实验3. 强化学习实践

MG1733092，张则君， 826320663@qq.com

2018年1月3日

综述

强化学习是一种交互式的学习，由环境、机器组成，在这个过程中涉及状态、动作、奖赏三个变量。与监督学习不同的是，即使状态对应样本，动作对应标记，但这里的标记在强化学习中是未知的，它所认为标记的好坏在某种程度上说，是跟奖赏成正比的。因此奖赏的多少对实验会有很大影响。换句话说，在某种程度上，强化学习还需要利用一些先验知识，因为环境的反馈奖赏也是需要根据环境的状态来进行量化。这里标记的好坏定义为Q值，而对Q值的学习又有不同的算法，实验这里采用Q-learning、DQN,DOUBLE DQN三种算法对三个环境CartPole-v0，MountainCar-v0，Acrobot-v1进行强化学习。算法的不同主要体现在对Q值的更新上，Q-learning直接迭代求得表格，其他2个算法利用了神经网络来求得Q的函数，即间接求Q。但从本质上说，三种算法共同依赖一个公式，这个公式是强化学习的核心，但三种算法实际也应该考虑到奖赏问题，其次而且在Q值的更新上所依赖的大量参数应该有所改变。这些在实验过程中，貌似陷入一个调参怪圈，从中意识到奖赏起到的作用和对公式的改变或者应该对核心公式进行改变，单纯转变求更新Q的方法实际并没有抓住本质。在实验中，其中对于CartPole-v0均达到了最好效果20000步，在MountainCar-v0， Acrobot-v1中Qlearning效果还可以，在DQN,DOUBLE DQN中出现的现象有些奇怪，出现最差情况是常态，随着episodes的增加，步数不会有明显减少，在实验中还需多次运行以找到一个好的状态初始点，并在出现好的结果提前终止。

实验二.

状态离散化

对状态区间均匀分割来离散状态。具体算法看Algorithm1。

Q-learning算法实现

Q-learning算法在训练时首先指定选择动作的策略这里采用ϵ贪心策略，之后利用公式1进行更新。

$$Q(x,a) = Q(x,a) + \alpha(reward + \gamma(Q(x^{'}, a^{'} - Q(x,a)))) \tag{1}$$

Algorithm 1 状态离散化

Input:

环境的某时刻状态,s 离散的状态数向量,num 状态空间,S;

Procedure:

离散的状态值,$index$

$index = 0$
for each $i \in s$ **do**
  按$num[i]$均匀划分$S[i]$得到区间bins
  将i映射$bins$的子区间位置pos，区间外的位置为0
  if i不是第一个状态 **then**
    $b = b * num[i]x$
  end if
  $index = index + pos * num[i]$
end for
return $index$;

Algorithm 2 Q-learning算法

Input:

环境:Env; 探索率:ϵ; 探索衰减率:ϵ_decay; 折扣因子:γ; 学习率:α;

Procedure:

将Q初始化为全0
for each $i \in episode$ **do**
  Env初始化状态 s
  s利用算法algorithm1离散化为s_index
  for each $t \in time$ **do**
    Env根据 $\pi^{\epsilon}(s_index)$选择动作a
    Env执行a返回奖赏$reward$下一状态$next_s$
    下一个新动作next_a=argmax(Q(s_index))
    s_index利用算法algorithm1离散化为$next_s_index$
    更新$Q(s_index, a) += \alpha$(reward+$\gamma Q(next\_s\_index, next\_a) - Q(s\_index, a)$)
    $s\_index = next\_s\_index$
  end for
end for
return Q;

超参设置

在Qlearning的超参的设置中，作业的说明说三个算法超参要保持一致，但这里考虑到Q-learnig与dqn的区别，如果对Q-learning设置成与dqn一样的参数，对Q-learning的算法不太公平，因此这里调参设置没有刻意设置的与其他两个算法不一样，而是设置不同的超参以达到其Qlearning的最好效果。其中需要说明cartpole的速度设置上下界对结果也会有很大影响（环境中速度区间是很大的），而且在任务失败时设置惩罚也是必要的，因为在角度超过15_{Θ}时，它当前状态的采取的动作有很大可能是错误的，因此惩罚这样的状态。其余的2个环境并没有做惩罚和相应的状态界设定。

