SOUTHEAST UNIVERSITY

强化学习与规划
强化学习在游戏中的应用

刘翔 吴江恒 张舒韬 赵倩隆

东南大学 计算机科学与工程学院 第 4 组

2018 年 5 月 14 日

总目录

第 I 部分

强化学习与规划

目录

马尔科夫决策过程
定义

一个马尔科夫决策过程是一个五元组 $D = \langle S, A, P, r, \gamma \rangle$，其中

- S 过程中的状态（state）集合
- A 过程中的动作（action）集合
- P 转移函数（transition function）$P(s, a, s')$ 定义为 $S \times A \times S \to [0, 1]$，表示在状态 s 时选择动作 a 达到状态 s' 的概率
- r 奖励函数（reward function）$r(s, a, s')$ 定义为 $S \times A \times S \to \mathbb{R}$，表示在状态 s 时选择动作 a 达到状态 s' 时得到的奖励
- γ 折扣因子（discount factor）$\gamma \in [0, 1]$

马尔科夫决策过程
定义

一个马尔科夫决策过程是一个五元组 $D = \langle S, A, P, r, \gamma \rangle$，其中

- S 过程中的状态（state）集合
- A 过程中的动作（action）集合
- P 转移函数（transition function）$P(s, a, s')$ 定义为 $S \times A \times S \to [0, 1]$，表示在状态 s 时选择动作 a 达到状态 s' 的概率
- r 奖励函数（reward function）$r(s, a, s')$ 定义为 $S \times A \times S \to \mathbb{R}$，表示在状态 s 时选择动作 a 达到状态 s' 时得到的奖励
- γ 折扣因子（discount factor）$\gamma \in [0, 1]$

累积奖励的定义

$$\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})$$

马尔科夫决策过程

问题

马尔科夫决策过程

问题

预测问题（Predicting） 已知起始状态、转移函数和决策策略，求在此情况下能够得到的累积奖励的期望；

马尔科夫决策过程
问题

预测问题（Predicting） 已知起始状态、转移函数和决策策略，求在此情况下能够得到的累积奖励的期望；

规划问题（Planning） 已知起始状态和转移函数，求使累积奖励的期望值最大的决策策略；

马尔科夫决策过程
问题

预测问题（Predicting） 已知起始状态、转移函数和决策策略，求在此情况下能够得到的累积奖励的期望；

规划问题（Planning） 已知起始状态和转移函数，求使累积奖励的期望值最大的决策策略；

强化学习（Reinforcement Learning） 转移函数或奖励函数未知，求使累积奖励的期望值最大的决策策略。

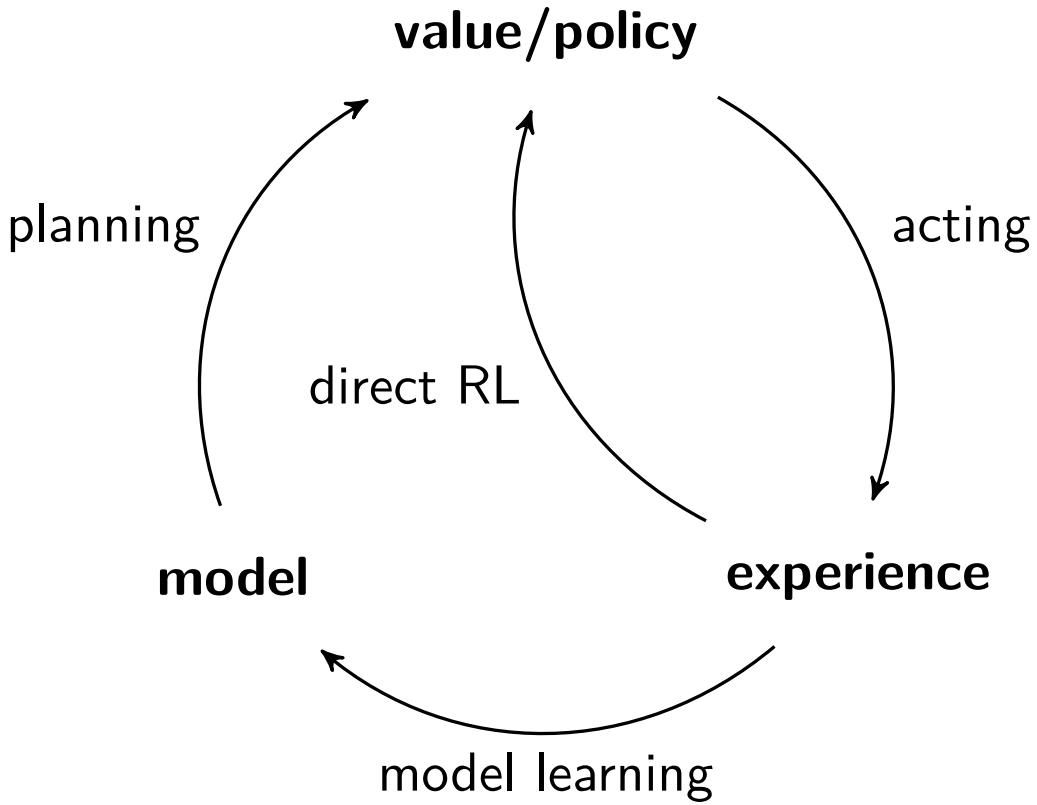
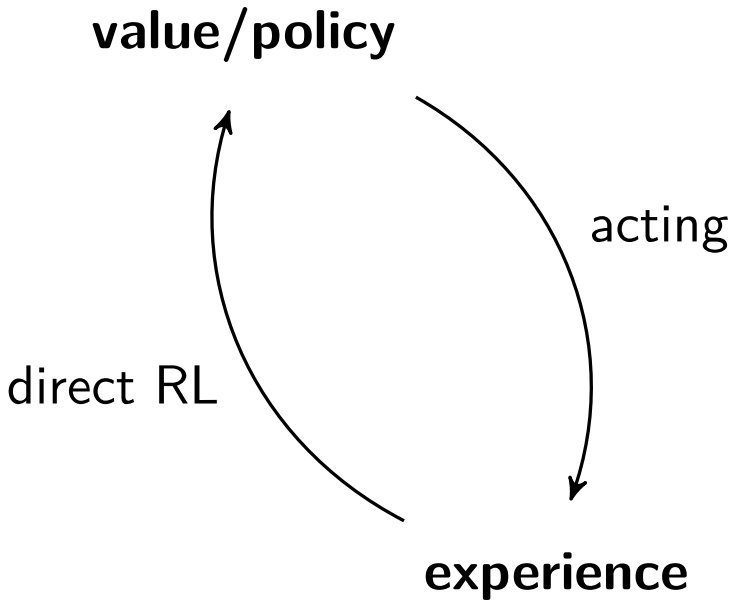


图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]



Model-Free RL

图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]

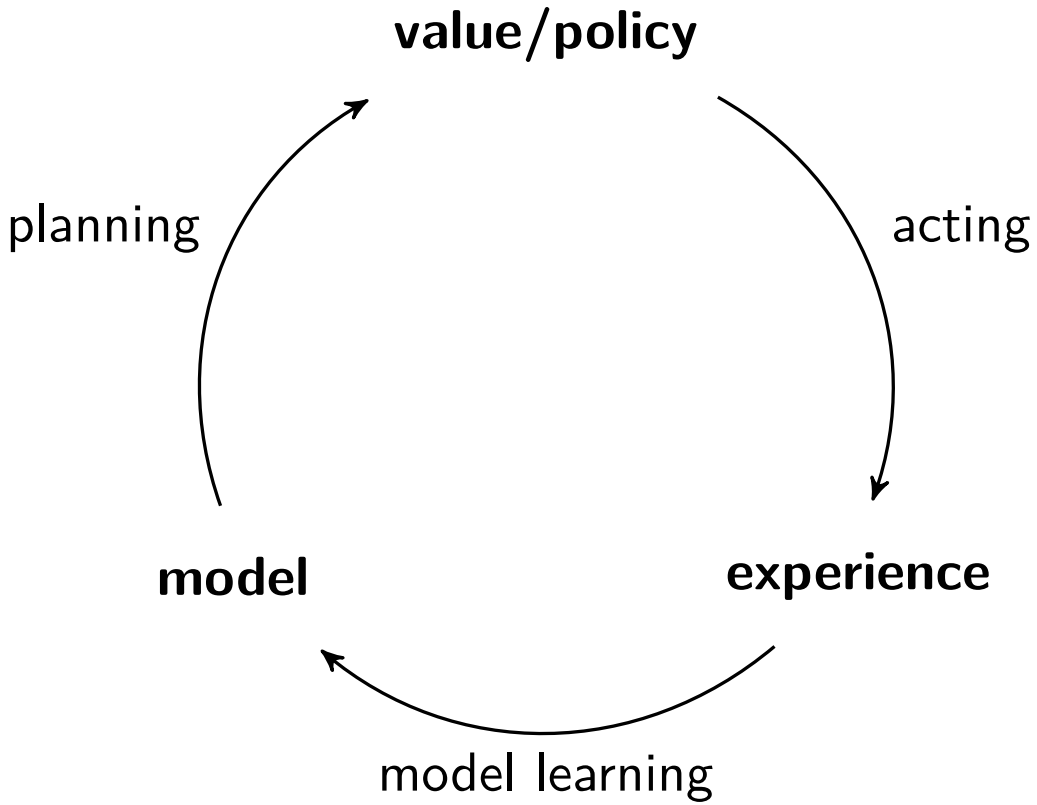


图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]

Q-Learning
定义

Q-Learning
定义

- Q-Learning 是一种 off-policy 的时序差分学习方法

Q-Learning
定义

- Q-Learning 是一种 off-policy 的时序差分学习方法
- Q-Learning 的目标是得到 Q^* 函数的估计，即

$$Q^*(s,a) = \max_{\pi} \mathbb{E}(R_t | s_t = t, a_t = a, \pi)$$

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a' \in A(s)} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right]$$

- Q-Learning 是一种 off-policy 的时序差分学习方法
- Q-Learning 的目标是得到 Q^* 函数的估计，即

$$Q^*(s,a) = \max_{\pi} \mathbb{E}(R_t | s_t = t, a_t = a, \pi)$$

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a' \in A(s)} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right]$$

- Q-learning 的定义为

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1},a) - Q(s_t,a_t) \right]$$

