多步强化学习算法的收敛性分析*

杨 瑞
（天津大学数学学院 天津 300072）

摘 要 在强化学习(Reinforcement Learning)算法理论中，最近有学者提出了一个新的估值算法 $Q(\sigma)$，这里 σ 是采样度(degree of sampling)，这是一个介于全采样(full-sampling)和非采样(no-sampling)的新方法。$Q(\sigma)$ 统一了 Sarsa 和 Expected Sarsa 等传统算法，但是 $Q(\sigma)$ 的提出者只在实验上验证了算法的有效性。该文对 $Q(\sigma)$ 的收敛性作了理论分析，证明了在一定条件下 $Q(\sigma)$ 是收敛的。

关键词 强化学习；值函数估计；优化；时间差分
中图分类号 TP301.6 **DOI**:10.3969/j.issn.1672-9722.2019.07.005

Convergence Analysis of Multistep Reinforcement Learning Algorithm

YANG Rui
(School of Mathematics, Tianjin University, Tianjin 300072)

Abstract Recently, a new algorithm called $Q(\sigma)$ has been presented to evalued value function in the theory of reinforcement learning algorithm, where σ is the degree of sampling. $Q(\sigma)$ is a new method between full-sampling and no-sampling and it unifies Sarsa and Expected Sarsa. However, the original paper only tests the performance of $Q(\sigma)$ on experiments. This paper gives a theoretical analysis of $Q(\sigma)$. It gives a proof that under some conditions, $Q(\sigma)$ can converge to the value functions.

Key Words reinforcement learning, value function estimate, optimization, temporal difference
Class Number TP301.6

1 引言

强化学习(Reinforcement Learning)[1]是解决这样一个问题的机器学习分支：智能体(Agent)与环境交互，如何自动地学习到最佳策略以使自身获得最大回报(Rewards)。最近 AlphaGo[2]击败了人类顶尖围棋职业选手，其核心技术之一就是强化学习。强化学习在现代人工智能学中有着举足轻重的地位，有着广泛的应用前景[3-4]。

强化学习是一种不告诉 Agent 如何去正确地决策，而是让 Agent 自己去探索环境，在与环境交互的过程获得知识，不断优化策略。Agent 与环境交互的过程如下：

1) Agent 感知当前环境状态 s；

2) 针对当前环境状态 s，Agent 根据当前行为策略选择一个动作 a；

3) 当 Agent 选择 a，环境由状态 s 转移到当前状态 s'，并给出奖赏值(Rewards) R；

4) 环境把 R 反馈给 Agent。

如此循环此过程，直至终止状态，在这个过程中，并没有告诉 Agent 如何采取动作，而是 Agent 根据环境的反馈自己发现的。

1.1 马尔科夫决策过程(MDPs)

马尔科夫决策过程(MDPs)是强化学习的基本框架[5]，可以由四元组< S，A，P，R>来表达。S 是系统的有限状态集，A 是 Agent 的有限动作空间，动作用来控制系统的状态转移，$P_{ss'}^{a}:S\times A\times S'\to[0,1]$ 为 Agent 执行动作 a 之后，系统由状态 s 转向 s' 的概率。$R_{ss'}^{a}:S'\times A\times S'\to 1R$ 表示当 Agent 执行动作 a，系统由状态 s 转到状态 s' 后，系统反馈给

* 收稿日期：2019年1月14日，修回日期：2019年2月27日
作者简介：杨瑞，女，硕士研究生，研究方向：应用数学。

Agent的rewards。

策略(policy)定义了Agent在给定状态下的行为方式，策略决定了Agent的动作。$\pi:S\times A\rightarrow[0,1]$，$\pi(s,a)$表示在状态$s$下，执行动作$a$的概率。

在MDP中，还定义了两种值函数(Value Function)：状态值函数$V^{\pi}(s)$(State Value Function)和状态-动作值函数$Q^{\pi}(s,a)$(State-act Value Function)。

$V^{\pi}(s)$表示Agent从状态s开始，根据策略π所获得的期望总回报(Rewards)：

$$V^{\pi}(s)=E_{\pi}[\sum_{k=0}^{\infty}\gamma^{k}R_{t+k+1}|s_t=s] \tag{1}$$

类似地：

$$Q^{\pi}(s,a)=E_{\pi}[\sum_{k=0}^{\infty}\gamma^{k}R_{t+k+1}|s_t=s,a_t=a] \tag{2}$$