表 1: 实验二超参设置

	eposides	max_timesteps	学习率	探索率	探索衰减率	折扣率	备注
CartPole-v0	2000	20000	0.1	1.0	0.995	0.95	状态数离散化为(8,8,8,8)，其中设置其状态上下界(-1,1)
MountainCar-v0	10000	200	0.1	1.0	0.995	0.95	状态数离散化为(40,40)
Acrobot-v1	3000	200	0.01	0.5	0.995	0.95	状态数离散化为(4,4,4,4,10,10)

实验结果

CartPole-v0是进行了50个episodes，Acrobot-v1，MountainCar-v0进行了100个episodes。

表 2: 实验二Q-learning的测试结果

	reward均值	reward标准差
CartPole-v0	20000	0.0
MountainCar-v0	166.48	21.6
Acrobot-v1	246.5	60.1

实验三.

DQN与Q-learning的算法本身思想一样，都要找到最佳的Q，但Q-learning是利用迭代直接求解的是一个固定的二维矩阵。而DQN利用神经网络不直接求Q，而是找一种函数，不断求解函数里的参数，使得Q不断收敛来间接达到目标。

其中在实验过程中对CartPole-v0设置reward对结果是有影响的，小车的杆的速度越小越是有利状态，小车的位置越处于中心越安全，因此$reward = \frac{|max(pos)-nextstate(0)|}{max(pos)} + \frac{|max(v_pole)-next(state(0))|}{max(pos)}$。在MountainCar-v0上，当小车动量越大时，前一次状态的所处位置与当前状态的所处位置距离会越大，因此这里的$reward = |state(0)-(-0.5)|+|state(0)-nextstate(0)|$（state(0)是小车的位置）。在 Acrobot-v1中由于两个连杆达到指定高度时，所处状态可以是多样的，实际可以根据角度和杆长计算出距离制定高度的位置修改reward，但这样有些麻烦，因此reward并未修改。

由于实验对MountainCar-v0， Acrobot-v1发现经常出现1999步因此采用了训练提前停止。当reward均值小于200时即不再训练。

超参设置

表 3: 实验三DQN超参设置

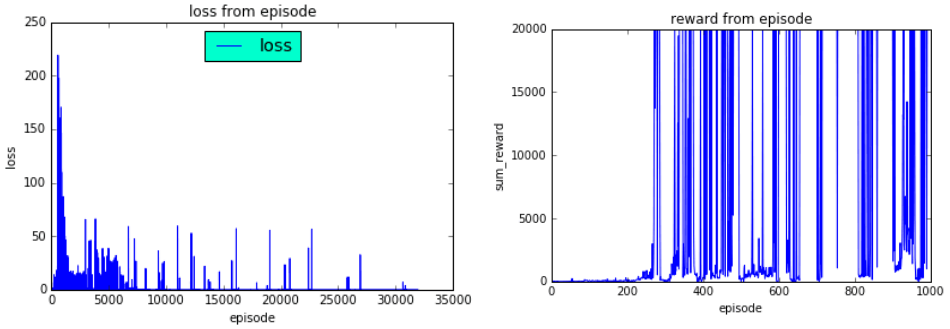
	M	T	网络层数	batchsize	学习率	折扣率	备注
CartPole-v0	1000	20000	2层全连接层每层24个神经元	32	0.001	0.95	
MountainCar-v0	2000	2000	2层全连接层每层24个神经元	32	0.001	0.95	在1010个episodes终止
Acrobot-v1	3000	2000	2层全连接层每层24个神经元	32	0.001	0.95	在430个episodes

