Under review as a conference paper at ICLR 2019

Remember and Forget for Experience Replay

Guido Novati & Petros Koumoutsakos
Computational Science and Engineering Lab.
ETH Zurich
{novatig,petros}@ethz.ch

【强化学习 51】RACER

张楚珩
清华大学 交叉信息院博士在读

文章提出了一种Experience Replay的改进，叫做Remember and Forget Experience Replay，简称ReF-ER；在此基础上把它应用到了之前的ACER（Actor-critic with Experience Replay），形成RACER。

原文传送门

Novati, Guido, and Petros Koumoutsakos. "Remember and Forget for Experience Replay." arXiv preprint arXiv:1807.05827 (2018).

特色

首先吸引我的是这篇文章把Mujoco control tasks分数刷的巨高。看完之后个人感觉造成其性能大幅提升的主要是把TRPO、PPO那一套里面最为精髓的conservative policy update（限制policy变化的幅度，比如用KL），通过ER反馈出来，这使得各种off-policy的RL算法都能用上。说白了，off-policy虽然能用到过期的数据，但是过期太久的总是不好用的，要想让算法有效地利用更多好的数据，就需要在保证ER容量的基础上保证ER的质量。

过程

1. ReF-ER

我们知道在ER用于off-policy算法的时候，需要根据behavior policy（样本采集时的策略）和target policy（当前策略）的差距来进行权重的调整，调整的权重为 $\rho_t = \pi^w(a_t|s_t)/\mu_t$ 。当它们差距过大的时候，要么权重过大带来很大的方差，要么权重过小使得该样本完全没用。ReF-ER的主要思想就是维持ER中大部分样本都是near-policy的，主要措施有以下几点

- ER中的每个transition会另外储存一个的 ρ_t ，在使用ER的时候，会对ER均匀采样，当该样本被采样的时候会重新计算 ρ_t 并更新。
- 对每个样本，如果 $1/c_{\max} < \rho_t < c_{\max}$ 那么它被看做是near-policy的，反之则是far-policy的，当新样本加入，ER容量达到上限 n_{obs} 的时候，就会删除far-policy比例最高的一些轨迹。
- 只使用near-policy的样本更新策略梯度，并且限制新策略不要离ER中的旧策略离得太远，即策略梯度为

$$\hat{\mathbf{g}}_t^{\text{ReF-ER}}(\mathtt{w}) = \begin{cases} \beta\, \hat{\mathbf{g}}_t(\mathtt{w}) - (1-\beta)\nabla D_{\text{KL}}\left[\mu_t \| \pi^{\mathtt{w}}(\cdot|s_t)\right] & \text{if } 1/c_{\max} < \rho_t < c_{\max} \\ -(1-\beta)\nabla D_{\text{KL}}\left[\mu_t \| \pi^{\mathtt{w}}(\cdot|s_t)\right] & \text{otherwise} \end{cases}$$

其中参数 $\beta \in [0,1]$ 是动态更新的，其目标是保持far-policy样本比例在 $D \in (0,1)$ 左右。

$$\beta \leftarrow \begin{cases} (1-\eta)\beta & \text{if } n_{\text{far}}/n_{\text{obs}} > D \\ (1-\eta)\beta + \eta, & \text{otherwise} \end{cases}$$

$c_{\max}$ 取的较小时，off-policy 算法就退化为 on-policy 算法，在迭代后期让算法缓慢过渡到 on-policy 算法能够提高算法渐近性能。

$$c_{\max}(k) = 1 + C/(1 + 5e{-}7 \cdot k),$$

ReF-ER 的超参数有

- Buffer 的大小 N ，buffer 较小时采集到样本不多样，估计到的梯度方向可能不准确；buffer 较大时对应的 far-policy 样本会很多；
- $c_{\max}$ 的参数 C ，有些任务对于噪声容忍度大，较大的 $c_{\max}$ 能提高较大的探索，学习较好，但是有些任务需要精细的控制，较大的 $c_{\max}$ 会使得结果变得更差；
- Far-policy 样本比例 D ；

