Freeway游戏实验报告

周韧哲（181220076、zhourz@smail.nju.edu.cn）

(南京大学 计算机科学与技术系, 南京 210093)

摘 要: 完成了Freeway游戏的任务

关键词: 强化学习

# 引言

在这次的实验中，使用强化学习方法玩Freeway游戏，对特征提取方法和强化学习参数进行了改进，提高了Agent的性能。

# 问题

## 问题一：策略模型用什么表示？该表示有何缺点？有何改进方法？

策略模型是带有-greedy的Q-learning方法。在框架代码中的具体实现为：实现框架为act函数调用learnPolicy来完成强化学习并返回一个最优动作以供游戏执行。在learnPolicy中外循环10次episode，每一次迭代都调用simulate来采样，采样的最大深度为20步。采样采用-greedy策略，有的概率不选取最大的Q值而是随机选取一个从而来探索，通过以下公式来更新Q值：

最后，learnPolicy使用函数fitQ来调用weka的REPTree模型来训练策略。

该策略模型表示的缺点有：特征空间与计算复杂度较大；一直选取最大的Q值可能会导致过于乐观的估计而放大了动作的效用值；在框架代码中取0.3，有0.3的概率不选取最优动作而是随机选取一个来探索，但在游戏后期探索的概率应该要有所降低因为更注重最优动作。改进方法有：调整参数来降低复杂度；使用Double Q-learning方法将Q学习中的选择和衡量进行解耦[1] ；设置多个值，根据游戏状态来选择使用较大或较小的值或者给设置一个衰变因子。

## 问题二：Agent.java 代码中 SIMULATION\_DEPTH, m\_gamma, m\_maxPoolSize 三个变量分别有何作用？

SIMULATION\_DEPTH是模拟采样的最大深度，在Agent.java中设置为20，即最大向下探索20步；m\_gamma为折扣因子，是考虑未来奖励的因子，m\_gamma越大说明其越重视未来的回报奖励，设置为0.99，在向下探索的过程中中会不断降低factor：factor\*=m\_gamma，可见执行动作越往后的状态在回报奖励中所占权重越小；m\_maxPoolSize是m\_dataset的大小，即存储的最大状态数，当Instance个数超过m\_maxPoolSize时，会删去前面的Instance。

## 问题三：QPolicy.java 代码中，getAction 和 getActionNoExplore 两个函数有何不同？分别用在何处？

getAction应用了-greedy策略，有的概率不选取最优动作而是随机选取一个来探索，而getActionNoExplore则没有应用-greedy，一直选取Q值最大的动作。getAction比getActionNoExplore多了以下代码：

//epsilon greedy

if( m\_rnd.nextDouble() < m\_epsilon ){  
 bestaction = m\_rnd.nextInt(m\_numActions);  
}

getAction用在simulate函数中，在探索时有概率探索到更多的状态，避免其陷入局部最优；getActionNoExplore用在act函数中，在决策阶段选择最优的动作。

# 修改特征提取方法

框架代码中的原有特征是记录了屏幕上所有的位置信息(每个位置是什么物体) 以及 4 个游戏状态信息:GameTick, AvatarSpeed, AvatarHealthPoints, AvatarType，还有动作信息和奖励信息。

由于Freeway游戏中没有NPC与Recource，所以在特征提取时可以忽略他们。

首先我加入了Avatar的位置信息：x坐标与y坐标，这显然是应该关注的。然后，考虑到Avatar的目标是到达最顶端的单个的格子，所以加入了Avatar到Protal的距离，并且为了突出竖直距离还加入了它到Protal的y坐标之差。加入了在Avatar前面一行的移动物体的信息和与Avatar平行的移动物体的信息，加入了在Avatar前面一行的固定物体的信息，还加入了Avatar是否可以向上移动的信息(前方是否有固定物体阻挡)。

Vector2d avatarPos=obs.getAvatarPosition();

double avatarX=avatarPos.x;

double avatarY=avatarPos.y;

double distToPortal=0;

double frontMoving=0;

double parallelMoving=0;

double portalToy=0;

double frontImmoving=0;

double up=100;

