人工智能作业

学生姓名：董安宁

班 学 号：111171-05

指导教师：叶雅琴

中国地质大学信息工程学院

2020 年 3 月 7 日

目录

作业五 Expectimax+ Evaluation Function

1、实现 Expectimax

期望最大(Expectimax)是在 MINIMAX 的基础上进行了概率的计算，这个最开始我还是很没有理解的，后来和叶老师与 **111173 田鑫**讨论后才悟了，如图 1:

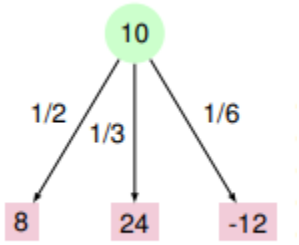

图 1 一颗普通的状态树

$$\sum_{i}^{n}(p_i * value_i)$$

图 2 新的计算公式

这一层原本是 MINI 层，按照原来的算法，应该在其子节点中选择一个最小值传递到上一层，因此结果应该是-12。但现在添加了到达每个节点的概率，计算的方式就变成了求期望值，即图 2 的公式。其中 p 表示概率，value 该点的值。这样一来，该点的值就是 8 * 1/2+ 24 * 1/3 + -12 * 1/6 = 10，就不是之前的-12 了。

在吃豆人问题中，由于向每个方向走的概率都是相同的，因此只需要将子节点求和后除以合 action 的数量，就可以达到 Expecrimax 所描述的计算要求。

1.2 结果展示

通过对比 AphasiaBeta 和 Expectimax 对同一个地图进行测试，可以得到如下结果：

```
PS E:\迅雷下载\multiagent> python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3 -q -n 10
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Average Score: -501.0
Scores:        -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0
Win Rate:      0/10 (0.00)
Record:        Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss
PS E:\迅雷下载\multiagent> python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Average Score: 118.4
Scores:        -502.0, -502.0, 532.0, -502.0, 532.0, -502.0, 532.0, 532.0, 532.0, 532.0
Win Rate:      6/10 (0.60)
Record:        Loss, Loss, Win, Loss, Win, Loss, Win, Win, Win, Win
```

图 3 结果截图

可以看到相比之下，AphasiaBeta 一场都没有赢过，而 Expectimax 赢了半数以上的对局，足以说明其策略更优。

2、实现 Reflex

在实现自定义评价函数的时候没懂是什么意思，因此和叶老师沟通后，开始着手做第一题。因为第一题也有自定义评价函数的内容，与第五题类似。Reflex 中基本的思路就是

① 离食物越近越好

② 离鬼越远越好

因此评分标准就要围绕着上面两个思想来进行。首先离食物越近越好这个可以用倒数的形式来完成，而离鬼越远越好的话，我通过很长一段时间考虑…觉得用那种“离鬼越近负分越高”的评价会比较好，因此我决定用一个呈指数级别的函数，即 e 的-x 次方。此时横坐标的 x 表示的就是吃豆人离鬼的距离。我们可以分析得到，当距离为 0 的时候是最危险的，因此其负的分数更高，如图 5 所示。

