Reinforcement Learning

Dept. of Artificial Intelligence

Chanbin Lee

2024.1.5

Outline

- Basic of RL
- RL for Language Models

Reinforcement Learning

Reinforcement Learning is science and framework of learning to make decisions from interactions. Based on Reward Hypothesis, we assume that all goals can be formalized as the outcome of maximizing a cumulative reward.

At timestep t, agents receive **Observation(O_t)**, scalar **Reward(R_t)** from the environment, and makes **Action(A_t)** to the environment.
Return is the cumulative reward over timesteps:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$$

Policy(π) maps agent state to action, being either deterministic or stochastic.

Exploration and Exploitation

Exploitation: Maximizing the performance based on current knowledge

Exploration: Increasing the knowledge

Simple Example: Multi-Armed Bandit

$\rightarrow$ Single state, optimal action?

Greedy

$$A_t = \underset{a}{\operatorname{argmax}}\, Q_t(a)$$

ϵ**-greedy** : Greedy + Random Exploration

$$\pi_t(a) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{|\mathcal{A}|}, & \text{if } Q_t(a) = \max_b Q_t(b) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{else} \end{cases}$$

UCB: Choose if high estimate or high uncertainty

$$A_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q_t(a) + c\sqrt{\frac{log\ t}{N_t(a)}}$$

Markov Decision Processes

Markov Decision Processes(MDPs) is tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ where

- $\mathcal{S}$ is the set of all possible states
- $\mathcal{A}$ is the set of all possible actions
- $p(s'|s, a)$ is transition probability
- $r(s, a) = \mathbb{E}[R|s, a]$ is expected reward from environment
- γ is the discount factor

All states in MDP should have **Markov Property**, which means for all state S_t,

$$p(r, s|S_t, A_t) = p(r, s|H_t, A_t)$$

state-value function

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s, \pi]$$

action-value function

$$q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a, \pi]$$

Markov Decision Processes

Bellman Expectation Equations

$$v_\pi(s) = \sum_a \pi(a|s)[r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s')]$$

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') q_\pi(s',a')$$

Bellman Optimality Equations

$$v^*(s) = \max_a \, [r(s,a) + \gamma \sum_{s'} p(s'|s,a) v^*(s')]$$

$$q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \, \max_{a'} \, q^*(s',a')$$

In the vector form, the bellman expectation equation is:

$$\boldsymbol{v} = \boldsymbol{r}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v}$$

So if we have the perfect model$(\boldsymbol{r}, \boldsymbol{P})$, this is generally solved as

$$\boldsymbol{v} = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{r}^\pi$$

$\rightarrow$ O$(\mathcal{S}^3)$ solution + can't directly obtain optimal value functions

Dynamic Programming

Dynamic Programming: algorithms solving MDPs, with **perfect model** of the environment.

Policy Evaluation

Bellman Expectation Operator $T^\pi : \mathcal{V} \to \mathcal{V}$:

$$T^\pi f(s) = \sum_a \pi(a|s)[r(s,a) + \gamma \sum_{s'} p(s'|s,a) f(s')]$$

$\to$ γ-contraction, fixed point iteration converges

$$v_{k+1}(s) = T^\pi v_k(s) = \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, \pi]$$

$$\text{as } k \to \infty, v_k \to v_\pi$$

Policy Iteration: Policy Evaluation(until convergence) + Greedy Improvement

$$\pi_{new}(s) = \operatorname*{argmax}_a q(s,a)$$

Value Iteration: Policy Evaluation(1 step) + Greedy Improvement

Bellman Optimality Operator $T^* : \mathcal{V} \to \mathcal{V}$:

$$(T^* f)(s) = \max_a \, [r(s,a) + \gamma \sum_{s'} p(s'|s,a) f(s')]$$

$$v_{k+1}(s) = T^* v_k(s) = \max_a \, \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, A_t = a]$$

$$\text{as } k \to \infty, v_k \to v^*$$

Model-Free Prediction

Unknown MDP $\rightarrow$ sample to estimate, learn without model

Monte Carlo Policy Evaluation(MC)

Estimate the value function.