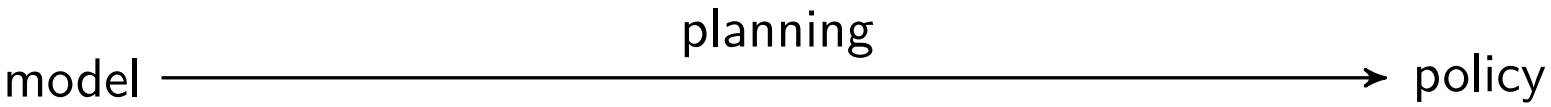
Q-Learning
算法描述

Algorithm: Q-Learning

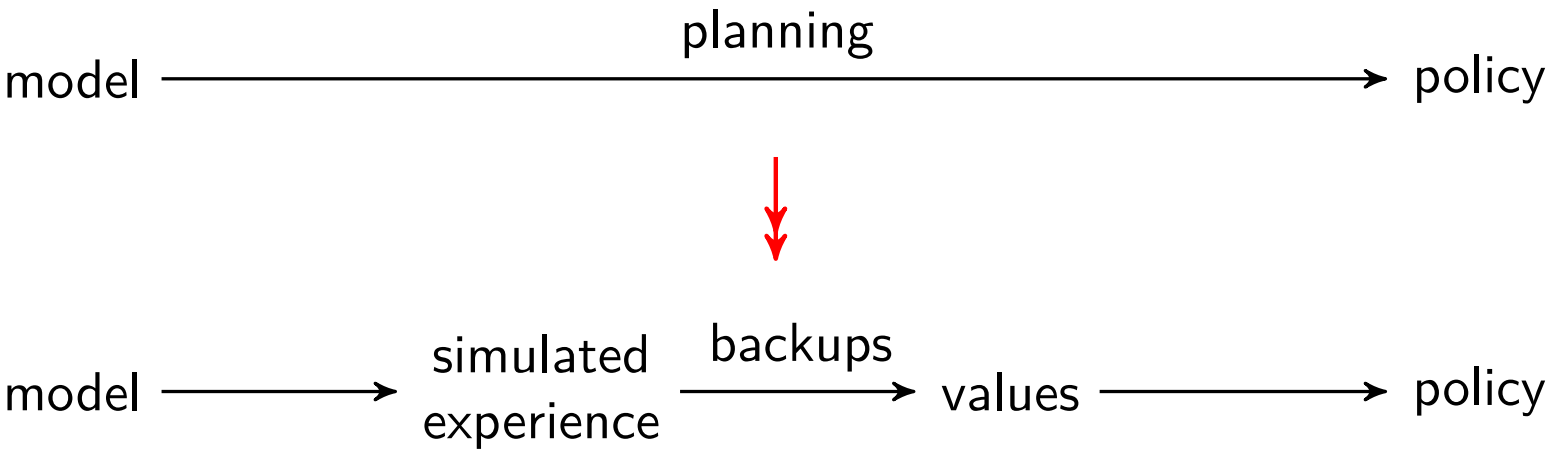
随机初始化 $Q(s, a)$;
foreach 每个周期（*episode*）**do**
    初始化 s;
    foreach 周期内的每一步，直到 s 为终止状态 **do**
        利用 Q 函数中获得的策略（例如 ϵ-greedy）选择状态 s 时采取的策略;
        执行 a，观察 s' 和 r';
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$;

基于强化学习的规划
Q-Planning

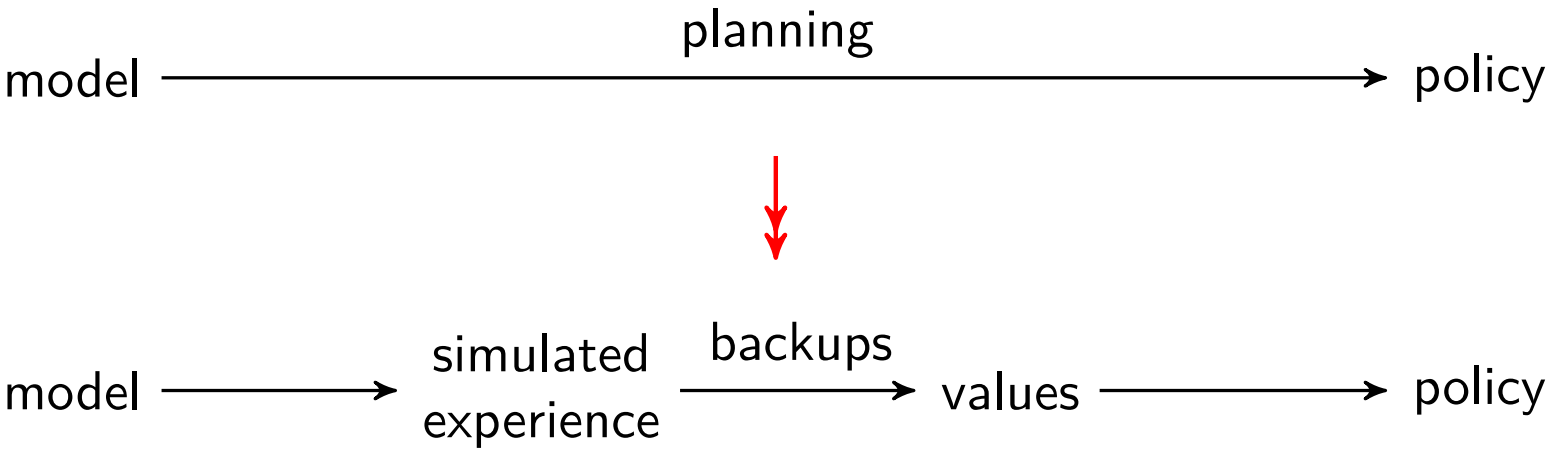
基于强化学习的规划
Q-Planning



基于强化学习的规划
Q-Planning



基于强化学习的规划
Q-Planning



Algorithm: 随机 Q-Planning

1 **repeat**
2 　随机选择 $s \in S$, $a \in A(s)$;
3 　根据模型模拟出下一状态 s' 和奖励 r;
4 　更新 $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$;
5 **until** *stop*;

Model-Based RL 优点：

- 有时直接从环境中学到值函数较难，而模型的 $P(s'|s,a)$ 和 $R(r|s,a)$ 很容易就能用监督学习去学；

缺点：

- 误差来源多了模型拟合的误差；

Model-Based RL 优点：

- 有时直接从环境中学到值函数较难，而模型的 $P(s'|s,a)$ 和 $R(r|s,a)$ 很容易就能用监督学习去学；

缺点：

- 误差来源多了模型拟合的误差；

Model-Free RL 优点：

- 不需要具体的环境模型；
- 使用时做出决策的时间快；

缺点：

- 优化过程可能不稳定且不收敛；
- 比较难适应变化的环境；

All models are wrong, but some are useful.

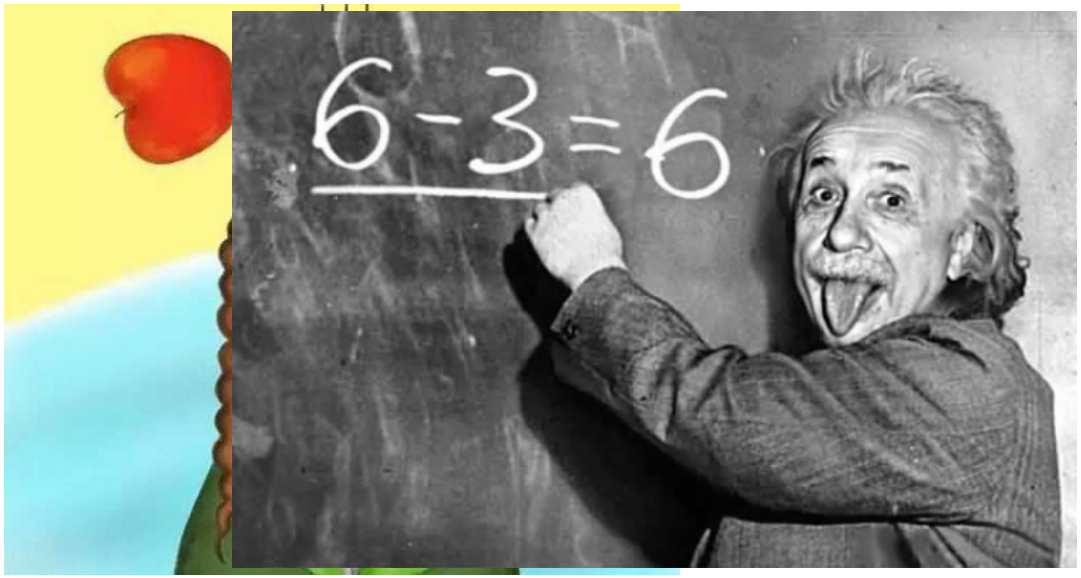
–George E.P. Box, Robustness in the strategy of scientific model building

All models are wrong, but some are useful.

–George E.P. Box, Robustness in the strategy of scientific model building

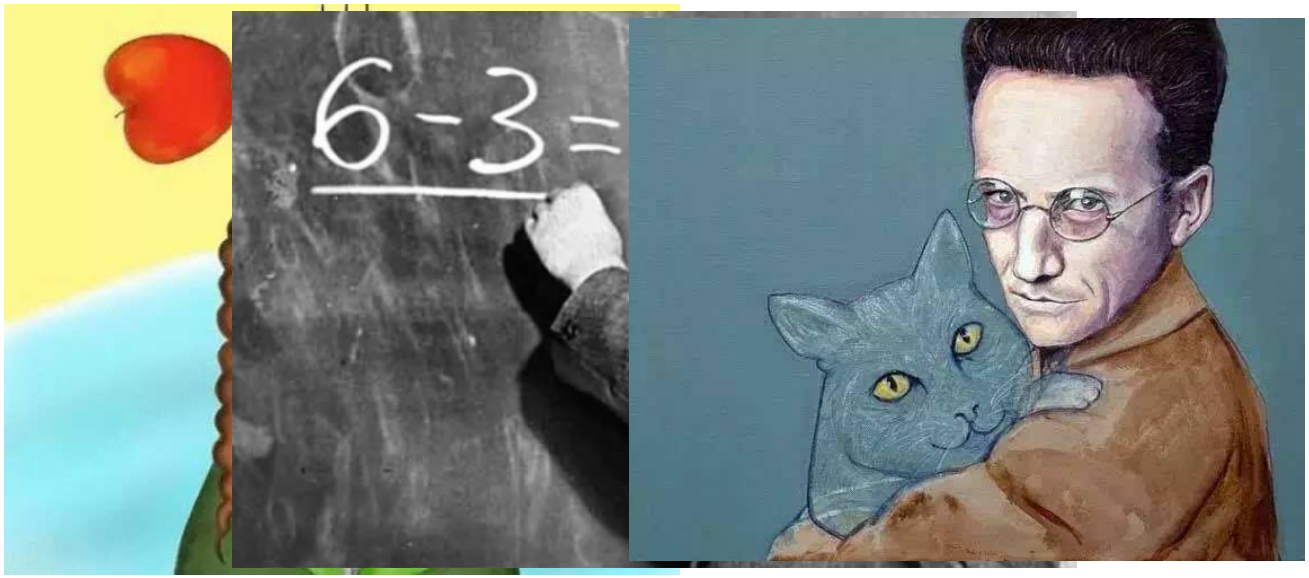
All models are wrong, but some are useful.

