

HW4: Freeway游戏实验报告

匡亚明学院 张子谦 191240076 191240076@smail.nju.edu.cn

本人承诺该实验全程由本人独立自主完成，无抄袭或给予他人抄袭行为，代码已上传至本人github

摘要：以Freeway游戏为基础，了解强化学习过程与强化学习中的Q-Learning学习策略。通过实践修改特征提取参数与学习策略中的参数，并优化Reward函数，改进学习性能。

1. 引言

Freeway游戏流程为通过控制Avatar找准时机躲避障碍，触碰目标获得分数。通过强化学习的策略，提取游戏中的特征参数，并根据执行动作后游戏进行好坏来延迟标记信息，不断学习，提高游戏分数。

2. 实验内容

2.1 Task 1

强化学习：Agent通过执行action，对environment产生影响，environment将执行此action后的observation与reward反馈给Agent，Agent通过observation和reward构成的history决定之后的action直至任务结束。强化学习具有“延迟标记信息”的特点，reward的值以及history \rightarrow action的函数决定了其学习情况的优劣。

代码理解：首先在Agent中进行初始化操作，通过learnPolicy函数更新Q-Table，然后获取当前state下Q值最大的action，将其返回。而learnPolicy实现了10次迭代更新Q-Table的操作。期间通过simulate函数迭代到SIMULATION_DEPTH或者游戏结束，每一次迭代的过程包括了以下行为：

Step1: 从当前state对应的actions中利用 $\epsilon - greedy$ 策略挑选出最优action，并对当前state执行action

Step2: 更新state，通过公式 $Q(s, a) + = \alpha[r + \gamma \max Q(s_{new}) - Q(s, a)]$ 更新Q-Table中原先的 $Q(s, a)$ 的值。其中 α, γ 都是已经定义的常数， r 是 reward

Step3: 将数据记录在sequence数组中

Q1: 策略模型用什么表示？优缺点？

A: 策略模型用Q-Table表示，表行索引是state，列索引是action。对应的 $Q(state, action)$ 是二元组 $(state, action)$ 的Q值，Q值决定了在exploration过程中选择该action的概率。策略模型的优点在于离散环境下所需参数少且评价策略比较容易，缺点在于数据量大的情况下更新维护Q-Table会增加时间与空间的开销，显得低效。且由于在进行exploration时，虽然使用了 $\epsilon - greedy$ 策略，但仍然会存在倾向于选择已经执行过的action的行为。改进方法：可以考虑Deep-Q-learning策略，在探索的过程中训练网络，用Deep Networks框架来近似逼近Q value；也可以考虑SARSA方法，先执行action再返回Q值。

Q2: SIMULATION_DEPTH, m_gamma, m_maxPoolSize

A: SIMULATION_DEPTH 是simulate中向下迭代（代码理解中的Steps）的次数，从当前state执行SIMULATION_DEPTH次action；m_gamma是公式中的 γ 参数，是reward的衰减值；m_maxPoolSize控制dataset的大小，保证训练数据容量一定，且在数量超过时，通过循环随机删除，保证了数据分布的随机性。

Q3: getAction 和 getActionNoExplore 两个函数有何不同？分别用在何处？

A: getActionNoExplore用于在给出最终返回的bestAct，只参考 $Q(s, a)$ 的值，选取最优解。

getAction用于在simulate迭代过程中的exploration, 利用 $\epsilon - greedy$ 策略, 获取随机数 r , $r > \epsilon$ 时随机选取action, 否则按照Q-Value选取最优action, 减少选择已执行过行为的倾向。在课堂中以K-摇臂老虎机为例对该策略进行过讲解。

2.2 Task 2

实验原先的特征提取方法也是将地图中每一格的信息导入, 不过框架代码中, 对地图尺寸表述有问题, 应为28x15, 对map数组大小进行了修改;

首先对游戏过程进行分析, **得分条件**为Avator触碰到Portal; 过程中需要避免被移动物体撞击; 可以通过停在无移动物体的行暂时躲避。无移动物体的行有障碍物, 需保证停在空隙中。

根据以上特点, 需要添加下列参数:

(1) xdis2portal&ydis2portal

计算Avator到Portal的x方向距离, 和y方向距离, 这是体现得分的最直接特征。且游戏的进程主要就体现在y坐标上的变动上, 因此将这两项数值抽出作为特征

```
double xdis2portal = 0;
double ydis2portal = 0;
if(obs.getPortalsPositions()!=null)
    for(ArrayList<Observation> l:obs.getPortalsPositions())
        allobj.addAll(l);
for(Observation o : allobj){
    Vector2d p = o.position;
    int x = (int)(p.x/28);
    int y= (int)(p.y/28);
    map[x][y] = o.itype;

    if(o.itype==4) {
        //set property dis2portal
        xdis2portal = Math.abs(x-avator_x);
        ydis2portal = Math.abs(y-avator_y);
    }
}
```

(2) xdis2gap&ydis2gap

在向Portal前进的过程中, 需要借助无移动物块的行的间隙处进行暂时的躲避, xdis2gap和ydis2gap分别表示avator到最近(曼哈顿距离度量)空隙的x距离和y距离, xdis2gap表征此时avator应该向左还是向右移动, ydis2gap与之后移动物块相关特征联合作为avator向前时机的判断依据。由于挡板和挡板之间一定有空隙, 所以不妨直接用挡板到avatar的距离作为代替。

```
Vector2d []blocks = new Vector2d[50];
int block_cnt = 0;
int highest_block = 15;
double xdis2gap = 100;
double ydis2gap = 100;
for(Observation o : allobj){
    Vector2d p = o.position;
    int x = (int)(p.x/28); //square size is 20 for pacman
    int y= (int)(p.y/28); //size is 28 for Freeway
    map[x][y] = o.itype;
    if(o.itype==13) {
        blocks[block_cnt].x = (double)x;
```

```

        blocks[block_cnt].y = (double)y;
        block_cnt++;
        if(y<highest_block) highest_block = y;
    }
}
if(avatar_y<highest_block) {
    xdis2gap = 0;
    ydis2gap = 0;
}
else {
    for(int i=0;i<block_cnt;i++) {
        if(blocks[i].y<avatar_y) {
            if(Math.abs((int)blocks[i].y-avatar_y)+Math.abs((int)blocks[i].x-
avatar_x<xdis2gap+ydis2gap) {
                xdis2gap = Math.abs((int)blocks[i].x-avatar_x);
                ydis2gap = Math.abs((int)blocks[i].y-avatar_y);
            }
        }
    }
}
}
}

```

(3) xdis2wedge&ydis2wedge

在Avatar前进过程中如果碰到移动物块就会被强制退回到最底部，因此有必要掌握在Avatar上方距离最近的车的x距离和y距离。具体方法与上两个特征提取方法类似。

(4) 部分保留原特征

对于原特征提取，其将整张地图的信息存放了进去，而对于FreeWay游戏，其无需关注身后的游戏状态，且可以将整个游戏过程划分为多个从一个安全行转移到另一个安全行的阶段性任务。因此不妨每次将Avatar前方4行的信息作为特征提取出来。同时注意到Avatar可能有向下躲避的需要，所以将所在行与其下方两行的信息也提取出来

```

int high_y = 4;
if(avatar_y>4) high_y = avatar_y;
for(int y=high_y-1;y>=high_y-4;y--){
    for(int x=0;x<28;x++) {
        feature[index++] = map[x][y];
    }
}
int low_y = 13;
if(avatar_y<13) low_y = avatar_y;
for(int y=low_y;y<low_y+2;y++){
    for(int x=0;x<28;x++) {
        feature[index++] = map[x][y];
    }
}
}

```

同时原有的Gametick,Avatar_speed等六个特征均保留。

修改后效果:

对特征提取进行修改后发现Avatar在躲避移动物块方面有了大的提升，且向Portal靠近。但仍然无法达到Portal获得分数，主要问题是仍然会出现Avatar在剩下最后一条命前对命并不在意，所以第一次的自杀行为导致之后几次仍然会自杀使得AvatarHealthPoints特征趋于相同数值，同时Avatar会向回走进行躲避。这些问题暂且留待task3中进行解决。

