人工智能导论:作业四实验报告

薛正海(181220063、xuezh@smail.nju.edu.cn)

(南京大学 人工智能学院, 南京 210093)

1 代码阅读

代码利用强化学习中的 Q-learning 进行学习和决策,涉及到的代码在 Agent 的 act 函数中.总体而言,代码通过用一个分类模型,建立从(s,a)对到 Q 值的映射 f,以此作为 Q 函数进行决策.

具体而言,act 函数首先利用 learnPolicy 函数更新 Q 函数.在 learnPolicy 函数的内部,代码进行了“采样+学习”的过程.其中采样过程为 simulate 函数.该函数根据当前 Q 值确定策略,在环境中一步一步采样,并将 state(提取出的环境特征),action(当前动作的编号)和 reward(前后两个状态启发函数的差值)保存在 sequence 数组中.注意我们希望建立从(s,a)到 Q 值的映射,但目前只记录了 reward 值,因此数据收集完成后还需要遍历一次,计算出 Q 值.按照其定义,我们只需要计算从当前步起的折扣累计回报即可.这样我们就拿到了特征(s,a)和映射的目标 Q 值,完成采样.学习过程为 m_policy.fitQ 函数,利用 REPTree 模型搭建起一个分类器,拟合出希望得到的映射 f.得到更新过的 Q 函数后,模型贪心地选择当前状态下 Q 值最大的动作进行返回,从而完成一步 act.

2 问题回答

2.1

模型中的 Q 函数用一个基于决策树的分类器表示,分类器的输出(类别)就是当前 state 的 Q 值.这样表示的缺点在于利用离散属性来刻画一个连续的函数,不可避免地带来较大误差.改进方法是将 Q 当做连续的变量,用一个回归算法来拟合之.考虑到输入是较低维的像素信息,决策树其实不能很好处理,可将特征提取器换成卷积神经网络.另外,数据收集的过程完全采用蒙特卡洛采样,方差较大,对环境中出现的随机性可能不能完全建模,可利用 Bootstrap 方法减小方差.

2.2

SIMULATION_DEPTH 控制一次蒙特卡洛采样中智能体执行动作的最大 timestep,m_gamma 是累计奖赏函数的折扣率,m_maxPoolSize 限制了 dataset,即训练数据的容量.

2.3

getAction 比 getActionNoExplore 多了 epsilon-greedy,即有 epsilon(0<epsilon<1)的概率根据 Q 函数贪心选择动作,有 1-epsilon 的概率随机选择动作,用于采样.而 getActionNoExplore 用于决策.

3 修改特征提取方法

观察游戏进行的过程,我发现:不同 GameObject 的移动方式不同,Agent 应该对其做出不同的反应.但原有的特征提取方法只反映出当前帧的静态信息,而无法捕捉帧与帧之间的时序信息.因此,我借鉴 LSTM 的思想,将过去几帧的状态特征加入到当前帧的状态特征中(具体的合并操作是直接堆叠).具体代码如下:

```
if (has_init==0)
{
    has_init = 1;
    for(int k=0; k<num_timestep; k++)
        for(int y=0; y<num_col; y++)
            for(int x=0; x<num_row; x++)
                feature[k*num_col*num_row+y*num_row+x] = map[x][y];
}
else
{
    for(int k=0; k<num_timestep-1; k++)
        for(int y=0; y<num_col; y++)
            for(int x=0; x<num_row; x++)
                feature[(k+1)*num_col*num_row+y*num_row+x] = feature[k*num_col*num_row+y*num_row+x];
    for(int y=0; y<num_col; y++)
        for(int x=0; x<num_row; x++)
            feature[(num_timestep-1)*num_col*num_row+y*num_row+x] = map[x][y];
}
      // 4 states
feature[num_row*num_col*num_timestep] = obs.getGameTick();
feature[num_row*num_col*num_timestep+1] = obs.getAvatarSpeed();
feature[num_row*num_col*num_timestep+2] = obs.getAvatarHealthPoints();
feature[num_row*num_col*num_timestep+3] = obs.getAvatarType();
```

