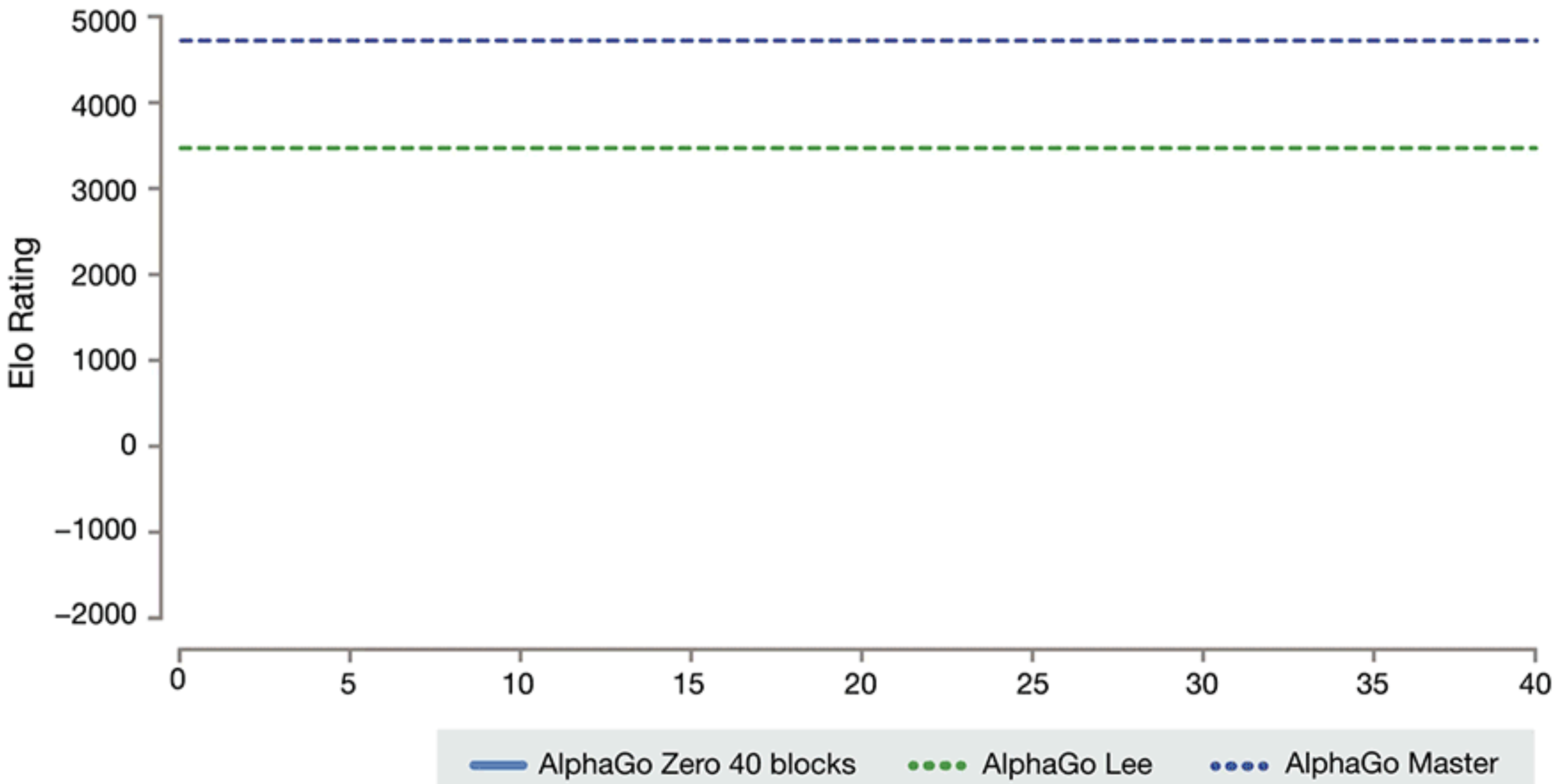
