Under review as a conference paper at ICLR 2019

COMPETITIVE EXPERIENCE REPLAY

Anonymous authors
Paper under double-blind review

【强化学习算法 35】CER

张楚珩
清华大学 交叉信息院博士在读

CER是Competitive Experience Replay的简称，是一种增大探索的方法。

原文传送门

Anonymous, Competitive experience replay, Submitted to International Conference on Learning Representations, 2019 (under review)

特色

使用了两个智能体的相互竞争关系去增大策略的探索，和本专栏前面讲到的【强化学习算法 26】RND有类似之处。RND是使用一个随机神经网络来判断某个状态是否比较“新颖”，这里使用了另一个同步学习的智能体来做类似的判断。

虽然文章一直在强调其在稀疏奖励任务上的出色表现，但是从实验结果上来看并不能证明CER算法能够在稀疏奖励的任务上有很好的表现，其良好的表现主要来自于附加使用的HER（参考本专栏前一篇【强化学习算法 34】HER）。但是这是一个使用多智能体来帮助探索的新思路。

过程

1. 附加目标的MDP

和【强化学习算法 34】HER类似，这里定义了附加目标时的MDP。相比于HER更简单的是，这里直接定义目标空间 G 为状态空间 S 的子集。因此各种表示更简洁，策略 $\pi: S \times G \to A$ ，奖励函数 $r_g: S \times A \times G \to R$ 定义为

$$r_t = r_g(s_t, a_t, g) = \begin{cases} 0, \text{if } |s_t - g| < \delta \\ -1, \text{otherwise} \end{cases}$$

2. 使用两个智能体来帮助探索

为了帮助探索，这里考虑使用两个智能体来学习同一个任务。智能体A是最好要使用的智能体，智能体A如果表现的和智能体B相似就会收到惩罚，但是智能体B如果表现的和智能体A相似就会受到奖励。

考虑两个智能体 π_A 和 π_B ，它们样本采集上都是独立完成的（**decentralized execution**），即它们都各自执行各自的策略 $\pi_k(a_k|s_k, g_k)$， $k \in \{A, B\}$ ，这样它们各自能形成自己经验池 $\{(s_A, a_A, g_A, r_A, s'_A)\}$ 和 $\{(s_B, a_B, g_B, r_B, s'_B)\}$ 。如果要在这上面做HER，同样也可以在各自的经验池里面独立地做。

接下来，需要对这两个智能体的奖励进行调整（**re-labeling**），用以实现前面制定的目标。在每次训练的时候会采样得到一个mini-batch

$$\{(s^i_A, a^i_A, g^i_A, r^i_A, s'^i_A), (s^i_B, a^i_B, g^i_B, r^i_B, s'^i_B)\}^m_{i=1}, \text{where } m \text{ is the size of the mini-batch.}$$

在这个mini-batch内，对于任意一个A的状态 s^i_A ，**如果存在任意一个** s^j_B ，使得 $|s^i_A - s^j_B| < \delta$ ，就把 r^i_A 重新标记为 $r^i_A - 1$ ；对于任意一个B的状态 s^i_B ，**每存在一个** s^j_A ，使得 $|s^i_B - s^j_A| < \delta$ ，都把 r^i_B 增加 1 。

最后，会把这两个智能体放在一块训练（**centralized training**）。考虑使用一个确定性策略，使用DDPG方法训练。这里使用了多智能体的DDPG方法（MADDPG），其实区别就在于使用了一个包含两个智能体状态、行动和目标的Q函数。训练方法类似，对于每一个策略来说按照如下梯度来做梯度上升

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\boldsymbol{\mu}}[\nabla_{\theta_i} \mu_i(s_i, g_i) \nabla_{a_i} Q^{\boldsymbol{\mu}}_i(\mathbf{s}, \mathbf{a}, \mathbf{g})|_{a_i = \mu_i(s_i, g_i)}], \tag{6}$$

