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。

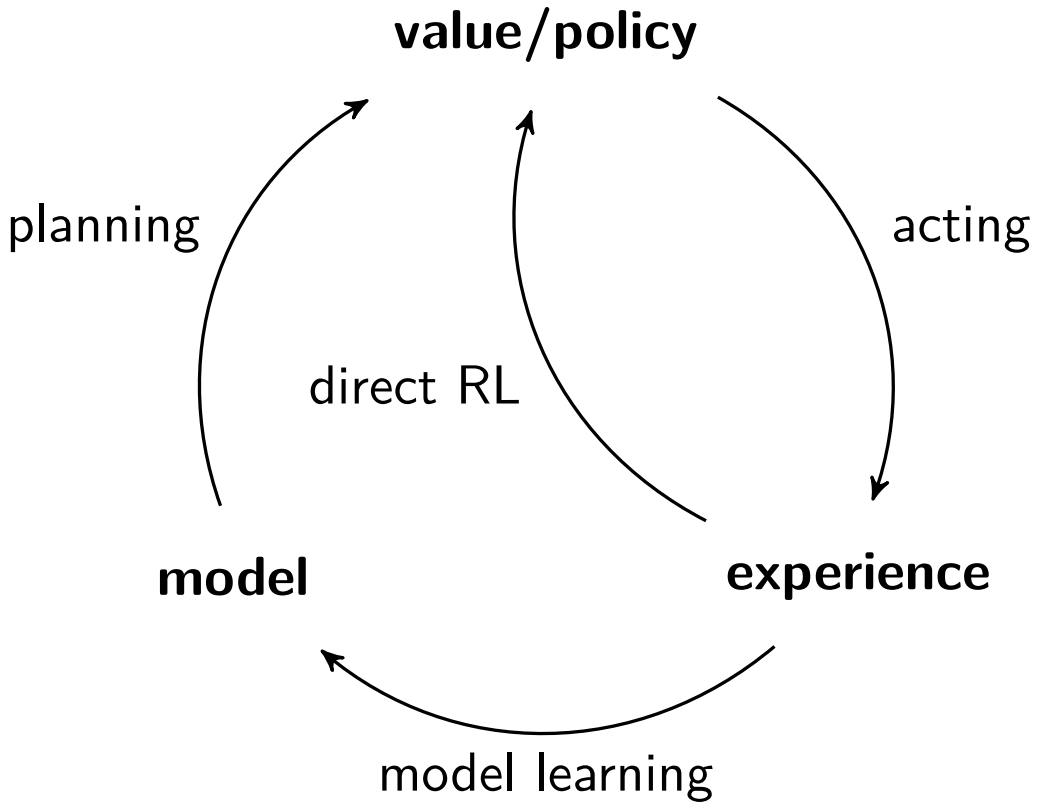
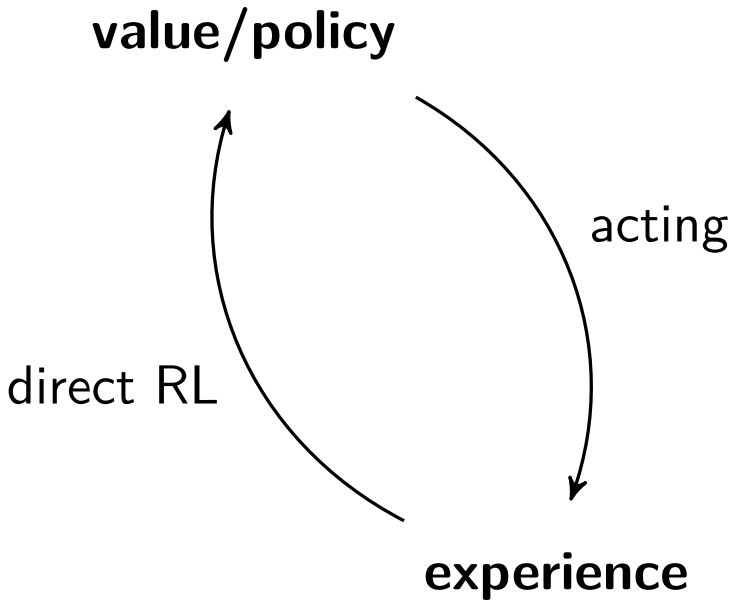


图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]

马尔科夫决策过程
规划与学习



Model-Free RL

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

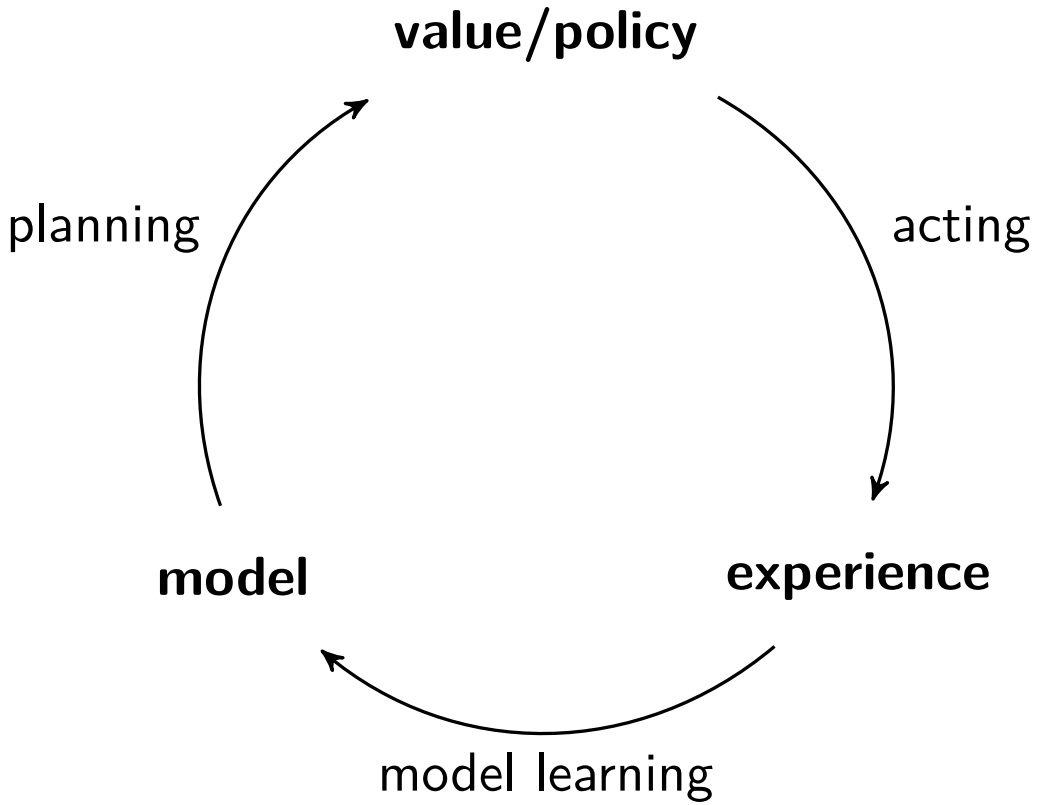


图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]

