PacMan 游戏实验报告

银琦（141220132、141220132@smail.nju.edu.cn）

(南京大学 计算机科学与技术系, 南京 210093)

1 策略模型用什么表示？该表示有何缺点？有何改进方法？

代码中采用的策略模型是 Q-Learning 算法，它是一个脱离策略(off-policy)的学习算法，该算法的伪代码如下：

1 设置 gamma 相关系数，以及奖励矩阵 R
2 将 Q 矩阵初始化为全 0
3 For each episode：
设置随机的初使状态
Do While 当没有到达目标时
选择一个最大可能性的 action
根据这个 action 到达下一个状态
根据计算公式：Q(state, action) += alpha * (R(state, action) + Gamma * (Max[Q(next state, all actions)] - Q(state,action))) 计算这个状态 Q 的值
设置当前状态为所到达的状态
End Do
End For

该算法的缺点：即使我们已经学习完毕了，已经知道了最优解，当我们选择一个动作时，它还是会继续采取随机的动作。

改进方法：每循环一次就减少 epsilon 的值，这样随着学习的进行，随机越来越不容易触发，从而减少随机对系统的影响。

2 Agent.java 代码中 SIMULATION_DEPTH, m_gamma, m_maxPoolSize 三个变量分别有何作用？

SIMULATION_DEPTH 限制了模拟的次数，代码中将该值设置为 20，即在每行动一步之前，对该状态模拟 20 次，得到最优解，避免由于一系列不常见的糟糕结果而错过最优行动的有限概率。

m_gamma 是一个 0-1 之间的值，在原代码中设置为了 0.99，如果 m_gamma 接近 0，对立即的奖励更有效；如果接近 1，整个系统会更考虑将来的奖励。

m_maxPoolSize 是限制数据集中实力个数的值，原代码中设置为 1000，即若经过模拟器得到的实例数超过了 1000，则从前开始删除，直到实例数小于 1000。

3 QPolicy.java 代码中，getAction 和 getActionNoExplore 两个函数有何不同？分别用在何处？

在 getAction 中，比 getActionNoExplore 多了三行代码：

```
if( m_rnd.nextDouble() < m_epsilon ){
    bestaction = m_rnd.nextInt(m_numActions);
}
```

这三行代码的作用：生成一个随机数，若这个随机数小于 epsilon，则进行 exploration，exploration 是指选择之前未执行过的 actions，从而探索更多的可能性；若随机数大于 epsilon，则进行 exploitation，即只根据当前的有效信息选择一个最佳的 action。

如果不进行 exploration 也许找到的只是个局部的极值，在 getAction 中进行了 exploration，而 getActionNoExplore 中没有进行 exploration，只是进行了 exploitation。

getAction 用在在模拟器上模拟获得数据集时，getActionNoExplore 用在模拟结束后实际选择下一步的操作时。

4 尝试修改特征提取方法，得到更好的学习性能，并报告修改的尝试和得到的结果。

增加三类特征，共 6 个特征，为方便描述，设这 6 个特征为(s1, s2, s3, s4, s5, s6)

第一类特征(s1, s2, s3, s4)：对应代码中 feature[874]-feature[877]，该类特征描述了上左下右四个方向在 5 步之内会不会遇到 monster，取值为 0 或 1，此处的步数为曼哈顿距离，且为启发式的，即没有探索路径，只是简单的取 d=|x1-x2|+|y1-y2|；

```
Vector2d ava = obs.getAvatarPosition();
int ava_x = (int)(ava.x/20);
int ava_y= (int)(ava.y/20);

for(Observation o : NPCobj){
    Vector2d p = o.position;
    int x = (int)(p.x/20);
    int y= (int)(p.y/20);
      System.out.println(o.itype + " " + x + " " + y);
    int dis = Math.abs(ava_x-x) + Math.abs(ava_y-y);
    feature[874]=0;
    feature[875]=0;
    feature[876]=0;
    feature[877]=0;
    feature[878]=0;
    feature[879]=-1;

    if (dis < 5) {
        if(ava_x < x && map[ava_x+1][ava_y]!=0) {  //right
            feature[877]=1;
        }
        else if (ava_x > x && map[ava_x-1][ava_y]!=0) {  //left
            feature[875]=1;
        }
        if(ava_y < y && map[ava_x][ava_y+1]!=0) {   //down
            feature[876]=1;
        }
        else if (ava_y > y && map[ava_x][ava_y-1]!=0) {  //up
            feature[874]=1;
        }
    }
}
```

