



東南大學
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

第 I 部分强化学习与规划

背景知识回顾

马尔科夫决策过程

Q-Learning

两种规划与学习结合的方法

Dyna

DARLING

事后经验回放与稀疏奖励下的规划

第 II 部分强化学习在 FPS 游戏中的应用

Atari 游戏与 DQN



東南大學
SOUTHEAST UNIVERSITY

第 I 部分

强化学习与规划

背景知识回顾

马尔科夫决策过程
Q-Learning

两种规划与学习结合的方法

事后经验回放与稀疏奖励下的规划

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

$S \times A \times S \rightarrow [0, 1]$, 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

$S \times A \times S \rightarrow \mathbb{R}$, 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) $\gamma \in [0, 1]$

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

$S \times A \times S \rightarrow [0, 1]$, 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

$S \times A \times S \rightarrow \mathbb{R}$, 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

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





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；

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；

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。

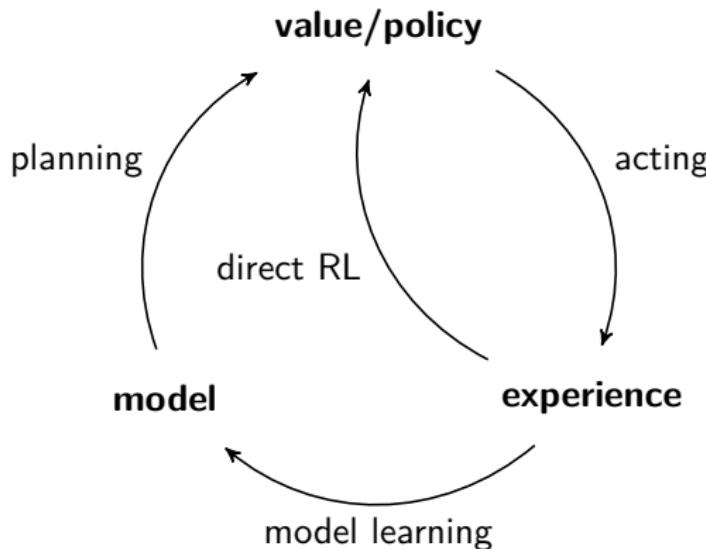
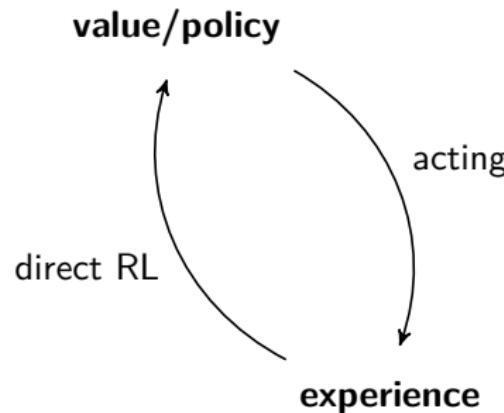


图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]



Model-Free RL

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

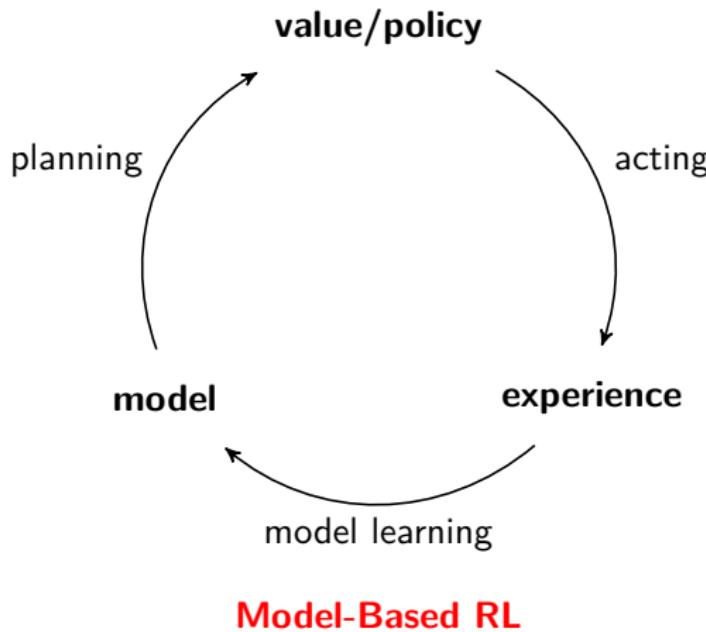


图 1: 学习、规划和执行之间的关系 [Sutton and Barto, 1998]