–George E.P. Box, Robustness in the strategy of scientific model building



All models are wrong, but some are useful.

–George E.P. Box, Robustness in the strategy of scientific model building



目录

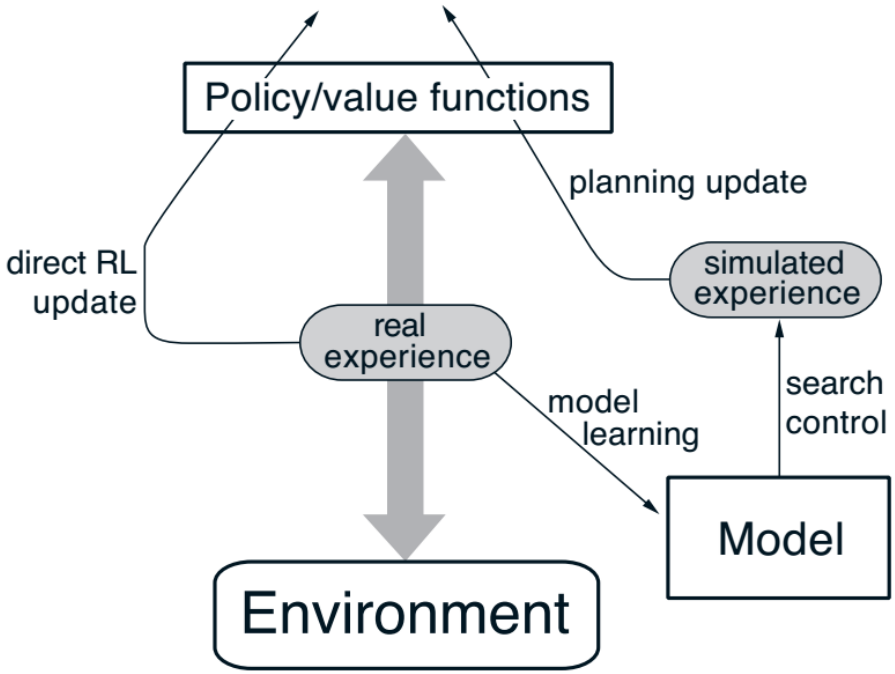


图 2: Dyna agent 的一般结构

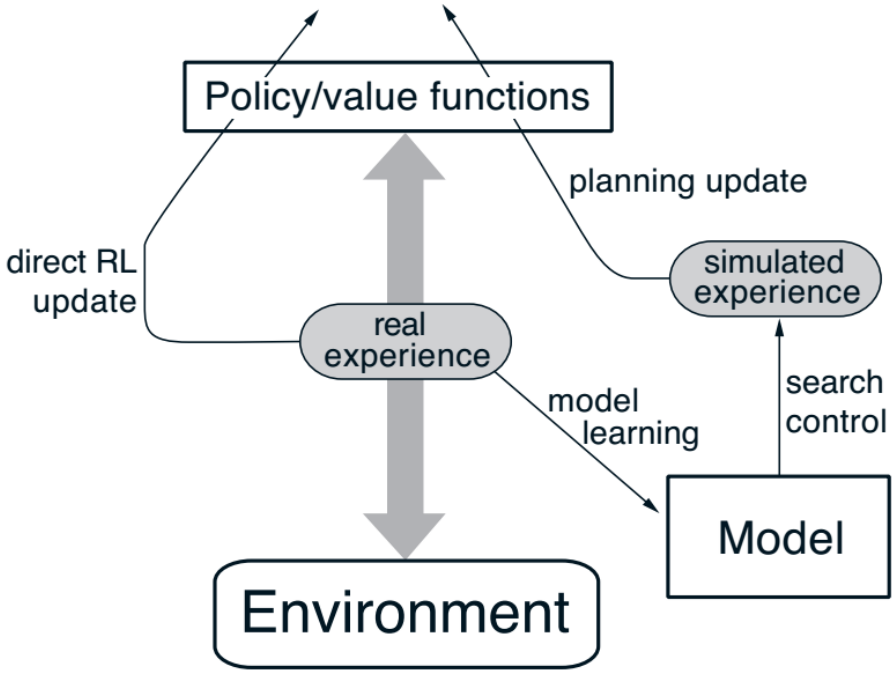


图 2: Dyna agent 的一般结构

- Dyna[Sutton, 1990] 包括了图1中的所有过程，即规划、执行、模型学习和值函数学习

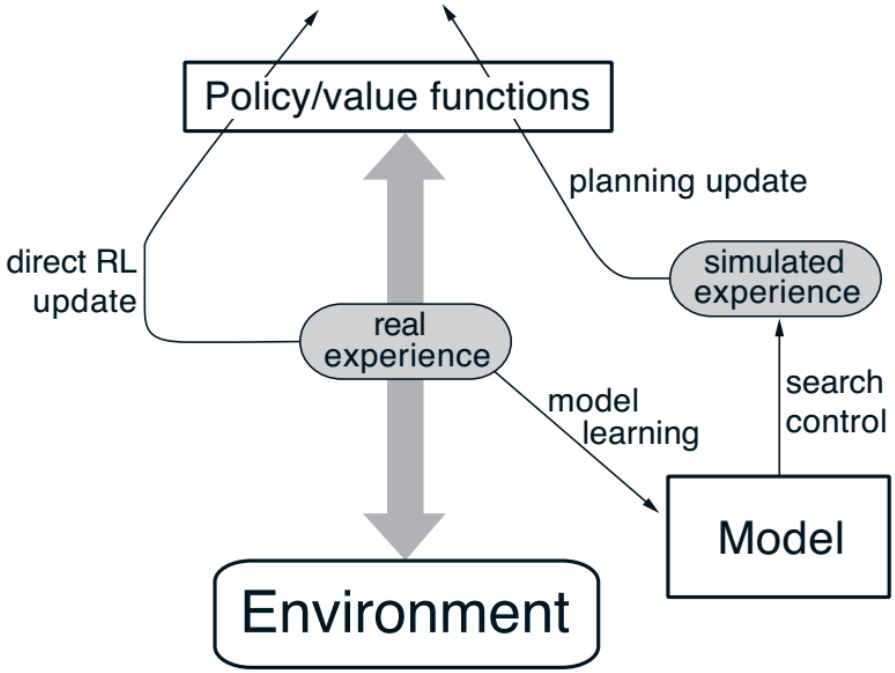


图 2: Dyna agent 的一般结构

- Dyna[Sutton, 1990] 包括了图1中的所有过程，即规划、执行、模型学习和值函数学习
- 规划时使用一个确定的模型（即 $P(s, a, s') \rightarrow \{0, 1\}$），该模型在执行过程中不断更新；

Algorithm: Dyna-Q 算法

对任意的 $s \in S$，$a \in A(s)$，初始化 $Q(s,a)$ 和 $Model(s,a)$;
repeat
  对当前状态 s，根据 Q 表选择 $a \in A(s)$ 并执行，得 s' 和 r;
  更新 $Q(s,a)$;
  用 s', r 更新 $Model(s,a)$;
  repeat // 在 $Model$ 上规划并更新 Q
    s 为任意观察到的状态，a 为 s 上进行过的任意操作;
    $s', r \leftarrow Model(s,a)$;
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$;
  until N 次循环;
until *stop*;

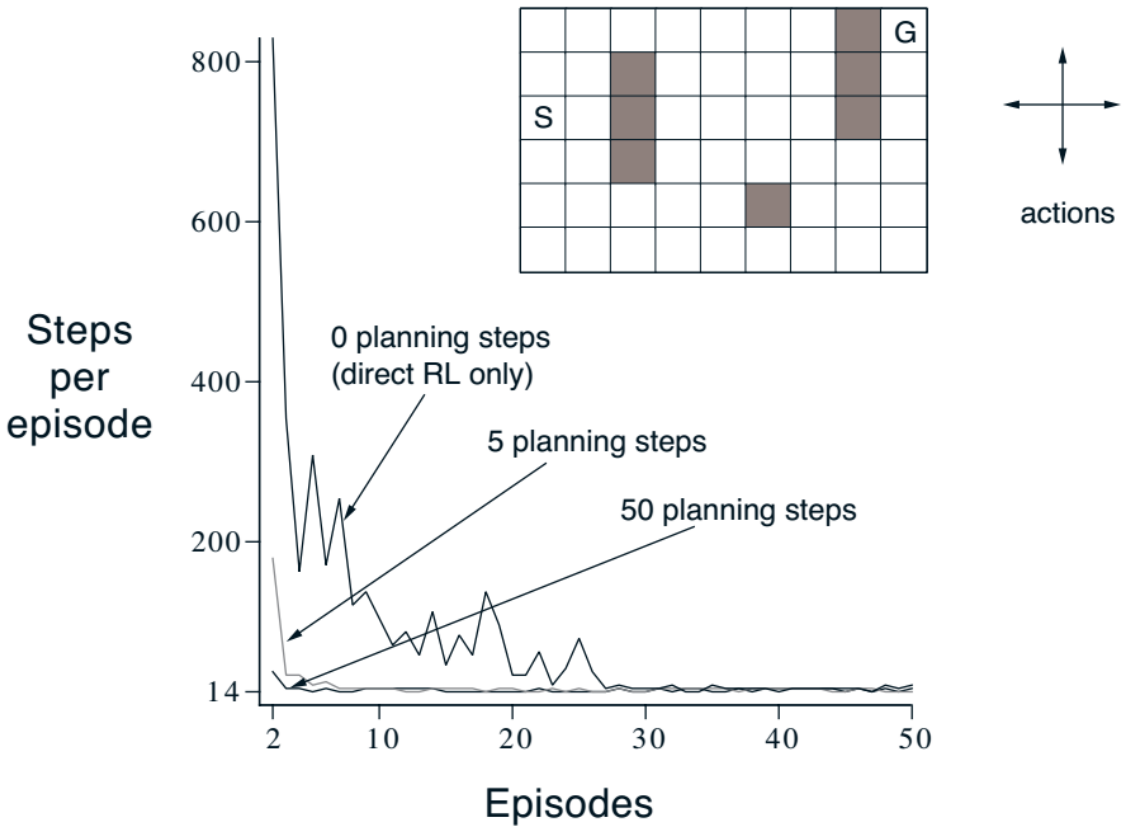


图 3: Dyna-Q 在一个迷宫问题上的实验，agent 需要从 S 走到 G，$r \to \{0, 1\}$。对于所有的 N，第一次探索是一样的，都是大约 1700 步；之后，Dyna-Q 和 Q-Learning 的差别开始显现

