

DIRECT PREFERENCE OPTIMIZATION: YOUR LANGUAGE MODEL IS SECRETLY A REWARD MODEL

Nguyen Dinh Huan



AGENDA



01

INTRODUCTION TO
REINFORCEMENT LEARNING

02

PROXIMAL POLICY
OPTIMIZATION (PPO)

03

REINFORCEMENT
LEARNING FROM HUMAN
FEEDBACK (RLHF)

04

DIRECT PREFERENCE
OPTIMIZATION (DPO)

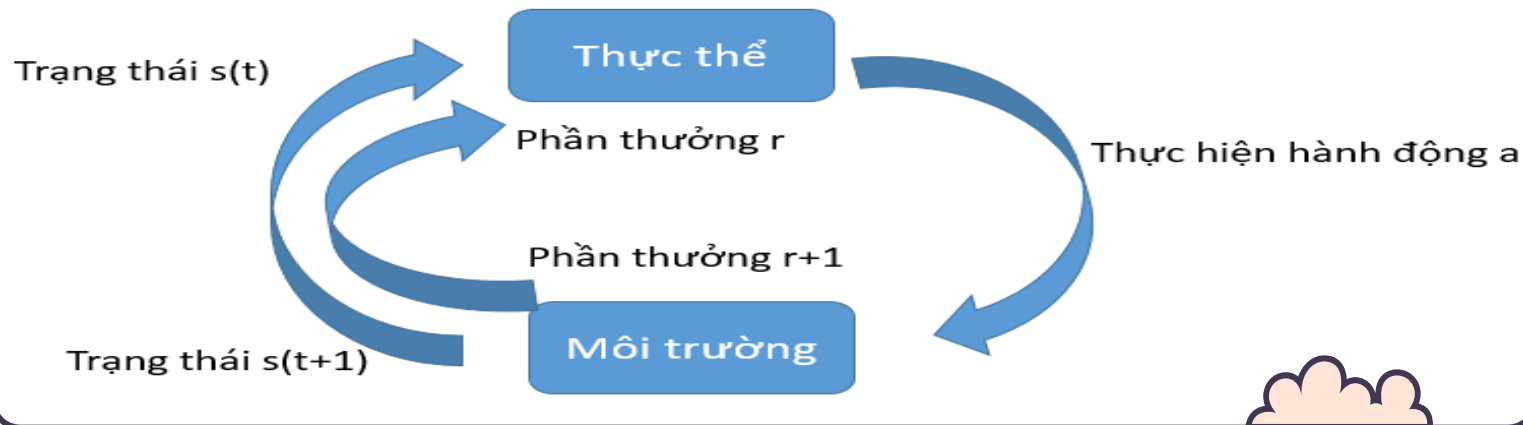




01

INTRODUCTION TO REINFORCEMENT LEARNING

REINFORCEMENT LEARNING



- **Học tăng cường (Reinforcement learning - RL)** là một nhánh của **học máy (Machine learning - ML)**, nghiên cứu cách thức một **thực thể (Agent)** trong một **môi trường (Environment)** đang ở một **trạng thái (State)** thực hiện một **hành động (Action)** để tối ưu hóa một **phần thưởng (Reward)** chung.
- Học tăng cường là một phương pháp phổ biến để giải các **bài toán quyết định Markov (Markov Decision Process - MDP)**.

REINFORCEMENT LEARNING



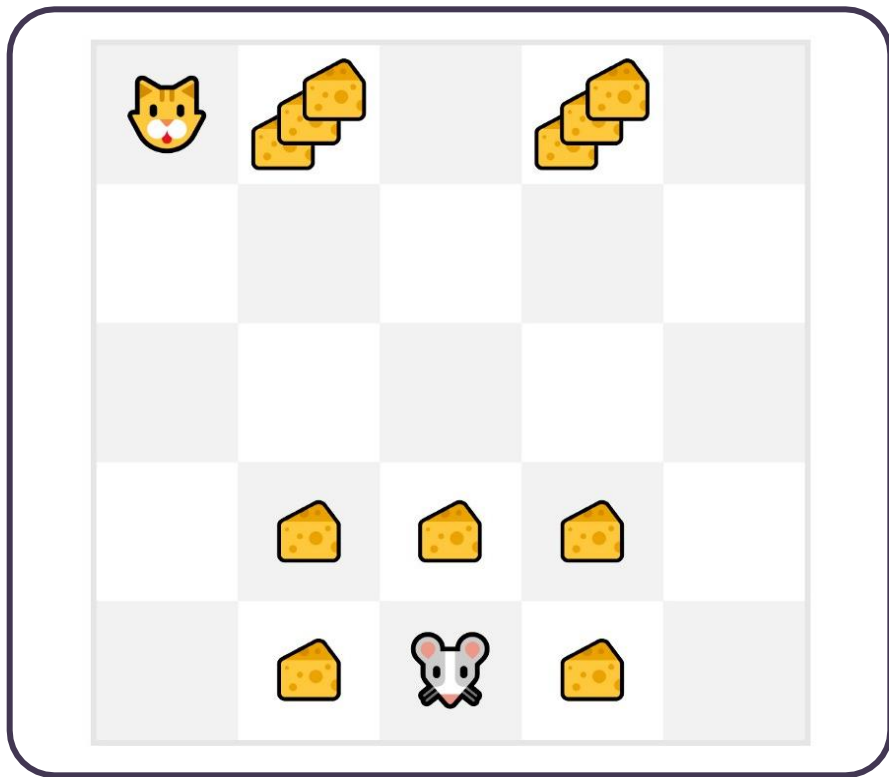
.....



Maximize total future reward

Mục tiêu của RL là để thực hiện một chuỗi các hành động để sao cho tổng phần thưởng nhận lại là lớn nhất.

REINFORCEMENT LEARNING



MARKOV DECISION PROCESS (MDP)



Tính chất của Markov:

- **Phần thưởng và trạng thái của tương lai chỉ phụ thuộc vào hành động và trạng thái ở hiện tại** chứ không phải tất cả hành động và trạng thái ở quá khứ.

$$P(\text{future} \mid \text{present, past}) = P(\text{future} \mid \text{present, ~~past~~})$$

Markov property →



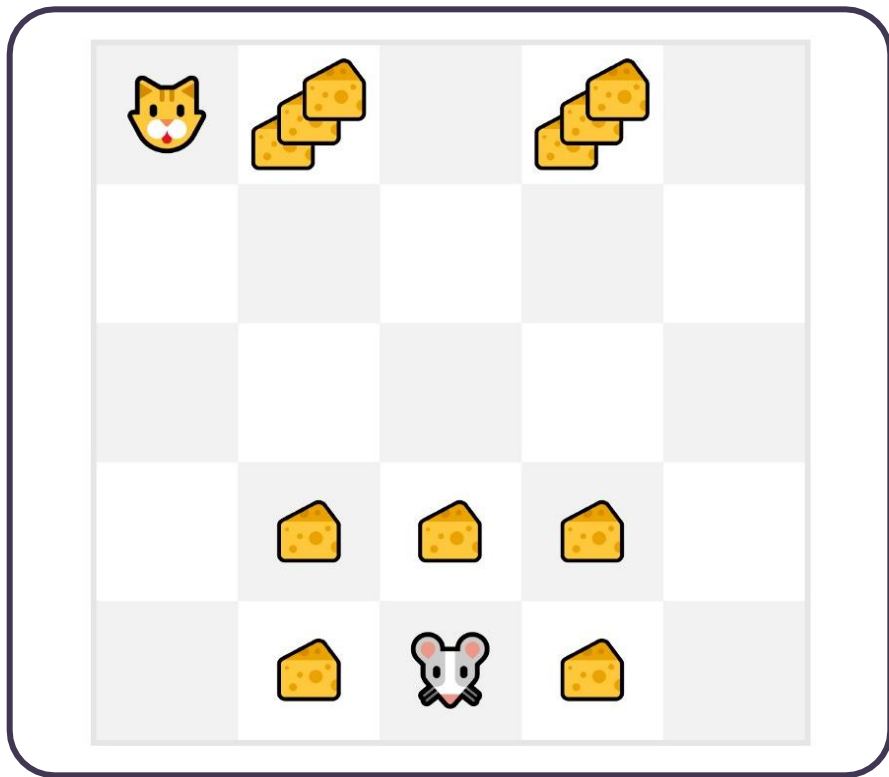
MARKOV PROCESS HAY MARKOV CHAIN



- **Markov Process** : là một chuỗi các trạng thái ngẫu nhiên tuân theo tính chất của Markov (hay còn gọi là Memoryless random process)
- **Markov Process** bao gồm các thành phần như sau:
 - \mathcal{S} là một tập (hữu hạn) **các trạng thái** ($s \in \mathcal{S}$)
 - \mathcal{P} là **phép chuyển đổi trạng** thái từ s sang s' $p(s_{t+1} = s' | s_t = s)$
- Có thể thấy là ở đây chưa có thành phần về phần thưởng và hành động.
- Nếu mà có N hữu hạn các trạng thái thì phép chuyển đổi P có thể được biểu diễn dưới dạng 1 ma trận.

$$P = \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix}$$

MARKOV PROCESS HAY MARKOV CHAIN



MARKOV REWARD PROCESS (MRP)