- ▶ Q-Learning 是一种 off-policy 的时序差分学习方法

- ▶ 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

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

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

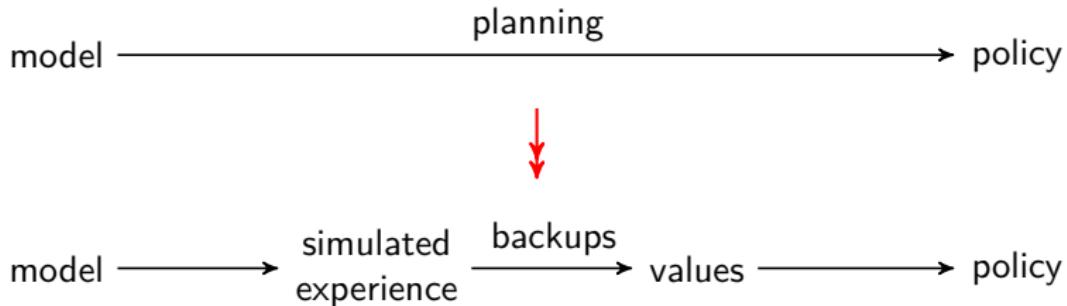
- ▶ Q-learning 的定义为

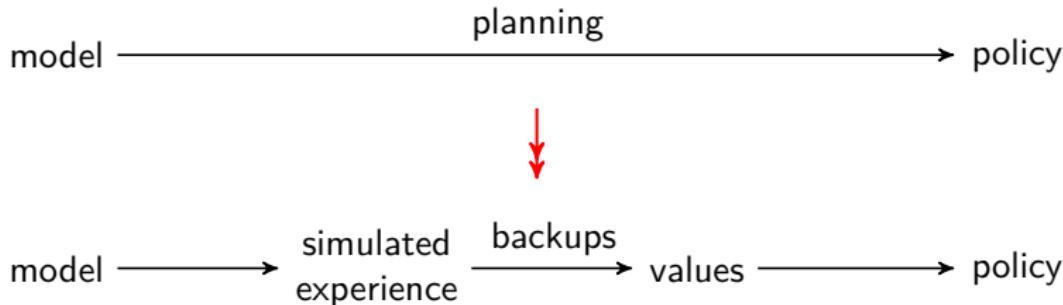
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Algorithm: Q-Learning

- 1 随机初始化 $Q(s, a)$;
 - 2 **foreach** 每个周期 (*episode*) **do**
 - 3 初始化 s ;
 - 4 **foreach** 周期内的每一步，直到 s 为终止状态 **do**
 - 5 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；
 - 6 执行 a , 观察 s' 和 r ;
 - 7
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)];$$
-







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.

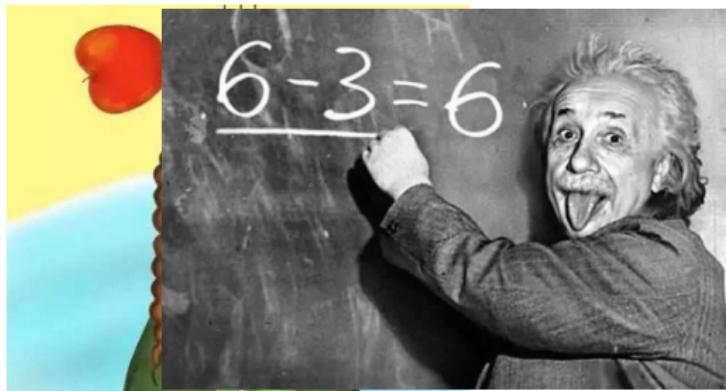
—George E.P. Box, Robustness in the strategy of scientific model building





All models are wrong, but some are useful.

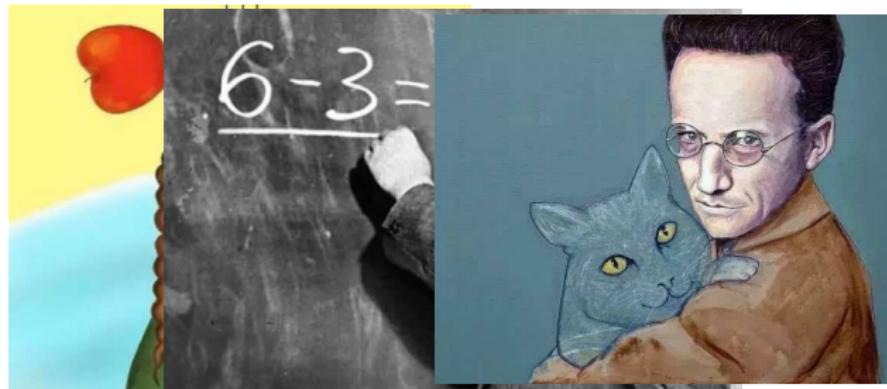
—George E.P. Box, Robustness in the strategy of scientific model building





All models are wrong, but some are useful.