模型出错的原因：

- 先验知识存在错误
- 随机环境难以用模型描述或估计
- 环境出现变化

处理方法：学习过程中不断使用从环境中获得的数据对模型进行更新和纠正

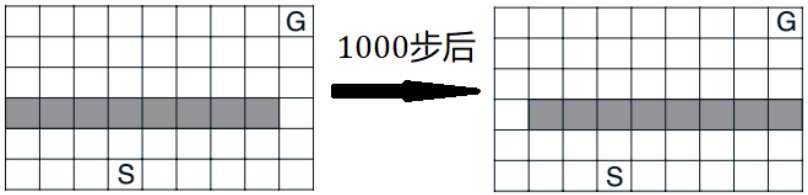


图 4: 1000 步之后，迷宫出现变化，左侧墙壁打开，右侧通路封闭

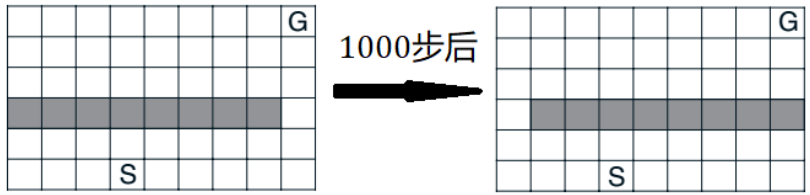


图 4: 1000 步之后，迷宫出现变化，左侧墙壁打开，右侧通路封闭

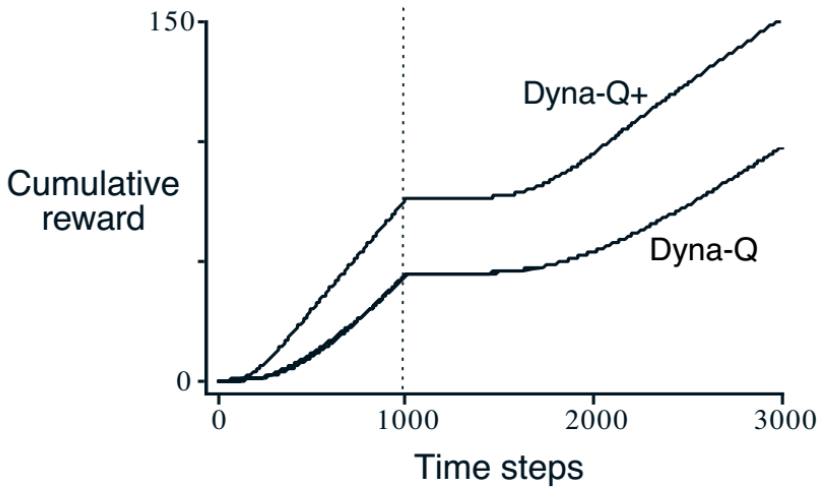


图 5: 由于 Dyna 算法包括了对模型的更新，agent 在一段时间后完成了对环境的重新建模并找到了正确的路径。图中 Q+ 是加强了探索能力的 Q-Learning，用 $r + \kappa\sqrt{n}$ 为奖励函数，n 为 s 状态下 a 连续未被选中的次数

DARLING 方法 [Leonetti et al., 2016] 的求解步骤：

DARLING 方法 [Leonetti et al., 2016] 的求解步骤：

1. 根据预设的模型，用规划求解器（例如 ASP 推理机）求解某个度量值（例如规划的步骤数量）在阈值内的规划方案；

DARLING 方法 [Leonetti et al., 2016] 的求解步骤：

1. 根据预设的模型，用规划求解器（例如 ASP 推理机）求解某个度量值（例如规划的步骤数量）在阈值内的规划方案；
2. 筛选合并求得的规划方案，删除包含冗余步骤的方案，融合后得到部分策略，即在各个状态下可选的行动集合；

DARLING 方法 [Leonetti et al., 2016] 的求解步骤：

1. 根据预设的模型，用规划求解器（例如 ASP 推理机）求解某个度量值（例如规划的步骤数量）在阈值内的规划方案；
2. 筛选合并求得的规划方案，删除包含冗余步骤的方案，融合后得到部分策略，即在各个状态下可选的行动集合；
3. 执行和学习，在执行中仅选择部分策略中的行为，学习它们的累积奖励的期望并优化策略。

建模（Modeling） 对环境 $D = \langle S, A, P, r, \gamma \rangle$ 进行建模，建立从环境状态到模型状态的函数 $o : S \to S_m$，得 $D_m = \langle S_m, A, P_m \rangle$，其中

建模（Modeling） 对环境 $D = \langle S, A, P, r, \gamma \rangle$ 进行建模，建立从环境状态到模型状态的函数 $o : S \to S_m$，得 $D_m = \langle S_m, A, P_m \rangle$，其中

规划（Planning） 利用规划工具，以一定的冗余度计算可行的方案。规划是在一个假设的模型上执行的，因此不能保证所得的方案是最优甚至可行的；

建模（Modeling） 对环境 $D = \langle S, A, P, r, \gamma \rangle$ 进行建模，建立从环境状态到模型状态的函数 $o : S \to S_m$，得 $D_m = \langle S_m, A, P_m \rangle$，其中

规划（Planning） 利用规划工具，以一定的冗余度计算可行的方案。规划是在一个假设的模型上执行的，因此不能保证所得的方案是最优甚至可行的；

筛选（Filtering） 如果规划方案中存在重复出现的状态和动作（例如存在环），则认为方案是冗余的，可以被更短的方案代替；

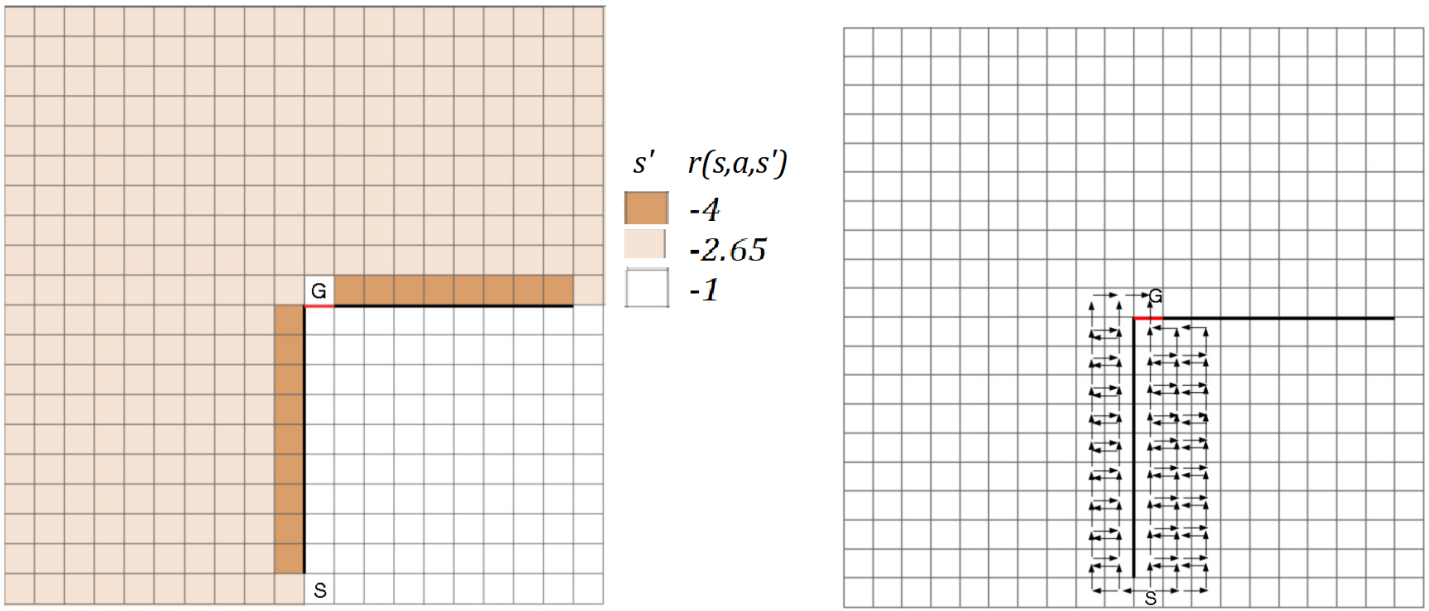
建模（Modeling） 对环境 $D = \langle S, A, P, r, \gamma \rangle$ 进行建模，建立从环境状态到模型状态的函数 $o : S \to S_m$，得 $D_m = \langle S_m, A, P_m \rangle$，其中

规划（Planning） 利用规划工具，以一定的冗余度计算可行的方案。规划是在一个假设的模型上执行的，因此不能保证所得的方案是最优甚至可行的；

筛选（Filtering） 如果规划方案中存在重复出现的状态和动作（例如存在环），则认为方案是冗余的，可以被更短的方案代替；

合并（Merging） 合并筛选过的方案，得到可达状态和这些状态下可选择的动作的集合，即一个简化的模型 D_r；

从 Planning 到 Merging 的过程其实是将假设中的模型 D_m 做了进一步的化简和压缩。



(a) 迷宫问题，要求机器人从 S 走到 G，(b) 部分策略，由筛选后的 non-redudant 方案融合而成
图中红色横线为一扇可能开启或关闭的门。
Planning 步骤求长度小于最短方案的 1.5 倍的方案

- 实际的转移函数和模型中设想的转移函数并不一致。这会导致 agent 在执行和学习的过程中进入了模型中没有的状态。此时应该以新出现的状态为初始状态，重新进行规划，并将所得的部分策略添加到已有的部分策略中；
- 在所得模型上的 RL 与其他的强化学习并无太大差别，任意的强化学习方法都可以实现。

- 实验采用的环境如图17所示。仅在 agent 到达门口时才获知门是否打开（Partial Observed MDP，POMDP），在周期 e 时门打开的概率为

$$p(e) = \begin{cases} 1 - \frac{e}{E-1} & 0 \leq e < E \\ 0 & \text{other wise} \end{cases}$$

- 实验采用的环境如图17所示。仅在 agent 到达门口时才获知门是否打开（Partial Observed MDP，POMDP），在周期 e 时门打开的概率为

$$p(e) = \begin{cases} 1 - \frac{e}{E-1} & 0 \leq e < E \\ 0 & \text{other wise} \end{cases}$$

- 实验中用到了以下几种 agent:

- 实验采用的环境如图17所示。仅在 agent 到达门口时才获知门是否打开（Partial Observed MDP，POMDP），在周期 e 时门打开的概率为

$$p(e) = \begin{cases} 1 - \frac{e}{E-1} & 0 \leq e < E \\ 0 & \text{other wise} \end{cases}$$

