浙江大学数学科学学院

前沿数学专题讨论 PR06

21 Dec 2022

Submitted To:
张南松老师
22-23 秋冬学期

Submitted By :
吴凡
农业工程 + 统计学

Contents

1 前言

今天介绍的主题为强化学习中的一种 actor critic 的提升方式 Deep Deterministic Policy Gradient (DDPG) 算法，DDPG 最大的优势就是能够在连续动作上更有效地学习。

概括来说，RL 要解决的问题是：让 agent 学习在一个环境中的如何行为动作 (act)，从而获得最大的奖励值总和 (total reward)。这个奖励值一般与 agent 定义的任务目标关联。agent 需要的主要学习内容：第一是行为策略 (action policy)，第二是规划 (planning)。其中，行为策略的学习目标是最优策略，也就是使用这样的策略，可以让 agent 在特定环境中的行为获得最大的奖励值，从而实现其任务目标。

行为 (action) 可以简单分为：

- 连续的：如赛车游戏中的方向盘角度、油门、刹车控制信号，机器人的关节伺服电机控制信号。
- 离散的：如围棋、贪吃蛇游戏。Alpha Go 就是一个典型的离散行为 agent。

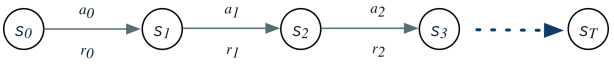
DDPG 是针对连续行为的策略学习方法。在 RL 领域，DDPG 主要从：PG -> DPG -> DDPG 发展而来。

相关基本概念：

1. s_t: 在 t 时刻，agent 观察到的环境状态，比如观察到的环境图像，agent 在环境中的位置、速度、机器人关节角度等；
2. a_t: 在 t 时刻，agent 选择的行为（action），通过环境执行后，环境状态由 s_t 转换为 s_{t+1}
3. $r(s_t, a_t)$ 函数：环境在状态 s_t 执行行为 a_t 后，返回的单步奖励值；
4. R_t: 是从当前状态直到将来某个状态，期间所有行为所获得奖励值的加权总和，即 discounted future reward

5. γ:discounted rate, 通常取 0.99.

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i) \tag{1}$$



1.1 PG

R.Sutton 在 2000 年提出的 Policy Gradient 方法，是 RL 中，学习连续的行为控制策略的经典方法，其提出的解决方案是：通过一个概率分布函数 $\pi_\theta(s_t|\theta^\pi)$ 来表示每一步的最优策略，在每一步根据该概率分布进行 action 采样，获得当前的最佳 action 取值; 即：

$$a_t \sim \pi_\theta(s_t|\theta^\pi) \tag{2}$$

生成 action 的过程，本质上是一个随机过程；最后学习到的策略，也是一个随机策略 (stochastic policy).

1.2 DPG

Deepmind 的 D.Silver 等在 2014 年提出 DPG：Deterministic Policy Gradient，即确定性的行为策略，每一步的行为通过函数 μ 直接获得确定的值：

$$a_t = \mu(s_t|\theta^\mu) \tag{3}$$

这个函数 μ 即最优行为策略，不再是一个需要采样的随机策略。

为何需要确定性的策略？简单来说，PG 方法有以下缺陷：

1. 即使通过 PG 学习得到了随机策略之后，在每一步行为时，我们还需要对得到的最优策略概率分布进行采样，才能获得 action 的具体值；而 action 通常是高维的向量，比如 25 维、50 维，在高维的 action 空间的频繁采样，无疑是很耗费计算能力的；

2. 在 PG 的学习过程中，每一步计算 policy gradient 都需要在整个 action space 进行积分:

$$\bigtriangledown_\theta = \int_S \int_A \rho(s)\pi_\theta(a|s)Q^\pi(s,a)dads \tag{4}$$