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



背景知识回顾

两种规划与学习结合的方法

Dyna
DARLING

事后经验回放与稀疏奖励下的规划

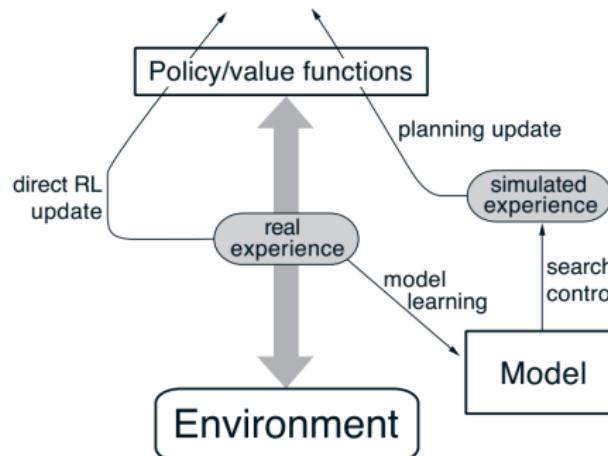


图 2: Dyna agent 的一般结构

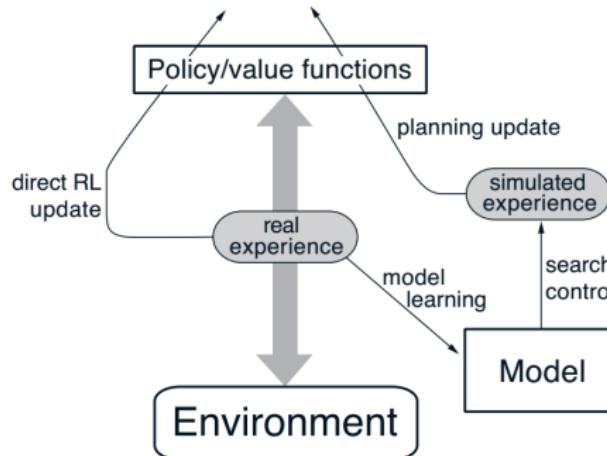


图 2: Dyna agent 的一般结构

- ▶ 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**

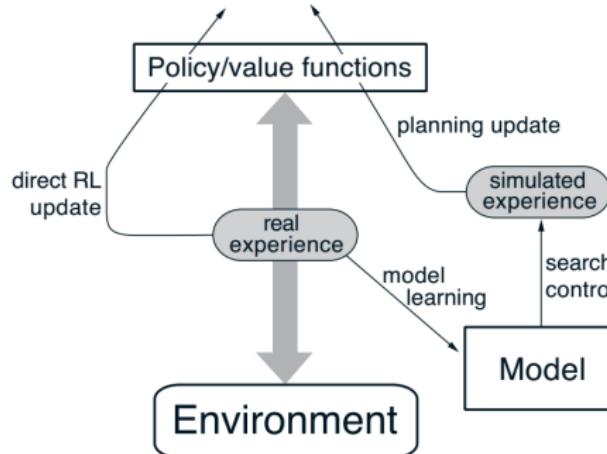


图 2: Dyna agent 的一般结构

- ▶ 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

- 1 对任意的 $s \in S$, $a \in A(s)$, 初始化 $Q(s, a)$ 和 $Model(s, a)$;
- 2 **repeat**
- 3 对当前状态 s , 根据 Q 表选择 $a \in A(s)$ 并执行, 得 s' 和 r ;
- 4 更新 $Q(s, a)$;
- 5 用 s', r 更新 $Model(s, a)$;
- 6 **repeat** // 在 $Model$ 上规划并更新 Q
- 7 s 为任意观察到的状态, a 为 s 上进行过的任意操作;
- 8 $s', r \leftarrow Model(s, a)$;
- 9 $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$;
- 10 **until** N 次循环;
- 11 **until** stop;

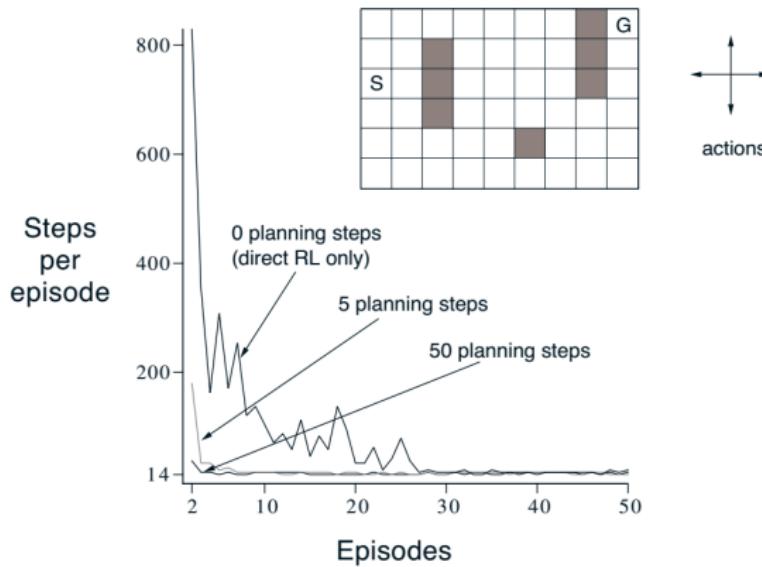


图 3: Dyna-Q 在一个迷宫问题上的实验, agent 需要从 S 走到 G, $r \rightarrow \{0, 1\}$ 。对于所有的 N , 第一次探索是一样的, 都是大约 1700 步; 之后, Dyna-Q 和 Q-Learning 的差别开始显现

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

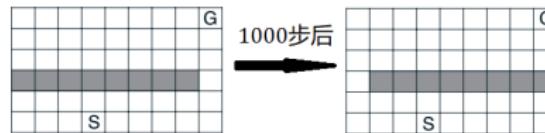


图 4: 1000 步之后, 迷宫出现变化, 左侧墙壁打开, 右侧通路封闭

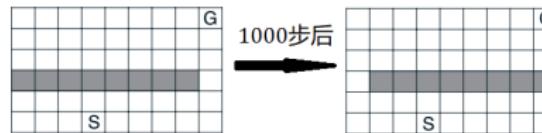


图 4: 1000 步之后, 迷宫出现变化, 左侧墙壁打开, 右侧通路封闭

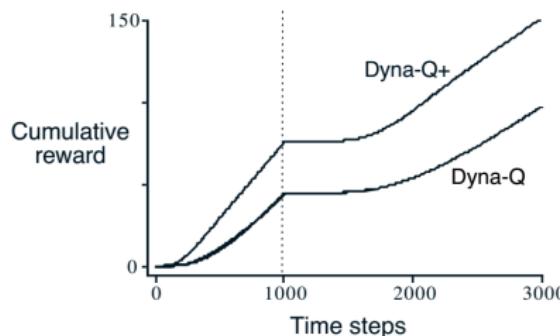


图 5: 由于 Dyna 算法包括了对模型的更新, agent 在一段时间后完成了对环境的重新建模并找到了正确的路径。图中 Q+ 是加强了探索能力的 Q-Learning, 用 $r + \kappa\sqrt{n}$ 为奖励函数, n 为 s 状态下 a 连续未被选中的次数