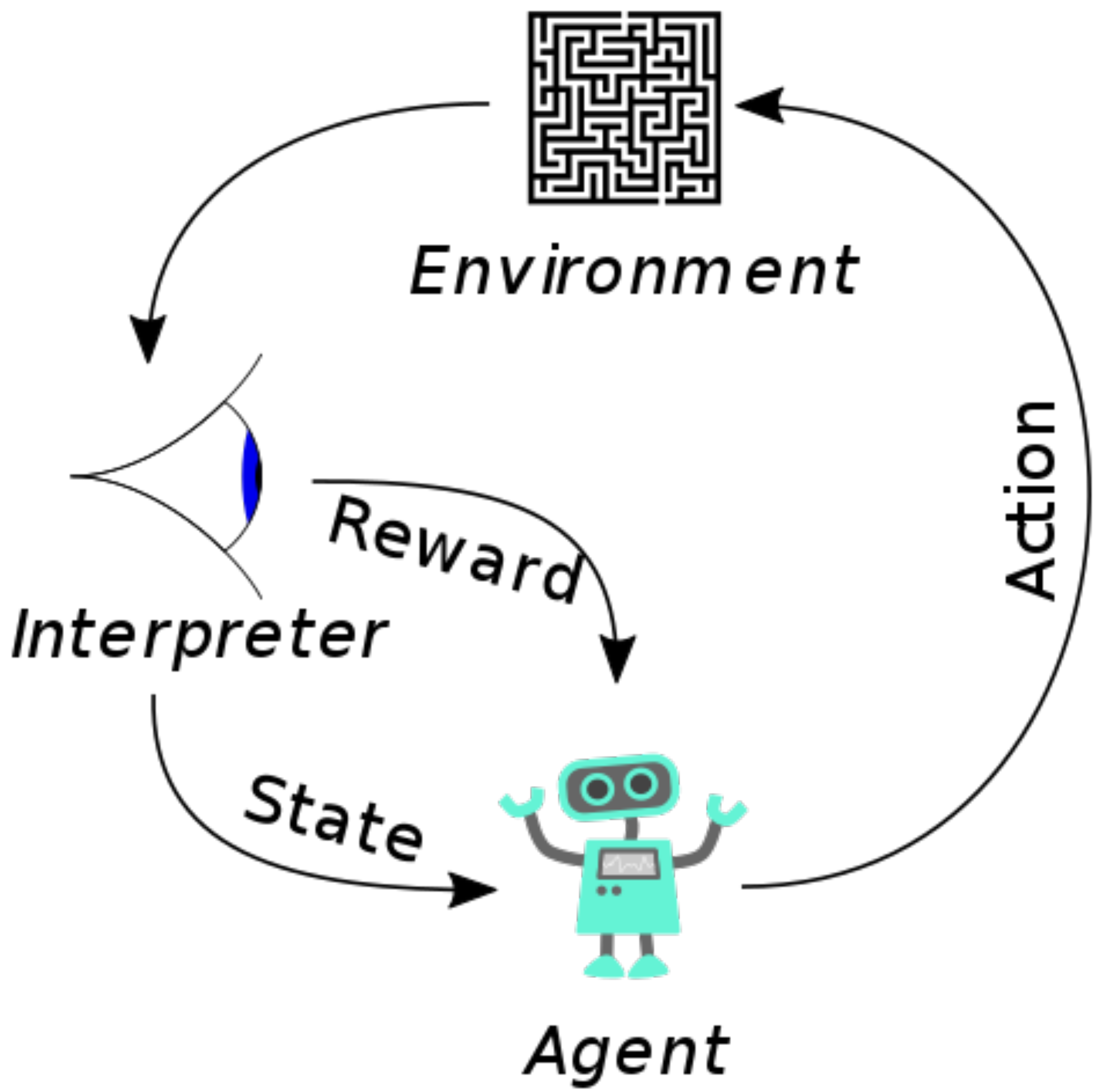
Reinforcement Learning: Basics