2.3 Task 3

首先注意到在原框架代码中，对于reward的计算方式过于简单。根据函数调用中的参数 StateHeuristic heuristic能发现原有的evaluateState函数只根据游戏胜负和Gamepoint给予reward。但实际上，一次Q-policy只向下进行20次迭代，大部分情况下Avatar无法到达Portal，只是单纯的返回0分数，或者在只剩一条命的时候尽力选择躲避。因此需要对评估函数先做一些改动。


首先为减少自杀情况的出现，将生命值作为分数的组成部分。其次将Portal到Avatar的距离也作为评估要素，同时判断Avatar是否处于block所在的行，以保证其安全性。

综上所述，对原代码进行修改：

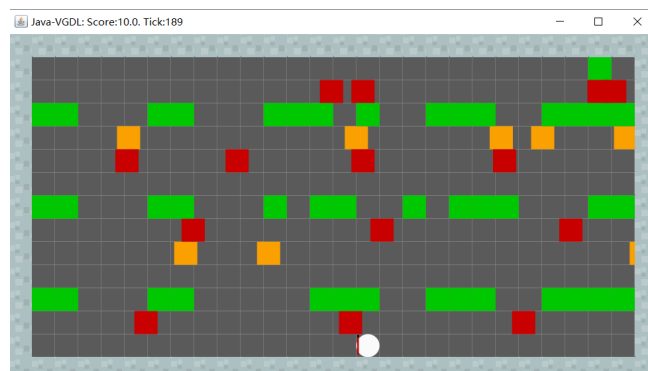
```
//增加Score权重
double rawScore = 30*stateObs.getGameScore();
//添加生命值分数
rawScore+=20*stateObs.getAvatarHealthPoints();
//添加Portal距离分数
rawScore+=(150-15*ydis2portal);
if(ydis2portal==0) {
    rawScore+=100-5*xdis2portal;
}
else {
    rawScore+=30-*xdis2portal;
}

//添加安全分数
boolean flag = true;
for(int i=0;i<y_wedge_cnt;i++) {
    if(y_wedge[i]==avatar_y) {
        flag = false;
        break;
    }
}
if(flag&&avatar_y!=1)    rawScore+=20;
else    rawScore-=20;
```

修改后，从第三次开始Avatar已经能够获得分数

 Java-VGDL: Score:10.0. Tick:130

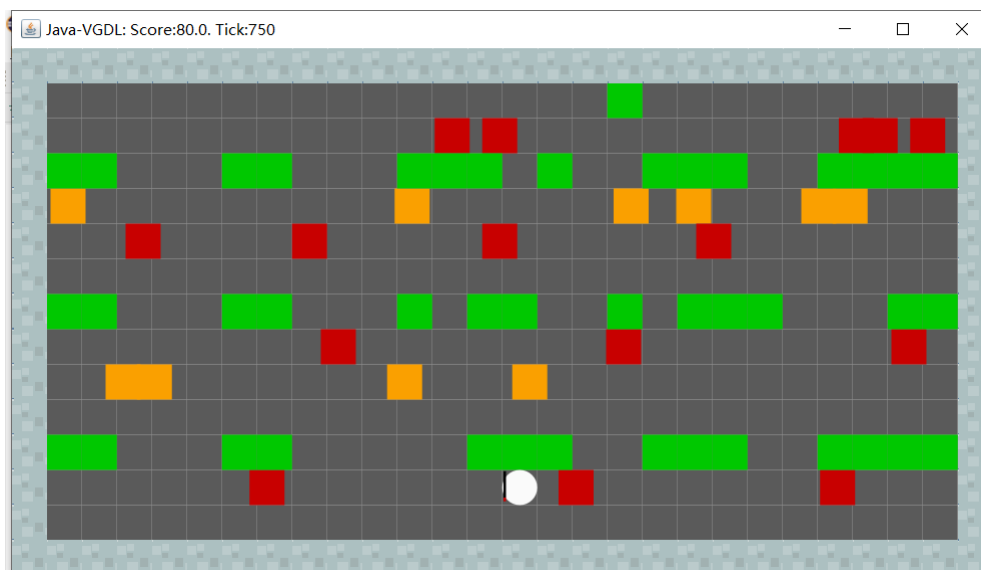
并通过在测试过程中调整权重，现大部分情况下每局游戏都能获得10分，且学习情况好时可以获得20分。



