一种加速时间差分算法收敛的方法

何 斌 [1] 刘 全 [1, 2, 3, 4] 张琳琳 [1] 时圣苗 [1] 陈红名 [1] 闫 岩 [1]

摘 要 时间差分算法 (Temporal difference methods, TD) 是一类模型无关的强化学习算法. 该算法拥有较低的方差和可以在线 (On-line) 学习的优点, 得到了广泛的应用. 但对于一种给定的 TD 算法, 往往只能通过调整步长参数或其他超参数来加速收敛, 这也就造成了加速 TD 算法收敛的方法匮乏. 针对此问题提出了一种利用蒙特卡洛算法 (Monte Carlo methods, MC) 来加速 TD 算法收敛的方法 (Accelerate TD by MC, ATDMC). 该方法不仅可以适用于绝大部分的 TD 算法, 而且不需要改变在线学习的方式. 为了证明方法的有效性, 分别在同策略 (On-policy) 评估、异策略 (Off-policy) 评估和控制 (Control) 三个方面进行了实验. 实验结果表明 ATDMC 方法可以有效地加速各类 TD 算法.

关键词 强化学习, 时间差分算法, 蒙特卡罗算法, 加速收敛

引用格式 何斌, 刘全, 张琳琳, 时圣苗, 陈红名, 闫岩. 一种加速时间差分算法收敛的方法. 自动化学报, 2021, **47**(7): 1679−1688

DOI 10.16383/j.aas.c190140

A Method of Accelerating the Convergence of Temporal Difference Learning

HE Bin[1] LIU Quan[1, 2, 3, 4] ZHANG Lin-Lin[1] SHI Sheng-Miao[1] CHEN Hong-Ming[1] YAN Yan[1]

Abstract Temporal difference methods (TD) methods are a class of model-free reinforcement learning methods. TD methods have been widely used, which have a low variance and can learn on-line. But for a given TD method, there is only one approach that adjusts the step size or other parameters to accelerate the convergence, which leads to a lack of methods to make it. To solve this problem, we introduce a method for accelerating TD methods based on Monte Carlo (MC) methods, which not only can be applied to most of the TD methods, but also do not need to change the way of on-line learning. In order to demonstrate the effectiveness of the method, experiments were carried out in the three aspects: the on-policy evaluation, off-policy evaluation and control. The experimental results show that the accelerate TD by MC (ATDMC) method can effectively accelerate TD methods.

Key words Reinforcement learning, temporal difference (TD) methods, Monte Carlo (MC) methods, accelerating convergence

Citation He Bin, Liu Quan, Zhang Lin-Lin, Shi Sheng-Miao, Chen Hong-Ming, Yan Yan. A method of accelerating the convergence of temporal difference learning. *Acta Automatica Sinica*, 2021, **47**(7): 1679−1688

收稿日期 2019-03-07 录用日期 2019-08-22
Manuscript received March 7, 2019; accepted August 22, 2019
国家自然科学基金项目 (61772355, 61702055, 61502323, 61502329), 江苏省高等学校自然科学研究重大项目 (18KJA520011, 17KJA520004),吉林大学符号计算与知识工程教育部重点实验室资助项目 (93K172014K04, 93K172017K18),苏州市应用基础研究计划工业部分 (SYG201422) 资助
Supported by National Natural Science Foundation of China (61772355, 61702055, 61502323, 61502329), Jiangsu Province Natural Science Research University Major Projects (18KJA520011, 17KJA520004), Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (93K172014K04, 93K172017K18), Suzhou Industrial Application of Basic Research Program (SYG201422)
本文责任编委 魏庆来
Recommended by Associate Editor WEI Qing-Lai
1. 苏州大学计算机科学与技术学院 苏州 215006 2. 苏州大学江苏省计算机信息处理技术重点实验室 苏州 215006 3. 吉林大学符号计算与知识工程教育部重点实验室 长春 130012 4. 软件新技术与产业化协同创新中心 南京 210000
1. School of Computer Science and Technology, Soochow University, Suzhou 215006 2. Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006 3. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012 4. Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210000

强化学习是机器学习领域中最接近人类和动物学习的方法[1], 在机器人自主决策和学习、复杂动态系统的优化控制和自动驾驶等领域中有着广泛的应用[2−5]. 强化学习算法分为基于模型和无模型两种. 基于模型的算法需要一个精确且完整的环境模型, 而无模型算法没有这个要求. 无模型的强化学习算法中基于值函数的算法较为常用. 该算法需要根据值函数得出最优的策略. 如果值函数无法准确地被评估, 或者评估的效率过低, 那么将得不到最优的策略, 或者耗时过长. 所以, 一个精确和高效的值函数评估方法对于基于值函数的强化学习算法至关重要[6−7].

