

多智能体强化学习实训

Multi-agent Reinforcement Learning

Lecture 5: Deep Q Network

教师：张寅

zhangyin98@zju.edu.cn

助教：邓悦

devindeng@zju.edu.cn

王子瑞

ziseoiwong@zju.edu.cn

李成林

chenglinli@zju.edu.cn

浙江大学计算机学院

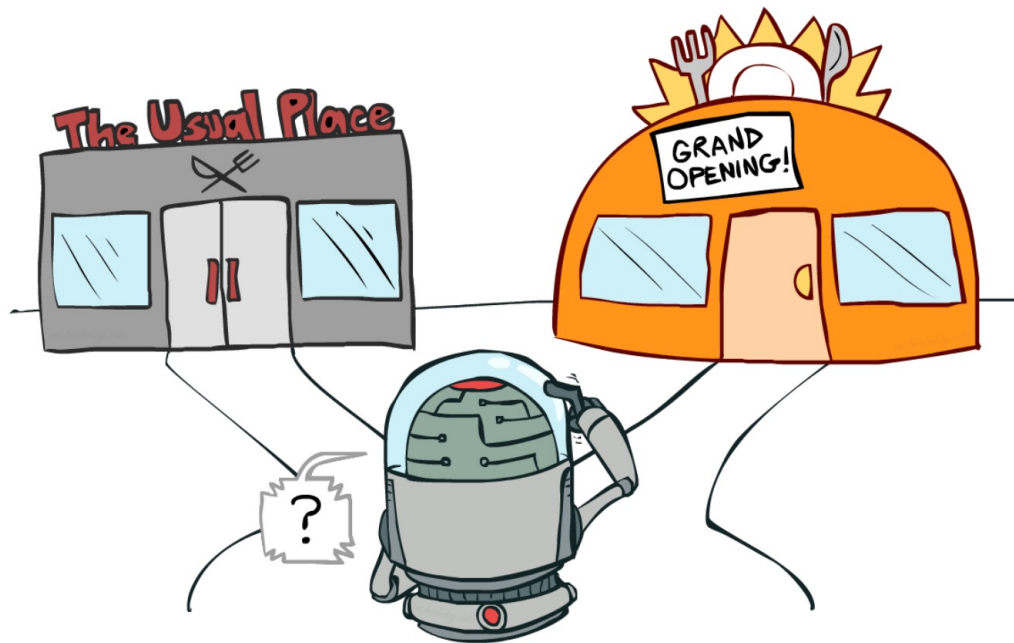
课程大纲

- 探索与利用
- 蒙特卡洛树搜索
- 深度Q网络
- DQN改进算法

探索与利用

已知局部最优解 vs 潜在最优解？

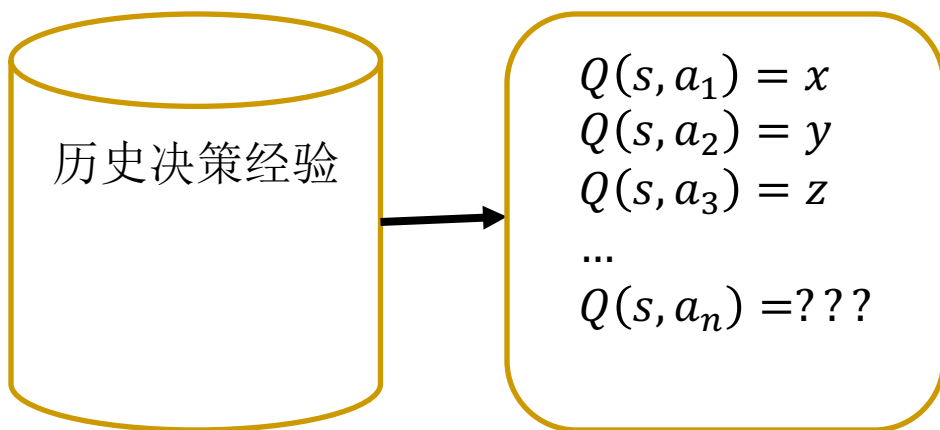
- 探索：承担潜在风险决策寻求可能更好的结果
- 利用：根据已有统计结果做出决策，陷入局部最优



探索与利用

为什么会出现探索与利用的困境？

- 环境信息不完全
- 每一种决策的真实价值无法获取，只能获取其统计价值



- 有些行为尚未被探索过，其价值未知；
- 已被探索过的行为可能因为偶然概率原因被高估或者低估；

探索与利用

- 合理的探索策略很难提出；如何提出一个最优的探索策略？
- 如何定义最优？
- 悔值最小化？贝叶斯优化？

Multi-arm Bandit
(无状态、单步、
强化学习)

Contextual Bandit
(单步强化学习)

小型的有限MDP

大型的、无限的
MDP、连续空间、
连续动作

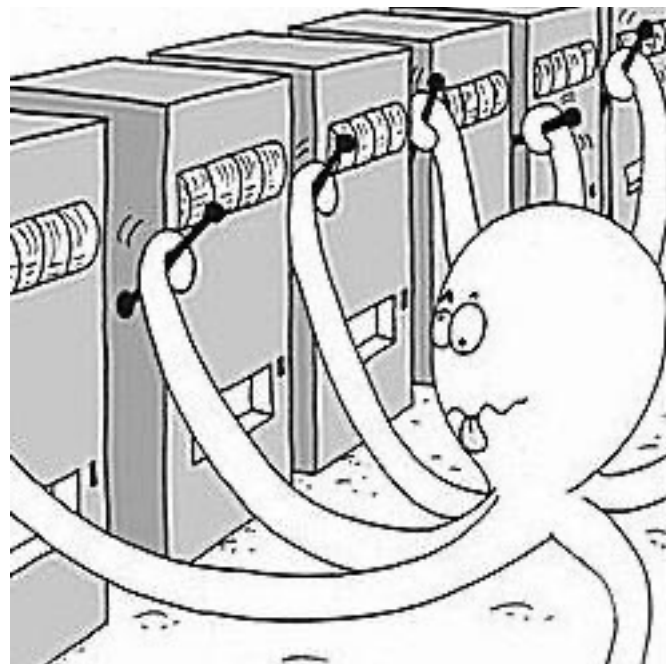


理论易证明

理论难证明

Multi-arm Bandit

- 假设你在一家赌场里，手中有有限的筹码，面对多台赌博机。其中每台赌博机都有预设的但是未知的奖励概率与分布。那么，获得最高长期回报的最佳策略是什么？
- 这里我们先讨论无限次实验的情况。
- 一种简单的方法是在每一台机器上一直尝试；然后根据大数定律估计出“真实”的奖励概率；
- 但这种做法相非常浪费，无法保证最好的长期回报。



Multi-arm Bandit

一个伯努利MAB问题可以被定义为：

- 有K个具有奖励概率的机器： $\{\theta_1, \theta_2, \dots, \theta_k\}$
- 在每个时间步 t ，在某一台赌博机上采取行动 a ，并获取奖励 r 。
- A 是动作集合，每个动作指对一台赌博机互动；动作 a 的值是预期奖励，即 $Q(a) = \mathbb{E}[r|a] = \theta$ 。如果行为 a_t 在时间步 t 中针对机器 i 进行操作，则 $Q(a_t) = \theta_i$ 。
- R 为奖励函数，在伯努利MAB问题中，奖励 r 以概率方式被观察到。在时间步 t 中， $r_t = R(a_t)$ 以 $Q(a_t)$ 的概率返回1，否则返回0。

Multi-arm Bandit

一个伯努利MAB问题可以被定义为：

- 有 K 个具有奖励概率的机器： $\{\theta_1, \theta_2, \dots, \theta_k\}$
- 在每个时间步 t ，在某一台赌博机上采取行动 a ，并获取奖励 r 。
- A 是动作集合，每个动作指对一台赌博机互动；动作 a 的值是预期奖励，即 $Q(a) = \mathbb{E}[r|a] = \theta$ 。如果行为 a_t 在时间步 t 中针对机器 i 进行操作，则 $Q(a_t) = \theta_i$ 。
- R 为奖励函数，在伯努利MAB问题中，奖励 r 以概率方式被观察到。在时间步 t 中， $r_t = R(a_t)$ 以 $Q(a_t)$ 的概率返回1，否则返回0。

MAB问题是一个没有状态的简化版的MDP

Multi-arm Bandit

- 解决MAB问题的目标是最大化累计回报，即 $\sum_{t=1}^T r_t$ 。
- 假如能获取最大奖励的最优选择是已知的，那么MAB问题的目标也可以转化为悔值的最小化问题，即没有选择最优行为所带来的损失。

