高级人工智能作业 5

强化学习算法

姓名：董广念 学号：11849058

本报告主要使用 Q-learning 与 Sarsa 算法玩 aliens 游戏。使用 assignment 5 中提供的 GVGAI_GYM 和 sakai 提供的文件作为实验环境。实验目标为：

（1） 控制 Agent 通关游戏

（2） 获得尽量多的分数

本次作业提交模型部分使用 Sarsa-lambda 算法训练。作业实现了 Q-learning 算法和 Sarsa 算法；与曾歆勋同学合作设计 reward 模型与缩小 state 模型。本文将对作业过程中使用过的训练方式和产生的数据进行分析给出报告。具体方法有。

表 4 作业 5 使用过的训练方法

序号	**state observation**	**reward**	**训练方式**	**算法**
1	full observation	经验设定	随机	Q-learning
2	3×3 part observation	经验设定	随机	Q-learning
3			引导	
4	9×3 part observation	经验设定	随机	Q-learning
5	key elements observation	设计 reward 模型	随机	Sarsa-lambda

5 为本次作业程序部分使用版本。

报告分为（1）问题描述（2）算法设计（3）实验设计（模型训练）（4）结论（5）参考文献，五部分组成。

1. 问题描述

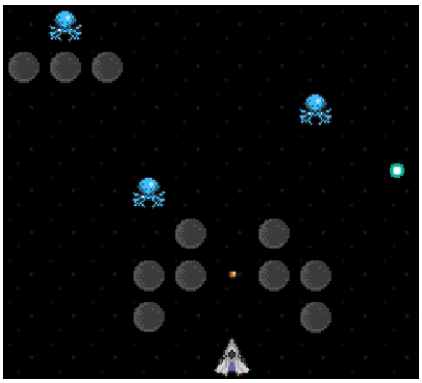


图 1 游戏界面

游戏界面如图所示，总共有 6 种物品，飞机，子弹，外星人，炸弹，石头，背景。原始环境反馈：子弹击中石头得 1 分，击中外星人得 2 分。消灭所有外星人游戏胜利、被外星人碰到或被炸弹碰到游戏失败。

目标为设计一款增强学习 Agent，输入为游戏界面 observation，每个 observation 的 reward, info 和 game ticks，输出 action 使得整场游戏拿到最多 reward。

2. 算法设计

本部分主要描述算法设计，给出（1）算法设计及分析、（2）算法流程、（3）伪代码。具体参数请见 **3.参数训练**

2.1 算法结构及分析

算法主要参考自[1]，分为 4 个部分：

1) 模型读取部分
2) 环境状态转换部分
3) 输出 action 部分
4) 学习部分

在以上算法结构中，能够影响算法表现的是（2）（4）两部分。

主要有如下问题：

问题一：如何寻找一个辨识度较高的空间划分方式？

环境状态转换决定了 Q-table 的大小，即训练空间、训练难度的大小。full observation 最多有$6^{90} \times 4$种可能性（每个网格有 6 种可能的角色，共 90 个网格，4 种 action），每个 state 的每个 action 需要足够的训练才能收敛，因此到达收敛的训练时间很长。

在训练过程中发现转换后的 state observation 只是作为状态索引存在 Q-table 中，并没有其他的用途，所以状态转换不一定围绕着以飞机为中心的范围截取状态。而是能够尽量在 stateObs 中提取状态代表能力强，action 效用区分能力强的元素作为建立 Q-table 的 state。

环境状态转换的目标有两个：提取有效元素使（1）Q-table 尽量小以缩短训练时间；（2）使状态所对应的 action 优劣区分尽量明显。

问题二：如何学习？（reward 分布问题）

学习有三种方式：（1）通过经验给 agent 限定规则，通过总是用期望的 action 序列训练的方式让 agent 执行期望 action（2）总是使用差的或导致失败的 action 序列训练 agent，使坏 action 价值降低的方式让 agent 执行好的 action。（3）随机训练。（在 **3.实验设计**中对比了限定规则与随机训练的得分情况。）

reward 反馈方式：reward 应对每种物品的得分难度进行评估进行对应的得分操作。且目标为得分时，胜利失败对应的奖励或惩罚应按照当前得分与均分的差距给出。

reward 反馈错误：解决该问题有两种方式：（1）lambda 方式：记录 agent 从开始到获取数据的所有(state, action)序列，每次获得 reward 给 Q-table 中所有前驱序列按照时间长短加分（3）人工设计符合 reward 反馈规律的状态转换方式，比如以得分动作开始到结束一段时间作为一个 state。

