基于强化学习和马尔可夫模型的出租车代理

Xiuyuan Qi
qixy1@shanghaitech.edu.cn

Ziyang Guo
guozy@shanghaitech.edu.cn

Abstract

本项目为使用人工智能完成一个出租车游戏，着重研究了三种出租车代理程序：Search Agent, Reinforcement Agent 和 Markov-Search Agent, 并对它们的不同表现进行比较。 Search Agent采用广度优先搜索算法；Reinforcement Agent采用强化学习算法；Markov-Search Agent 使用隐马尔科夫模型进行决策。 结果说明，Search Agent 一定得到最优解；在一定的训练之后Reinforcement Agent也可以得到最优解；而在加入了迷雾和天气系统的游戏中，Markov-Search Agent 有着不错的表现。

1 问题描述

在一个$M \times N$ 的地图中，每个格子与相邻的四个格子连通，如果相邻的两个格子之间存在墙壁，则两者不再连通。

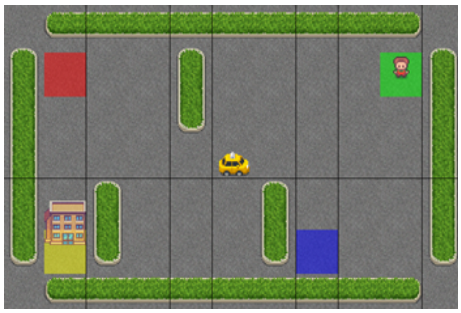


Figure 1: A possible map.

每一轮有1名乘客和1个乘客想前往的目的地。

地图中一共有4个乘车点，每局游戏开始时乘客和目的地会随机刷新在不同的乘车点。出租车会随机出生在地图中。

出租车每步可以进行上/下/左/右移动（从一个格子移动到与之连通的另一个格子），或者进行载客/下客操作。

当出租车位于乘客所在的乘车点，且乘客不在车内时，进行上车操作会使乘客转移到车中。否则操作无效。

当出租车位于目的地，且乘客在车内时，进行下车操作会使乘客转移到车所在的地块，并且本局游戏结束。否则操作无效。

得分规则

- 无效的上下车：-10 （与乘客不在同一个格子的情况下上车、车上无乘客或不在目的地时下车）

Preprint. Under review.

- 将乘客送达目的地：+20（乘客在车中、车在目的地时进行下车操作）
- 其他：-1 （移动、合法上下车）

Agent 的目的是使得分最大化，即以尽量少的步数将乘客送达目的地。

若出租车未能在200步内将乘客送达目的地，则本局游戏将强制结束，以最终得分为本局得分。

1.1 附加规则

为了增加难度与不确定性、以及添加前后局之间的关联性，对于部分游戏局，我们添加了以下的附加规则：

1.1.1 迷雾

在添加了迷雾的游戏中，存在一个额外的参数V ，只有当乘客与出租车的横、纵距离均小于V时，出租车才能收到乘客的位置信息，否则出租车无法知道乘客的位置。

1.1.2 天气

在连续的多局游戏中，存在一个参数“天气”W。 共有3种天气：晴天($W=0$)、阴天($W=1$)、雨天($W=2$)。

每局天气固定。 本局的天气决定下一局各天气出现的概率， 同时决定本局中乘客在各乘车点的出现概率。

暂定天气转移矩阵T如下：

$$T = P(W_{t+1}|W_t) = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.15 & 0.4 & 0.45 \\ 0.3 & 0.4 & 0.3 \end{pmatrix}$$

其中W_t为第t局时的天气变量。设W_t的取值为w_t，则$(T)_{w_t,w_{t+1}} = P(w_{t+1}|w_t)$，即$T$中第$w_t$行第$w_{t+1}$列元素为本局天气为$w_t$的情况下下一局天气为$w_{t+1}$的概率。

暂定天气影响乘客概率的矩阵如下：

$$P(L_t|W_t) = \begin{pmatrix} 0.2 & 0.1 & 0.1 & 0.6 \\ 0.1 & 0.4 & 0.4 & 0.1 \\ 0.7 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

其中L_t为第t局中的乘客所在乘车点变量，即乘客出现在了4个乘车点中的哪一个。

暂定初始天气分布$P(W_0)$如下：

$$P(W_0) = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

即第0局中3种天气的出现概率均等。

2 使用的模型

本项目着重研究了三种游戏 Agent 模型：Search Agent, Reinforcement Agent 和 Markov-Search Agent.

