Distral: Robust Multitask Reinforcement Learning

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, Razvan Pascanu
DeepMind, London, UK
{ywteh,vbapst,lejlot,johnquan,kirkpatrick,raia,heess,razp}@google.com

【强化学习算法 32】Distral

张楚珩
清华大学 交叉信息院博士在读

中Distral是**Dis**till & **tra**nsfer **l**earning的缩写。

原文传送门

Teh, Yee, et al. "Distral: Robust multitask reinforcement learning." Advances in Neural Information Processing Systems. 2017.

特色

提出了一种同时在多个任务上训练的强化学习方法，主要的想法是把各个任务上学到的策略进行提纯（distill，本意是蒸馏）得到一个共有的策略，然后再使用这个共有的策略去指导各个特定任务上的策略进行更好的学习。文章称，这种多任务的强化学习方法避免了不同任务产生互斥的梯度，反而干扰学习；同时，也避免了各个任务学习进度不一致，导致某个任务的学习主导了整体的学习。个人感觉，这种各个任务间提纯的方法也说不定能用到多智能体间的相互交互上。

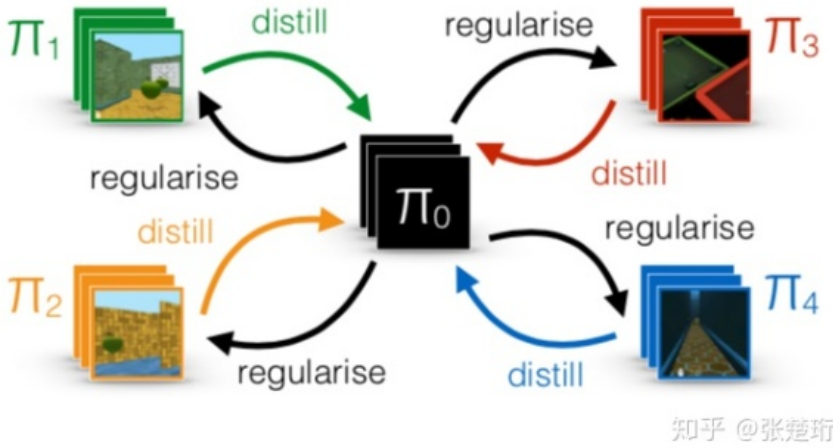
背景

为什么要做多任务学习？需要过多的交互（采样）是目前强化学习的一大重要问题，这阻碍了强化学习应用到模拟环境以外的其他地方。如果能通过进行多任务学习让智能体学习到一些共有的知识，这样在一个新环境下就能通过少量样本学习到好的策略了，这样就相当于从另一个角度降低了学习算法所需要的样本。

过程

1. 大致思想

在各个环境各自学习各自策略的基础上建立一个中间的策略 π_0 ，各个策略进行学习的时候会在中间策略的正则下来学习，而各个不同的策略综合起来由提纯得到这个中间策略。



2. 目标函数

考虑多个任务 $\pi_i = (S, A, p_i(s'|s,a), \gamma, R_i(a,s))$ （这里的 π_i 既表示任务，也表示任务下相应的策略）和由这些策略提纯得到的策略 π_0 。

设定如下的最大化目标

$$J(\pi_0, \{\pi_i\}_{i=1}^n) = \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t R_i(a_t,s_t) - c_{\mathrm{KL}}\gamma^t \log\frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} - c_{\mathrm{Ent}}\gamma^t \log\pi_i(a_t|s_t)\right]$$
$$= \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t R_i(a_t,s_t) + \frac{\gamma^t\alpha}{\beta}\log\pi_0(a_t|s_t) - \frac{\gamma^t}{\beta}\log\pi_i(a_t|s_t)\right]$$

即主要约束了各个策略 π_i 不要偏离中心策略 π_0 太远，同时再加上了entropy项以鼓励探索。

3. 优化方式

文章讨论了两种优化方式，一种是联合优化，一种是交替优化。前一种就是每次都对所有的策略 $\{\pi_i\}, \pi_0$ 的参数化表示做SGD；后一种就是每次固定一个训练另一个，即固定 π_0 优化 π_i ，再固定 π_i 优化 π_0 。在后一种情况下，第一步可以使用和已有的soft Q-learning一样的实现，第二步可以使用和已有的一些distillation方法已有的实现，实践上这两者已经是稳定的了。