AlphaGo Zero

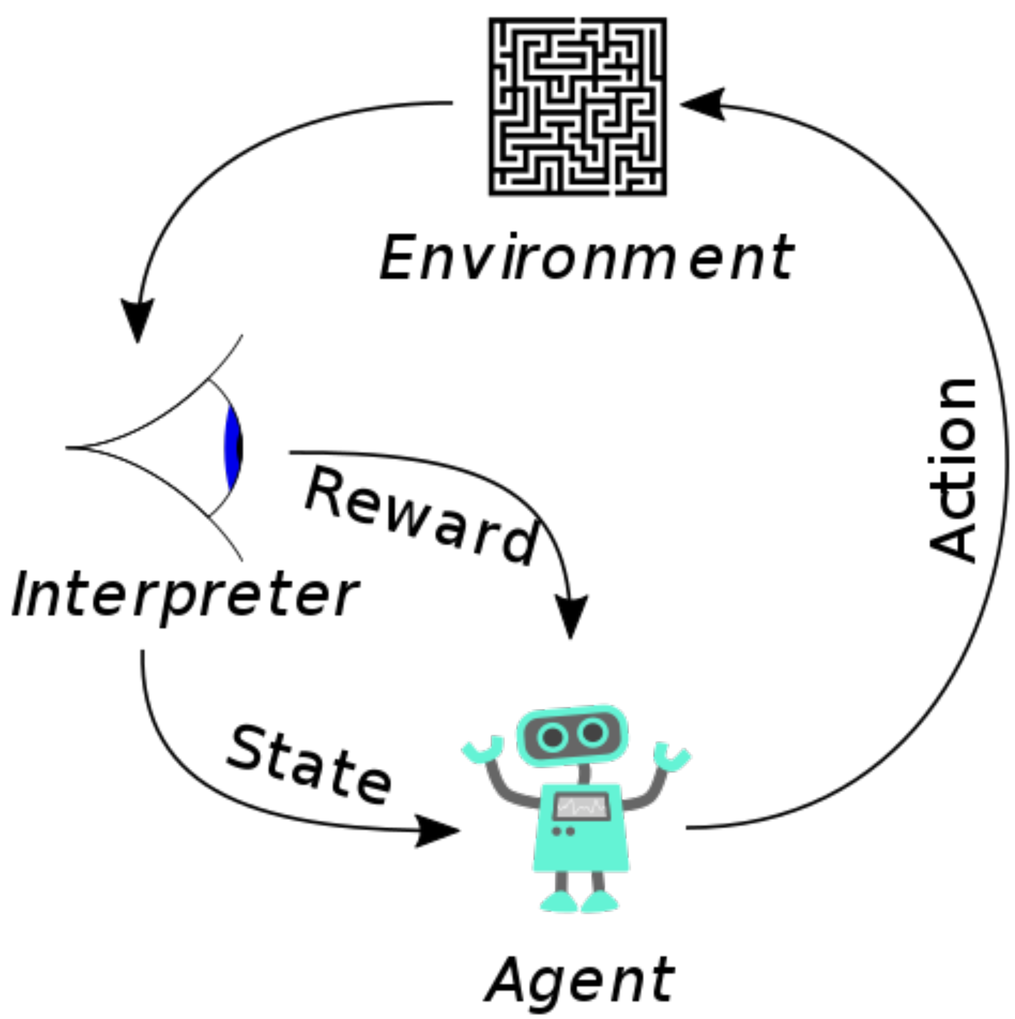


https://deepmind.com/blog/alphago-zero-learning-scratch/

问题框架



问题框架



- Environment / Agent
- 序列决策
- 反馈

强化学习

- 目标：优化策略，以最大化总收益
- 问题特点：
  - 序列性
  - 弱监督（反馈信号）
  - 延迟收益（长期反馈）

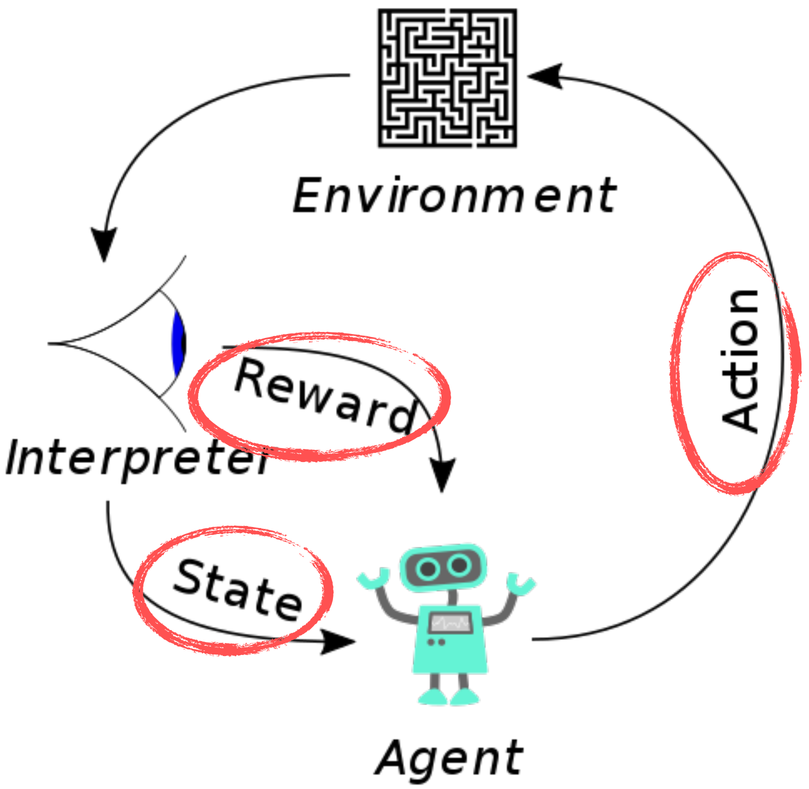
应用场景

- 游戏：棋类游戏、牌类游戏、电子游戏等
- 控制：飞行器控制、机器人行走、自动驾驶、电梯调度等
- 决策：炒股、投资、推荐系统、医疗决策等
- 设计：电路设计、网络设计、汽车设计等
- 序列生成：音乐生成、翻译、对话生成等
- Meta learning：学习模型超参、动态模型等