2.2 算法流程

agent 训练算法流程描述如下：

(1) 在文件中读取模型参数，载入已经训练好的 Q 表。
(2) 启动游戏，运行 10000 次
(3) 获取当前环境 ticks, observation((90, 100, 4) ndarray), 并转换为 state, 9×10 的数组。即将 observation 的前 2 维每 10 行 10 列为一个单元求和，state 中每个不同的数字代表游戏界面中不同的角色。
(4) 将 ticks 和 state 飞机位置，炸弹位置，子弹位置组成数字串作为该状态的索引 stateIdx。
(5) 传入 stateIdx 执行ε-greedy 算法产生 action。
(6) 将 action 传入游戏环境，获取 ticks, newObservation, reward, info（游戏反馈）
(7) 转换 reward，见**算法 4.1**
(8) ticks, observation, action, newObservation, reward, info 传入学习函数，使用 Sarsa-lambda 算法更新 Q 表。
(9) observation = newObservation，到步骤(3)

agent 使用 Sarsa-lambda 算法训练。Q-table value function 为：

$$\mathrm{Q(S,A)} = \mathrm{Q(S,A)} + \alpha[\mathrm{R} + \gamma \mathrm{Q}(S', A') - \mathrm{Q(S,A)}]\mathrm{E(S,A)} \quad (1)$$

$$\mathrm{E(S,A)} = \gamma\lambda E(S, A) \quad (2)$$

公式(1)为 Sarsa 算法 value function，E 为一个形如 Q 表的表格，存储自游戏开始至游戏结束所有经过的路径。

2.3 伪代码

(1) 主程序

主程序完成输入参数，获取模型，多次运行游戏训练模型（多次用模型玩游戏）的过程。

算法 1	**runGame**
输入	**ALPHA, GAMMA, EPSILON, LAMBDA, iterNum**
输出	Trained model
1	**Agent** ← Read model from file “aliensRLModel.json”
2	**Loop iterNum** times:
3	Game reset
4	**Loop** 2000 ticks:
5	give **observation** to the **Agent** and get **nextAction**
6	**return action** to the game
7	get **newObservation, reward, info** from the game
8	**Agent.learn(ticks, observation, action, newObservation, reward, info)**
9	**Observation ← newObservation**
10	**end loop**
11	**end loop**

（2） 状态获取部分

进行状态空间的转换，将游戏返回的 observation 转换关键元素序列（飞机，炸弹，子弹，时间）

算法 2	**transObs**
输入	**ticks, stateObs** //从游戏中获取的状态
输出	**stateIdx**//状态索引
1	**fullState**←add every (10,10,4) sub-ndarray in original (90,100,4) ndarray
2	**stateIdx = “”**
3	**if** no **AIRCRAFT** in **fullState**
4	**return** “TERMINAL”
5	**stateIdx +=** pos(**AIRCRAFT**)
6	**if BOMB** in **fullState**:
7	**stateIdx +=** pos(**BOMB**)
8	**if SAM** in **fullState**:
9	**stateIdx +=** pos(**SAM**)
10	**return stateIdx**

（3） 根据状态和参数选择部分

给出了随机 action，ε-greedy 算法两种选项

算法 3	**chooseAction**
输入	**ticks, stateObs, randomAct** //gema ticks；环境状态；布尔型，否返回随机 action；[0, 1]的小数
输出	**action**
1	**if randomAct** == True: return random action
2	**else**:
3	**shrinkedStateIdx**, **shrinkedState**← **transObs(stateObs)**
4	**if** randomNumber<EPSILON
5	**return** random action
6	**else**:
7	**return** action with maximum value in **Q-table[shrinkedStateIdx]**

（4） 学习部分

学习部分使用 Sarsa-lambda 算法，增加了 reward 立即返回给前驱(S, A)的 E-table。

算法 4	**learn**
输入	**ticks, stateObs0, actionID, stateObs1, reward, info** //game ticks；action 执行前状态；action；action 执行后状态；reward；环境信息
1	**stateIdx0, shrinkedState0**← **transObs(ticks, stateObs0)**
2	**stateIdx1, shrinkedState1**← **transObs(ticks+1, stateObs1)**