2. RACER

RACER与本专栏前面讲的ACER算法类似，都是从ER里采集off-policy的数据，并且使用这些数据去估计on-policy的价值函数，并且利用得到的价值函数来进行策略梯度更新。

RACER 使用同一个多端输出的神经网络来得到所有需要的参数，其拟合的所有参数如下面右图所示。

- 其中 m, Σ 是策略网络需要的参数；
- 为了更新策略梯度，需要 on-policy 的 advantage 估计 A_t^{ret} ，该估计由 $Q_t^{\text{ret}} - V^{w}$ 得到，因此又新增了 $V^{w}(s)$ 去估计 $V^{\pi}(s)$ ；
- 估计 Q_t^{ret} 又需要 $Q^{\pi}(s,a)$ 的估计，因为已经有了 $V^{\pi}(s)$ 的估计，这样就只需要外加一个 A^{π} 的估计了，为了编码它，需要用到 m, Σ, K, L 。

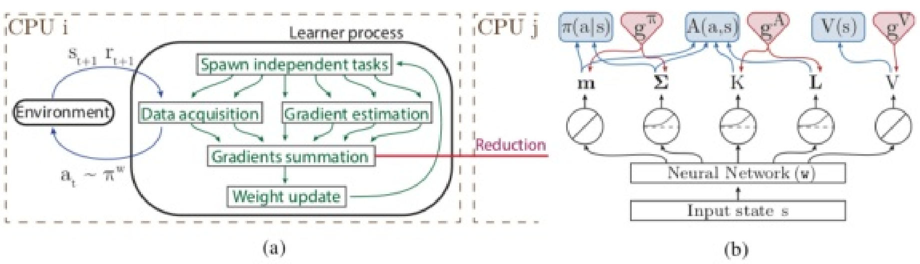

Figure 1: (a) Diagram of asynchronous ER-based RL algorithms. (b) Neural network architecture employed by RACER. Blue arrows connect each output with the elements of the actor-critic framework for which it is used. Red arrows represent the flow of the error signals.

2.1. 学习 m,Σ

使用 off-PG 来计算策略梯度

$$\hat{\mathbf{g}}_t^{\pi}(\mathtt{w}) = \rho_t \hat{A}_t^{\text{ret}} \nabla_{\{\mathbf{m},\,\boldsymbol{\Sigma}\}} \log \pi^{\mathtt{w}}(a_t \mid s_t)$$

这样就需要一个对于 A^{π} 的估计，这里使用 Retrace 技术（见本专栏 ACER 文章）来估计

$$\hat{Q}_t^{\text{ret}} = r_{t+1} + \gamma V^{\mathtt{w}}(s_{t+1}) + \gamma \min\{1, \rho_{t+1}\} \left[\hat{Q}_{t+1}^{\text{ret}} - Q^{\mathtt{w}}(s_{t+1}, a_{t+1})\right]$$

$$\hat{A}_t^{\text{ret}} := \hat{Q}_t^{\text{ret}} - V^{\mathtt{w}}(s_t).$$

这样就需要神经网络估计的 V 函数和 Q 函数，和 dueling network 的做法类似，同一个神经网络输出 V 函数和 A 函数，这样 $Q^w(s,a) = V^w(s) + A^w(s,a)$ 。

2.2 学习 V

和 Retrace 技术类似，可以得到 V 函数的更新目标

$$\hat{V}_t^{\text{tbc}} = V^{\mathtt{w}}(s_t) + \min\{1, \rho_t\}(\hat{Q}_t^{\text{ret}} - Q^{\mathtt{w}}(s_t, a_t))$$