If we run the episode until it ends, we get a sample of G_t. Update over every timesteps as:

$$v_{n+1}(S_t) = v_n(S_t) + \alpha(G_t - v_n(S_t))$$

$\rightarrow$ But, have to wait until the episode ends & High **variance**

Temporal-Difference Learning(TD)

Instead of using G_t, do **bootstraping**

$$v_{t+1}(S_t) = v_t(S_t) + \alpha_t((R_{t+1} + \gamma v_t(S_{t+1})) - v_t(S_t))$$

TD can learn **online**, during incomplete episodes. But **biased**.

n-step TD: control variance/bias

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n v(S_{t+n})$$

$$v(S_t) \leftarrow v(S_t) + \alpha(G_t^{(n)} - v(S_t))$$

Model-Free Control

Optimize the value function using policy iteration. Since model-free, we need $q(s, a)$.

Model-free prediction + **Greedy in the Limit with Infinite Exploration(GLIE)** policy

$$\forall s, a \lim_{t\to\infty} N_t(s, a) = \infty$$

$$\lim_{t\to\infty} \pi_t(a|s) = I(a = \underset{a'}{\text{argmax}}\ q_t(s, a'))$$

All (s, a) pairs **explored infinitely**, while policy converges to greedy policy w.r.t q_t.

$\rightarrow$ converge to q^*

ex)

$$\epsilon \leftarrow \frac{1}{\#\ \text{episode}}$$

$$\pi \leftarrow \epsilon\text{-greedy}$$

$$q_{t+1}(S_t, A_t) = q_t(S_t, A_t) + \alpha_t(G_t - q_t(S_t, A_t)) \quad \text{MC control}$$

$$q_{t+1}(S_t, A_t) = q_t(S_t, A_t) + \alpha_t((R_{t+1} + \gamma q_t(S_{t+1}, A_{t+1})) - q_t(S_t, A_t)) \quad \text{TD control (SARSA)}$$

Model-Free Control

Off-Policy

Behavior Policy(μ): policy used to take actions

Target Policy(π): policy trying to evaluate and improve

On-Policy Learning: $\mu = \pi$

Off-Policy Learning: $\mu \neq \pi$, can learn optimal policy while exploring

Off-Policy Control: **Q-learning**

$$q_{t+1}(S_t, A_t) = q_t(S_t, A_t) + \alpha_t((R_{t+1} + \underset{a'}{max}\ q_t(S_{t+1}, a')) - q_t(S_t, A_t))$$

$\rightarrow$ behavior policy need not be greedy

Importance Sampling Correction

Using the fact that:

$$\mathbb{E}_{x\sim d}[f(x)] = \mathbb{E}_{x\sim d'}[\frac{d(x)}{d'(x)}f(x)]$$

Apply correction for sampling in another distribution:

$$v_{t+1}(S_t) = v_t(S_t) + \alpha_t(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma v_t(S_{t+1})) - v_t(S_t))$$

$\rightarrow$ Off-Policy TD

Function Approximation

Tabular cases $\rightarrow$ every s, a has corresponding entry $v(s)$ and $q(s, a)$

Larger state/action spaces $\rightarrow$ need **function approximation**

Linear Approximation

$$v_{\mathbf{w}}(s) = \mathbf{w}^T \mathbf{x}(s)$$

If we set the quadratic loss function:

$$J(\mathbf{w}) = \mathbb{E}_{S \sim d}[(v_\pi(S) - \mathbf{w}^T \mathbf{x}(s))^2]$$

This is convex, so SGD converges to global optimum. Since $\nabla_{\mathbf{w}} v_{\mathbf{w}}(S_t) = \mathbf{x}(S_t) = \mathbf{x}_t$,

$$\Delta \mathbf{w} = \alpha(v_\pi(S_t) - v_{\mathbf{w}}(S_t))\mathbf{x}_t$$

$$\mathbf{w}_{\text{MC}} = \mathbb{E}_\pi[\mathbf{x}_t \mathbf{x}_t^T]^{-1} \mathbb{E}_\pi[G_t \mathbf{x}_t]$$

$$\mathbf{w}_{\text{TD}} = \mathbb{E}_\pi[\mathbf{x}_t(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^T]^{-1} \mathbb{E}_\pi[R_{t+1} \mathbf{x}_t]$$

Using **bootstrapping, off-policy learning, function approximation** together may diverge. (deadly triad)

Using **differentiable functions**, we can apply deep reinforcement learning.