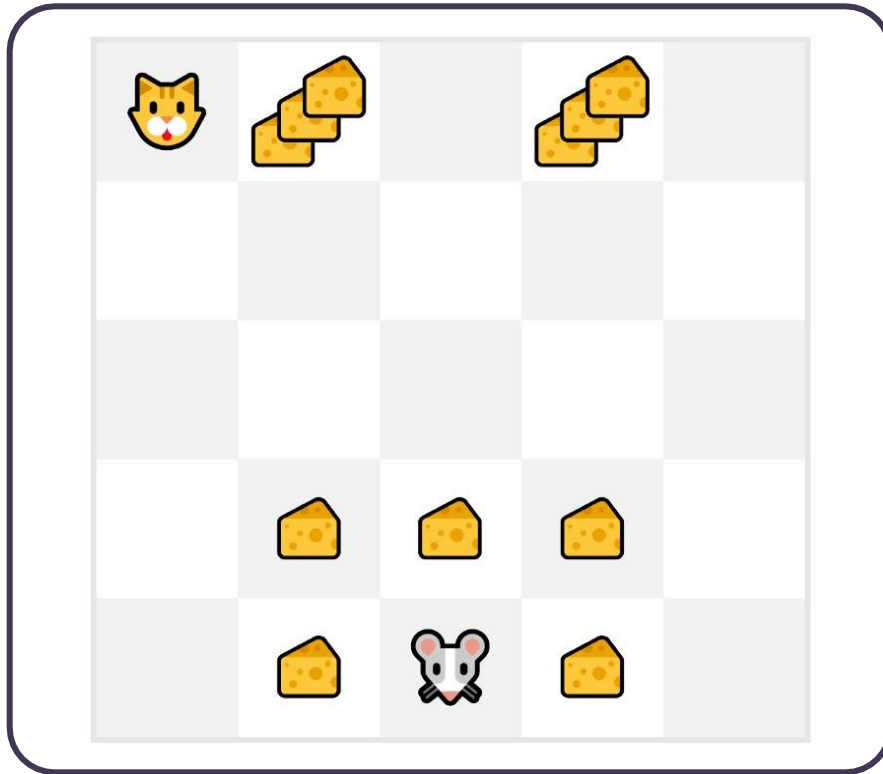


- **Markov Reward Process:** Là Markov Proces mà có thêm phần thưởng.
- **Markov Reward Process** bao gồm:
 - \mathcal{S} là một tập (hữu hạn) các **trạng thái** ($s \in \mathcal{S}$)
 - \mathcal{P} là **phép chuyển đổi trạng thái** từ s sang s' $p(s_{t+1} = s' | s_t = s)$
 - \mathcal{R} là **hàm phần thưởng** $\mathcal{R}(s_t = s) = \mathbb{E}[r_t | s_t = s]$
 - **Khấu hao** $\gamma \in [0,1]$
- Có thể thấy rằng ở đây vẫn chưa có định nghĩa nào về hành động.
- Nếu mà có N hữu hạn các trạng thái thì hàm phần thưởng R có thể được biểu diễn dưới dạng 1 vector

$$R = \begin{bmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{bmatrix}$$



MARKOV REWARD PROCESS (MRP)



MARKOV DECISION PROCESS (MDP)

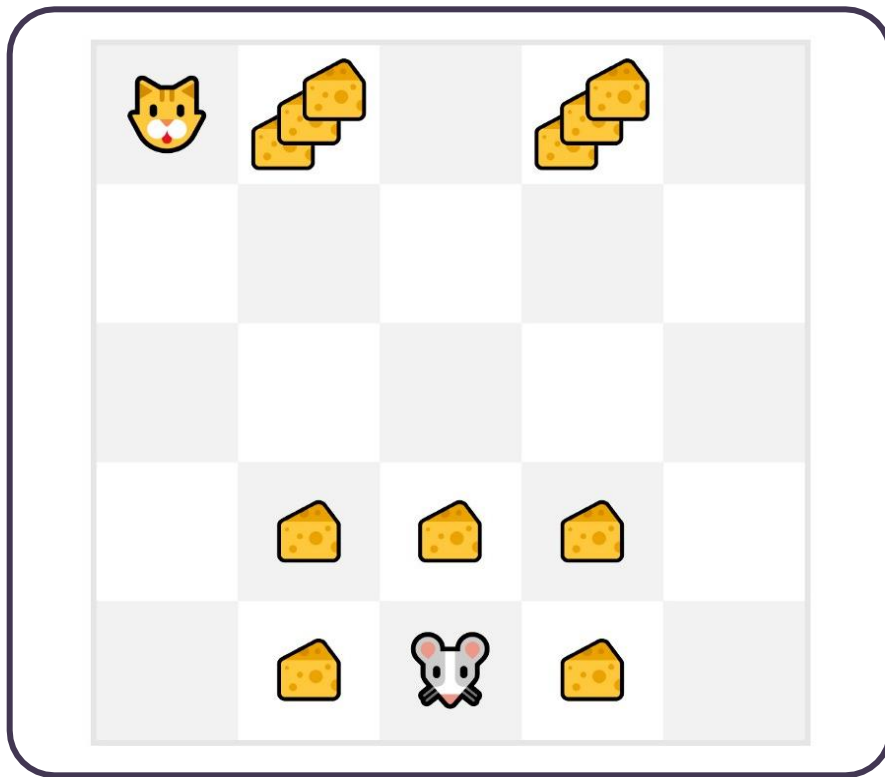


- **Markov Decision Process (MDP)**: là một công cụ toán học dùng để mô hình hóa bài toán decision-making trong một môi trường biến động. Mục đích của MDP là để tìm ra một chiến lược (Policy) tối ưu để có được tổng phần thưởng tối đa.
- **MDP gồm có:**
 - \mathcal{S} là một tập (hữu hạn) các **trạng thái** ($s \in \mathcal{S}$)
 - \mathcal{A} là một tập (hữu hạn) các **hành động** có thể thực hiện ($a \in \mathcal{A}$)
 - \mathcal{P}_{sa} là **phép chuyển đổi trạng thái** từ s sang s' khi thực hiện **hành động** a ($p(s_{t+1} = s' | s_t = s, a_t = a)$)
 - \mathcal{R} là **hàm phần thưởng** $\mathcal{R}(s_t = s) = \mathbb{E}[r_t | s_t = s, a_t = a]$
 - **Khấu hao** $\gamma \in [0, 1]$

$$MDP(\mathcal{S}, \mathcal{A}, \mathcal{P}_{sa}, \mathcal{R}, \gamma)$$



MARKOV DECISION PROCESS (MDP)



RETURN



- **Return (G_t)** là tổng giá trị các phần thưởng khấu hao nhận được tính từ thời điểm t trở đi

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

- γ là **discount factor (khấu hao)** có giá trị trong khoảng $[0,1]$ cái này add vào có 2 mục đích:
 - Vì tổng này là vô hạn nên có discount factor vào để đảm bảo tiến tới 1 giá trị cố định thay vì vô cực.
 - Lấy cảm hứng từ bên khái niệm của kinh tế là Time value of Money (Tính Present Value dựa vào Future Value).

- $PV = \frac{FV}{(1+r)^n} = \gamma^n FV$ đặt $\frac{1}{(1+r)}$ là γ

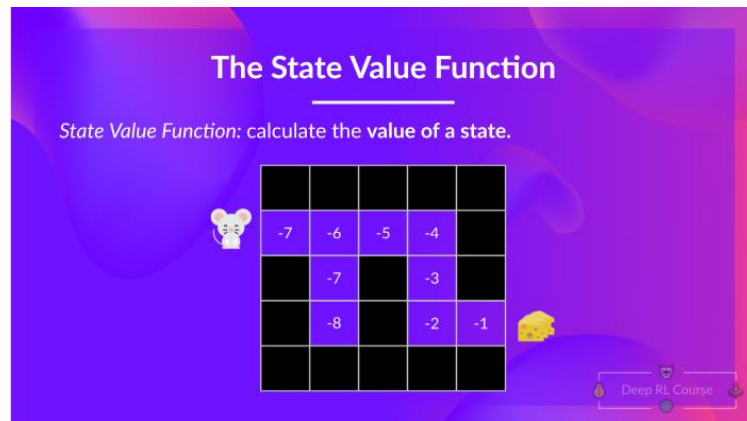
VALUE FUNCTION

- State - value function là cách để dự đoán phần thưởng trong tương lai.
- Thường dùng để so sánh độ tốt/không tốt của một trạng thái (state)
- Từ kết quả của value function ta có thể căn cứ để chọn hành động tương ứng.