- 因此最优行为对应的最优奖励概率表示为：

$$\theta^* = Q(a^*) = \max_{a \in A} Q(a) = \max_{1 \leq i \leq K} \theta_i$$

- 损失函数即累积悔值为：

$$\mathcal{L}_T = \mathbb{E}\left[\sum_{t=1}^T (\theta^* - Q(a_t))\right]$$

ϵ -Greedy 算法

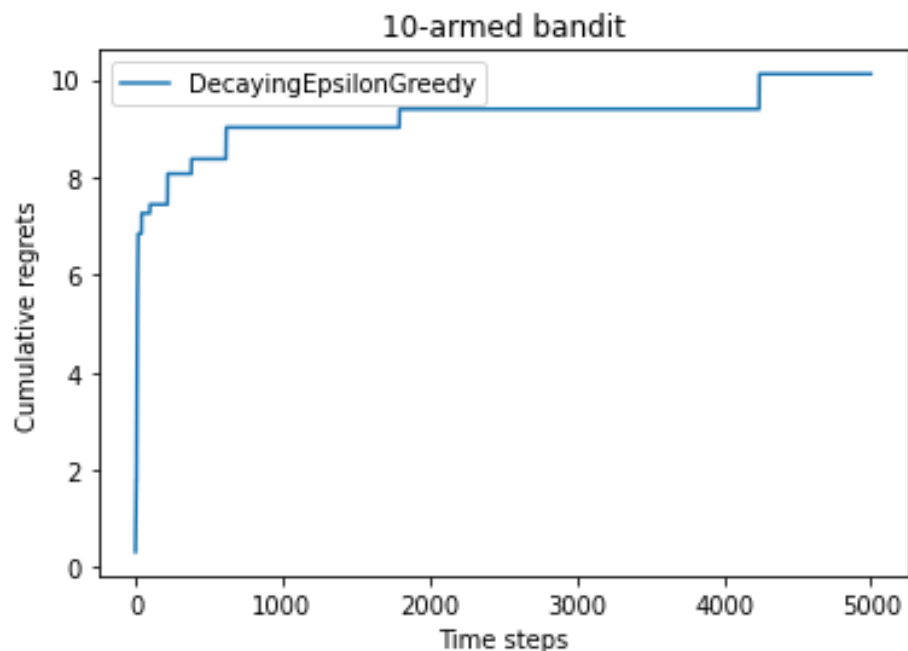
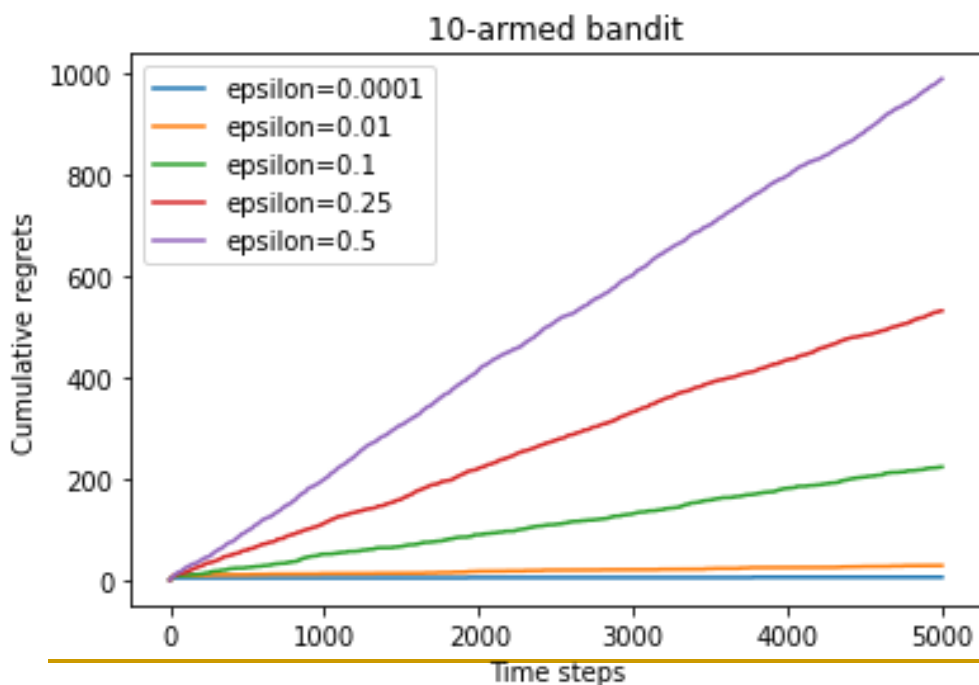
- ϵ -greedy策略：大多数时间（概率 $1 - \epsilon$ ）采用当前统计的最优的行为，其余时间随机选择一个行为（概率 ϵ ）。
- 行为价值可以通过求取历史决策经验中该行为获得的奖励的均值来获取：

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^t r_{\tau} \delta[a_{\tau} = a]$$

- 其中 δ 为二元指示函数； $N_t(a)$ 为到目前为止行为 a_t 被采取次数，即 $N_t(a) = \sum_{\tau=1}^t \delta[a_{\tau} = a]$
- ϵ 值可以被设定为超参，也可以随训练的进行逐步衰减。
- 优势：足够简单

ϵ -Greedy算法

- 固定的 ϵ 值探索会导致累计悔值线性增大；
- 随训练时间衰减的 ϵ 可以令累计悔值呈现次线性增大趋势。



UCB算法

- 虽然 ϵ -greedy策略可以探索其他潜在最优行为，但是其随机性忽略了训练过程中的已有前置信息。即虽然每个行为都有可能是最优行为，但是其成为最优的可能性是不同的，同理选择具备不同潜力的行为的策略也可以不完全随机。
- UCB算法通过奖励值的上置信界（Upper Confidence Bound）来衡量每一个动作附加其“潜力”后的价值。行为的真实价值低于附加潜力后的价值即： $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ 。
- 上界函数 $\hat{U}_t(a)$ 与 $N_t(a)$ 相关。因为大的行为访问次数使得对应的行为的价值更准确，因此大行为访问次数将会得到较小的置信上界。
- 上置信界怎么评估？

Hoeffding不等式

- 假设 $\{X_1, \dots, X_n\}$ 为独立同分布的取值 $[0,1]$ 的随机变量，从该变量中采样得到样本，其样本均值为 $\bar{X}_t = \frac{1}{t} \sum_{\tau=1}^t X_\tau$ ，那么对于给定 $u \geq 0$ ：

$$\mathbb{P}[\mathbb{E}[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- 如果替换：
 - $r_t(a)$ 是上述的随机变量；
 - $Q(a)$ 是真实期望值；
 - $\hat{Q}_t(a)$ 是样本均值；
 - $U_t(a)$ 为上置信界；

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2tU_t(a)^2}$$

UCB算法续

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2tU_t(a)^2}$$

- 令 $p = e^{-2tU_t(a)^2}$, 则:

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- 一种启发方法是及时降低 p 的阈值, 因为在观察到更多的奖励样本的前提下可以得到更有把握的边界估计。这里我们设定 $p = t^{-4}$, 则:

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

$$a_t^{UCB1} = \arg \max_{a \in A} (Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}})$$

- 更一般地:

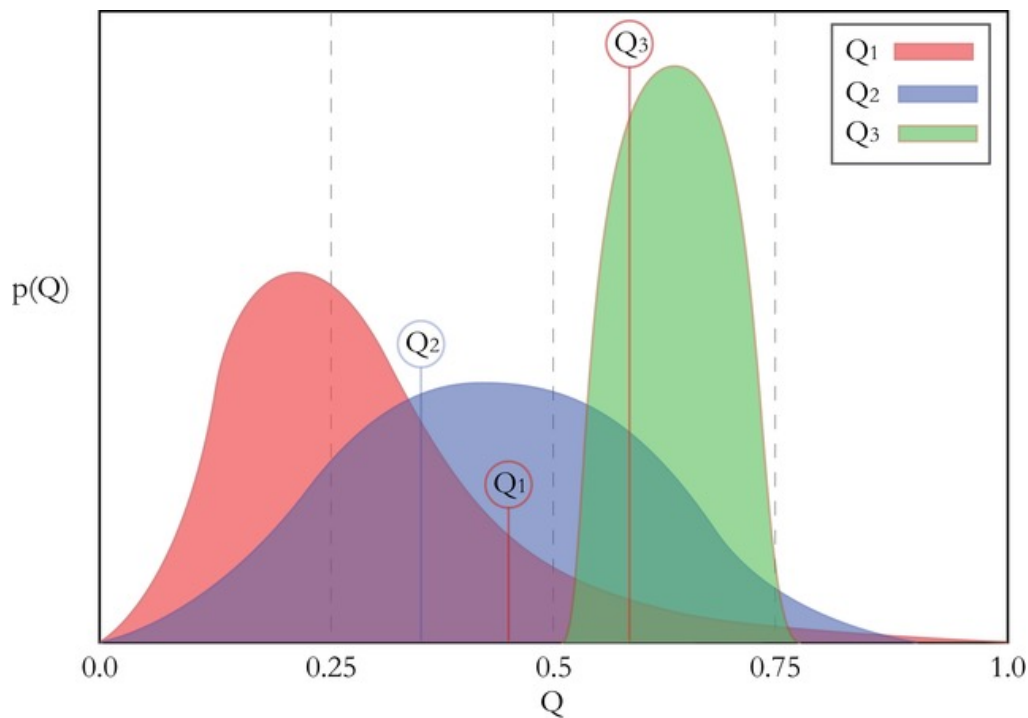
$$U_t(a) = \sqrt{\frac{c \log t}{N_t(a)}}$$

汤普森采样

- 先假设拉动每根拉杆的奖励服从一个特定的概率分布，然后根据拉动每根拉杆的期望奖励来进行选择。
- 由于计算所有拉杆的期望奖励的代价比较高，汤普森采样算法使用采样的方式，即根据当前每个动作 a 的奖励概率分布进行一轮采样，得到一组每根拉杆的奖励样本，再选择样本中奖励最大的动作。
- 因此，汤普森采样是一种计算所有拉杆的最高奖励概率的蒙特卡洛采样方法。

汤普森采样

- 通常用 Beta 分布对当前每个动作的奖励概率分布进行建模。
- 若某拉杆被选择了 k 次，其中 m_1 次奖励为 1， m_2 次奖励为 0，则该拉杆的奖励服从参数为 $(m_1 + 1, m_2 + 1)$ 的 Beta 分布。



其他探索相关论文

- Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers.
- Stadie, B. C., Levine, S., & Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.
- Osband, I., Blundell, C., Pritzel, A., & Van Roy, B. (2016). Deep exploration via bootstrapped DQN. *Advances in neural information processing systems*, 29.
- Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). **Exploration by random network distillation**. *arXiv preprint arXiv:1810.12894*.
- Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29.
- Tang, H., Houthoofd, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., ... & Abbeel, P. (2017). # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30.
- Fu, J., Co-Reyes, J., & Levine, S. (2017). Ex2: Exploration with exemplar models for deep reinforcement learning. *Advances in neural information processing systems*, 30.