这个积分我们一般通过 Monte Carlo 采样来进行估算，需要在高维的 action 空间进行采样，耗费计算能力。

1.3 DDPG

Deepmind 在 2016 年提出 DDPG，全称是：Deep Deterministic Policy Gradient，是将深度学习神经网络融合进 DPG 的策略学习方法。相对于 DPG 的核心改进是：采用卷积神经网络作为策略函数 μ 和 Q 函数的模拟，即策略网络和 Q 网络；然后使用深度学习的方法来训练上述神经网络。

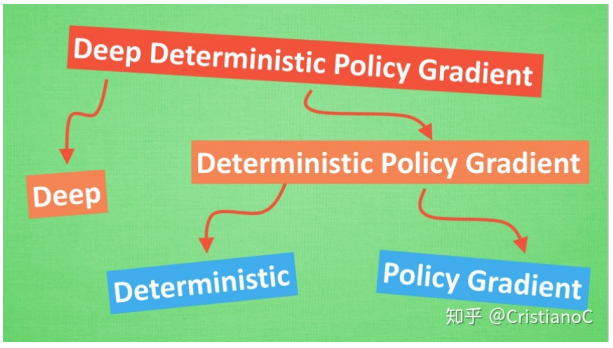
Q 函数的实现和训练方法，采用了 Deepmind 2015 年发表的 DQN 方法，即 Alpha Go 使用的 Q 函数方法。

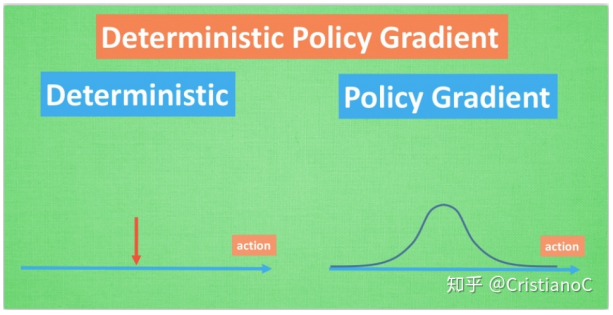
2 DDPG 算法

2.1 算法背景

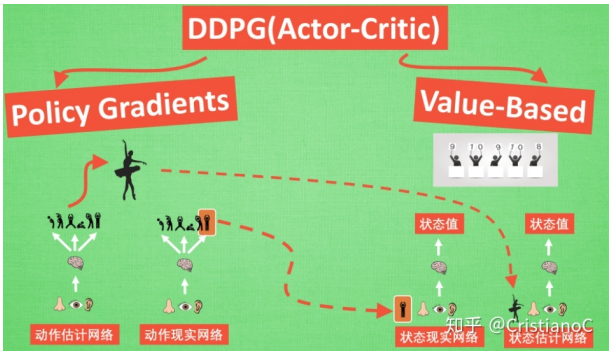
DDPG 算法它吸收了 Actor-Critic 让 Policy gradient 单步更新的精华，而且还吸收让计算机学会玩游戏的 DQN 的精华，合并成了一种新算法，叫做 Deep Deterministic Policy Gradient. 那 DDPG 到底是什么样的算法呢，我们就拆开来分析。我们将 DDPG 分成'Deep' 和'Deterministic Policy Gradient'，然后'Deterministic Policy Gradient' 又能细分为'Deterministic' 和'Policy Gradient'，接下来我们就开始一个个分析。

Deep 顾名思义，就是走向更深层次，我们在 DQN 中提到过，我们使用一个经验池和两套结构相同，但参数更新频率不同的神经网络能有效促进学习。那我们也把这种思想运用到 DDPG 中，使 DDPG 也具备这种优良形式。但是 DDPG 的神经网络形式却比 DQN 的要复杂一点。