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

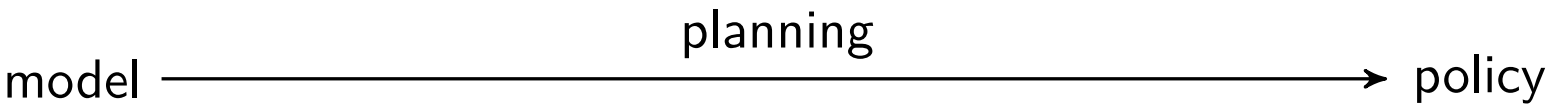
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1},a) - Q(s_t,a_t) \right]$$

Algorithm: Q-Learning

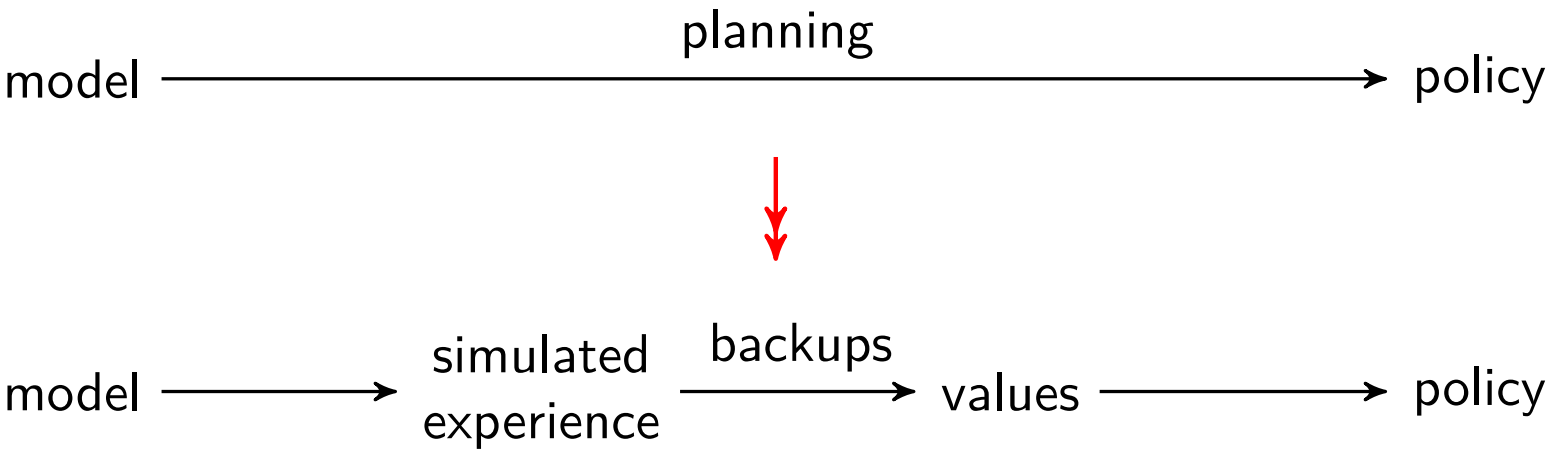
随机初始化 $Q(s, a)$;
foreach 每个周期（*episode*）**do**
    初始化 s;
    foreach 周期内的每一步，直到 s 为终止状态 **do**
        利用 Q 函数中获得的策略（例如 ϵ-greedy）选择状态 s 时采取的策略;
        执行 a，观察 s' 和 r';
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$;

基于强化学习的规划
Q-Planning

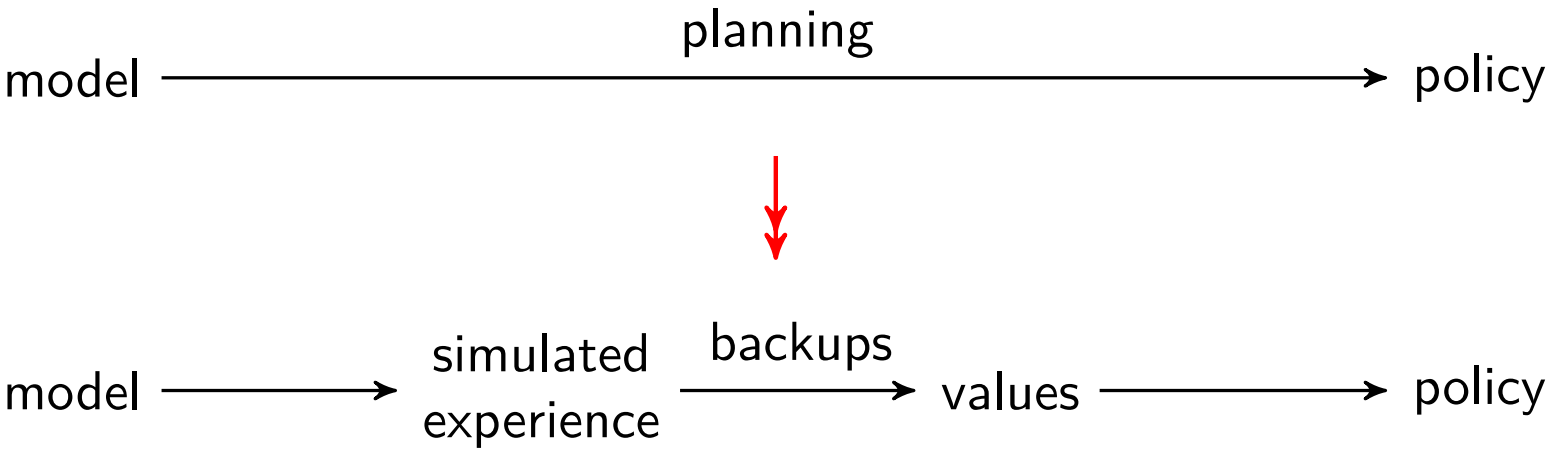
基于强化学习的规划
Q-Planning



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

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

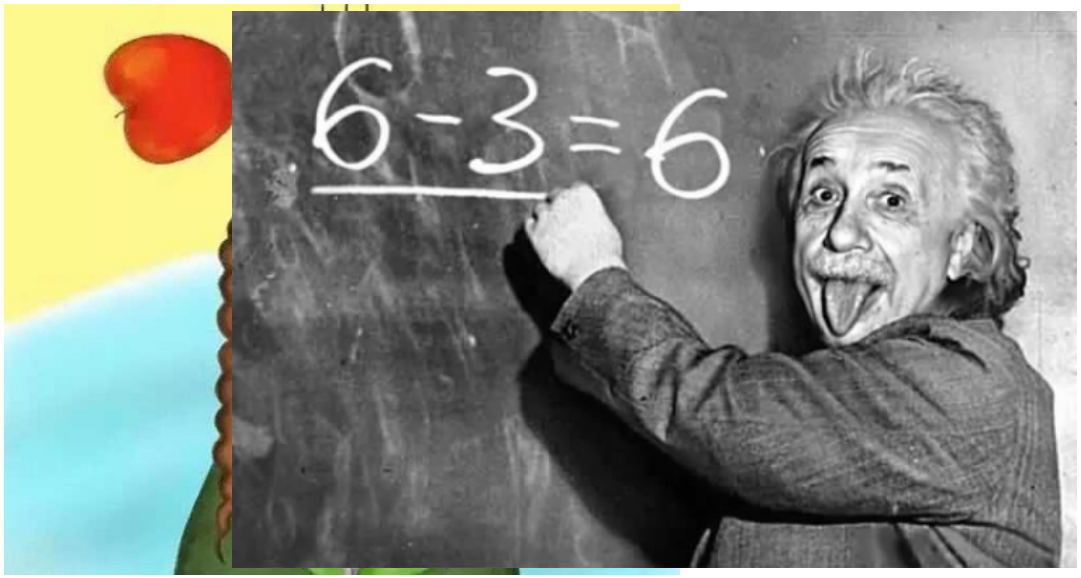
–George E.P. Box, Robustness in the strategy of scientific model building

All models are wrong, but some are useful.

