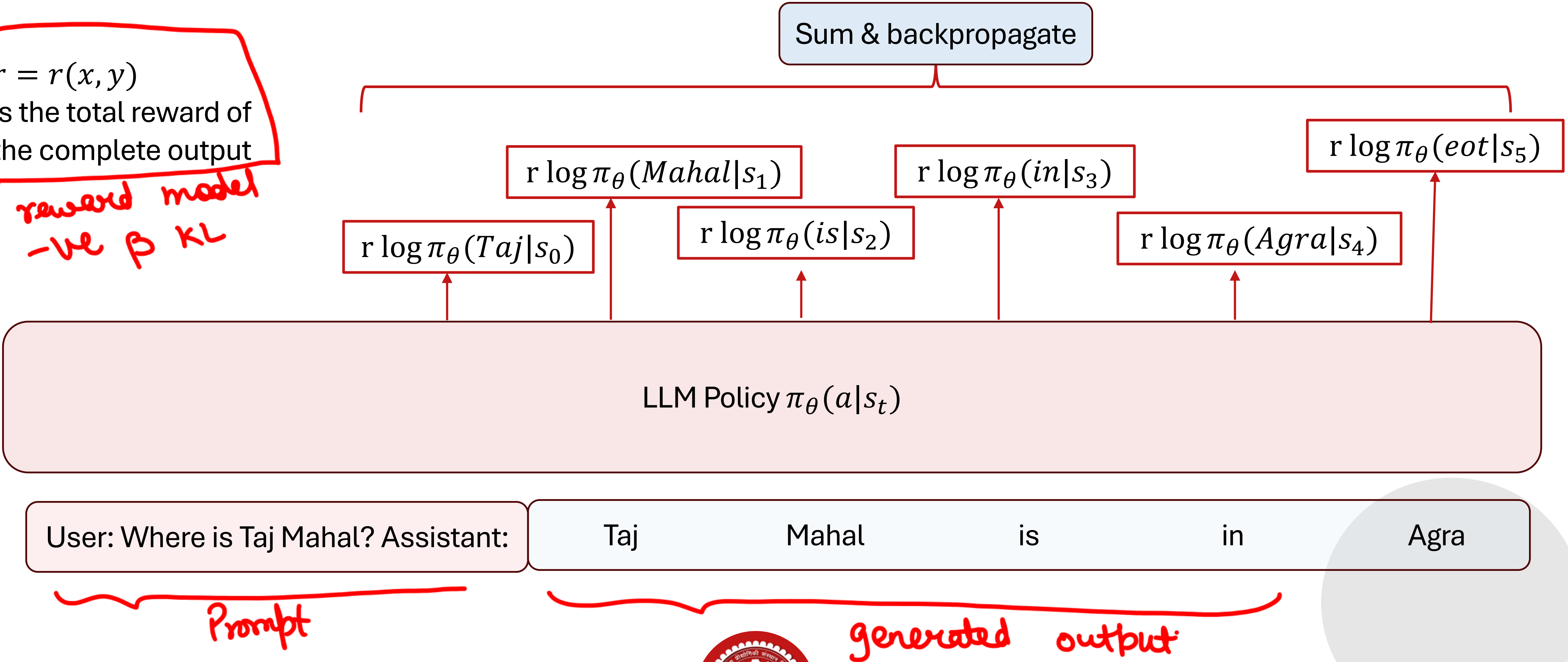
$$= r(x,y) \sum_{j=1}^{m} \nabla_\theta \log \pi_\theta (a_j \mid a_1 \ldots a_{j-1}, x).$$

$$= \sum_{j=1}^{m} r(x,y)\, \nabla_\theta \log \pi_\theta (a_j \mid \underbrace{a_1 \ldots a_{j-1}, x}_{s_j})$$

Implementing the REINFORCE gradient estimate

$r = r(x, y)$
is the total reward of the complete output

reward model
-ve β KL



The Vanilla Policy Gradient Algorithm

- Repeat until convergence
  - Sample a batch of prompts B
  - For each prompt x, sample one or more outputs - $y_1, \ldots, y_n \sim \pi_\theta(y|x)$
  - For each output y
    - Compute the total reward of the output $r(x, y)$
    - Compute the log-probability of the output $\log \pi_\theta(y|x)$
  - To get the loss, multiply the reward with the negative log-probability of the output
    - For each output of the prompt
    - For each prompt in the batch

    and average.
  - Backpropagate the loss to compute the gradient.
  - Use Adam or any other optimizer to update the weights.

