
ChatGPT
狂飙
原理剖析
ZOMI

ChatGPT Talk Overview

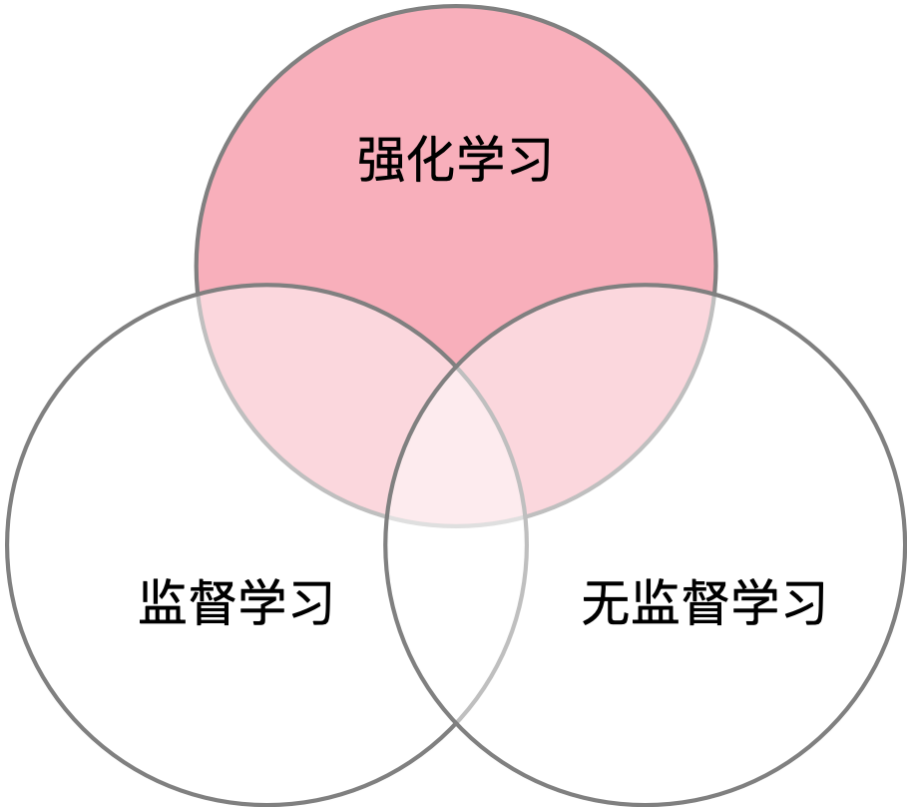
1. BERT 模型与 GPT 模型系列
2. 强化学习加入人类反馈 RLHF 模式
3. 强化学习 PG 和 PPO 算法
4. InstructGPT 原理深度剖析

RL引入人类反馈

RL + HF 模式

强化学习 RL

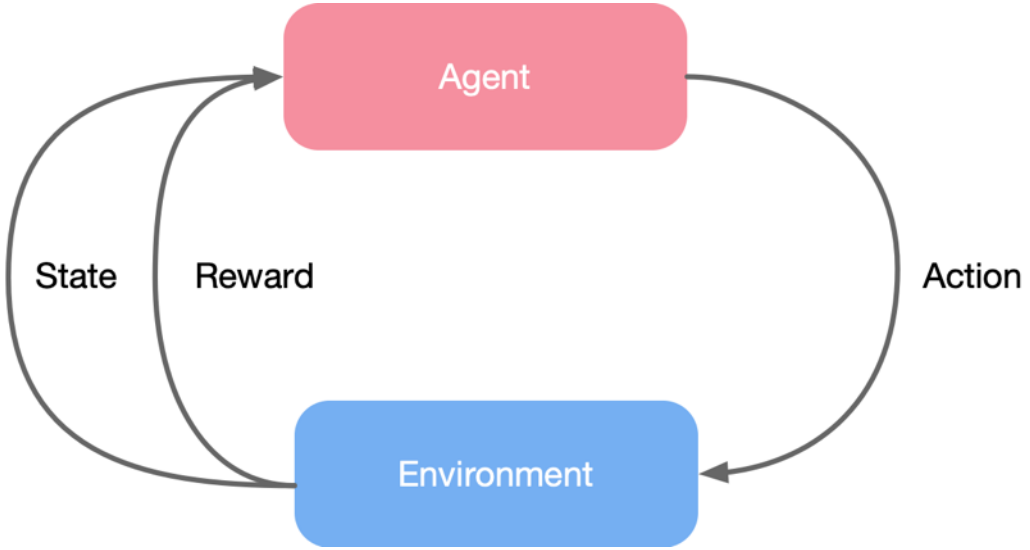
- 强化学习（Reinforcement Learning, RL），是机器学习的范式和方法论之一，用于描述和解决智能体（Agent）在与环境（Environment）的交互过程中通过学习策略以达成回报最大化或实现特定目标的问题。