3	**next_action = chooseAction(ticks+1, stateObs1)**
4	**if stateIdx0** not in **Q-table**:
5	insert **stateIdx0** with all 0 elements in **Q-table** and **E-table**
6	**if stateIdx1** not in **Q-table**:
7	insert **stateIdx1** with all 0 elements in **Q-table** and **E-table**
8	**reward←transformReward(reward)**
9	**INCREASE** = **reward**+**GAMMA***Q-table[stateIdx1,nextAction]- Q-table[stateIdx0, actionID]
10	**for** all **(stateIdx, action)**:
11	Q[stateIdx, action]← Q[stateIdx, action]+ ALPHA*INCREASE*E[stateIdx, action]
12	E-table[stateIdx, action] = GAMMA*LAMBDA*E[stateIdx, action]

reward 转换函数，本模型以得分和胜利为导向。

本文认为一个合理的 reward 一定与 agent 表现有关，即 reward 通过要求 agent 每轮训练超过自己以往训练均值的方式来逐步提高 agent 的表现。胜利失败得分依据为当前总分距离平均总分的距离，距离越远，奖励/惩罚就越高，呈指数型增长。胜利得分比当前平均分越高，获得奖励越多。失败得分比当前平均分越低，获得惩罚越多；当得分与当前均分相近时给出微弱的奖励或惩罚。

中间过程的得分以得分难度为导向。打死外星人比打掉石头难，故设成 3 次方。

算法 5	**transformReward**
输入	**increScore, totalScore, info** //当前反馈得分，总反馈得分
1	if PLAYER_WINS:
2	**if totalScore < averageScore in game:**
3	**reward = totalScore**
4	**else:**
5	**reward** = **totalScore**+exp(|**totalScore**-**averageScore**|)
6	**else if PLAYER_LOSE:**
7	**if totalScore < averageScore in game:**
8	**reward = increScore-** exp(|**totalScore**-**averageScore**|)
9	**else:**
10	**reward** = **increScore**
11	**else**: **reward** = **increScore**^3
12	**return reward**

3. 实验设计（模型训练）

本部分首先给出实验所用计算资源，再给出 agent 参数以及实验结果，最后对作业过程

中使用过的方法进行比对。

3.1 计算资源

所用计算资源如表 2 所示

表 2 实验计算资源	
操作系统	计算能力
Windows10 PC	Intel i7-8700 3.2GHz CPU; 24GB RAM

单线程每训练 1000 轮用时 1 小时左右。

3.2 Agent 模型参数

算法参数如下:

表 3 算法参数设置			
序号	**参数名**	**意义**	**数值**
1	ALPHA	Q-learning 学习率	0.9
2	GAMMA	Q-table 折扣系数	0.9
3	LAMBDA	E-table 折扣系数	0.9
4	EPSILON	ε-greedy，探索概率	0.01
5	shrinkedObs	状态空间大小	3×3 方格
6	iterNum	迭代停止轮次	1000

模型为以上参数训练 460 轮得到。因为训练时间过长，本次实验未进行网格搜索调参，而是针对同一组参数使用不同的方式训练模型做出对比。以上参数为与同学讨论得来。

图 2 为 agent 在训练 1000 次过程中的得分变化图（40 轮游戏取一次均值）。

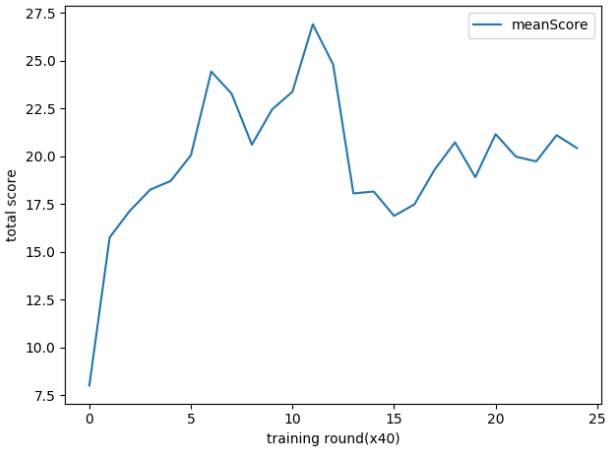


图 2 方法 5 得分随训练变化图

在 440-480 轮时，平均得分达峰值 27 分。

在本提交模型下运行游戏 100 次平均得分：22.53 score/183.74 ticks

3.3 对比实验

本部分分析实验过程中使用过方法的实验效果。以下实验因为操作原因各训练过不同的轮次，以下未对迭代轮次做出说明的表格数据默认为训练 1000 轮，取 1001-1200 轮实验数据的平均数。