课程大纲

- 探索与利用
- **蒙特卡洛树搜索**
- 深度Q网络
- **DQN改进算法**

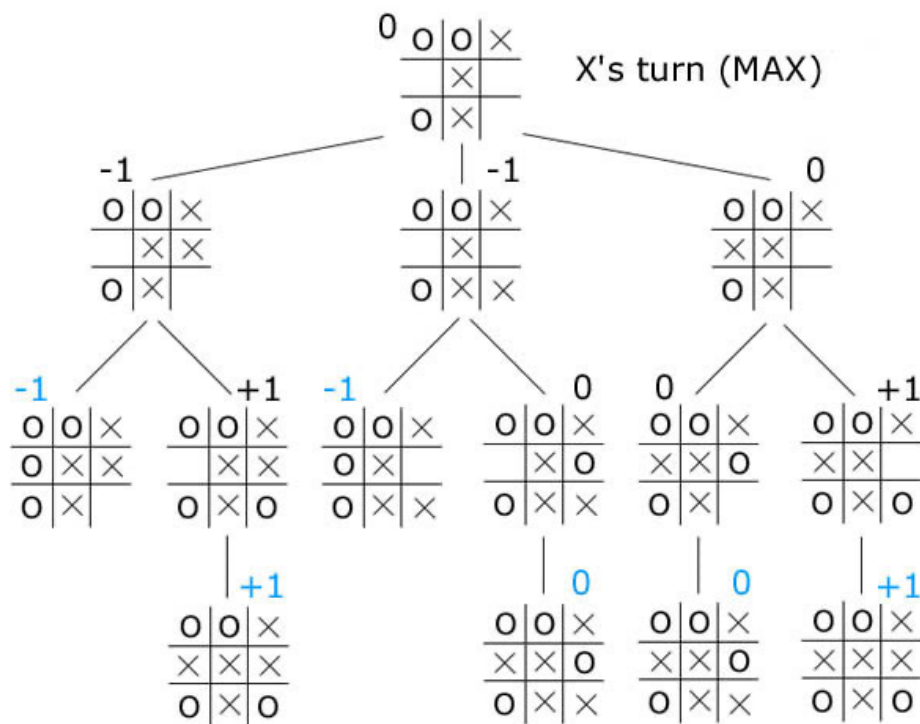
博弈

- 双人、多人博弈中的决策



博弈树

- 枚举所有可能出现的博弈场景，使得在每一步中最小化对手分数同时最大化自己的分数（minimax算法）；
- 如果资源足够，通过博弈树和minimax算法，最优行为可以被直接规划得到；
- 适合在有限博弈步数、小动作空间的序列决策问题中；
- 但是，资源足够？

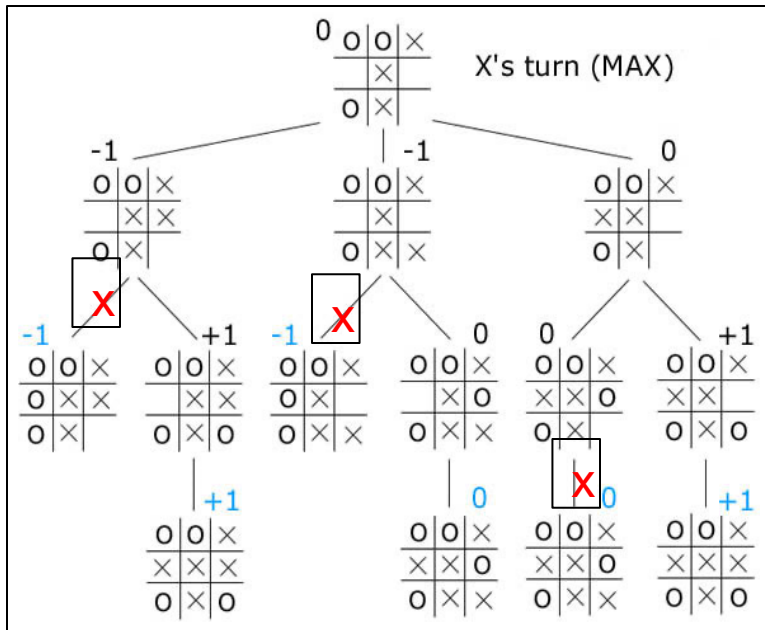


博弈树

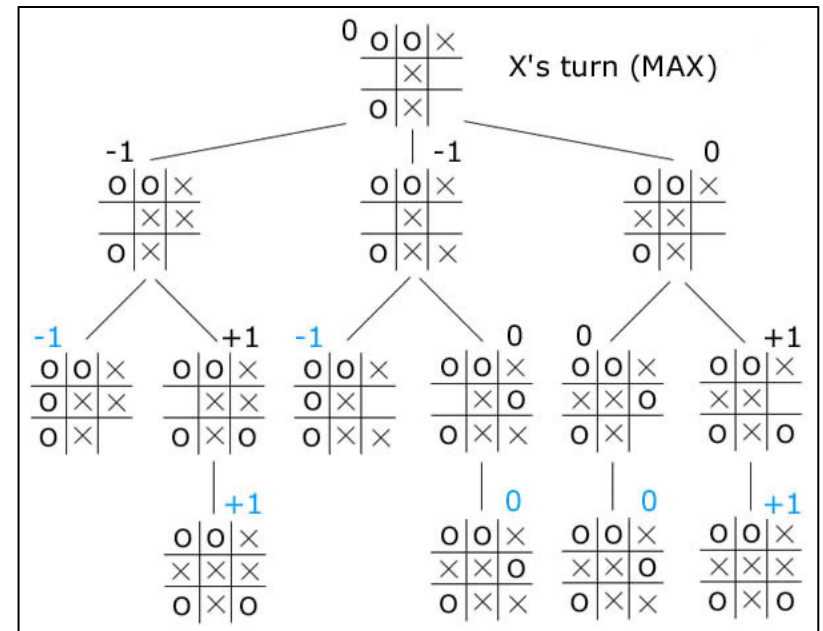
- 随着博弈步数的增大（博弈树加深），博弈树的节点指数级增长，即：其复杂度大约为： $O(b^d)$ 。
- 其中： b 为分枝系数（动作空间）， d 为博弈树的深度。
 - 围棋：大概 10^{170} 个节点
 - 国际象棋：超过 10^{40} 个节点
- 因此，几乎没有可能通过暴力搜索得到最优博弈路径。

潜在解决办法

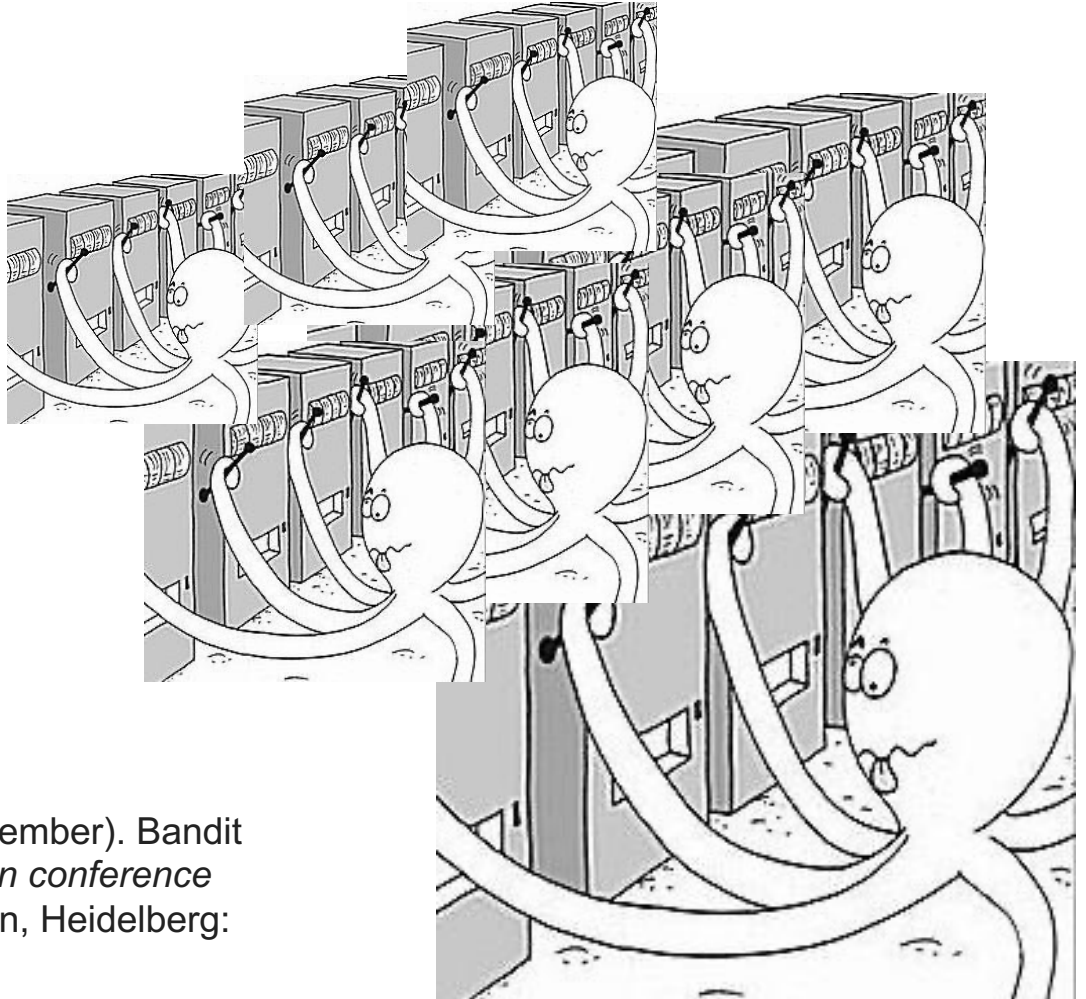
- 减小分枝系数（剪枝）
- 仅探索行为空间的一部分子空间，剪掉明显效果差的枝干