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 $o : S \rightarrow S_m$ ，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 $o : S \rightarrow S_m$ ，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 $o : S \rightarrow S_m$ ，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；

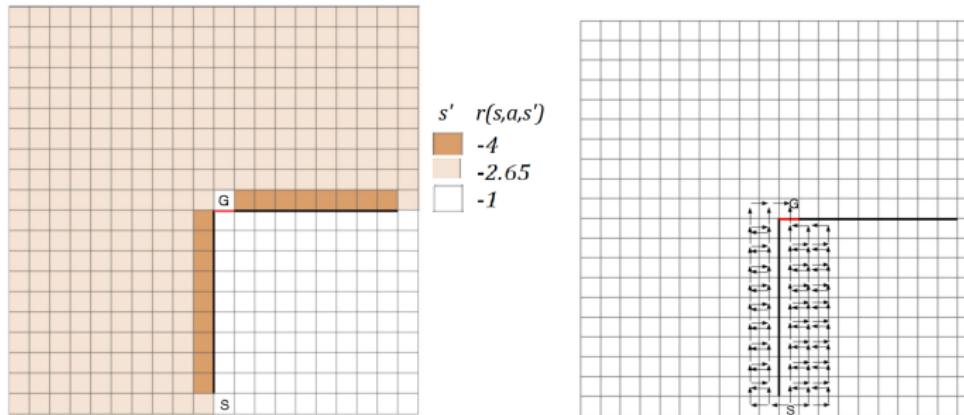
建模 (Modeling) 对环境 $D = \langle S, A, P, r, \gamma \rangle$ 进行建模，建立从环境状态到模型状态的函数 $o : S \rightarrow S_m$ ，得 $D_m = \langle S_m, A, P_m \rangle$ ，
其中

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-redundant 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