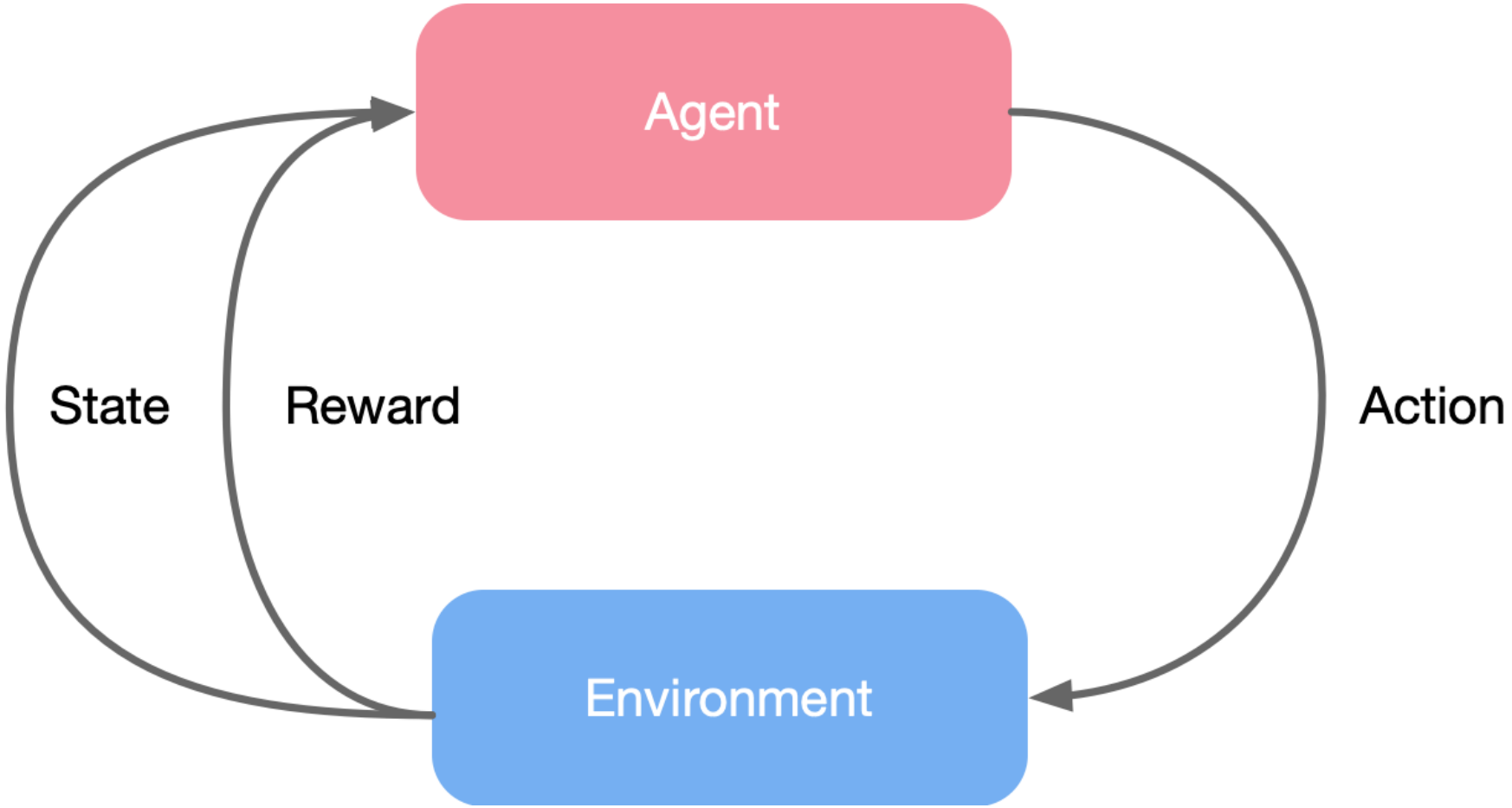
强化学习 RL

强化学习中有两个可以进行交互的对象：

1. **智能体（Agent）**：感知环境状态（State），根据反馈奖励（Reward）选择合适动作（Action）最大化长期收益，在交互过程进行学习；
2. **环境（Environment）**：接收智能体执行的一系列动作，对这一系列动作进行评价并转换为一种可量化的信号，最终反馈给智能体。



强化学习 RL



强化学习 RL 基本概念

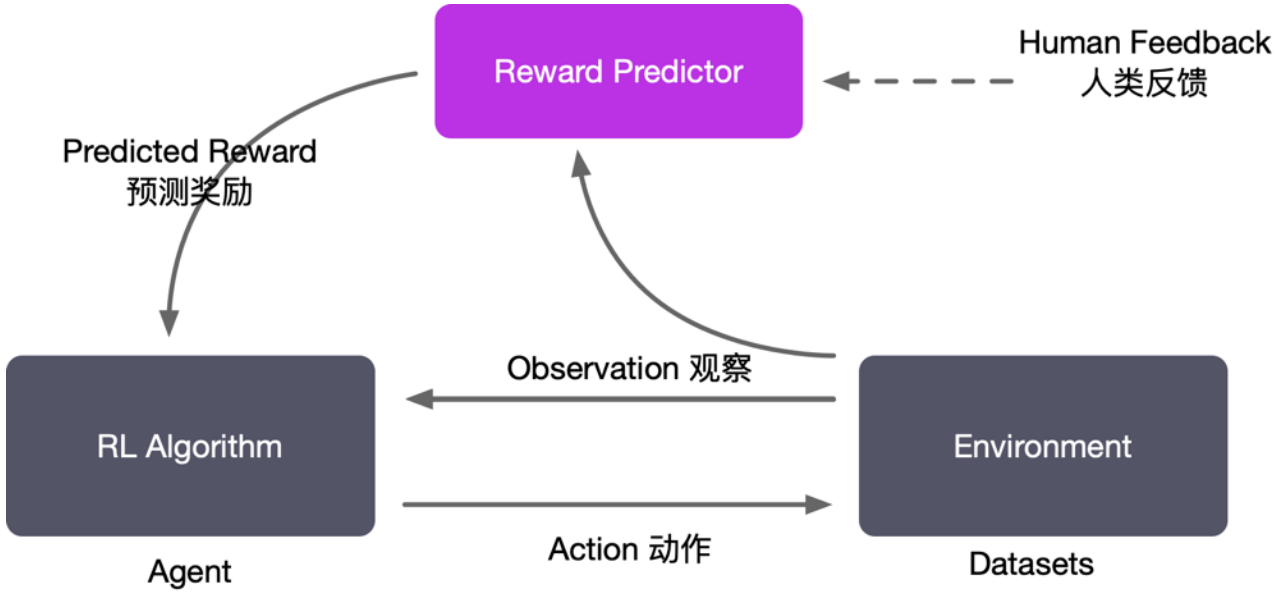
1. **策略 (Policy)**：定义智能体在特定时间 t 选择的行为方式，策略是环境状态到动作的映射。
2. **奖励函数 (Reward Function)**：在每一步中，环境向智能体发送一个奖励收益 Reward，而这个收益通过奖励函数计算得到。
3. **价值函数 (Value Function)**：从长远的角度看什么是好的，一个状态的价值是一个智能体从这个状态开始，对将来累积的总收益的期望。
4. **环境模型 (Environment Model)**：是一种对环境的反应模式的模拟，它允许对外部环境的行为进行推断。