Policy-Based Learning

Directly optimizing the policy $\pi_\theta(s,a) = p(a|s,\theta)$ on the Objective:

$$J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \sum_r p(r|s,a) r$$

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi [q_{\pi_\theta}(S_t, A_t) \nabla_\theta log\ \pi_\theta(A_t|S_t)]$$

$\rightarrow$ MC sampling and apply SGA (REINFORCE):

$$\Delta\theta_t = \alpha G_t \nabla_\theta log\ \pi(A_t|S_t)$$

If **baseline** $b(s)$ doesn't depend on the action,

$$\begin{aligned}
\mathbb{E}[b\nabla_\theta \log\ \pi_\theta(A_t|S_t)] &= \mathbb{E}[\sum_a \pi(a|S_t) b \nabla_\theta \log\ \pi_\theta(a|S_t)] \\
&= \mathbb{E}[b\nabla_\theta \sum_a \pi(a|S_t)] \\
&= \mathbb{E}[b\nabla_\theta 1] = 0
\end{aligned}$$

Policy-Based Learning

Using state-value function $v_\pi(S_t)$ as baseline, the **advantage function**:

$$A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$$

The policy gradient becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[A_{\pi_\theta}(s_t,a_t)\nabla_\theta log\ \pi_\theta(a_t|s_t)]$$

Actor-Critic: Policy gradient(π_θ) + Value prediction($v_{\mathbf{w}}$). Repeat:

$$\begin{aligned}
&\text{Sample } A_t, R_{t+1}, S_{t+1} \\
&\delta_t = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t) \\
&\Delta \mathbf{w}_t = \beta \delta_t \nabla_{\mathbf{w}} v_{\mathbf{w}}(S_t) \\
&\Delta \theta_t = \alpha \delta_t \nabla_\theta log\ \pi_\theta(A_t|S_t)
\end{aligned}$$

δ_t is one-step TD error, and advantage here.

Increasing Stability

In RL, data depends on policy, but it keeps changing

$\rightarrow$ limit the difference between subsequent policies

policy gradient with constraint:

$$J(\theta) - \eta \mathrm{KL}(\pi_{\text{old}}|\pi_\theta)$$

RLHF

Deep Reinforcement Learning from Human Preferences (OpenAI, 2017)

Environment giving reward? No.

$\Longrightarrow$ **human overseer** who can express preferences between trajectory segments

How to make the reward model $\hat{r}(o_t, a_t)$ for this?

If we have two trajectory with following **preference**,

$$((o_0^1, a_0^1), \ldots, (o_{k-1}^1, a_{k-1}^1)) \succ ((o_0^2, a_0^2), \ldots, (o_{k-1}^2, a_{k-1}^2))$$

It should mean that

$$r(o_0^1, a_0^1) + \ldots + r(o_{k-1}^1, a_{k-1}^1) > r(o_0^2, a_0^2) + \ldots + r(o_{k-1}^2, a_{k-1}^2)$$

Bradley-Terry model

For trajectory σ_1, σ_2 generated by policy π:

$$\hat{P}(\sigma_1, \sigma_2) = \frac{\exp\left(\sum \hat{r}(o_t^1, a_t^1)\right)}{\exp\left(\sum \hat{r}(o_t^1, a_t^1)\right) + \exp\left(\sum \hat{r}(o_t^2, a_t^2)\right)}$$

$$\text{loss}(\hat{r}) = -\sum_{(\sigma_1,\sigma_2)\in D} I(\sigma_1 \succ \sigma_2)\log \hat{P}(\sigma_1, \sigma_2) + I(\sigma_1 \prec \sigma_2)\log \hat{P}(\sigma_2, \sigma_1)$$

$\rightarrow$ Estimate $\hat{r}$ as adding data to D. Apply RL algorithms to update π. Can be done asynchronously.