Motivating the baseline

- Let X and Y be random variable defined as below

$$P(X = x) = \begin{cases} \frac{1}{10}, & 95 < x \leq 105 \\ 0, & otherwise. \end{cases}, \qquad \mathrm{P(Y = y)} = \begin{cases} \frac{1}{2}, & x \in \{-1,1\} \\ 0, & otherwise \end{cases}$$

- What do samples of XY look like?

$$95, -96, 103, -104$$

(very +ve or very -ve)

- The variance is too high.
- What do samples of $(X - 100)Y$ look like?

E[X]

$$-5, 4, 3, -4$$

(variance is low)

- Centering X can reduce the variance.

Baseline Subtraction to reduce variance

- The REINFORCE gradient estimate is an average of the product of 2 quantities:
  - The reward of the response $r(x, y)$
  - The gradient of the log-prob of output under the policy $\nabla_\theta \log \pi_\theta(y|x)$
- By centering the reward, we can reduce the variance of the estimator.
- Define $\bar{r}(x) = \frac{1}{n}\sum_{i=1}^{n} r(x, y_i)$
- The centered REINFORCE gradient estimate is given by

$$\frac{1}{n}\sum_{i=1}^{n} \underbrace{\left(r(x, y_i) - \bar{r}(x)\right)}_{\hat{r}(x, y_i)} \nabla_\theta \log \pi_\theta(y_i|x)$$

The Vanilla Policy Gradient Algorithm (with baseline)

- Repeat until convergence
  - Sample a batch of prompts B
  - For each prompt x, sample one or more outputs - $y_1, \ldots, y_n \sim \pi_\theta(y|x)$
  - For each output y
    - Compute the total reward of the output $r(x, y)$
    - Compute the log-probability of the output $\log \pi_\theta(y|x)$
  - Compute the average reward (per prompt or per batch)
    - Subtract it from the reward of each output to get baselined reward
    - $\hat{r}(x, \mathrm{y}) = \mathrm{r}(\mathrm{x}, \mathrm{y}) - \bar{r}(x, y)$
  - To get the loss, multiply and average the baselined reward with the negative log-probability of the output
  - Backpropagate the loss to compute the gradient.
  - Use Adam or any other optimizer to update the weights.

Reward Maximization

- Best-of-N policy
- Reward Maximization Objective Formulation
- Vanilla Policy Gradient
  - The REINFORCE gradient estimator
  - Baseline subtraction
  - Final algorithm
- Towards Actor/Critic methods
  - The problem of credit assignment
  - Q-functions, Value functions and advantage
  - Estimating the value function and advantage
  - Actor-Critic Family of algorithms

The REINFORCE gradient estimate

$r = r_s(x, y)$
is the total reward of
the complete output

Sum & backpropagate

$\text{r} \log \pi_\theta(Taj|s_0)$

$+1 \times \log \pi_\theta(Taj|s_0)$

$\text{r} \log \pi_\theta(Mahal|s_1)$

$\text{r} \log \pi_\theta(is|s_2)$

$\text{r} \log \pi_\theta(in|s_3)$

$\text{r} \log \pi_\theta(Agra|s_4)$

$\text{r} \log \pi_\theta(eot|s_5)$

LLM Policy $\pi_\theta(a|s_t)$

User: Where is Taj Mahal? Assistant: Taj Mahal is in Agra

The problem of credit assignment

- The reward of generating a token (like “Taj”) depends on how the future sequence unfolds.
  - Example: “Taj Mahal is in Paris” → negative reward.
  - Example: “Taj Mahal is in Agra” → positive reward.
- This makes credit assignment hard: should “Taj” be encouraged or discouraged?
- The **Q-function** solves this by taking the **expected reward** of generating “Taj” and then continuing according to the policy.
- In other words, the Q-function averages over all possible continuations that follow “Taj,” giving a stable estimate for learning.

using the policy.