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

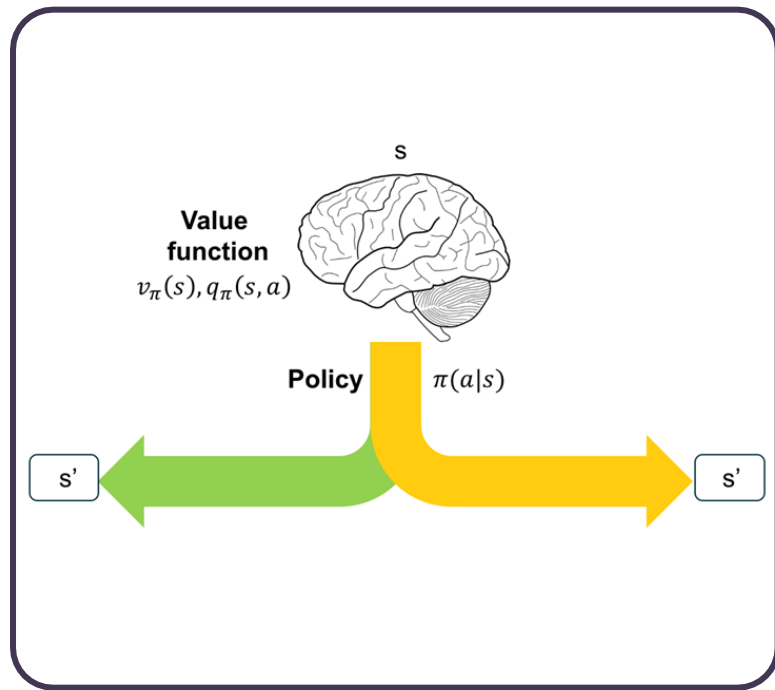
$$= \mathbb{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots | S_t = s]$$

$$= \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s \rightarrow s'} v(s')$$



POLICY

- **Policy** là một mô hình tượng trưng cho hành vi của agent.
- Nó là một hàm ánh xạ từ trạng thái của môi trường sang hành động của agent.
- Policy là một phân phối xác suất của tất cả các hành động có thể thực hiện được khi ở một trạng thái nhất định.
- Policy trong Markov Decision Process chỉ phụ thuộc vào trạng thái hiện tại
$$\pi(a|s) = P[A_t = a | S_t = s]$$



VALUE FUNCTION WHEN HAVING POLICY

- State - value function trong MDP khi có 1 Policy là:

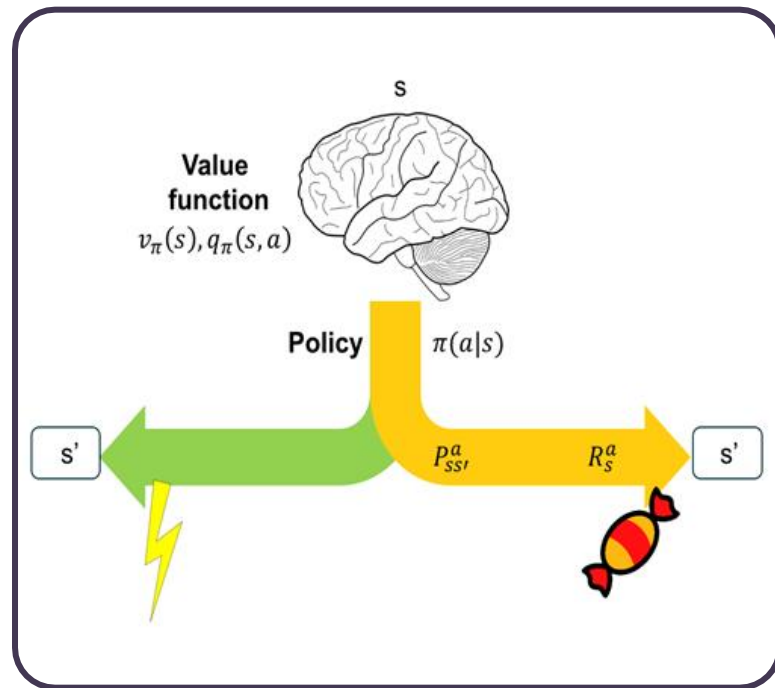
$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s \rightarrow s'}^a v(s')$$

- Khi có policy action a sẽ được quyết định dựa theo policy:

$$a = \pi(s)$$

- Action - value function:

$$q(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a) v_{\pi}(s')$$

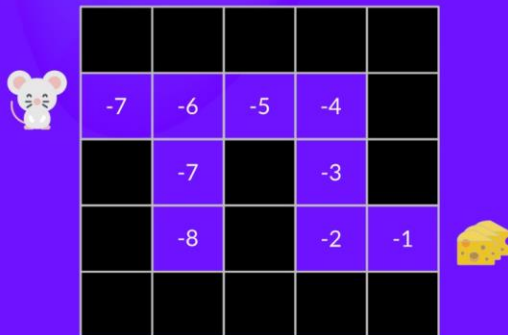


VALUE FUNCTION WHEN HAVING POLICY

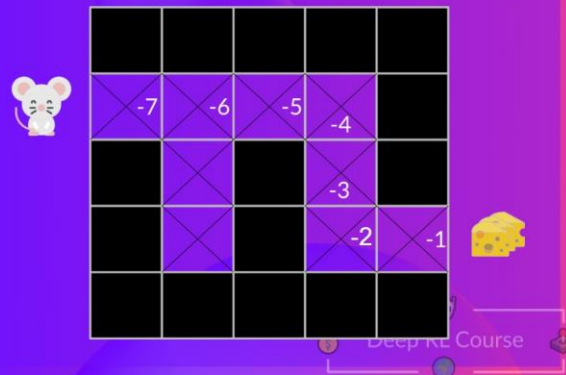


Two types of Value-Based Methods

State Value Function:
calculate the **value of a state**.

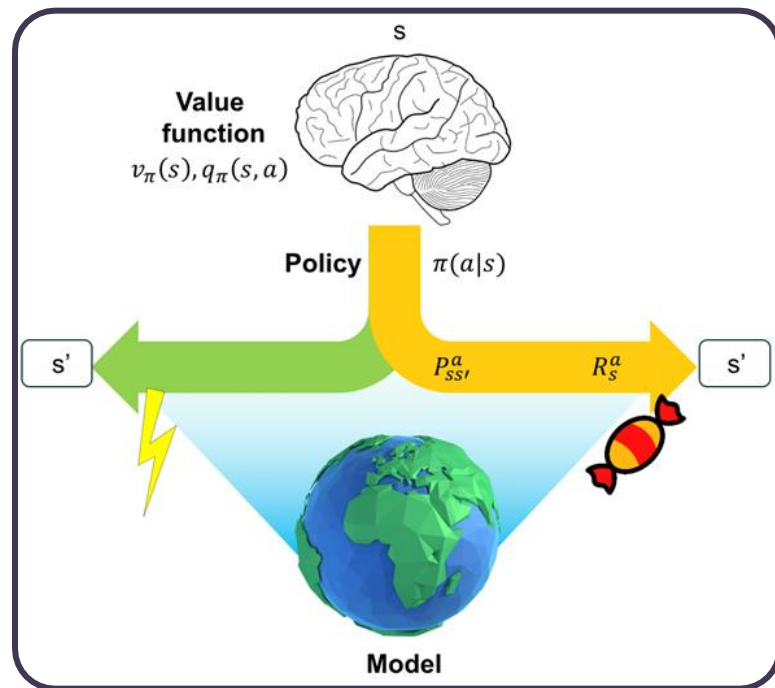


Action Value Function:
calculate the **value of state-action pair**.

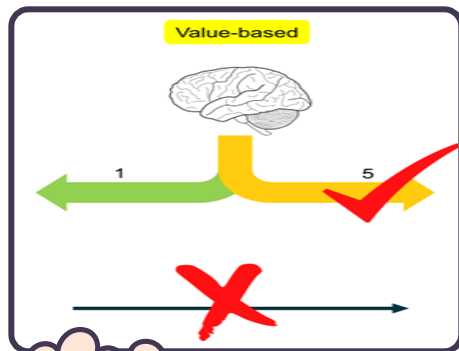


MODEL

- Model là 1 mô hình cố gắng giả lập lại những hành vi của môi trường.
- Model sẽ học 2 phần của MDP là $\mathcal{P}_{ss'}^a$ (Transition probability) và \mathcal{R}_s^a (Reward function).
 - $\mathcal{P}_{ss'}^a$ dự đoán trạng thái kế tiếp của môi trường.
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$
 - \mathcal{R}_s^a dự đoán phần thưởng (tức thời) kế tiếp.
$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} = s' | S_t = s, A_t = a]$$

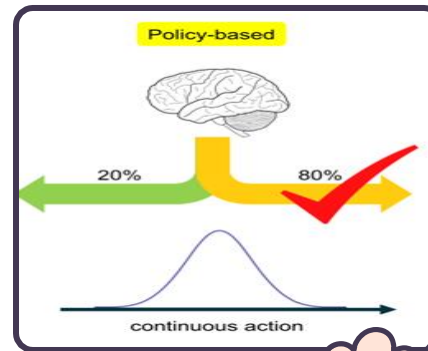


CATEGORIZING RL AGENTS



VALUE BASED

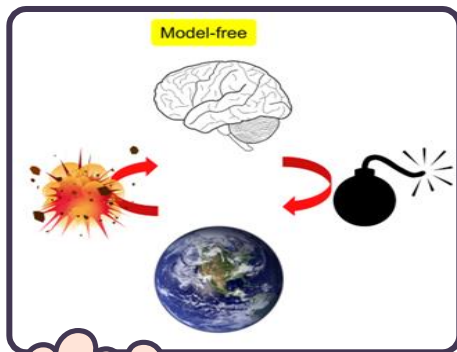
- No Policy (Implicit)
- Value Function



POLICY BASED

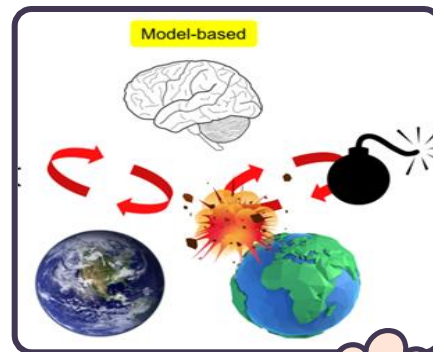
- Policy
- No Value Function

CATEGORIZING RL AGENTS



MODEL FREE

- Policy and/or Value Function
- No Model



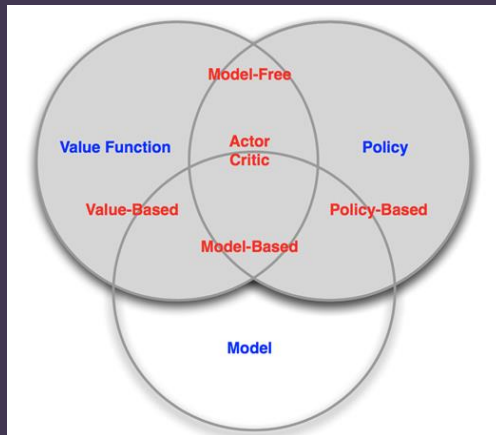
MODEL BASED

- Policy and/or Value Function
- Model

CATEGORIZING RL AGENTS



ACTOR-CRITIC



- ☐ Policy
- ☐ Value Function

FIND OPTIMAL POLICY THROUGH VALUE FUNCTION

- Nếu Policy π_* có giá trị lớn nhất tại mọi trạng thái s so với các policy khác, ta gọi π_* là chiến lược tối ưu.
- Tương tự ta có state-value function, action-value function tối ưu khi sử dụng policy tối ưu.