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

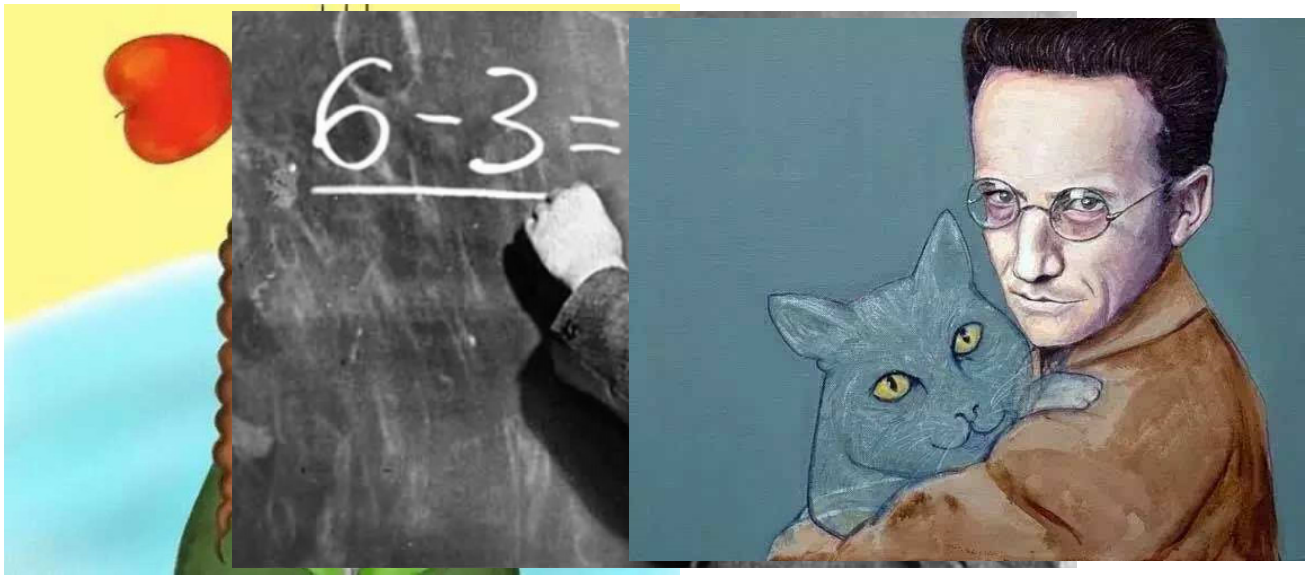
All models are wrong, but some are useful.