具体如下：

if( obs.getImmovablePositions()!=null ) {  
 for (ArrayList<Observation> l : obs.getImmovablePositions()) {  
 allobj.addAll(l);  
 for(Observation o:l){  
 if(avatarY==o.position.y+28) frontImmoving=Math.*min*(frontImmoving,Math.*abs*(o.position.x-avatarX));  
 if(o.position.x==avatarX && o.position.y+28==avatarY)  
 up=0;  
 }  
 }  
}  
  
if( obs.getMovablePositions()!=null ){  
 for(ArrayList<Observation> l : obs.getMovablePositions()) {  
 allobj.addAll(l);  
 for(Observation o:l){  
 if(o.position.y==avatarY) parallelMoving=Math.*min*(parallelMoving,Math.*abs*(o.position.x-avatarX));  
 if(o.position.y+28==avatarY)  
 frontMoving=Math.*min*(frontMoving,Math.*abs*(o.position.x-avatarX));  
 }  
 }  
}  
  
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) {  
 distToPortal=avatarPos.dist(p);  
 portalToy=avatarY-o.position.y;  
 }  
}  
for(int y=0; y<31; y++)  
 for(int x=0; x<28; x++)  
 feature[y\*28+x] = map[x][y];  
  
// 4 states  
feature[868] = obs.getGameTick();  
feature[869] = obs.getAvatarSpeed();  
feature[870] = obs.getAvatarHealthPoints();  
feature[871] = obs.getAvatarType();  
feature[872] = avatarX;  
feature[873] =avatarY;  
feature[874]=distToPortal;  
feature[875]=frontImmoving;  
feature[876]=frontMoving;  
feature[877]=portalToy;  
feature[878]=up;  
feature[879]=parallelMoving;

并在datasetHeader中加入相应的Attribute：

Attribute att = new Attribute("GameTick" ); attInfo.addElement(att);  
att = new Attribute("AvatarSpeed" ); attInfo.addElement(att);  
att = new Attribute("AvatarHealthPoints" ); attInfo.addElement(att);  
att = new Attribute("AvatarType" ); attInfo.addElement(att);  
att = new Attribute("avatarX" ); attInfo.addElement(att);  
att = new Attribute("avatarY" ); attInfo.addElement(att);  
att = new Attribute("distToPortal");attInfo.addElement(att);  
att = new Attribute("frontImmoving");attInfo.addElement(att);  
att = new Attribute("frontMoving");attInfo.addElement(att);  
att = new Attribute("portalToy");attInfo.addElement(att);  
att = new Attribute("up");attInfo.addElement(att);  
att = new Attribute("parallelMoving");attInfo.addElement(att);  
//action

在原来的特征下，Avatar只会待在最下面两层，从来没有上去过。在加入新的特征后，在平行的格子中，Avatar学会了躲避移动的物体，并且会向上冲，最远能到达最后一层阻挡，但常常由于血条原因，当吃了一次亏掉下来后它就很保守地移动，而非大胆地向上探索了。

# 修改强化学习参数

我发现框架代码中learnPolicy调用的启发式函数十分简单，仅仅判断游戏的输赢来给出得分。因此，我重写了SimpleStateHeuristic.java中的evaluateState函数。启发思路是离目标的距离、游戏的得分、游戏的输赢、Avatar的健康值、距离最近的移动物体和距离最近的固定物体。在以下代码中，avatarPos是Avatar的位置，won代表游戏输赢的启发得分，score为总得分，minImmoving为距离最近的固定物体的位置，immovingDist为距离最近的固定物体离Avatar的距离，minMoving为距离最近的移动物体的位置， movingDist为距离最近的移动物体离Avatar的距离，protalPos为目标的位置，protalDist为Avatar离目标的距离。Score为这些参数的加权和。另外，再考虑惩罚项，若Avatar与移动物体处于同一格子，则直接令score极小。

public double evaluateState(StateObservation stateObs) {  
 Vector2d avatarPos=stateObs.getAvatarPosition();  
 ArrayList<Observation>[] movingObj=stateObs.getMovablePositions();  
 ArrayList<Observation>[] immovingObj=stateObs.getImmovablePositions();  
 ArrayList<Observation>[] protal=stateObs.getPortalsPositions();  
  
 double won=0;  
 double score=0;  
 if(stateObs.getGameWinner()==Types.WINNER.*PLAYER\_WINS*) won=1000000;  
 else if(stateObs.getGameWinner()==Types.WINNER.*PLAYER\_LOSES*) won=-9999999;  
  
 Vector2d minImmoving;  
 double immovingDist=9999999;