- 减小搜索深度
- 仅探索到固定深度的博弈树，并用一些价值函数评估该深度的节点



- 博弈树展开后计算每个节点的价值的时候采用UCB的计算

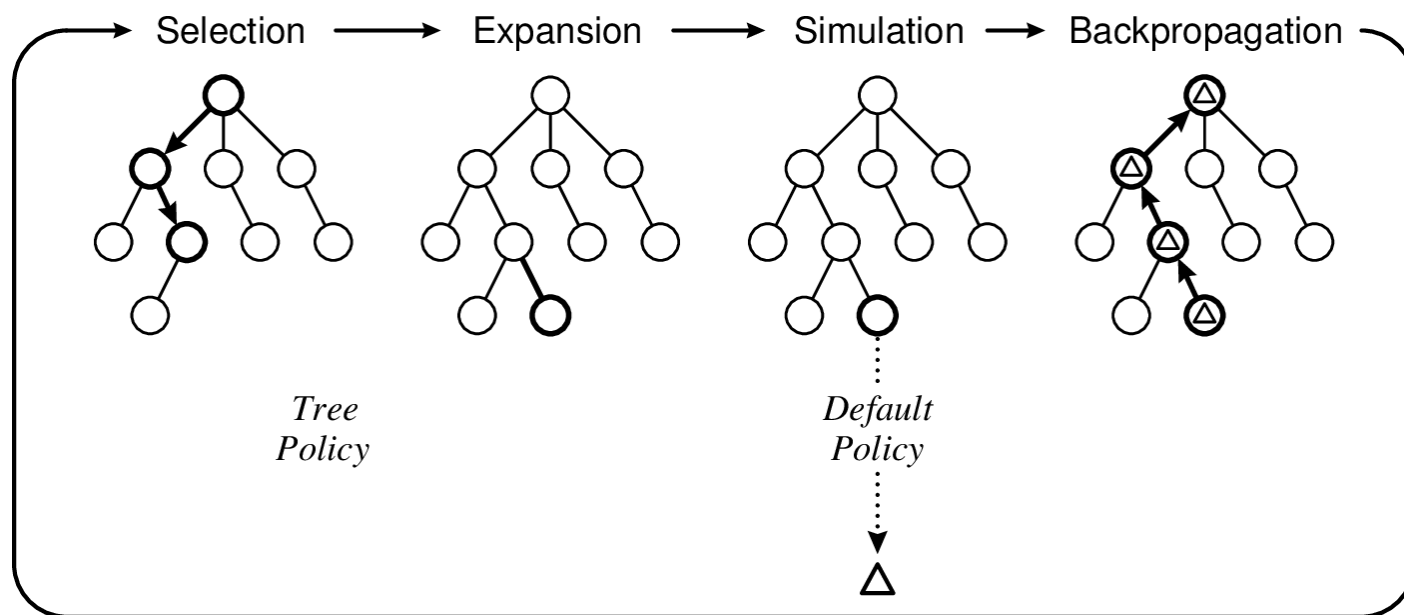


Kocsis, L., & Szepesvári, C. (2006, September). Bandit based monte-carlo planning. In *European conference on machine learning* (pp. 282-293). Berlin, Heidelberg: Springer Berlin Heidelberg.

- 蒙特卡洛树搜索：Monte-Carlo Tree Search(MCTS)
 - 该术语于2006年被提出，但是其概念至少可以追溯到1987年；
 - 维护一个已探索的博弈状态树；
 - 记录博弈树中每一个状态的平均奖励和访问次数；
 - 核心思路：不需要手动设计一个关于状态的启发函数，而是从该状态开始随机模拟一局博弈获得胜负信息。
 - 增加UCB计算可以给状态更好的预估价值。

MCTS迭代

- 选择
- 扩展
- 模拟
- 回传



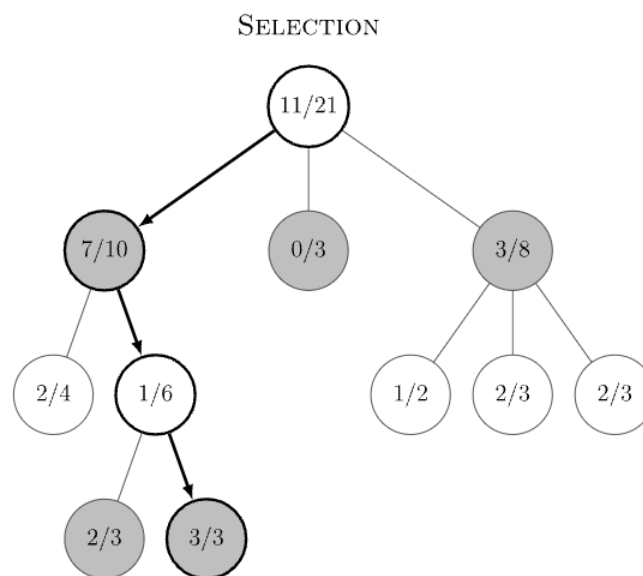
MCTS迭代

- 选择过程：递归选择最大UCB值的子节点

$$\frac{1}{n_i} \sum_{t=1}^{n_i} r_t + c \sqrt{\frac{\log N}{n_i}}$$

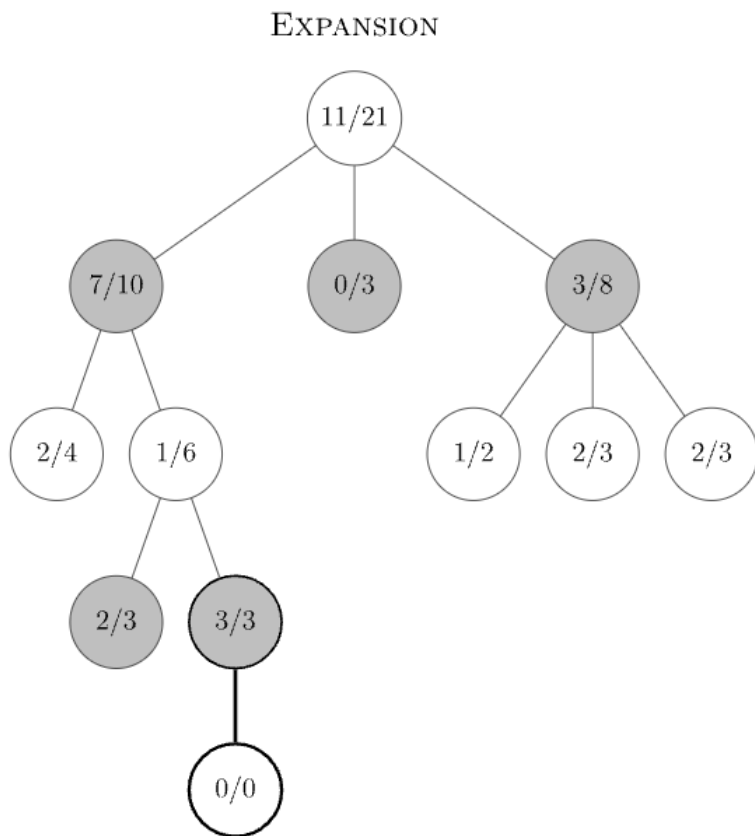
其中：

- N ：父节点被访问次数
- n_i ：子节点被访问次数
- r_t ：第 t 次从父节点到子节点的奖励
- c ：平衡探索超参



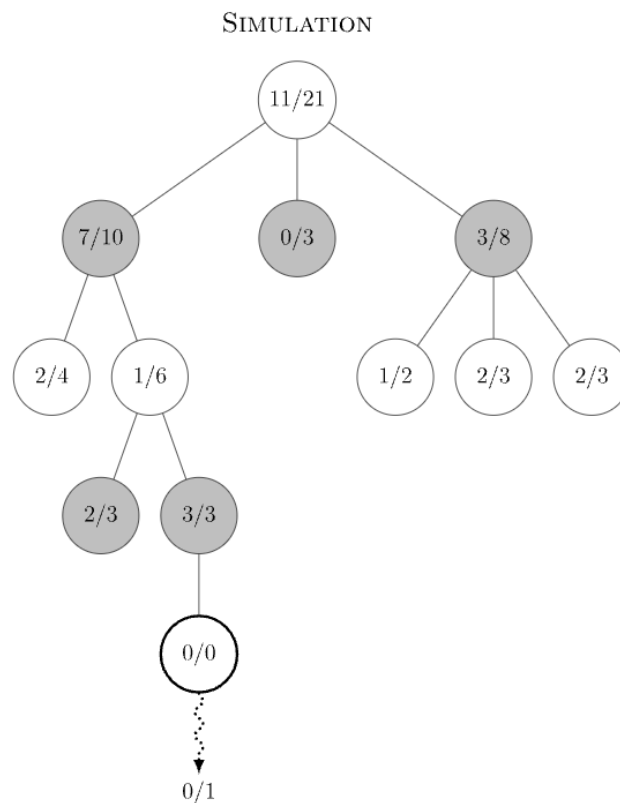
MCTS迭代

- 扩展过程：为子节点扩充未访问的节点并加进蒙特卡洛树中



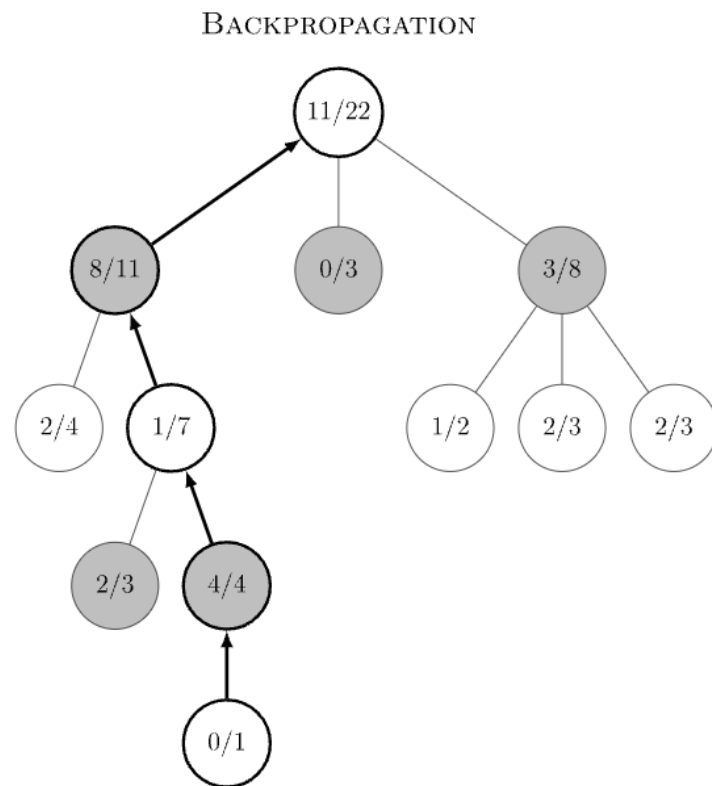
MCTS迭代

- 模拟过程：从新扩展的节点开始，采用固定的策略进行博弈模拟（rollout），直到博弈结束。
- rollout过程采用任何策略皆可



MCTS迭代

- 回传过程：将模拟的结果递归回传到路径经过的所有节点直到根节点。
- 一般更新节点价值和节点访问次数
- 如果预存节点UCB值，则更新该路径上所有节点的子节点的UCB值。



