	Definition of c_s	Estimation variance	Guaranteed convergence†
ling	$\frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)}$	High	for any π, μ
	λ	Low	only for π close to μ
	$\lambda\pi(a_s\|x_s)$	Low	for any π, μ
	$\lambda \min(1, \frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)})$	Low	for any π, μ

RL Algorithm: Retrace

starimpact
计算视觉

春节假期没啥意思，加了几天班，在老丈人家继续追了AlphaStar相关的技术（效率极低，后来演变成每天看一部电影了~流浪地球不错~绝对国产第一部硬科幻电影），然后就看到了这个DeepMind在2016年出的一篇论文。个人觉得这个论文很有用，显然2017年DeepMind出的V-trace算法和它也有不少渊源，甚至让人联想到计算 $TD(\lambda)$ 用到的Eligibility Trace算法。

$Retrace(\lambda)$ 出自论文：Safe and efficient off-policy reinforcement learning。

它有三个方面的特性：

(1)低方差[low variance]：这个意思是说对总Reward的评估方差会比较低。

(2)安全性[safety]：对于off-policy的方法来说，如果行动策略和目标策略相差太大，那么它能确保在训练目标策略时仍可以安全使用行动策略采集的样本。[是不是很开心^_^]

(3)高效性[efficiency]：有时候收集样本的行动策略是很接近目标策略的，那它能确保对这种样本的高效利用。有些方法是有了安全性，但是缺失了这种情况下的样本利用的高效性。

$Retrace(\lambda)$ 是一种return-based off-policy方法。

啥叫“return-based”？这个“return”即，一条轨迹上（或时间轴上）所有衰减奖励的和： $\sum_t \gamma^t r_t$ 。这个定义的重点还是在一段轨迹或一段时间，而不是一般的一个点（即t的长度是1）。

----------[2.10]我是可爱的分割线----------

下面给出return-based off-policy算法的通用表达式(4)：

$$RQ(x,a) := Q(x,a) + E_\mu[\sum_{t\geq 0} \gamma^t (\Pi_{s=1}^t c_s)(r_t + \gamma E_\pi Q(x_{t+1},\cdot) - Q(x_t,a_t))], (4)$$

其中，等式左边的 R 被称作operator，它对 Q 进行作用，形成了等式右边的部分。它实际上是指return-based off-policy operator。大家不要把 RQ 理解成 $R \times Q$ 。

π 是target policy或evaluation policy。

μ 是behavior policy或action policy。

$E_\pi Q(x,\cdot) := \sum_a \pi(a|x) Q(x,a)$ ，这里的 E_π 意思是在 π 策略分布下求期望（或加权平均值）。

c_s 是非负的系数，当 $t=0$ 时有 $(\Pi_{s=1}^{t} c_s)=1$ 。 c_s 在这里又被称为trace of the operator，比较重要，在下面会详细说明它的不同表达式的作用，最终引入主角 $Retrace(\lambda)$ ：

Importance sampling (IS): $c_s=\pi(a_s|x_s)/\mu(a_s|x_s)$ 。重要性采样表达式的直接应用。这是修正 π 和 μ 差异的最简单方法（off-policy correction）。它通过在时间轴上（或称path上）的连积来纠正该path的采样概率。但是这方法有个问题，有可能会出现大的variance，原因就在于 $\Pi_{s=1}^{t} c_s$ 这个在时间轴上的连积有可能很大也有可能很小。

Off-policy $Q^{\pi}(\lambda)$ **and** $Q^{*}(\lambda)$ **：** $c_s=\lambda$ 。这是另外一种off-policy correction的方法。由于 λ 是一个稳定的数值，所以不会出现IS中的那种连积后有可能会很大的情况。但是一方面这个数值比较难定，要有一定的实际经验；另一方面这种方法并不能保证对任意的 π 和 μ 安全，实际会比较适用于 π 和 μ 区别不大的时候。

Tree-backup $TB(\lambda)$ **：** $c_s=\lambda\pi(a_s|x_s)$ 。这里用上了target policy π 。这下是安全了，但是对于两种策略相近时（称为near on-policy）的样本利用效率下降了。因为它同样会将一些比较远的轨迹切掉。而在near on-policy的情况下，通常是不希望这样去做的。

$Retrace(\lambda)$ **：** $c_s=\lambda min(1,\pi(a_s|x_s)/\mu(a_s|x_s))$ 。这下好了，不仅在 π 和 μ 相差比较大时安全了，而且在它俩比较接近时也不会出现切掉较远轨迹的问题。由于最大值是1，所以也不会出现 $\Pi_{s=1}^{t} c_s$ 数值很大的情况。可谓是上面三个方法的优点的集大成者，而且还避开了它们的缺点。

下表是一个比较好的总结：

	Definition of c_s	Estimation variance	Guaranteed convergence†	Use full returns (near on-policy)
Importance sampling	$\frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)}$	High	for any π, μ	yes
Q(λ)	λ	Low	only for π close to μ	yes
TB(λ)	$\lambda\pi(a_s\|x_s)$	Low	for any π, μ	no
Retrace(λ)	$\lambda \min(1, \frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)})$	Low	for any π, μ	yes

Table 1: Properties of several algorithms defined in terms of the general operator given in (4).
†Guaranteed convergence of the expected operator $\mathcal{R}$.