$$v_*(s) = v_{\pi_*}(s) \geq v_{\pi}(s), \forall s, \forall \pi = \max_{\pi} v_{\pi}(s)$$

$$q_*(s, a) = q_{\pi_*}(s, a) \geq q_{\pi}(s, a), \forall s, \forall a, \forall \pi = \max_{\pi} q_{\pi}(s, a)$$

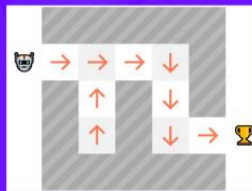
- Khi đó Policy tối ưu sẽ được tính bằng cách:

$$\pi_*(s) = \operatorname{argmax} v_{\pi}(s)$$

$$\pi_*(s) = \operatorname{argmax} q_{\pi}(s, a)$$

Two approaches to find optimal policy π^* :

Policy-Based methods: train the agent to learn which **action to take**, given a state.



Value-Based methods: train the agent to learn which state is **more valuable** and take the action that leads to it.



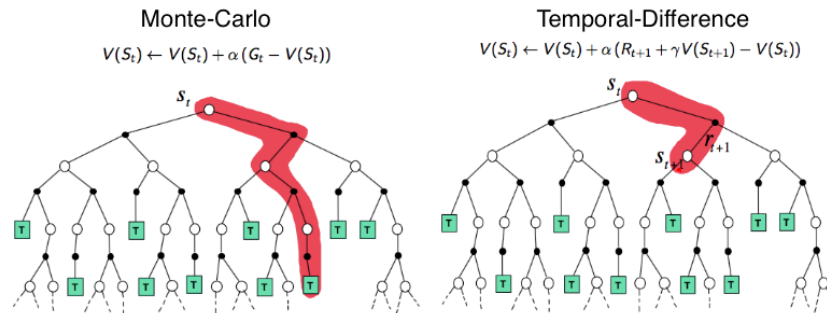
POLICY EVALUATION

- Khi chúng ta có một Policy cố định trong Markov Decision Process thì chúng ta có thể đo lường độ tốt/tệ của Policy đó bằng Markov Reward Process.
- Khi đó Markov Reward Process sẽ bao gồm 4 phần $(\mathcal{S}, \mathcal{R}^\pi, \mathcal{P}^\pi, \gamma)$ với:

$$\mathcal{P}^\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}(s'|s, a)$$

$$\mathcal{R}^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}(s, a)$$

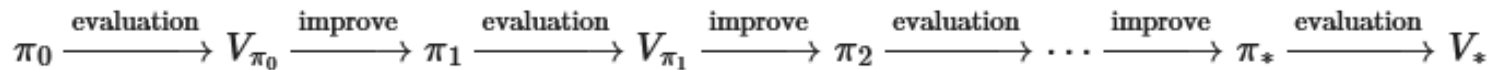
- Khi đó chúng ta có 2 cách để đánh giá 1 cái Policy là tốt hay không:
 - Monte Carlo (MC) Policy Evaluation
 - Temporal Difference (TD) Policy Evaluation



POLICY ITERATION



- Cách tìm Policy tối ưu theo phương pháp vòng lặp cải tiến:
- Gán $i = 0$
- Khởi tạo ngẫu nhiên Policy $\pi_0(s)$ cho tất cả các state s
- While $i \neq 0$ hay $\|\pi_i - \pi_{i-1}\|_1 > 0$ (Kiểm tra xem Policy còn thay đổi hay không)
 - $V^{\pi_i} \leftarrow$ Tính ra giá trị Value bằng cho Policy π_i
 - $\pi_{i+1} \leftarrow$ Cải thiện Policy
 - $i = i+1$



POLICY EVALUATION – MONTE CARLOS

- Trong MC Policy Evaluation chúng ta sẽ chạy giả lập nhiều lần. Mỗi lần chạy giả lập sẽ có một chuỗi các hành động khác nhau được tạo ra nhờ Policy π .
- Chúng ta tính return cho mỗi lần chạy bằng công thức:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$$

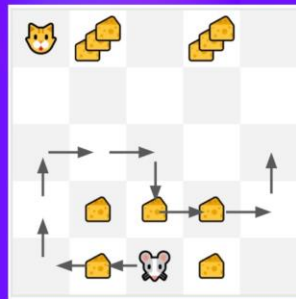
- Sau đó tính Value bằng cách lấy trung bình các return:

$$V^\pi(s) = \text{mean } G_t$$

- Cập nhật lại Value bằng cách chạy vòng lặp, ở vòng lặp thứ i:

$$V^\pi(s) = V^\pi(s) + \alpha(G_{i,t} - V^\pi(s))$$

Monte Carlo Approach:



- Calculate the return G_t .

$$G_t = R_t + \gamma V_{t+1} + \gamma^2 V_{t+2} + \dots$$

$$G_t = 1 + 0 + 0 + 0 + 0 + 0 + 1 + 1 + 0 + 0$$

$$G_t = 3$$

- We can now update $V(S_0)$.

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

$$\text{New } V(S_0) = V(S_0) + \alpha[G_t - V(S_0)]$$

$$\text{New } V(S_0) = 0 + 0.1 * [3 - 0]$$

$$\text{New } V(S_0) = 0.3$$

POLICY EVALUATION – MONTE CARLOS



- Hạn chế:
 - Phải đợi đến khi hết mỗi lần chạy mới có return và mới tính V được.
 - Yêu cầu mỗi lần chạy phải có điểm dừng nếu không sẽ chạy không ngừng.
 - Cần phải chạy nhiều lần (cần nhiều data).
- Ưu điểm:
 - Không cần giả định các state phải tuân theo tính chất của Markov.
 - Không cần mô hình reward và transition của MDP

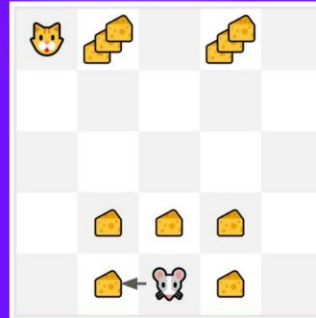
POLICY EVALUATION – TEMPORAL DIFFERENCE (TD(0))

- Temporal Difference muốn tìm một cách khác để có thể ước lượng được G_t mà không cần phải đợi đến hết lần giả lập.

$$V^\pi(s) = V^\pi(s) + \alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s))$$

Ước lượng G_t bằng cách nhìn về phía trước 1 bước

TD Approach:



- We can now update $V(S_0)$:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

New $V(S_0) = 0 + 0.1 * [1 + 1 * 0 - 0]$


The new $V(S_0) = 0.1$


So we just updated our value function for State 0.


Now we continue to interact with this environment with our updated value function.


VALUE FUNCTION APPROXIMATION


$V(s)$ và $Q(s,a)$ thường được biểu diễn dưới dạng vector hay là ma trận, nhưng trong thực tế có những môi trường có rất nhiều trạng thái hoặc hành động.

























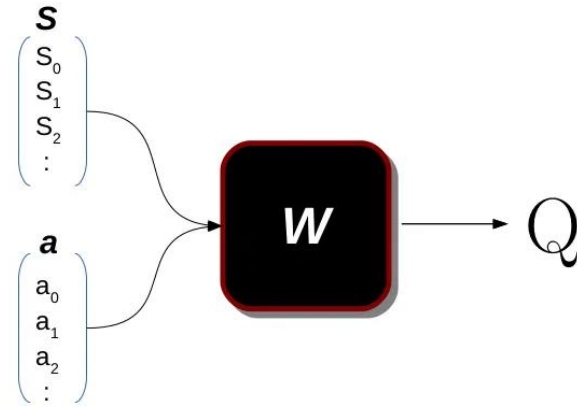
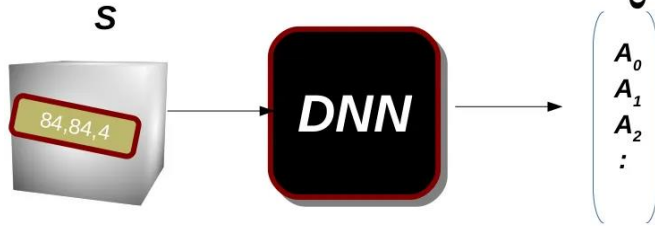
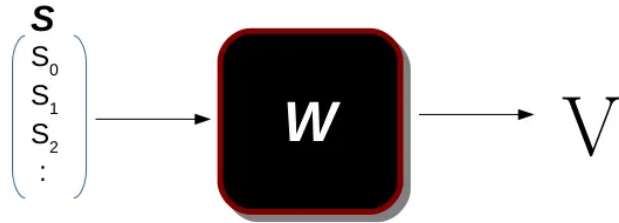


End

Actions :    

Start	0	0	0	0
Nothing / Blank	0	0	0	0
Power	0	0	0	0
Mines	0	0	0	0
END	0	0	0	0

VALUE FUNCTION APPROXIMATION



VALUE FUNCTION APPROXIMATION



- Lợi ích khi áp dụng value function approximation:
 - Khiến agent trở nên **Generalization** (tổng quát hóa) để mô hình có thể học cách đưa ra hành động khi gặp một trạng thái mà agent chưa từng gặp.
 - Giảm bộ nhớ lưu trữ.
 - Giảm tài nguyên tính toán.