Policy Gradient 相比其他的强化学习方法，它能被用来在连续动作上进行动作的筛选。而且筛选的时候是根据所学习到的动作分布随机进行筛选，而 Deterministic 有点看不下去，Deterministic 说：我说兄弟，你其实在做动作的时候没必要那么不确定，反正你最终都只是要输出一个动作值，干嘛要随机。所以 Deterministic 就改变了输出动作的过程，只在连续动作上输出一个动作值。



现在我们来说说 DDPG 中所用到的神经网络（粗略）。它其实和我们之前提到的 Actor-Critic 形式差不多，也需要有基于策略 Policy 的神经网络和基于价值 Value 的神经网络。但是为了体现 DQN 的思想，每种神经网络我们都需要再细分

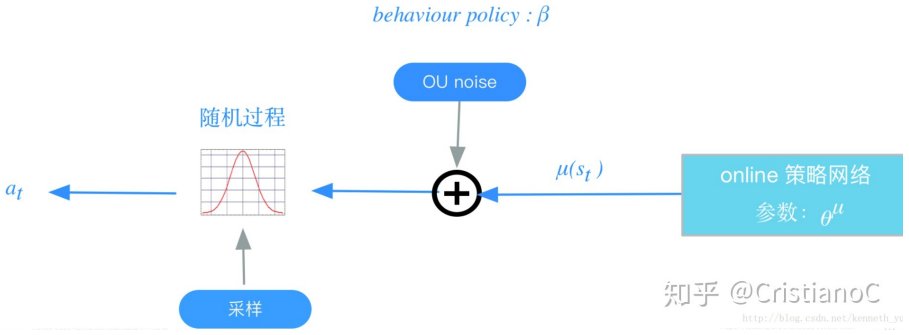
成两个，Policy Gradient 这边，我们有估计网络和现实网络，估计网络用来输出实时的动作，供 actor 在现实中实行。而现实网络则是用来更新价值网络系统的。所以我们再来看看价值系统这边，我们也有现实网络和估计网络，他们都在输出这个状态的价值，而输入端却有不同，状态估计网络则是拿着当时 Actor 施加的动作当做输入。在实际运用中，DDPG 这种做法确实带来了更有效的学习过程。

2.2 算法相关概念和定义

1. 确定性行为策略 μ：定义为一个函数，每一步的行为可以通过 $s_t = \mu(s_t)$ 计算获得。

2. 策略网络：用一个卷积神经网络对 μ 函数进行模拟拟合，这个网络我们就叫做策略网络，其参数为 θ^μ;

3. behavior policyβ：在 RL 训练过程中，我们要兼顾两个 e：exploration 和 exploit（也就是之前说过的探索和开发）；exploration 的目的是探索潜在的更优策略，所以训练过程中，我们为 action 的决策机制引入随机噪声：将 action 的决策从确定性的过程变为一个随机过程，再从这个随机过程中采样得到 action，下达给环境执行，过程如下图所示: 上述这个策略叫做 behavior



策略，用 β 来表示，这时 RL 的训练方式叫做 off-policy。这里与 $\epsilon - greedy$ 的思路是类似的。DDPG 中，使用 Uhlenbeck-Ornstein 随机过程（下面简称 UO 过程），作为引入的随机噪声：UO 过程在时序上具备很好的相关性，可以使 agent 很好的探索具备动量属性的环境。

4. Q 函数：即 action-value 函数，定义在状态 s_t 下，采取动作 a_t 后，且如果持

续执行策略 μ 的情况下，所获得的 R_t 期望值，用 Bellman 等式来定义：

$$Q^{\mu}(s_t, a_t) = E[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))] \tag{5}$$