问题建模

Environment

- 环境状态 (state) 及状态转移
- 可供执行的动作 (action)
- 瞬时反馈 (reward)、总收益 (return)



将以上要素建模为 Markov Decision Process (MDP)

MDP

A Markov decision process is a 5-tuple $(S, A, P_\cdot(\cdot,\cdot), R_\cdot(\cdot,\cdot), \gamma)$, where

状态空间 • S is a finite set of states,

指令空间 • A is a finite set of actions (alternatively, A_s is the finite set of actions available from state s),

转移矩阵 • $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t+1$,

反馈 • $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a

衰减因子 • $\gamma \in [0, 1]$ is the discount factor, which represents the difference in importance between future rewards and present rewards.

https://en.wikipedia.org/wiki/Markov_decision_process

Return

- 用G_t表示 t 时刻开始的所有操作的**收益（return）**：

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{a_{t+k}}(s_{t+k}, s_{t+k+1})$$

- 收益即：total discounted reward

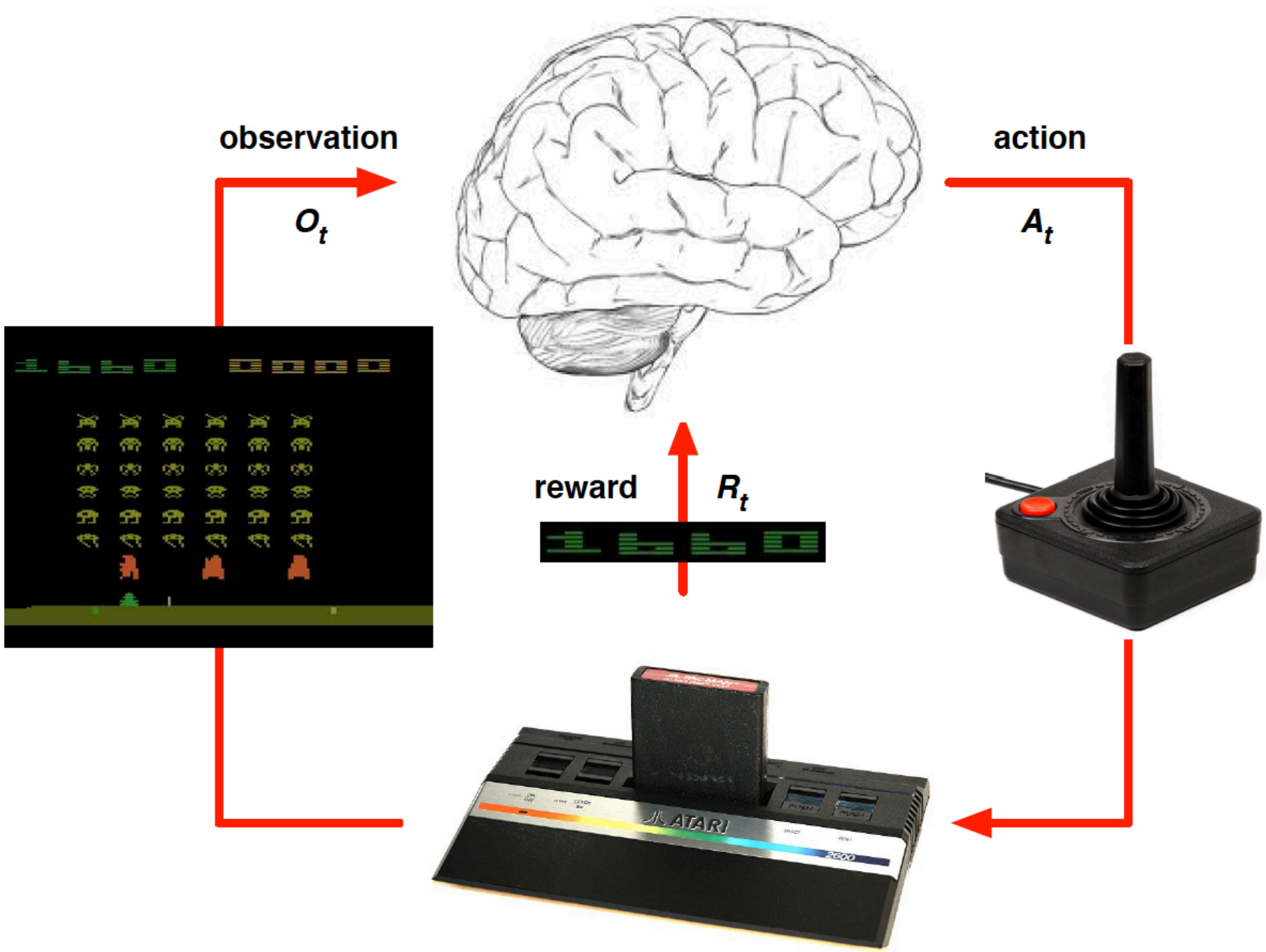
- 为什么要 discount?