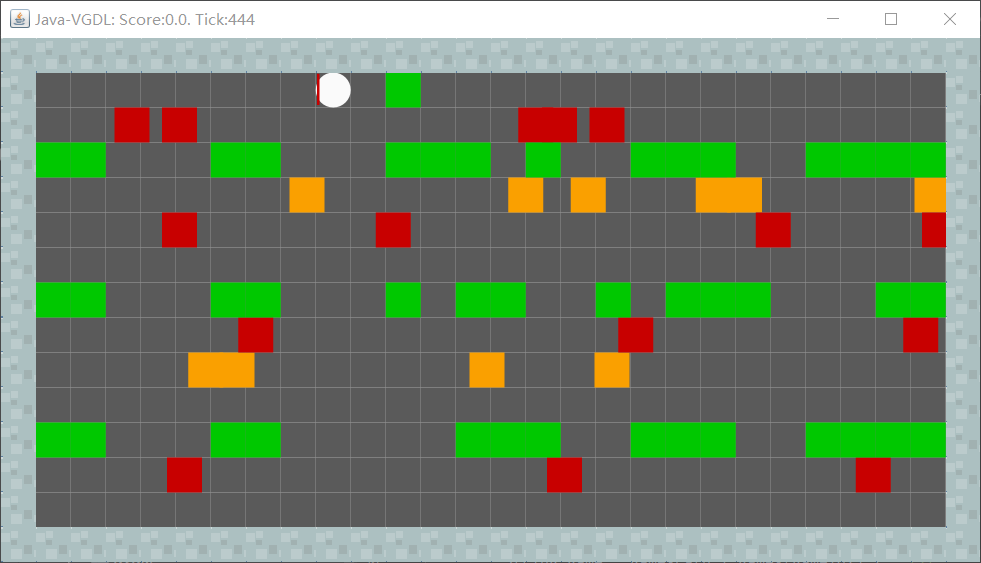
Vector2d minMoving;  
 double movingDist=9999999;

Vector2d protalPos=null;  
 double protalDist=0;

for(ArrayList<Observation> l:immovingObj){  
 if(l.size()>0){  
 for(Observation obs:l) {  
 if(immovingDist>avatarPos.dist(obs.position)) {  
 minImmoving = obs.position;  
 immovingDist=avatarPos.dist(obs.position);  
 }  
 }  
 }  
 }  
   
 for(ArrayList<Observation> l:movingObj){  
 if(l.size()>0){  
 for(Observation obs:l){  
 if(movingDist>avatarPos.dist(obs.position)) {  
 minMoving = obs.position;  
 movingDist=avatarPos.dist(obs.position);  
 }  
 }  
 }  
 }  
   
 if(protal!=null){  
 protalPos=protal[0].get(0).position;  
 protalDist=avatarPos.dist(protalPos);  
 }  
 if(movingDist==0)  
 score= stateObs.getAvatarHealthPoints()+stateObs.getGameScore()+won-protalDist\*100-999 +immovingDist;  
 else  
 score= stateObs.getAvatarHealthPoints()+stateObs.getGameScore()+won-protalDist\*100 +movingDist\*10+immovingDist;

return score;  
}

修改了m\_gamma与m\_epsilon，发现将m\_gamma调小一些(0.8~0.9)能稍微提高性能，Avatar表现更优，更能冲上去。考虑到为了提高游戏分数，我提高了模拟采样的最大深度SIMULATION\_DEPTH(30~50)与m\_dataset的大小m\_maxPoolSize(1500~2500)。并且，为了有利于训练我将游戏文件稍加修改，将Avatar的生命值改成了100。在游戏中后期400~700tick的时候，Avatar终于能越过最后一层障碍：



做了大量重复实验后，平均每一轮Avatar能到达一次Protal，与刚开始相比可以说是有很大的改进了，从而可以认识到特征提取和启发式函数设计的重要性(没改启发式函数的时候最高Avatar只能到达最后一层障碍下面)。但是，Avatar表现得并不是特别优秀，它训练需要较长的时间，有时候也来不及躲避或者碰到了移动的物体而掉下来，到达Protal有时候是需要一些运气。

总的来说，在这次实验中，使用QPolicy方法玩Freeway游戏，修改了特征提取方法和强化学习参数，重写了状态的启发式函数，一定程度上提高了Agent的性能。

附中文参考文献:

1. 博客,<https://blog.csdn.net/Gin077/article/details/82987599>