Addressing Function Approximation Error in Actor-Critic Methods-hoho

研究现状

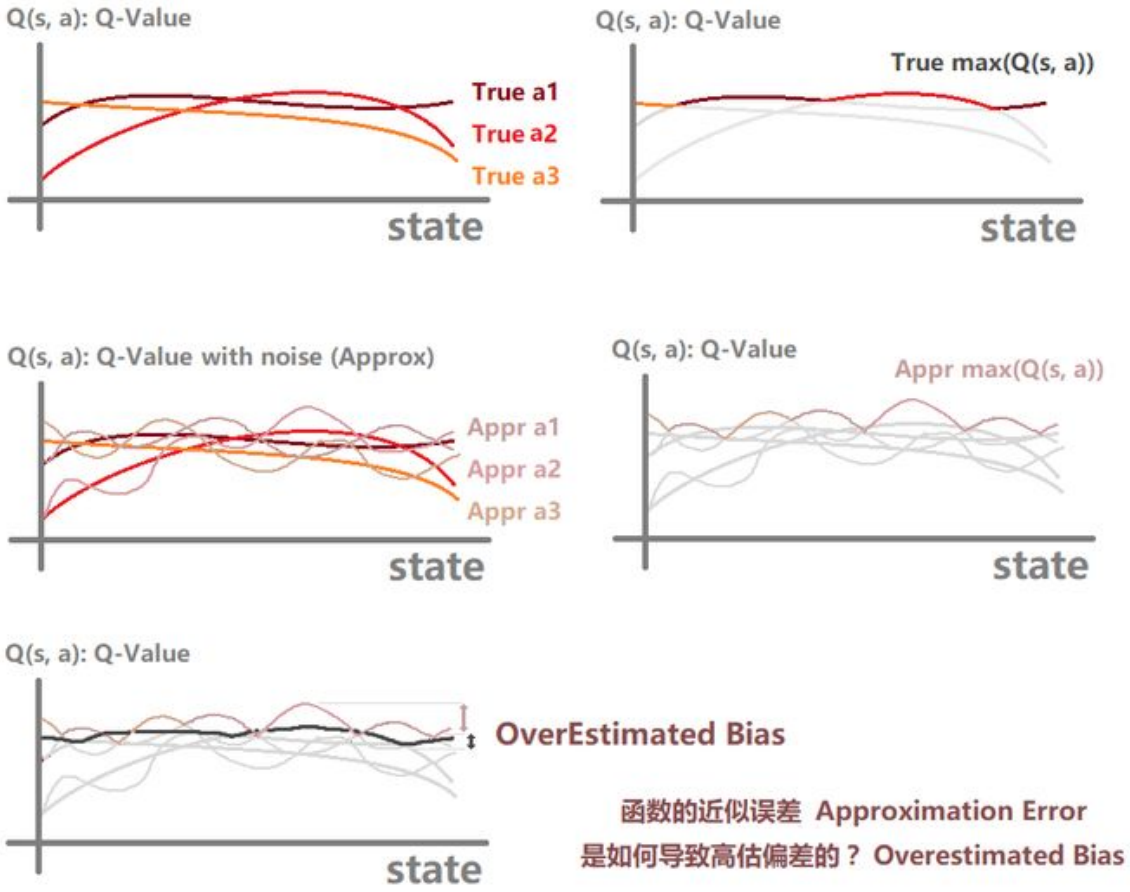
基于值函数的强化学习方法中，函数逼近不可避免的引入误差。使用这些不精确的估计，随时更新的迭代，这种误差就会积累的越来越多，最终导致Q值高估，只产生次优策略。

尽管使用目标网络是估值（如Double DQN），由于策略更新很慢，训练网络跟目标网络依然太过相似而不能避免大的偏差。

本文认为目标网络的技巧通过减少积累误差在减少方差方面至关重要，并提出了一种延迟更新策略的方案，直到策略估值收敛时才更新策略：Twin Delayed Deep Deterministic policy gradient alogirthm（TD3）

研究方法

1. 函数逼近的误差如何引起Q值高估：[1]



第1行，左边是实际状态下采取的连续动作a1,a2,a3的Q值，右边是综合3个动作进行最大化的结果（学习的目的就是最大化动作的Q值）

第2行，对动作Q值的预测，一般来说会逼近真实的动作曲线，右边同样采取最大化操作

第3行，可见估计的最大化与实际的最大化存在的偏差（实际会被高估）

2. 在actor-critic框架中“裁剪”双Q值学习

使用两个估值网络Q1与Q2使用相同的记忆同时进行估值，那么用$\min_{i=1,2} Q_{\theta_i}(s', \pi_{\phi_1}(s'))$选出那个高估的更少的作为估计值，可以缓解高估现象：

$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta_i}(s', \pi_{\phi_1}(s'))$$

尽管这样会导致低估偏差，但低估比高估要好，因为被低估的误差不会通过策略更新被传播。

另外一个好处是，通过把函数误差看作随机变量，最小化操作可以让误差的方差更小（hoho：大概是这样吧）。

3. 延迟进行策略更新直到值误差足够小

没有目标网络会导致学习的发散（更新不稳定）是由于策略更新伴随者估值的高方差。当策略不适合（the policy is poor）值会被高估而发散，而且由于估值的不准确又导致策略选的不适合。