对于值函数评估问题, 经典的解决方法是 TD (Temporal difference methods)算法和 MC (Monte Carlo methods) 算法. 其中, TD 算法可以像动态规划 (Dynamic program) 方法一样使用自举 (Bootstrap) 的方式, 即利用已经评估过的状态值来进行

新的评估. 这也使得 TD 算法可以不需要等待情节结束, 就可以进行值函数的更新, 从而实现了在线学习的方式. MC 算法必须等到情节结束后才能进行值函数的更新, 因而只能使用离线学习 (Off-line) 的方式[1]. 但是如果 TD 算法所利用的评估过的状态值有偏差, 那么评估的结果往往在真实值附近. 另外, TD 算法的收敛效率很大程度上取决于初值和步长参数的设定. 优秀的初值和恰当的步长参数可以极大地缩小所需要策略迭代的次数, 但是初值的设定往往需要丰富的经验和相关领域的专业知识. 对于步长参数设定的问题, 有许多自动设置步长参数的算法可以解决[8−9]. 然而, 尽管这些算法有许多理论上的优势, 但是由于其自身的复杂性和进一步增加了 TD 算法的时间复杂度而得不到广泛的运用, 实际应用中更多地采用固定步长参数的方法[10]. 这也就造成了加速一个特定 TD 算法收敛速率的方法的匮乏.

论文提出的 ATDMC 方法是利用 MC 算法的无偏的性质, 在 TD 算法完成一次值函数评估后, 进行值函数改进. 即在广义策略迭代中加入减少值函数和真实值函数的之间距离的步骤, 使得在不改变 TD 算法的在线学习的方式基础上, 加速算法的收敛.

文中第 1 节描述了强化学习的相关基本概念和所用到的符号的定义. 第 2 节给出 TD 算法的收敛速率的分析和 ATDMC 方法的详细推导过程. 第 3 节用实验分别展示在同策略评估、异策略评估和控制三个方面的加速效果. 最后一节给出结论和后续的工作方向.

1 强化学习相关基本概念

1.1 马尔科夫决策过程

马尔科夫决策过程 (Markov decision process, MDP) 是研究各种强化学习算法的理论基础, 可以对大多数强化学习问题进行建模[11]. 这是因为 MDP 是在交互过程中学习以期达到某种目标的问题的框架, 而强化学习问题是指如何在和环境交互过程中使得收到的累计奖赏最大化的问题. 其中, 学习和决定动作的对象称为智能体 (Agent), 智能体交互的对象为环境.

MDP 是由一个 4 元组 $(\mathcal{S},\ \mathcal{A},\ \mathcal{R},\ \mathcal{P})$ 构成. 其中, $\mathcal{S}$ 是指智能体在和环境进行交互过程中, 环境的所有状态构成的集合. $\mathcal{A}$ 是指智能体采取的动作构成的集合, 收到的奖赏构成的集合为 $\mathcal{R}$. $\mathcal{P}$ 是由状态转移函数构成. 智能体和环境的一次交互是指智能体采取一个动作 (Action) 后, 环境返回一个奖赏 (Reward), 然后智能体对环境进行观测, 得到一个新的状态 (State). 为了表述简单, 将一次完整的交互定义在时间步 t 发生, 智能体所处的状态记为 S_t, 采取的动作记为 A_t, 获得的奖赏记为 R_{t+1}, 到达的下一个状态记为 S_{t+1}. 将由时间步 0 到终止时间步 T 的整个过程称为一个情节 (Episode). 可见, 一个情节中智能体经过的状态、采取的动作和获得的奖赏构成了一个序列:

$$S_0, A_0, R_1, S_1, A_1, R_2, \cdots$$

将其中的奖赏的累计求和称为累计奖赏. 从状态 S_t 开始, 直至结束状态 S_T 获得的累计奖赏定义为:

$$G_t = \sum_{i=t+1}^{T} R_i \tag{1}$$

在一些强化学习任务中, 最近获得的奖赏往往被赋予更大的权重值. 因为相比于过去获得的奖赏, 新的奖赏更值得关注. 另外, 当情节的时间步较长时, G_t 将会变为一个比较大的值. 对此, 往往需要对获得的奖赏乘上一个折扣因子 γ, 来控制 G_t 的大小, 则:

$$G_t = \sum_{i=t+1}^{T} \gamma^{i-t-1} R_i \tag{2}$$

可见, 如何使 G_t 最大化的问题即为强化学习在 MDP 框架下所要解决的问题.

1.2 广义策略迭代

广义策略迭代 (Generalized policy iteration, GPI) 是强化学习中最一般的思想, 几乎所有的强化学习算法都可以描述成广义策略迭代的形式. 强化学习算法所要最大化的目标 G_t 的值是由在各个状态下采取的动作决定的, 而在某个状态采取何种动作是由策略决定的. 强化学习的目标即为寻找一个策略使得 G_t 最大化, GPI 对此提供了具体思路.

GPI 可以分为策略评估 (Policy evaluation) 和策略改进 (Policy improvement) 两部分, 由这两部分交替进行从而实现该思想. 其中, 策略评估是指在一个特定的状态下, 对遵循一个固定策略的智能体将会收到的累计奖赏的期望进行估值. 对于一个给定的策略 π, 从 t 时间步开始, 获得的累计奖赏的期望记为:

$$v_\pi(S_t) = \mathrm{E}_\pi[G_t|S_t] \tag{3}$$

这个值被称为状态 S_t 的 v 值, 也称为 S_t 在策略 π 的真实值. 相应地, 如果在时间步采取的动作为 A_t, 那么获得的累计奖赏的期望为:

$$q_\pi(S_t, A_t) = \mathrm{E}_\pi[G_t|S_t, A_t] \tag{4}$$

这个值也被称为状态动作对 (S_t, A_t) 的 q 值. 由 v 值和 q 值的定义可知:

$$v_\pi(S_t) = \mathrm{E}_{A_t}[\pi(A_t|S_t)q_\pi(S_t, A_t)] \tag{5}$$

其中, $\pi(A_t|S_t)$ 是在状态 S_t 采取动作 A_t 的概率.

策略评估后, 会根据评估出来的 q 值函数进行策略改进. 策略改进一般采用 ϵ-greedy 算法, 即最大的 q 值对应的动作被选取的概率为 $1-\epsilon$, 另外有 ϵ 的概率任取一个动作. 一次策略评估和策略改进形成了一次完整的策略迭代. 下一轮迭代将根据新的策略继续评估, 然后再改进策略. 经过多次策略迭代后, 策略达到了稳定的状态, 各个状态的值函数达到最大值. 此时的策略称为最优策略, 对应的值函数称为最优值函数, 记为

$$v_\star(S_t) = \max_\pi v_\pi(S_t) \tag{6}$$

1.3 TD 算法

TD 算法是强化学习中最核心的算法, 可以用来评估 v 值和 q 值函数. 如果一些状态的值函数已被评估过或是被赋予了初值 (虽然初值可能是错误的, 但也可以视为被评估过), 则可以利用这些评估过的值来评估其他状态的值函数. TD 算法会对每一个 v 值或者 q 值进行赋初值, 再利用这些初值进行新的评估, 然后进行策略改进, 不断循环进行从而实现 GPI.