VALUE FUNCTION APPROXIMATION – OPTIMIZE WEIGHT

- Tìm w sao cho L2 error giữa giá trị xấp xỉ từ value function $\hat{v}(s, w)$ và giá trị thực tế từ value function $v_\pi(s)$ là nhỏ nhất.

$$J(w) = \mathbb{E}_\pi[(\hat{v}(s, w) - v_\pi(s))^2]$$

- Gradient descent thì w được cập nhật bằng cách:

$$\begin{aligned}w &= w - \alpha \frac{dJ(w)}{dw} = w - \alpha(\hat{v}(s, w) - v_\pi(s)) \frac{d\hat{v}(s, w)}{dw} \\ \frac{dJ(w)}{dw} &= \frac{1}{2} \left(\mathbb{E}_\pi \left[(\hat{v}(s, w) - v_\pi(s))^2 \right] \right)' \\ &= \frac{1}{2} 2 \mathbb{E}_\pi(\hat{v}(s, w) - v_\pi(s)) \frac{d(\hat{v}(s, w) - v_\pi(s))}{dw} \\ &= (\hat{v}(s, w) - v_\pi(s)) \frac{d\hat{v}(s, w)}{dw}\end{aligned}$$

VALUE FUNCTION APPROXIMATION – OPTIMIZE WEIGHT

- Gọi $\phi(s)$ là 1 linear layer hay neural network dùng để encode trạng thái thành embedding vector của trạng thái hay có thể gọi là feature vector:

$$\phi(s) = \begin{pmatrix} \phi_1(s) \\ \vdots \\ \phi_n(s) \end{pmatrix}$$

- Nếu biểu diễn value function dưới dạng 1 linear layer thì:

$$\hat{v}(s, w) = \phi(s)^T w$$

- Trong thực tế thì ta sẽ thay thế $v_\pi(s)$ bằng return G_t . Khi đó w được cập nhật bằng cách:

$$w = w - \alpha \frac{dJ(w)}{dw} = w - \alpha (\phi(s)^T w - G_t) \frac{d\phi(s)^T w}{dw} = w - \alpha [\phi(s)^T w - G_t] \phi(s)$$

Step-size

Prediction error

Feature value



02

PROXIMAL POLICY OPTIMIZATION (PPO)

- **PPO:** là một phương pháp học Policy để cải thiện quá trình học tập của agent một cách ổn định bằng cách **tránh cập nhật những thay đổi lớn đối với Policy**.
- Lý do nên tránh cập nhật những thay đổi lớn cho Policy:
 - Khi cập nhật những thay đổi nhỏ với Policy sẽ dễ đi đến kết quả tối ưu (Optimal solution).
 - Thay đổi Policy quá khác biệt với Policy cũ có thể gây ra kết quả giống như “rơi xuống núi” và sẽ tốn rất nhiều thời gian hoặc không bao giờ tìm được Policy tốt (leo lên lại núi)



ADVANTAGE ACTOR-CRITIC (A2C)

- **Actor – Critic** có thể hiểu ở trường hợp đơn giản nhất là có 2 model 1 model đóng vai là actor và 1 model đóng vai critic.
- **Actor** là model dùng để **học ra policy**. **Critic** là 1 **baseline** đưa ra các **pseudo groundtruth**
- Trong quá trình học:
 - **Critic** sẽ cập nhật action-value function để đưa ra value tốt hơn (value based).
 - **Actor** sẽ cập nhật policy dựa vào hướng dẫn của critic (policy based).



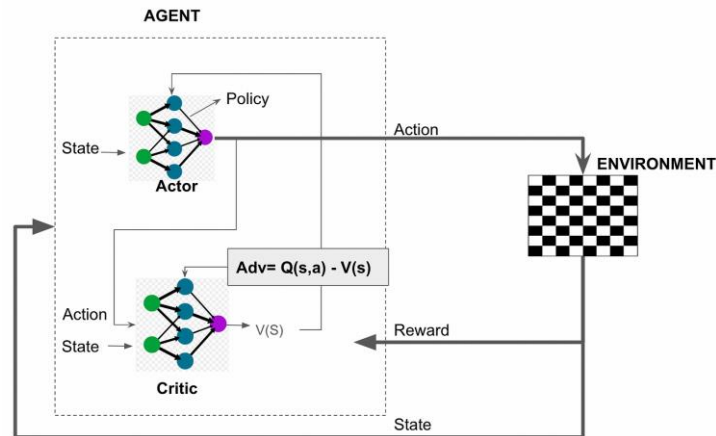
ADVANTAGE ACTOR-CRITIC (A2C)

- Advantage function A là cách critic dùng để đánh giá mối tương quan giữa hành động mà policy chọn thực hiện đối với hành động khác ở trong cùng 1 trạng thái.

$$A = Q(s, a) - V(s)$$

- Nếu $A > 0$ nghĩa là hành động mà actor chọn thực hiện cho kết quả tốt hơn những hành động khác.

$$J(\theta) = \mathbb{E}_t[\log \pi_{\theta}(a_t | s_t) A(a_t, s_t)]$$



Advantage Function

$$A(s, a) = \underbrace{Q(s, a)}_{\text{q value for action a in state s}} - \underbrace{V(s)}_{\text{average value of that state}}$$

q value for action a
in state s

average
value
of that
state

GENERALIZED ADVANTAGE ESTIMATION (GAE)



- $Q(s,a)$ có thể được ước lượng bằng nhiều cách như sau:

$$Q^{\pi,\gamma}(s, u) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \mid s_0 = s, u_0 = u] \quad (\text{Monte Carlo})$$

$$= \mathbb{E}[r_0 + \gamma V^{\pi}(s_1) \mid s_0 = s, u_0 = u] \quad (1\text{-step TD})$$

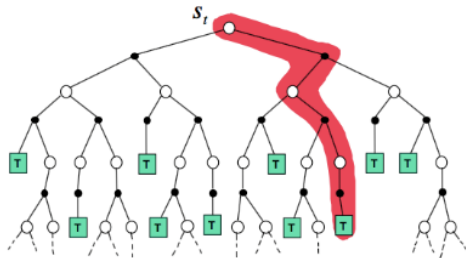
$$= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^{\pi}(s_2) \mid s_0 = s, u_0 = u] \quad (2\text{-step TD})$$

$$= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V^{\pi}(s_3) \mid s_0 = s, u_0 = u]$$

$$= \dots$$

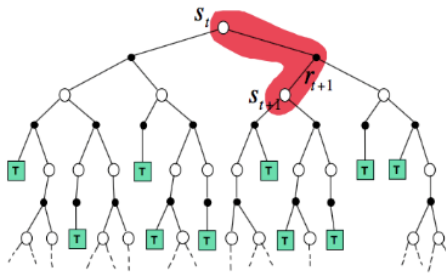
Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$



Temporal-Difference

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



GENERALIZED ADVANTAGE ESTIMATION (GAE)



- Generalized advantage estimation là 1 cách để ước lượng $Q(s,a)$ bằng cách kết hợp giữa phương pháp TD và MC:

$$\hat{A}_t^{(1)} := \delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1})$$

$$\hat{A}_t^{(2)} := \delta_t^V + \gamma \delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$$

$$\hat{A}_t^{(3)} := \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3})$$

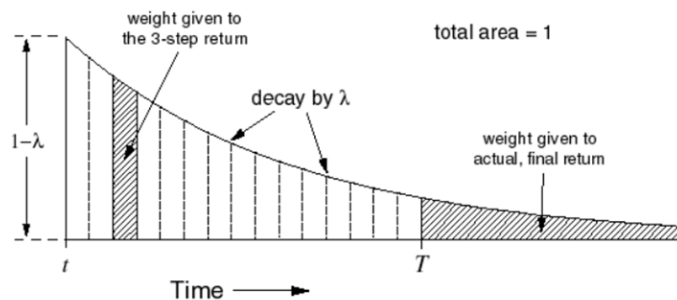
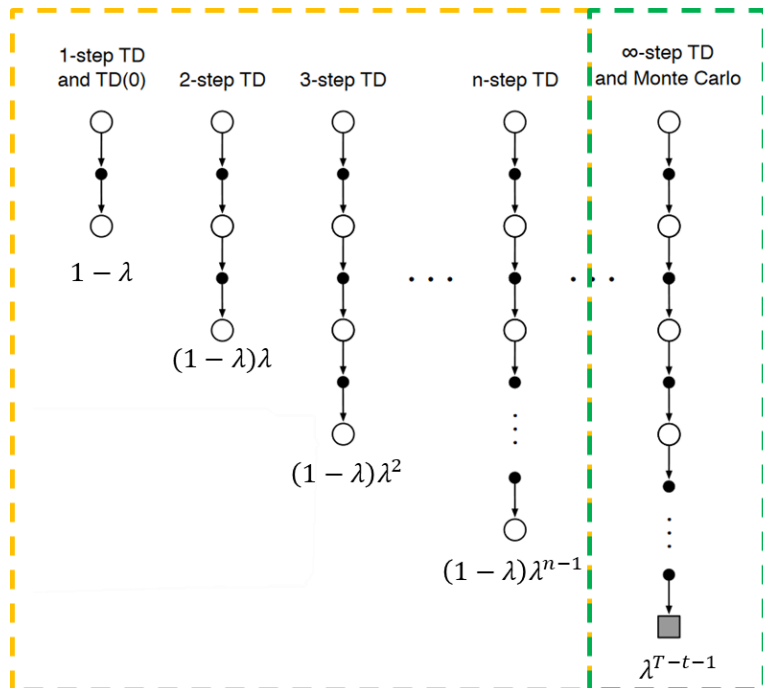
$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})$$