通过最小化均方误差可以得到梯度下降更新

$$\hat{\mathbf{g}}_t^{V}(\mathtt{w}) = \hat{V}^{\text{tbc}}(s) - V^{\mathtt{w}}(s_t) = \min\{1,\ \rho_t\} \left[\hat{Q}_t^{\text{ret}} - Q^{\mathtt{w}}(s_t, a_t)\right]$$

2.3. 学习 K,L

A 函数的神经网络表示有几种可能的形式：要么把 (s,a) 编码作为输入传入到神经网络里面，但是神经网络的输出不能自然满足 $\mathbb{E}_{a\sim\pi} A^{\pi}(s,a) = 0$ ，因此需要额外进行类似 ACER 文章里面Stochastic Dueling Networks的操作，另外对 A^w 采样，并减去采样平均；要么规定 A^w 为一个可解析的凸函数，然后用神经网络输出这个函数的参数，其好处是有可能直接用相关的参数定义 A^w 使其自然满足 $\mathbb{E}_{a\sim\pi} A^{\pi}(s,a) = 0$ 。本文采用了后一种方法

定义

$$A^{\mathtt{w}}(s,a) := f^{\mathtt{w}}(s,a) - \mathbb{E}_{a'\sim\pi}\left[f^{\mathtt{w}}(s,a')\right]$$

其中

$$f^{w}(s,a) = K(s)\ \exp\left[-\frac{1}{2}\mathbf{a}_+^{\intercal}\,\mathbf{L}_+^{-1}(s)\,\mathbf{a}_+ - \frac{1}{2}\mathbf{a}_-^{\intercal}\,\mathbf{L}_-^{-1}(s)\,\mathbf{a}_-\right]$$

Here $\mathbf{a}_- = \min\left[a - \mathbf{m}(s), \mathbf{0}\right]$ and $\mathbf{a}_+ = \max\left[a - \mathbf{m}(s), \mathbf{0}\right]$ (both element-wise operations).

参数 $K(s), L_+(s), L_-(s)$ 都由神经网络生成，并且 $L_+(s), L_-(s)$ 是对角矩阵，因此表示这三个参数只需要一个 $2d_A+1$ 维的向量。

而后面的 $\mathbb{E}_{a'\sim\pi}[f^{w}(s,a')]$ 可以解析地写出来

$$\mathbb{E}_{a'\sim\pi}\left[f^{w}_{\mathrm{DG}}(s,a')\right] = K(s)\prod_{i=1}^{d_A}\frac{\sqrt{\frac{L_{+,i}(s)}{L_{+,i}(s)+\Sigma_i(s)}} + \sqrt{\frac{L_{-,i}(s)}{L_{-,i}(s)+\Sigma_i(s)}}}{2}$$

这些参数的学习可以通过最小化 $(\hat{A}_t^{\mathrm{ret}} - A^{w}(s_t,a_t))^2$ 得到，

$$\hat{\mathbf{g}}_t^{A}(\mathtt{w}) = \rho_t\left[\hat{A}_t^{\mathrm{ret}} - A^{\mathtt{w}}(s_t,a_t)\right]\nabla_{\{K,\,\mathbf{L}\}}A^{\mathtt{w}}(s_t,a_t)$$

3. 实验设计

ReF-ER 可以用在任何 off-policy 算法上，文章尝试了三种算法 DDPG、NAF 和 RACER。这是三种不同的适用连续动作空间的 off-policy 算法。DDPG 使用确定性策略，每次把策略网络输出的行动往所估计的 $Q^{w}(s_t,a)$ 梯度上升方向更新；NAF 是连续动作版本的 Q-learning。

其他的工程细节包括

- linear scheduled learning rate annealing
- normalized reward
- normalize the states in a batch
- bound action within $[-1,1]^{d_A}$ in DDPG

实验结果

DDPG+Ref-ER和NAF+Ref-ER就不放上来了，效果最好的是RACER+Ref-ER。实验效果是我见过的最佳实验效果，如果大家看到更牛的实验结果，欢迎评论。