RLHF：从人类反馈中学习

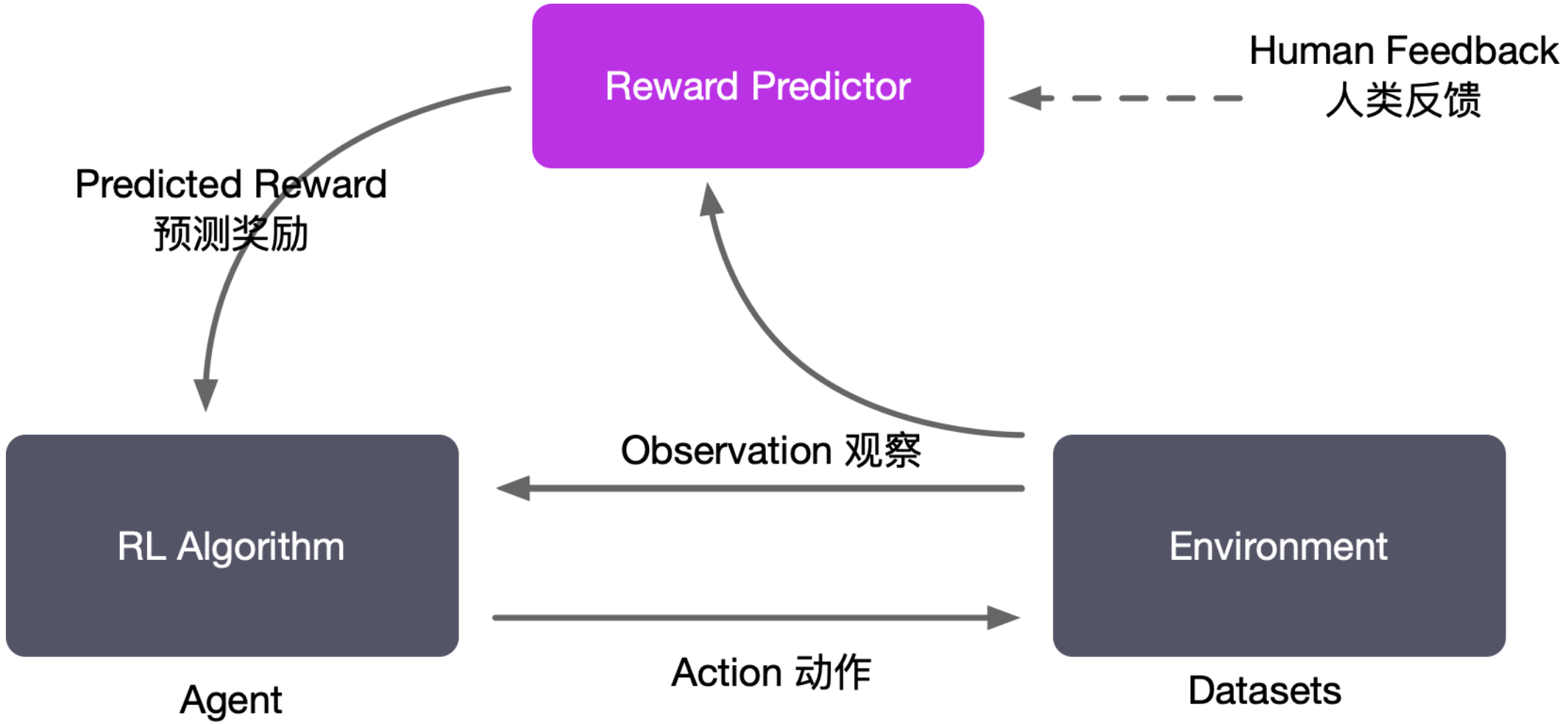
- Native 强化学习里，有 Environment 和 Reward Model，但逆强化学习没有奖励函数，只有一些人类/专家的示范，怎么办呢？