- 实验中用到了以下几种 agent:

  P agent 仅在 D_m 上进行规划;

- 实验采用的环境如图17所示。仅在 agent 到达门口时才获知门是否打开（Partial Observed MDP，POMDP），在周期 e 时门打开的概率为

$$p(e) = \begin{cases} 1 - \frac{e}{E-1} & 0 \leq e < E \\ 0 & \text{other wise} \end{cases}$$

- 实验中用到了以下几种 agent:

P agent 仅在 D_m 上进行规划;
RL agent 仅在 D 上进行强化学习;

- 实验采用的环境如图17所示。仅在 agent 到达门口时才获知门是否打开（Partial Observed MDP，POMDP），在周期 e 时门打开的概率为

$$p(e) = \begin{cases} 1 - \frac{e}{E-1} & 0 \leq e < E \\ 0 & \text{other wise} \end{cases}$$

- 实验中用到了以下几种 agent:

P agent	仅在 D_m 上进行规划;
RL agent	仅在 D 上进行强化学习;
PRL agent	采用 DARLING 方法，在 D_m 上计算部分策略，并将强化学习的探索限制在 D_r 上;

- 实验采用的环境如图17所示。仅在 agent 到达门口时才获知门是否打开（Partial Observed MDP，POMDP），在周期 e 时门打开的概率为

$$p(e) = \begin{cases} 1 - \frac{e}{E-1} & 0 \leq e < E \\ 0 & \text{other wise} \end{cases}$$

- 实验中用到了以下几种 agent:

P agent 仅在 D_m 上进行规划;
RL agent 仅在 D 上进行强化学习;
PRL agent 采用 DARLING 方法，在 D_m 上计算部分策略，并将强化学习的探索限制在 D_r 上;
Pmem-n agent 采用 DARLING 方法，有前 n 个周期内观察门是否打开的记忆，并以此估计当前门是否打开;

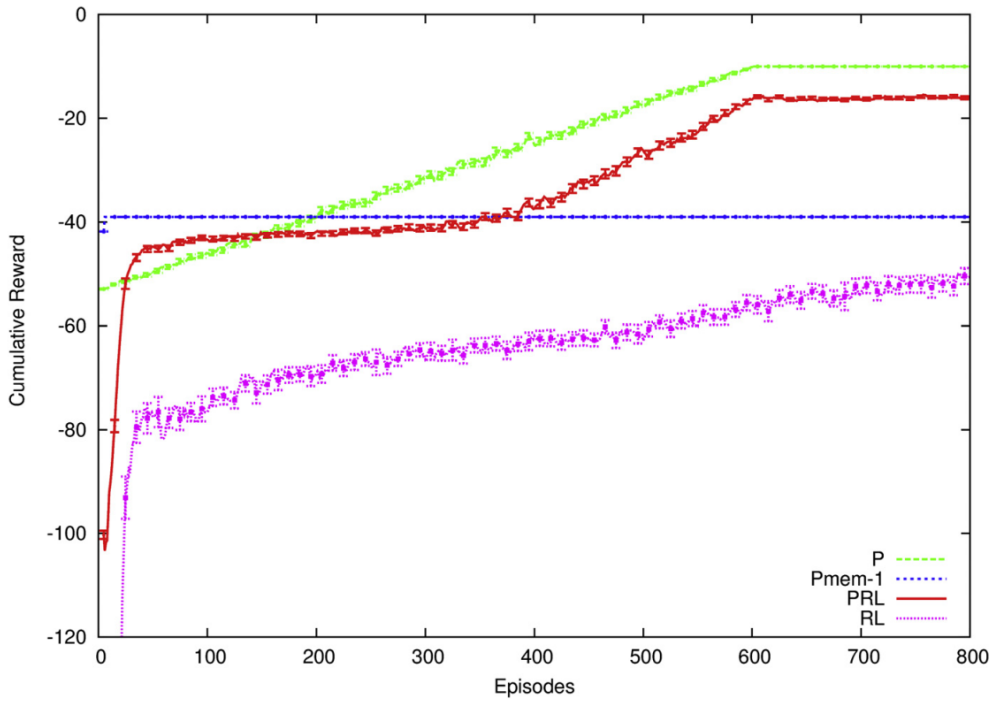

Cumulative Reward
Episodes
0
-20
-40
-60
-80
-100
-120
0
100
200
300
400
500
600
700
800
P
Pmem-1
PRL
RL