值函数的确定了从一个状态出发，按照π所能获得的期望总回报。

1.2 Bellman 方程

由于MDP中，状态转移满足Markov性，1957年，Bellman证明了他的著名方程[6]：

$$V^{\pi}(s)=\sum_{a}\pi(s,a)\sum_{s'}P_{ss'}^{a}[R_s^a+\gamma V^{\pi}(s')] \tag{3}$$

类似地：

$$Q^{\pi}(s,a)=\sum_{s'}P_{ss'}^{a}(R_s^a+\gamma\sum_{a'}\pi(s',a')Q^{\pi}(s',a')) \tag{4}$$

由此可知：系统的状态转移矩阵和回报均已知，则易求出$V^{\pi}(s)$和$Q^{\pi}(s,a)$

强化学习的目标是导出最优策略π^*：

$$V^*(s)=\max_{\pi}V^{\pi}(s)$$
$$Q^*(s,a)=\max_{\pi}Q^{\pi}(s,a)$$
$$\pi^*(s)=\arg\max_{a\in A}Q^*(s,a) \tag{5}$$

1.3 模型未知的强化学习

MDPs为强化学习提供了统一的框架，但实际问题中有如下几个问题：

1)“维数灾难”：状态空间超级巨大，要学习的参数随着状态空间的维数指数级爆炸。

2)收敛速度慢：很多强化学习算法的收敛性分析都要依赖“任意状态都要被无数次访问到”这样的前提条件。

3)很多实际问题，我们是无法获得系统的状态转移概率和回报函数的，对这种情况建立模型的算法称为模型未知的强化学习算法。

最近，有研究人员提出$Q(\sigma)$算法来估计$Q^{\pi}(s,a)$，并且原文作者通过实验发现收敛效果很好。本文将对$Q(\sigma)$的收敛性给予一个证明。

2 时间差分学习算法框架

时间差分算法(Temporal Difference)是强化学习最核心的一类算法，这项工作首次是在Suton 1988[7]年提出的。这类算法不需要知道系统的状态转移矩阵，能够直接学习。假设Agent在与环境交互中产生的一个轨道为

$$s_0,a_0,r_1,s_1,a_1\cdots s_{T-1},a_{T-1},s_T$$

其中s_T是终止状态。

2.1 Sarsa算法

Sarsa[8]算法是根据上述轨迹来迭代Bellman方法的解，其名称是由Agent学习的数据结构而来的，数据是由一个(s_t,a_t)转移到下一个(s_{t+1},a_{t+1})的五元组$(s_t,a_t,r_{t+1},s_{t+1},a_{t+1})$

Sarsa估值计算公式为

$$Q_{t+1}(s_t,a_t)\leftarrow Q_t(s_t,a_t)+\alpha\delta^{S}$$
$$\delta^{S}=r_{t+1}+\gamma Q_t(s_{t+1},a_{t+1})-Q_t(s_t,a_t) \tag{6}$$

α为学习率，δ称为Sarsa算法的时间差分。

2.2 Expected Sarsa算法

Expected Sarsa[9]：

$$Q_{t+1}(s_t,a_t)\leftarrow Q_t(s_t,a_t)+\alpha\delta^{ES}$$
$$\delta^{ES}=r_{t+1}+\gamma\sum_{a'}\pi(s_{t+1},a')Q_t(s_{t+1},a')-Q_t(s_t,a_t) \tag{7}$$

由Sarsa和Expected Sarsa的迭代形式可知，Sarsa是一个全采样的算法，即在t时刻，只用$r_{t+1}+\gamma Q_t(s_{t+1},a_{t+1})$来作为目标(target)进行估值。而Expected sarsa是采用下一状态s_{t+1}的Q值的期望：$r_{t+1}+\gamma\sum_{a'}\pi(s_{t+1},a')Q_t(s_{t+1},a')$来估值。根据Bellman等式(4)，从直觉上讲，Expected Sarsa的估值要稳定一些[10]。首次证明了Expected Sarsa和Sarsa有相同的bias，但是Expected Sarsa的方差更小一些。

2.3 $Q(\sigma)$ 算法

$Q(\sigma)$[11]算法的设计思想是：引入了参数σ，σ是采样的程度，如果$\sigma=0$，表示no-sampling，$\sigma=1$表示full-sampling。

单步$Q(\sigma)$算法的迭代形式：

$$G_t^{(1)}=r_{t+1}+\gamma[\sigma Q_t(s_{t+1},a_{t+1})+(1-\sigma)\overline{Q_{t+1}}] \tag{8}$$