–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

Dyna
主要思想和结构

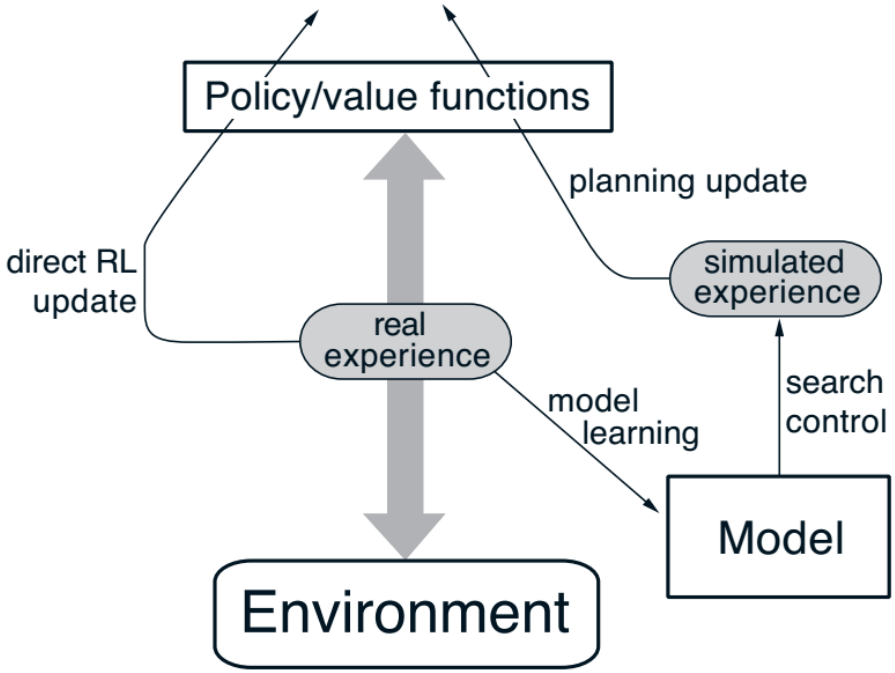


图 2: Dyna agent 的一般结构

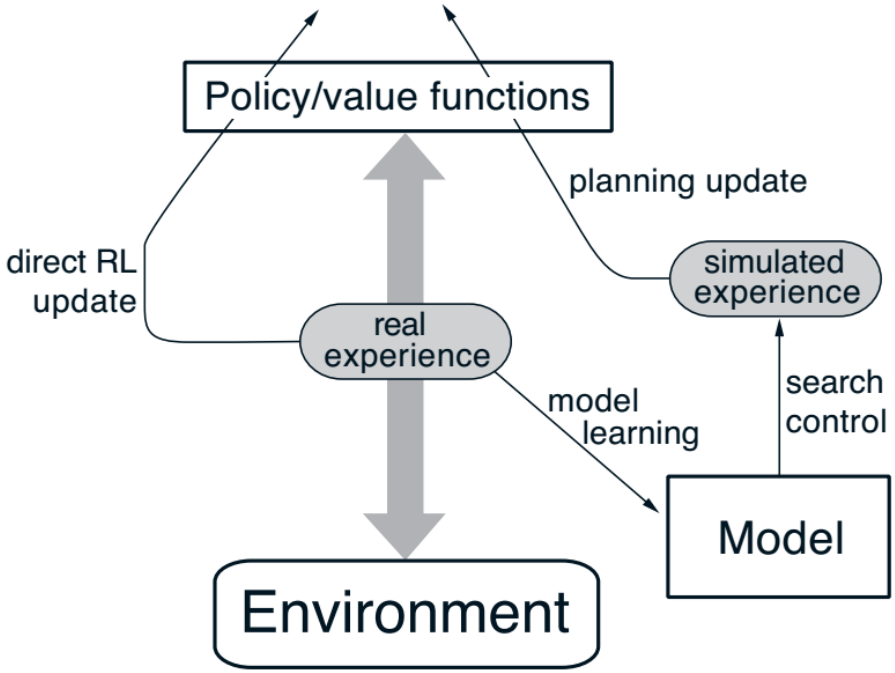


图 2: Dyna agent 的一般结构

- 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

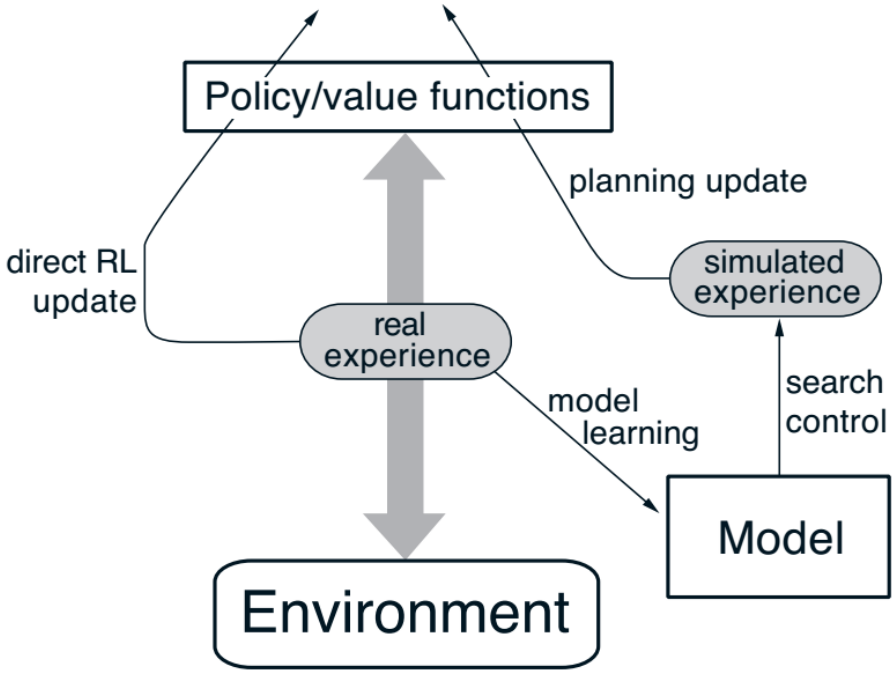


图 2: Dyna agent 的一般结构

- 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*;

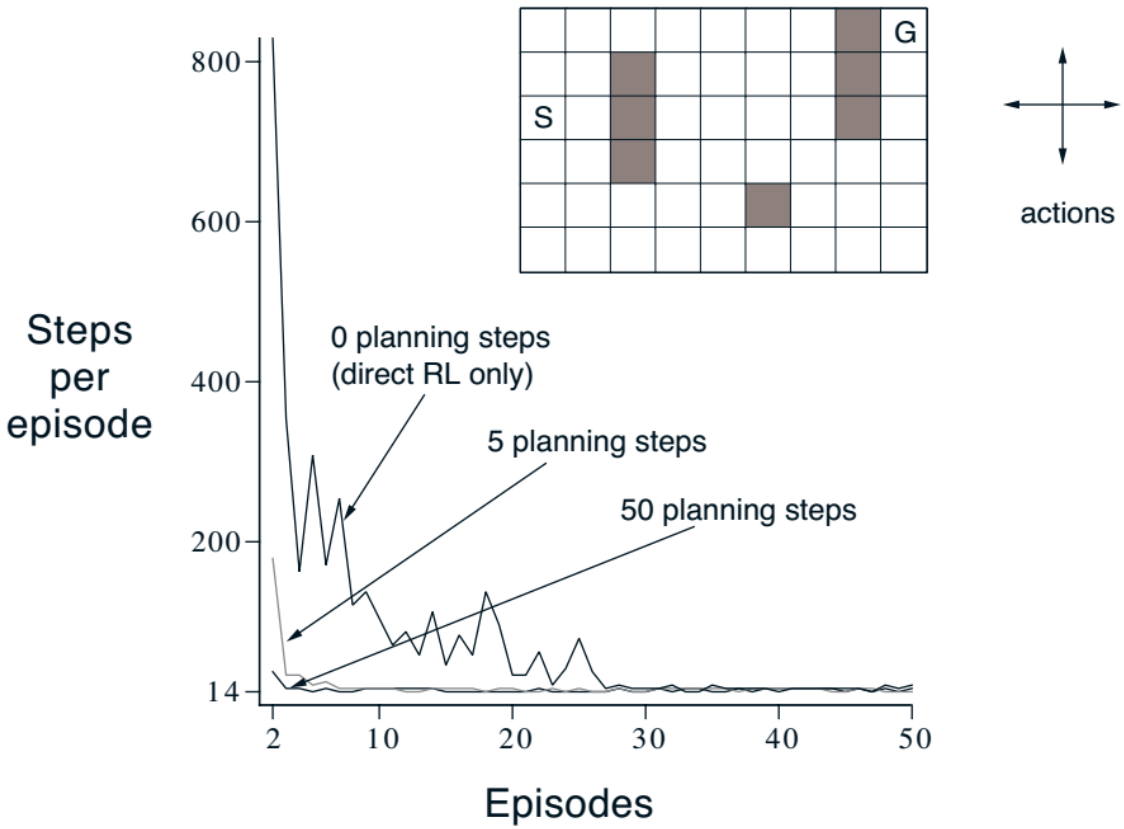


图 3: Dyna-Q 在一个迷宫问题上的实验，agent 需要从 S 走到 G，$r \to \{0, 1\}$。对于所有的 N，第一次探索是一样的，都是大约 1700 步；之后，Dyna-Q 和 Q-Learning 的差别开始显现