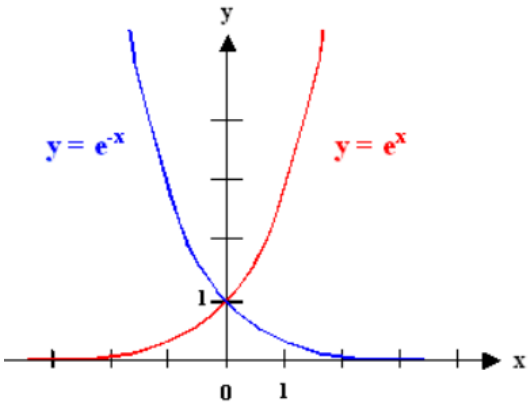

图 4 给我启发的 exp 的图象

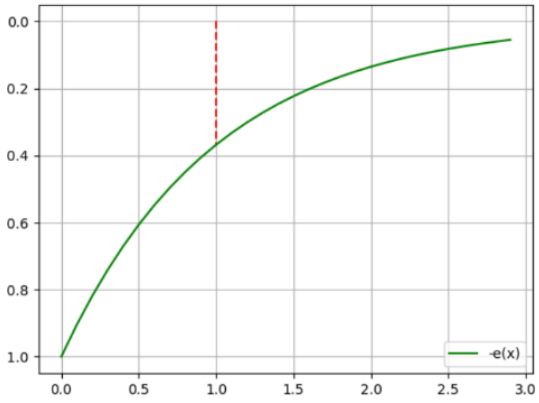

图 5 最终使用的-exp 的图象

接下来的代码就比较玄学了，我直接**(1/到最近食物距离)-exp(到鬼的距离)**，并且也尝试了很久调试参数，但是无论如何都不能使其运作的很好…最终我在 github 上参考了别人的代码，发现他也用的 exp 进行权值计算。但是他首先利用到了“oldFood”，用移动之前的食物记录，来判断当前这次有没有吃到食物，有的话就设置一个很高的分数，没有的话就设置为 0。然后她也计算出了“到最近食物距离”和“到鬼的距离”，用刚才的分数减去这两个值。虽然不知道他咋想的，但这样确实管用…下面附上这位同学的代码链接：

https://github.com/mrbeiley/multi-agent-pacman/blob/master/multiAgents.py

```
PS E:\迅雷下载\multiagent> python autograder.py -q q1 --no-graphics
Starting on 3-7 at 11:11:05

Question q1
===========

Pacman emerges victorious! Score: 1257
Pacman emerges victorious! Score: 1259
Pacman emerges victorious! Score: 1248
Pacman emerges victorious! Score: 1254
Pacman emerges victorious! Score: 1257
Pacman emerges victorious! Score: 1237
Pacman emerges victorious! Score: 1245
Pacman emerges victorious! Score: 1255
Pacman emerges victorious! Score: 1259
Pacman emerges victorious! Score: 1254
Average Score: 1252.5
Scores:        1257.0, 1259.0, 1248.0, 1254.0, 1257.0, 1237.0, 1245.0, 1255.0, 1259.0, 1254.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```

图 6 这位同学的精妙算法结果

3、实现 Evaluation Function

实现 Evaluation Function 的时候，我先按照自带的评价函数 scoreEvaluationFunction 直接计算 state 的得分，并作为评价值返回。运行结果如下：

```
PS E:\迅雷下载\multiagent> python autograder.py -q q5 --no-graphics
Starting on 3-6 at 17:29:02

Question q5
===========

Pacman emerges victorious! Score: 833
Pacman emerges victorious! Score: 1285
Pacman emerges victorious! Score: -260
Pacman died! Score: -487
Pacman emerges victorious! Score: -178
Pacman died! Score: -139
Pacman emerges victorious! Score: -584
Pacman emerges victorious! Score: 779
Pacman emerges victorious! Score: -494
Pacman emerges victorious! Score: 390
Average Score: 114.5
Scores:        833.0, 1285.0, -260.0, -487.0, -178.0, -139.0, -584.0, 779.0, -494.0, 390.
Win Rate:      8/10 (0.80)
Record:        Win, Win, Win, Loss, Win, Loss, Win, Win, Win, Win
*** FAIL: test_cases\q5\grade-agent.test (3 of 6 points)
***     114.5 average score (0 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     8 wins (2 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 3/6 ###


Finished at 17:31:30

Provisional grades
==================
Question q5: 3/6
------------------
Total: 3/6
```

图 7 系统自带的评价函数结果

效果似乎还不错…因此我决定在这个的基础上进行改进。

由于阅读源码没有看懂它自带的 score 打分规则，因此我通过输出原来的得分进行观察，想知道他的分数有什么规律。打印的结果如右图所示。通过将分数和局面进行对比，我发现正的分数就是正常显示在左下角的 Score 分数，表明了当前吃着豆子得到的分数。而当分数为负数的时候，是鬼就在吃豆人附近的时刻。此时吃豆人因为分数变成负数，因此会在接下来的操作中选择躲避，提高自己的存活率。