对于critic来说，最小化如下损失函数（文章写得不仔细，这里的 θ 怎么又变成了Q的参数了？）

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}'}[(Q^{\boldsymbol{\mu}}_i(\mathbf{s}, \mathbf{a}, \mathbf{g}) - y)^2], \quad y = r_i + \gamma Q^{\boldsymbol{\mu}'}_i(\mathbf{s}', a'_1, \ldots, a'_N, \mathbf{g})\big|_{a'_j = \boldsymbol{\mu}'_j(s_j)}, \tag{7}$$

3. 不同版本

文章提出了两种方法，一种方法让B和A一样使用任务所规定的初始状态分布（ind-CER）；另一种方法让B使用A轨迹上的状态分布作为初始状态分布（当然，这要求环境能够支持这一设定初始状态，即int-CER）。效果上而言，后一种方法效果更好。

算法

Algorithm 1 HER with CER

Initialize a random process $\mathcal{N}$ for action exploration, max episode length to L
Set CER to *ind-CER* or *int-CER*
for episode $= 1$ to M **do**
  Receive initial state s_A
  Receive a goal r_A for this episode
  Initialize episode buffer *buffer*$_A$
  for $t = 1$ to L **do**
    select action $a_A = \boldsymbol{\mu}_{\theta_i}(s_A) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions a_A and observe reward r_A and new state s'_A
    Store $(s_A, a_A, g_A, r_A, s_A{}')$ in *buffer*$_A$
    $s_A \leftarrow s'_A$
  end for
  Receive initial state s_B or Receive initial state s_B, where s_B is a state sampled from *buffer*$_A$
  Receive a goal r_B for this episode
  Initialize episode buffer *buffer*$_B$
  for $t = 1$ to L **do**
    select action $a_B = \boldsymbol{\mu}_{\theta_i}(s_B) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions a_B and observe reward r_B and new state s'_B
    Store $(s_B, a_B, g_B, r_B, s_B{}')$ in *buffer*$_B$
    $s_B \leftarrow s'_B$
  end for
  Concatenate *buffer*$_A$ and *buffer*$_B$ and store $(\{\mathbf{s^i}\}^T_{i=1}, \{\mathbf{a^i}\}^T_{i=1}, \{\mathbf{g^i}\}^T_{i=1}, \{\mathbf{r^i}\}^T_{i=1}, \{\mathbf{s'^i}\}^T_{i=1})$ in replay buffer $\mathcal{D}$
  // Optimization based on off-policy samples
  for k $= 1$ to K **do**
    // Relabelling off-policy samples
    Sample a random minibatch of S samples $(\mathbf{s}^j, \mathbf{a}^j, \mathbf{g}^j, \mathbf{r}^j, \mathbf{s}'^j)$ from $\mathcal{D}$
    Apply HER strategy on samples
    Apply *ind-CER* or *int-CER* strategy on samples
    for agent $i = A, B$ **do**
      Do one step optimization based on Eq (6) and Eq (7), and update target networks.
    end for
  end for
end for

评论

个人认为，该方法不论是从想法还是实验效果上来说，基本上被RND方法（【强化学习算法26】RND）dominate。

- 这里策略B的目的就是看看什么状态比较容易达到，从而惩罚策略A不要老去到达比较容易达到的状态；而RND里面直接用一个随机神经网络更加直接地达到目的。
- 这里发现策略B如果从策略A轨迹上随机选择状态作为初始分布更好，是为了防止策略B与策略A在轨迹末端重合过少；而RND里面直接用A的轨迹来训练，直接就规定了状态分布的匹配。
- 这里相对于RND唯一的好处可能在于它更加的是针对目标的探索，不过文中没有给出很有力的说明。

文献由 @小红菌 推荐

发布于 2018-12-03

机器学习 算法 强化学习 (Reinforcement Learning)

文章被以下专栏收录

强化学习前沿
读呀读paper

进入专栏