现在的Avatar能主动向上探索，注意躲避。但是表现仍然存在一些问题，下通过修改强化学习的参数尝试进行优化。

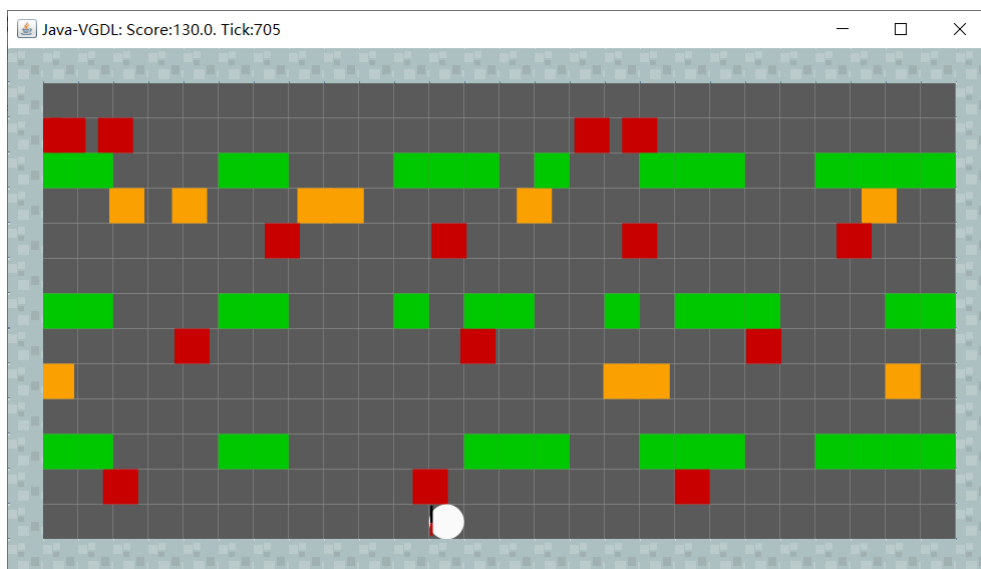
(1) Simulation Depth

原本认为Simulation Depth越深，结果应该更好，但事实却与直观相悖。当将Simulation Depth修改为5后，性能有了显著的提升，仅仅在第一局中就能够获得80分。通过测试，当Simulation Depth设为4-6时效果最佳。分析认为对于FreeWay游戏本身而言，其可被划分为多个从一个安全行转移到另一个安全行的阶段性任务，正如前文在特征提取方法改进中所提到的那样。每进行4-6步的模拟，大概率能够到达下一个安全阶段，保证了速度与分数。但当模拟层数过多时，一方面因为 $\epsilon - greedy$ 策略增多了行动的不确定性，另一方面联系评估函数，会将不利于最终解的某种局部最优解被求得，Avatar朝着此局部最优解运动。



(2) m_gamma

此参数的意义，之前也已经进行了解释，是reward的衰减值，因为reward是对局面的评估，不等于实际获得的奖励，所以需要有一个衰减，同时，我们也认为衰减过度无法确保reward能够进行有效的“扩散”。在本游戏中认为0.99的取值过高，通过测试调整，发现m_gamma取0.9时效果较好，调整后在第一局游戏中性能又获得了提升。



(3) ϵ

测试将 ϵ 值调大，性能下降，因为需要充分利用已学习的知识，保证学习性能，所以产生随机动作的概率不应过大， ϵ 过小又容易增大选择已选取动作的倾向， ϵ 处于0.2-0.3时性能差距不大，故保留原值0.3。

3. 结束语

在本次实验中，更加深入具体的了解了深度学习的过程与Q-Learning算法的整体运行流程。对特征参数的提取和启发式函数的设计有了进一步的了解。同时根据调整参数提高学习性能，结合游戏理解每个参数的含义。游戏在改进后由原来的无法获得分数到现在的学习的第一局就能获得130分体现了性能的极大提升。

References:

《机器学习》 -- 周志华

<https://www.cnblogs.com/yifdu25/p/8169226.html>