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

- ▶ 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

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

- ▶ 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

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

- ▶ 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

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

- ▶ 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

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

- ▶ 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

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

- ▶ 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；

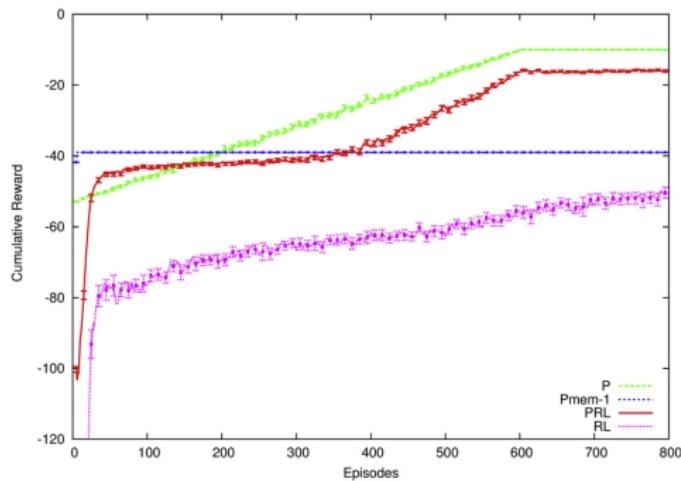


图 6: 实验结果

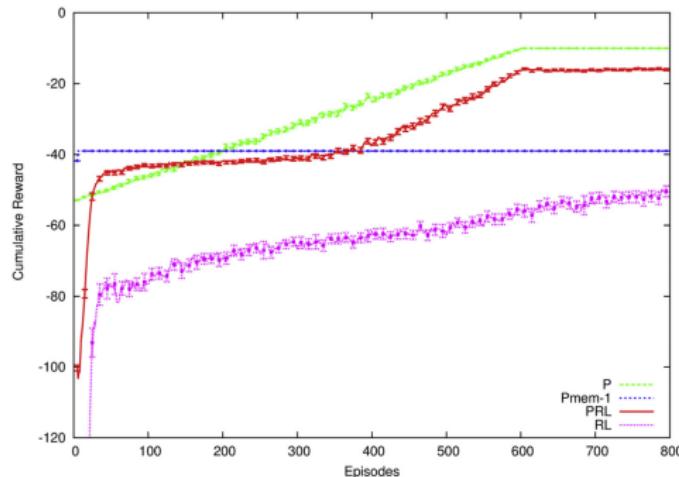


图 6: 实验结果

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。

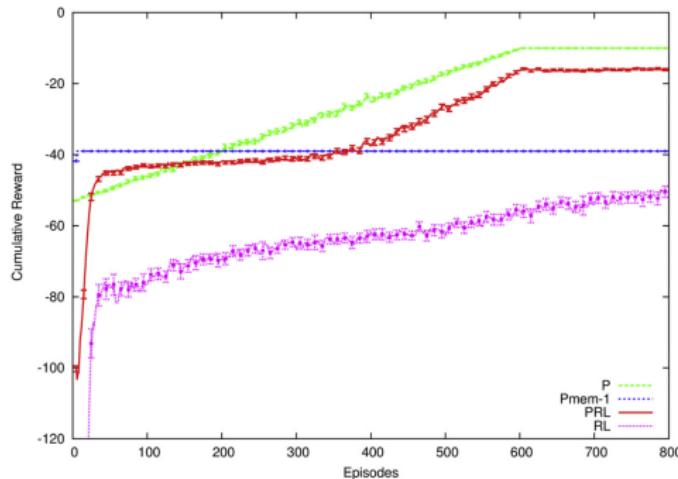


图 6: 实验结果

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。

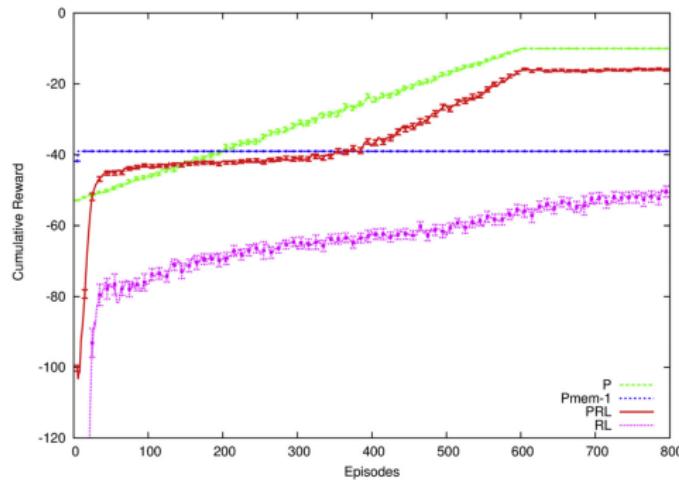


图 6: 实验结果

PRL agent 学到了最优策略，并在环境变化后选择了更优的策略。

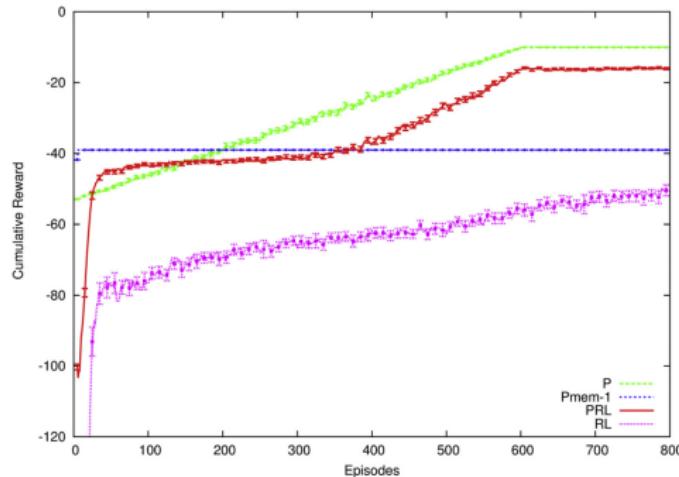


图 6: 实验结果

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。

背景知识回顾

两种规划与学习结合的方法

事后经验回放与稀疏奖励下的规划



第 II 部分

强化学习在游戏中的应用

Atari 游戏与 DQN



第 III 部分

FPS 游戏与 DRQN

DRQN

用 DRQN 玩 FPS 游戏

- ▶ 前面提到，DQN 在 Atari 游戏中，有能力学习到不输人类的控制策略。
- ▶ 它仅仅将游戏最近 4 帧的原始画面作为输入，就使 agent 在 Atari 游戏中有了能与人类相抗衡的表现。
- ▶ 但是，DQN 在需要记忆**超过最近 4 帧**画面的游戏中表现不佳，这类游戏未来的 state 和 reward 不仅依赖于当前输入的画面，还依赖于更早画面中的信息。
- ▶ 上述现象是一种 Partially-Observable Markov Decision Process (POMDP)。

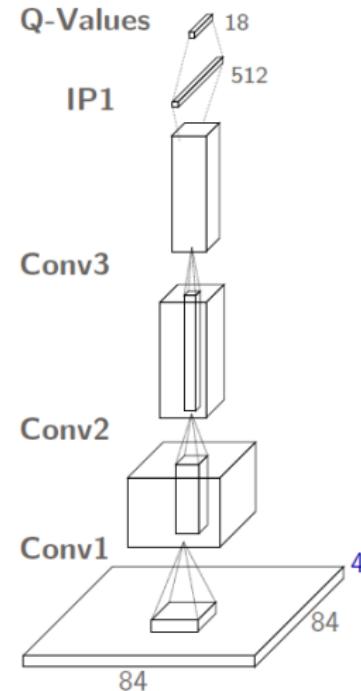
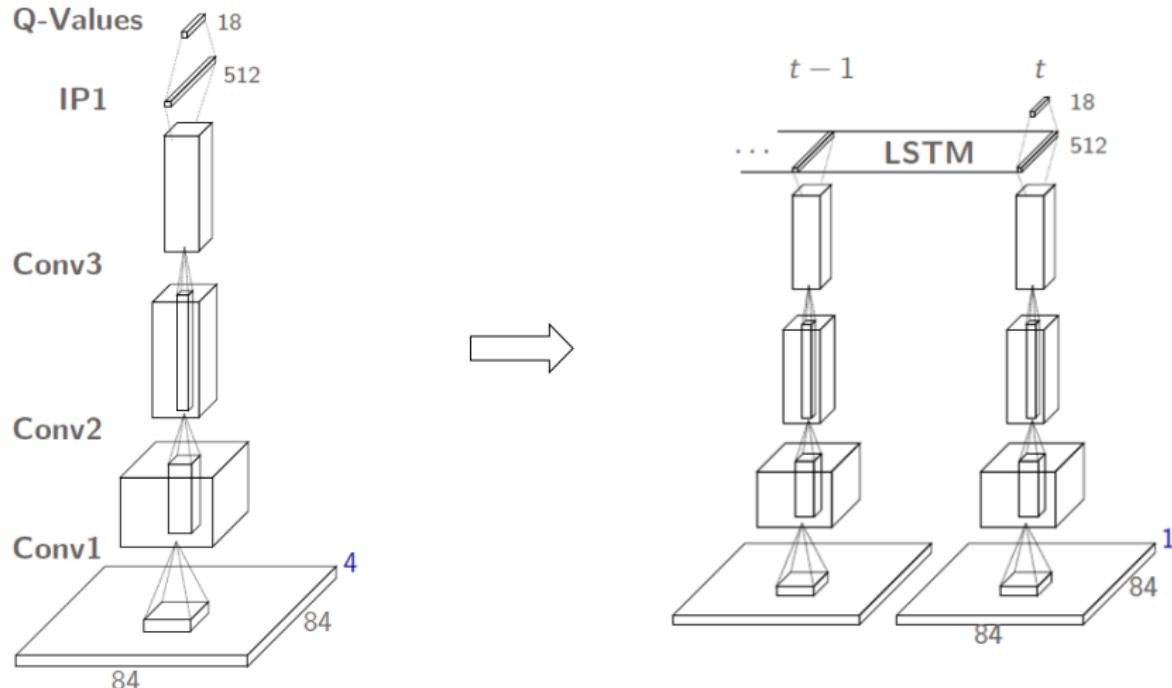