TD 算法策略迭代的过程可以看作是值函数不断向 $v_\star$ 靠近的过程. 在具体的 TD 算法中, $v_\star$ 是未知的值, 往往被替换成相应算法的目标 U_{TD}. 根据所利用的评估过的值函数是一步还是多步之后的状态的值函数, 将 TD 算法分为一步或 n 步 TD. 对于 n 步 TD 算法而言, U_{TD} 为:

$$G_{t:t+n} = \sum_{i=t+1}^{t+n} \gamma^{i-t-1}R_i + \gamma^n V(S_{t+n}) \tag{7}$$

其中, $V(S_{t+n})$ 为 n 步之后的状态的 v 值的估计值, 而这个值可能是错误的. 如果该值为对应状态的真实值, 则显然 $G_{t:t+n}$ 的期望也为真实值. 另外, $n=1$ 的情况即为一步 TD 的 U_{TD}. 利用 n 步 TD 算法的目标, TD 算法的值函数的更新公式为:

$$V(S_t) = V(S_t) + \alpha(G_{t:t+n} - V(S_t)) \tag{8}$$

其中, α 为步长参数.

强化学习在处理状态空间很大的问题时, 往往采用函数逼近的方法. 使用状态的特征向量 $\boldsymbol{x}(S_t)$ 和权重向量 $\boldsymbol{w}$ 来表示状态的值函数. 在使用线性逼近方法的情况下, 值函数记为:

$$V(S_t, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}(S_t) \tag{9}$$

显然在函数逼近的条件下, 并不一定存在一个权重向量 $\boldsymbol{w}$ 使得所有状态的值都能等于对应的 v_π. 所以, 为了求出使得值函数和 v_π 之间的距离的平方和最小的 $\boldsymbol{w}$, 将损失函数定义为:

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=t+1}^{T} (v_\pi(S_t) - V(S_t, \boldsymbol{w}))^2 \tag{10}$$

由于 $v_\pi(S_t)$ 值未知, 将其替换为 U_{TD}. 那么 TD 算法的权重向量的更新公式定义为:

$$\boldsymbol{w} = \boldsymbol{w} + \alpha(U_{TD} - V(S_t, \boldsymbol{w}))\nabla V(S_t, \boldsymbol{w}) \tag{11}$$

1.4 MC 算法

MC 算法是另外一种强化学习中评估值函数的算法. MC 算法的值函数的更新公式为:

$$V(S_t) = V(S_t) + \frac{G_t - V(S_t)}{n} \tag{12}$$

其中, n 是情节数. 从公式可见 MC 算法的目标为 G_t, 而由 G_t 的定义可知 G_t 的值是整个情节中所有的奖赏的累计和. 也就意味着如果情节不结束, G_t 的值都是未知的, 从而使得 MC 算法只能使用离线学习的方式.

从 v_π 的定义可知 MC 算法的更新目标 G_t 的期望值为 v_π, 即 G_t 是 v_π 的无偏估计. 而 TD 算法的目标只有在 $V(S_{t+n})$ 为真实值的情况下, 其期望值才等于真实值. 再有, 策略迭代开始时 $V(S_{t+n})$ 的值都为初值 (往往都被设为 0), 距离真实值较远. 所以在初期, MC 算法的更新量 $G_t - V(S_t)$ 是大于 TD 算法的更新量. 另外, 对于使用步长参数 $1/n$ 的 MC 算法和使用固定步长参数的 TD 算法, 它们步长参数的不同也使得策略迭代初期 MC 算法的收敛速率更快.

2 ATDMC 方法

本节将通过分析得出影响 TD 算法收敛速率的因素, 然后通过减少这些因素的影响来加快 TD 算法的收敛速率.

2.1 TD 算法的收敛速率

任取一个状态 S_t, 将其 $n+1$ 次更新后的估计值记为 V^{n+1}, 策略 π 下该状态的真实值记为 v_π. 将 V^{n+1} 值和真实值之间距离的平方的期望定义为误差 e^{n+1}, 则:

$$e^{n+1} = \mathrm{E}[(V^{n+1} - v_\pi)^2] \tag{13}$$

显然 e^{n+1} 减小到 0 的速率即为 TD 算法收敛的速率. 将 TD 算法的更新公式代入式 (13),得

$$e^{n+1} = \mathrm{E}\left[(V^n - v_\pi + \alpha(U_{TD} - v_\pi - (V^n - v_\pi)))^2\right] \tag{14}$$

假定 U_{TD} 的期望等于 v_π, 将式 (14) 展开得:

$$\mathrm{E}[(1-\alpha)^2(V^n - v_\pi)^2 + \alpha^2(U_{TD} - v_\pi)^2)] \tag{15}$$

