Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans Jonathan Ho Xi Chen Szymon Sidor Ilya Sutskever
OpenAI

【强化学习算法 9】ES

张楚珩
清华大学 交叉信息院博士在读

4 人赞同了该文章

ES 指的是 evolutionary strategy，本来是一个很宽泛的概念，但是代表一种强化学习算法的时候主要指的是这篇文章的工作。

原文传送门：Salimans, Tim, et al. "Evolution strategies as a scalable alternative to reinforcement learning." arXiv preprint arXiv:1703.03864 (2017).

特色：当做黑盒子去用演化算法优化，甚至能够优化神经网络表示的策略；最后能达到state-of-the-art的效果。

分类：Model-free、***Derivative-free***、Continuous State Space、Continuous Action Space（Discrete效果更好）、Support High-dim Input

过程：和ARS类似，都是在参数空间上做小扰动，不过ARS是做的mirrored sampling，即正负成对的扰动，而这里就直接是一个Guassian球做扰动了。可以求得参数关于目标函数的梯度

$$\nabla_\theta \mathbb{E}_{\epsilon\sim N(0,I)}\, F(\theta+\sigma\epsilon) = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim N(0,I)}\,\{F(\theta+\sigma\epsilon)\,\epsilon\}$$

最后形成的算法也很简单，就是不停的朝着目标函数更大的方向更新策略参数就可以了

Algorithm 1 Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha\frac{1}{n\sigma}\sum_{i=1}^{n} F_i\epsilon_i$
end for

知乎 @张楚珩

当把ES方法做并行运算的时候，也有相应的算法。

Algorithm 2 Parallelized Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
Initialize: n workers with known random seeds, and initial parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  for each worker $i = 1, \ldots, n$ **do**
    Sample $\epsilon_i \sim \mathcal{N}(0, I)$
    Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$
  end for
  Send all scalar returns F_i from each worker to every other worker
  for each worker $i = 1, \ldots, n$ **do**
    Reconstruct all perturbations ϵ_j for $j = 1, \ldots, n$ using known random seeds
    Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$
  end for
end for

知乎 @张楚珩

主要用到如下技术：

1. Common random numbers：master和worker之间的通讯内容主要是rollout得到的return $F(\cdot)$ （标量）和网络的扰动。（矢量），高维度的矢量通讯会降低并行运算的效率，因此这里使用了common random numbers技术，实现生成高斯分布向量表存储在共享内存里面，这样只需要通讯这个向量表里面的index（标量）就可以得知相应的扰动了。

2. Virtual batch normalization：做法就是把输入的states按照训练之前固定下来的统计量记性归一化处理。之所以这么做是因为，在ES里面网络的权重和扰动都是服从均匀的高斯分布的，归一化处理之后，结果对于参数变化更敏感，算法的效果更好。

3. Discretize the actions：把输出的动作离散化，使得产生的动作相对于参数变化更不平滑，这样有利于探索到更多不同类别的行动。反过来想，如果参数一直到产生的return都是光滑的，再加上本身采样的variance比较高，得到的结果可能是根据小扰动采样得到的return们都差不多，然后算法也没法知道该往哪边走了。

文章还通过一系列很简明的数学，比划着说明了ES相对于policy gradient等算法的优势：

1. 方便进行并行运算；

2. 对于长回合、奖励延迟、奖励稀疏等RL里面比较棘手的问题都能很好地应对；

3. 不用算梯度，因此每回合运算量更小；与此同时梯度运算中会面临的一些梯度爆炸等问题不会出现（同时参考ACER为了避免importance ratio爆炸做的努力）；为了梯度不爆炸，需要计算硬件精度较高，这样就不好用TPU什么的，但是ES就可以；

4. 在策略里面可以使用不可导的元素，而且甚至此类元素能使得ES发挥地更好（增加了non-smoothness）；

另外：

文中提到一个很有意思的事情。我们一般参数化一个产生行动策略 $a(\theta)$ ，目标是优化通过找到好的 θ ，优化此策略下的每回合总收益 $F(\theta) = R(a(\theta))$ 。从参数 θ 到最后的的目标 $F(\theta)$ 中间有很多不平滑的