图 7:



为解决前述 POMDP 问题，我们需要对原始的 DQN 加以改进，Deep Recurrent Q-Network 应运而生。



与 DQN 相比，DRQN：

- ▶ 将原本的全连接层替换为相同维度的 Long-Short Term Memory (LSTM)。
- ▶ 每一个 timestep 只接受一帧画面作为输入。
- ▶ LSTM 为过去的状态提供记忆能力。

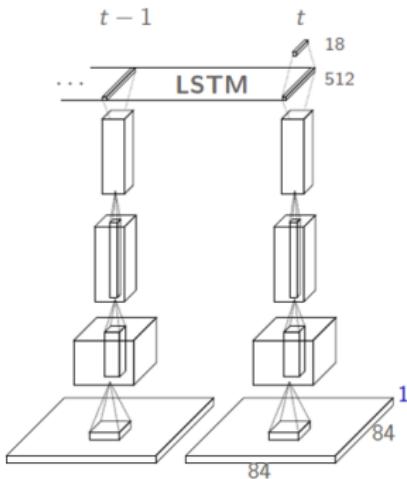


图 8：

- ▶ Intelligent decision making is the heart of AI.
- ▶ Desire agents capable of learning to act intelligently in diverse environments.
- ▶ Reinforcement learning provides a general learning framework.
- ▶ RL + deep neural networks yields robust controllers that learn from pixels. (DQN)
- ▶ DQN lacks mechanisms for handling partial observability.
- ▶ Extend DQN to handle Partially Observable Markov Decision Processes (POMDPs).

- ▶ 比较对象：
 - ▶ DRQN;
 - ▶ 扩展 DQN (将普通 DQN 的输入扩展到 10 张图片);
 - ▶ DQN.
- ▶ 实验目标：
 - ▶ Standard Atari games
 - ▶ Flickering Atari games

Game	DRQN $\pm std$	Ours	DQN $\pm std$
	Mnih et al.		
Asteroids	1020 (± 312)	1070 (± 345)	1629 (± 542)
Beam Rider	3269 (± 1167)	6923 (± 1027)	6846 (± 1619)
Bowling	62 (± 5.9)	72 (± 11)	42 (± 88)
Centipede	3534 (± 1601)	3653 (± 1903)	8309 (± 5237)
Chopper Cmd	2070 (± 875)	1460 (± 976)	6687 (± 2916)
Double Dunk	-2 (± 7.8)	-10 (± 3.5)	-18.1 (± 2.6)
Frostbite	2875 (± 535)	519 (± 363)	328.3 (± 250.5)
Ice Hockey	-4.4 (± 1.6)	-3.5 (± 3.5)	-1.6 (± 2.5)
Ms. Pacman	2048 (± 653)	2363 (± 735)	2311 (± 525)

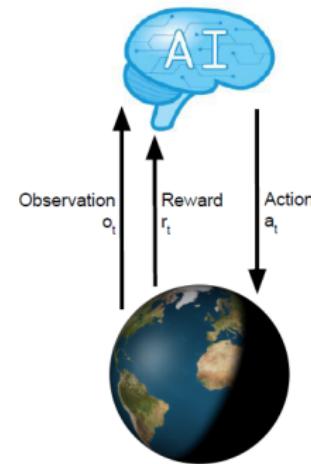
图 9: DRQN 的表现勉强达到 DQN 的水平



以一个随机概率遮蔽屏幕，使游戏呈现
partially observability.

$$o_t \begin{cases} s_t & \text{with } p = \frac{1}{2} \\ \langle 0, \dots, 0 \rangle & \text{otherwise} \end{cases}$$