固定 π_0 的时候，我们可以定义一个正则化的奖励

$$R_i'(a,s) := R_i(a,s) + \tfrac{\alpha}{\beta}\log\pi_0(a|s)$$

这样问题就变成了一个附加entropy项的单任务强化学习问题了，使用Soft Q-learning的框架（不熟悉的可以参考本专栏的文章【强化学习算法 10】SQL）就是学习这样的任务（红色删除线应为 s_{t+1}，文章打印错误）

$$V_i(s_t) = \frac{1}{\beta}\log\sum_{a_t}\pi_0^{\alpha}(a_t|s_t)\exp\left[\beta Q_i(a_t,s_t)\right]$$
$$Q_i(a_t,s_t) = R_i(a_t,s_t) + \gamma\sum_{s_t} p_i(s_{t+1}|s_t,a_t)V_i(s_{t+1})$$

它相当于是以 π_0^{α} 为先验来学习的，这是一个比 π_0 更为鼓励探索的一个先验， $0\le\alpha\le 1$ 的作用后面会再提到。相应的Boltzmann策略是

$$\pi_i(a_t|s_t) = \pi_0^{\alpha}(a_t|s_t)e^{\beta Q_i(a_t|s_t)-\beta V_i(s_t)} = \pi_0^{\alpha}(a_t|s_t)e^{\beta A_i(a_t|s_t)}$$

固定 π_i 的时候，目标函数里面只有一项与 π_0 相关

$$\frac{\alpha}{\beta}\sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\ge 0}\gamma^t\log\pi_0(a_t|s_t)\right]$$

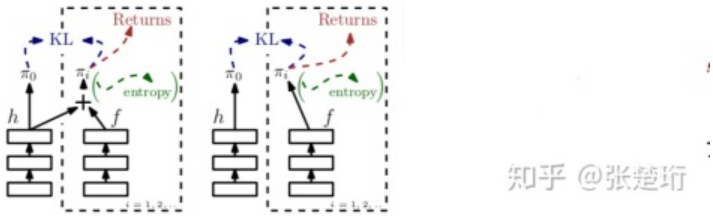
可以使用maximum likelihood estimator或者对目标函数做SGD得到，这就是一个distillation的过程。

4. 策略的表示

一种自然的策略表示方法是都采用Boltzmann策略的表示方法

$$\hat{\pi}\ (a_t|s_t) = \frac{\exp(h_\theta\ (a_t|s_t)}{\sum_{a'}\exp(h_\theta\ (a'|s_t))}$$

不过文中还提出了另一种更好的方法，即各个特定的策略 π_i 表示为与 π_0 共有的部分和其自己特有部分的加和，这样各个特定策略学习过程中就只需要集中精力学习自己特有的部分了。



左边是文中提到的更好的一种表示方式，右边是各自表示各自的

中心策略 π_0 使用一个神经网络 $h_{\theta_0}(a|s)$ 来表示

$$\hat{\pi}_0(a_t|s_t) = \frac{\exp(h_{\theta_0}(a_t|s_t)}{\sum_{a'}\exp(h_{\theta_0}(a'|s_t))}$$

特定的策略 π_i 使用使用中心策略的神经网络 $h_{\theta_0}(a|s)$ 和各自的神经网络 $f_{\theta_i}(a|s)$ 来表示

$$\hat{\pi}_i(a_t|s_t) = \hat{\pi}_0^{\alpha}(a_t|s_t)\exp(\beta\hat{A}_i(a_t|s_t)) = \frac{\exp(\alpha h_{\theta_0}(a_t|s_t) + \beta f_{\theta_i}(a_t|s_t))}{\sum_{a'}\exp((\alpha h_{\theta_0}(a'|s_t) + \beta f_{\theta_i}(a'|s_t))}$$

注意到这就是一个以 π_0^{α} 为先验的Boltzmann策略，其中advantage使用的是soft advantage

$$\hat{A}_i(a_t|s_t) = f_{\theta_i}(a_t|s_t) - \frac{1}{\beta}\log\sum_a \hat{\pi}_0^{\alpha}(a|s_t)\exp(\beta f_{\theta_i}(a|s_t))$$