2.1 Search Agent

Search Agent 在拥有完整信息的地图中活动（Agent 得知乘客与目的地的确切位置）。

Search Agent 采用广度优先搜索算法 (BFS)，先搜索一条通往乘客位置的路径，沿路径前往上车点进行上车操作后，再搜索一条通往目的地的路径，前往目的地让乘客下车。

2.2 Reinforcement Agent

同 Search Agent 一样， Reinforcement Agent 也得知乘客与目的地位置。

Reinforcement Agent 采用了Q-Learning算法，其中与课堂中例子的区别是存在多个结束状态， 算法上采用observation state-action pair作为Q table的索引。相比可以画在地图上的Q Table ，这样会形成一个多维的Q Table，更类似吃豆人的例子。

其次，采用exploration function的方式来鼓励探索。更新Q value的公式如下:

$$Q(s,a) = \alpha R(s,a,s') + \gamma max_{a'} Q(s',a') + \frac{explore\_constant}{N(s,a)}$$

其中， $N(s,a)$ 对应在 s状态下选择a行动的次数；explore_constant 为超参数，默认为1。

另外，我们在编写对应的函数时还考虑了训练时间的问题：为了避免在学习次数或其他参数改变 时需要较多时间重新训练模型，函数中支持传入已经

2.3 Markov-Search Agent

在附加规则下，天气系统可看作是一个隐马尔科夫模型 (HMM)，能以贝叶斯网络的形式表示为下图:

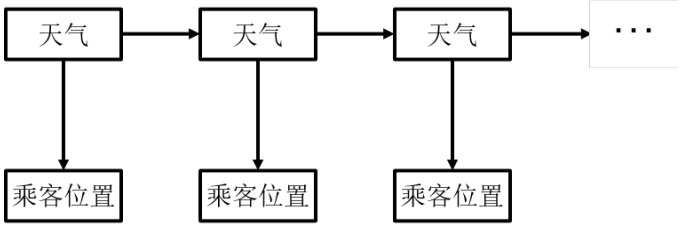


Figure 2: Weather HMM

Markov-Search Agent 可基于已知信息，计算出当前游戏局乘客在各乘车点的出现概率，据此决定出租车的行动。具体算法如下:

1. 使用 Filtering Algorithm 计算当前各天气的概率:

$$P(W_t|l_{0:t}) = \alpha P(l_t|W_t) * (P(W_{t-1}|l_{0:t-1})P(W_t|W_{t-1}))$$

   其中$*$号表示矩阵对应元素相乘，不显示符号的乘法表示矩阵乘法。α为归一化系数，使得结果向量中的各元素和为1.

   l_t表示L_t的取值。$l_{0:t}$等同于$l_0, l_1, \ldots, l_t$.

2. 计算当前乘客出现在各乘车点的概率:

$$P(L_t|l_{0:t-1}) = P(W_{t-1}|l_{0:t-1})P(W_t|W_{t-1})P(L_t|W_t)$$

3. 根据概率计算每个位置的期望得分
4. 搜索一条通往期望最高的乘车点的路径；如果到了附近看到没有乘客，则搜索通往期望第二高的乘车点的路径，如此以往

3 表现评估

3.1 Search Agent

由于 BFS 算法性质保证了搜索结果为最短路径，且载客、送客两步均不可跳过，可知 Search Agent 必然能得到最优解。因此 Search Agent 可作为另外两种 Agent 的表现参考。

3.2 Reinforcement Agent

3.2.1 学习过程

在本游戏中，如果agent成功将乘客送达目的地，其得分一般为正数。 并且由于存在200次的时限，agent的得分区间为[-2000,20)。在训练过程中可以发现，一个常见的失败情况是agent一直向着某个方向移动，并在-200分时退出本局。将不同学习率下agent训练不同次数的得分情况 可视化后得到下图。

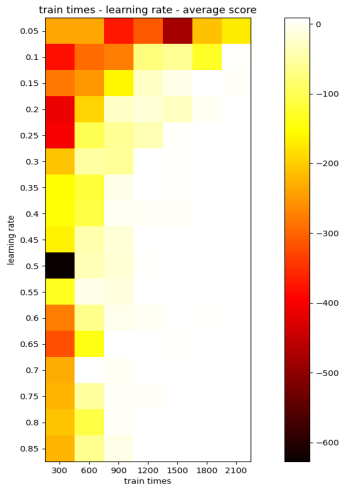
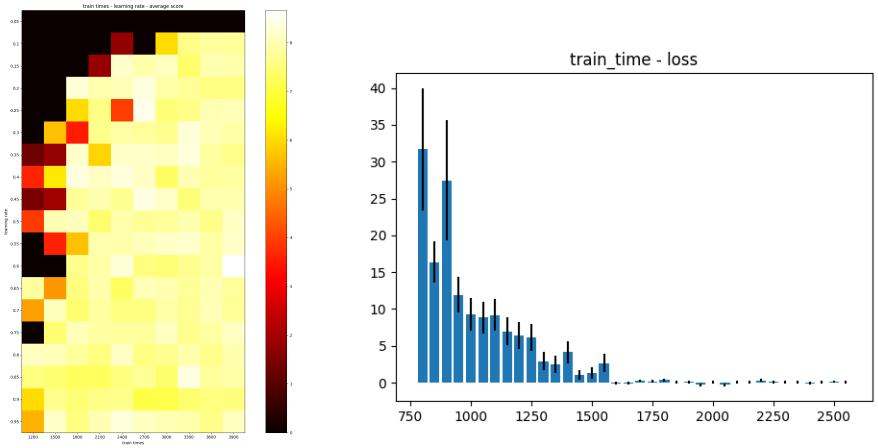