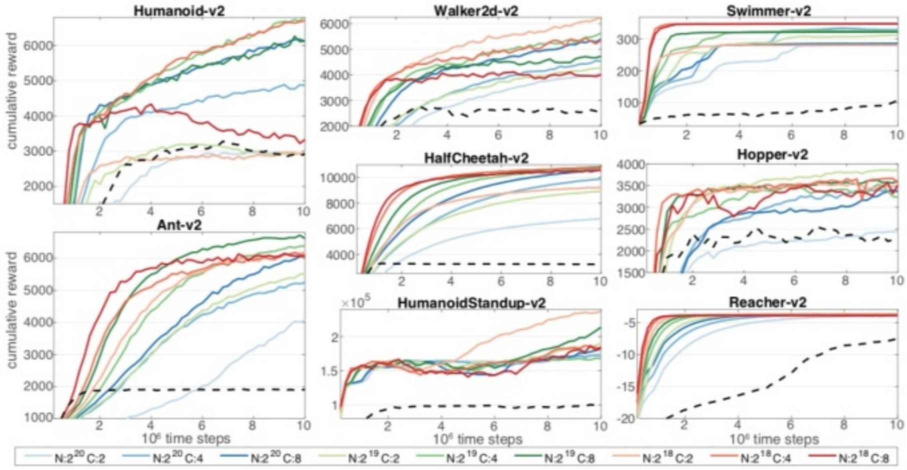

Figure 5: Average cumulative rewards on MuJoCo OpenAI Gym tasks obtained with `PPO` (dashed black lines) and with `RACER` by independently varying the two main hyper-parameters of `ReF-ER`: the RM size N and C (colored lines).

这篇文章用到了Retrace，这种技术可以使用off-policy的数据来估计on-policy的价值函数。理论推导上最为重要的技巧是weight truncation and bias correction trick，这里再复习一下。

1 Retrace Operator

Let importance ration $\rho_t = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}$, wherer π is the target policy whose value we wish to estimate, μ is the target policy where the samples are drawn from.

Observe the on-policy action value function can be expressed as expectation under behavior policy.

$$
\begin{aligned}
Q^\pi(s_t, a_t) &= \mathbb{E}_{r_t, s_{t+1} \sim P(\cdot|s_t, a_t)} \mathbb{E}_{a_{t+1} \sim \pi(\cdot|s_{t+1})} [r_t + \gamma Q^\pi(s_{t+1}, a_{t+1})] \\
&= \mathbb{E}_{r_t, s_{t+1} \sim P(\cdot|s_t, a_t)} [r_t + \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1}) \gamma Q^\pi(s_{t+1}, a_{t+1})] \\
&= \mathbb{E}_{r_t, s_{t+1} \sim P(\cdot|s_t, a_t)} [r_t + \sum_{a_{t+1}} \mu(a_{t+1}|s_{t+1}) \rho_{t+1} \gamma Q^\pi(s_{t+1}, a_{t+1})] \\
&= \mathbb{E}_{r_t, s_{t+1} \sim P(\cdot|s_t, a_t)} [r_t + \mathbb{E}_{a_{t+1} \sim \mu(\cdot|s_{t+1})} \rho_{t+1} \gamma Q^\pi(s_{t+1}, a_{t+1})] \\
&= \mathbb{E}_{r_t, s_{t+1} \sim P(\cdot|s_t, a_t)} \mathbb{E}_{a_{t+1} \sim \mu(\cdot|s_{t+1})} [r_t + \rho_{t+1} \gamma Q^\pi(s_{t+1}, a_{t+1})]
\end{aligned}
$$

