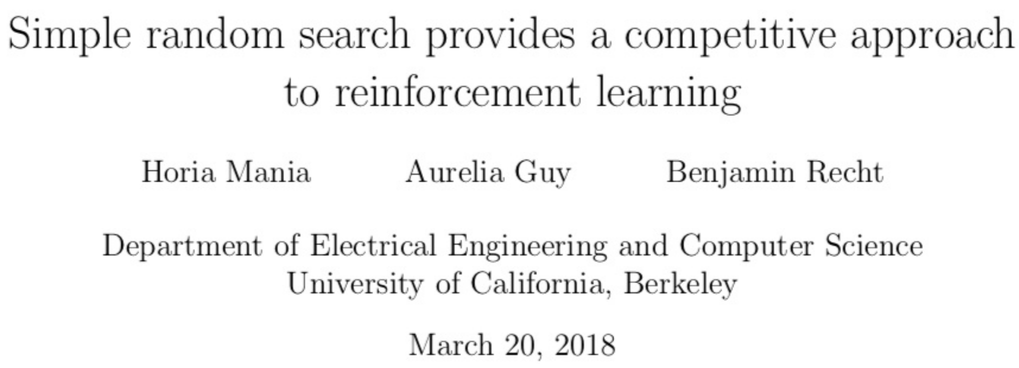
Simple random search provides a competitive approach to reinforcement learning

Horia Mania Aurelia Guy Benjamin Recht

Department of Electrical Engineering and Computer Science
University of California, Berkeley

March 20, 2018

【强化学习算法 8】ARS

张楚珩
清华大学 交叉信息院博士在读

ARS 指的是 augmented random search。

原文传送门：

Mania, Horia, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning." arXiv preprint arXiv:1803.07055 (2018).

特色：这篇文章类似前面说的CEM算法，都在说用最简单的线性策略和derivative-free的方法就可以吊打各种高级的value Iteration/policy gradient方法。并且文章提出sample complexity不应该是做了一堆调参，然后把超参数定好，看用多少样本能够训出来一个好的模型；而是sample complexity应该算上整个调参过程中所用到的所有样本。

分类：Model-free、***Derivative-free***、Continuous State Space、Continuous Action Space、Not Support High-dim Input（线性策略）、Deterministic Policy

过程：

1. 仍然是一个线性的策略，策略是 $\pi(x) = Mx$ ；
2. 主要的过程是一个random search，即每次在当前策略参数M的基础上做微小的正负扰动 $M \pm \nu\delta_k$ ，形成策略 $\pi_{k,\pm}(x) = (M \pm \nu\delta_k)x$ ，然后通过rollouts看扰动的效果如何，做 $M \leftarrow M + \alpha[r(\pi_{k,+}) - r(\pi_{k,-})]\delta_k$ 的更新。
3. 改进一：对于得到的状态进行mean-std filter，得到的策略是 $\pi_{k,\pm}(x) = (M \pm \nu\delta_k) diag(\Sigma)^{-1/2}(x-\mu)$ ，其中均值方差是之前遇到的所有状态的均值和方差；
4. 改进二：更新的时候对于步长做自适应调整，除以得到returns的方差，这样如果不确定性高的时候，步长就比较小，更新更保守；
5. 改进三：对于正负扰动都不能得到很好结果的扰动就将其舍弃，只取 $\max\{r(\pi_{k,+}), r(\pi_{k,-})\}$ 最大的前几个进行更新。

算法：

Algorithm 2 Augmented Random Search (ARS): four versions **V1**, **V1-t**, **V2** and **V2-t**

Hyperparameters: step-size α, number of directions sampled per iteration N, standard deviation of the exploration noise ν, number of top-performing directions to use b ($b < N$ is allowed only for **V1-t** and **V2-t**)

Initialize: $M_0 = \mathbf{0} \in \mathbb{R}^{p \times n}$, $\mu_0 = \mathbf{0} \in \mathbb{R}^n$, and $\Sigma_0 = \mathbf{I}_n \in \mathbb{R}^{n \times n}$, $j = 0$.

while ending condition not satisfied **do**

Sample $\delta_1, \delta_2, \ldots, \delta_N$ in $\mathbb{R}^{p \times n}$ with i.i.d. standard normal entries.

Collect $2N$ rollouts of horizon H and their corresponding rewards using the $2N$ policies

$$\textbf{V1:} \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)x \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)x \end{cases}$$

$$\textbf{V2:} \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)\,\mathrm{diag}\,(\Sigma_j)^{-1/2}\,(x - \mu_j) \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)\,\mathrm{diag}(\Sigma_j)^{-1/2}(x - \mu_j) \end{cases}$$

for $k \in \{1, 2, \ldots, N\}$.

Sort the directions δ_k by $\max\{r(\pi_{j,k,+}), r(\pi_{j,k,-})\}$, denote by $\delta_{(k)}$ the k-th largest direction, and by $\pi_{j,(k),+}$ and $\pi_{j,(k),-}$ the corresponding policies.

Make the update step:

$$M_{j+1} = M_j + \tfrac{\alpha}{b\sigma_R} \sum_{k=1}^{b} \left[r(\pi_{j,(k),+}) - r(\pi_{j,(k),-}) \right] \delta_{(k)},$$

where σ_R is the standard deviation of the $2b$ rewards used in the update step.

V2 : Set μ_{j+1}, Σ_{j+1} to be the mean and covariance of the $2NH(j+1)$ states encountered from the start of training.[2]

$j \leftarrow j + 1$

end while