```
329.0
329.0
329.0
-171.0
-171.0
329.0
329.0
-171.0
-171.0
329.0
```

图 8 打印分数

由于原来的评分功能已经很好的实现了正常吃豆和躲避鬼的代码，因此这里我添加了吃“药丸”的权重判断(在阅读源码的时候发现了 getCapsules 这个函数，可以获得地图中药丸的坐标)。在这个判断的规则下，吃豆人会更倾向于靠近药丸，并且在吃到药丸后，也更倾向于追逐鬼。这里我参考了 Reflex 中的计算方法，即计算得到曼哈顿距离后用 exp 计算获得一个较大的分数。代码如下：

```
# 离药丸越近分数越高
capsules = currentGameState.getCapsules()
capsulesDis = [manhattanDistance(newPos,capsule) for capsule in capsul
capsuleValue = 0
if len(capsulesDis)!=0:
    capsuleValue = min(capsulesDis)
    capsuleValue = math.exp(-(capsuleValue-5))

# 当吃了药丸后，离鬼越近分数越高
gostValue = 0
if newScaredTimes[0]>0:
    gostDis = [manhattanDistance(newPos,ghost) for ghost in gostPos]
    if len(gostDis)!=0:
        gostValue = min(gostDis)
        gostValue = math.exp(-(gostValue-5))
else:
    gostValue = -math.exp(-(gostValue-5))
```

图 9 评价计算公式

这两段代码的计算方式十分相似，都是得到所有药丸/鬼的位置，然后计算曼哈顿得到一个最近的距离，对这个距离进行形如 math.exp(-(Value-5))的计算，便可得到评价值。需要说明的是，当吃豆人没有吃到药丸的时候，对鬼的分数依然是要打负分，因此有了图 9 中的那个 else 语句。

最终将原本的评分数值加上图 9 两个计算出来的两个值，就是最终的评分了，代码如下

```
# 原本系统得分
systemValue = scoreEvaluationFunction(currentGameState)
finalValue = systemValue+1.5*capsuleValue+gostValue
return finalValue
```

图 10 最终评价计算

需要注意的是，这里加了一个小 tick：药丸得到的分数我乘以了 1.5 倍，因为如果不乘以这个系数的话，最终的平均得分是小于 1000 的，只能拿到这题的 5/6 分，如图 11：

```
Average Score: 943.3
Scores:        1362.0, 1150.0, 1220.0, 569.0, 1197.0, 1226.0, 329.0, 525.0, 840.0, 1015.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** FAIL: test_cases\q5\grade-agent.test (5 of 6 points)
***     943.3 average score (1 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 5/6 ###
```

图 11 一次不太完美的结果

之所以乘以 1.5，这是在和 **111172 周伟国**同学讨论 Reflex 的时候他给我提供的思路，就是当好几个评分结果都影响最终得分的时候，可以设置权重系数，把你认为影响最大的那个结果通过系数放大，影响小的就通过系数变得更小。这样一来就能将局势判断朝你更倾向的那个思路进行下去。最终 PASS 的结果如图 12 所示：

```
PS E:\迅雷下载\multiagent> python autograder.py -q q5 --no-graphics
Starting on 3-7 at 16:24:36

Question q5
===========

Pacman emerges victorious! Score: 1362
Pacman emerges victorious! Score: 1150
Pacman emerges victorious! Score: 1220
Pacman emerges victorious! Score: 569
Pacman emerges victorious! Score: 1265
Pacman emerges victorious! Score: 1345
Pacman emerges victorious! Score: 1156
Pacman emerges victorious! Score: 1261
Pacman emerges victorious! Score: 723
Pacman emerges victorious! Score: 406
Average Score: 1045.7
Scores:        1362.0, 1150.0, 1220.0, 569.0, 1265.0, 1345.0, 1156.0, 1261.0, 723.0, 406.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q5\grade-agent.test (6 of 6 points)
***     1045.7 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 6/6 ###


Finished at 16:25:50

Provisional grades
==================
Question q5: 6/6
------------------
Total: 6/6
```

图 12 Perfect!

4、总结

学到的东西

① 想要放大距离对分数的影响可以用指数函数，其曲线满足分数距离越近越高，越远越低。同样的，也可以用系数来扩大分数的影响。

② 相比预测全部确定状态做出的决定，有时不如考虑随机性的因素而做出的决定