图 6: 实验结果

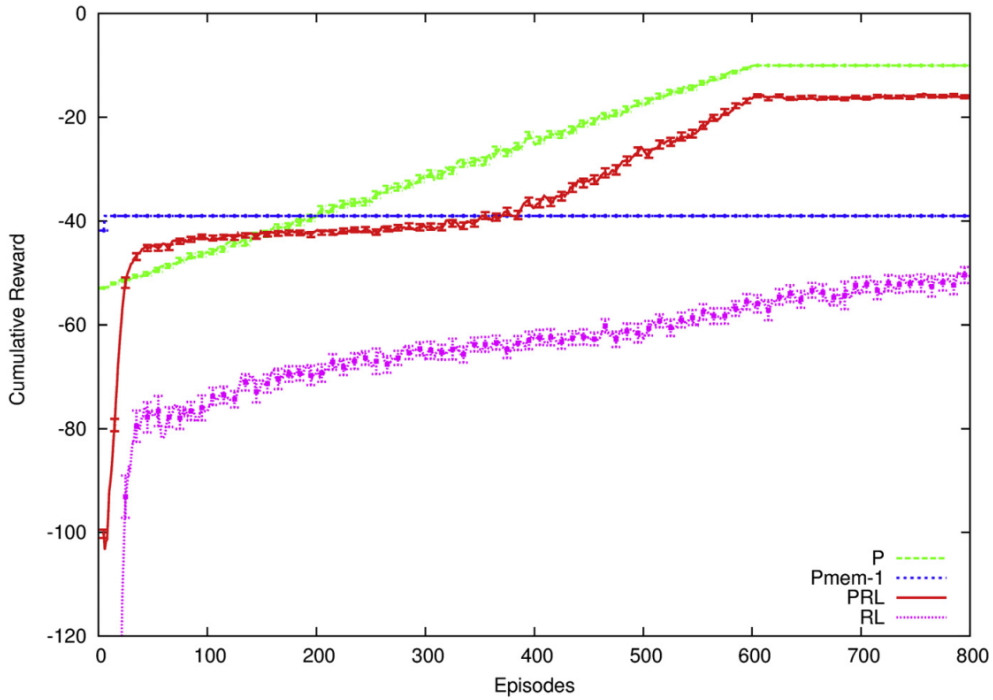


图 6: 实验结果

P agent 的累积奖励变化与 $p(e)$ 的概率保持一致。但由于实际应用中，D_m 和 D 差距较大，因此直接使用规划方法不可能取得如此良好的结果。

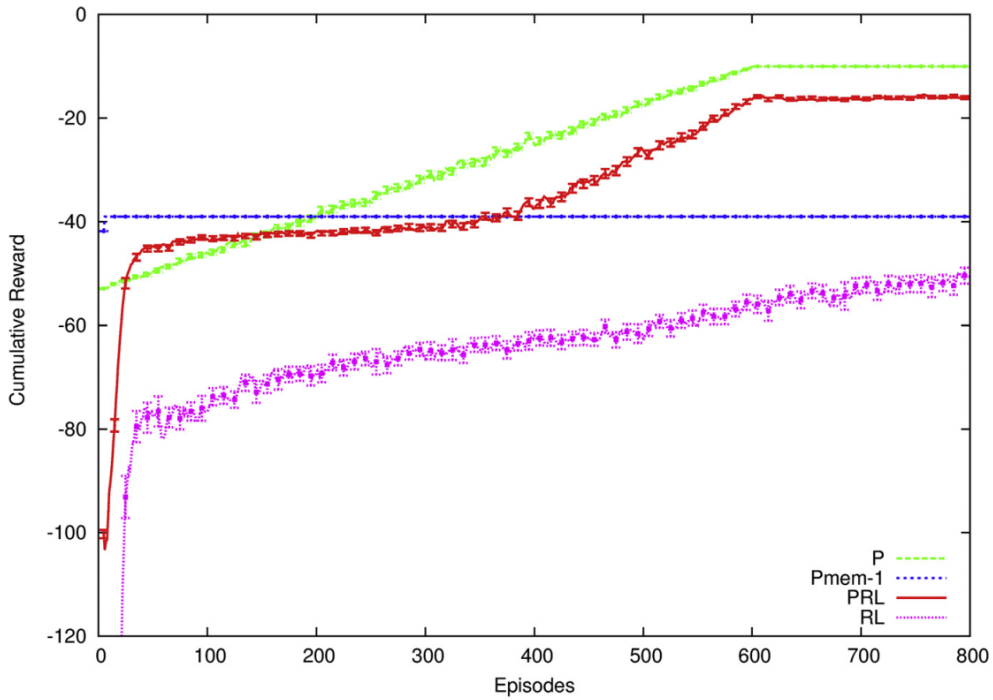


图 6: 实验结果

RL agent 很快学到了最优策略，但环境变化后没有能发现更优的策略。

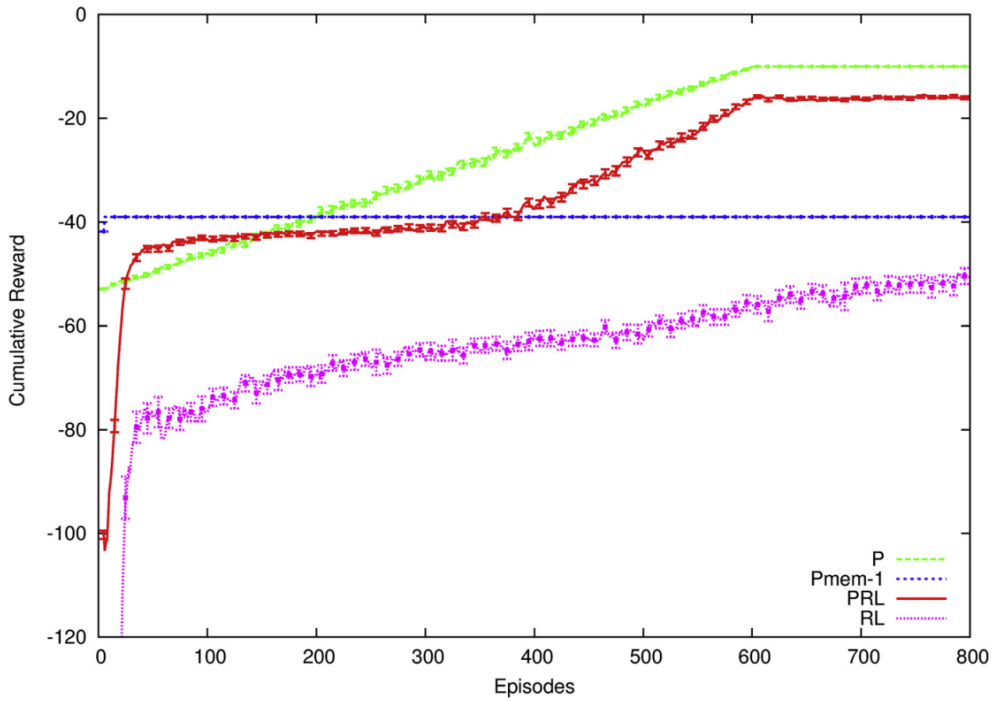


图 6: 实验结果

PRL agent 学到了最优策略，并在环境变化后选择了更优的策略。

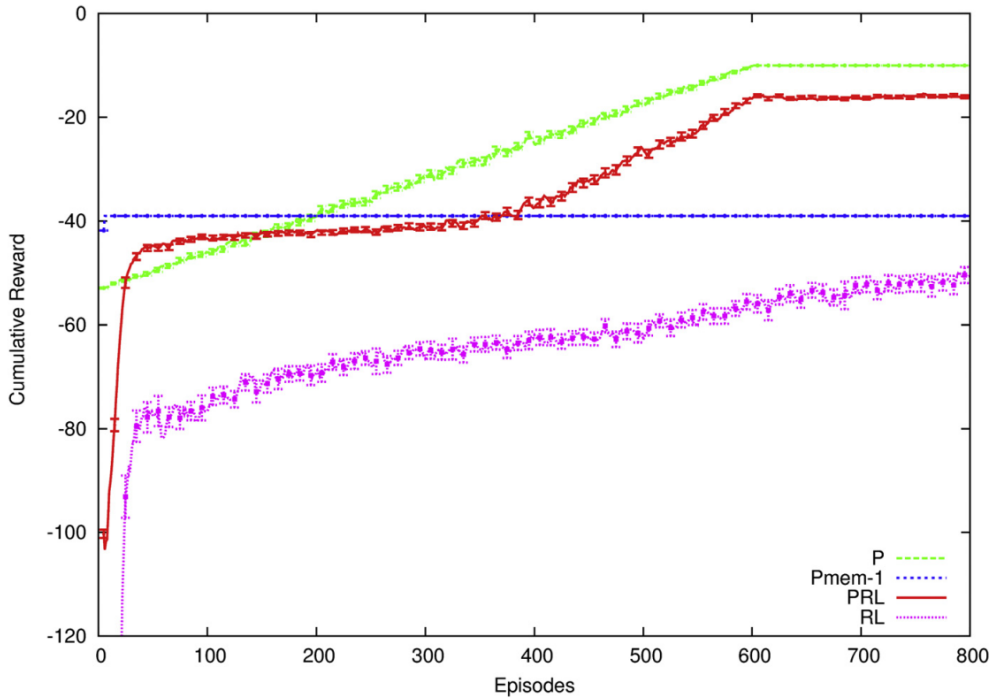


图 6: 实验结果

Pmem-1 agent 被起初的观察限制，没有发现环境的变化，因此没有发现更优的策略。实际上，无论 n 的取值如何，agent 在不稳定的环境中都没有获得最优策略。

東南大學
SOUTHEAST UNIVERSITY

第 II 部分

事后经验回放

目录

相关背景
简介

- 强化学习（RL）与神经网络的结合被广泛应用于序列决策问题中；

- 强化学习（RL）与神经网络的结合被广泛应用于序列决策问题中；
- 一个必须要面对的问题（尤其对于机器人设计来说）：通常需要设计一个回报函数（reward function），它不仅要体现目标任务，而且还要能够指导 agent 优化决策策略（policy optimization）；

- 强化学习（RL）与神经网络的结合被广泛应用于序列决策问题中；
- 一个必须要面对的问题（尤其对于机器人设计来说）：通常需要设计一个回报函数（reward function），它不仅要体现目标任务，而且还要能够指导 agent 优化决策策略（policy optimization）；
- 由此带来的问题是：这不仅需要 RL 专业知识，还需要领域特定的知识（domain-specific knowledge）；

- 强化学习（RL）与神经网络的结合被广泛应用于序列决策问题中；
- 一个必须要面对的问题（尤其对于机器人设计来说）：通常需要设计一个回报函数（reward function），它不仅要体现目标任务，而且还要能够指导 agent 优化决策策略（policy optimization）；
- 由此带来的问题是：这不仅需要 RL 专业知识，还需要领域特定的知识（domain-specific knowledge）；

Problem：能否设计一个算法，能够从 unshaped reward signals（e.g. 一个 0-1 reward 表明任务是否成功完成）中学习?

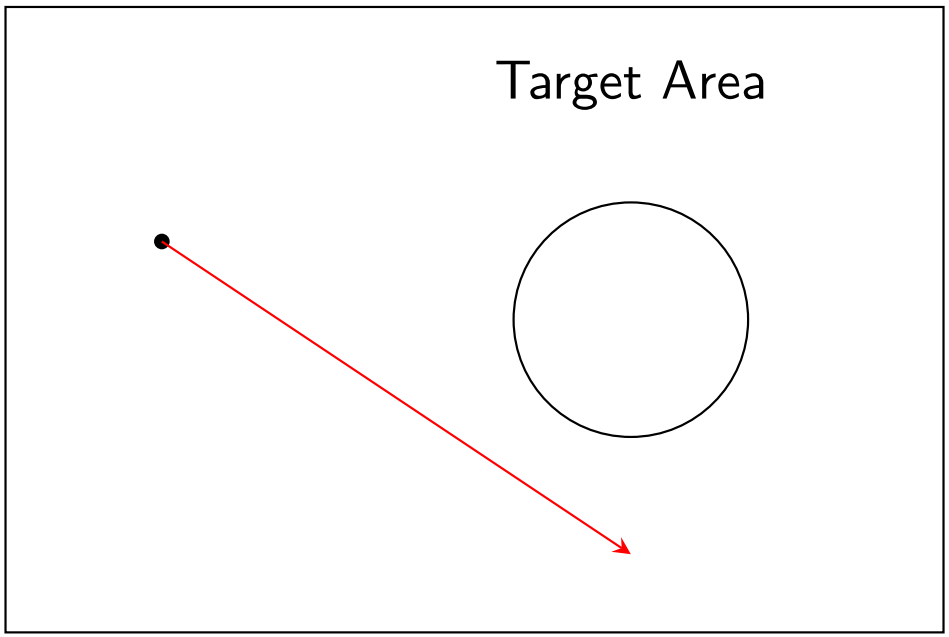


图 7: 人类拥有的一种能力是从未达到预期的结果那里获得与希望获得的结果几乎一样的知识。

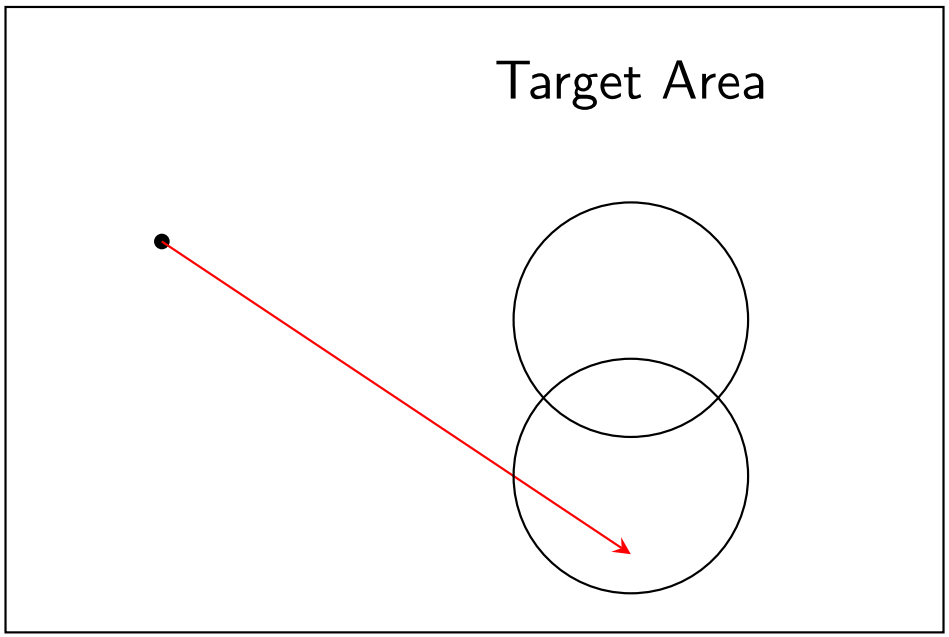


图 7: 人类拥有的一种能力是从未达到预期的结果那里获得与希望获得的结果几乎一样的知识。

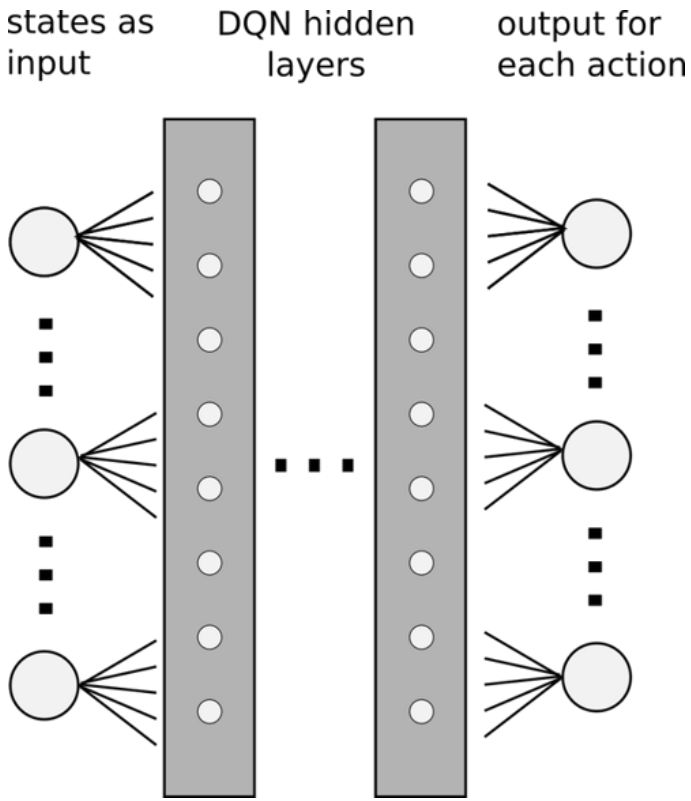


图 8: 使用神经网络来近似 Q 函数。该网络将一个状态 s 作为输入，并为每个动作 a 输出 Q 函数的估计值。

- 对于不可枚举状态的环境（如连续的状态空间），一般采用值函数逼近（Value Function Approxiamation）来拟合和泛化 Q 函数。
- 经验回放（Experience replay）是从一个 memory pool 中随机选取 expeirence 更新网络。
- DQN 是基于监督学习的框架，而 loss function 基于 RL，因此会有这样的问题：监督学习需要假设样本是独立同分布，而 RL 中的强序列关联性不符合这个假设。如果没有 experience replay，可能会出现不收敛的情况；
- 采用 Experience Replay，会缓和样本的关联性。其优点包括：
  1. 算法收敛；
  2. 提高泛化能力;