在这种表示方法下就可以自然使用策略梯度方法进行联合优化

$$\begin{aligned}\nabla_{\theta_i}J =& \mathbb{E}_{\hat{\pi}_i}\left[\left(\sum_{t\geq 1}\nabla_{\theta_i}\log\hat{\pi}_i(a_t|s_t)\right)\left(\sum_{u\geq 1}\gamma^u(R_i^{\text{reg}}(a_u,s_u))\right)\right] \\ =& \mathbb{E}_{\hat{\pi}_i}\left[\sum_{t\geq 1}\nabla_{\theta_i}\log\hat{\pi}_i(a_t|s_t)\left(\sum_{u\geq t}\gamma^u(R_i^{\text{reg}}(a_u,s_u))\right)\right]\end{aligned}$$

$$\begin{aligned}\nabla_{\theta_0}J =& \sum_i \mathbb{E}_{\hat{\pi}_i}\left[\sum_{t\geq 1}\nabla_{\theta_0}\log\hat{\pi}_i(a_t|s_t)\left(\sum_{u\geq t}\gamma^u(R_i^{\text{reg}}(a_u,s_u)\right)\right] \\ &+ \frac{\alpha}{\beta}\sum_i \mathbb{E}_{\hat{\pi}_i}\left[\sum_{t\geq 1}\gamma^t\sum_{a'_t}(\hat{\pi}_i(a'_t|s_t) - \hat{\pi}_0(a'_t|s_t))\nabla_{\theta_0}h_{\theta_0}(a'_t|s_t)\right]\end{aligned}$$

其中正则化的奖励为

$$R_i^{\text{reg}}(a,s) = R_i(a,s) + \tfrac{\alpha}{\beta}\log\hat{\pi}_0(a|s) - \tfrac{1}{\beta}\log\hat{\pi}_i(a|s)$$

5. α 的选择

下面考虑优化目标里面 α 不同数值带来的不同含义

$$\begin{aligned}J(\pi_0,\{\pi_i\}_{i=1}^n) =& \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t R_i(a_t,s_t) - c_{\text{KL}}\gamma^t\log\frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} - c_{\text{Ent}}\gamma^t\log\pi_i(a_t|s_t)\right] \\ =& \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t R_i(a_t,s_t) + \frac{\gamma^t\alpha}{\beta}\log\pi_0(a_t|s_t) - \frac{\gamma^t}{\beta}\log\pi_i(a_t|s_t)\right]\end{aligned}$$

- α 的时候就等于没有中心策略，即每个任务各学各的；
- α 的时候相当于在最大化cumulated discounted return的同时，还需要最小化一个 $KL(\pi_i||\pi_0)$ 项；当 $\pi_i = \pi_0$ 的时候， KL 项为零，这时候相当于 π_i 在找一个在这个任务上的greedy策略；
- $0 < \alpha < 1$ 的时候在最小化KL项的同时，还要最小化 $\log \pi_0(a_t|s_t)$ ，这相当于附加了激励各个策略 π_i 不要局限在中心策略附加，鼓励了相对于中心策略的探索。

实验

根据前面提到的目标的不同（选择不同的 α ）、优化方式的不同（是分别优化还是联合优化）以及结构的不同（ π_i 的表示是否使用中心策略），文章做了以下的7个实验组。

	$h_{\theta_0}(a\|s)$	$f_{\theta_i}(a\|s)$	$\alpha h_{\theta_0}(a\|s) + \beta f_{\theta_i}(a\|s)$
$\alpha = 0$	A3C multitask	A3C	A3C 2col
$\alpha = 1$		KL 1col	KL 2col
$0 < \alpha < 1$		KL+ent 1col	KL+ent 2col

Table 1: The seven different algorithms evaluated in our experiments. Each column describes a different architecture, with the column headings indicating the logits for the task policies. The rows define the relative amount of KL vs entropy regularization loss, with the first row comprising the A3C baselines (no KL loss).

根据目标、优化方法、参数化策略结构不同产生的7个实验组算法

实验主要在一个简单的世界任务和3D的第一人称迷宫任务上做的。个人感觉有如下几个点

- 多个任务联合起来学习可以使得单个任务的渐进性能都稍微好一点点，这一点说明中心策略 π_0 确实还是代表了一些共有的知识。（不然，搞了这么一通不如单独每个任务去训练）
- 格子世界任务选择了一个具有长长走廊的格子世界，中心策略的主要优势就体现在这个走廊上，实验结果表明，使用了中心策略之后，智能体能够快速地朝一个方向通过这个走廊，以此证明这种算法不会产生相互冲突的中心策略更新；但是个人认为其主要原因是这里的state选择并不仅仅是所处的格子，还包括了上一步的行动，正是这样状态空间的选择导致了不会产生相互冲突的中心策略。
- 就相对于超参数的稳定性来说，KL+ent 2col算法（即使用 $\alpha = 0.5$ 和相互耦合的策略表示）具有最好的稳定性。
- 如果学习相对独立的策略，还是不要使用相互耦合的策略表示（即1col）更好。

发布于 2018-11-28

强化学习 (Reinforcement Learning)