ChatGPT

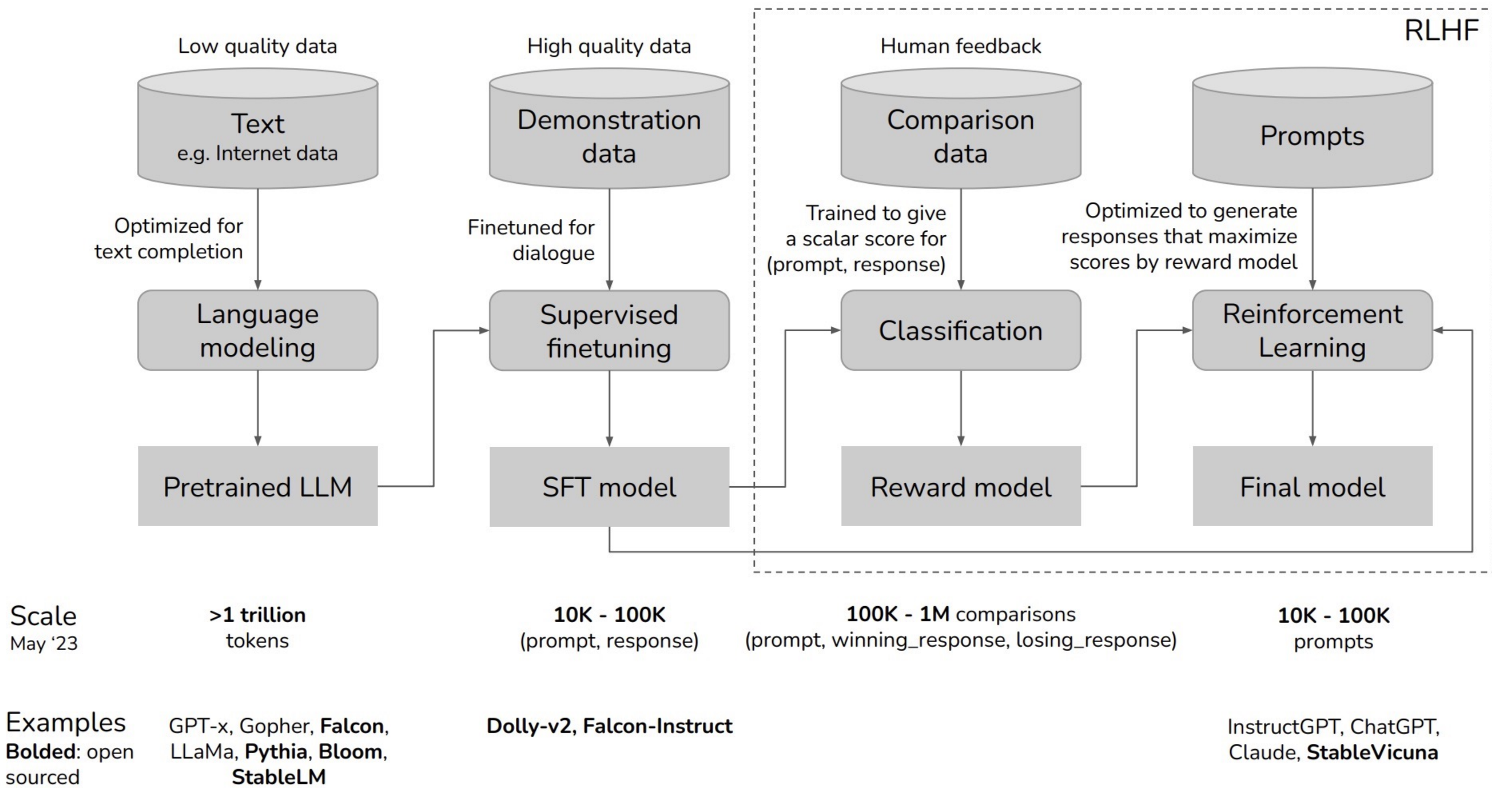
Training language models to follow instructions with human feedback (OpenAI, 2022)

Let's Apply RLHF in **Language Models**.

