强化学习：作业三

李龙飞 DZ2037003

2020年12月4日

1 作业内容

在gym Atari环境中实现DQN, Double DQN和Dueling DQN算法。

2 实现过程

2.1 DQN

上次作业我们实现了Q-Learning算法，Q-Learning算法在小规模问题上表现得很好，但难以应用于大规模问题，原因在于Q-Learning算法依赖于Q值表的构建，但在大规模问题中，无法在内存中维护如此大的Q值表。

由于问题的状态集合规模大，一个可行的建模方式是使用价值函数来近似表示，而深度神经网络往往拥有很强的表示能力，因此DQN算法使用深度神经网络来近似拟合Q值表。神经网络输入为状态s，输出为大小为$\mathcal{A}$的向量，每一维代表对应状态动作对的Q值$Q(s,a)$, 令θ表示神经网络权重，值函数可以表示为$Q(s,a,\theta)$。

DQN中有两个非常重要的思想，经验回放和使用目标网络：经验回放是指将探索环境得到的数据储存起来，然后随机采样样本更新深度神经网络的参数。使用目标网络是指使用一个MainNet产生当前Q值，使用另外一个Target产生Target Q，每次反向传播只更新MainNet的参数，每经过一定次数的迭代，再将MainNet的参数复制给TargetNet。其作用如下：

- 经验回放：
  1. 神经网络要求数据满足独立同分布假设，而采样得到的数据是有相关性的，经验回放通过存储-采样的方法可以打破关联性。
  2. 数据利用率高，因为一个样本被多次使用。
- 使用目标网络：引入TargetNet，一段时间内目标Q值保持不变，一定程度降低了当前Q值和目标Q值的相关性，提高了算法稳定性。

Algorithm 1 QDN with experience replay

初始化经验集合D
随机初始化当前网络Q的所有参数θ,初始化目标网络$\hat{Q}$的参数$\theta^{-}=\theta$
for episode = 1, . . . , M **do**
  初始化状态s_1, 得到其特征向量$\phi_1=\phi(s_1)$
  for t = 1, . . . T **do**
    以概率ϵ随机选择一个动作a_t,否则选择$a_t=\arg\max_a Q(\phi(s_t),a;\theta)$
    执行动作a_t,得到新状态s_{t+1}对应特征向量$\phi(s_{t+1})$，得到奖励r_t
    $s_{t+1}=s_t$
    将四元组$(\phi_t,a_t,r_t,\phi_{t+1})$ 放入经验集合D
    从经验集合D中采样一批样本$(\phi_t,a_t,r_t,\phi_{t+1})$
    令

$$y_i=\begin{cases} r_j, & \text{如果游戏在}j+1\text{时刻结束} \\ r_j+\gamma\max\limits_{a'}\hat{Q}(\phi_{j+1},a';\theta^{-}) & \text{其他}\end{cases}$$

    损失函数为$(y_i-Q(\phi_j,a_j;\theta))^2$，使用梯度下降更新$Q$网络参数$\theta$
    每隔C步令$\hat{Q}=Q$
  end for
end for

DQN的更新方式和Q-Learning相同，DQN的损失函数如下：

$$y_i=\begin{cases} r_j, & \text{如果游戏在}j+1\text{时刻结束} \\ r_j+\gamma\max\limits_{a'}\hat{Q}(\phi_{j+1},a';\theta^{-}) & \text{其他}\end{cases} \tag{1}$$

$$L(\theta)=(y_i-Q(\phi_j,a_j;\theta))^2 \tag{2}$$

2.2 Double DQN(DDQN)

虽然DQN算法用了两个Q网络并使用目标Q网络计算Q值，样本的目标Q值的计算仍是贪婪法得到的,使用max虽然可以快速让Q值向可能的优化目标靠拢，但是容易导致过度估计(Over Estimation)，即对于真实的策略来说并在给定的状态下并不是每次都选择使得Q值最大的动作，因为一般真实的策略都是随机性策略，所以目标值直接选择动作最大的Q值往往会导致目标值要高于真实值。为了解决这个问题， Double DQN通过解耦目标Q值动作的选择和目标Q值的计算这两步，来达到消除过度估计的问题。

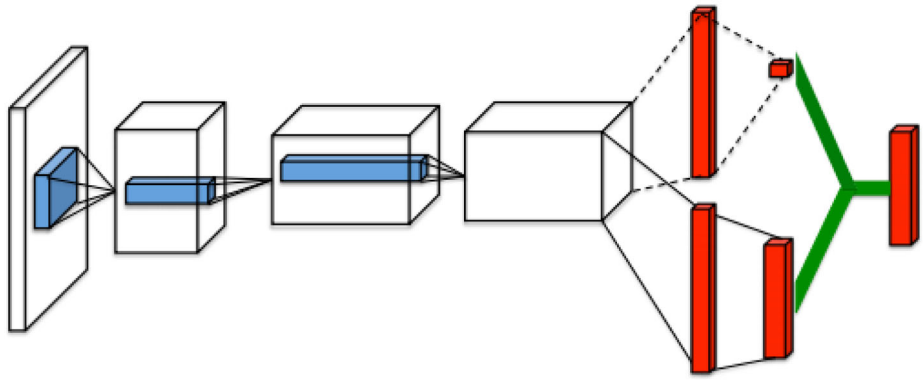

Figure 1: Dueling DQN

DDQN不再是直接在目标Q网络里面找各个动作中最大Q值，而是先在当前Q网络中先找出最大Q值对应的动作，即

$$a^{\max}(\phi_{j+1},\theta) = \arg\max_{a'} Q(\phi_{j+1},a',\theta) \tag{3}$$

然后利用这个选择出来的动作$a^{\max}(\phi_{j+1},\theta)$根据目标网络计算目标Q值。即：

$$y_i = r_j + \gamma\hat{Q}(\phi_{j+1},a^{\max}(\phi_{j+1},\theta);\theta^-) \tag{4}$$