表 4 作业 5 使用过的训练方法

序号	**state observation**	**reward**	**训练方式**	**算法**
1	full observation	经验设定	随机	Q-learning
2	3×3 part observation	经验设定	随机	Q-learning
3			引导	
4	9×3 part observation	经验设定	随机	Q-learning
5	key elements observation	设计 reward 模型	随机	Sarsa-lambda

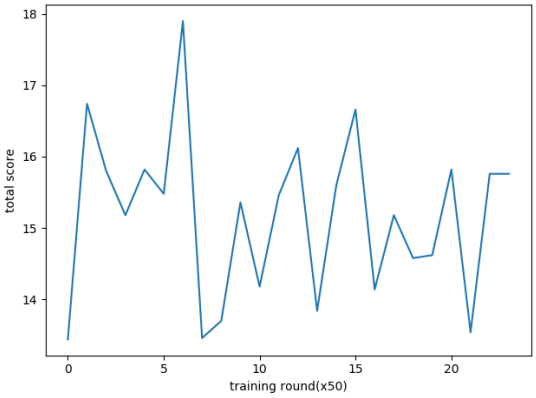
其中，引导式训练为在模型中添加经验性规则，用确定性得分 action 训练模型（经验性规则产生的概率为 0.45）。打中外星人奖励 20，打中石头奖励 5，游戏胜利奖励 1000，游戏失败奖励-100。其他的参数均与表 3 相同。

表 5 6 种方法实验数据

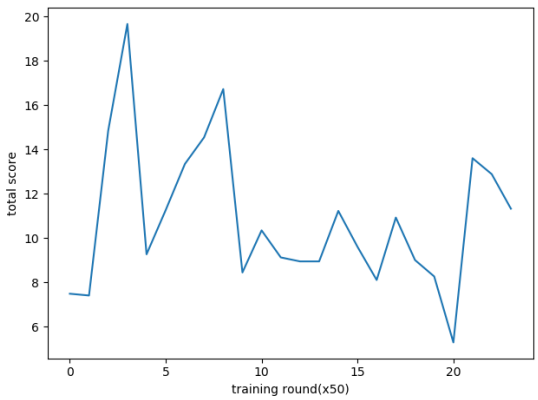
序号	平均 reward	平均 ticks	模型文件大小
1	15.22	141	73MB
2	10.77	191	21KB
3	37.17	352	81KB
4	23.28	278	5.72MB
5	18.075	NaN	2.01MB

对于表 4，表 5 所示数据来说，各种不同方式在当前对比条件下的表现排名为 3、4、5、1、2。

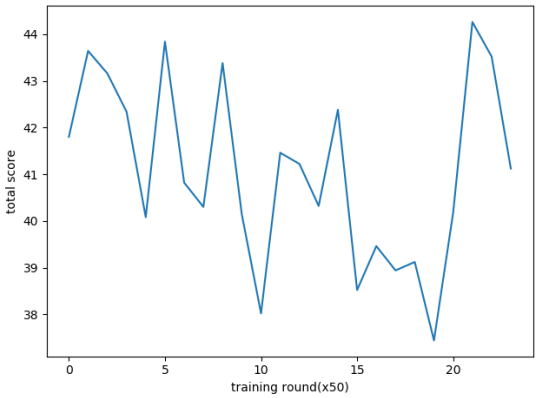
表 5 说明 full observation 执行效果比随机值好，这符合预期。图 3(a)，说明 full observation 无法在 1000 次训练中收敛，表现出较大的波动性。

