Modelling Policies in MDPs in Reproducing Kernel Hilbert Space

Guy Lever
University College London
London, UK
g.lever@cs.ucl.ac.uk

Ronnie Stafford
University College London
London, UK
ronnie.stafford@gmail.com

【强化学习 78】RKHS-AC

张楚珩
清华大学 交叉信息院博士在读

RKHS-AC 是 RKHS actor-critic framework 的简称，不是这篇文章自己叫的官方名字，是后续文章引用的时候给安的名字。

原文传送门

Lever, Guy, and Ronnie Stafford. "Modelling policies in mdps in reproducing kernel hilbert space." Artificial Intelligence and Statistics. 2015.

Reproducing kernel Hilbert space (Lecture notes from Purdue)

特色

在很多工作里面策略的表示是参数化的表示，学习的目标是学习策略表示的参数。这里考虑一个 non-parametric 的模型，它有如下好处：它具有较好的表示能力，其策略梯度能够被简单地表示出来，通过一些方法能够让非参数化的表示比较 compact。更重要的是，non-parametric 的模型能够避免 incremental learning 带来的 sample inefficiency。同时，这篇文章更多得强调说，文章介绍的这种方法自然地在策略空间上平滑地更新，而不像其他方法一样实质上是在参数空间上更新，从而需要做 remetrization 把策略空间的距离度量转化到参数空间上（参考 TRPO 所做的事情）。

过程

1. RKHS

Reproducing kernel Hilbert space（RKHS）的相关知识，见前一篇：张楚珩：【数学】RKHS。

2. 在 RKHS 中表示一个策略

考虑一个 stochastic 的高斯策略

$$\pi_{h,\mathbf{\Sigma}}(a|s) := \frac{1}{Z} e^{-\frac{1}{2}(h(s)-a)^\top \mathbf{\Sigma}^{-1}(h(s)-a)}, \qquad (6)$$

待优化的是一个确定性的函数 $h \in \mathcal{H}, h: \mathcal{S} \to \mathcal{A} \subseteq \mathbb{R}^m$ ，该函数在 RKHS 中，因此可以表示为

$$h(\cdot) = \sum_i K(s_i, \cdot)\alpha_i \in \mathcal{H}_K, \tag{7}$$

其中 $s_i \in \mathcal{S}, \alpha_i \in \mathcal{A}$ 。

要学习这个策略，就是在学习 h 和 Σ ；对于 h 来说，可以把 $[K(s_1,\cdot), K(s_2,\cdot), \cdots]$ 看做基底，该基底可以使用所遇到样本对应的核函数来表示，而系数 α_i 就是需要学习的参数，同时也需要维护相应的基底。在学习过程中，核函数 $K: \mathcal{S} \times \mathcal{S} \to \mathcal{L}(\mathcal{A})$ 是给定的，其中 $\mathcal{L}(\mathcal{A})$ 表示 $\mathcal{A}$ 上的线性算子，核函数一般还会包含一些刻画 kernel 平滑程度相关的参数（比如 bandwidth），在算法里面这些参数作为超参数，不学习。

3. RKHS 中的策略梯度

策略梯度定理中涉及到下面这一项

$$\log \pi_{h,\mathbf{\Sigma}}(a|s) = -\log Z - \frac{1}{2}(h(s) - a)^\top \mathbf{\Sigma}^{-1}(h(s) - a),$$

它对 h 的导数也应该是一个函数，那么其导数是什么呢？

Claim 2.1. *The derivative of the map $f : \mathcal{H}_K \to \mathbb{R}$, $f : h \mapsto \log \pi_{h,\mathbf{\Sigma}}(a|s)$, at h, is the bounded linear map $Df|_h : \mathcal{H}_K \to \mathbb{R}$ defined by,*

$$\begin{aligned} Df|_h : g &\mapsto (a - h(s))^\top \mathbf{\Sigma}^{-1} g(s) \\ &= \langle K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)), g\rangle_K. \end{aligned} \tag{8}$$

Thus the direction of steepest ascent is the function,

$$\nabla_h(\log \pi_{h,\mathbf{\Sigma}}(a|s)) = K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)) \in \mathcal{H}_K. \tag{9}$$