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

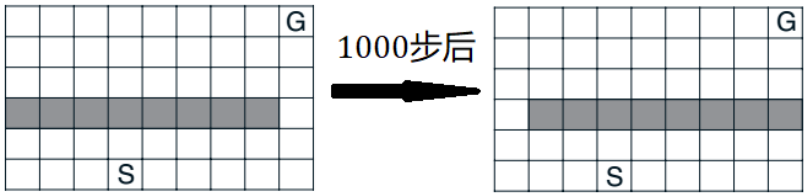


图 4: 1000 步之后，迷宫出现变化，左侧墙壁打开，右侧通路封闭

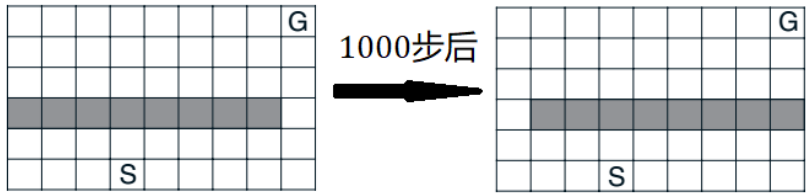


图 4: 1000 步之后，迷宫出现变化，左侧墙壁打开，右侧通路封闭

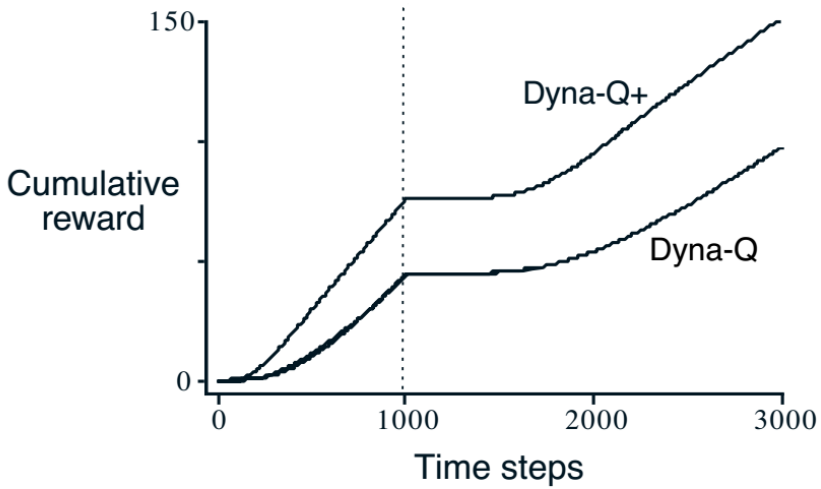


图 5: 由于 Dyna 算法包括了对模型的更新，agent 在一段时间后完成了对环境的重新建模并找到了正确的路径。图中 Q+ 是加强了探索能力的 Q-Learning，用 $r + \kappa\sqrt{n}$ 为奖励函数，n 为 s 状态下 a 连续未被选中的次数

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；

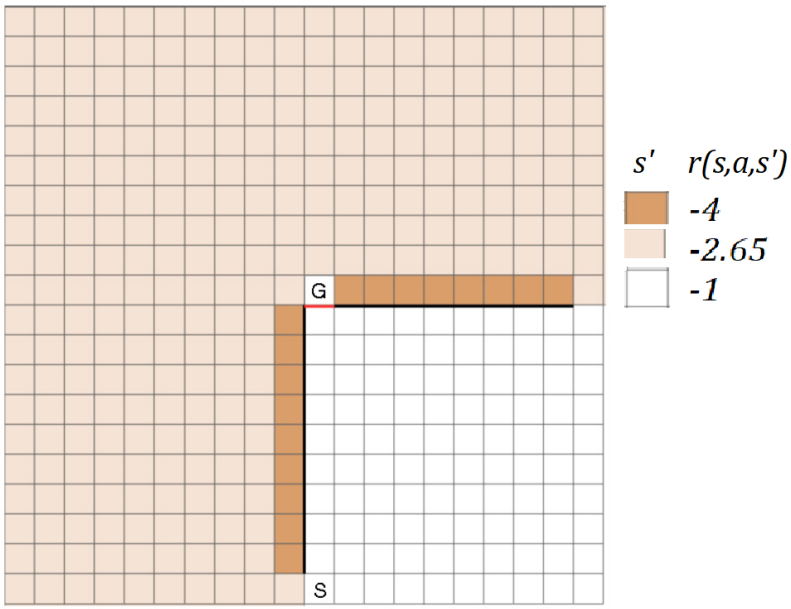
建模（Modeling） 对环境 $D = \langle S, A, P, r, \gamma \rangle$ 进行建模，建立从环境状态到模型状态的函数 $o : S \to S_m$，得 $D_m = \langle S_m, A, P_m \rangle$，其中

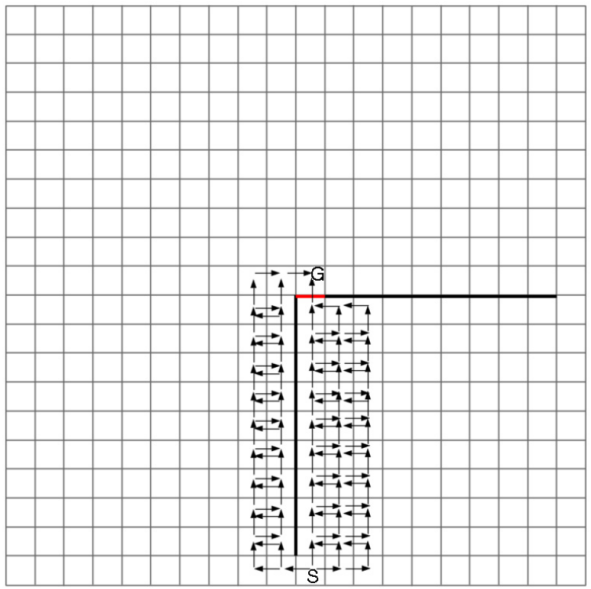
规划（Planning） 利用规划工具，以一定的冗余度计算可行的方案。规划是在一个假设的模型上执行的，因此不能保证所得的方案是最优甚至可行的；

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；

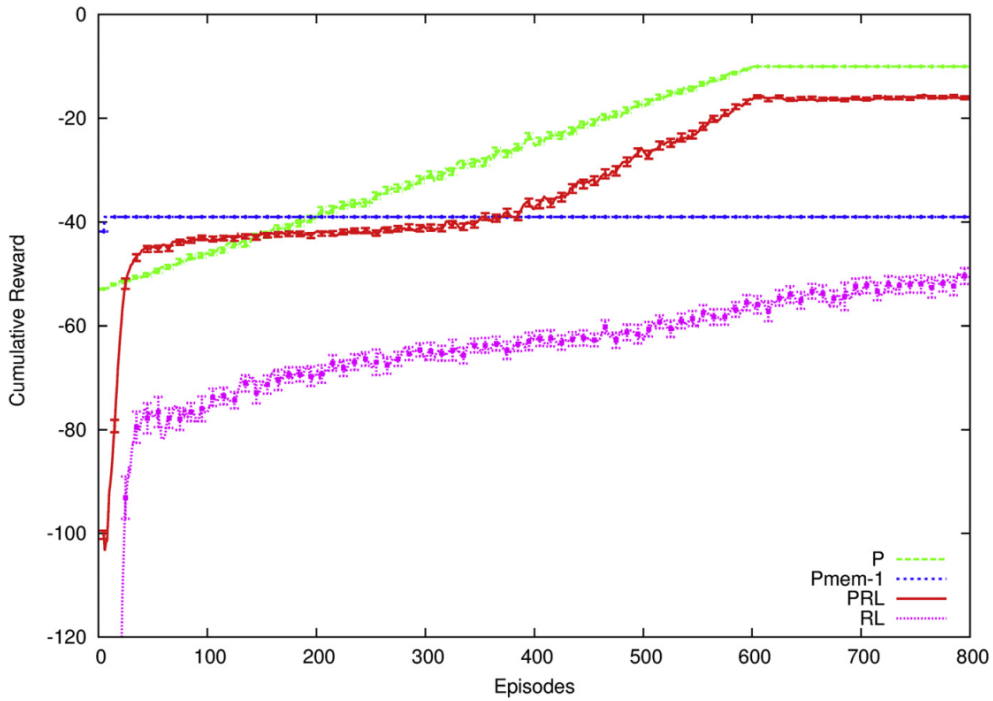

Cumulative Reward
Episodes
0
-20
-40
-60
-80
-100
-120
0
100
200
300
400
500
600
700
800
P
Pmem-1
PRL
RL