所以如果使用目标网络（actor的target network）降低误差，并且在高误差、训练不稳定的情况下进行策略更新，那么策略网络（actor的policy network）更新频率应该要比价值网络（critic的value network）要低，在进行策略更新时要最小化误差。

本文提出的改进：在更新d轮critic之后，才去更新actor的策略网络与目标网络。为了保证TD误差足够小；使用软更新目标网络：$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$

4. 目标策略的平滑正则化

确定性策略梯度算法中一个问题：估值函数会过拟合。为此，本文模仿SARSA算法，相似的行为应该具有相似的价值，通过自举（bootstrapping）的方法对动作价值进行估计，使估值更加平滑：

$$y = r + \mathbb{E}_{\epsilon}[Q_{\theta'}(s', \pi_{\phi'}(s') + \epsilon)]$$

实际使用时，对于以上期望的计算可以在目标策略网络上添加一个小的噪声，并且平均化这个批次：

$$y = r + \gamma Q_{\theta'}(s', \pi_{\phi'}(s') + \epsilon),$$
$$\epsilon \sim clip(N(0, \sigma), -c, c)$$

最终TD3算法流程如图：

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

研究结论

使用TD3的曲线更加平滑：

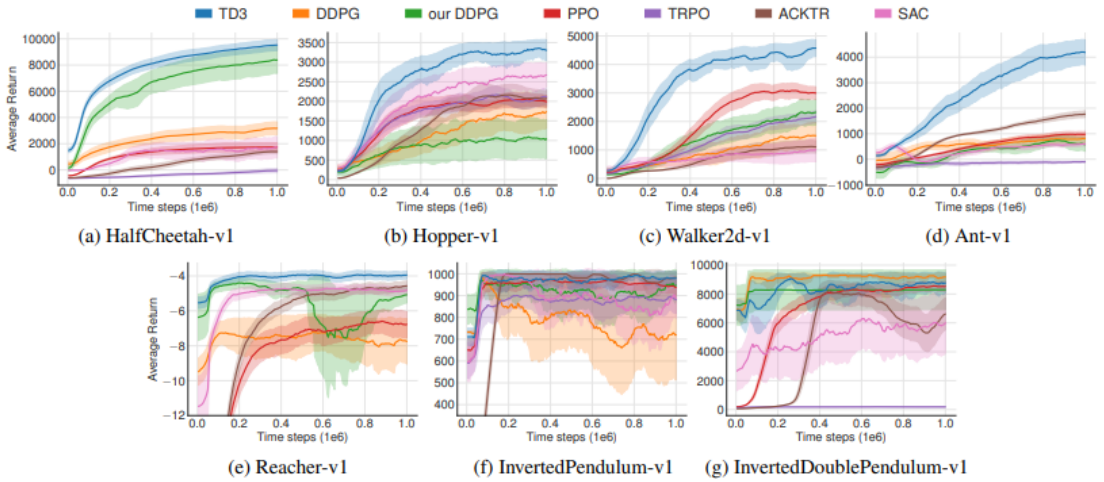


Figure 5. Learning curves for the OpenAI gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Curves are smoothed uniformly for visual clarity.

十轮，每轮1百万个时间步的最大平均回报，可见TD3收敛更快：

Table 1. Max Average Return over 10 trials of 1 million time steps. Maximum value for each task is bolded. ± corresponds to a single standard deviation over trials.

Environment	TD3	DDPG	Our DDPG	PPO	TRPO	ACKTR	SAC
HalfCheetah	**9636.95 ± 859.065**	3305.60	8577.29	1795.43	-15.57	1450.46	2347.19
Hopper	**3564.07 ± 114.74**	2020.46	1860.02	2164.70	2471.30	2428.39	2996.66
Walker2d	**4682.82 ± 539.64**	1843.85	3098.11	3317.69	2321.47	1216.70	1283.67
Ant	**4372.44 ± 1000.33**	1005.30	888.77	1083.20	-75.85	1821.94	655.35
Reacher	**-3.60 ± 0.56**	-6.51	**-4.01**	-6.18	-111.43	-4.26	-4.44
InvPendulum	**1000.00 ± 0.00**	**1000.00**	**1000.00**	**1000.00**	985.40	**1000.00**	**1000.00**
InvDoublePendulum	**9337.47 ± 14.96**	**9355.52**	8369.95	8977.94	205.85	9081.92	8487.15

附

- 代码：https://github.com/sfujim/TD3点补充：
- 知识点补充：

1. 强化学习中on-policy 与off-policy有什么区别？[1]

TD3基于DDPG，DDPG基于DQN，DQN基于Q-learning，而Q-learning就是异策略，里面的策略函数没有直接在环境中学习，而是隔着一个估值网络与间接地与环境进行交流：策略网络做出一个动作后，根据估值网络的评价对动作进行评估。

而PPO（近端策略优化）基于TRPO，TRPO（信任域策略优化）基于SARSA，SARSA就是同策略，它直接与环境进行交流：策略网络做出一个动作后，直接在环境中进行尝试，然后直接对自己进行优化。（进行动作选择的神经网络函数，与 根据Q值进行参数更新的神经网络 是同一个）

参考

[1] https://zhuanlan.zhihu.com/p/86297106?from_voters_page=true