可以看到，Q 函数的定义是一个递归表达，在实际情况中，我们不可能每一步都递归计算 Q 的值，可行的方案是通过一个函数对 Bellman 等式表达进行模拟。

5. Q 网络：DDPG 中，我们用一个卷积神经网络对 Q 进行模拟，这个网络我们就叫做 Q 网络，其参数为 θ^Q，采用了 DQN 相同的方法。
6. 如何衡量一个策略 μ 的表现：用一个函数 J 来衡量，我们叫做 performance objective，针对 off-policy 学习的场景.
7. 训练的目标：最大化 $J_\beta(\mu)$，同时最小化 Q 网络的 Loss,$\mu = argmax_\mu J(\mu)$
8. 最优 Q 网络定义：具备最小化的 Q 网络 Loss；训练 Q 网络的过程，就是寻找 Q 网络参数的最优解的过程，我们使用 SGD 的方法。

2.3 DDPG 实现框架和算法

以往的实践证明，如果只使用单个 Q 神经网络的算法，学习过程很不稳定，因为 Q 网络的参数在频繁梯度更新的同时，又用于计算 Q 网络和策略网络的 gradient。基于此，DDPG 分别为策略网络、Q 网络各创建两个神经网络拷贝，一个叫做 online，一个叫做 target：

$$\text{策略网络} \begin{cases} online : \mu(s|\theta^{\mu}) : gradient\text{更新}\theta^{\mu} \\ target : \mu'(s|\theta^{\mu'}) : soft\ update\ \theta^{\mu'} \end{cases}$$

$$Q\text{网络} \quad \begin{cases} online : Q(s,a|\theta^{Q}) : gradient\text{更新}\theta^{Q} \\ target : Q'(s,a|\theta^{Q'}) : soft\ update\ \theta^{Q'} \end{cases}$$

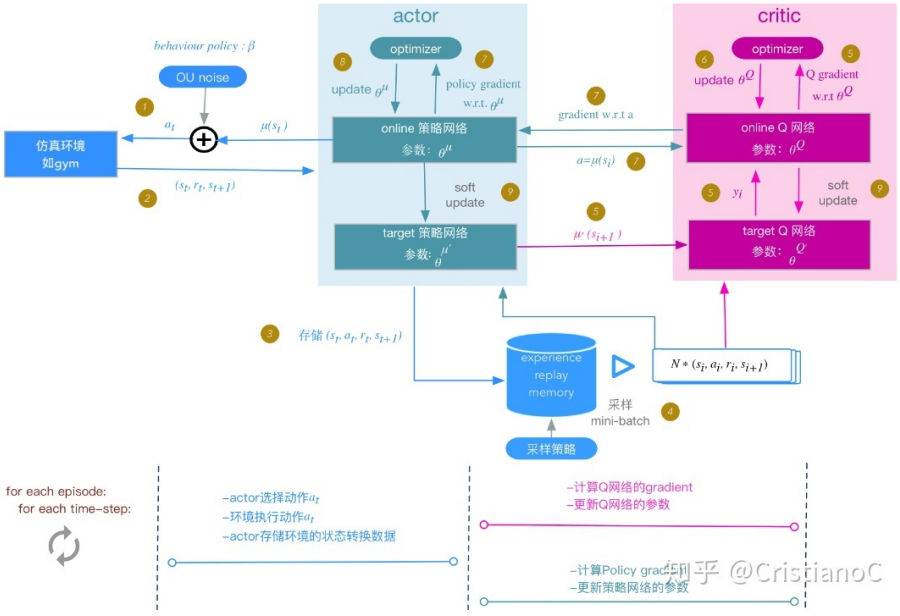
在训练完一个 mini-batch 的数据之后，通过 SGA/SGD 算法更新 online 网络的参数，然后通过 soft update 算法更新 target 网络的参数。soft update 是一种 running average 的算法：

$$soft\ update: \begin{cases} \theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{cases}$$

τ一般取值0.001

- target 网络参数变化小，用于在训练过程中计算 online 网络的 gradient，比较稳定，训练易于收敛。
- 参数变化小，学习过程变慢。

2.3.1 算法流程



1. actor 根据 behavior 策略选择一个 a_t，下达给 gym 执行该 a_t。behavior 策略是一个根据当前 online 策略 μ 和随机 UO 噪声生成的随机过程, 从这个随机过程采样获得 a_t 的值。