- DQN 是一个面向离散控制的算法，即输出的动作是离散的；

- DQN 是一个面向离散控制的算法，即输出的动作是离散的；
- 深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法是利用 DQN 扩展 Q 学习算法的思路对确定性策略梯度（Deterministic Policy Gradient，DPG）方法进行改造，提出的一种基于行动者-评论家（Actor-Critic，AC）框架的算法，该算法可用于解决连续动作空间上的 DRL 问题。

- DQN 是一个面向离散控制的算法，即输出的动作是离散的；
- 深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法是利用 DQN 扩展 Q 学习算法的思路对确定性策略梯度（Deterministic Policy Gradient，DPG）方法进行改造，提出的一种基于行动者-评论家（Actor-Critic，AC）框架的算法，该算法可用于解决连续动作空间上的 DRL 问题。
- 随机性策略和确定性策略：

- DQN 是一个面向离散控制的算法，即输出的动作是离散的；
- 深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法是利用 DQN 扩展 Q 学习算法的思路对确定性策略梯度（Deterministic Policy Gradient，DPG）方法进行改造，提出的一种基于行动者-评论家（Actor-Critic，AC）框架的算法，该算法可用于解决连续动作空间上的 DRL 问题。
- 随机性策略和确定性策略：

  随机性策略 策略输出的是动作的概率，比如连续动作控制，使用的是一个正态分布对动作进行采样选择，即每个动作都有概率被选到；

  优点 将探索和改进集成到一个策略中；

  缺点 需要大量训练数据；

- DQN 是一个面向离散控制的算法，即输出的动作是离散的；
- 深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法是利用 DQN 扩展 Q 学习算法的思路对确定性策略梯度（Deterministic Policy Gradient，DPG）方法进行改造，提出的一种基于行动者-评论家（Actor-Critic，AC）框架的算法，该算法可用于解决连续动作空间上的 DRL 问题。
- 随机性策略和确定性策略：

  随机性策略 策略输出的是动作的概率，比如连续动作控制，使用的是一个正态分布对动作进行采样选择，即每个动作都有概率被选到；

  优点 将探索和改进集成到一个策略中；

  缺点 需要大量训练数据；

  确定性策略 策略输出即是动作

  优点 需要采样的数据少，算法效率高；

  缺点 无法探索环境；

目录

例

- 环境描述:

  状态空间 $S = \{0, 1\}^n$
  动作空间 $\{0, 1, \ldots, n-1\}$
  奖励函数 $r(s, a) = -sgn(s \neq g)$

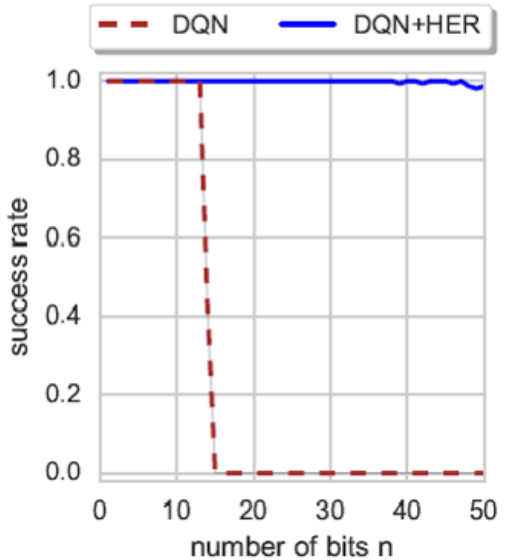


图 9: Bit Flipping 实验结果

s:0100101010
g:1101010100

例

- 环境描述:

  状态空间 $S = \{0, 1\}^n$
  动作空间 $\{0, 1, \ldots, n-1\}$
  奖励函数 $r(s, a) = -sgn(s \neq g)$

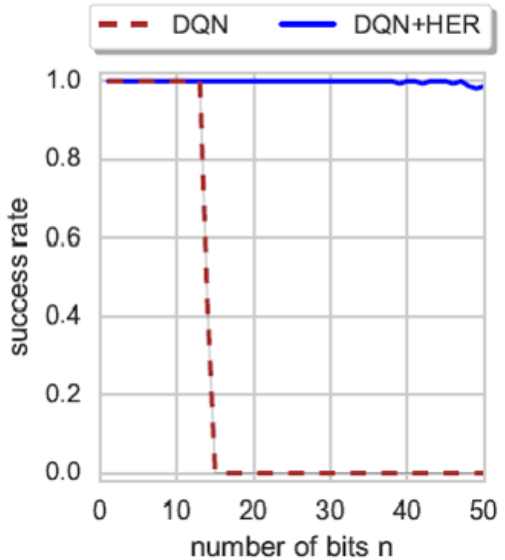


图 9: Bit Flipping 实验结果

Action:0

s:0100101010
g:1101010100

例

- 环境描述：

  状态空间 $S = \{0, 1\}^n$
  动作空间 $\{0, 1, \ldots, n-1\}$
  奖励函数 $r(s, a) = -sgn(s \neq g)$

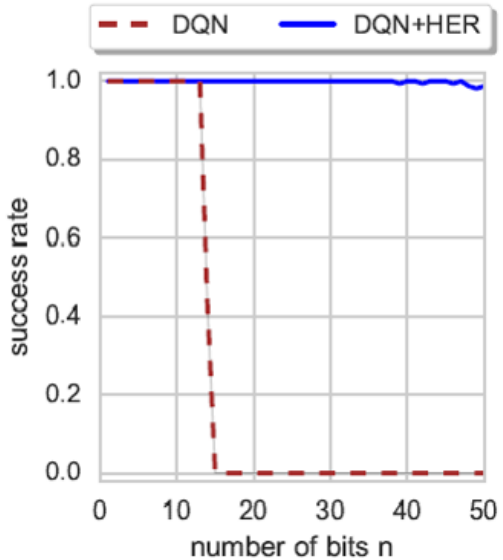


图 9: Bit Flipping 实验结果

$$s{:}0100101010 \xrightarrow{Action{:}0} 1100101010$$
$$g{:}1101010100$$

例

- 环境描述：

  状态空间 $S = \{0, 1\}^n$
  动作空间 $\{0, 1, \ldots, n-1\}$
  奖励函数 $r(s, a) = -sgn(s \neq g)$

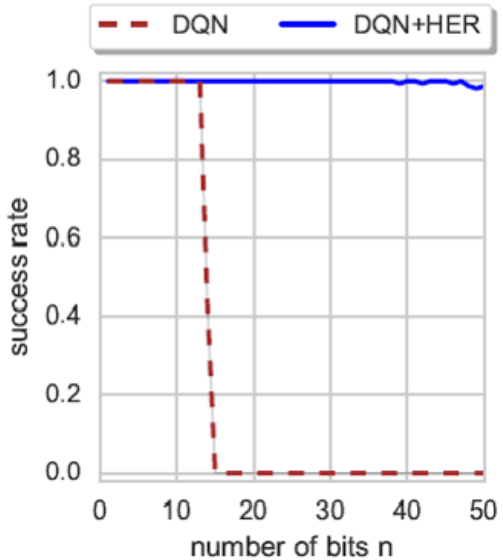


图 9: Bit Flipping 实验结果

$$s{:}0100101010 \xrightarrow[Reward{:}\text{-}1]{Action{:}0} 1100101010$$
$$g{:}1101010100$$

例

- 环境描述：

  状态空间 $S = \{0, 1\}^n$
  动作空间 $\{0, 1, \ldots, n-1\}$
  奖励函数 $r(s, a) = -sgn(s \neq g)$

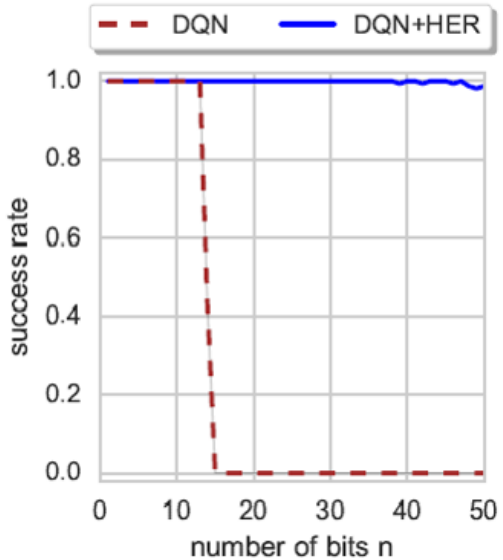


图 9: Bit Flipping 实验结果

$$s{:}0100101010 \xrightarrow[Reward{:}\text{-}1]{Action{:}0} 1100101010 \longrightarrow \cdots$$
$$g{:}1101010100$$

- 在这样的环境下，尤其是比特位数 $n>40$ 时，经典的强化学习算法如 DQN 很难学习到从起始状态到终点状态的路径；因为它探索到的 reward 值基本都是-1；

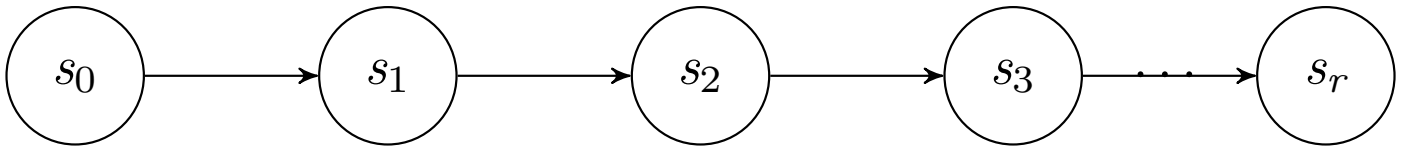
- 在这样的环境下，尤其是比特位数 n>40 时，经典的强化学习算法如 DQN 很难学习到从起始状态到终点状态的路径；因为它探索到的 reward 值基本都是-1；
- 在这样的环境中，一个比较好的解决方法是用一个 shaped reward function，以此来给予 agent 更多到达终点的信息；如：$r_g(s, a) = -||s - g||^2$；