POMDP and Belief States

- 无法得到真实状态：Partially Observable
- Observation and observation function
- Belief state: 由之前的 belief、action、observation 得到
- 以 belief state 为环境状态，转化为 MDP

https://en.wikipedia.org/wiki/Partially_observable_Markov_decision_process

举个例子



Agent

三个要素：

- Policy
- Value function
- Model

Policy

Policy 形式化地表示 agent 如何做出决策：

$$\pi(a|s) = \Pr(a_t = a|s_t = s)$$

- 定义了 agent 的行为
- 在 MDP 中，决策方式**时间无关**
- 给定决策后，MDP 中每个状态的期望 return 即可确定

Value function

- 给定 policy π，状态的**价值** state-value function 定义为收益期望：

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

- 类似地，动作的**价值** action-value function 定义为：

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

二者可互相推得

最大收益与最优策略

- RL 的目标可形式化为：寻找最优策略 π^*，使得对于每个状态 s，其价值函数 $v_{\pi^*}(s)$ 均为最大

- 等价于寻找最大价值函数 $q^*(s,a)$，取策略

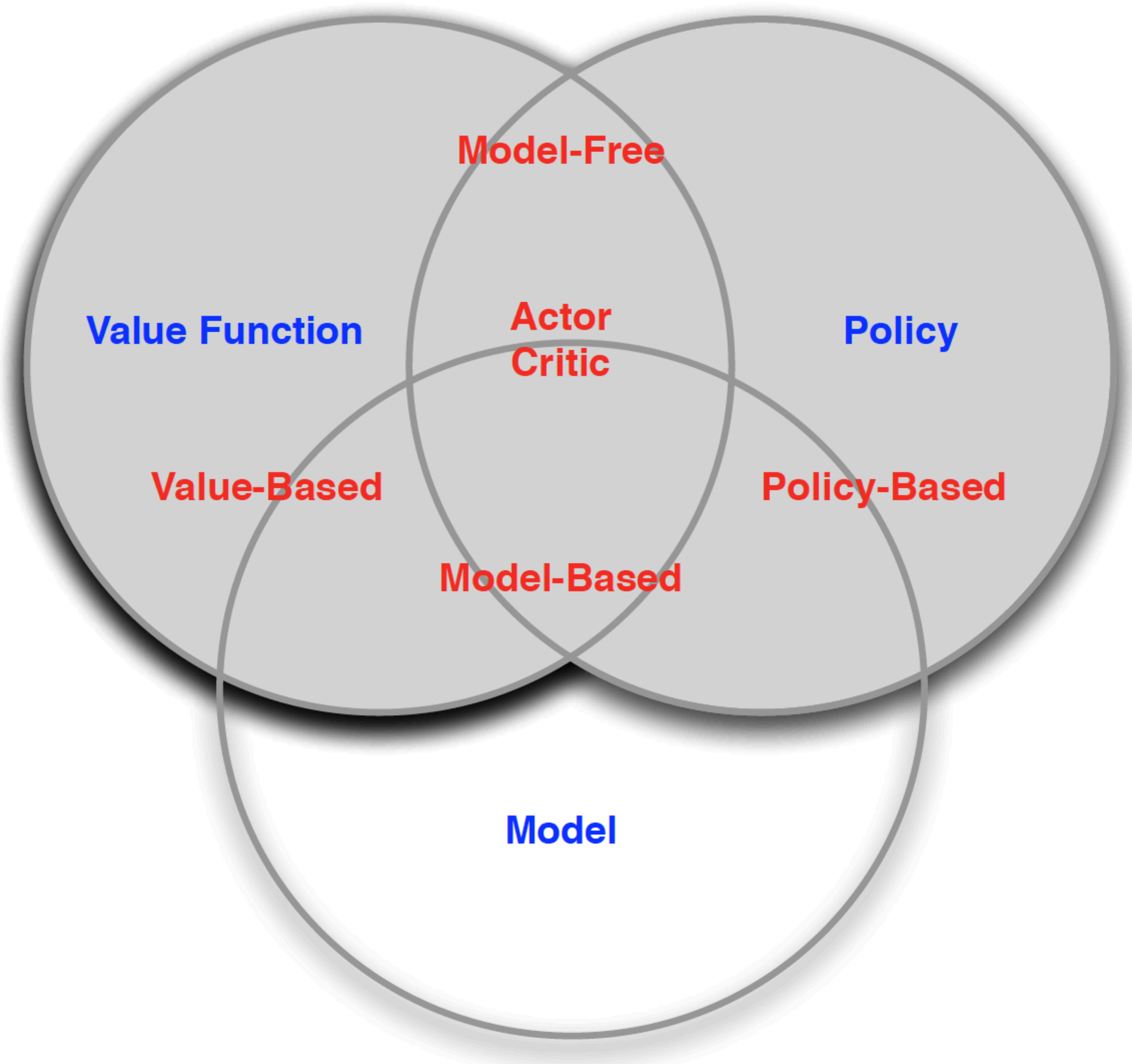
$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a \in \mathcal{A}} q^*(s,a) \\ 0, & \text{otherwise} \end{cases}$$

