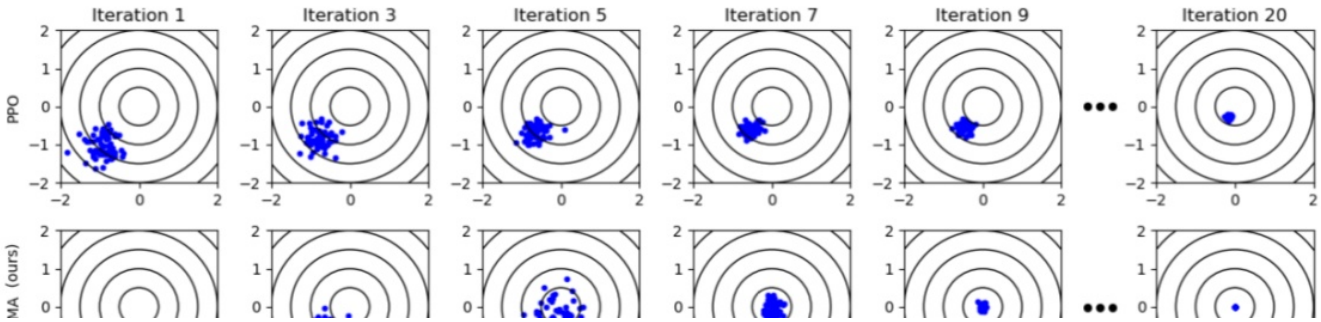
Perttu Hämäläinen Aalto University
Amin Babadi Aalto University
Xiaoxiao Ma Aalto University
Jaakko Lehtinen Aalto University and NVIDIA
{perttu.hamalainen,amin.babadi,xiaoxiao.ma,jaakko.lehtinen}@aalto.fi



【强化学习算法 36】PPO-CMA

张楚珩
清华大学 交叉信息院博士在读

PPO-CMA算法是一个把PPO和CMA-ES算法相结合的一个model-free强化学习算法。

原文传送门

Hämäläinen, Perttu, et al. "PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation." arXiv preprint arXiv:1810.02541 (2018).（under review ICLR 2019）

特色

把零阶优化算法（不使用导数的优化算法）CMA-ES的思想放到PPO的优化上面，并且**声称**PPO算法里面advantage为负的样本会导致其不稳定，去掉advantage为负的样本又会导致其过早收敛到suboptimal的位置，而这里提出的PPO-CMA算法可以避免这个问题，在Mujoco任务上面取得了较好的效果。

过程

1. 高斯策略下的PPO算法

PPO算法的优化目标如下，其中样本 (s_i, a_i) 都是使用当前策略产生的（on-policy），每一步使用这些on-policy的样本来更新下一步的策略。

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{M} A^{\pi}(\mathbf{s}_i, \mathbf{a}_i) \log \pi_{\theta}(\mathbf{a}_i|\mathbf{s}_i),$$

对于连续控制问题，文献中常用的是使用高斯策略（Gaussian policy），策略神经网络输出动作的均值 $\mu_\theta(s)$ 和协方差矩阵 $C_\theta(s)$ ，然后根据输出的均值-方差进行采样得到最后的行动。但是在实际情况下，协方差矩阵输出来比较复杂，一般都会认为各个维度不相关，输出的是方差 $c_\theta(s) = diag(C_\theta(s))$ ；或者设定为一个不可学习的（人为控制的）各向同性高斯方差 $C = I$ 。

在前一种情况下，损失函数可以写为

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{M} A^{\pi}(\mathbf{s}_i, \mathbf{a}_i) \sum_j [(a_{i,j} - \mu_{j;\theta}(\mathbf{s}_i))^2 / c_{j;\theta}(\mathbf{s}_i) + 0.5 \log c_{j;\theta}(\mathbf{s}_i)],$$

在后一种情况下，损失函数变为

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{M} A^{\pi}(\mathbf{s}_i, \mathbf{a}_i) ||\mathbf{a}_i - \mu_{\theta}(\mathbf{s})||^2,$$

对此可以进行另一种理解，即每次训练策略去拟合采样到加权的行动分布，所加的权重为 A^{π} 。前一种情况下使用的是Bayesian loss，后一种情况下使用的是L2 loss。

2. PPO存在的问题

由于强化学习的时间序列特性，每次采样的样本 (s_i, a_i) 中的状态都各不相同，每个相应的行动是从策略网络的输出中采样得到的，而策略网络的输入每次都不一样，对于仔细研究产生了很大的不便。文中构造了一个stateless的情形来说明PPO的更新过程，即构造和状态无关的奖励 $r(a) = -a^T a$ ，每次状态输入都为一个常数状态。

通过这样的例子得到两个观察

- 如果更新的时候同时使用advantage为正和advantage为负的样本的话（下图第一行的情形），advantage为正的样本相当于在告诉策略网络说“这片区域很好，输出的均值应该往这边靠一点”，advantage为负的样本相当于在告诉策略网络说“这片区域很垃圾，别往这边来”。通过这样一推一拉，下一回合策略网络输出的均值就会往advantage为正的方向走。**文章称**这样的效果可能产生类似超调的效果，导致不稳定，即从第一行第一幅图到第一行第二幅图，输出的均值变得离最优点更远了。
- 如果只使用advantage为正的样本，可能会导致过早的收敛。原因文章没有说，个人认为可以这样理解，考虑所有的样本都在损失函数的“坡”上（比如第二行第一幅图的情况）。由于得到的较好的样本只是在这一批样本里面相对较好，并不是全局较好，这些较好的样本里面有大比例的样本（68.3%）在一倍标准差之内，而在一倍标准差之内的好样本会导致方差的缩小，因此产生了过早的收敛。