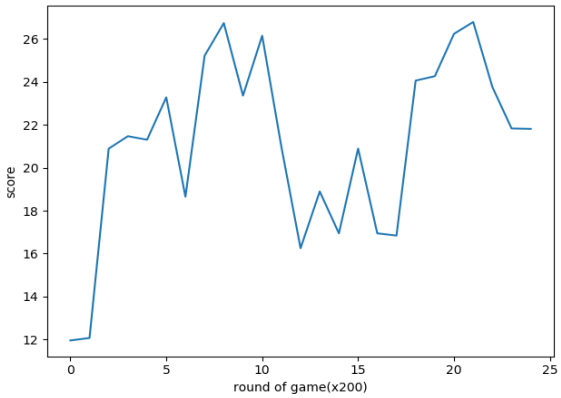


(a)方法 1



(b) 方法 2



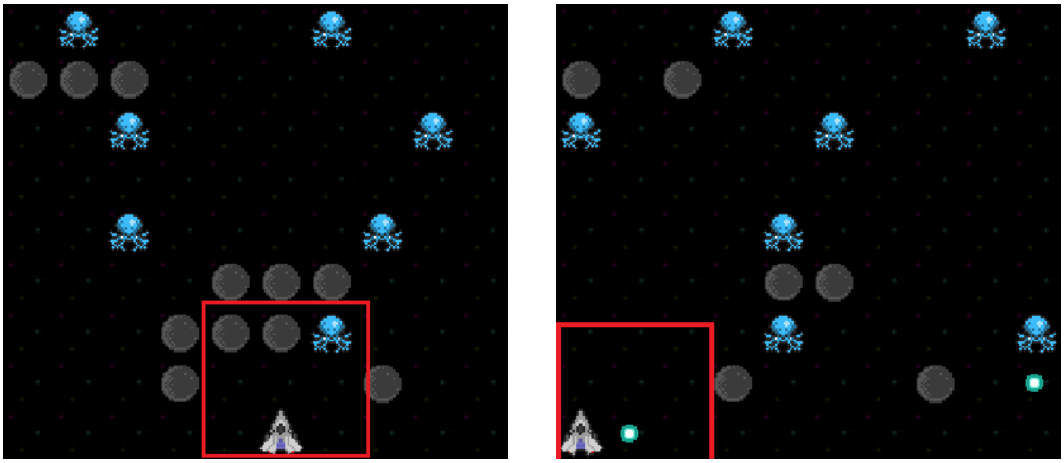


(c) 方法 3　　(d) 方法 4

图 3 方法 1，2，3，4 得分随训练变化图（每 50 次求平均）

图 3(b)表明 3×3 网格作为 state 同样难以收敛。3×3 网格作为 state 在训练 500 轮左右表现出连续游戏胜利的特点。但是继续训练 6 小时以后得分便下降到较低的水平。原因可能是因为 3×3 网格本身无法区分很多 full observation 的反馈。如打到视野范围外的石头或外星人所产生的 reward。表现在训练后期便是有外星人在 agent 上方，agent 总是左右乱晃。而不发射。

3×3 网格的好处就是便于编写引导训练程序。如图 4 所示(a)中射击一定能打到外星人；(b)中也可以轻易躲开炸弹。图 3(c)实验数据也证实了引导训练的效果。通过引导让 agent 的效果变好，这也符合我们的常识。但因为该方式有 panelty，故不作为最终结果上交。



(a)　　(b)

图 4 3×3 state 实例

图 3(d)说明，以上排名不能表明方法的优劣。不同的 state 空间表示与 reward 表示方法会在不同的训练轮次下达到局部峰值。如图 3(d)所示，方法 4 的局部最优训练轮次在 1500 轮和 4000 轮左右。引导式训练效果最显著。

4. 结论

full observation 效果好，但是搜索空间巨大，文件大小远远超出其他的训练方式。对于更大一些的场景很难应用。

part observation 的效果应该与如何建立 part observation 有关系，在本次实验中 part observation 的效果较差，原因可能是视野之外的 reward 影响了各个 action 的得分。但是加以引导也可以达到比较好的效果。

作业提交版本的模型不太很符合我的预期，原因可能是训练轮次过短导致，将尝试更长时间的训练。

除去已经分析的因素外其他还可以做的工作:

1. **定制训练：**在本例中，整个界面不确定性只来源于外星人的随机炸弹，我们可以首先用删除随机炸弹的版本训练出一套得分技能，然后再加入不确定性的炸弹学习躲避技能。
2. **状态转换：** 存储子弹位置可以换成存储是否有子弹，因为决定子弹射出得分的是石头，ticks 和发射位置。与子弹在哪里飞无关；存储炸弹位置可以改成当炸弹在周边时再存储炸弹位置，因为炸弹在远处或不在面前时不会导致 agent 死亡；存储石头位置，通过观察很多可以在无遮挡下打中外星人的子弹被石头挡了下来。
3. **无随机性训练：**即训练时将 EPSILON 调为 0，使用 Sarsa-lambda 算法。猜测这样可以使得 agent 形成一套游戏胜利的范式，但是不会去探索新的空间提高分数。

5. 参考文献

[1] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction, 2012.

[2] 强化学习 Reinforcement Learning 教程系列 | 莫烦 Pyhton.(n.d.). Retrieved December 30, 2018, from: *https://morvanzhou.github.io/tutorials/machine-learning/reinforcement-learning/*