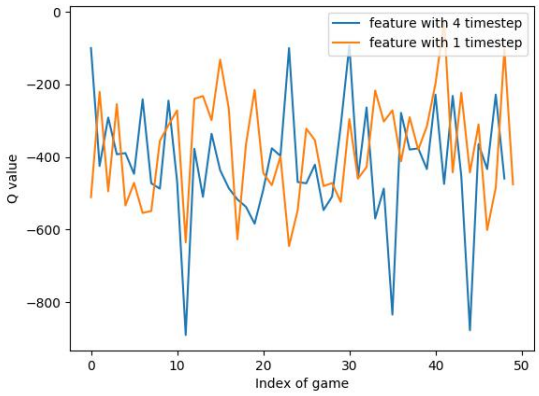
注意游戏画面的行数,列数等信息已用变量表示.代码利用 has_init 变量指示是否对特征数组初始化.若是,则将特征数组中的信息整体向“更旧”移动一帧,并用当前 map 信息填充最新的一帧.若否,则将所有帧的特征初始化为当前 map 的信息.在 datasetHeader 中修改的代码如下:

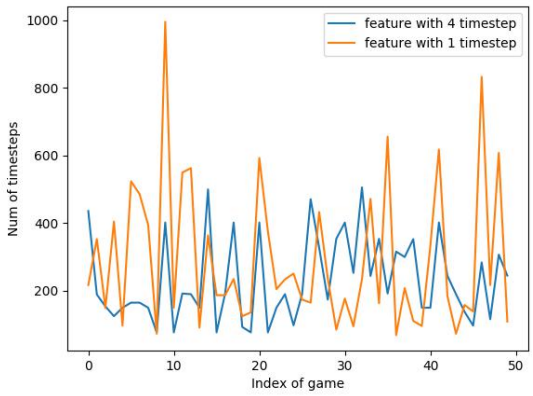
```
for (int k = 0; k<4; k++){
    for(int y=0; y<31; y++){
        for(int x=0; x<28; x++){
            Attribute att = new Attribute("timestep" + k + "object_at_position_x=" + x + "_y=" + y);
            attInfo.addElement(att);
        }
    }
}
```

其多加了一重循环,加入了每组特征的时间戳信息.

```
for (depth = depth - 1; depth >= 0; depth--) {
    accQ += sequence[depth].classValue();
    sequence[depth].setClassValue(accQ);
    // System.out.println(accQ);
    data.add(sequence[depth]);
}
nr++;
if (nr>100)
{
    System.out.println(accQ);
    nr=0;
}
```

在实验对比中,我将重复进行游戏的次数设置为 50,并用左图中的代码输出学习过程中的 Q 值.实验具体对比了只包含当前帧信息的特征提取和包含前 4 帧信息的特征提取方法的效果,simulation_depth 为 20.实验结果如下图所示.

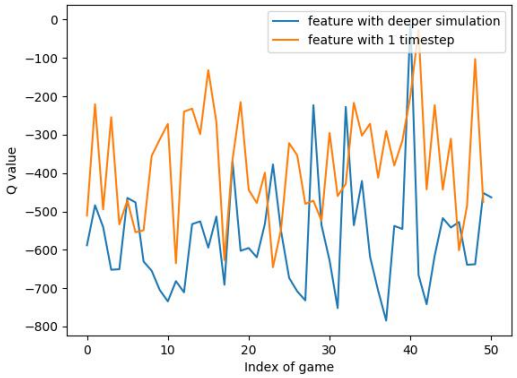


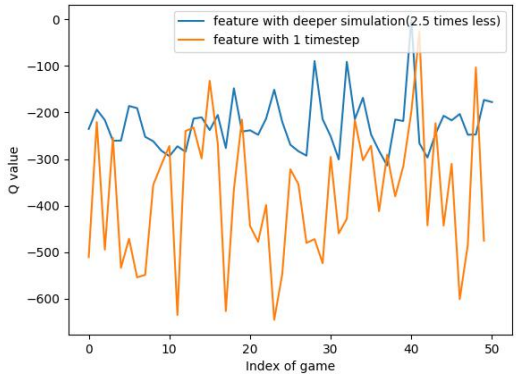


可见经过改进后,智能体的总体表现和学习中表现均未明显提升.且改进后,每次游戏的持续时间(total timesteps)有所降低,说明智能体没能学到避免游戏过早结束的方法.而总体性能不佳可能是因为学习样本量不够,每次模拟的深度(simulation_depth)不够,也可能是目前提取出的特征都过于底层,需要进一步实验来进行判断.

