高级人工智能作业 5

强化学习算法

姓名：董广念 学号：11849058

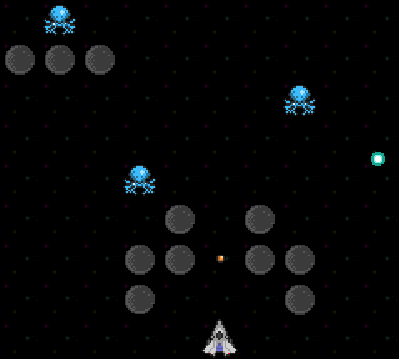
本报告主要使用Q-learning算法玩aliens游戏。使用assignment 5中提供的GVGAI\_GYM和sakai提供的文件作为实验环境。实验目标为：

1. 控制Agent通关游戏
2. 获得尽量多的分数

本次作业使用基本Q-learning算法完成。对比了训练方式和状态转换对算法的影响。具体对比了（1）诱导训练和随机训练（2）不同的状态转换方法（3）奖励型训练和惩罚型训练。

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。

# 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。子弹击中石头得1分，击中外星人得2分。消灭所有外星人、被外星人碰到或被炸弹碰到游戏结束。消灭所有外星人游戏胜利，后两者游戏失败。

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和info，输出action使得整场游戏拿到最多reward。

# 算法设计

本部分主要描述算法设计，给出（1）算法结构及分析、（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。

引入了如下问题：

**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大小，即训练空间、训练难度的大小。full observation最多有种可能性（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。

**环境状态转换的目标有两个：**（1）使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）通过人类观察给agent限定规则，通过总是用好的action序列训练使好action的价值增大的方式让agent执行好的action（2）总是使用差的或失败的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。

**reward反馈错误：**解决该问题有三种方式：（1）增大λ，使正确动作接收到迟来的reward；（2）物理意义上缩小观察空间，即只存储飞机附近的observation，使得reward更大比例的反馈到ACTION\_USE上；（3）人工设计符合reward反馈规律的状态转换方式。

**胜败反馈有两种方式：**（1）奖励型反馈，即游戏胜利时给出大额奖励，失败时小额惩罚（2）惩罚型反馈：失败大额惩罚，胜利小额奖励（3）混合型反馈：如果失败，比较当前得分与期望得分或历史平均得分。依据距离给出定额的奖励或惩罚。

|  |  |
| --- | --- |
| C:\Users\97439\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Snipaste_2018-12-27_17-34-12.png  (a) | C:\Users\97439\Desktop\Snipaste_2018-12-27_17-35-05.png  (b) |

**图2** 缩小版state对比

本次作业采用以飞机为底边中心的3×3网格，采用启发式训练方式，即在训练的时候按概率返回由规则产生的一定能带来reward的action训练模型。采用奖励型反馈模型，即WIN给出大额奖励，LOSE给出偏低额度惩罚。

## 2.2 算法流程

算法流程描述如下：

1. 在文件中读取模型参数，已经训练好的Q表。
2. 启动游戏，运行10000次
3. 获取当前环境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. 获取state中飞机位置，将其左右两列以及所在列，上面两行以及所在行(3×3数组)记录在shrinkedState中。如果左边没有列，则只包含所在列及右边列(3×2数组)，如果右边没有列，则只包含所在列及左边列(3×2数组)，将shrinkedState转化为字符串作为该状态的索引。
5. 以rcmd为概率执行推荐action。（1-rcmd）为概率执行ε-greedy算法产生action。
6. 将action传入游戏环境，获取newObservation，reward，info（游戏反馈）
7. 转换reward，各部分reward转换值见表2（算法参数设置）
8. observation, action, newObservation, reward, info传入学习函数，使用Qlearning算法更新Q表。
9. observation = newObservation，到步骤(3)

## 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, iterNum** |
| **输出** | Trained model |
| 1 | **Agent** 🡨 Read model from file “aliensRLModel.json” |
| 2 | **Loop** **iterNum** times: |
| 3 | Game reset |
| 4 | **Loop** 1000 ticks: |
| 5 | give **observation** to the **Agent** and get an **action** |
| 6 | **return action** to the game |
| 7 | get **newObservation, reward, info** from the game |
| 8 | **Agent.learn(observation, action, newObservation, reward, info)** |
| 9 | **Observation 🡨 newObservation** |
| 10 | **end loop** |
| 11 | **end loop** |

1. 状态获取部分

进行状态空间的转换和压缩，将游戏返回的observation转换成自己想要的observation。

|  |  |
| --- | --- |
| **算法 2** | **transObs** |
| **输入** | **stateObs** //从游戏中获取的状态 |
| **输出** | **shrinkedStateIdx, shrinkState** |
| 1 | **state**🡨add every (10,10,4) sub-ndarray in original (90,100,4) ndarray |
| 2 | **if** no **AIRCRAFT** in **state** |
| 3 | return “TERMINAL”, None |
| 4 | **else:if AIRCTAFT** is at side of **state**://如果飞机在边上，只截取周边(3,2)大小背景 |
| 5 | **shrinkedState**🡨(3,2) size array in **state** around the **AIRCTAFT** |
| 6 | **return** str(**shrinkedState**), **shrinkedState** //str(**shrinkedState**)为状态索引 |
| 7 | **else**: //如果飞机不边上，截取周边(3,3)大小背景 |
| 8 | **shrinkedState**🡨(3,3) size array in **state** around the **AIRCTAFT** |
| 9 | **return** **shrinkedState** with (3,3) size around the **AIRCTAFT** |