这里$\overline{Q_{t+1}}=\sum_{a'}\pi(s_{t+1},a')Q_t(s_t,a')$。

$$Q_{t+1}(s_t,a_t)=Q_t(s_t,a_t)+\alpha\left[G_t^{(1)}-Q_t(s_t,a_t)\right]=Q_t(s_t,a_t)+\alpha\delta_t^{\sigma} \tag{9}$$

这里 $\delta_t^\sigma = r_{t+1} + \gamma\left[\sigma Q_t(s_{t+1}, a_{t+1}) + (1-\sigma)\overline{Q_{t+1}}\right] - Q_t(s_t, a_t)$。

$G_t^{(1)}$ 是 Q 函数的估值,此估值可以看成是Sarsa估值和Expected Sarsa估值的凸组合。

$$G_t^{(1)} = \sigma[r_{t+1} + Q_t(s_{t+1}, a_{t+1}) + (1-\sigma)[r_{t+1} + \overline{Q_{t+1}}] \tag{10}$$

并且 δ_t^σ 也是 δ^S 和 δ^{ES} 的凸组合:

$$\delta_t^\sigma = \sigma\left[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right] + (1-\sigma)\left[r_{t+1} + \gamma\sum_{a'}\pi(s_{t+1}, a')Q_t(s_{t+1}, a') - Q_t(s_t, a_t)\right] = \sigma\delta^S + (1-\sigma)\delta^{ES}$$

2.4 多步时间差分算法

Sarsa, Expected Sarsa, $Q(\sigma)$ [12~14]均可以推广至多步的情况。

2.4.1 多步Sarsa

多步Sarsa值函数的估计公式:

$$\hat{Q}_t^{(n)} = \sum_{i=0}^{n-1}\gamma^i r_{t+1+i} + \gamma^n\sum_a \pi(s_{t+n}, a)\hat{Q}_{t+n-1}(s_{t+n}, a)$$

多步Sarsa值函数的更新公式:

$$\hat{Q}_{t+n}(s_t, a_t) = \hat{Q}_{t+n-1}(s_t, a_t) + \alpha[\hat{Q}_t^{(n)} - \hat{Q}_{t+n-1}(s_t, a_t)] \tag{11}$$

2.4.2 多步Expected Sarsa算法,也称多步Tree Backup算法

多步Expected Sarsa也是类似的,但有一些差别。

多步Expected Sarsa[11]值函数的估计公式:

$$\hat{Q}_t^{(n)} = r_{t+1} + \gamma\sum_{a\neq a_{t+1}}\pi(s_{t+1}, a)Q_t(s_{t+1}, a) + \gamma\pi(s_{t+1}, a_{t+1})\hat{Q}_{t+1}^{(n)} = Q_{t-1}(s_t, a_t) + \sum_{k=t}^{\min(t+n-1, T-1)}\delta_k^{ES}\prod_{i=t+1}^{k}\gamma\pi(s_i, a_i)$$

多步Expected Sarsa值函数的更新公式:

$$\hat{Q}_{t+n}(s_t, a_t) = \hat{Q}_{t+n-1}(s_t, a_t) + \alpha[\hat{Q}_t^{(n)} - \hat{Q}_{t+n-1}(s_t, a_t)]$$

2.4.3 多步 $Q(\sigma)$

多步 $Q(\sigma)$ 值函数的估计公式:

$$\hat{Q}_t^{(n)} = r_{t+1} + \gamma\sum_{a'}\pi(s_{t+1}, a')\hat{Q}_{t-1}^{(n)}(s_{t+1}, a') = \hat{Q}_{t-1}(s_t, a_t) + \sum_{k=t}^{\min(t+n-1, T-1)}\delta_k^\sigma\prod_{i=t+1}^{k}\gamma[(1-\sigma_i)\pi(s_i, a_i) + \sigma_i]$$

多步 $Q(\sigma)$ 值函数的更新公式:

$$\hat{Q}_{t+n}(s_t, a_t) = \hat{Q}_{t+n-1}(s_t, a_t) + \alpha[\hat{Q}_t^{(n)} - \hat{Q}_{t+n-1}(s_t, a_t)]$$

3 $Q(\sigma)$ 算法的收敛性分析

引理1[15~16]:假设一个随机过程 (ζ_t, Δ_t, F_t),$\zeta_t, \Delta_t, F_t: X \to R$ 满足方程:

$$\Delta_{t+1}(x_t) = (1-\zeta_t(x_t))\Delta_t(x_t) + \zeta_t(x_t)F_t(x_t),$$

这里 $x_t \epsilon X, t = 0, 1, 2, \cdots$ 假设 P_t 是 σ -fields的递增序列,ζ_0, Δ_0 是 P_0 -measurable,ζ_t, Δ_t 以及 F_{t-1} 是 P_t -measurable, $t \geq 1$。假定以下条件成立:

1) X 是有限集;

2) $\zeta_t(x_t) \in [0, 1], \sum_t \zeta_t(x_t) = \infty, \sum_t (\zeta_t(x_t))^2 < \infty$ $w.p.1$ 且 $\forall x \neq x_t: \zeta_t(x) = 0$;

3) $/\!/ E\{F_t|P_t\} /\!/ \leq \kappa /\!/ \Delta_t /\!/ + c_t$, $\kappa \in [0, 1)$, c_t 依概率收敛于0;

4) $\mathrm{Var}\{F_t(x_t)|P_t\} \leq K(1 + \kappa /\!/ \Delta_t /\!/)^2$, K 是常数。

其中 $/\!/ \cdot /\!/$ 表示最大模;则 Δ_t 依概率收敛到0。

定理1:在MDP框架下,对于任意的初始化 $Q(s, a)$, $\forall \gamma \in (0, 1)$。

$$\forall (s, a) \in S' \times A$$

Q 的更新按以下方式:

$$\hat{Q}_{k+1}^{(n+1)}(s, a) = r_{t+1} + \gamma\sum_a \pi(s_{t+1}, a)\hat{Q}_k^{(n)}(s_{t+1}, a)$$

令 $\Delta_n = E\left[\hat{Q}_{k+1}^{(n)}(s, a)\right] - Q^\pi(s, a)$,则有 $\|\Delta_{n+1}\| \leq \gamma\|\Delta_n\|$,$\Delta_n$ 按最大值范数是压缩序列,即 Δ_n 依概率收敛于0。

证明:由数学归纳法证明之。

For n=1:

$$\max_{(s,a)}\left|E\left[\hat{Q}_{k+1}^{(1)}(s, a)\right] - Q^\pi(s, a)\right| = \max_{(s,a)}\left|\begin{matrix} R_s^a + \gamma\sum_{(s',a')}p_{ss'}^a\pi(s', a')\hat{Q}_{k+1}^{(1)}(s', a') - R_s^a - \\ \gamma\sum_{(s',a')}p_{ss'}^a\pi(s', a')Q^\pi(s', a') \end{matrix}\right| \leq \gamma\max_{(s,a)}|\hat{Q}_{k+1}^{(1)}(s, a) - Q^\pi(s', a')|$$

假设对 n 也成立:

$$\max_{(s,a)}|E\left[\hat{Q}_{k+1}^{(n)}(s, a)\right] - Q^\pi(s, a)| \leq \gamma\max_{(s,a)}|\hat{Q}_{k+1}^{(n)}(s', a') - Q^\pi(s', a')|$$

以下证明对于 $\hat{Q}_{k+1}^{(n+1)}(s, a)$ 同样成立:

$$\hat{Q}_{k+1}^{(n+1)}(s, a) = R_{t+1} + \gamma\sum_{a'\in A}\pi(s_{t+1}, a')\left(\hat{Q}_k(s_{t+1}, a')\right)(1 - I(a', a_{t+1}) + I(a', a_{t+1})\hat{Q}_k^n(s_{t+1}, a')(s_{t+1}, a')$$

这里 $I(a', a_{t+1})$ 是一个指示函数:

$$I(a', a_{t+1}) = \begin{cases} 1 & if\ a' = a_{t+1} \\ 0 & if\ 其它 \end{cases}$$

$$\max_{(s,a)}\left|E\left[\hat{Q}_{k+1}^{(n+1)}(s, a)\right] - Q^\pi(s, a)\right| =$$

$$\max_{(s,a)}\left|\begin{array}{l}R_s^a+\gamma\sum_{s'}p_{ss'}^a\sum_{a'}\pi(s',a')\left\{E\left[\left(1-I(a',a_{t+1})\right)\hat{Q}_k^{(n)}(s',a')\right]\right\}\\+I(a',a_{t+1})\hat{Q}_k^{(n)}(s',a')-R_s^a-\gamma\sum_{s'}p_{ss'}^a\sum_{a'}\pi(s',a')Q^\pi(s',a')\end{array}\right|$$

$$=\gamma\max_{(s,a)}\left|\sum_{s'}p_{ss'}^a\sum_{a'}\pi(s',a')\left\{E\left[\left(1-I(a',a_{t+1})\right)(\hat{Q}_k^{(n)}(s',a')-Q^\pi(s',a'))\right]\right\}\right.$$