RLHF：从人类反馈中学习

- 通过人类标注数据训练得到 Reward Model（相当于有了人类标注数据，则相信它是不错的，然后反推人类因为什么样的奖励函数才会采取这些行为），有了奖励函数之后，就可以使用一般的强化学习的方法去找出最优策略/动作。



RLHF：从人类反馈中学习

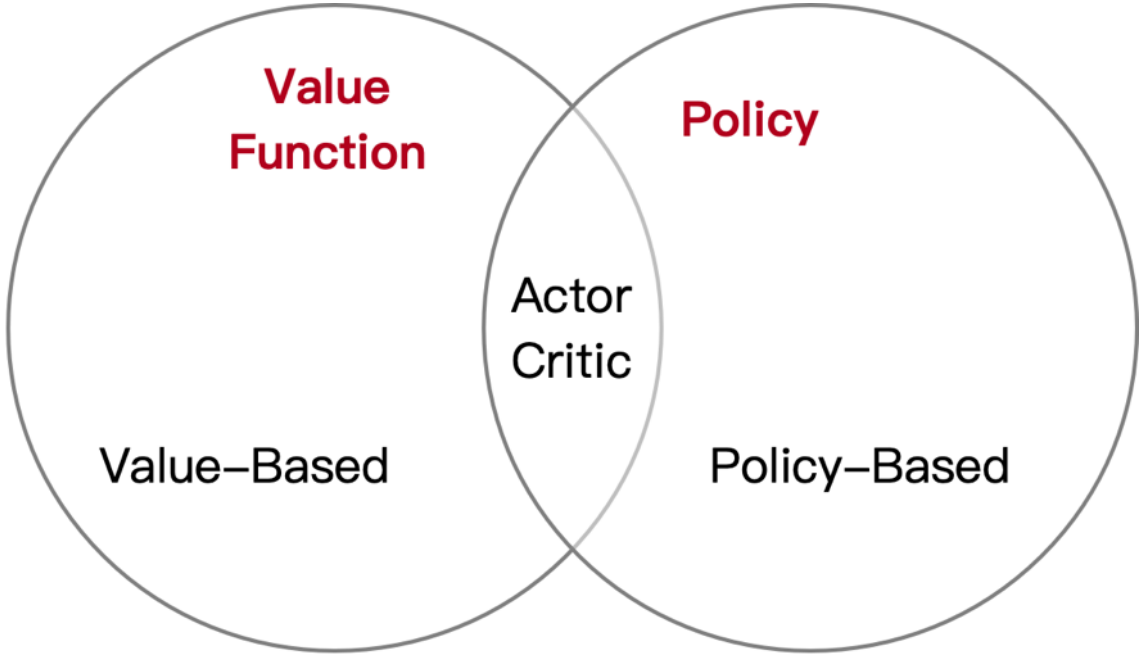


强化学习算法

策略梯度 PG 到 PPO 算法

强化学习与深度学习结合

- 强化学习可以按照方法学习策略来划分成 Value-based 和 Policy-based 两种。在深度强化学习领域将深度学习与基于值 Q-Learning 算法相结合产生了 DQN 算法，通过经验回放池与目标网络成功的将深度学习算法引入了强化学习算法。其中最具代表性分别是 Q-Learning 与 Policy Gradient 算法。