DQN训练结果

实验对MountainCar-v0， Acrobot-v1由于实验发现经常出现1999步因此采用了训练提前停止。当reward均值小于200时即不再训练。

CartPole-v0训练了1000个episodes，MountainCar-v0在1010个episodes终止， Acrobot-v1在430个episodes终止

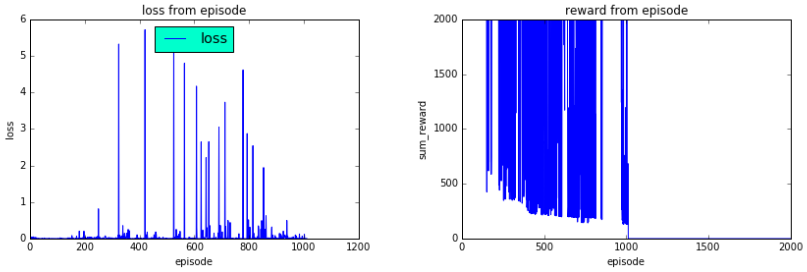
CartPole-v0



(a) cartpolev0-trainloss图 (b) cartpolev0-trainreward图

图 1: DQN-CartPole-v0 训练结果

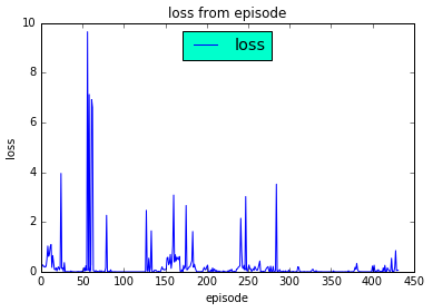
MountainCar-v0



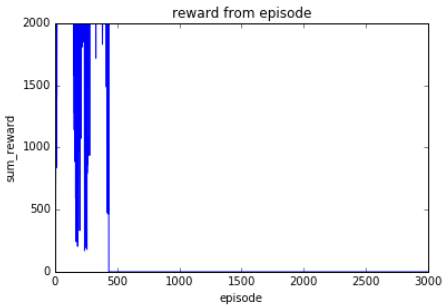
(a) MountainCar-v0-trainloss图 (b) MountainCar-v0-trainreward图

图 2: DQN-MountainCar-v0 训练结果

Acrobot-v1



(a) Acrobot-v1-trainloss图



(b) Acrobot-v1-trainreward图

图 3: DQN-Acrobot-v1 训练结果

DQN测试结果

CartPole-v0是进行了50个episodes，Acrobot-v1，MountainCar-v0进行了100个episodes。

表 4: 实验三DQN的测试结果

	reward均值	reward标准差
CartPole-v0	20000	0.0
MountainCar-v0	162.2	42.74
Acrobot-v1	209.46	370.05

实验四.

DQN在任务环境中每探索一步都要更新网络权重,，这样会造成过高估计action的Q值，尤其是过高估计带来偏差造成错过最优的action的Q值，因此改进的DQN通过每次训练时保存当前的网络权重，再下次训练时，仍利用保存的上次网络权重计算新动作的Q值。

Double DQN超参设置

实验四的超参设置与实验三中除了MountainCar-v0其余参数设置与实验三一致。具体解释看训练结果。这里单独列出MountainCar-v0的实验参数。

表 5: 实验四DQN-MountainCar-v0超参设置

	M	T	网络层数	batchsize	学习率	折扣率	备注
MountainCar-v0	2000	2000	2层全连接层每层24个神经元	32	0.1	0.95	在680个episodes终止