综合起来即得到DDQN算法损失函数：

$$y_i = \begin{cases} r_j, & \text{如果游戏在}j+1\text{时刻结束} \\ r_j + \gamma\hat{Q}(\phi_{j+1},\arg\max_{a'} Q(\phi_{j+1},a',\theta);\theta^-) & \text{其他} \end{cases} \tag{5}$$

除了目标Q值的计算方式以外，DDQN算法和DQN的算法流程完全相同。

2.3 Dueling DQN

Dueling DQN尝试通过优化神经网络的结构来优化算法。具体来说，Dueling DQN考虑将Q网络分成两部分，第一部分是仅仅与状态S有关，与具体要采用的动作A无关，这部分称为价值函数部分，记做$V(s,\theta,\alpha)$,第二部分同时与状态状态S和动作A有关，这部分叫做优势函数(Advantage Function)部分,记为$A(s,a,\theta,\beta)$,那么最终我们的价值函数可以重新表示为：

$$Q(s,a,\theta,\alpha,\beta) = V(s,\theta,\alpha) + A(s,a,\theta,\beta) \tag{6}$$

其中，θ是公共部分的网络参数，α是价值函数独有部分的网络参数，β是价值函数独有部分的网络参数，如图1所示。

Dueling DQN原则上可以和上面任意一个DQN算法结合，只需要将Dueling DQN的模型结构去替换上面任意一个DQN网络的模型结构,但一般在使用中会对优势函数A做中心化处理:

$$Q(s,a,\theta,\alpha,\beta) = V(s,\theta,\alpha) + A(s,a,\theta,\beta) - \frac{1}{\mathcal{A}}\sum_{a' \in \mathcal{A}} A(s,a',\theta,\beta) \qquad (7)$$

3 复现方式

1. 修改配置文件`atari_ddqn.py`
    - 复现DQN:令`config.DQN = True, config.Double_DQN = False, config.Dueling_DQN = False`
    - 复现Double DQN:令`config.DQN = False, config.Double_DQN = True, config.Dueling_DQN = False`
    - 复现Dueling DQN:令`config.DQN = True, config.Double_DQN = False, config.Dueling_DQN = True`
    - 复现Dueling Double DQN:令`config.DQN = False, config.Double_DQN = True, config.Dueling_DQN = True`
2. 在主文件夹下运行 `python atari_ddqn.py --train`

4 实验效果

算法累计奖励和样本训练量的关系如图2，3所示，可以看到不同算法的奖励均随着样本训练量增加而增加，但波动比较大，同时可以发现:

- DQN比Double DQN能够更快的收敛，原因在于DQN中样本的目标Q值的计算是贪婪法得到的,使用max可以快速让Q值向可能的优化目标靠拢，本次实验中过度估计的影响体现得不明显。
- Dueling DQN相比DQN有更好的性能，原因在于在一些情形下，不同的状态动作对的值函数是不同的，但是在某些状态下，值函数的大小与动作无关，例如在该游戏中存在某些状态无论采取什么动作都会输掉游戏。

实验过程中我发现实验结果的随机性也比较大，即使是同样一组参数多次运行得到的结果差异依然会很大，甚至算法结果的优劣对比甚至会发生改变。

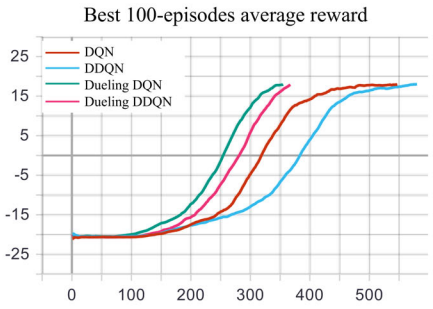

Figure 2: Best 100-episodes average reward

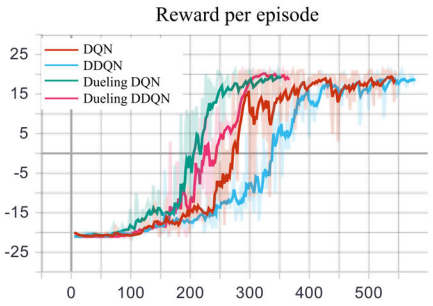

Figure 3: Reward per episode

5 小结

在这次实验中，我有以下几点发现:

- 与Q-Learning相比，DQN利用深度神经网络近似估计Q值，解决了Q-Learning中Q值表过大无法存储的问题，因此更适用于大规模问题。
- 在PONG环境中DQN比Double DQN能够更快的收敛，原因在于DQN中样本的目标Q值的计算是贪婪法得到的,使用max可以快速让Q值向可能的优化目标靠拢，本次实验中过度估计的影响体现得不明显，但在其他环境中Double DQN中可能会有更好的表现，因此不同的环境中不同的算法优劣性可能会发生改变，这也体现了机器学习中“No free Launch”原则。
- Dueling DQN相比DQN有更好的性能，原因在于在一些情形下，不同的状态动作对的值函数是不同的，但是在某些状态下，值函数的大小与动作无关。Dueling DQN的竞争结构能学到在没有动作的影响下环境状态的价值$V(s)$，动作的选择对$V(s)$的影响不大，而对$A(S)$的影响很大。
- 环境对算法影响很大，即使是同样一组参数多次运行得到的结果差异依然会很大。
- 算法学习率要合适，学习率过大会导致算法不收敛，过小会使得算法收敛很慢。