成分，比如离散的action space、不连续的reward机制、环境的非线性等。要想更新参数，一般都希望找个方向（梯度），这样的不平滑的函数就不好找梯度。因此，我们之前所做的都可以看做把这个函数给弄得光滑。

Policy gradient可以看做是在状态空间上做smoothing，要求action是一个stochastic的，比如categorical distribution或者Gaussian distribution，这样随着参数 θ 的变化，action distribution的参数光滑变化，action的概率光滑变化，然后形成的总收益 $F(\theta)$ 也光滑变化。即， $F_{PG}(\theta)=\mathbb{E}_{\epsilon}R(a(\epsilon,\theta))$ 。这样就可以对其求导，有policy gradient theorem。

$$\nabla_\theta F_{PG}(\theta)=\mathbb{E}_{\epsilon}\left\{R(\mathbf{a}(\epsilon,\theta))\nabla_\theta\log p(\mathbf{a}(\epsilon,\theta);\theta)\right\}.$$

Deterministic policy gradient更是可以看做是对于reward机制smoothness直接做了假设，认为Q函数的可导。

这里的ES则是在参数空间上面做了smoothing， $F_{ES}(\theta)=\mathbb{E}_{\xi}R(a(\xi,\theta))$ 。

$$\nabla_\theta F_{ES}(\theta)=\mathbb{E}_{\xi}\left\{R(\mathbf{a}(\xi,\theta))\nabla_\theta\log p(\tilde{\theta}(\xi,\theta);\theta)\right\}.$$

如何理解 $\nabla_\theta\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[F(\theta+\sigma\epsilon)]=\frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[F(\theta+\sigma\epsilon)\epsilon]$ ？

尝试了严格的证明，没有很证明出来，如果有证明出来的欢迎私信。不过有一个观察是 $\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[F(\theta)\epsilon]=0$ ，因此有

$$\begin{aligned}R.H.S.&=\frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[(F(\theta+\sigma\epsilon)-F(\theta))\epsilon]\\&\approx\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\epsilon(\nabla_\theta F(\theta+\sigma\epsilon)\cdot\epsilon)]\\&\approx\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\epsilon(g\cdot\epsilon)]\\&\propto\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[g]\\&\approx L.H.S\end{aligned}$$

倒数第二步是由对称性得到的。

由 $F_{ES}(\theta)=\mathbb{E}_{\xi}R(a(\xi,\theta))$ ，到 $\nabla_\theta F_{ES}(\theta)=\mathbb{E}_{\xi}[R(a(\xi,\theta))\nabla_\theta\log p(\tilde{\theta}(\xi,\theta);\theta)]$ ？

$$\begin{aligned}\nabla_\theta F_{ES}(\theta)&=\nabla_\theta\mathbb{E}_{\xi}R(a(\tilde{\theta}(\xi,\theta)))\\&=\sum_{\xi}p(\xi)\sum_{a}\nabla_\theta p(a=a(\tilde{\theta}(\xi,\theta)))R(a)\\&=\sum_{\xi}p(\xi)\sum_{a}p(a=a(\tilde{\theta}(\xi,\theta)))\nabla_\theta p(a=a(\tilde{\theta}(\xi,\theta)))R(a)/p(a=a(\tilde{\theta}(\xi,\theta)))\\&=\mathbb{E}_{\xi}[\nabla_\theta\log p(a=a(\tilde{\theta}(\xi,\theta)))R(a(\tilde{\theta}(\xi,\theta)))]\\&=\mathbb{E}_{\xi}[\nabla_\theta\log p(\theta=\tilde{\theta}(\xi,\theta))R(a(\tilde{\theta}(\xi,\theta)))]\end{aligned}$$

如果不做最后一步就证明了 $F_{PG}(\theta)$ 那个式子。

编辑于 2018-10-01

强化学习 (Reinforcement Learning) 机器学习 算法

▲ 赞同 4 ▼ 添加评论 分享 喜欢 收藏 ···

文章被以下专栏收录

强化学习前沿

进入专栏