$$\left.+I(a',a_{t+1})\left(\hat{Q}_k^{(n)}(s',a')-Q^\pi(s',a')\right)\right|$$

$$\le\gamma\max_{(s,a)}\left|\hat{Q}_k^{(n)}(s,a)-Q^\pi(s,a)\right|$$

定理 2：在 MDP 框架下，对于任意的初始 $Q(s,a)$，$\forall\gamma\in(0,1)$。

$$\forall(s,a)\in S'\times A$$

Q 的更新按式(11)更新，则 $\hat{Q}_t^{(n)}(s,a)$ 按概率收敛于 $Q^\pi(s,a)$。

事实上，如果定理 1 中的 $\pi(s_{t+1},a')$ 满足：

$$\pi(s_{t+1},a')=\begin{cases}1 & a'=a_{t+1}\\0 & \text{其它}\end{cases}$$

则这就是式(11)所表示的算法，也即是定理 1 中算法的特殊情况，由定理 1 的收敛性可以得到该算法也是收敛的。

定理 3：如果 $Q(\sigma)$ 算法满足条件：

1)状态空间是有限集；

2) $\sum_{i=1}^{\infty}\alpha_i=\infty$，$\sum_{i=1}^{\infty}\alpha_i^2<+\infty$。

则 $Q(\sigma)$ 迭代产生的 $Q_t(s,a)$ 依概率收敛于 $Q^\pi(s,a)$。

定理 3 的证明：

$Q(\sigma)$ 是定理 1 和定理 2 所述算法的凸组合，易知 $Q(\sigma)$ 迭代产生的 $Q_t(s,a)$ 依概率收敛于 $Q^\pi(s,a)$。

4 结语

本文以时间差分学习算法为主线，系统简要地介绍了强化学习中的几个重要估值算法，并对最近提出的一个新的时间差分学习算法 $Q(\sigma)$ 算法作了理论分析，同时给出了 $Q(\sigma)$ 算法的收敛性证明，这在强化学习理论研究中具有重要意义。

参考文献

[1] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction [M]. London, England: MIT Press, Cambridge, 1998.

[2] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.

[3] PILARSKI P M, DAWSON M R, DEGRIS T, et al. Adaptive artificial limbs: a real-time approach to prediction and anticipation [J]. Robotics & Automation Magazine IEEE, 2013, 20(1): 53-64.

[4] PARKER P, ENGLEHART K, HUDGINS B. Myoelectric signal processing for control of powered limb prostheses. [J]. Journal of Electromyography & Kinesiology Official Journal of the International Society of Electrophysiological Kinesiology, 2006, 16(6): 541.

[5] PUTERMAN M L. Markov Decision Problems [M]. New York: Wiley, 1994.

[6] BELLMAN R E. A markov decision process[J]. Journal of Mathematical Fluid Mechanics, 1957, 6(1): 65-73.

[7] SUTTON R S. Learning to predict by the methods of temporal differences [J]. Machine Learning, 1988, 3 (1): 9-44.

[8] RUMMERY G A, NIRANJAN M. On-line Q-learning using connectionist systems [M]. University of Cambridge, Department of Engineering, 1994.

[9] VAN S H, VAN H H, WHITESON S, et al. A theoretical and empirical analysis of Expected Sarsa[J]. Proceedings of the IEEE Symposium on Adaptive Dynamic Programming & Reinforcement Learning, 2009: 177-184.

[10] VAN H H. Insights in Reinforcement Learning: Formal Analysis and Empirical Evaluation of Temporal-Difference Learning Algorithms[R]. Wöhrmann Print Service, 2010. ISBN 978-90-39354964.

[11] DE A K, HERNANDEZ-GARCIA J F, HOLLAND G Z, et al. Multi-step Reinforcement Learning: A Unifying Algorithm[J]. arXiv preprint arXiv: 1703.01327, 2017.

[12] WATKINS C J C H. Learning from Delayed Rewards[J]. Robotics & Autonomous Systems, 1989, 15 (4): 233-235.

[13] CICHOSZ P. Truncating Temporal Differences: On the Efficient Implementation of TD (lambda) for Reinforcement Learning[J]. 1995, 2: 287-318.

[14] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction. Cambridge [M]. London, England: MIT Press, Cambridge, 2017.

[15] JAAKKOLA T, JORDAN M I, SINGH S P. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms [J]. Neural Computation, 1993, 6 (6): 1185-1201.

[16] SINGH S, JAAKKOLA T, LITTMAN M L. Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms [J]. Machine Learning, 2000, 38 (3): 287-308.