现在，当前的游戏状态只能够从历史状态
中获取到了。



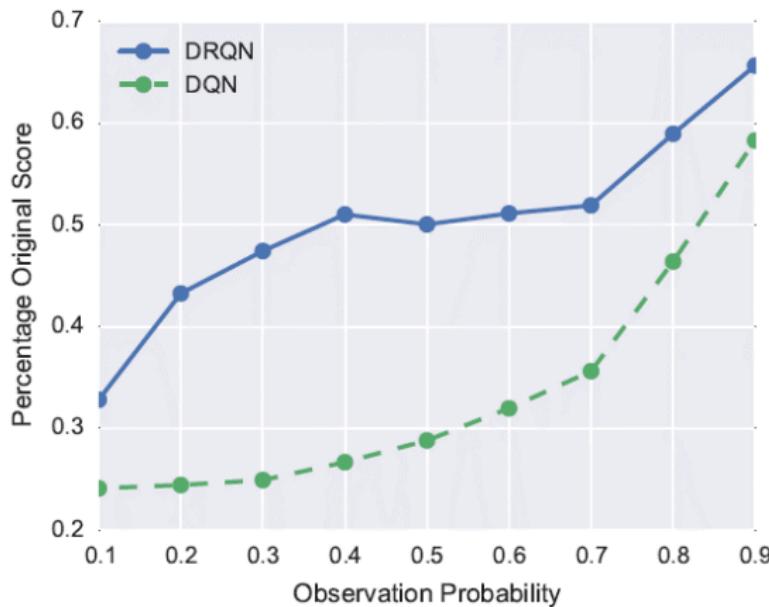


图 10: 横轴: 某时间步能够观测到游戏画面的概率; 纵轴: 各模型与自己 Standard Atari games 得分的比值

- ▶ 在 Standard Atari games 中，DRQN 的表现勉强比上 DQN
- ▶ 在 Flickering Atari game 中，DRQN 表现下降的幅度比 DQN 小得多

- ▶ 在 Standard Atari games 中，DRQN 的表现勉强比上 DQN
- ▶ 在 Flickering Atari game 中，DRQN 表现下降的幅度比 DQN 小得多

但这样的结果远远不能让人满意……

DRQN

用 DRQN 玩 FPS 游戏

FPS 即第一人称射击游戏 (First person shooting games)，与 Atari 2600 相比，FPS 游戏具有以下特点：

- ▶ 操作更加复杂，行为更加丰富 (搜索地图，收集道具，识别并打败敌人…);
- ▶ 画面只能观测到部分游戏状态 (partially observable);
- ▶ FPS 与现实场景相似，相关技术适合在机器人开发中得到应用。

VizDoom 是一个基于 Doom 的 AI 研究平台，主要针对面向原始视觉信息输入的增强学习。

Doom Deathmatch 规则介绍：<http://vizdoom.cs.put.edu.pl/>

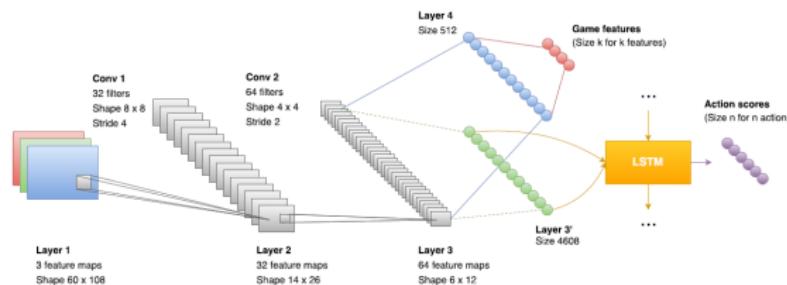


首先，直接利用原始的 DRQN 运用在 ViZDoom 上，效果并不好。

- ▶ 训练出的 agent 会随意开火；
- ▶ 对使用弹药加以惩罚：如果惩罚过重，agent 就不会开火；过轻，agent 依然会随意开火。
- ▶ agent 不能够准确地探测敌人

DRQN augmented with game features 使用了一个增加游戏信息 (game features) 的 DRQN 模型来提升性能。

Divide and conquer 将游戏过程分为两个阶段分别训练，提升性能并加快了训练速度。



- ▶ 训练时加入了当前画面的游戏状态信息作为输入；
- ▶ 网络的损失函数是 DRQN 的损失函数合并上交叉熵损失 (cross-entropy loss)，同时训练 DRQN 和游戏状态信息的侦测，使卷积层也能够捕捉到与游戏相关的信息；
- ▶ 虽然可以获得许多游戏信息，但仅考虑画面中是否有敌人的信息，就能使模型的表现有显著的改进。