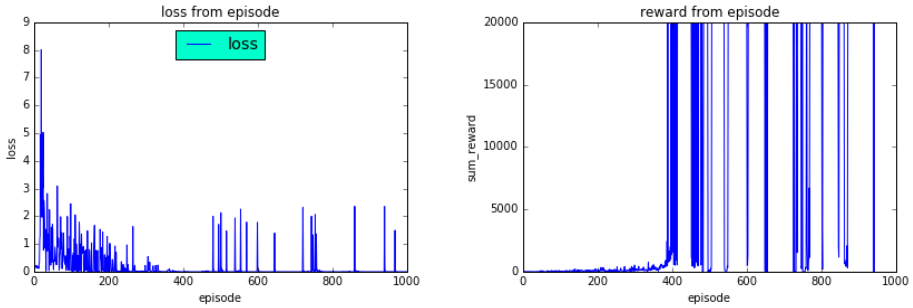
Double DQN训练结果

实验四，如同实验三对MountainCar-v0， Acrobot-v1由于实验发现经常出现1999步因此采用了训练提前停止。当reward均值小于200时即不再训练。

CartPole-v0训练了1000个episodes；MountainCar-v0之所以没有参数与实验三中的DQN设置一致是因为即使运行多次并且采用提前终止也无法解决一直出现1999步的问题，因此探索率最小设置为了0.2，学习率也进行提升到了0.1，最终在680个episodes终止； Acrobot-v1在90个episodes终止。

CartPole-v0

CartPole-v0的实验结果见图4

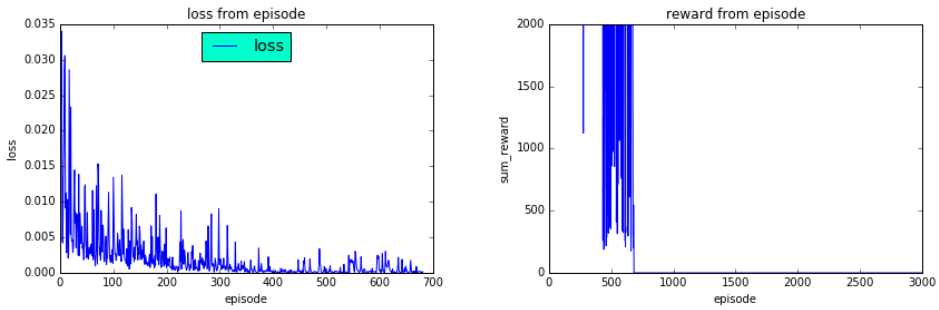


(a) cartpolev0-trainloss图　　(b) Size of disk files scanned

图 4: DDQN-CartPole-v0训练结果

MountainCar-v0

MountainCar-v0的实验结果见图5

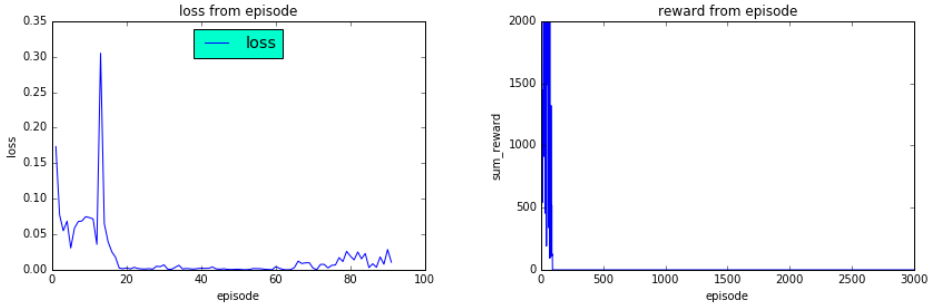


(a) MountainCar-v0-trainloss图　　(b) Size of disk files scanned

图 5: DDQN-MountainCar-v0 训练结果

Acrobot-v1

Acrobot-v1的实验结果见图6



(a) Acrobot-v1-trainloss图　　(b) Size of disk files scanned

图 6: DDQN-Acrobot-v1 训练结果

Double DQN测试结果

CartPole-v0是进行了50个episodes，Acrobot-v1，MountainCar-v0进行了100个episodes。

表 6: 实验四Double DQN测试结果

	reward均值	reward标准差
CartPole-v0	20000	0.0
MountainCar-v0	185.99	22.96
Acrobot-v1	93.62	12.4649749298