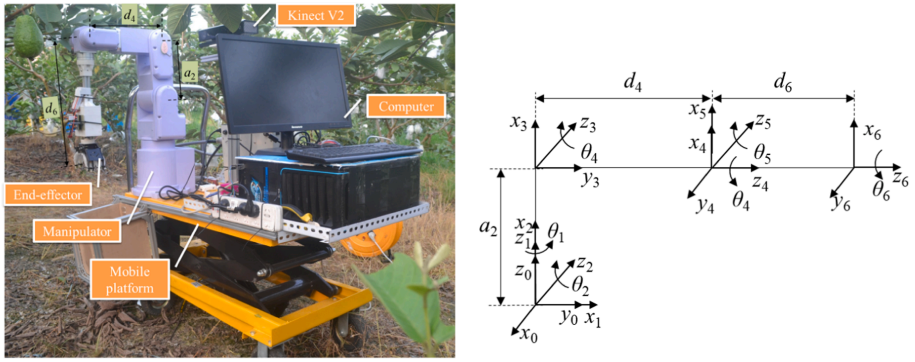
$$a_t = \mu(s_t|\theta^{\mu}) + N_t \tag{6}$$

2. gym 执行 a_t，返回 reward r_t 和新的状态 s_{t+1};
3. actor 将这个状态转换过程 (transition): (s_t, a_t, r_t, s_{t+1}) 存入 replay memory buffer 中，作为训练 online 网络的数据集。

4. 从 replay memory buffer R 中，随机采样 N 个 transition 数据，作为 online 策略网络、online Q 网络的一个 mini-batch 训练数据。我们用 (s_t, a_t, r_t, s_{t+1}) 表示 mini-batch 中的单个 transition 数据。
5. 计算 online Q 网络的 gradient：
6. update online Q: 采用 adam optimizer 更新 θ^Q;
7. 计算策略网络的 policy gradient;
8. update online 策略网络：采用 Adam optimizer 更新 θ^Q;
9. soft update target 网络 μ' 和 Q'

3 应用

得益于 DDPG 算法能够对连续性信息进行学习，因而其在机器人自主避障、农业机器人自主采摘等方面有着广阔的应用前景。现以 COMPAG 中 “Collision-free path planning for a guava-harvesting robot based on recurrent deep reinforcement learning” 一文具体介绍 DDPG 算法的农业机器人领域的应用。



(a) (b)

Figure 1: (a) 机器人实图 (b) 机械手连路坐标系

1. 学习系统输入信息：植株图像 这一步是为了让机器人 “看见” 树枝和果实并作为输出信息，机器人需学习得到避开障碍物、成功抓取果实的最优路径。
2. 建立 DDPG 算法模型，伪代码如下：

(a) (b) (c)

Figure 2: (a) 原始图像 (b) 图像分割结果 (c) 图像 3D 建模

Algorithm 1 recurrent DDPG

1: Initialize critic network $Q(s, a|\theta^Q)$ and actor network $\pi(s|\theta^\pi)$ with weights θ^Q and θ^π, initialize target networks $Q^{'}(s, a|\theta^{Q^{'}})$ and $\pi^{'}(s|\theta^{\pi^{'}})$ with weights $\theta^{Q^{'}}$ and $\theta^{\pi^{'}}$, and set replay buffer $D = \varnothing$

2: **For** episode = 1 to M:

3: Sample an RGB-D image from the training set, detect 3D fruits and line segments and record their positions as D_{fruits} and D_{lines}, and perform a random rigid transform to D_{fruits} and D_{lines} to improve randomization

4: Randomly select a goal x_g from D_{fruits}, set the manipulator to a default pose, and observe a state $\mathbf{s}_0$

5: **For** t = 0 to T-1:

6: A random action a_t is selected with probability 25%; otherwise, a_t is set to $\pi(s|\theta^\pi)$