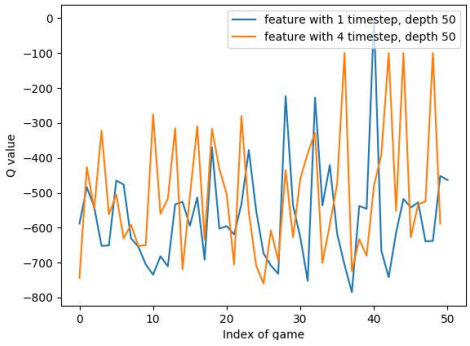
4 调整强化学习参数

可供调整的参数有:reward 函数的种类,simulation_depth,gamma,epsilon-贪心中的 epsilon,充当 Q 函数的分类器的种类等.首先,直观来看,simulation_depth 与智能体性能成正相关.实验结果如下:



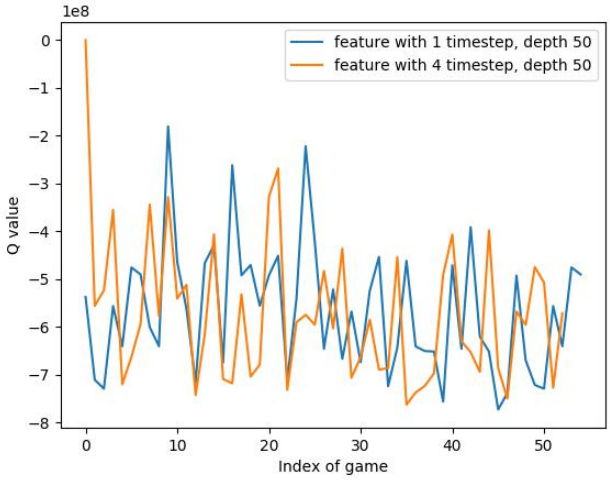


左图为不同 simulation depth 的模拟过程的 Q 值的直接对比. 考虑到 Q 值是对整个轨迹的回报累计, 若要对不同长度的轨迹的 Q 值相对比, 应消除轨迹不同长度的影响, 如右图所示. 可见加大 simulation_depth 确实提升了智能体的性能. 下图是 simulation_depth 为 50, 提取的特征中包含不同数量的 timestep 训练出的 Qvalue 的对比.

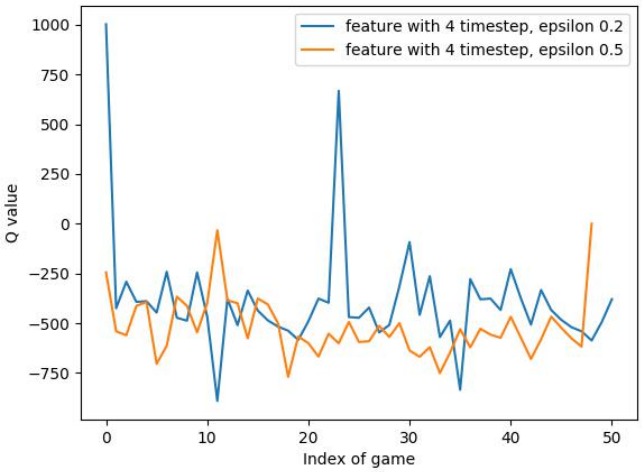


可见 simulation_depth 较大时,包含 4 个 timesteps 特征的 Agnet 比只有当前 timestep 特征的 Agent 能产生更好 performance,尤其在训练到 40 轮之后.

其次,原算法选择的 reward 函数为 WinScoreHeuristic, 我将其更换为 SimpleStateHeuristic 函数,并进行了相应实验.结果如下:



可见新的 reward 函数并没有让 Agent 的性能显著提升,两种不同的特征提取方法也没有明显区别.下面比较 epsilon-greedy 中 epsilon 值不同时,Agent 的性能:



可见 epsilon 增大时,智能体的性能有所下降,原因在于其动作随机性较大,没有充分利用学习到的信息.

总体而言,无论特征提取与超参数的选择如何,经过训练后,智能体的性能都没有明显提升.可能的原因有:每次蒙特卡洛模拟时,方差较大,导致训练信息不稳定,难以学到有用信息;特征过于底层,模型较难建立从特征到动作的映射;直接用分类模型当做 Q 函数,建模较为简单.

以上就是我本次人工智能导论报告的全部内容,感谢您的阅读!