整理得:

$$e^{n+1} = (1-\alpha)^2 e^n + \alpha^2 \mathrm{D}[U_{TD}] \tag{16}$$

进一步, 展开得:

$$e^{n+1} = (1-\alpha)^{2(n+1)} e^0 + \frac{\alpha}{2-\alpha}\left(1-(1-\alpha)^{2(n+1)}\right)\mathrm{D}[U_{TD}] \tag{17}$$

式 (17) 中 e^0 是由初值所确定的, 而 U_{TD} 的方差则是由具体的 TD 算法所决定的. 随着 $\alpha \to 0$ 且 $n \to \infty$, e^{n+1} 逐步变小, 最后收敛到 0. 可见, 收敛的速率是由步长参数、初值和 TD 算法本身的性质三者共同决定的. 在实际使用 TD 算法时, 往往采用固定的步长参数, e^{n+1} 变为一个有上界的量. 如果设置步长参数较小时, 等式的第一项收敛到 0 的速率较慢, 第二项一直保持为一个很小的值, 所以最终的收敛的效果是速率慢, 但是很稳定. 而如果步长参数设置较大时, 第一项会很快收敛, 第二项的值对算法的收敛速率起主要的影响作用, 表现为虽然收敛速率快, 但是波动程度大. 这一点与调试步长参数的过往经验是一致的, 只有当步长参数设置适中时, 算法的效果才能达到最优.

对于一个给定的策略和一个固定的步长参数条件下, 如果想要加速 TD 算法, $\mathbf{D}[U_{TD}]$ 对于给定 TD 算法是无法改变的, 那么只有减少 e^n . 虽然 e^0 是由初值确定的, 但是 $e^1, e^2, \cdots$ 等都是可以减小的.

2.2 TD 算法的目标

在第 2.1 节中, 论文假定 U_{TD} 的期望等于 v_π, 而在策略迭代初期该条件是不满足的. 实际上, 在一个给定策略下, 一个情节中不同时间步的同一状态的 TD 算法的目标 U_{TD} 构成了一个不稳定的序列 (A non-stationary series). 即在收敛之前, U_{TD} 的期望值一直在变化, 直至收敛后, U_{TD} 的期望才能达到稳定状态. 假设所有状态初值都为 0, 奖赏都为正时, 构成的序列如图 1 所示, 图中的点是 U_{TD} 的估计值, 图中的线是由 U_{TD} 的期望构成. 这也说明在策略迭代初期, 各个状态的值和 U_{TD} 之间的距离较小. 并且加上步长参数的作用, 使得初期的收敛速率较慢. 但是 MC 方法的目标是一个稳定的序列, 其期望值始终等于 v_π . 如果采用 MC 算法的目标作为再次更新的目标则可以减小 e^n, 从而加速收敛.

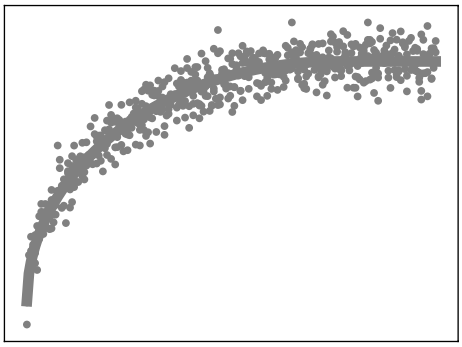


图 1 由 U_{TD} 构成的不稳定序列

Fig. 1 A non-stationary series

2.3 ATDMC 方法的推导过程

根据前面两节的分析, TD 算法在策略迭代初期收敛速率慢的问题可以通过减小 e^n 来解决. 而 MC 方法的目标在策略迭代初期相较于 TD 算法的目标距离真实值更近, 可以借助其来减小 e^n. 对此, 在一般的 GPI 中加入值改进的步骤, 使得 e^n 减小. 改进后的 GPI 如图 2 所示. 具体的方式是在 TD 算法更新完成后, 再以 MC 算法的目标作为新目标来更新.