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} := (1 - \lambda) \left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots \right)$$

$$\text{GAE}(\gamma, 0) : \hat{A}_t := \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\text{GAE}(\gamma, 1) : \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t)$$

GENERALIZED ADVANTAGE ESTIMATION (GAE)

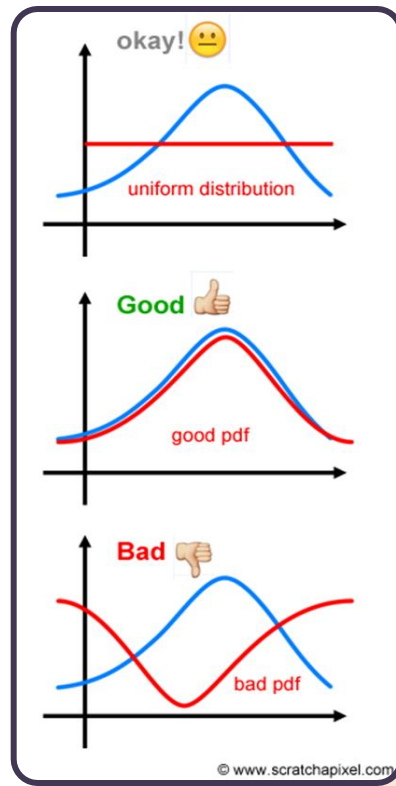


IMPORTANCE SAMPLING

- Importance Sampling là một kỹ thuật ước tính giá trị kỳ vọng của $f(x)$ trong đó x có phân phối dữ liệu p . Tuy nhiên, Thay vì lấy mẫu từ p , chúng ta sẽ tính kết quả từ việc lấy mẫu q .

$$E_p[f(x)] = E_q\left(\frac{f(X)p(X)}{q(X)}\right)$$

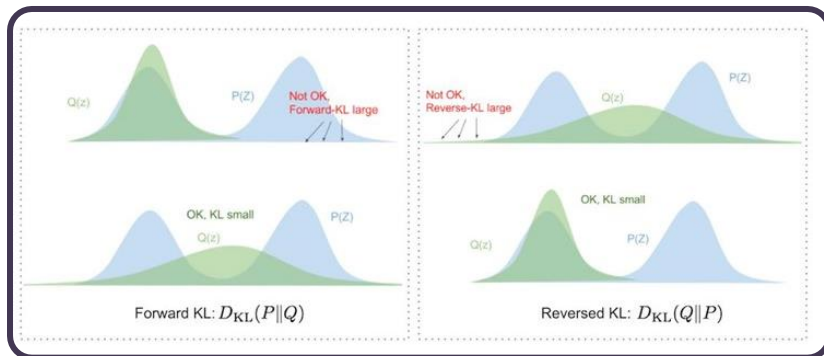
- Để có thể việc ước lượng được chính xác thì phân phối p và q phải gần giống nhau có nghĩa là $\frac{p(X)}{q(X)}$ phải không được cách biệt nhau quá lớn. Do đó tốt nhất thì $\frac{p(X)}{q(X)}$ chỉ nên nằm trong khoảng nào đó 0 đến 1.



KL DIVERGENCE

- Trong thống kê và lý thuyết thông tin, độ đo Kullback–Leibler divergence (còn hay gọi là Entropy tương đối, viết tắt KL divergence) là một độ đo mức độ lệch của một phân bố đối với phân bố được chỉ định.
- Nói một cách đơn giản, KL divergence là độ đo sự khác nhau giữa hai phân bố xác suất.

$$D_{KL}(p||q) = - \sum_{i=1}^n p_i \log_b \frac{q_i}{p_i}$$



POLICY GRADIENT



The Policy Gradient (one trajectory)

$$\nabla_{\theta} J(\theta) \approx \hat{g} = \sum_{t=0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau)$$

Estimation of
the gradient
(given we use
only one
trajectory to
estimate the
gradient)

Probability of the agent to
select action a_t from state s_t
given our policy

Cumulative
return

Direction of the steepest
increase
of the (log) probability of
selecting action a_t from
state s_t

PUT ALL TOGETHER



$$L^{VF}(\theta) = \widehat{\mathbb{E}}_t[(V_\theta(s_t) - G_t)^2]$$

A2C

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Important sampling

$$L^{PG(CLIP)}(\theta) = \widehat{\mathbb{E}}_t[\min(r_t(\theta)\widehat{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\widehat{A}_t)]$$

Or

$$L^{PG(KLPEN)}(\theta) = \widehat{\mathbb{E}}_t[r_t(\theta)\widehat{A}_t - \beta KL(\pi_\theta(\cdot|s_t), \pi_{\theta_{old}}(\cdot|s_t))]$$

$$L(\theta) = L^{PG} + c_1 L^{VF}$$

Vì chúng ta lấy important sampling nên chúng ta phải đảm bảo 2 phân phối không nên quá khác biệt

PUT ALL TOGETHER



Algorithm 1 PPO, Actor-Critic Style

for iteration=1, 2, ... **do**

for actor=1, 2, ..., N **do**

 Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps

 Compute advantage estimates $\hat{A}_1, \dots, \hat{A}_T$

) interacting w/ the environment &
generating sequences for calculating advantage function

end for

 Optimize surrogate L wrt θ , with K epochs and minibatch size $M \leq NT$

$\theta_{\text{old}} \leftarrow \theta$

) Run SGD on $L^{\text{sur}}(\theta)$
every so often

end for

03

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

OVERVIEW



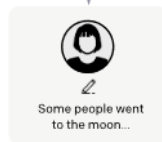
Step 1

**Collect demonstration data,
and train a supervised policy.**

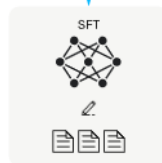
A prompt is
sampled from our
prompt dataset.



A labeler
demonstrates the
desired output
behavior.



This data is used
to fine-tune GPT-3
with supervised
learning.



Step 2

**Collect comparison data,
and train a reward model.**

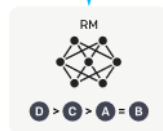
A prompt and
several model
outputs are
sampled.



A labeler
ranks the outputs
from best to worst.



This data is used
to train our
reward model.



Step 3

**Optimize a policy against
the reward model using
reinforcement learning.**

A new prompt
is sampled from
the dataset.



The policy
generates
an output.

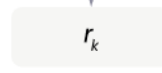


Once upon a time...

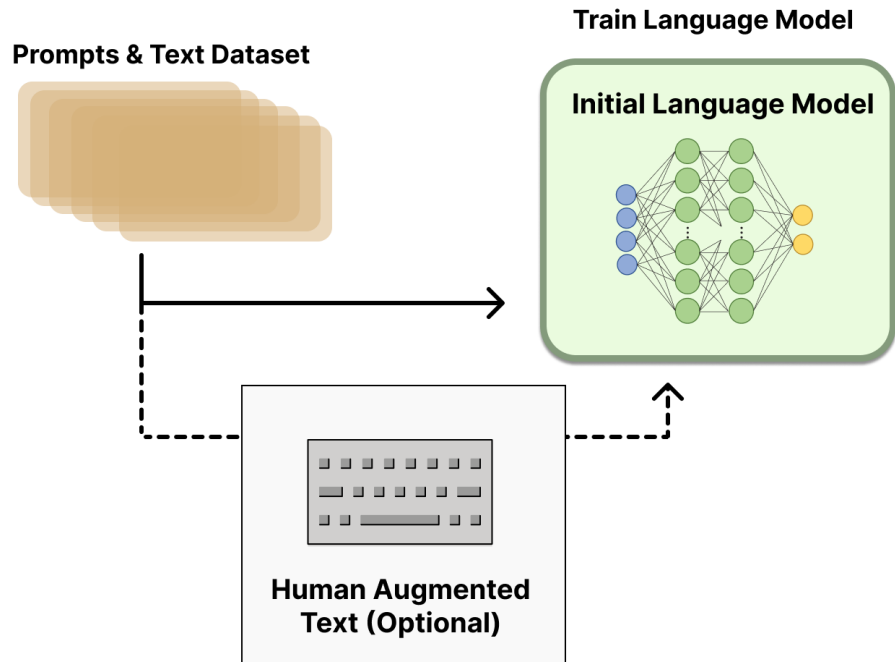
The reward model
calculates a
reward for the
output.