1. 根据状态和参数选择部分

给出了随机action，推荐action，ε-greedy算法三种选项

|  |  |
| --- | --- |
| **算法 3** | **chooseAction** |
| **输入** | **stateObs, randomAct, rcmdProb**  //环境状态；布尔型，否返回随机action；[0, 1]的小数，返回建议action概率 |
| **输出** | **action** |
| 1 | **if** **randomAct** == True: return random action |
| 2 | **if** randomNumber< **rcmdProb: return** **recommededAction(stateObs)** |
| 3 | **else**: **return** **epsilon-greedy(stateObs)** |

|  |  |
| --- | --- |
| **算法3.1** | **recommededAction** |
| **输入** | **stateObs** |
| **输出** | **action** |
| 1 | **shrinkedStateIdx**, **shrinkedState**🡨 **transObs(stateObs)** |
| 2 | **if** **shrinkedState** has bomb: |
| 3 | **return** action can escape |
| 4 | **if** **shrinkedState** has aliens: |
| 5 | **if** we can hit it: |
| 6 | **return** action fire on it |
| 7 | **if** we can not hit it: |
| 8 | **return** action chase it |
| 9 | **if** **shrinkedState** has stone: |
| 10 | **return** action fire on it |

|  |  |
| --- | --- |
| **算法3.2** | **epsilon-greedy** |
| **输入** | **stateObs** |
| **输出** | **action** |
| 1 | **shrinkedStateIdx**, **shrinkedState**🡨 **transObs(stateObs)** |
| 2 | **if** randomNumber<EPSILON |
| 3 | **return** random action |
| 4 | **else**: |
| 5 | **return** action with maximum value in **Q-table[shrinkedStateIdx]** |

1. 学习部分

|  |  |
| --- | --- |
| **算法 4** | **learn** |
| **输入** | **stateObs0, actionID, stateObs1, reward, info**  //action执行前状态；action；action执行后状态；reward；环境信息 |
| 1 | **stateIdx0, shrinkedState0🡨 transObs(stateObs0)** |
| 2 | **stateIdx1, shrinkedState1🡨 transObs(stateObs1)** |
| 3 | **if stateIdx0** not in **Q-table**: |
| 4 | insert **stateIdx0** with all 0 elements in **Q-table** |
| 5 | **if stateIdx1** not in **Q-table**: |
| 6 | insert **stateIdx1** with all 0 elements in **Q-table** |
| 7 | **reward🡨transformReward(reward)**//transformReward()在参数表中给出 |
| 8 | **if info**==NO\_WINNER: |
| 9 | **Q-table[stateIdx0,actionID]**🡨  **Q-table[stateIdx0,actionID]+ALPHA\*(reward+**  **GAMMA\*max(Q-table[stateIdx1])- Q-table[stateIdx0, actionID])** |
| 10 | else: |
| 11 | **Q-table[stateIdx0,actionID]**🡨  **Q-table[stateIdx0,actionID]+ALPHA\*reward** |

# 实验设计（模型训练）

本部分首先给出实验所用计算资源，再给出模型参数，并依据实验结果即理论分析给出解释。最后对作业过程中使用过的方法进行比对。

## 计算资源

所用计算资源如表1所示

|  |  |
| --- | --- |
| **表1** 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。

## Agent模型参数

算法参数如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **表2** 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 | 折扣系数 | 0.9 |
| 3 | EPSILON | ε-greedy，探索概率 | 0.01 |
| 4 | shrinkedObs | 状态空间大小 | 3×3方格 |
| 5 | iterNum | 迭代停止轮次 | 5000 |
| 6 | rcmdProb | 执行推荐行为的概率 | 0.45 |
| 7 | reward-win | 游戏胜利reward | 1000 |
| 8 | reward-failure | 游戏失败reward | -100 |
| 9 | reward-stone | 打到石头reward | 10 |
| 10 | reward-alien | 打到外星人reward | 20 |

模型为以上参数训练1200次得到。因为训练时间过长，本实验未进行网格搜索调参，以上参数都是基于有理由的猜测和与同学讨论得来。猜测原因如下：

1. ALPHA：训练速度低，提高ALPHA加快模型收敛速度。
2. GAMMA：使reward反馈到ACTION\_USE
3. EPSILON：非视野内reward可能提升3×3 observation中无用action的value。针对这次作业的问题，小视野的agent只学会处理自己视野范围内的stone以及aliens即可获得胜利，因此事业外获得的reward均是该状态的“有害reward”，因为上一刻发射的子弹会将reward加到向左向右或静止不动的操作上。实验结果也证实了在随即状态下训练出来的agent会认为左右乱晃是得分的最佳操作。
4. 参数4，6-10均在实验中加以探索。