即可获得该最大收益

Model

- Model 预测模型的状态转移和反馈
- 完美的 model 即 MDP 中的 P、R 函数
- 最优策略可通过规划 (planning) 得到

RL agent 分类



Model-Free
Value Function
Actor
Critic
Policy
Value-Based
Policy-Based
Model-Based
Model

强化算法

Value-Based: Q Learning

思路：

- 使用神经网络 $Q(s, a; \theta_i)$ 来估计最大期望收益 $q^*(s, a)$
- 使用估计的价值函数做决策

(Deep) Q-Learning 算法

- 最优估价函数应满足 **Bellman 方程**：

$$q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a' \in \mathcal{A}} q^*(s',a') \,|\, s, a\right]$$

- 将右边作为 target，可得到损失函数，用于随机梯度下降：

$$L_i(\theta_i) = \mathcal{L}\left[r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right]$$

MSE, Huber loss, etc.

之前的参数，定期更新

Experience Replay

- 注意到每个样本为一个四元组：$\langle s, a, s', r \rangle$
- 数据分布变化、数据间 correlation 导致训练**不稳定**
- 从既有的经验中随机采样：$\langle s, a, s', r \rangle \sim U(D)$

experience

一些改进

- **Double DQN**： policy 选择与 value 估计使用两套参数

$$L_i(\theta_i) = \mathcal{L}\left[r + \gamma Q\left(s', \arg\max_{a'} Q(s', a'; \theta_i); \theta_i^-\right) - Q(s, a; \theta_i)\right]$$

策略选择　估价

- **Dueling Network**： 将 action-value 的估计解构为对状态固有价值和对动作附加价值各自的估计之和

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

- **Prioritized replay**： 根据样本的误差大小采样

Policy-Based: Policy Gradient Descent

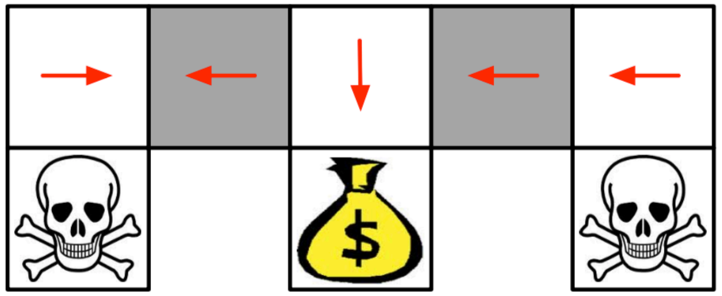
思路：

- 神经网络直接输出策略 $\pi(a|s;\theta)$ 或 $a = \pi(s;\theta)$
- 目标为最大化最终收益
- 对每次完整决策序列（如一场完整游戏）进行一次 BP

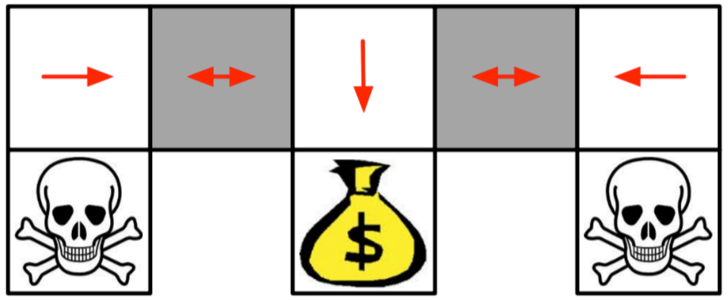
特点

Policy-based RL：

- 更容易收敛
- 但更容易收敛于局部最优
- 难以评估训练程度、最终效果
- 可学习**随机策略**（**stochastic policies**） ←信息不完整的情况下有意义
- 适于学习高维 / **连续值**的 action ←如机器人控制



deterministic policy



stochastic policy

Policy Gradients

- 目标函数（以 return 为例）：

$$J(\theta) = v_{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[q_{\pi_\theta}(s_1, a)]$$