第二类特征(s5)：对应代码中 feature[879]，该特征描述了 PacMan 在 5 步之内不遇到 monster 的情况下，哪个方向有 food、pellet 或者 power，取值为 0,1,2,3，分别代表上左下右四个方向。

```
if(map[ava_x][ava_y-1]==3 && feature[874]!=1) {    //up
    feature[879]=0;
}
else if(map[ava_x-1][ava_y]==3 && feature[875]!=1) {    //left
    feature[879]=1;
}
else if(map[ava_x+1][ava_y]==3 && feature[876]!=1) {    //down
    feature[879]=2;
}
else if(map[ava_x][ava_y+1]==3 && feature[877]!=1) {    //right
    feature[879]=3;
}

if (feature[879]==-1) {
    if(map[ava_x][ava_y-1]==4 && feature[874]!=1) {    //up
        feature[879]=0;
    }
    else if(map[ava_x-1][ava_y]==4 && feature[875]!=1) {    //left
        feature[879]=1;
    }
    else if(map[ava_x+1][ava_y]==4 && feature[876]!=1) {    //down
        feature[879]=2;
    }
    else if(map[ava_x][ava_y+1]==4 && feature[877]!=1) {    //right
        feature[879]=3;
    }
}
```

第三类特征(s6)：对应代码中 feature[878]，该特征描述了 monster 是否被困住了，即上左下右四个方向是否除了墙就是 monster，取值为 0 或 1。

```
int trapped = 0;
if (feature[874]==1 || map[ava_x][ava_y+1]==0)
    trapped++;
if (feature[875]==1 || map[ava_x-1][ava_y]==0)
    trapped++;
if (feature[876]==1 || map[ava_x][ava_y-1]==0)
    trapped++;
if (feature[877]==1 || map[ava_x+1][ava_y]==0)
    trapped++;

if(trapped==4) {
    feature[878]=1;
}
else {
    feature[878]=0;
}
```

修改特征后学习性能得到了很大的提高，下表为修改前后 tick 和 score 的比较（因为先做的第五题，所以修改前的数据为第五题修改后的数据，只测试了 level1）：

	修改特征前		修改特征后	
次数	tick	score	tick	score
1	1000	144	552	226
2	718	198	1000	188
3	1000	109	479	67
4	88	12	407	184
5	556	87	735	162
6	1000	203	212	45
7	631	118	516	83
8	401	49	1000	188
9	184	33	1000	90
10	537	83	398	62
平均	611.5	103.6	629.9	129.5

从表中可以看出，修改特征后，tick 有了略微的提高，但是 score 有了很大提高。

5 尝试修改强化学习参数，得到更好的学习性能，并报告修改的尝试和得到的结果。

修改参数：采用了 1 中的改进方法，最初设置 epsilon 为 0.5，设置一个量 beta 为 0.9，在每次循环中将 epsilon 赋值为 epsilon*beta，逐渐减小 epsilon 的值，代码如图：（插图）

修改后学习性能得到了很大的提高，下表为修改前后 tick 和 score 的比较（只测试了 level1）：

	修改参数前		修改参数后	
次数	tick	score	tick	score
1	639	71	1000	144
2	213	50	718	198
3	164	29	1000	109
4	188	34	88	12
5	1000	135	556	87
6	377	61	1000	203
7	469	89	631	118
8	535	82	401	49
9	1000	152	184	33
10	222	44	537	83
平均	480.7	74.7	611.5	103.6

从表中可以看出修改参数后，tick 和 score 都得到了很大的提升，虽然 score 的最大值变大了，但是最小值也变小了，所以修改后可能更不稳定了。

因为 tick 限制在 1000 以内，所以所有的结果都是 lose，没有 win 的结果出现。

6 问题及心得

我对本次代码有个疑惑，也可能是我理解有误：我认为在特征提取部分，map 的值应该是每个位置上物体的类型，所以根据 map 应该可以区分出 PacMan 的上下左右为墙还是路，或者是 food，power 等，但是打印出 map 后发现墙、路及 power 的值都是 0，难以区分，查看代码后发现 map 的值取的是 itype，在代码中还有一个值：category，但是结合 category 和 itype 仍然不能完全区分每种物体，并且我没有找到别的可以判断物体类型的方法，所以这个问题没有解决，导致第 4 问中第二类特征没有加入对 power 的判断，仅判断了 pellet 和 food。

从上面的问题中可以看出，我读大型项目代码的能力还有所欠缺，需要进一步提高。

7 致谢

最后感谢这一学期老师的教导，也感谢辛勤的助教。

References:

[1] http://blog.csdn.net/coffee_cream/article/details/57085729

[2] https://segmentfault.com/a/1190000007813298

[3] http://202.119.32.195/cache/3/03/www.cs.uoi.gr/56d5aa6692a475b2f6b62b843e9e0d4c/PacMan_SETN2014.pdf

附中文参考文献:

[4] 人工智能 一种现代的方法 第三版