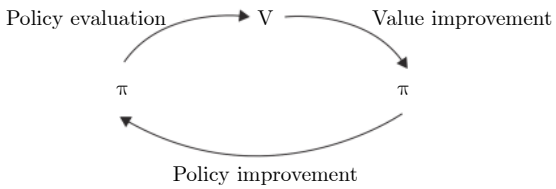


图 2 改进后的 GPI

Fig. 2 The improved GPI

为了再以 MC 算法的目标进行更新, 线性函数逼近的方式下, 将损失函数定义为:

$$\mathcal{L}(\boldsymbol{w}) = \sum_{t=1}^{T}\left(G_t - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_t\right)^2 \tag{18}$$

其中, $\boldsymbol{w}$ 为经过 TD 算法更新后的权重向量. 对损失函数求 $\boldsymbol{w}$ 的偏导:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = -2\sum_{t=1}^{T}\left(G_t - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_t\right)\boldsymbol{x}_t \tag{19}$$

然后, 令 $\boldsymbol{N}_i = \sum_{t=1}^{i}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_t\right)\boldsymbol{x}_t$,则显然有:

$$\boldsymbol{N}_i = \boldsymbol{N}_{i-1} + \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right)\boldsymbol{x}_i \tag{20}$$

接着, 令 $\boldsymbol{M}_i = \sum_{t=1}^{i} G_t\boldsymbol{x}_t$. 从 G_t 的定义可知, 只有到情节结束才可以计算出该值. 这也就意味着需要存储情节中的所有状态和收到的奖赏, 从而使得算法的空间复杂度增加了. 另外, 由于 G_t 的值在

情节结束之后才得出, 所以大部分的计算都被集中到情节结束. 更恰当方法应该是将计算分散到每个时间步中执行. 对此, 论文引入了均摊迹 (Dutch traces) 来解决. 均摊迹不但可以分散运算, 而且不需要保存情节中的每个细节. 将 $\boldsymbol{M}_i$ 展开得 (为了表述方便, G_t 使用了不带折扣因子的定义):

$$\begin{aligned}\boldsymbol{M}_i =&(r_T + r_{T-1} + \cdots + r_1)\boldsymbol{x}_1 +\\ &(r_T + r_{T-1} + \cdots + r_2)\boldsymbol{x}_2 +\\ &\cdots +\\ &r_T\boldsymbol{x}_T =\\ &r_T(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_T) +\\ &r_{T-1}(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_{T-1}) +\\ &\cdots +\\ &r_1\boldsymbol{x}_1\end{aligned}$$

观察第二个等号后面的式子, 可知 $r_1\boldsymbol{x}_1$ 可以在时间步 1 求出, 相应地 $r_{T-1}(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_{T-1})$ 可以在情节结束前一个时间步求出. 再令 $\boldsymbol{P}_i = \sum_{t=1}^{i} \boldsymbol{x}_t$, 则:

$$\boldsymbol{P}_i = \boldsymbol{P}_{i-1} + \boldsymbol{x}_i \tag{21}$$

$$\boldsymbol{M}_i = \boldsymbol{M}_{i-1} + \boldsymbol{P}_i r_i \tag{22}$$

可见, 使用均摊迹只需要保存两个变量 $\boldsymbol{P}_i$ 和 $\boldsymbol{M}_i$, 然后在各个时间步不停地更新这两个值. 最终 ATDMC 方法的更新公式为:

$$\boldsymbol{w} = \boldsymbol{w} + \beta(\boldsymbol{M}_T - \boldsymbol{N}_T) \tag{23}$$

另外, $\boldsymbol{M}_0$、$\boldsymbol{N}_0$、$\boldsymbol{P}_0$ 都为 $\boldsymbol{0}$ 向量. β 为步长参数. 式 (20)、(21) 和 (22) 在 TD 算法更新完权重向量后执行, 式 (23) 在情节结束后执行. 这样并没有改变原 TD 算法的结构, 也就不会改变在线学习的方式.

在异策略学习中, 由于目标策略和行为策略不一致, 需要采取重要性采样方法来放大或缩小值函数. 只需要将式 (22) 改为

$$\boldsymbol{M}_i = \boldsymbol{M}_{i-1} + \boldsymbol{P}_i \rho_i r_i \tag{24}$$

其中, ρ_i 为重要性采样因子.