- 策略梯度（policy gradients）：

分布 $\pi(a|s;\theta) \rightarrow \nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi(a|s;\theta) q_{\pi_\theta}(s,a)]$

连续值 $a = \pi(s;\theta) \rightarrow \nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\frac{\partial q_{\pi_\theta}(s,a)}{\partial a} \nabla_\theta \pi(s;\theta)\right]$

Policy Gradient Theorem

Actor-Critic Algorithm

- **Actor-Critic** Value、policy 均用神经网络拟合：
  - Critic $Q(s,a;\omega) \rightarrow q_{\pi_\theta}(s,a)$ ←**policy evaluation**
  - Actor $\pi(a|s;\theta)$ $a = \pi(s;\theta)$ ←**policy gradient ascend**
- 分别、交替训练

Asynchronous **Advantage** Actor-Critic (A3C)

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi(a|s;\theta) q_{\pi_\theta}(s,a)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi(a|s;\theta)(V(s) + A_{\pi_\theta}(s,a))\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi(a|s;\theta) A_{\pi_\theta}(s,a)\right]\end{aligned}$$

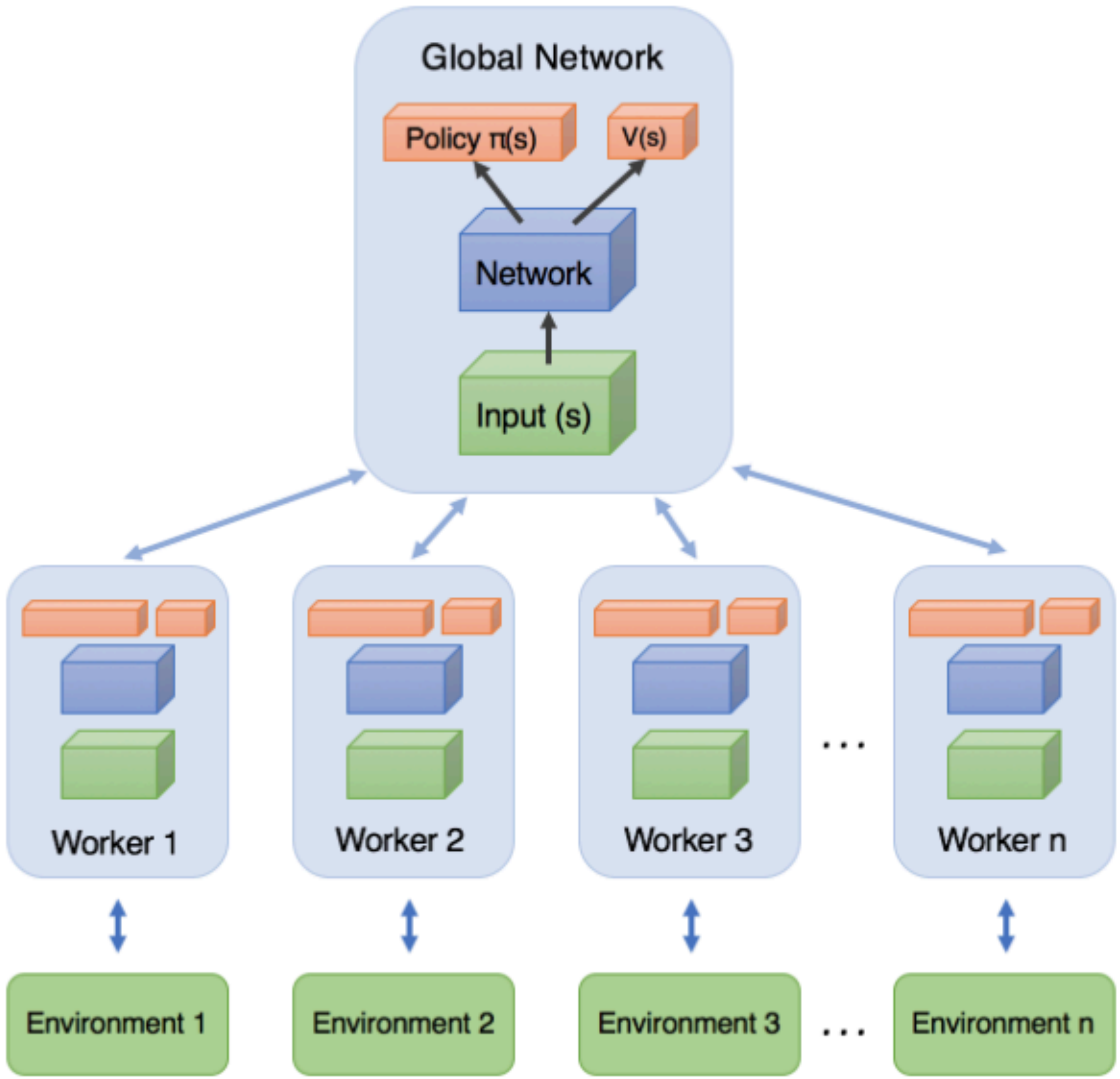
Advantage

- 据此使用两个估值函数：

$$V(s;\nu) \rightarrow V_{\pi_\theta}(s), Q(s,a;\omega) \rightarrow Q_{\pi_\theta}(s,a)$$

- 可直接将 return 用作 Q 的估计

Asynchronous Advantage Actor-Critic (A3C)



Deterministic & Continuous: DDPG

Deep Deterministic Policy Gradient Algorithm:

- Critic 更新采用类似 DQN 的方式：

$$loss = \mathcal{L}\left[r + \gamma Q(s', \pi(s'; \theta^-); \omega^-) - Q(s, a; \omega_i)\right]$$

- Actor 根据连续值策略的 policy gradient 更新

Model-Based

- Model 已知且完全可见（如围棋）
- Model 未知：建模
- 根据模型进行**规划**或**模拟+学习**

应用举例

对话生成

- 针对的问题：聊天机器人容易把天聊死

A: Where are you going? (1) B: I'm going to the restroom. (2) A: See you later. (3) B: See you later. (4) A: See you later. (5) B: See you later. (6) ... ...
A: how old are you? (1) B: I'm 16. (2) A: 16? (3) B: I don't know what you are talking about. (4) A: You don't know what you are saying. (5) B: I don't know what you are talking about . (6) A: You don't know what you are saying. (7) ...

- 方案：利用 RL 鼓励好的对话（有前瞻性、交互性、相关性、信息丰富）

[Deep Reinforcement Learning for Dialogue Generation]

对话生成

一轮对话可建模为MDP：

- **State** 对话历史，由 LSTM encoder 转换为模型 state
- **Action** 生成一句话
- **Reward** 鼓励后续性、关联性，惩罚重复信息
- **Policy** LSTM encoder-decoder 结构，输出随机策略
- 训练：policy gradient descend

效果对比

