强化学习—DDPG算法原理详解

2017-11-19 | DDPG |

一、 概述

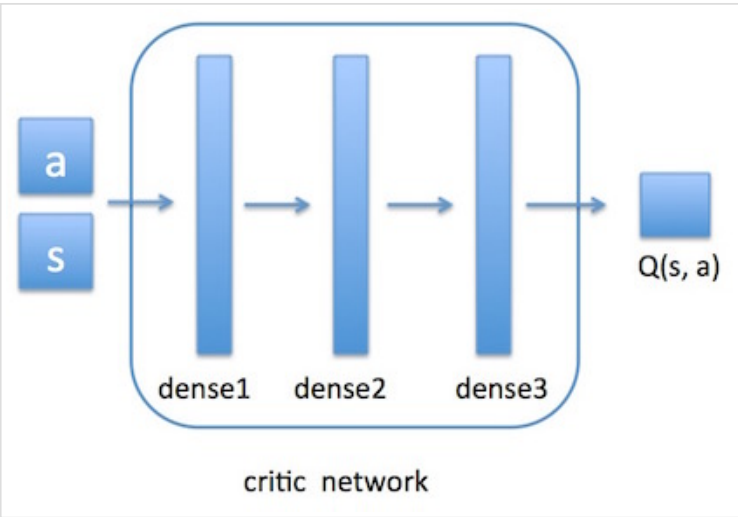
在DQN中有讲过，DQN是一种 model free（无环境模型）, off-policy（产生行为的策略和进行评估的策略不一样）的强化学习算法。DDPG (Deep Deterministic Policy Gradient)算法也是model free, off-policy的，且同样使用了深度神经网络用于函数近似。但与DQN不同的是，DQN只能解决离散且维度不高的action spaces的问题，这一点请回忆DQN的神经网络的输出。而DDPG可以解决**连续动作空间**问题。另外，DQN是value based方法，即只有一个值函数网络，而DDPG是actor-critic方法，即既有值函数网络(critic)，又有策略网络(actor)。

DDPG算法原文链接： DDPG

二、算法原理

在基本概念中有说过，强化学习是一个反复迭代的过程，每一次迭代要解决两个问题：给定一个策略求值函数，和根据值函数来更新策略。

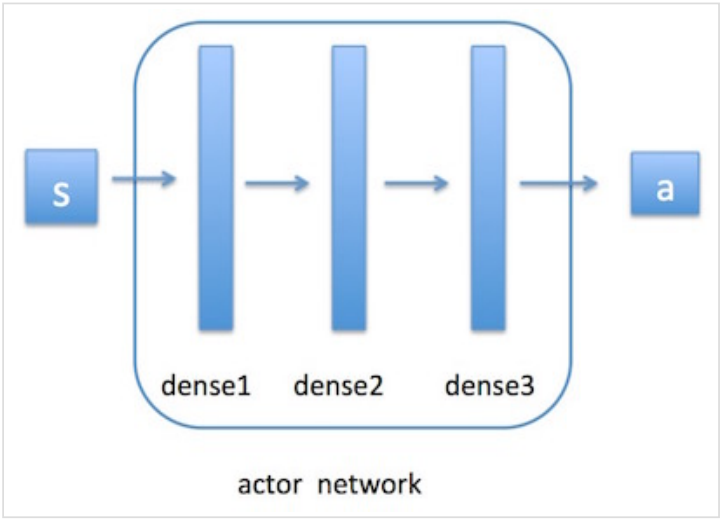
DDPG中使用一个神经网络来近似值函数，此值函数网络又称**critic网络**，它的输入是 action与observation $[a, s]$，输出是 $Q(s, a)$；另外使用一个神经网络来近似策略函数，此policy网络又称**actor网络**，它的输入是observation s，输出是action a.



critic–network

critic: $Q(s,a;\omega)$

target critic: $Q(s,a;\omega^-)$



actor–network

actor: $a=\pi(s;\theta)$

target actor: $a=\pi(s;\theta^-)$

这两个网络之间的联系是这样的：首先环境会给出一个obs，智能体根据**actor网络**（后面会讲到在此网络基础上增加噪声）做出决策action，环境收到此action后会给出一个奖励Rew，及新的obs。这个过程是一个step。此时我们要根据Rew去更新**critic网络**，然后沿**critic**建议的方向去更新**actor**网络。接着进入下一个step。如此循环下去，直到我们训练出了一个好的actor网络。

那么每次迭代如何更新这两个神经网络的参数呢?