The cumulative discounted reward at time t is also known as reward-to-go or return at time t

The Q-function

- Given a state s and action a, the Q function for a policy π is defined as below:

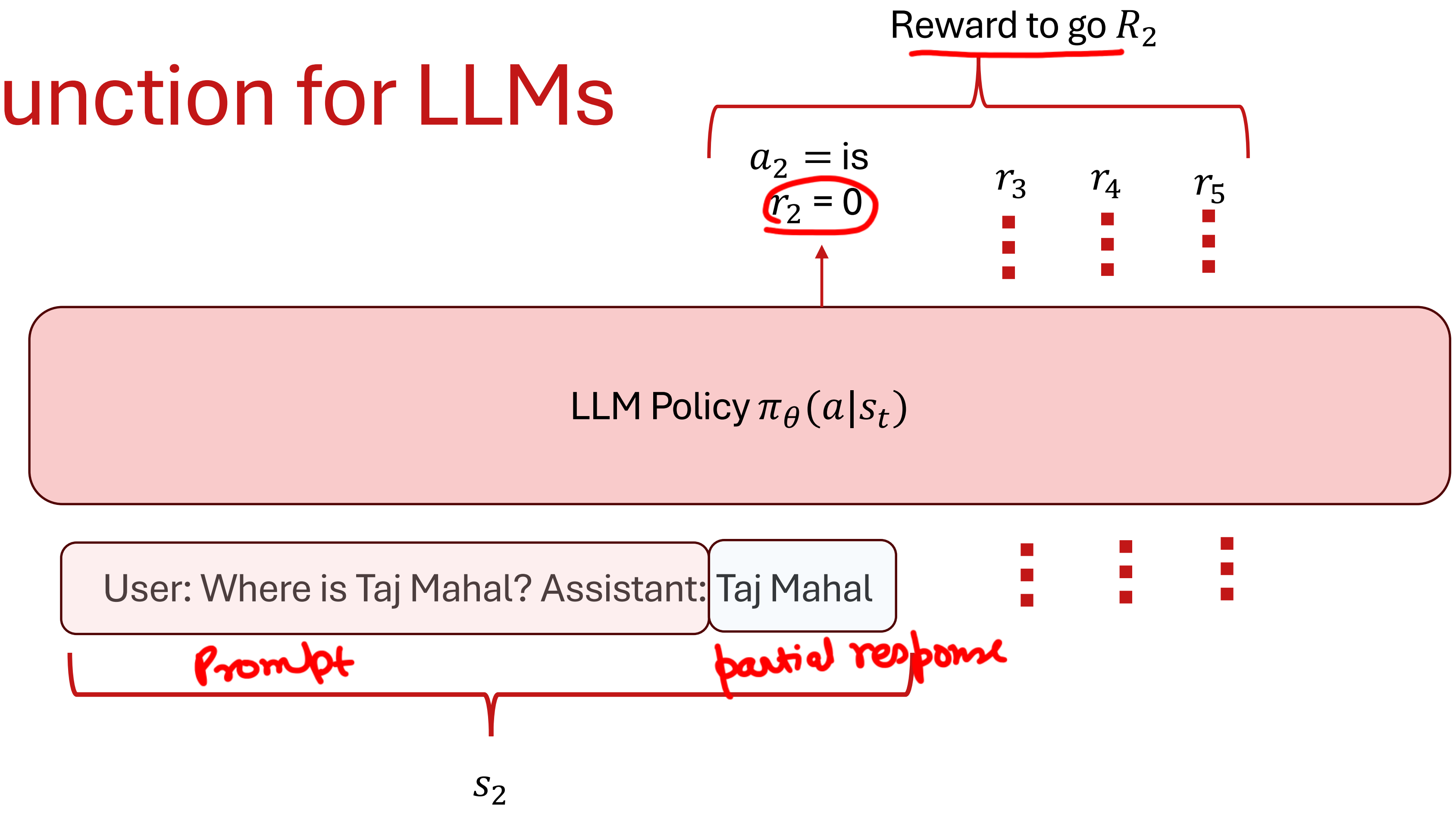
$$Q(s,a) = \mathop{E}_{a_{t+1}, a_{t+2} \ldots} \left[r(s,a) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) \ldots \right]$$

- s: state (context before "Taj")
- a: action (choose token "Taj")
- r_t: reward at step t
- Expectation is over all future actions sampled from policy π

Why discounting?