----------[2.16]我是可爱的分割线----------

一周工作结束了，终于可以继续了。

原文中有一段对 $Retrace(\lambda)$ 的收敛性的分析，这里就不说了。这次继续说的是它的在线更新算法（Online Algorithm）。这里给出的更新公式与上面提到的 RQ 有不太一样的地方：

$$Q_{k+1}(x,a) \leftarrow Q_k(x,a)+\alpha_k \sum_{t\geq 0} \delta_t^{x_k} \sum_{j=0}^{t} \gamma^{t-j}\left(\prod_{i=j+1}^{t} c_i\right) 1\{x_j,a_j=x,a\}$$

$$\delta_t^{x_k}=r_t+\gamma E_{\pi_k} Q_k(x_t+1,\cdot)-Q_k(x_k,a_t)$$

式中的 k 表示第 k 条轨迹。

上面的有关 Q 的更新式被称为 $every\ visit$ 形式：对于有限状态和有限动作条件下这样做是有意义的，因为会有可能发生同一状态和动作组合在不同的时刻上，上式的 $1\{x_j,a_j=x,a\}$ 符号即是二元操作符（只有符合条件时才是1，否则就是0），作用是判断是否碰到了相同的状态动作组合 (x,a) 。但是，如果在连续状态和连续动作空间下，在一个轨迹上出现相同的状态动作组合的概率几乎为0，那么，此时 $first\ view$ 会比较适用，即一种情况只会出现一次。

将上式变成 *first view* 的形式：

$$Q_{k+1}(x,a) \leftarrow Q_k(x,a) + \alpha_k \sum_{t \geq s} \delta_t^{\pi_k} \gamma^{t-s} \left(\prod_{i=s+1}^{t} c_i \right)$$

由于式中的 s 是指当前起始时刻，可以设置为0，于是上式就和 $Retrace(\lambda)$ 基本一样了。

有了 Q 之后， π 或 μ 就可以确定出来。我认为类似于DQN的方法去确定即可。

但是，DQN是在离散动作空间用 Q 来确定策略的，对于连续动作空间怎么办了？有人会想到DDPG。但是DDPG的actor输出的是确定性策略，无法给出当前动作的概率信息。

不知道大家有没有学过VAE(Variational AutoEncoder)，里面的Encoder部分是输出了 μ 和 $variance$ 值，但是从这个分布中采样再去Decoder是不可导的。为了解决这个问题，VAE采用了reparameterization（重参数化）的方法，具体可以去看这个论文了。重参数化这个方法还被用在了SAC里（Soft Actor-Critic）。

嗯……大家懂我的意思了吧。要解决连续动作空间的问题， $Retrace(\lambda)$ 中就要有 $actor$ 网络来做类似VAE中的事情。

再有，对于连续动作空间下， c_i 的表达式也是要变的，即：

$$c_i = min(1, d\pi/d\mu)$$

这里要用概率密度，说白了就是将 μ 和 $variance$ 代入 $gaussian\ function$ 就行啦。

再补一个实验结果图：瞬间被结果感动，有没有！

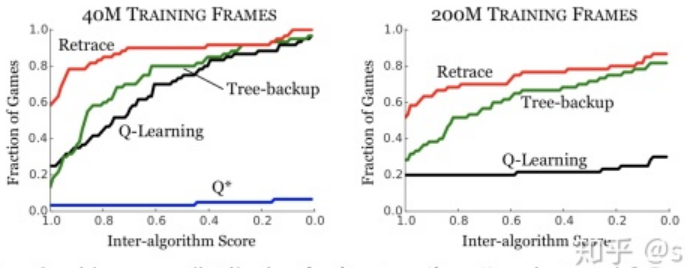

Figure 1: Inter-algorithm score distribution for λ-return ($\lambda = 1$) variants and Q-Learning ($\lambda = 0$).

图中线越高效果越好，这些是60种Atari 2600游戏的实验对比结果。Retrace的方法相对于原始的Q-Learning领先非常明显。

----------我是可爱的分割线----------

结语：这篇论文理论证明非常多，看得我是眼花花~要想深入理解，还是需要花一些工夫的，我也没看太多里面的证明部分，对我来说确实更有难度，在这方面还是希望有大神交流一下。

迹(Trace)是一个神奇的东西，有了它，善加利用，RL就更高效和安全了~

至此，此文完毕。

突然想到：年前DeepMind声称要在今晚（2.15）来一场AlphaStar和星际2职业选手的在线对决，是摄像机视角（非全局视角）真正对决。很期待最终的结果^_^。

编辑于 2019-02-17

强化学习 (Reinforcement Learning)

文章被以下专栏收录

强化学习知识大讲堂
让机器人学会思考

进入专栏

强化学习前沿
读呀读paper

进入专栏