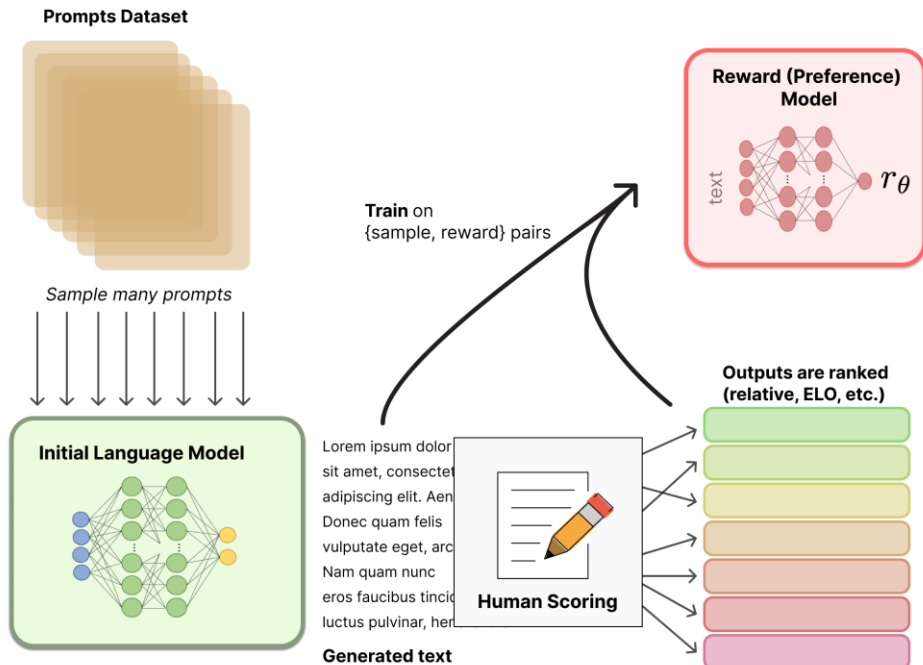
The reward is
used to update
the policy
using PPO.



PRETRAINING LANGUAGE MODELS



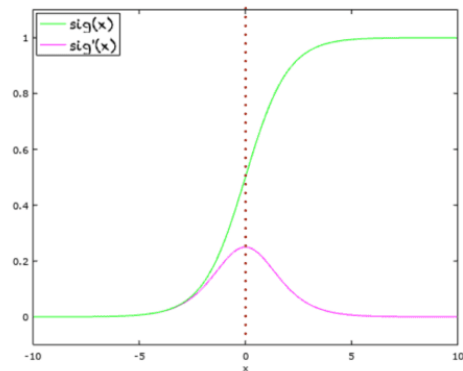
REWARD MODEL TRAINING



REWARD MODEL

- Objective function của reward model xuất phát từ Bradley-Terry model. Đó là một mô hình xác suất dùng để so sánh kết quả trả về của 2 đối tượng. Ở đây ta dùng nó để so sánh độ ưa thích giữa 2 câu sau khi được con người đánh giá.

$$p(y_1 > y_2) = \frac{\exp(r_\theta(x, y_1))}{\exp(r_\theta(x, y_1)) + \exp(r_\theta(x, y_2))}$$
$$p(y_1 > y_2) = \sigma(r_\theta(x, y_1) - r_\theta(x, y_2))$$



Plot of $\sigma(x)$ and its derivative $\sigma'(x)$

Domain: $(-\infty, +\infty)$

Range: $(0, +1)$

$\sigma(0) = 0.5$

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

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

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

REWARD MODEL



- Objective function của reward model xuất phát từ Bradley-Terry model. Đó là một mô hình xác suất dùng để so sánh kết quả trả về của 2 đối tượng. Ở đây ta dùng nó để so sánh độ ưa thích giữa 2 câu sau khi được con người đánh giá.

$$p(y_1 > y_2) = \frac{\exp(r_\theta(x, y_1))}{\exp(r_\theta(x, y_1)) + \exp(r_\theta(x, y_2))}$$
$$p(y_1 > y_2) = \sigma(r_\theta(x, y_1) - r_\theta(x, y_2))$$

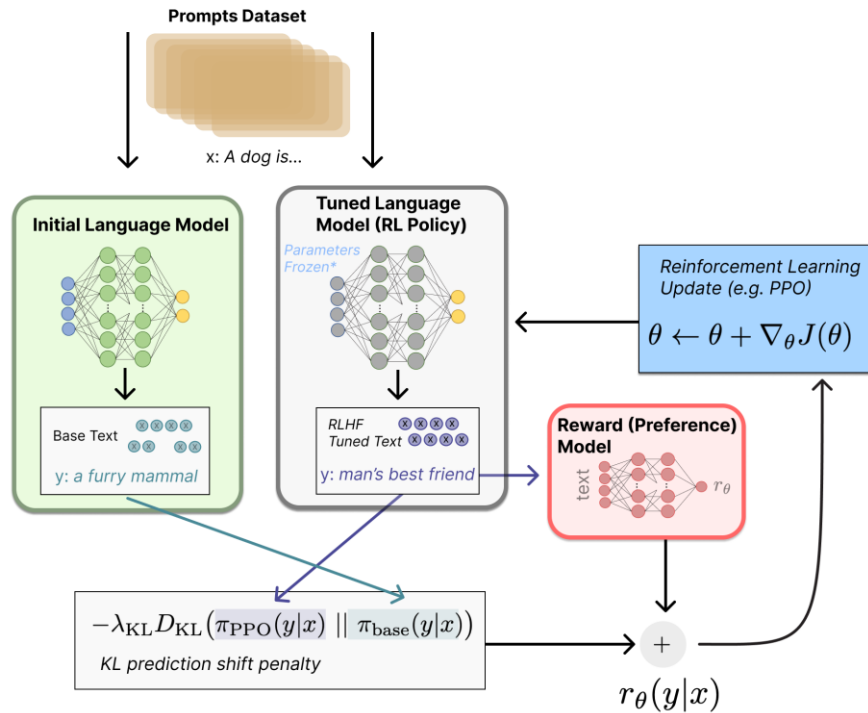
- Loss function:

$$L(r_\theta) = -\log(\sigma(r_\theta(x, y_1) - r_\theta(x, y_2)))$$

- Khi reward trả ra từ reward model dùng cho PPO tác giả add thêm KL Divergence:

$$R(x, y) = r_\theta(x, y) - \beta KL(\pi_\theta^{RL}(y|x) || \pi_{\theta_{old}}^{SFT}(y|x))$$

FINE-TUNING WITH RL – PPO



PPO



```
for epoch, batch in tqdm(enumerate(ppo_trainer.dataloader)):
    if epoch >= config.total_ppo_epochs:
        break

    question_tensors = batch["input_ids"]

    response_tensors = ppo_trainer.generate(
        question_tensors,
        return_prompt=False,
        length_sampler=output_length_sampler,
        **generation_kwargs,
    )
    batch["response"] = tokenizer.batch_decode(response_tensors, skip_special_tokens=True)

    # Compute reward score (using the sentiment analysis pipeline)
    texts = [q + r for q, r in zip(batch["query"], batch["response"])]
    pipe_outputs = sentiment_pipe(texts, **sent_kwargs)
    rewards = [torch.tensor(output[0]["score"] - script_args.reward_baseline) for output in pipe_outputs]

    # Run PPO step
    stats = ppo_trainer.step(question_tensors, response_tensors, rewards)
    ppo_trainer.log_stats(stats, batch, rewards)

    if script_args.save_freq and epoch and epoch % script_args.save_freq == 0:
        ppo_trainer.save_pretrained(script_args.output_dir + f"step_{epoch}")
```



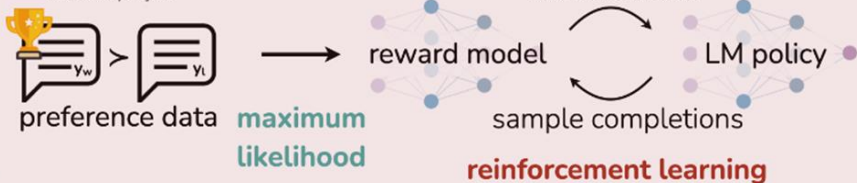
04

DIRECT PREFERENCE OPTIMIZATION (DPO)

OVERVIEW

Reinforcement Learning from Human Feedback (RLHF)

x: "write me a poem about
the history of jazz"



Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"

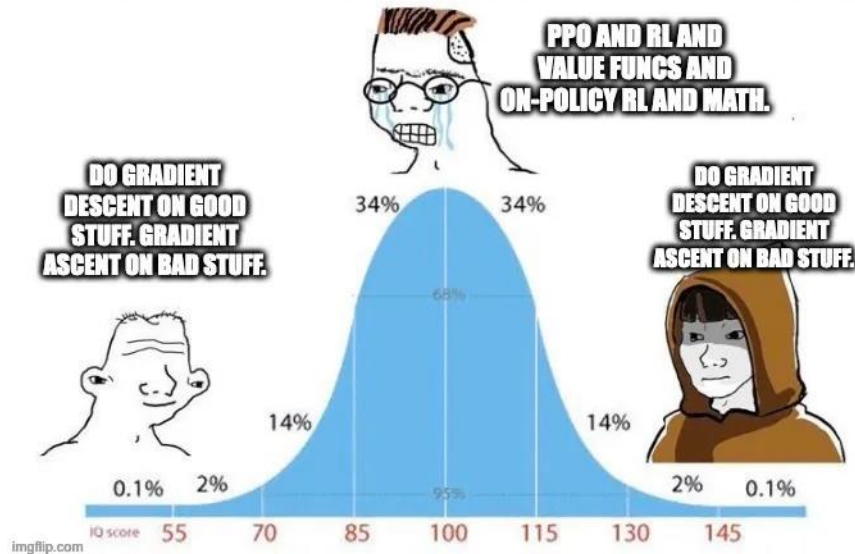


- Đây là một kỹ thuật dùng để huấn luyện language model sao cho model đưa ra câu trả lời đúng theo ý người dùng một cách trực tiếp.
- Trước đây các Language model thường được train theo hướng maximum likelihood estimation (MLE) hoặc reinforcement learning (RL).