可以证明上面写出来的这种形式是 log-probability 的 Frechet derivative。

$$
\begin{aligned}
f(h+g) &= \log \pi_{h+g,\mathbf{\Sigma}} \\
&= -\log Z - \frac{1}{2}(h(s)+g(s)-a)^\top \mathbf{\Sigma}^{-1}(h(s)+g(s)-a),
\end{aligned}
$$

so with $Df|_h(g)$ defined as in (8) we have that

$$
\begin{aligned}
\frac{|f(h+g)-f(h)-Df|_h(g)|}{||g||_K} &= \frac{g(s)^\top \mathbf{\Sigma}^{-1} g(s)}{2||g||_K} \\
&= \frac{\langle g, K(s,\cdot)\mathbf{\Sigma}^{-1} g(s)\rangle_K}{2||g||_K} \\
&\leq \frac{||g||_K (\mathbf{\Sigma}^{-1} g(s))^\top K(s,s)\mathbf{\Sigma}^{-1} g(s)}{2||g||_K} \\
&= (\mathbf{\Sigma}^{-1} g(s))^\top K(s,s)\mathbf{\Sigma}^{-1} g(s)/2 \\
&\to 0, \qquad (10)
\end{aligned}
$$

其中 (10) 式的第一行到第二行利用了 reproducing property，第二行到第三行利用 Cauchy–Schwarz 不等式。由此可以得到策略梯度的期望

$$
\nabla_h U(\pi_h) = \int \rho_h(z) Q^{\pi_h}(z) K(s,\cdot)\mathbf{\Sigma}^{-1}(a-h(s))\mathrm{d}z. \qquad (11)
$$

可以得到策略梯度在样本上的估计

$$
\begin{aligned}
&(1-\gamma)\nabla_h U(\pi_h) \\
&\approx \sum_{k=1}^{T} Q^{\pi_h}(z_k) K(s_k,\cdot)\mathbf{\Sigma}^{-1}(a_k - h(s_k)) \in \mathcal{H}_K. \qquad (12)
\end{aligned}
$$

样本 (s_k, a_k) 都是执行当前策略采样到的。由此，策略梯度可以看做是基函数 $K(s_k, \cdot)$ 的线性组合。做梯度上升的时候，只需要找到策略和策略梯度的基函数的并集，然后把相同位置上的基函数做加减即可。

可以看出，随着所遇到的状态越来越多，这个基函数族会越来越大，这样计算也就会越来越慢，由此在每一次做过梯度上升之后，我们需要对策略 $h(\cdot) = \sum_{i=1}^{N} K(s_i, \cdot)\alpha_i$ 做 sparsification，使得该策略能够尽可能等效地用更少的基函数表示出来，即 $\hat{h}(\cdot) = \sum_{i=1}^{n} K(s_i, \cdot)\alpha_i$，$n \ll N$ 。

Sparsification 可以通过 kernel matching pursuit 算法得到，它是每次从之前的基函数族里面找出一个能够最大程度减少误差 $\sum_{i \in [N]} \|h(s_i) - \hat{h}(s_i)\|_A^2$ 的基函数，然后加入到被挑选出来的基函数集合里面。 n 的大小可以不固定，而是通过设定最大允许的误差来间接确定 n （即，希望在动作空间上精确到什么程度），这相当于是一个自适应的办法，能够解决无法很好事先确定特征向量维度的问题。

4. Compatible function approximation

回顾 deterministic policy gradient（DPG）[1] 里面讨论的关于 compatible 的问题。compatible 是针对一个函数拟合的 critic 而言的。我们称这样的一个 critic 是 compatible 的：如果这个 critic 在其能够拟合的范围内准确拟合了相应的价值函数（比如 Q^π ），那么用这个拟合的价值函数去替换策略梯度里面的价值函数，得到的仍然是无偏的策略梯度。即，用来拟合价值函数的函数族，需要和用来拟合策略的函数族兼容。

针对前面提出来的策略梯度，文章给出了与其相兼容的价值函数拟合。Q 函数定义在 state-action space，由此先定义一个定义在该空间上的标量核函数。

$$
\begin{aligned}
&K_h : (\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A}) \to \mathbb{R}, \\
&K_h((s,a),(s',a')) \\
&:= \left(K(s,s')\boldsymbol{\Sigma}^{-1}(a - h(s))\right)^\top \boldsymbol{\Sigma}^{-1}(a' - h(s')), \quad (13)
\end{aligned}
$$

由此可以得到相应的 feature map

$$
\phi : (s,a) \mapsto K(s,\cdot)\boldsymbol{\Sigma}^{-1}(a - h(s)) \in \mathcal{H}_K =: \mathcal{F}_\phi, \quad (14)
$$

一个 Q 函数可以被写作

$$
\begin{aligned}
q(s,a) &= \langle w, \phi(s,a) \rangle_{\mathcal{F}_\phi} \\
&= \langle w, K(s,\cdot)\boldsymbol{\Sigma}^{-1}(a - h(s)) \rangle_{\mathcal{H}_K}. \quad (15)
\end{aligned}
$$

