DISTRIBUTED DISTRIBUTIONAL DETERMINISTIC POLICY GRADIENTS

Gabriel Barth-Maron,* Matthew W. Hoffman,* David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, Timothy Lillicrap
DeepMind
London, UK
{gabrielbm, mwhoffman, budden, wdabney, horgan, dhruvat, alimuldal, heess, countzero}@google.com

【强化学习 50】D4PG

张楚珩
清华大学 交叉信息院博士在读

全称是Distributed Distributional Deep Deterministic Policy Gradient，简称D4PG。

原文传送门

Barth-Maron, Gabriel, et al. "Distributed distributional deterministic policy gradients." arXiv preprint arXiv:1804.08617 (2018).

特色

本来以为这一篇是把distributional RL的想法用在policy-based算法上的，但其实只是用在了actor-critic类算法的critic上，不过本文做了很多实验，探究了一些实验技术对于最终效果的影响。

过程

1. 算法描述

本文主要在DDPG的基础上加上了如下一些技术，并且观察了它们各自的贡献

- 分布式：由于是off-policy的算法，因此可以使用多个actor去分布式地采样，然后存储在同一个replay buffer中，learner从buffer中采样，更新之后再将权重同步到各个actor上。参考Ape-X[1]。
- critic使用价值函数分布：critic权重的损失函数如下，其中带撇的表示target network，$\mathcal{T}_\pi$ 代表distributional Bellman operator，d 代表分布之间的距离度量，文章中实际使用cross-entropy。

$$L(w) = \mathbb{E}_\rho \Big[d(\mathcal{T}_{\pi_\theta} Z_{w'}(\mathbf{x}, \mathbf{a}), Z_w(\mathbf{x}, \mathbf{a})) \Big]$$

- 使用n-step TD误差：这样可以减少更新的variance。
- 使用prioritized experience replay：可以加速学习。

Algorithm 1 D4PG

Input: batch size M, trajectory length N, number of actors K, replay size R, exploration constant ϵ, initial learning rates α_0 and β_0

Initialize network weights (θ, w) at random
Initialize target weights $(\theta', w') \leftarrow (\theta, w)$
Launch K actors and replicate network weights (θ, w) to each actor
for $t = 1, \ldots, T$ **do**
  Sample M transitions $(\mathbf{x}_{i:i+N}, \mathbf{a}_{i:i+N-1}, r_{i:i+N-1})$ of length N from replay with priority p_i
  Construct the target distributions $Y_i = \left(\sum_{n=0}^{N-1} \gamma^n r_{i+n}\right) + \gamma^N Z_{w'}(\mathbf{x}_{i+N}, \pi_{\theta'}(\mathbf{x}_{i+N}))$
  Note, although not denoted the target Y_i may be projected (e.g. for Categorical value distributions).
  Compute the actor and critic updates

$$\delta_w = \frac{1}{M} \sum_i \nabla_w (Rp_i)^{-1} d(Y_i, Z_w(\mathbf{x}_i, \mathbf{a}_i))$$

$$\delta_\theta = \tfrac{1}{M} \sum_i \nabla_\theta \pi_\theta(\mathbf{x}_i)\, \mathbb{E}[\nabla_{\mathbf{a}} Z_w(\mathbf{x}_i, \mathbf{a})]\big|_{\mathbf{a}=\pi_\theta(\mathbf{x}_i)}$$

  Update network parameters $\theta \leftarrow \theta + \alpha_t\, \delta_\theta$, $w \leftarrow w + \beta_t\, \delta_w$
  If $t = 0 \mod t_{\text{target}}$, update the target networks $(\theta', w') \leftarrow (\theta, w)$
  If $t = 0 \mod t_{\text{actors}}$, replicate network weights to the actors
end for
return policy parameters θ

Actor

repeat
  Sample action $\mathbf{a} = \pi_\theta(\mathbf{x}) + \epsilon\mathcal{N}(0, 1)$
  Execute action $\mathbf{a}$, observe reward r and state $\mathbf{x}'$
  Store $(\mathbf{x}, \mathbf{a}, r, \mathbf{x}')$ in replay
until learner finishes

2. 实验任务

主要在三组连续控制任务上做了实验

- 标准的Mujoco benchmark，例如包含Cheetah、Humanoid、Hopper这些；
- 机械手控制任务，主要包括三个任务用手接住掉下来的圆柱体、捡起一个物体并且把物体移动到指定位置和朝向、旋转圆柱体到指定朝向；
- 跑酷任务（Parkour），二维或者三维的小人，控制小人越过障碍物，第一人称的障碍物表示也会作为状态传送给agent。

3. 实验结果

- 没有研究非分布式结果会怎样；

- 加上价值函数分布之后，性能普遍有所提升；
- 使用n-step target效果普遍有提升；
- 使用prioritized experience replay在不使用价值函数分布的时候帮助比较大，但是使用价值函数分布之后效果不明显；

参考文献

[1] Horgan, Dan, et al. "Distributed prioritized experience replay."*arXiv preprint arXiv:1803.00933*(2018).

发布于 2019-04-01

强化学习 (Reinforcement Learning)