- Rewards that you get immediately after an action should get higher weight.
- What happens with discounting?
  - If $\gamma = 1$: final reward is propagated **equally to all tokens**.
  - If $\gamma < 1$: earlier tokens get **less credit**, later tokens get more.
- Response generation works well with $\gamma = 1$.
- Discounting is useful mainly when there are **intermediate rewards** (games, robotics)

The Q-function for LLMs



REINFORCE gradient estimation with Q functions

Sum & backpropagate

$r = r(x, y)$
is the reward of
the final output

$Q^{\pi_\theta}(s_1, \text{Mahal}) \log \pi_\theta(Mahal|s_1)$

$Q^{\pi_\theta}(s_3, \text{in}) \log \pi_\theta(in|s_3)$

$Q^{\pi_\theta}(s_5, \text{eot}) \log \pi_\theta(eot|s_5)$

$Q^{\pi_\theta}(s_0, \text{Taj}) \log \pi_\theta(Taj|s_0)$

$Q^{\pi_\theta}(s_2, \text{is}) \log \pi_\theta(is|s_2)$

$Q^{\pi_\theta}(s_4, \text{Agra}) \log \pi_\theta(Agra|s_4)$

LLM Policy $\pi_\theta(a|s_t)$

User: Where is Taj Mahal? Assistant: | Taj | Mahal | is | in | Agra

Prompt

Doesn't matter what gets generated in the future. The "reward" at token "Taj" is fixed.

Value functions – A baseline for the Q functions

- Mean-centering the Q-function can further reduce variance in the estimator.
- The expected Q-function at a state s under policy π is referred to as the value function $V^{\pi}(s)$

$$V^{\pi}(s) = E_{\pi(a|s)}\mathrm{Q}(\mathrm{s}, \mathrm{a})$$

- The baseline subtracted Q is referred to as advantage function.

$$A^{\pi}(s,a) = \underline{Q^{\pi}(s,a) - V^{\pi}(s)} \rightarrow \text{Advantage function}$$

- Intuitively, advantage function captures contribution of the current action over an average action at the same state.

REINFORCE with advantage functions

Sum & backpropagate

$A^{\pi_\theta}(\mathrm{s}_1, \mathrm{Mahal}) \log \pi_\theta(Mahal|s_1)$

$A^{\pi_\theta}(\mathrm{s}_3, \mathrm{in}) \log \pi_\theta(in|s_3)$

$A^{\pi_\theta}(\mathrm{s}_5, \mathrm{eot}) \log \pi_\theta(eot|s_5)$

$A^{\pi_\theta}(\mathrm{s}_0, \mathrm{Taj}) \log \pi_\theta(Taj|s_0)$

$A^{\pi_\theta}(\mathrm{s}_2, \mathrm{is}) \log \pi_\theta(is|s_2)$

$A^{\pi_\theta}(\mathrm{s}_4, \mathrm{Agra}) \log \pi_\theta(Agra|s_4)$

LLM Policy $\pi_\theta(a|s_t)$

User: Where is Taj Mahal? Assistant:	Taj	Mahal	is	in	Agra

Doesn't matter what gets generated in the future. The "reward" at token "Taj" is fixed.