MCTS

MCTS优势：

- 在无专家知识的时候很有效；
- 节点大量模拟后可以逼近真实节点价值；
- MCTS过程可以并行化，例如可以在扩展的节点上进行多进程并行模拟；

MCTS劣势：

- MCTS的计算结果有较大的方差；
 - 模拟过程的策略对搜索效率有较大影响；
- ➔
- 学习更好的状态评估方法
 - 学习更“聪明”的评估策略

搜索问题

Initial State

1	2	3
8		4
7	6	5

Goal State

2	8	1
	4	3
7	6	5



启发式搜索

- 核心思想：

$$F(s) = G(s) + H(s)$$

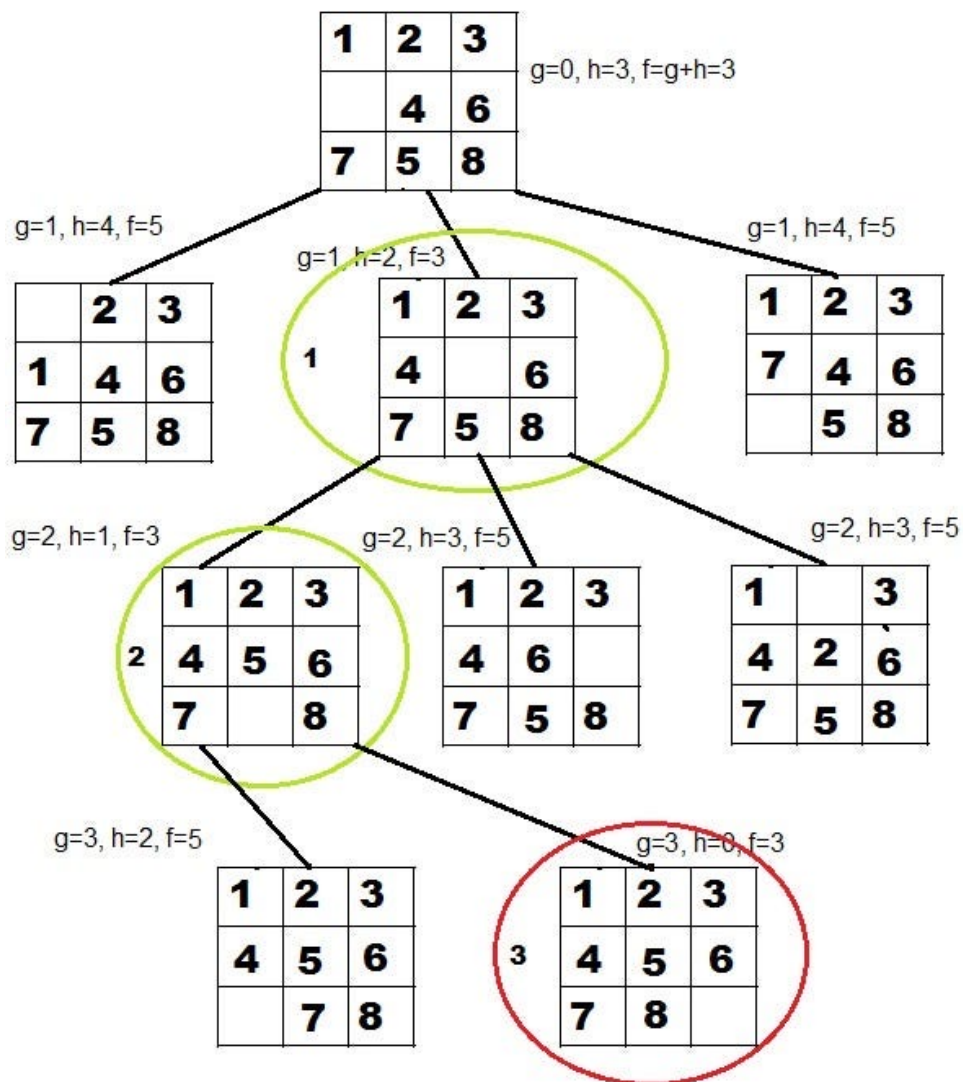
- $G(s)$ ：从初始状态到 s 状态的序列累计奖励；
- $H(s)$ ：启发函数，从 s 状态到最终状态的预估价值；
- 根据 $F(s)$ 的值进行直接最大化行为选择或最大化UCB值；
- 需要人为设置启发函数；

启发式搜索

- 可容许性admissible: 一个具备可容许性的启发式函数永远不会高估到达某个目标的代价，即价值不会过高估计。
- 一致性consistency: 假设节点 s 的后续节点是 s' ，则从 s 到目标节点 s_T 之间的开销代价一定小于从 s 到 s' 的开销 $c(s, a, s')$ 再加上从 s' 到目标节点之间的开销，即 $h(s) \leq c(s, a, s') + h(s')$ 。
- 因此，人为设定的启发函数可以不需要与 $V(s)$ 相等。

启发式搜索

- 曼哈顿距离
- 直线距离
- 最小生成树
- 松弛问题的解
- ...



- 学习更好的状态评估方法 $V(s)$:
 - 给定任意博弈状态，预测计算状态价值；
 - 可替代MCTS中的模拟过程；
- 学习更“聪明”的评估策略 $\pi(a|s)$:
 - 给定状态的行为概率分布；
 - 令搜索策略偏向更优的行为选择；
 - 可以替代rollout过程中的随机策略；
- 在每轮博弈后或博弈期间更新策略和价值网络。

课程大纲

- 探索与利用
- 蒙特卡洛树搜索
- 深度Q网络
- DQN改进算法

回顾Q-learning

- 表格式Q学习：如果状态是无法枚举的？例如连续空间？
 - 一种方式可以是离散化连续空间，将连续空间归纳为多个区间，然后转换为表格式的区间-行为价值计算。
 - 需要太多的先验离散化知识。

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a))$$

- Q学习以时序差分的形式，以学习目标 $r + \gamma \max_{a' \in A} Q(s', a')$ 来增量式更新 $Q(s, a)$ ，即令 $Q(s, a)$ 贴近其TD目标。
- 如果我们将表格式的 $Q(s, a)$ 的取值用神经网络代替，且该网络以状态+行为作为输入，以该状态行为价值作为输出，那么Q-learning算法就可以直接扩展为深度Q学习。

深度Q学习

以学习目标 $r + \gamma \max_{a' \in A} Q(s', a')$ 来增量式更新 $Q(s, a)$ ，即令 $Q(s, a)$ 贴近其TD目标。

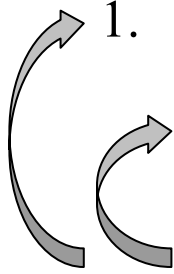
- 状态价值网络： $Q_{\omega}(s, a)$
- 时序差分目标结果： $r + \gamma \max_{a'} Q_{\omega}(s', a')$

给定一组状态转移数据： $\{(s_i, a_i, r_i, s'_i)\}$ ：深度Q网络的损失函数构造成为均方误差形式：

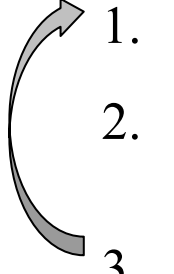
$$\omega^* = \operatorname{argmin}_{\omega} \frac{1}{2N} \sum_{i=1}^N \left[Q_{\omega}(s_i, a_i) - \left(r_i + \gamma \max_{a'} Q_{\omega}(s'_i, a') \right) \right]^2$$

深度Q学习

Fitted Q值迭代:

- 
1. 通过某些策略收集状态转移数据集 $\{(s_i, a_i, r_i, s'_i)\}$;
 2. 计算TD目标: $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_\omega(s'_i, a'_i)$
 3. 更新网络参数: $\omega \leftarrow \operatorname{argmin}_\omega \frac{1}{2} \sum_i \|Q_\omega(s_i, a_i) - y_i\|^2$


在线Q值迭代算法:

- 
1. 环境中根据策略采取行为 a_i , 并从环境获取 (s_i, a_i, r_i, s'_i) ;
 2. 计算TD目标: $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_\omega(s'_i, a'_i)$;
 3. 更新网络参数: $\omega \leftarrow \omega - \alpha \frac{dQ_\omega}{d\omega}(s_i, a_i)(Q_\omega(s_i, a_i) - y_i)$

深度Q学习

但此时版本深度Q学习有一些问题：

在线Q值迭代算法：

- 
1. 环境中根据策略采取行为 a_i ，并从环境获取 (s_i, a_i, r_i, s'_i) ；
 2. 计算TD目标： $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_\omega(s'_i, a'_i)$ ；
 3. 更新网络参数： $\omega \leftarrow \omega - \alpha \frac{dQ_\omega}{d\omega}(s_i, a_i)(Q_\omega(s_i, a_i) - y_i)$