Figure 3: scores under different learning rate and training times.

从图中可以看出：随着学习率下降，达到正得分并收敛所需的训练次数逐渐减少。 然而由于负数部分绝对值较大，较难判断完成游戏时采取的策略的好坏（即正分数的相对大小）因此采用

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

函数来对得分进行处理，结果如下图左。收敛后的得分基本在8分左右，观察学习率为0.4的损失曲线 得知在约1600次训练后loss基本稳定为零，即收敛到了最优解。



(a) scores under relu function.

(b) loss figure with error

Figure 4: scores and loss

3.2.2 表现对比

在本游戏中，由于Search Agent一定会得到最优解，因此损失曲线同时也是其他Agent与Search Agent的得分差距。学习率>0.2 时Reinforcement Agent 基本都可以在2000次迭代后收敛 并达到与Search Agent相同的最优效果。

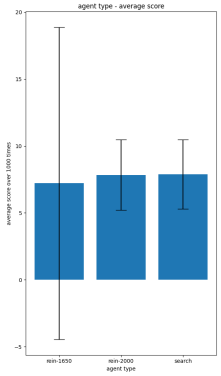


Figure 5: scores under different learning rate and training times.

另外，在未收敛的情况下，很大一部分得分为-200；也有少部分情况有更低分。随着训练次数增加 得分一般会呈现先快后慢的增长趋势。

此外，我们还对游戏环境进行了一定的改动，来测试在更大的地图下该AI的表现。 大地图的字符串如下

```
"+-----------------------+",
"|␣|R:␣:␣:␣:␣|␣:␣:␣|␣:␣:␣|",
"|␣|␣:␣|␣:␣:␣|␣:␣:G|␣:␣:␣|",
"|␣:␣:␣:␣|␣|␣:␣:␣:␣:␣|␣:␣:␣|",
"|␣:␣:␣:␣|␣|␣|␣:␣:␣:␣|␣:␣:␣|",
"|␣:␣:␣:␣:␣|␣|␣|␣:␣:␣:␣|B|␣:␣|",
"|␣:␣:␣:␣:␣|␣:␣|␣|␣:␣:␣:␣|␣|␣|",
"|␣:␣:␣:␣:Y|␣:␣|␣:␣:␣:␣:␣:␣|",
"+-----------------------+",
```

其中，| 代表墙壁; : 代表可以通过; R,G,B,Y代表四个上下车点。在经过测试后发现 其学习过程的得分曲线与小地图中基本一致：都是类似反比函数的形式； 且随学习率提高收敛所需训练次数逐渐降低，并在0.5左右达到最低值。

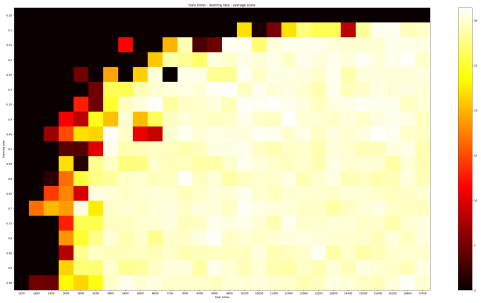


Figure 6: scores on big map after ReLU.

3.3 **Markov-Search Agent** 在附加规则下的表现

在添加了迷雾以及前后连续的天气的游戏中，我们对Markov-Search Agent进行了测试。 得到如下结果：

Average Score	Standard Division
5.72233	6.4862

其得分大概是无迷雾条件下最优得分的72%，并且标准差也有一定程度的增大， 不过因为存在较大的不确定性，这种差距也是可以理解的。

4 主要贡献

郭子杨：Search Agent；天气系统；Markov-Search Agent

齐修远：基础环境配置；迷雾系统；强化学习Agent；表现可视化

项目GitHub仓库：`https://github.com/PandragonXIII/AI-team-project`

5 参考文献

Gymnasium API: `https://gymnasium.farama.org/environments/toy_text/taxi/`

CS181 Slides