图 6: 实验结果

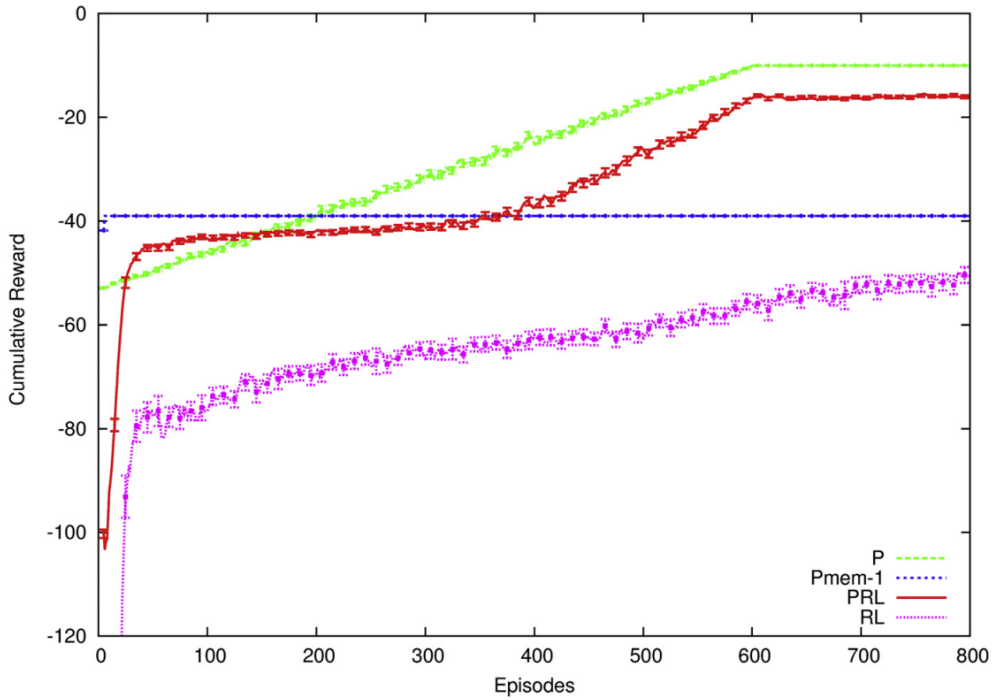


图 6: 实验结果

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。

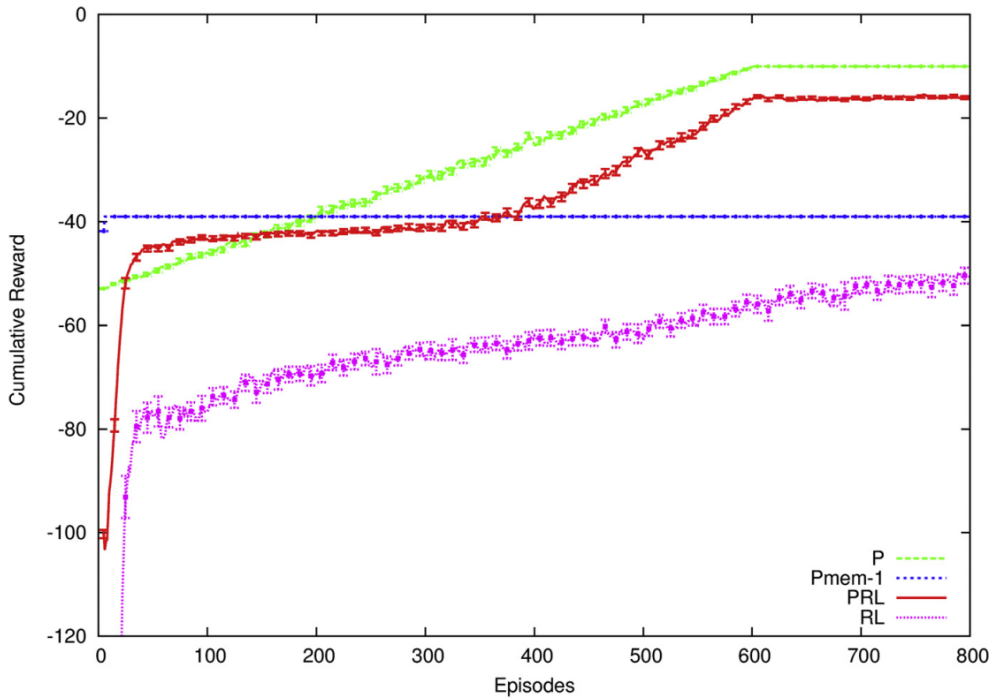


图 6: 实验结果

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。

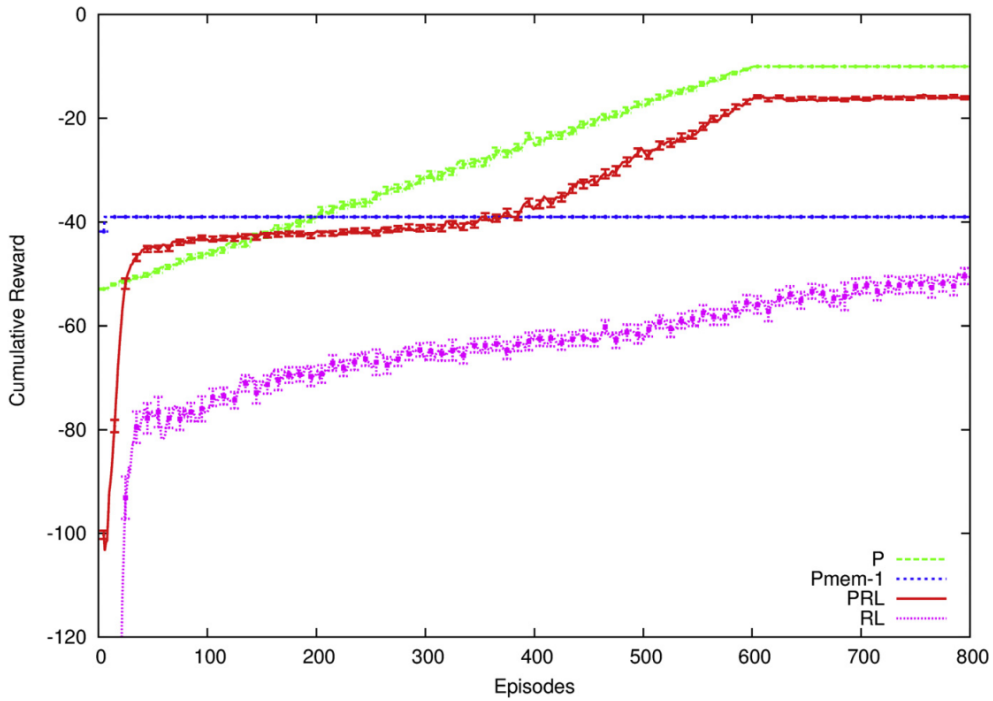


图 6: 实验结果

PRL agent 学到了最优策略，并在环境变化后选择了更优的策略。

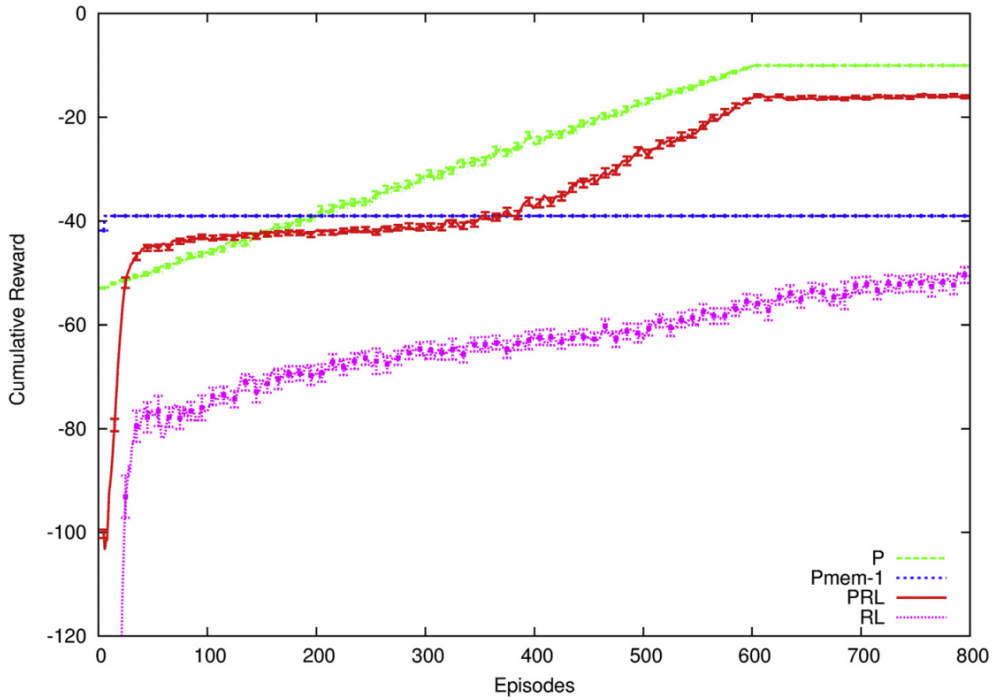


图 6: 实验结果

Pmem-n 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。

目录

東南大學
SOUTHEAST UNIVERSITY

第 II 部分

DQN 与 DRQN

目录

背景知识
Atari 2600 games

- 雅达利（1972，美国）
- 主要产品为 Atari 2600 游戏主机
- 游戏环境 ε
  - 屏幕输出 x_t
  - agent 执行的动作 a_t
  - 游戏分数的变化 r_t
  - 屏幕输出与动作序列表示状态 s_t

$$s_t = x_1, a_1, x_2, a_2, \ldots, a_{t-1}, x_t$$

图 7: Atari 2600

  - 期望累积奖励

$$R_t = \sum_{t=t'}^{T} \gamma^{t-t'} r_{t'}$$

图 8: Atari 2600 上的五个游戏，分别是 Pong，Breakout，Space Invader，Seaquest，Beam Rider

- $Q^{\pi}(s,a)$ 定义为从状态 s 出发，执行动作 a 后，再使用策略 π 所得到的累计奖励

$$Q^{\pi}(s,a) = E[R_t | s_t = s, a_t = a, \pi]$$

- $Q^{*}(s,a)$ 定义为从状态 s 出发，执行动作 a 后，再使用最优策略 π^{*} 得到的最优累积奖励

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

- agent 的目标就是找到最优策略 π^{*}
- 如果下一个状态 s' 和其所对应的所有动作 a' 的最优值函数 $Q^{*}(s',a')$ 已知，那么满足贝尔曼等式

$$Q^{*}(s,a) = \mathbb{E}_{s' \sim \epsilon}[r + \gamma \max_{a'} Q^{*}(s',a') | s,a]$$

- Q-Learning 的定义为：

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Algorithm: Q 学习算法

Input: 状态空间 S，动作空间 A，折扣率 γ，学习率 α

Output: 策略 $\pi(s) = \arg\max_{a \in |A|} Q(s, a)$

随机初始化 $Q(s, a)$;

$\forall s, \forall a, \pi(a|s) = \frac{1}{|A(s)|}$;