- 神经网络训练需要独立同分布数据，但是状态转移数据强相关；
- 更新神经网络参数并不是梯度下降， y_i 的计算也更新梯度；
- Q值的更新不稳定

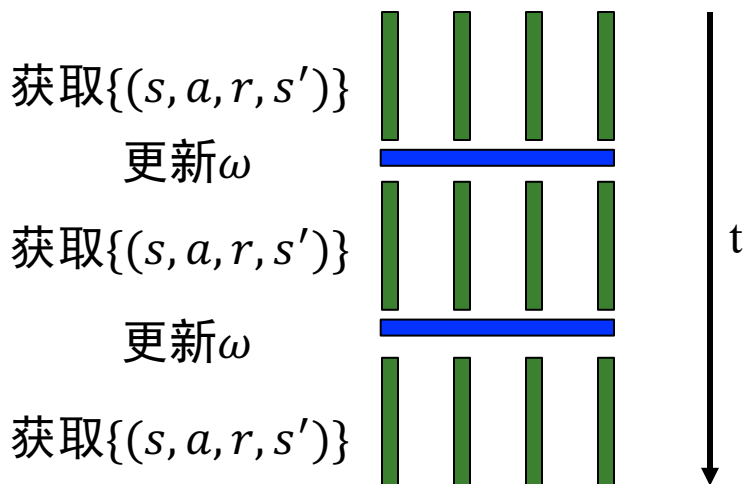
数据相关性问题

在线Q值迭代算法：

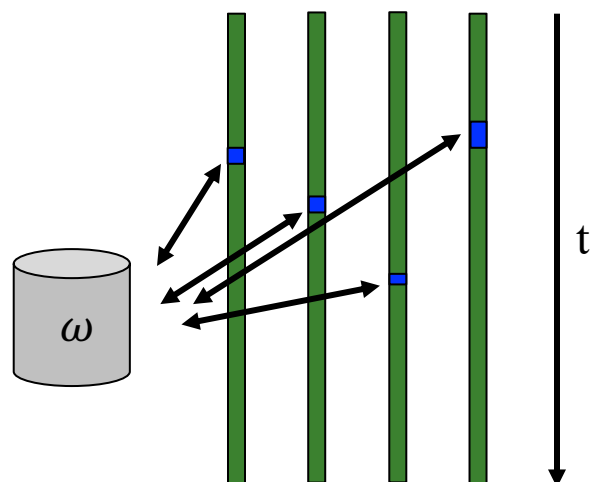
1. 环境中根据策略采取行为 a_i ，并从环境获取 (s_i, a_i, r_i, s'_i) ；
2. 更新网络参数： $\omega \leftarrow \omega - \alpha \frac{dQ_\omega}{d\omega}(s_i, a_i)(Q_\omega(s_i, a_i) - [r_i + \gamma \max_{a'_i} Q_\omega(s'_i, a'_i)])$

通过一些措施解决数据的相关性：并行Q-learning，用其他进程的状态转移更新：

同步并行Q-learning



异步并行Q-learning

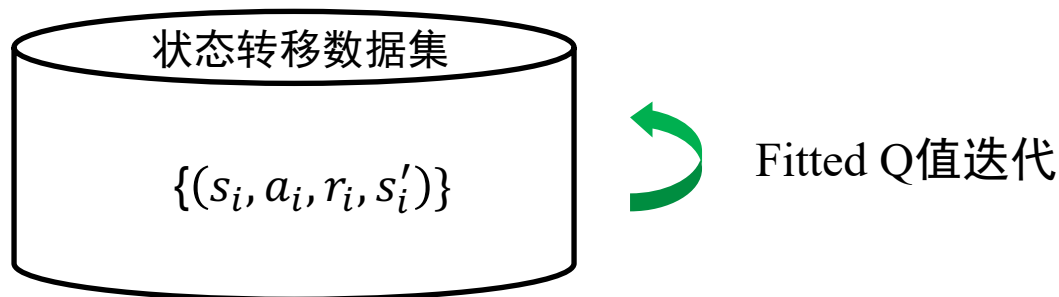


经验回放缓存

Fitted Q值迭代:

- ~~1. 通过某些策略收集状态转移数据集 $\{(s_i, a_i, r_i, s'_i)\}$;~~
2. 计算TD目标: $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_\omega(s'_i, a'_i)$
3. 更新网络参数: $\omega \leftarrow \operatorname{argmin}_\omega \frac{1}{2} \sum_i ||Q_\omega(s_i, a_i) - y_i||^2$

- 不通过某些策略收集数据，而是从经验回放缓存中采样；
- 经验回放缓存中数据可以来源于任何策略；



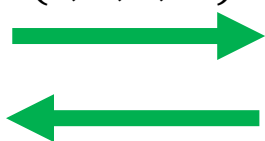
经验回放缓存

结合经验缓存的深度Q学习：

1. 通过当前策略与环境交互，收集状态转移数据 (s_i, a_i, r_i, s'_i) 并将数据存入经验缓存 B 中；
2. 从经验缓存 B 中采样得到数据集 $\{(s_i, a_i, r_i, s'_i)\}$
3. 计算TD目标：
$$y_i \leftarrow r_i + \gamma \max_{a'_i} Q_{\omega}(s'_i, a'_i)$$
4. 更新网络参数：
$$\omega \leftarrow \operatorname{argmin}_{\omega} \frac{1}{2} \sum_i \|Q_{\omega}(s_i, a_i) - y_i\|^2$$



(s, a, r, s')



$\pi(a|s)(\epsilon - greedy)$

状态转移数据集

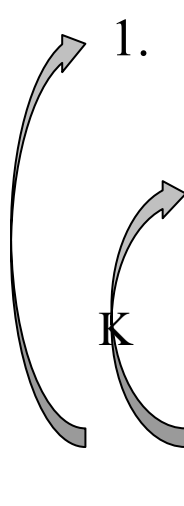
$\{(s_i, a_i, r_i, s'_i)\}$



Fitted Q值迭代

目标网络

结合经验缓存的深度Q学习：

- 
1. 通过当前策略与环境交互，收集状态转移数据 (s_i, a_i, r_i, s'_i) 并将数据存入经验缓存 B 中；
 2. 从经验缓存 B 中采样得到数据集 $\{(s_i, a_i, r_i, s'_i)\}$
 3. 计算TD目标： $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_\omega(s'_i, a'_i)$
 4. 更新网络参数： $\omega \leftarrow \operatorname{argmin}_\omega \frac{1}{2} \sum_i \|Q_\omega(s_i, a_i) - y_i\|^2$

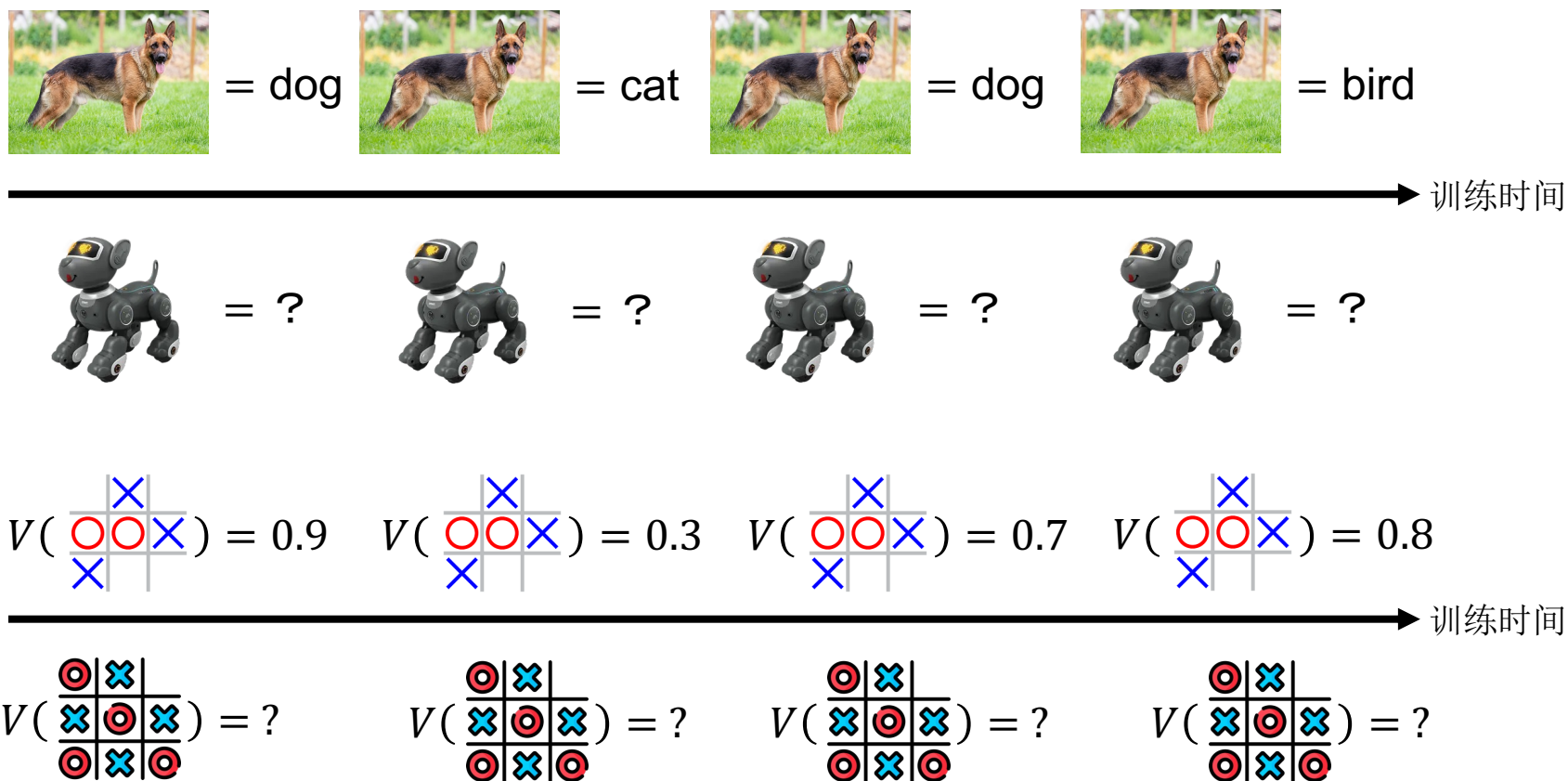
经验回放缓存

- ~~神经网络训练需要独立同分布数据，但是状态转移数据强相关，~~
- 更新神经网络参数并不是梯度下降， y_i 的计算也更新梯度；
- Q值的更新不稳定

目标网络

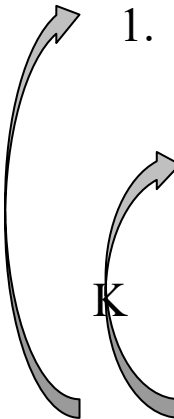
目标网络

深度Q学习更新不稳定原因：对于同样的状态转移数据，短时间内同样的输入得到不同的TD目标作为监督信号。



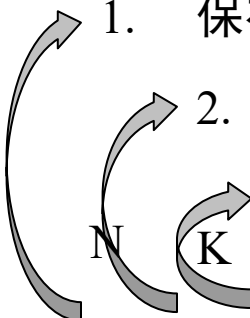
目标网络

深度Q学习更新不稳定原因：对于同样的状态转移数据，短时间内同样的输入得到不同的TD目标作为监督信号。


- 
1. 通过当前策略与环境交互，收集状态转移数据 (s_i, a_i, r_i, s'_i) 并将数据存入经验缓存 B 中；
2. 从经验缓存 B 中采样得到数据集 $\{(s_i, a_i, r_i, s'_i)\}$
3. 计算TD目标： $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_{\omega-}(s'_i, a'_i)$ **令 y_i 阶段性保持不变且不参与梯度回传**
4. 更新网络参数： $\omega \leftarrow \operatorname{argmin}_{\omega} \frac{1}{2} \sum_i ||Q_{\omega}(s_i, a_i) - y_i||^2$
- K