- 在这样的环境下，尤其是比特位数 n>40 时，经典的强化学习算法如 DQN 很难学习到从起始状态到终点状态的路径；因为它探索到的 reward 值基本都是-1；
- 在这样的环境中，一个比较好的解决方法是用一个 shaped reward function，以此来给予 agent 更多到达终点的信息；如：$r_g(s,a) = -||s-g||^2$；
- 在更为复杂的环境中，设置一个有效的合适的回报函数有时会比较困难；因此本文目的是设计一个更为普遍的通用于多种环境中的方法，而不要有针对环境的特定领域知识。

- 训练 agent 能够学会到达多个目标点；

$$f_g\left((x, y)\right) = \mathrm{sgn}(s = g)$$

- 使用 Universal Value Function Approximators，在训练 policy 和 value function 的过程中，将 state s 和目标 g 作为输入；



Algorithm: 事后经验回放算法

A episode: $s_0, s_1, s_2, ..., s_T$;
foreach *Time-stem* t **do**
    使用现有的策略 A，从中选取一个 actiona_t，即 $a_t \leftarrow \pi_b(s_t||g)$;
    计算 reward 值：$r_t = r(s_t, a_t, g)$;
    Experience Replay：将状态转移 $s_t \rightarrow s_(t+1)$ 以 $(s_t||g, a_t, r_t, s_(t+1)||g)$ 形式保存到 replay buffer 中;
利用 replay buffer 进行更新已有策略 A;

目录

- 采用在现有的硬件机器人的基础上的控制环境
- 可用一个物理引擎模拟机器人的控制和反馈

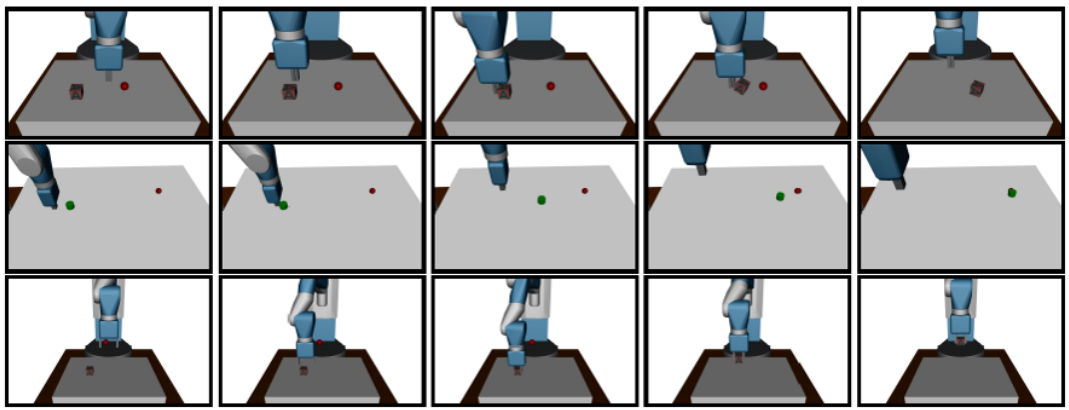


图 10: 三种不同的任务：pushing（第一行），sliding（第二行），pick and place（第三行）

Pushing A box is placed on a table in front of the robot and the task is to move it to the target location on the table. The robot fingers are locked to prevent grasping. The learned behavior is a mixture of pushing and rolling.

Sliding A puck is placed on a long slippery table and the target position is outside of the robot's reach so that it has to hit the puck with such a force that it slides and then stops in the appropriate place due to friction.

Pick-and-place The target position is in the air and the fingers are not locked.

States The state of the system is represented in the MuJoCo physics engine and consists of angles and velocities of all robot joints as well as positions, rotations and velocities (linear and angular) of all objects.

Goals Goals describe the desired position of the object (a box or a puck depending on the task) with some fixed tolerance of ϵ, i.e. , $f_g(s) = \text{sgn}(g - s_{object} \leq \epsilon)$.

Rewards Use binary and sparse rewards $\{-1, 0\}$;

State-goal distributions For all tasks the initial position of the gripper is fixed, while the initial position of the object and the target are randomized;

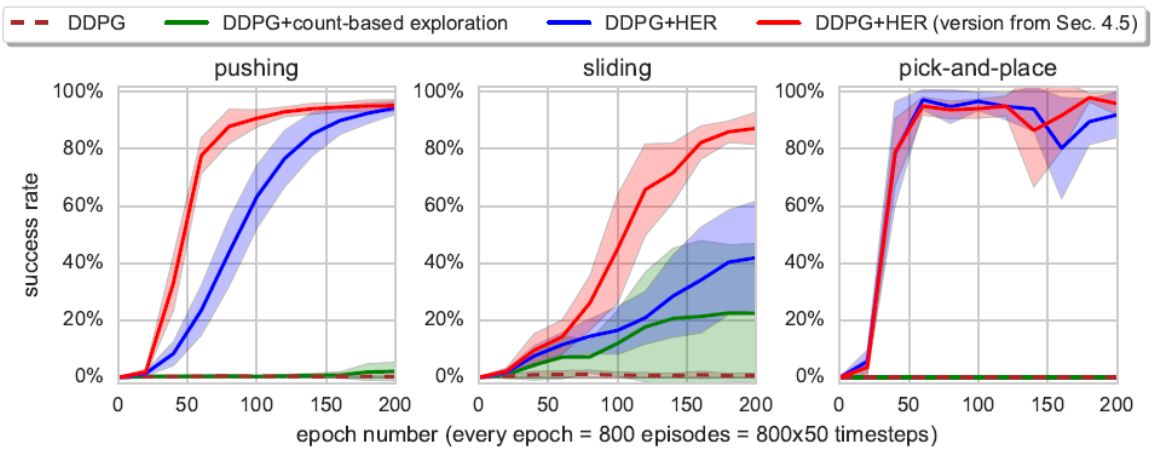

DDPG
DDPG+count-based exploration
DDPG+HER
DDPG+HER (version from Sec. 4.5)
pushing
sliding
pick-and-place
success rate
100%
80%
60%
40%
20%
0%
0
50
100
150
200
epoch number (every epoch = 800 episodes = 800x50 timesteps)

图 11: 多目标学习曲线

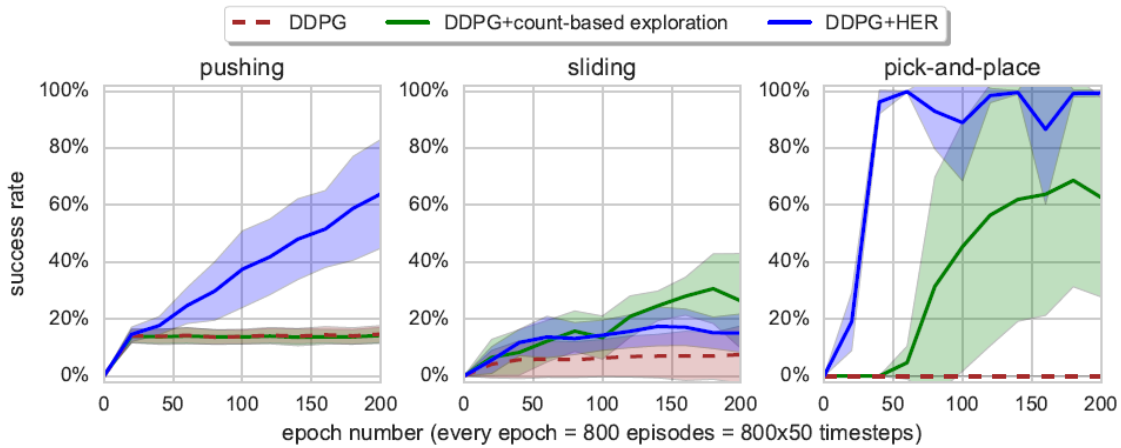

DDPG
DDPG+count-based exploration
DDPG+HER
pushing
sliding
pick-and-place
success rate
100%
80%
60%
40%
20%
0%
0
50
100
150
200
epoch number (every epoch = 800 episodes = 800x50 timesteps)

图 12: 单目标学习曲线

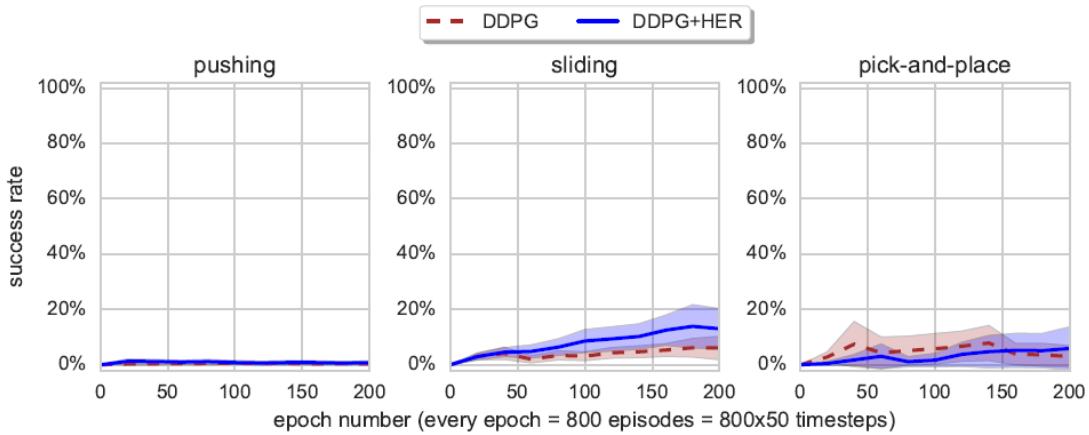


图 13: 变更奖励函数后的学习曲线，其中 $r(s, a, g) = -|g - s'_{object}|^2$

目录

结论

1. 提出了 HER 算法，能够将强化学习方法应用到稀疏回报值的问题中；
2. HER 算法具有很强的通用性，能够结合任意的 off-policy RL 算法，在实验中结合的是 DQN 和 DDPG 方法；
3. 在实验中，将 HER 算法应用到让机器臂完成不同的移动物品的任务上，展示了突出的效果。

第 III 部分

强化学习在游戏中的应用

目录

目录

参考文献

[Leonetti et al., 2016] Leonetti, M., Iocchi, L., and Stone, P. (2016).
A synthesis of automated planning and reinforcement learning for efficient, robust decision-making.
Artificial Intelligence, 241:103–130.

[Sutton, 1990] Sutton, R. S. (1990).
Integrated architectures for learning, planning, and reacting based on approximating dynamic programming.
In *Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990*, pages 216–224. Morgan Kaufmann.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998).
Reinforcement learning: an introduction.
Adaptive computation and machine learning. MIT Press, Cambridge, Massachusetts, London, England, 2 edition.

東南大學
SOUTHEAST UNIVERSITY

感谢观看！

Q & A