repeat

  初始化起始状态 s;

  repeat

    在状态 s，选择 $a \in \pi^{\epsilon}(s)$;

    执行动作 a，得到即时奖励 r 和新状态 s';

    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$;

    $s \leftarrow s'$;

  until s 为终止状态;

until $\forall s, a$，$Q(s, a)$ 收敛;

目录

- 游戏中的状态数量太多，无法遍历和存储所有的值函数

- 游戏中的状态数量太多，无法遍历和存储所有的值函数
- 通常使用一个含参函数来估计值函数

- 游戏中的状态数量太多，无法遍历和存储所有的值函数
- 通常使用一个含参函数来估计值函数
- 强化学习很难从高维数据（例如视频、声音）中直接学习控制策略。许多应用于视频和声音领域的强化学习策略都是基于手工指定的特征

- 游戏中的状态数量太多，无法遍历和存储所有的值函数
- 通常使用一个含参函数来估计值函数
- 强化学习很难从高维数据（例如视频、声音）中直接学习控制策略。许多应用于视频和声音领域的强化学习策略都是基于手工指定的特征
- 深度卷积神经网络可以直接从视频等高维信息中学习抽象的高层特征

- 深度学习方法要求大量人工标记的训练数据。强化学习算法通常是从数值奖励中学习的，这些奖励反馈通常是稀疏的、有噪声、有延迟的。

- 深度学习方法要求大量人工标记的训练数据。强化学习算法通常是从数值奖励中学习的，这些奖励反馈通常是稀疏的、有噪声、有延迟的。
- 大多数深度学习方法假定数据样本是独立的，然而强化学习算法的数据样本通常是具有高度相关性的。

- 深度学习方法要求大量人工标记的训练数据。强化学习算法通常是从数值奖励中学习的，这些奖励反馈通常是稀疏的、有噪声、有延迟的。
- 大多数深度学习方法假定数据样本是独立的，然而强化学习算法的数据样本通常是具有高度相关性的。
- 在强化学习中，数据样本的分布随着学习到的策略的改变而改变，而深度学习中假定数据样本的分布是固定的

- 训练一个参数为 θ 的卷积神经网络来估计值函数
- 使用经验回放来减少数据间的相关性，平缓数据分布的变化
- 使用两个网络 target Q 以及 Q 网络，两个网络结构一样，参数不同。target Q 网络的参数 θ_i^-，是 Q 网络的参数 θ_i 的历史版本。Q 网络的作用是在每次迭代 i 中改变参数 θ_i，估计值函数。target Q 网络的作用是给出每次迭代 i 中的估计目标

$$y_i = E_(s' \in \epsilon)[r + \gamma \max_{a'} target\_Q(s', a', \theta_i^-)]$$

- 每次迭代 i 的损失函数可以表示为

$$L_i(\theta_i) = E_{s,a,r,s'}[(y_i - Q(s, a|\theta_i))^2]$$

- target Q 网络使用 Q 网络的参数 θ_i 的历史版本能够平缓数据分布的变化
- 第 i 次迭代的损失函数为

$$L_i(\theta_i) = E_{s,a,r,s'}[(r + \gamma \max_{a'} target\_Q(s', a', \theta_i^-) - Q(s, a, \theta_i))^2]$$