OVERVIEW

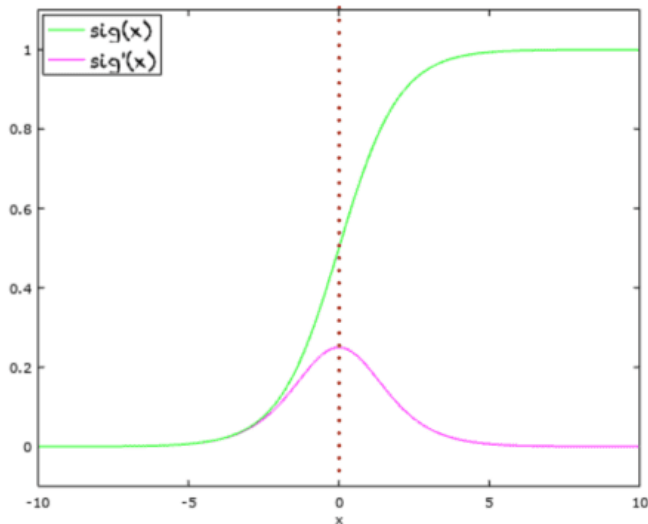


LEARNING FROM HUMAN FEEDBACK



$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\nabla_{\theta} \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right) \right] \quad (21)$$

OVERVIEW



Plot of $\sigma(x)$ and its derivate $\sigma'(x)$

Domain: $(-\infty, +\infty)$

Range: $(0, +1)$

$\sigma(0) = 0.5$

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

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

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\nabla_{\theta} \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right) \right] \quad (21)$$

SECRET BEHIND DPO



- Từ RLHF ta có những công thức sau:

$$p(y_1 > y_2) = \frac{\exp(r_\theta(x, y_1))}{\exp(r_\theta(x, y_1)) + \exp(r_\theta(x, y_2))}$$

$$L(r_{\theta_{RM}}) = -\log(\sigma(r_{\theta_{RM}}(x, y_1) - r_{\theta_{RM}}(x, y_2)))$$

$r_{\theta_{RM}}$ is a reward model

$$\begin{aligned} R(x, y) &= r_{\theta_{RM}}(x, y) - \beta KL(\pi_\theta^{RL}(y|x) || \pi_{\theta_{(old)}}^{SFT}(y|x)) \\ &= r_{\theta_{RM}}(x, y) - \beta \left(\log(\pi_\theta^{RL}(y|x)) - \log(\pi_{\theta_{(old)}}^{SFT}(y|x)) \right) \end{aligned}$$

SECRET BEHIND DPO



$$\begin{aligned} & \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi} [r(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi(y|x) \parallel \pi_{\text{ref}}(y|x)] \\ &= \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \\ &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \\ &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right)} - \log Z(x) \right] \quad (12) \end{aligned}$$

where we have partition function:

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

Note that the partition function is a function of only x and the reference policy π_{ref} , but does not depend on the policy π . We can now define

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



SECRET BEHIND DPO



$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \left[\mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi^*(y|x)} \right] - \log Z(x) \right] = \quad (13)$$

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} [\mathbb{D}_{\text{KL}}(\pi(y|x) \parallel \pi^*(y|x)) - \log Z(x)] \quad (14)$$

Now, since $Z(x)$ does not depend on π , the minimum is achieved by the policy that minimizes the first KL term. Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical. Hence we have the optimal solution:

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right) \quad (15)$$

for all $x \in \mathcal{D}$. This completes the derivation.

SECRET BEHIND DPO



It is straightforward to derive the DPO objective under the Bradley-Terry preference model as we have

$$p^*(y_1 \succ y_2 | x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} \quad (16)$$

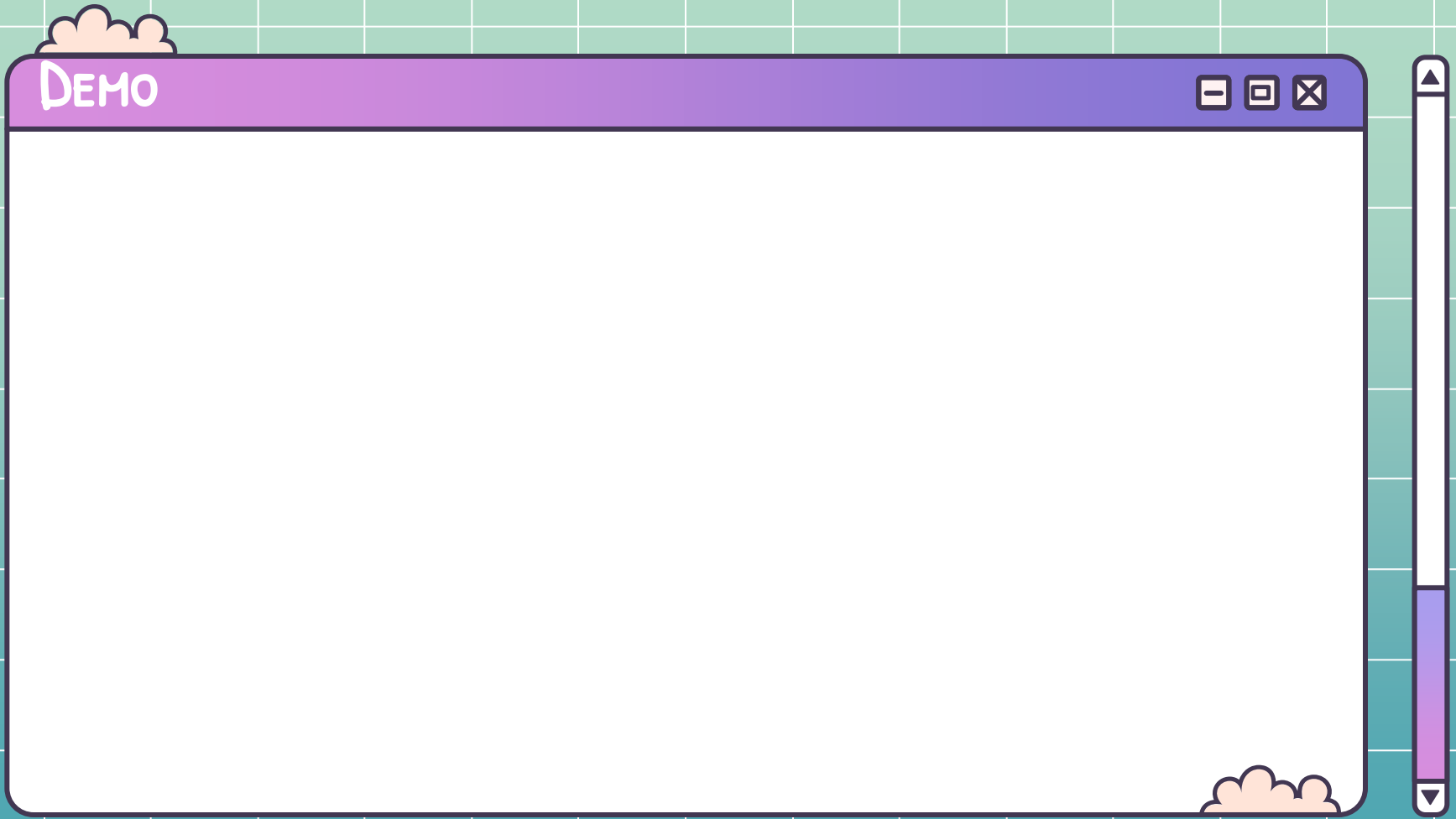
In Section 4 we showed that we can express the (unavailable) ground-truth reward through its corresponding optimal policy:

$$r^*(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x) \quad (17)$$

Substituting Eq. 17 into Eq. 16 we obtain:

$$\begin{aligned} p^*(y_1 \succ y_2 | x) &= \frac{\exp\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta \log Z(x)\right)}{\exp\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta \log Z(x)\right) + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} + \beta \log Z(x)\right)} \\ &= \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} \\ &= \sigma\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right). \end{aligned}$$

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\nabla_{\theta} \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right) \right] \quad (21)$$





THANKS FOR LISTENING

REFERENCES



- <http://www.juyang.co/reinforcement-learning-ii-markov-decision-process-and-rl-agent/>
- <https://www.davidsilver.uk/teaching/>
- https://www.researchgate.net/figure/Classification-of-RL-algorithms-inspired-from-UCL-Course-on-RL-by-David-Silver-41_fig2_346808682
- https://www.researchgate.net/figure/Reinforcement-Learning-Agent-Taxonomy_fig2_327733315
- <https://medium.com/@khalil.hennara.247/value-function-approximation-dqn-43df289a3380>
- <https://tek4.vn/phuong-phap-importance-sampling-hoc-tang-cuong>
- <https://jonathan-hui.medium.com/rl-policy-gradients-explained-advanced-topic-20c2b81a9a8b>
- <https://blog.mlreview.com/making-sense-of-the-bias-variance-trade-off-in-deep-reinforcement-learning-79cf1e83d565>
- <https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/>



REFERENCES



- <https://www.eeweb.com/tools/calculus-derivatives-and-limits-reference-sheet/>
- <https://huggingface.co/blog/deep-rl-ppo>
- <https://arshren.medium.com/unlocking-the-secrets-of-actor-critic-reinforcement-learning-a-beginners-guide-3c5953b13551>
- <https://huggingface.co/blog/deep-rl-a2c>
- https://hugocisneros.com/notes/kullback_leibler_divergence/
- https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf
- <https://arxiv.org/pdf/1707.06347.pdf>
- <https://blog.tylertaewook.com/post/proximal-policy-optimization>
- <https://huggingface.co/learn/deep-rl-course/>
- <https://lilianweng.github.io/posts/2018-02-19-rl-overview/>
- https://web.stanford.edu/class/cme241/lecture_slides/rich_sutton_slides/13-multistep.pdf
- <https://www.interconnects.ai/p/the-dpo-debate>
- <https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/>