Apply the weight truncation as bias correction trick

$$
\begin{aligned}
\mathbb{E}_{a \sim \mu}[\rho_t(a) \cdots] &:= \sum_a \mu(a|s_t) \frac{\pi(a|s_t)}{\mu(a|s_t)} \cdots \\
&= \mathbb{E}_{a \sim \mu}[\bar{\rho}_t(a) \cdots + [\rho_t(a) - 1]_+ \cdots] \\
&= \mathbb{E}_{a \sim \mu}[\bar{\rho}_t(a) \cdots + \mathbb{E}_{a \sim \pi}[[\frac{\rho_t(a) - 1}{\rho_t(a)}]_+ \cdots]] \\
&= \mathbb{E}_{a \sim \mu}[\bar{\rho}_t(a) \cdots + \mathbb{E}_{a \sim \pi}[M_t(a) \cdots]]
\end{aligned}
$$

where $\bar{\rho}_t = \min(c, \rho_t)$ and $M_t(a) = [\frac{\rho_t(a) - c}{\rho_t(a)}]_+$.

$$
\begin{aligned}
Q^\pi(s_0,a_0) &= \mathbb{E}_{r_0,s_1}\mathbb{E}_{a_1\sim\mu}[r_0 + \bar\rho_1\gamma Q^\pi(s_1,a_1) + \gamma\mathbb{E}_{a\sim\pi}\left(M_1(a)Q^\pi(x_1,a)\right)] \\
&= \mathbb{E}_1[r_0 + \bar\rho_1\gamma(\mathbb{E}_2[r_1 + \bar\rho_2\gamma Q^\pi(s_2,a_2) + \gamma\mathbb{E}_{a\sim\pi}\left(M_2(a)Q^\pi(x_2,a)\right)]) \\
&\quad + \gamma\mathbb{E}_{a\sim\pi}\left(M_1(a)Q^\pi(x_1,a)\right)] \\
&= \cdots \\
&= \mathbb{E}_{1,2,\cdots}[r_0 + \gamma\bar\rho_1 r_1 + \gamma^2\bar\rho_1\bar\rho_2 r_2 + \cdots \\
&\quad + \gamma\mathbb{E}_{a\sim\pi}\left(M_1(a)Q^\pi(x_1,a)\right) + \gamma^2\bar\rho_1\mathbb{E}_{a\sim\pi}\left(M_2(a)Q^\pi(x_2,a)\right) + \cdots] \\
&= \mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^t\bar\rho_i)(r_t + \gamma\mathbb{E}_{a\sim\pi}\left(M_{t+1}(a)Q^\pi(x_{t+1},a)\right))]
\end{aligned}
$$

The above can be written in fixed point iteration form, and we define the iteration as the retrace operator, which is the form defined in [1].

$$
\mathcal{R}Q(x_0,a_0) = \mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^t\bar\rho_i)(r_t + \gamma\mathbb{E}_{a\sim\pi}\left(M_{t+1}(a)Q(x_{t+1},a)\right))] \tag{1}
$$

For convenience in contraction analysis, it is better to be written as $\mathcal{R}Q(s,a) = Q(s,a) + \cdots$ form. Then we rewrite it. Observe that

$$
\mathbb{E}_{a\sim\pi}[Q(x_t,a)] = \mathbb{E}_{a\sim\mu}[\bar\rho_t Q(x_t,a)] + \mathbb{E}_{a\sim\pi}[M_t(a)Q(x_t,a)]
$$

and

$$
\begin{aligned}
&=\mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^t\bar\rho_i)\gamma\bar\rho_{t+1}Q(x_{t+1},a_{t+1})] \\
&=\mathbb{E}_\mu[\sum_{t\geq 1}\gamma^t(\prod_{i=1}^t\bar\rho_i)Q(x_t,a_t)] \\
&=\mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^t\bar\rho_i)Q(x_t,a_t)] - Q(x_0,a_0)
\end{aligned}
$$

Therefore, the retrace operator can also be written as, which is the form defined in [2].