3　实验及结果分析

本节将分别在同策略评估、异策略评估和控制三个方面进行实验, 展示 ATDMC 方法可以对多种 TD 算法进行加速收敛, 以及原 TD 算法的步长参数 α 和 ATDMC 方法的步长参数 β 对加速效果的影响. 其中, 策略评估的实验侧重覆盖多种 TD 算法, 而控制的实验则更多的体现步长参数的影响.

3.1　同策略评估

博扬链 (Boyan chain) 是强化学习中用于比较不同 TD 算法性能的问题[12]. 这个问题中一共有 13 个状态. 其中, 状态 12 是开始状态, 状态 0 是终止状态. 每个状态都有两个动作 (状态 1 两个动作都到达终止状态 0), 如图 3 所示. 除了状态 1 到状态 0 的奖赏是−2, 其他收到的奖赏都是−3. 状态 12、8、4、0 的特征向量分别为 [1, 0, 0, 0]、[0, 1, 0, 0]、[0, 0, 1, 0]、[0, 0, 0, 1]. 处于这些状态之间的状态的特征向量可以用插值法求得. 例如, 状态 11、10、9 的特征向量分别为 [3/4, 1/4, 0, 0]、[1/2, 1/2, 0, 0]、[1/4, 3/4, 0, 0]. 所要评估的策略设定为每个状态的两个动作被执行的概率相等.

为了覆盖较多的 TD 算法, 用 ATDMC 方法分别加速 TD(0)、GTD[13]、GTD2 和 TDC[14] 算法. 这 4 种算法相比于其他的 TD 算法 (如 LSTD、KTD 和 GPTD 等) 收敛速率更快[15]. 另外, 实验通过比较均方根误差 (Root mean square of error, RMSE) 的减小速率来体现 ATDMC 方法的加速效果. RMSE 是衡量 TD 算法求出的值和真实值之间误差的常用指标. 如果 ATDMC 方法的 RMSE 减小速率更快, 则可以体现 ATDMC 方法的加速效果. 为了让步长参数的大小一致, 原 TD 算法和加速后的算法的步长参数 α 都设为 0.5. 如果算法存在次级权重向量, 那么其对应的步长参数也都设为 0.25. ATDMC 方法的步长参数 β 都为 0.1. 每个实验都是重复 100 次取平均值, 且折扣因子 γ 都为 1, 权重向量的初值都为 $\boldsymbol{0}$ 向量.

比较结果如图 4 所示. 图中实线对应原 TD 算法, 虚线是加速后的效果. 可见, 虽然各个 TD 算法收敛速率不尽相同, 但是都得到了加速. 另外, ATDMC 方法利用 MC 算法目标的无偏性质进行一次值改进后, 误差就已经显著减少.