其中 w 是需要学习的函数，它仍然可以用基底+系数的形式表示出来。

下面的定理说明了如下定义的价值函数是一个 compatible function。

Theorem 2.2. *Suppose that* $\hat{Q}^{\pi_h} \in \mathcal{H}_K^h$ *is such that,*

$$\hat{Q}^{\pi_h} = \underset{Q \in \mathcal{H}_K^h}{\operatorname{argmin}} \int \rho_h(z)(Q(z) - Q^{\pi_h}(z))^2 \mathrm{d}z. \quad (16)$$

Then $\hat{Q}^{\pi_h}$ *is a compatible function approximator for* Q^{π_h}*, i.e.*

$$\nabla_h U(\pi_h) = \int \rho_h(z)\hat{Q}^{\pi_h}(z)K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s))\mathrm{d}z. \quad (17)$$

证明它只需要注意到，当取到最优拟合的 Q 函数的时候，最小化的目标对 w 的导数需要等于零。其证明过程和 DPG 文章以及 Sutton 书里面几乎一样。

Proof. From (15) there exists a $w_h \in \mathcal{H}_K$ such that,

$$\hat{Q}^{\pi_h}(s,a) = \langle w_h, K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s))\rangle_{\mathcal{H}_K}. \quad (18)$$

Therefore $\nabla_{w_h}\hat{Q}^{\pi_h}(s,a) = K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)) \in \mathcal{H}_K$. And now from (16),

$$\begin{aligned} 0 &= \int \rho_h(z)(\hat{Q}^{\pi_h}(z) - Q^{\pi_h}(z))\nabla_{w_h}\hat{Q}^{\pi_h}(z)\mathrm{d}z \\ &= \int \rho_h(z)(\hat{Q}^{\pi_h}(z) - Q^{\pi_h}(z))K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s))\mathrm{d}z \end{aligned}$$

which implies (17).

在实际计算的时候， w 可以通过 matching pursuit 算法回归得到。

5. 算法

总体算法流程图如下图所示。

Algorithm 1 Compatible RKHS Actor-Critic

Input: MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, P\}$; kernel $K : \mathcal{S} \times \mathcal{S} \to \mathcal{L}(\mathcal{A})$; initial covariance $\mathbf{\Sigma}_1$
Initialize: $h_1 = 0$
Parameters: tolerance δ
for $j = 1, 2, \ldots$ **do**
  Collect data from the system using policy π_{h_j}
  Compatible Q approximation: Fit $Q^{\pi_{h_j}}$ with $\hat{Q}^{\pi_{h_j}}$ in $\mathcal{H}_K^h$ as in Section 2.3
  Compute policy gradient: Approximate the policy gradient $\nabla_h U(\pi_{h_j, \mathbf{\Sigma}_j})$ using (12)
  Update policy: $h_{j+1} = h_j + \eta_j \nabla_{h_j} U(\pi_{h_j, \mathbf{\Sigma}_j})$ for step size η_j
  Sparsify policy: Sparsify $h_{j+1} \in \mathcal{H}_K$ with tolerance δ as in Section 2.2
  Update $\mathbf{\Sigma}$: reduce $\mathbf{\Sigma}_j$ (or take a gradient step w.r.t. $\mathbf{\Sigma}_j$)
end for

其实验部分效果不是很好，大概这是这篇文章只发到 AISTAT 上的原因？

该做法和 NEC 做法的异同

- NEC 是 value-based，这里是 policy-based（actor-critic）；
- NEC 里面的 memory 中储存 $_{(s,a)}$ 和相应的 $_{R_t}$ （文章写的是 n-step bootstrapped Q value，由于实际上 n 比较大，我更愿意把它理解为 Monte Carlo return）；而这里存储的是一些列具有代表性的 $_s$ / $_{(s,a)}$ 对应的核函数和相应的加权系数；

- NEC 里面的 memory 会随着采样而增加，存储的 R_t 会按照 Q-learning 的公式产生的梯度更新；这里的基函数在做策略梯度时会增加，但是又通过 sparsification 的操作使得基函数维持在一个比较固定的水平，其系数按照 policy gradient 和 regression（matching pursuit 算法）来更新；
- 对于 extrapolation 来说：NEC 会找最近的点，并把它们的价值函数值的平均作为改点的价值函数；而这里所有的点都通过加权的 kernel 算得，对于 extraplation 来说，得到的数值会偏向于隐含的 prior 数值（比如零）。这一点是我瞎猜的。

参考文献

[1] Silver, David, et al. "Deterministic policy gradient algorithms." ICML. 2014.

发布于 2019-06-29

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

进入专栏