$$
\begin{aligned}
\mathcal{R}Q(x,a) &= \mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^t\bar\rho_i)(r_t + \gamma\mathbb{E}_{a\sim\pi}Q(x_{t+1},a) - \gamma\bar\rho_{t+1}Q(x_{t+1},a_{t+1}))] \\
&= Q(x_0,a_0) + \mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^t\bar\rho_i)(r_t + \gamma\mathbb{E}_{a\sim\pi}Q(x_{t+1},a) - Q(x_t,a_t))]
\end{aligned}
$$

Theorem 1.1. *The operator $\mathcal{R}$ is a contraction operator such that*

$$
||\mathcal{R}Q - Q^\pi||_\infty \leq \gamma||Q - Q^\pi||_\infty
$$

2 ACER

2.1 Learn approximated on-policy value function

Suppose we have on-policy estimates $Q_\theta(s,a)$ and $V_\theta(s)$ and off-policy samples $\{s_t, r_t, a_t\}$. How do we set update target for $Q_\theta(s,a)$ and $V_\theta(s)$?

The update target for $Q_\theta(s_t, a_t)$ is $Q^{\text{ret}}(s_t, a_t)$.

$$
\begin{aligned}
Q^{\text{ret}}(s_t, a_t) &= Q^\pi(s_t, a_t) \\
&= \mathbb{E}_\mu[r_t + \gamma \rho_{t+1} Q^\pi(s_{t+1}, a_{t+1})] \\
&= \mathbb{E}_\mu[r_t + \gamma \bar{\rho}_{t+1} Q^\pi(s_{t+1}, a_{t+1}) + \gamma \mathbb{E}_{a\sim\pi}[M_{t+1}(a) Q^\pi(s_{t+1}, a)]] \\
&= \mathbb{E}_\mu[r_t + \gamma \bar{\rho}_{t+1} Q^{\text{ret}}(s_{t+1}, a_{t+1}) + \gamma \mathbb{E}_{a\sim\pi}[Q^\pi(s_{t+1}, a)] - \gamma \mathbb{E}_\mu[\bar{\rho}_{t+1} Q^\pi(s_{t+1}, a)]] \\
&\to \mathbb{E}_\mu[r_t + \gamma V_\theta(s_{t+1}) + \gamma \bar{\rho}_{t+1}(Q^{\text{ret}}(s_{t+1}, a_{t+1}) - Q_\theta(s_{t+1}, a_{t+1}))]] \\
&\to r_t + \gamma V_\theta(s_{t+1}) + \gamma \bar{\rho}_{t+1}(Q^{\text{ret}}(s_{t+1}, a_{t+1}) - Q_\theta(s_{t+1}, a_{t+1}))]
\end{aligned}
$$

In the third line, the first two terms are dominating term, and $\mathbb{E}_\mu[Q^\pi(s_{t+1}, a_{t+1})]$ can be replaced by Monte Carlo sample $Q^{\text{ret}}(s_{t+1}, a_{t+1})$. The last term is correction term, which can be approximated by neural network predictions. In the last line, $\mathbb{E}_\mu$ is replaced by samples, and $Q_\theta(s, a)$ and $V_\theta(s)$ is used making it a bootstrapped target. This target can be calculated from the end of an episode to the start.

Similarly, we can obtain the target for $V_\theta(s)$.

$$
\begin{aligned}
V^{\text{ret}}(s_t) &= V^\pi(s_t) \\
&= \mathbb{E}_\mu[\bar{\rho}_t Q^\pi(s_t, a)] + \mathbb{E}_\pi[M_t(a) Q^\pi(s_t, a)] \\
&\to \mathbb{E}_\mu[\bar{\rho}_t Q^{\text{ret}}(s_t, a_t)] + \mathbb{E}_\pi[Q^\pi(s_t, a)] - \mathbb{E}_\mu[\bar{\rho}_t Q^\pi(s_t, a)] \\
&\to \mathbb{E}_\mu[\bar{\rho}_t Q^{\text{ret}}(s_t, a_t) + V_\theta(s_t) - \bar{\rho}_t Q_\theta(s_t, a_t)] \\
&\to \bar{\rho}_t(Q^{\text{ret}}(s_t, a_t) - Q_\theta(s_t, a_t)) + V_\theta(s_t)
\end{aligned}
$$

At last, given samples, we can update neural network parameters along the gradient of the mean squared loss w.r.t. $Q^{\text{ret}}(s_t, a_t)$ and $V^{\text{ret}}(s_t)$.