7: Execute a_t, receive a reward r_t, observe a new state s_{t+1}, and add (s_t, a_t, r_t, s_{t+1}) to D

8: Sample N trajectories from D, sample a sub-trajectory with a length from each of these trajectories: $\{\{(s_0^n, a_0^n, r_0^n, s_1^n), \cdots, (s_{L-1}^n, a_{L-1}^n, r_{L-1}^n, s_L^n)\}\}_{n=0}^{N-1}$, and set the initial hidden state of the LSTM of each network for each sub-trajectory to zero

9: **For** l = 0 to L-1:

10: Calculate $\{y_i^n = r(s_i^n, a_i^n) + \gamma Q^{'}(s_i^n, \pi^{'}(s_i^n|\theta^{\pi^{'}})|\theta^{Q^{'}})\}_{n=0}^{N-1}$

11: Update θ^Q by minimizing $L(\theta^Q) = \sum_{n=0}^{N-1}(y_i - Q(s_i^n, a_i^n|\theta^Q))^2$

12: Fix θ^Q and update θ^π by maximizing $L(\theta^\pi) = \sum_{n=0}^{N-1}(Q(s_i^n, \pi(s_i^n|\theta^\pi)|\theta^Q))^2$

13: Update the hidden state of the LSTM of each network

14: Update the target networks: $\theta^{\pi^{'}} = \tau\theta^\pi + (1-\tau)\theta^{\pi^{'}}, \theta^{Q^{'}} = \tau\theta^Q + (1-\tau)\theta^{Q^{'}}$

3. 训练结果

Path lengths of recurrent DDPG, DDPG, and bidirectional RRT on test set.

Algorithm	Average (mm)	Standard deviation (mm)
Recurrent DDPG	498.81	123.05
DDPG	770.58	344.83
Bidirectional RRT	585.91	127.88

Running time of recurrent DDPG, DDPG, and bidirectional RRT on test set.

Algorithm	Average (s)	Standard deviation (s)
Recurrent DDPG	0.029	0.030
DDPG	0.040	0.029
Bidirectional RRT	8.005	7.167

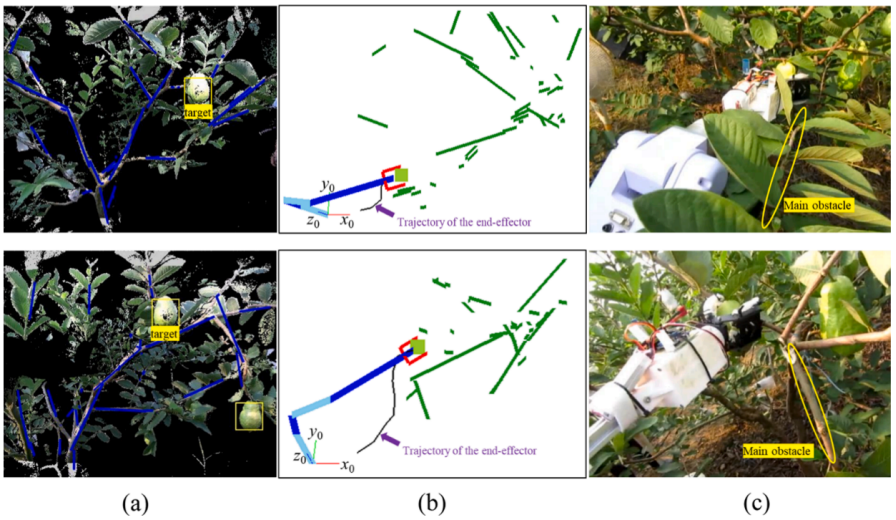


Figure 3: 机器人使用循环 DDPG 作为现场路径规划器：(a) 水果和障碍物定位；(b) 无碰撞路径规划；(c) 不与障碍物碰撞地接近目标.