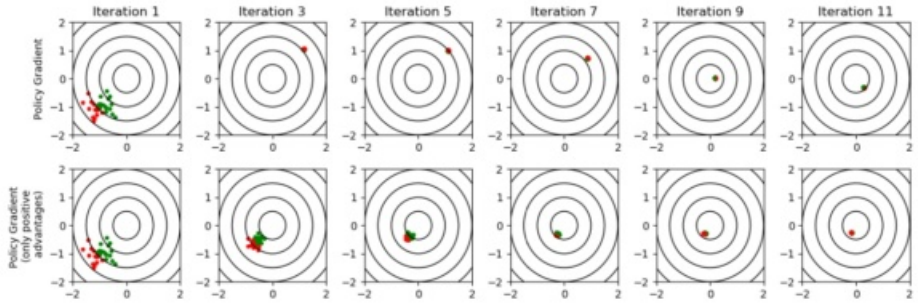


Figure 2: Comparing policy gradient variants in the didactic example of Figure 1, when doing multiple minibatch updates with the data from each iteration. Actions with positive advantage estimates are shown in green, and negative advantages in red. Top: Basic policy gradient is highly unstable. Bottom: Using only positive advantage actions, policy gradient is stable but converges prematurely.

图：第一行是使用所有样本进行更新，第二行仅使用advantage为正的样本进行更新；红色的点是每回合产生的较差一半样本，绿色的点是每回合产生的较好一半样本

3. 提出的算法

文章提出了以下PPO-CMA算法

Algorithm 3 PPO-CMA

for iteration=1,2,... **do**
  while iteration simulation budget N not exceeded **do**
    Reset the simulation to a (random) initial state
    Run agent on policy π_θ for T timesteps or until a terminal state
  end while
  Train critic network for K epochs using the experience from the current iteration
  Estimate advantages A^π using GAE (Schulman et al. (2015b))
  Clip negative advantages to zero, $A^\pi \leftarrow \max(A^\pi, 0)$
  Train policy variance for K epochs using experience from past H iterations and Eq. 6
  Train policy mean for K epochs using the experience from this iteration and Eq. 6
end for

有两点看似不经心，但是有比较深的考虑

- 注意到这里是分别更新方差和均值的，并且是先更新方差再更新均值。先要注意到更新一个的时候另一个是固定的，因此从损失函数可以看到方差的更新实际上不依赖均值的数值，而均值的更新依赖于方差。之前有研究表明，先更新方差再更新均值有一种动量的效果，能够更快找到最优的点。下图是一个很好的说明

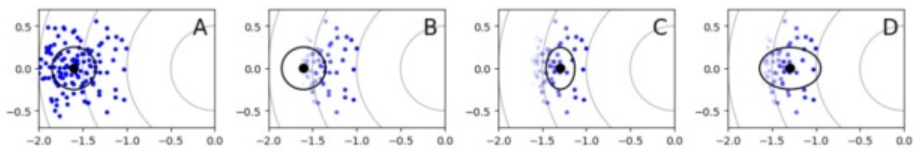


Figure 4: The difference between joint and separate updating of mean and covariance, denoted by the black dot and ellipse. A) sampling, B) pruning and weighting of samples based on fitness, C) EMNA-style update, i.e., estimating mean and covariance based on weighted samples, D) CMA-ES update, where covariance is estimated before updating the mean.

图：A表示采样到的点，B表示去掉advantage为负的点之后的加权情况（颜色越深权重越大），C表示同时更新方差和均值，D表示先更新方差后更新均值

- 方差的更新不止用了这一回合完全on-policy的样本，还是用了近期H次迭代产生的准on-policy样本，这里借鉴了evolution path的想法，保证了方差估计能在最近几回合的平均演化方向上更大一些，仍然相当于一个动量的效果。

实验结果

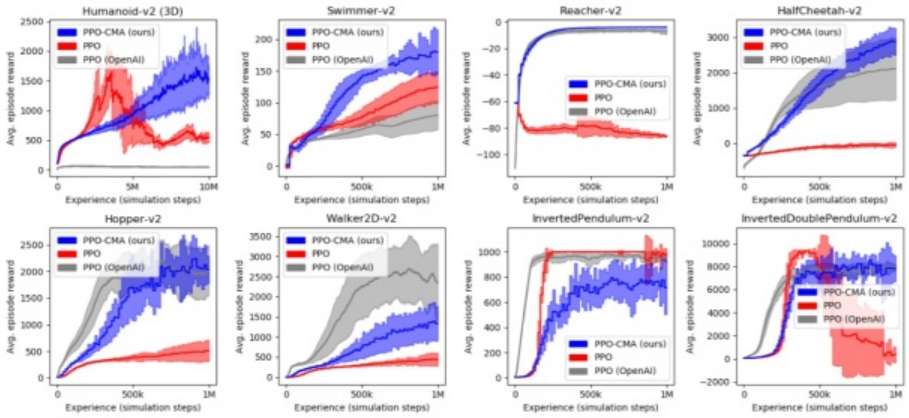


Figure 5: Means and standard deviations of training curves in OpenAI Gym MuJoCo tasks. The humanoid results are from 5 runs and others from 10 runs. Red denotes our vanilla PPO implementation with the same hyperparameters as PPO-CMA, providing a controlled comparison of the effect of algorithm changes. Gray denotes OpenAI's baseline PPO implementation using their default hyperparameters and training scripts for these MuJoCo tasks.

更新：这篇paper投ICLR被拒了。实验上，他们调的别人的算法可能步长太大，会导致性能突然drop（比如PPO在Humanoid上）；他们的算法其实和CMA扯不上太大的关系。

编辑于 2019-05-18

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