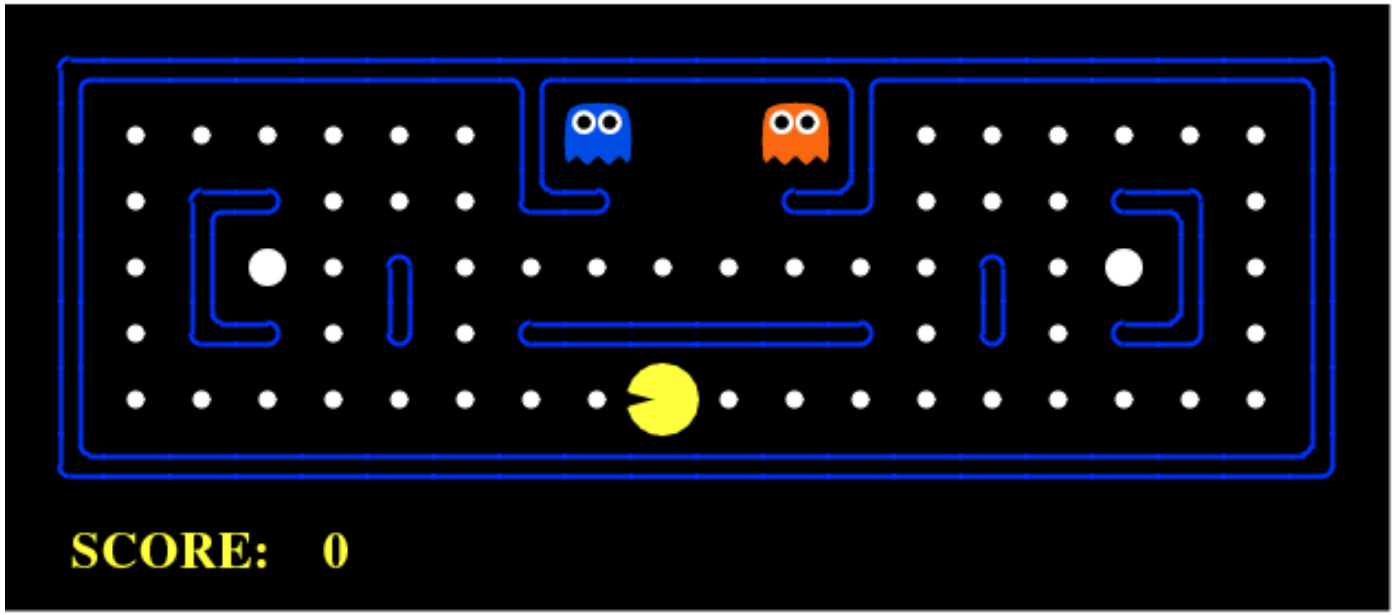
- Value Based
  - Learn value function
  - implicit policy
- Policy Based
  - no value function
  - learn policy
- Actor-Critic

PG Policy Gradient 策略梯度下降

- **Value-based**：比较 a1, a2, a3 三个个动作的期待值（Q-value），选取Q最大的那个作为本次选择的动作。
- **Policy-based**：有一个计算此刻选择哪个动作的函数（actor），并得到概率概率 $p(s,a)$，根据概率 $p(s,a)$ 选取动作。