Baseline mutual information model (Li et al. 2015)	Proposed reinforcement learning model
A: Where are you going? (1) B: I'm going to the restroom. (2) A: See you later. (3) B: See you later. (4) A: See you later. (5) B: See you later. (6)	A: Where are you going? (1) B: I'm going to the police station. (2) A: I'll come with you. (3) B: No, no, no, no, you're not going anywhere. (4) A: Why? (5) B: I need you to stay here. (6) A: I don't know what you are talking about. (7) ...
A: how old are you? (1) B: I'm 16. (2) A: 16? (3) B: I don't know what you are talking about. (4) A: You don't know what you are saying. (5) B: I don't know what you are talking about . (6) A: You don't know what you are saying. (7) ...	A: How old are you? (1) B: I'm 16. Why are you asking? (2) A I thought you were 12. (3) B: What made you think so? (4) A: I don't know what you are talking about. (5) B: You don't know what you are saying. (6) ... ...

A: What’s your name ?
B: Daniel.
A: How old are you ?
B. Twelve. What’s your name ?
A. Daniel.
B: How old are you ?
A: Twelve. What’s your name ?
B: Daniel.
A: How old are you ?
B ...

在线竞价

京东将其在线竞价系统建模为 POMDP：

- **State** t 时刻状态京东系统中由一段文字描述表示
- **Action** t 时刻的动作为一次 bid
- **Reward** 每次 bid 的净利润
- **Value Function** 使用 CNN 网络作为估值函数
- 训练：文章提出的 DQN 的并发版

LADDER: A Human-Level Bidding Agent for Large-Scale Real-Time Online Auctions

Thanks

1. Silver D. UCL Course on RL
2. Silver D. Tutorial: Deep Reinforcement Learning
3. Gosavi A. Reinforcement learning: A tutorial survey and recent advances[J]. INFORMS Journal on Computing, 2009, 21(2): 178-192.
4. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
5. Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.
6. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.
7. Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning[C]//AAAI. 2016: 2094-2100.
8. Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay[J]. arXiv preprint arXiv:1511.05952, 2015.
9. Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[J]. arXiv preprint arXiv:1511.06581, 2015.
10. Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]//International Conference on Machine Learning. 2016: 1928-1937.
11. Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.