与DQN一样，DDPG中也使用了target网络来保证参数的收敛。假设critic网络为$Q(s,a;\omega)$， 它对应的target critic网络为$Q(s,a;\omega^-)$。actor网络为$\pi(s;\theta)$，它对应的target actor网络为$\pi(s;\theta^-)$。

1、critic网络更新

critic网络用于值函数近似，更新方式与DQN中的类似。

$$target_t=R_{t+1}+\gamma Q(S_{t+1},\pi(S_{t+1};\theta^-);\omega^-)$$

$$Loss=\frac{1}{N}\sum_{t=1}^{N}(target_t-Q(S_t,a_t;\omega))^2$$

然后使用梯度下降法进行更新。注意，actor和critic都使用了target网络来计算target。

2、actor网络更新

actor网络用于参数化策略。这里涉及到强化学习中一个非常重要的概念：**策略梯度Policy Gradient**。

如何评价一个策略的好坏？首先我们要有一个目标，称为policy objective function，记为$J(\theta)$。我们希望求得θ使得$J(\theta)$取得最大值。$J(\theta)$对θ的导数 $\bigtriangledown_\theta J(\theta)$即为**策略梯度**。

策略梯度这一块可以分为四种情况分别讨论：stochastic on-policy, stochastic off-policy, deterministic on-policy 和 deterministic off-policy。David Silver的课程中详细的介绍了第一种。DPG论文的第二部分讲了第二种，第四部分讲了第三四种。由于DDPG中的策略是deterministic的，本文只介绍最后两种。

直观上来说，我们应该朝着使得值函数Q值增大的方向去更新策略的参数θ。记策略为 $a = \pi_\theta(s)$，
$J(\pi_\theta) = \int_s d^\pi(s)Q(s, \pi_\theta(s))ds = E_{s\sim d^\pi}[Q(s, \pi_\theta(s))]$，有以下定理：

Deterministic Policy Gradient Theorem:
$\bigtriangledown_\theta J(\pi_\theta) = \int_s d^\pi(s) \bigtriangledown_\theta \pi_\theta(s) \bigtriangledown_a Q^\pi(s,a)|_{a=\pi_\theta(s)} ds = E_{s\sim d^\pi}[\bigtriangledown_\theta \pi_\theta(s) \bigtriangledown_a Q^\pi(s,a)|_{a=\pi_\theta(s)}]$

确定性策略梯度定理提供了更新确定性策略的方法。将此方法用到Actor-Critic算法中：

(1) On-Policy Deterministic Actor-Critic

TD Error: $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, a_{t+1}; \omega) - Q(S_t, a_t; \omega)$
更新critic: $\Delta\omega = \alpha_\omega \cdot \delta_t \cdot \bigtriangledown_\omega Q(S_t, a_t; \omega)$ (SARSA)
更新actor: $\Delta\theta = \alpha_\theta \cdot \bigtriangledown_\theta \pi_\theta(S_t) \bigtriangledown_a Q(S_t, a_t; \omega)|_{a=\pi_\theta(s)}$

(2) Off-Policy Deterministic Actor-Critic

TD Error: $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, \pi_\theta(S_{t+1}); \omega) - Q(S_t, a_t; \omega)$
更新critic: $\Delta\omega = \alpha_\omega \cdot \delta_t \cdot \bigtriangledown_\omega Q(S_t, a_t; \omega)$ (Q-Learning)
更新actor: $\Delta\theta = \alpha_\theta \cdot \bigtriangledown_\theta \pi_\theta(S_t) \bigtriangledown_a Q(S_t, a_t; \omega)|_{a=\pi_\theta(s)}$

注意，在off-policy中，用于生成行为数据的策略和用于评估的策略不是同一个策略，也就是说，智能体实际上采取的action a_{t+1} 不是由π_θ生成的。假设它是由β生成的。在DDPG中，β策略是在π策略上增加了随机噪声random process，用来保证探索exploration。

理论上，这里引入了两种bias：一个是Deterministic Policy Gradient Theorem中的$Q^\pi(s,a)$，我们实际上用的是它的近似函数$Q(s,a;\omega)$；另一个是在off-policy中，行为策略与评估策略不同，理论上是需要引入importance sampling来进行修正的。实际上，这两个bias都通过满足了一个定理的条件来得以保证。详见Compatible Function Approximation.

三、算法整体流程

回顾DQN在Q-Learning基础上所做的改进：使用了深度神经网络做函数近似；使用经验回放；使用target网络。DDPG类似的也使用了深度神经网络，经验回放和target网络。不过DQN中的target更新是hard update，即每隔固定步数更新一次target网络，DDPG使用soft update，每一步都会更新target网络，只不过更新的幅度非常小。

附上原文的算法流程：

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

DDPG

参考资料：

David Silver的课程：www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html

强化学习

‹ 强化学习—DQN算法原理详解

更博计划 ›