经典深度Q学习 (DQN)

带有经验缓存和目标网络的深度Q学习：


- 
1. 保存目标网络的网络参数： $\omega^- \leftarrow \omega$;
 2. 利用某些策略收集 $\{(s_i, a_i, r_i, s'_i)\}$ 并加入到缓存 B 中;
 3. 从缓存 B 中采样数据
 4. 更新网络参数 $\omega \leftarrow \omega - \alpha \frac{dQ_\omega}{d\omega}(s_i, a_i)(Q_\omega(s_i, a_i) - [r_i + \gamma \max_{a'_i} Q_{\omega^-}(s'_i, a'_i)])$

经典深度Q学习：

- 
1. 基于当前策略采取行为 a_i ，观测 (s_i, a_i, r_i, s'_i) 并加入到缓存 B ;
 2. 均匀从缓存 B 中采样得到小组数据 $\{(s_j, a_j, r_j, s'_j)\}$;
 3. 利用目标 Q_{ω^-} -网络计算 $y_j \leftarrow r_j + \gamma \max_{a'_j} Q_{\omega^-}(s'_j, a'_j)$;
 4. $\omega \leftarrow \omega - \alpha \frac{dQ_\omega}{d\omega}(s_j, a_j)(Q_\omega(s_j, a_j) - y_j)$
 5. 更新目标网络参数：每 N 个步骤 $\omega^- \leftarrow \omega$

经典深度Q学习 (DQN)

经典深度Q学习：

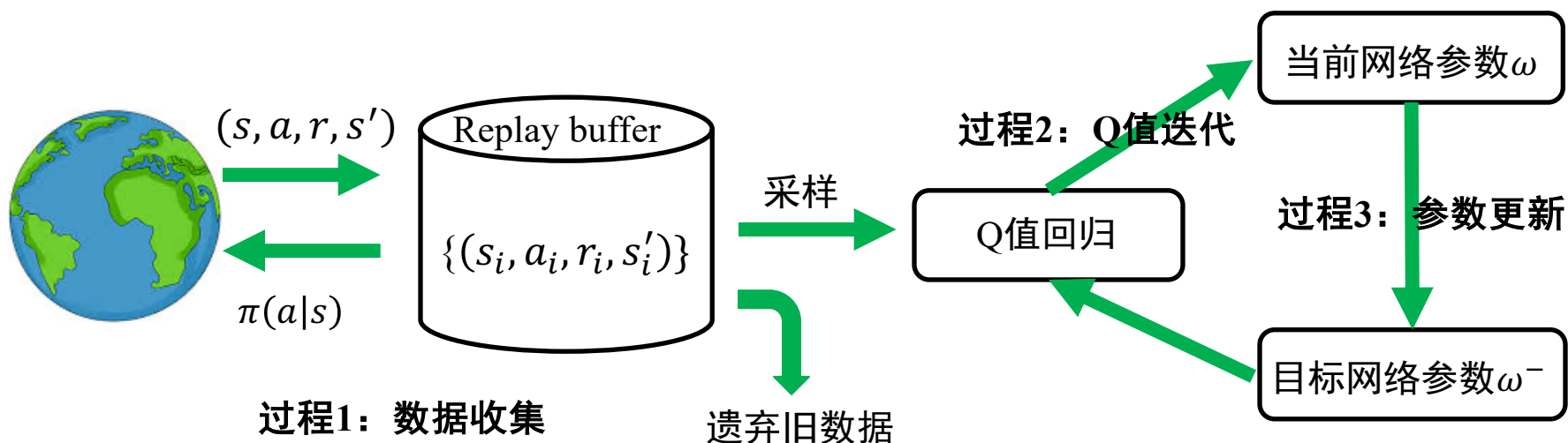
- 
1. 基于当前策略采取行为 a_i ，观测 (s_i, a_i, r_i, s'_i) 并加入到缓存 B ；
 2. 均匀从缓存 B 中采样得到小组数据 $\{(s_j, a_j, r_j, s'_j)\}$ ；
 3. 利用目标 Q_{ω^-} -网络计算 $y_j \leftarrow r_j + \gamma \max_{a'_j} Q_{\omega^-}(s'_j, a'_j)$ ；
 4. $\omega \leftarrow \omega - \alpha \frac{dQ_{\omega}}{d\omega}(s_j, a_j)(Q_{\omega}(s_j, a_j) - y_j)$ ；
 5. 更新目标网络参数 ω^- ；

- 更新目标网络参数可以采用滑动平均的方法：即 $\omega^- \leftarrow \tau \omega^- + (1 - \tau)\omega, \tau = 0.99$
- 这种方法在为新的 ω^- 赋值时可以保留旧有的 ω 的影响；

经典深度Q学习 (DQN)

经典深度Q学习：

1. 基于当前策略采取行为 a_i ，观测 (s_i, a_i, r_i, s'_i) 并加入到缓存 B ；
2. 均匀从缓存 B 中采样得到小组数据 $\{(s_j, a_j, r_j, s'_j)\}$ ；
3. 利用目标 Q_{ω^-} -网络计算 $y_j \leftarrow r_j + \gamma \max_{a'_j} Q_{\omega^-}(s'_j, a'_j)$ ；
4. $\omega \leftarrow \omega - \alpha \frac{dQ_{\omega}}{d\omega}(s_j, a_j)(Q_{\omega}(s_j, a_j) - y_j)$ ；
5. 更新目标网络参数 ω^- ；



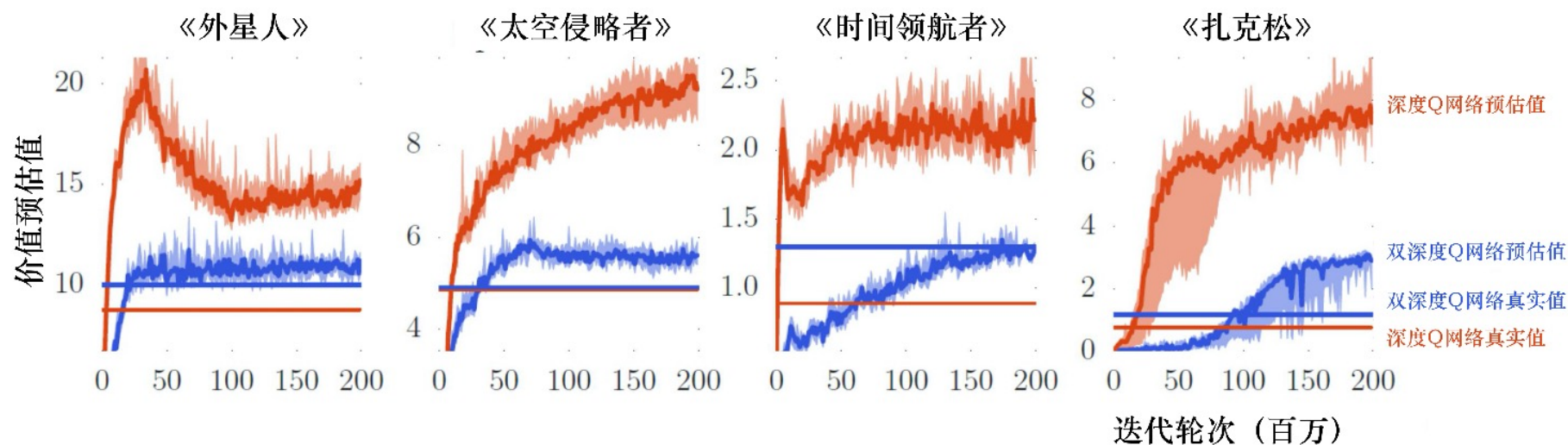
经典深度Q学习 (DQN)



课程大纲

- 探索与利用
- 蒙特卡洛树搜索
- 深度Q网络
- **DQN改进算法**

Q值过高估计



- 上图将DQN和DDQN应用在Atari环境上，得到的真实累计奖励与Q网络输出的预测奖励的对比。
- DQN中的Q网络容易过高估计所得的回报，这也是基于自举（Boostrapping）方法的价值估计的共有问题

Q值过高估计

为什么会出现过高估计的问题

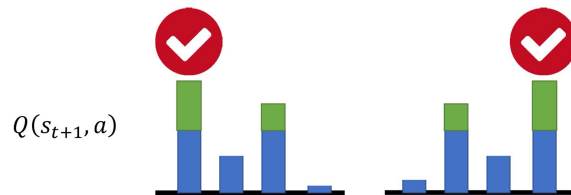
- 一个数学直觉：最大值的期望大于期望的最大值，即假设两个随机变量 X_1, X_2 ：

$$\mathbb{E}[\max(X_1, X_2)] \geq \max(\mathbb{E}[X_1], \mathbb{E}[X_2])$$

- 假设Q值网络的状态-动作价值估计含有随机噪音 $X_1, X_2 \sim N(0, \epsilon)$ ：

$$\begin{aligned} & \mathbb{E}_X[\max(Q(s, a_1) + X_1, Q(s, a_2) + X_2)] \\ & \geq \max(\mathbb{E}_X[Q(s, a_1) + X_1], \mathbb{E}_X[Q(s, a_2) + X_2]) = \max(Q(s, a_1), Q(s, a_2)) \end{aligned}$$