## 实验分析

本次实验评估中使用转化reward传入Q-learning算法更新Q-table，故认为平均reward越大，模型越好。

提交agent以参数表为参数训练1200次的performance如下表所示

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表3** 提交model的performance | | | | |
| 状态空间 | 最小reward | 平均reward | 最大reward | 平均ticks |
| 3×3 part observation | 7 | 37.17 | 46 | 352 |

本部分进行了3组对比试验：

1. Full observation, Part observation分别作为state建立Q表训练对比。
2. 随机action、导向型action训练模型对比。
3. 惩罚型模型和奖励型模型对比。

以下数据均为训练1200次，取后200次作为样本数据得来，其中导向型action数据为设置引导概率为0。

### 3.4.1 full observation vs. part observation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **表4** full observation vs. part observation | | | | | |
| 状态空间大小 | 最小reward | 平均reward | 最大reward | 平均ticks | 模型文件 |
| full observation | 3 | 15.22 | 36 | 141 | 73MB |
| 3×3 part observation | 0 | 10.77 | 47 | 191 | 81KB |
| 9×3 part observation | 2 | 23 | 46 | 278 | 5.72MB |

可见full observation的performance优于part observation，可能是由于part observation的不可见空间对于模型的影响，即打到的远处的石头反馈到了目前的ACTION\_USE动作上。反馈到游戏界面即为模型高频左右晃动，外星人在面前也不打。。

### 3.4.2随机action vs. 导向型action

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表5**随机action vs. 导向型action | | | | |
| 状态空间大小 | 最小reward | 平均reward | 最大reward | 平均ticks |
| 随机action | 0 | 10.77 | 47 | 191 |
| 导向型action(0.6) | 7 | 37.17 | 46 | 352 |

导向型action的强大之处在于将已有的得分经验教给模型，使得模型知道怎样的action可以带来得分，另一方面也保证了agent不会乱开枪，受视野外reward的影响。对于本次实验来讲，现有的简单规则即可达成胜利。但简单规则不能涵盖所有的随机情况，所以也会有低分7分的情况存在。

### 3.4.3惩罚型模型 vs. 奖励型模型

惩罚型模型设置reward-failure=-1000，reward-win=100

奖励型模型设置reward-failure=-100，reward-win=1000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表5**惩罚型模型 vs. 奖励型模型 | | | | |
| 状态空间大小 | 最小reward | 平均reward | 最大reward | 平均ticks |
| 惩罚型模型 | 0 | 11.045 | 45 | 165 |
| 奖励型模型 | 0 | 10.77 | 47 | 191 |

依照观察和总结，惩罚项游戏失败是直接作用于当前的state和action上的，应该平均ticks要长一些（观察到飞机学会作弊不断闪过身边的外星人），reward与奖励型相当才符合预期。reward与奖励型相当是因为ticks长便有更多机会得分。该表数据暂时无法解释。

# 结论

full observation效果好，但是搜索空间巨大，仅训练1000次便是part observation的1000倍，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，在本例中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。加以引导也可以达到比较好的效果。

除去已经分析的因素外其他会影响Q-learning算法效果的因素：

1. **状态转换：**在本算法中，对于状态的转化是在总状态中截取了围绕飞机的3×3网格。但到最后发现截取状态的唯一作用只是将其变为索引唯一标识该状态。其中的信息除去引导外并没有用其他地方可以用到。也没有考虑到这样截取状态是否是最佳的截取方式。

所以再改进有两种方式。（1）尝试更大的方格。（2）尝试只截取我们需要的元素，如只截取最近的石头，最近的外星人，炸弹等非网格形状但能标识状态的元素。如果能在原始状态中选择出对于action好坏区分度更高，对于整个状态空间大小压缩度更高的标识，应该能提升算法效果。

1. **反馈：**导致单纯的Qlearning效果不好的直接原因我认为是反馈延迟造成的收益错误导致。因此这也是接下来需要继续改进的地方。可以使用Q(λ)算法，DQN算法持续改进。
2. **奖励规则：**除去环境本身给的惩罚外还可以增加与预期符合程度的奖励规则。即设定一轮游戏预期得分，如果这一轮得分低于预期则对该轮的所有action序列。如果得分高于预期则奖励该轮所有的action序列。此方法应该比较适用于MC方法。

本次作业实验数据与实际使用的模型设置发生了冲突，因为较早使用full observation产出结果不佳。而后导向型part observation产生了比较好的结果。便想当然的认为part observation各方面都优于full observation。实验数据在作业的最后一天产出，没有更多的时间修改模型和分析实验数据。态度的不严谨导致这次作业做得比较糟糕。

# 参考文献

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