Alignment

"Follow the user's instructions helpfully and safely"

$\rightarrow$ fine-tune LM on the reward model representing human preference scores



ChatGPT

Simplify the problem, assume **Contextual Bandit**

O: prompt x

A: response y

For K responses, 1 vs 1 preference comparison $\rightarrow$ get $\binom{K}{2}$ tuples of (x, y_w, y_l)

Reward Modeling

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))\right]$$

$$\left(\because \sigma(a-b) = \frac{1}{1+e^{-(a-b)}} = \frac{e^a}{e^a+e^b}\right)$$

RL(policy gradient), by sampling response y from π_ϕ^{RL}

$$\text{objective}(\phi) = \mathbb{E}_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x, y) - \beta \log \frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right]$$

* Response y is **tokenized** as $y_1, ..., y_n$

$$\pi(y|x) = \pi(y_1|x)\pi(y_2|x, y_1)...\pi(y_n|x, y_1, ..., y_{n-1})$$

DPO

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

(Stanford, 2023)

RLHF is unstable $\rightarrow$ Do we really need Reward Modeling + RL ?

Generalized form of RLHF using KL constraint:

Train the reward model with loss function:

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))\right] \tag{1}$$

And find optimal policy that maximizes:

$$\mathbb{E}_{(x\sim D, y\sim\pi_\theta(y|x))}\left[r_\phi(x, y)\right] - \beta\mathbb{D}_{\mathsf{KL}}\left[\pi_\theta(y|x) \,||\, \pi_{\mathsf{ref}}(y|x)\right] \tag{2}$$

The optimal solution of (2) is:

$$\pi_r(y|x) = \frac{1}{Z(x)}\pi_{\mathsf{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x, y)\right) \tag{3}$$

with partition function

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

DPO

Proof of (3)

$$
\begin{aligned}
&\underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{x\sim\mathcal{D},y\sim\pi}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathsf{KL}}\left[\pi(y|x)||\pi_{\mathsf{ref}}(y|x)\right] \\
&= \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[r(x,y) - \beta\log\frac{\pi(y|x)}{\pi_{\mathsf{ref}}(y|x)}\right] \\
&= \underset{\pi}{\operatorname{argmin}}\, \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\mathsf{ref}}(y|x)} - \frac{1}{\beta}r(x,y)\right] \\
&= \underset{\pi}{\operatorname{argmin}}\, \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\mathsf{ref}}(y|x)} - \frac{1}{\beta}r(x,y) + \log Z(x) - \log Z(x)\right] \\
&= \underset{\pi}{\operatorname{argmin}}\, \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\mathsf{ref}}(y|x)\exp(\frac{1}{\beta}r(x,y))} - \log Z(x)\right]
\end{aligned}
$$

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

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

$\pi^*(y|x) \geq 0$ and $\sum_y \pi^*(y|x) = 1 \implies \pi^*$ is valid policy.

Also, $Z(x)$ is independent to π

DPO

Continue. Substituting π^*,

Proof.

$$
\begin{aligned}
\ldots &= \underset{\pi}{\operatorname{argmin}}\, \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] \\
&= \underset{\pi}{\operatorname{argmin}}\, \mathbb{E}_{x\sim D}\left[\mathbb{D}_{\mathrm{KL}}(\pi(y|x)||\pi^*(y|x)) - \log Z(x)\right] \\
&= \underset{\pi}{\operatorname{argmin}}\, \mathbb{E}_{x\sim D}\left[\mathbb{D}_{\mathrm{KL}}(\pi(y|x)||\pi^*(y|x))\right] \\
&= \pi^*(y|x)
\end{aligned}
$$

□

From (3), we get:

$$
r(x,y) = \beta\log\frac{\pi_r(y|x)}{\pi_{\mathsf{ref}}(y|x)} + \beta\log Z(x) \tag{4}
$$

Apply (4) to (1), and $Z(x)$ cancels:

$$
\therefore \mathcal{L}_{\mathsf{DPO}}(\pi_\theta;\pi_{\mathsf{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\mathsf{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\mathsf{ref}}(y_l|x)})\right] \tag{5}
$$

$\rightarrow$ No longer a RL problem. No need of reward model. **Direct optimization**

$+$ Enhanced Stability

Future Research

So, is RL useless for finetuning language models?

We want to keep on communicating with LMs, tons of problems occur in **multi-turn** conversation:
ex) Snowballing Hallucination, Repeating, Forgetting, ...
$+$Lifelong learning language models, personalized to user

$\rightarrow$Need a design for long-term goal