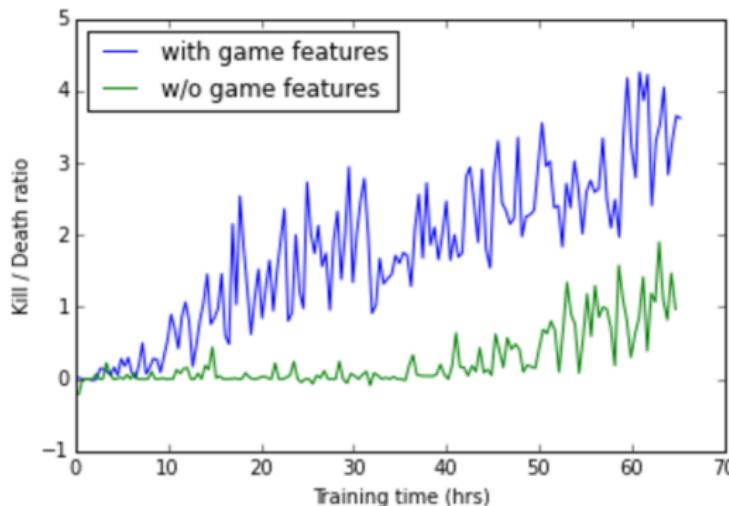


图 11: 横轴: 训练时间; 纵轴: 击杀死亡比

Doom Deathmatch 游戏过程被分成两个阶段：导航阶段和行动阶段

导航阶段 (the navigation) 探索地图，收集，发现敌人

行动阶段 (the action) : 攻击敌人

Doom Deathmatch 游戏过程被分成两个阶段：导航阶段和行动阶段

导航阶段 (the navigation) 探索地图，收集，发现敌人

行动阶段 (the action) : 攻击敌人

当前游戏过程处何阶段由画面中是否有敌人来决定 (game feature)。



图 12: 导航阶段



图 13: 行动阶段

对不同的阶段，使用了不同的模型进行训练：

导航阶段 DQN

行动阶段 DRQN with game features

对不同的阶段，使用了不同的模型进行训练：

导航阶段 DQN

行动阶段 DRQN with game features

目前，DQN 并不支持结合多个网络来优化不同的目标，但当前的游戏信息可以由改进的 DRQN 进行侦测。

这样的分治策略带来许多优点：

这样的分治策略带来许多优点：

- ▶ 两个网络可以并行训练，大大加快了模型训练速度。

这样的分治策略带来许多优点：

- ▶ 两个网络可以并行训练，大大加快了模型训练速度。
- ▶ 导航阶段只包含了三种操作：向左、向右和前进，大大减少了模型中 state-action pairs 的数量，进一步提高效率。

这样的分治策略带来许多优点：

- ▶ 两个网络可以并行训练，大大加快了模型训练速度。
- ▶ 导航阶段只包含了三种操作：向左、向右和前进，大大减少了模型中 state-action pairs 的数量，进一步提高效率。
- ▶ 相比使用单个网络训练，分治有效避免了 agent 的“蹲点”行为。

在 Doom deathmatch 中，游戏分数通过击杀数/死亡数 (K/D ratio) 进行计算。如果回馈只通过计算分数得到，那么 DRQN 训练时的 replay table 会非常稀疏，导致模型在训练时难以收敛。
作者们采取了回馈共享 (Reward shaping) 的思路，修改回报函数，包含一些小的中间回报来加速学习过程。

基于击杀给予正回馈，死亡基于负回馈的基础上，增加了以下中间回馈

行动网络

- ▶ 拾取物品，正回馈
- ▶ 失去生命值，负回馈
- ▶ 开枪射击导致弹药减少，负回馈

导航网络

- ▶ 拾取物品，正回馈
- ▶ 在岩浆上行走，负回馈
- ▶ 移动距离，正回馈——有利于让 agent 更快速地探索地图



赛制分两种：

- ▶ 已知地图上的受限制死亡竞赛 (Limited Deathmatch): 武器只有火箭炮, agent 可以捡血包和弹药;
- ▶ 未知地图上的的无限制死亡竞赛 (Limited Deathmatch): agent 初始只有手枪, 可以捡各种武器弹药和血包。提供了两张地图用于训练, 3 张未知地图用于测试。



Evaluation Metric	Limited Deathmatch		Full Deathmatch			
	Known Map		Train maps		Test maps	
	Without navigation	With navigation	Without navigation	With navigation	Without navigation	With navigation
Number of objects	14	46	52.9	92.2	62.3	94.7
Number of kills	167	138	43.0	66.8	32.0	43.0
Number of deaths	36	25	15.2	14.6	10.0	6.0
Number of suicides	15	10	1.7	3.1	0.3	1.3
Kill to Death Ratio	4.64	5.52	2.83	4.58	3.12	6.94



Place	Team	Bot	1	2	3	4	5	6	7	8	9	10	11	12	Total frags
1	F1	F1	56	62	n/a	54	47	43	47	55	50	48	50	47	559
2	The Terminators	Arnold	36	34	42	36	36	45	36	39	n/a	33	36	40	413
3	CLYDE	CLYDE	37	n/a	38	32	37	30	46	42	33	24	44	30	393

Place	Team	Bot	1	2	3	4	5	6	7	8	9	10	11	12	Total frags
1	IntelAct	IntelAct	29	21	23	21	6	11	9	6	30	32	33	35	256
2	The Terminators	Arnold	22	17	21	15	13	12	6	5	14	13	13	13	164
3	TUHO	TUHO	8	11	13	12	0	-1	-1	-4	2	2	6	3	51



Single player 人类和 agent 分别在独立的游戏中与机器人较量

Multiplayer 人类与 agent 相互切磋

Evaluation Metric	Single Player		Multiplayer	
	Human	Agent	Human	Agent
Number of objects	5.2	9.2	6.1	10.5
Number of kills	12.6	27.6	5.5	8.0
Number of deaths	8.3	5.0	11.2	6.0
Number of suicides	3.6	2.0	3.2	0.5
K/D Ratio	1.52	5.12	0.49	1.33

图 14:

Track 1: Limited Deatchmatch

Team	Bot	1	2	3	4	5	6	7	8	9	10	Total:
Marvin	Marvin	23	23	22	23	28	29	26	21	32	21	248
Terminators	Arnold2	24	19	23	33	24	22	19	33	22	26	245
Axon	Axon	29	22	13	22	28	21	24	22	16	18	215

Track 2: Full Deatchmatch

Team	Bot	1	2	3	4	5	6	7	8	9	10	Total:
Terminators	Arnold4	15	20	25	24	18	32	42	42	32	25	275
TSAIL	YanShi	24	31	31	35	34	21	28	28	21	20	273
IntelAct	IntelAct	17	26	21	24	24	22	27	24	15	21	221

[Leonetti et al., 2016] Leonetti, M., locchi, 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