As you can see, in retrace the target of $V_\theta(s)$ also involves $Q_\theta(s, a)$. In fact, you can estimate on-policy $V_\theta(s)$ without maintain action value function by V-trace[3].

2.2 Network representation

ACER uses dueling network to represent $A_\theta(s, a)$ and $V_\theta(s)$ and let $Q_\theta(s, a) := V_\theta(s) + A_\theta(s, a)$. The advantage estimator should sataisfy normalization condition $\sum_a \pi(a|s) A_\theta(s, a) = 0$. ACER uses stochastic dueling network that

$$
Q_\theta(s, a) \sim V_\theta(s) + A_\theta(s, a) - \frac{1}{n}\sum_{i=1}^{n} A_\theta(s, a_i'), \qquad a_i' \sim \pi_\theta(\cdot|s) \tag{2}
$$

2.3 Learn policy

The policy can be learnt from off-policy policy gradient. We start from off-policy policy gradient with baseliene and then apply weight truncation and bias correction trick.

$$
\begin{aligned}
g &= \mathbb{E}_\mu[\rho_t \nabla_\theta \log \pi_\theta(a_t|s_t)(Q^\pi(s_t, a_t) - V_\theta(s_t))] \\
&= \mathbb{E}_\mu[\bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|s_t)(Q^\pi(s_t, a_t) - V_\theta(s_t)) + \mathbb{E}_\pi[M_t(a) \nabla_\theta \log \pi_\theta(a|s_t)(Q^\pi(s_t, a) - V_\theta(s_t))]] \\
&\to \mathbb{E}_\mu[\bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|s_t)(Q^{\text{ret}}(s_t, a_t) - V_\theta(s_t)) + \mathbb{E}_\pi[M_t(a) \nabla_\theta \log \pi_\theta(a|s_t)(Q_\theta(s_t, a) - V_\theta(s_t))]]
\end{aligned}
$$

Stochastic gradient estimator

$$
g_t = \bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|s_t)(Q^{\text{ret}}(s_t, a_t) - V_\theta(s_t)) + [M_t(a') \nabla_\theta \log \pi_\theta(a'|s_t)(Q_\theta(s_t, a') - V_\theta(s_t))] \tag{3}
$$

where $a' \sim \pi_\theta(\cdot|s_t)$.

However, this gradient is not applied directly and an additional trust region optimization in statistics space of policy distribution is used.

$$\begin{aligned} &\min_{z_t} \frac{1}{2}||g_t - z_t||_2^2 \\ &s.t.\ \nabla_{\phi_\theta(s_t)} D_{KL}(\pi(\cdot|\phi_{\theta'}(s_t)\|\pi(\cdot|\phi_\theta(s_t)))^T z_t \leq \delta \end{aligned}$$

which can be solve analytically

$$z_t^* = g_t - \max(0, \frac{k^T g_t - \delta}{||k||_2^2})k \tag{4}$$

where $k = \nabla_{\phi_\theta(s_t)} D_{KL}(\pi(\cdot|\phi_{\theta'}(s_t)||\pi(\cdot|\phi_\theta(s_t)))$.

At last z_t^* is used to update network parameters.

References

[1] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[2] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

[3] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

发布于 2019-04-05

强化学习 (Reinforcement Learning)

▲ 赞同 17 ▼ 8 条评论 分享 喜欢 收藏 ···

文章被以下专栏收录

强化学习前沿
读呀读paper
进入专栏