- 算法总是选择价值最高的动作作为目标进行更新，因此存在被高估的问题。



Double DQN

选择动作和计算价值不使用同一个网络

- DQN在通过目标网络计算目标值的时候，基于 Q_{ω^-} -选择最优动作，再使用 Q_{ω^-} -计算目标值：

$$\max_{a'} Q_{\omega^-}(s', a') = Q_{\omega^-}(s', \arg \max_{a'} Q_{\omega^-}(s', a'))$$

- Double DQN使用两个网络分别进行目标值计算与动作选择：

$$Q_{\omega_1} \leftarrow r + \gamma Q_{\omega_2}(s', \arg \max_{a'} Q_{\omega_1}(s', a'))$$

- 如果两个网络的误差不同，则可以在一定程度上解决问题。

Double DQN

Double DQN使用两个网络分别进行目标值计算与动作选择：

➤ 两个网络来源：当前策略网络和延迟目标网络

➤ DQN：

$$y = r + \gamma Q_{\omega^-}(s', \operatorname{argmax}_{a'} Q_{\omega^-}(s', a'))$$

➤ Double DQN：

$$y = r + \gamma Q_{\omega^-}(s', \operatorname{argmax}_{a'} Q_{\omega}(s', a'))$$

➤ 使用当前策略网络选择行为；使用目标网络计算TD目标值。

Dueling DQN

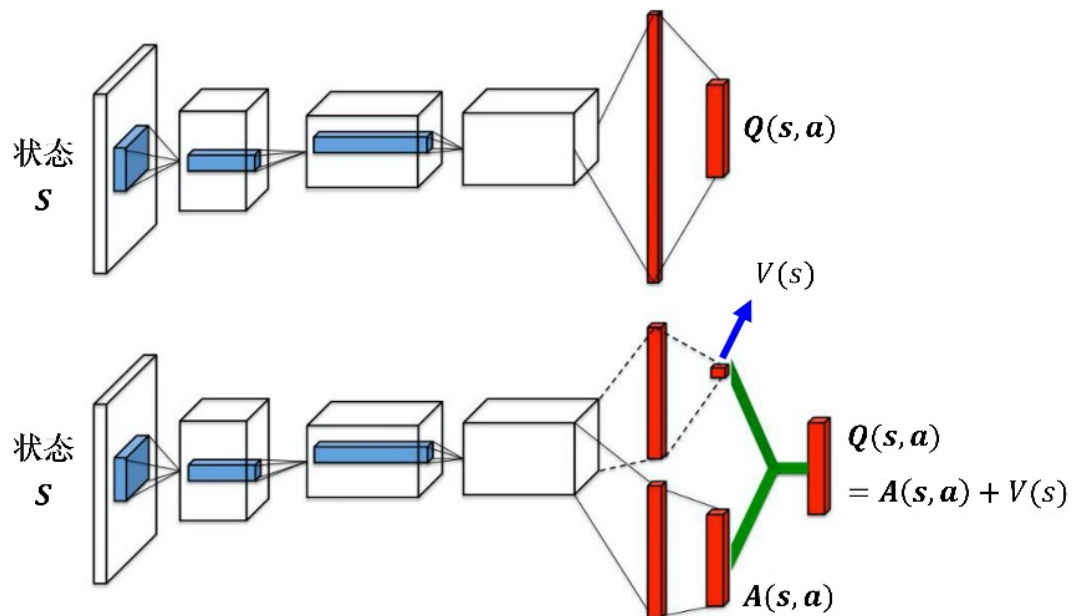
将原来的Q网络拆分成两个部分：V网络和A网络

- V网络：以状态为输入、以实数为输出的表示状态价值的网络；
- A网络：优势网络，它用于度量在某个状态 s 下选取某个具体动作 a 的合理性，它直接给出动作 a 的性能与所有可能的动作的性能的均值的差值。如果该差值(优势)大于0，说明动作 a 优于平均，是个合理的选择；如果差值(优势)小于0，说明动作 a 次于平均，不是好的选择；
- 一般来说： $Q(s, a) = V(s) + A(s, a)$ 。

Dueling DQN

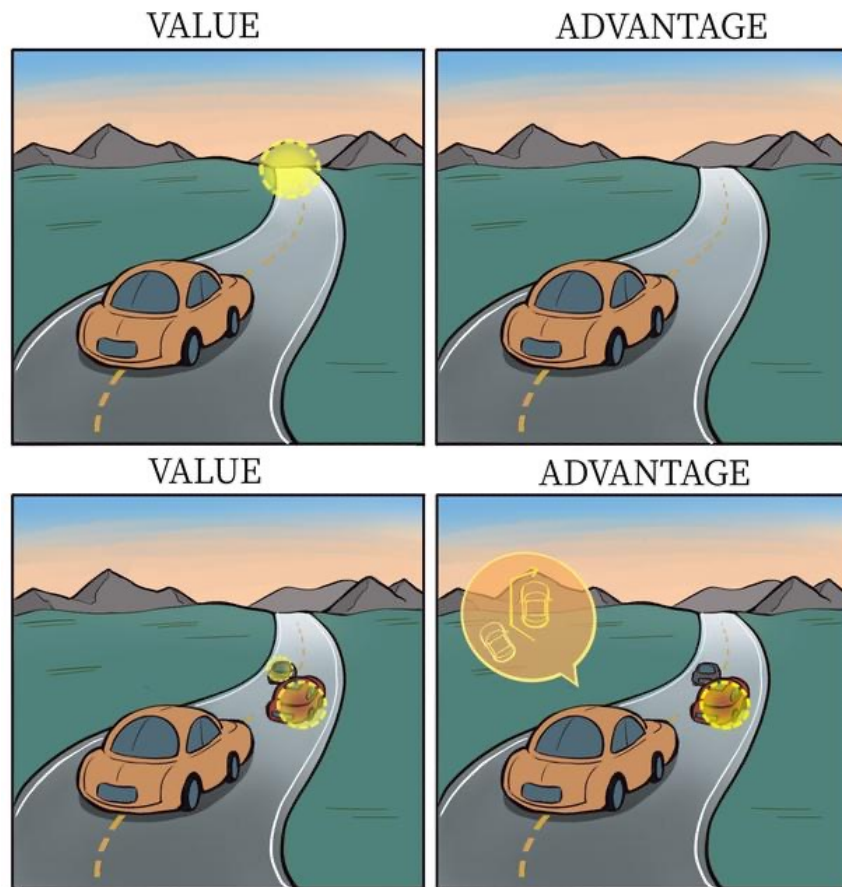
将原来的Q网络拆分成两个部分：V网络和A网络

- $V_{\omega}(s)$: 状态s的平均价值
- $A_{\theta}(s, a)$: 在状态s下，动作a相对其他行为的优势价值
- $A(s, a) = Q(s, a) - V(s)$



Dueling DQN

- 分辨当前的价值是由状态价值提供还是行为价值提供，进而有针对性的更新，增加样本利用率。
- 右图展示了价值函数和优势函数关注的区域
 - 当前方无车时，不同动作的优势价值 $A_{\theta}(s, a)$ 应无明显差异。
 - 而当前方有车时，不同动作应有不同的优势价值 $A_{\theta}(s, a)$



Dueling DQN

训练出来的 $V_\omega(s)$ 和 $A_\theta(s, a)$ 是否合理？

- 根据优势函数的定义 $A^*(s, a) = Q^*(s, a) - V^*(s)$ 以及 $V^*(s) = \max_{a'} Q^*(s, a)$:

$$\max_{a'} A^*(s, a') = \max_{a'} Q^*(s, a') - V^*(s) = 0$$

$$Q^*(s, a) = V^*(s) + A^*(s, a) - \max_{a'} A^*(s, a')$$

- 在计算时使用 $Q_{\omega, \theta}(s, a) = V_\omega(s) + A_\theta(s, a) - \max_{a'} A_\theta(s, a')$ 而非

$Q_{\omega, \theta}(s, a) = V_\omega(s) + A_\theta(s, a)$, 可以使 $\max_{a'} A_\theta(s, a')$ 在收敛时候趋于0。

- 在实际使用中, 往往用均值替代最大化操作, 即:

$$Q_{\omega, \theta}(s, a) = V_\omega(s) + A_\theta(s, a) - \sum_{a'} A_\theta(s, a') / |A|$$

优先经验回放池PER

一般来说，具有较大TD误差的样本应该给予更高的优先级。

- 采样第 t 个样本的概率 p_t 正比于TD误差 δ_t ：

$$p_t \propto |\delta_t| + \epsilon$$

其中 ϵ 是一个小正数，防止采样概率为0。

- 采样第 t 个样本的概率 p_t 反比于TD误差在全体样本中的排位 $rank(t)$ ：

$$p_t \propto \frac{1}{rank(t)}$$

- 第二种方法相对于第一种针对异常数据更鲁棒，过大或过小的异常点不影响其排位。

优先经验回放池PER

除了更改每个样本的采样概率之外，还需要相应调整样本的学习率：

- 引入参数 $\beta \in [0,1]$ 来调整各个样本的学习率 α_t ：

$$\alpha_t \leftarrow \alpha \cdot (np_t)^{-\beta}$$

其中 n 为样本数目

- 均匀采样时 $p_1 = p_2 = p_3 = \dots = p_n = \frac{1}{n}$ ，此时 $(np_t)^{-\beta} = 1$ ，所有样本的学习率均为 α ，就回归到普通的经验回放池。
- 基于优先采样时，具有高优先级的样本使用较低的学习率。

更多的基于DQN的改进

- 在蒙特卡洛方法和时序差分中平衡
- 噪声网络
- 分布Q学习
- Rainbow
-

下午实验课

需要完成：

- 完成经典DQN算法书写；
- 完成Atari中Freeway环境的测试；
- 尝试其他Atari环境

作业：

- 修改实现DDQN算法；

可选：

- 修改实现Dueling DQN算法；

提问环节

谢谢