Estimating Q and Value functions for an LLM

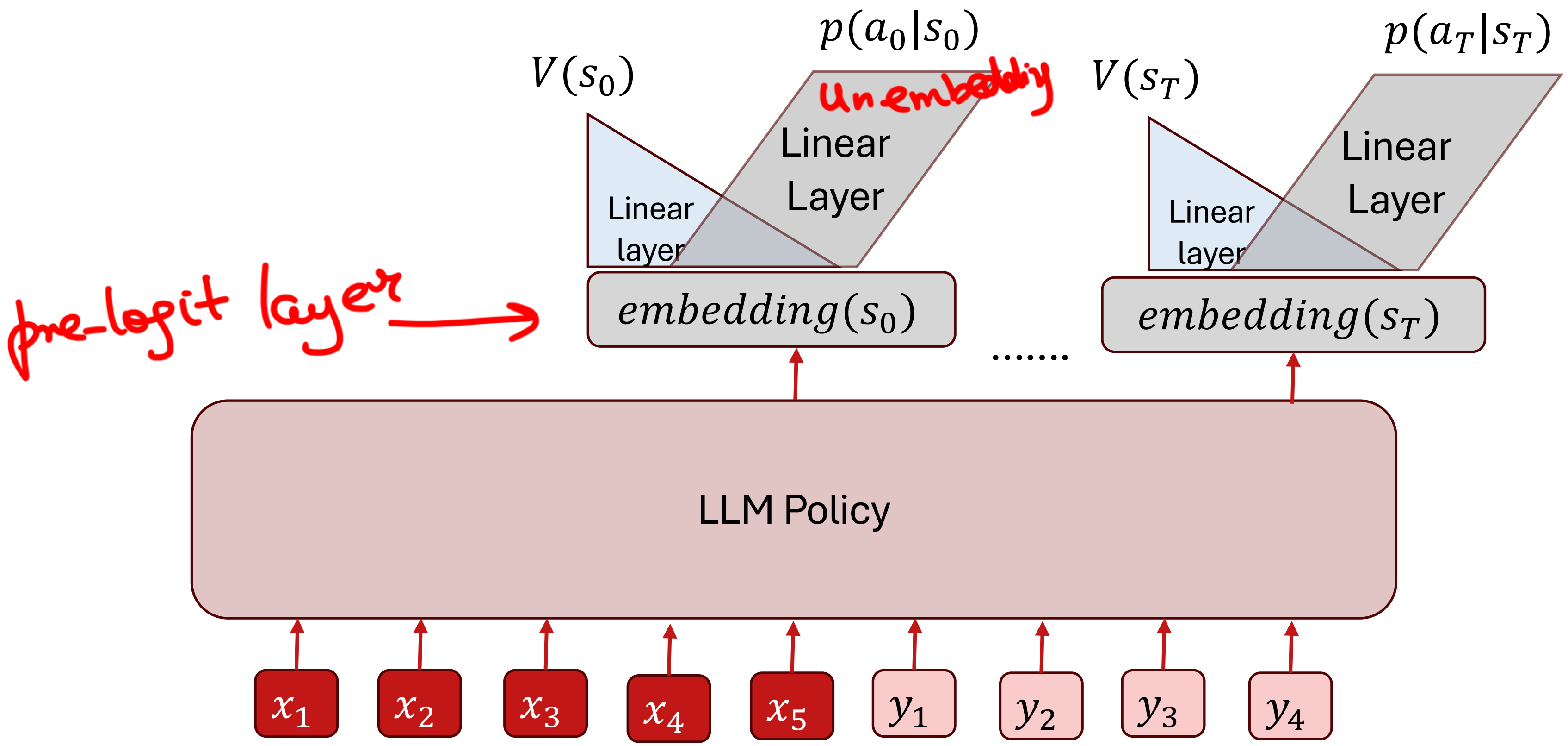
- Given: an input x, sample $y = (a_0, \dots, a_T)$ from the policy $\pi_\theta(y|x)$
- The Q-function for a state-action pair is estimated using a single sample

$$Q(s_t, a_t) = \underbrace{r(s_t, a_t) + r(s_{t+1}, a_{t+1}) \dots + r(s_T, a_T)}$$

- The value function is estimated by training a neural network
  - The cumulative discounted reward for each time-step is computed.
  - MSE between predicted value and the cumulative discounted reward is minimized.

$$\frac{1}{T} \sum_{t=0}^{T} \left(V_\phi(s_t) - [r(s_t, a_t) + r(s_{t+1}, a_{t+1}) \dots] \right)^2.$$

The Value function network V_ϕ



The Actor-Critic Family

- The actor is the policy network that chooses actions given states.
  - It is parameterized as π_θ
- The critic is the value function approximator V_ϕ
- The actor is trained using the following gradient estimate.

$$E[\nabla_\theta \log \pi_\theta(a_t|s_t)(R_t - V_\phi(s_t))]$$

- The critic is trained to minimize the MSE

$$E[(R_t - V_\phi(s_t)^2]$$

- This family of algorithms includes PPO.