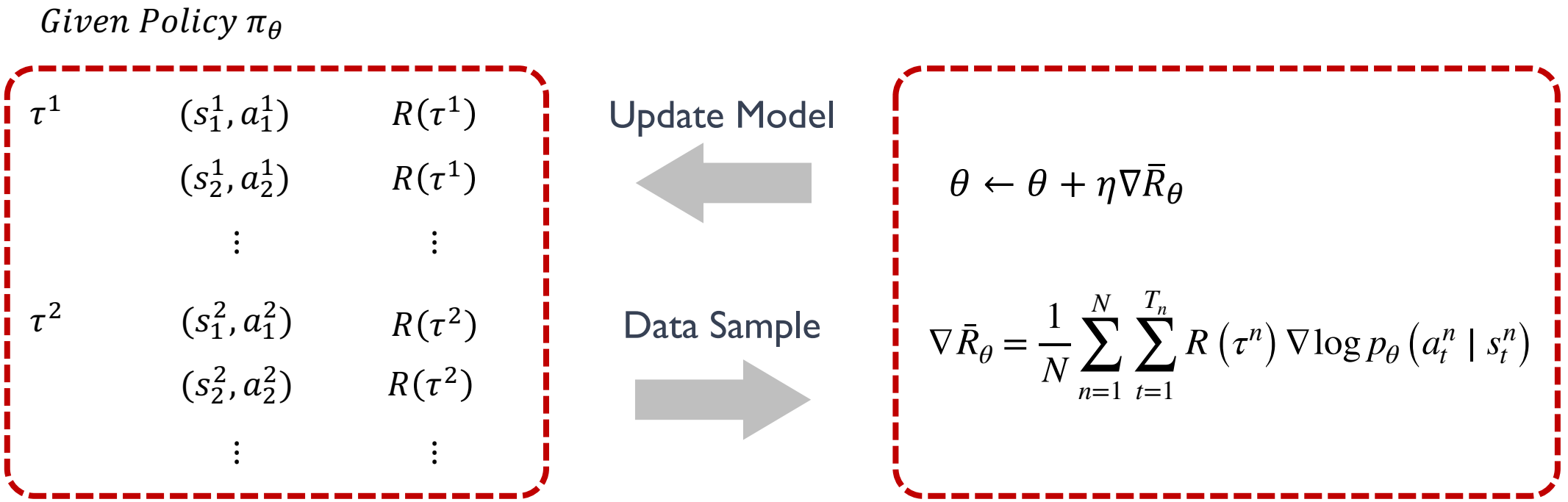
PG Policy Gradient 策略梯度下降

- 相比于 Q-learning 和 DQN，策略梯度下降 PG 的神经网络不再输出 Q 值，而是直接输出采用动作的概率，能够更好地处理连续的动作。



PG Policy Gradient 策略梯度下降

- 在实际实验中，会让 actor 和 environment 进行互动，产生一系列采样数据（Episode sample / Trajectory），即获得很多 (s, a) 的 Pair（表示在状态 s 下采取动作 a ，得到当前奖励 $R(\tau)$，然后将这些数据送入训练过程中计算，并更新模型的参数 θ。



Proximal Policy Optimization PPO 算法原理

- 对于 PG 算法来说，最大的问题是在策略参数更新后，还要需要重新使用同环境互动收集数据再进行下一轮迭代。
- PPO 算法是利用了重要性采样的思想，在不知道策略路径的概率 p 情况下，通过模拟一个近似的 q 分布，只要 p 同 q 分布不差得太远，通过多轮迭代可以快速参数收敛。

Proximal Policy Optimization PPO 算法原理

- PPO算法结合 Actor-Critic 方式， Agent 由两部分组成，Actor 负责与环境互动收集样本，等同于原来 PG 的情况，其更新即 PPO 梯度的更新，添加了 Critic，用于负责评判 Actor 的动作好坏。

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

Proximal Policy Optimization PPO 算法原理

- PPO算法结合 Actor-Critic 方式， Agent 由两部分组成，Actor 负责与环境互动收集样本，等同于原来 PG 的情况，其更新即 PPO 梯度的更新，添加了 Critic，用于负责评判 Actor 的动作好坏。

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

引用

1. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)
2. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9
3. Dale, Robert. "GPT-3: What's it good for?." Natural Language Engineering 27.1 (2021): 113-118
4. Floridi, Luciano, and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences." Minds and Machines 30 (2020): 681-694.
5. Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901
6. Ouyang, Long, et al. "Training language models to follow instructions with human feedback." arXiv preprint arXiv:2203.02155 (2022)
7. https://openai.com/blog/instruction-following/
8. https://openai.com/blog/chatgpt/
9. https://sh-tsang.medium.com/review-instructgpt-training-language-models-to-follow-instructions-with-human-feedback-7fce4bf9059a
10. https://jalammar.github.io/how-gpt3-works-visualizations-animations/
11. https://jalammar.github.io/