- 更新梯度为

$$\nabla_{\theta_i} L_i(\theta_I) = E_{s,a,r,s'}[(r + \gamma \max_{a'} target\_Q(s', a'|\theta_i^-) - Q(s, a|\theta_i))\nabla_{\theta_i} Q(s, a|\theta_i)]$$

- 与 Q-learning 相比

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

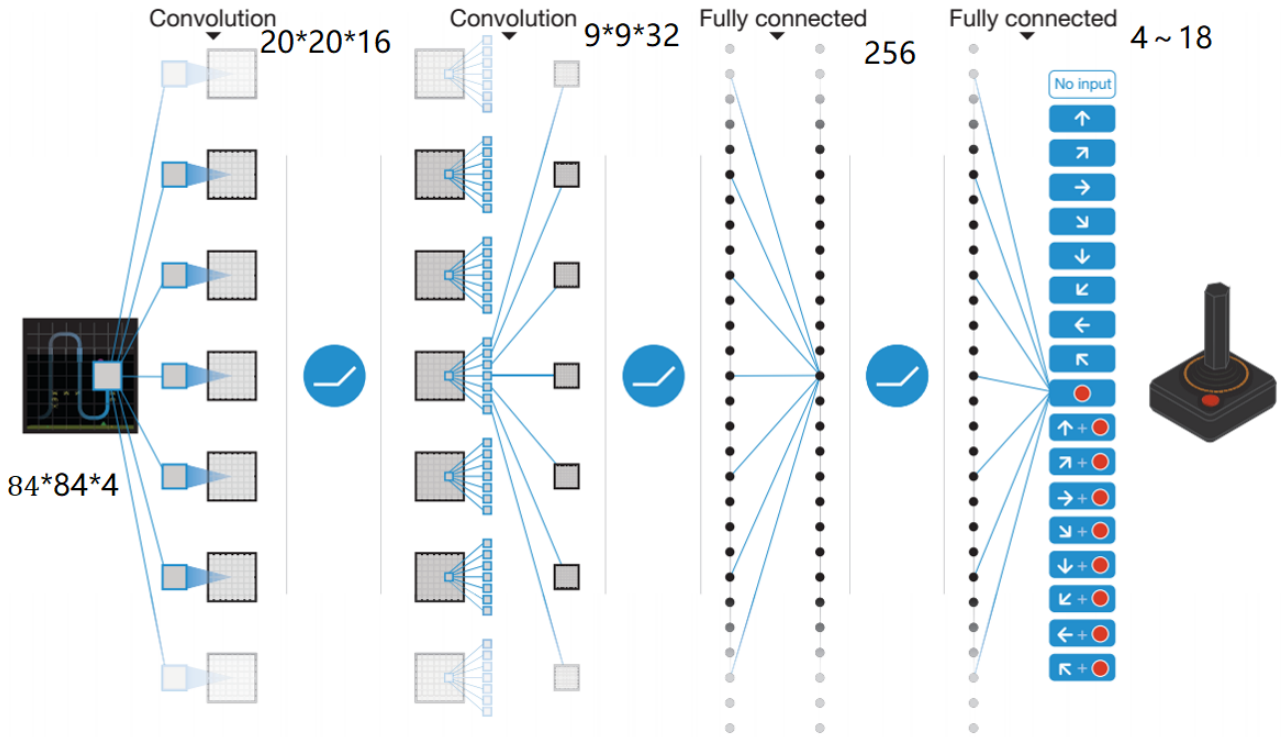

Convolution
20*20*16
Convolution
9*9*32
Fully connected
256
Fully connected
4 ~ 18
No input
84*84*4

图 9:

目录

DQN
经验回放

- 维持一个数量有限的经验池 $D = e_1, \ldots, e_t$
- 每次 agent 在时间 t 行动后获得的经验 $e_t = (s_t, a_t, r_t, s_{t+1})$ 存入经验池 D 中
- 在训练时，随机地从经验池 D 中选择经验 e_t 来训练网络

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1$,T **do**

With probability ε select a random action a_t

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

Execute action a_t in emulator and observe reward r_t and image x_{t+1}

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

Every C steps reset $\hat{Q} = Q$

End For

End For

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Table 1: The upper table compares average total reward for various learning methods by running an ϵ-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an ϵ-greedy policy with $\epsilon = 0.05$.

图 11:

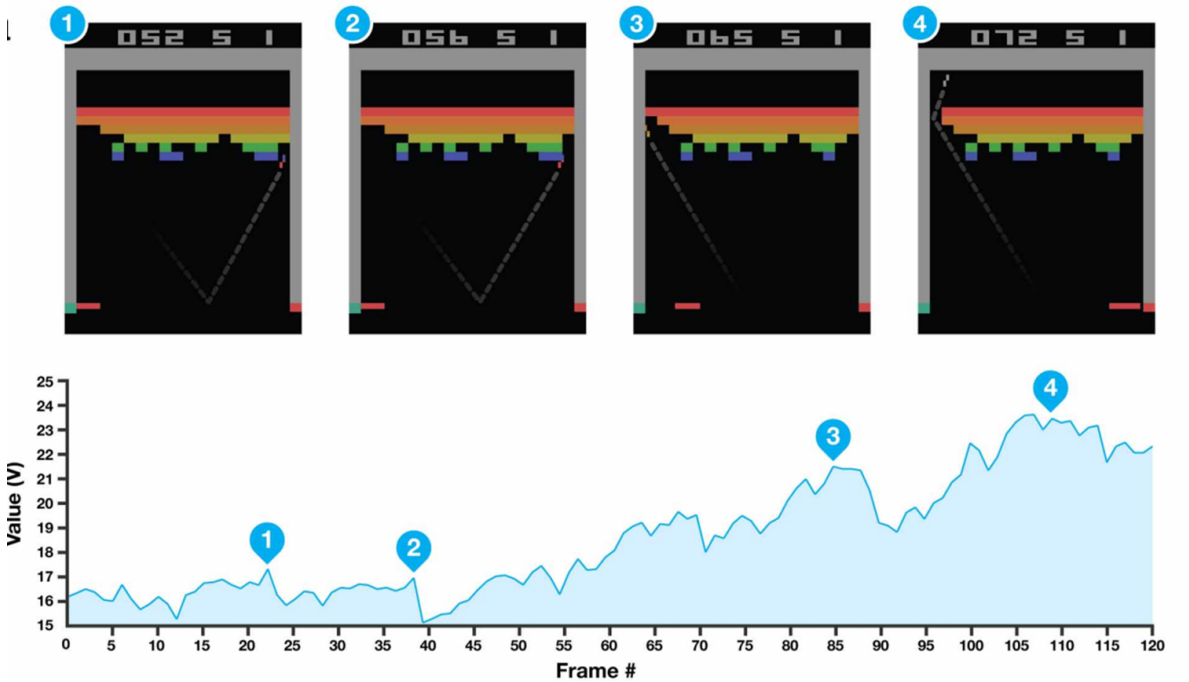

1
052 5 1
2
056 5 1
3
065 5 1
4
072 5 1
Value (V)
Frame #
25
24
23
22
21
20
19
18
17
16
15
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120

图 12:

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