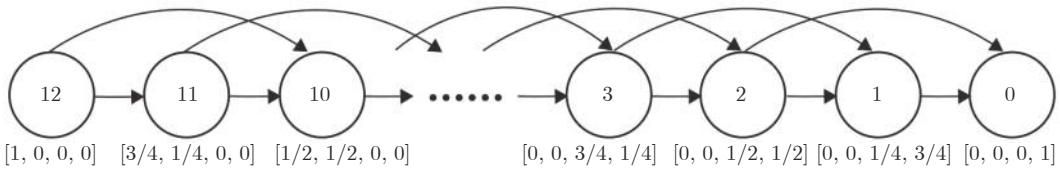


图 3　博扬链

Fig. 3　Boyan chain

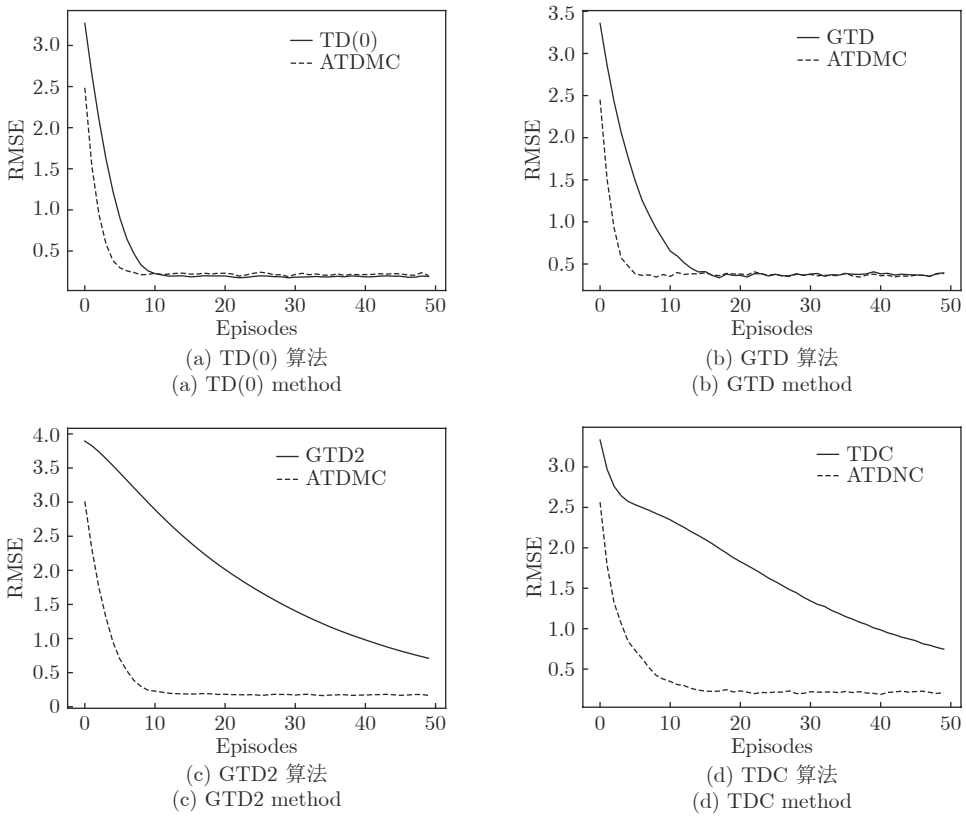


图 4 同策略评估

Fig. 4 On-policy estimation

3.2 异策略评估

本节是在 11 状态的随机漫步问题上做实验, 并在一般的随机漫步问题上进行了修改[16], 如图 5 所示. 这个问题共有 11 个状态, 状态 1 是开始状态, 状态 11 是终止状态, 每个状态都有向左和向右两个动作, 除了状态 1 的向左的动作是到达自己外, 其他的动作都是到达相邻状态. 可以看出, 由于向左和向右动作的作用使得实验一个情节的时间步数可能会很长. 而较长的情节才能体现异策略评估算法的优劣性. 前一节中的博扬链的情节最长为 12 步, 最短为 6 步, 不再适用于异策略评估, 所以换为随机漫步问题. 只有到达终止状态 11 才能获得 1 的奖赏. 其他所有的奖赏都为 0. 特征向量是一个 10 维向量. 状态 1 的特征向量是前 1 维为 $\sqrt{1}$, 后面为 0. 状态 2 的特征向量是前 2 维为 $\sqrt{2}$, 后面为 0, 以此类推. 状态 11 的特征向量为 $\mathbf{0}$ 向量. 评估的目标策略为采取向左的动作概率为 0.4, 采取向右的动作概率等于 0.6. 而行为策略采取向左和向右动作的概率相等.

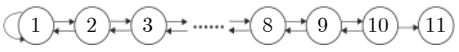


图 5 11 状态的随机漫步

Fig. 5 11-State random walk

资格迹是强化学习的重要方法[17−19]. 在异策略评估方面, TD 算法和资格迹相结合使得收敛速率和准确度都得到了提升. 为了证明使用资格迹加速后, ATDMC 方法仍然可以加速, 实验采用了 TD (0) 和 GTD 算法的资格迹版本: TD (λ) [20−21]、GTD (λ) [22]. 另外, 为了进一步覆盖更多的 TD 算法, 又选取了另外两种算法: HTD (λ) [23] 和 ETD (λ) [24−25]. 参数设置方面, λ 都设为 0.6, ATDMC 方法的步长参数 β 都为 0.001. 其他参数与同策略评估设置一致. 每个实验也同样重复 100 次取平均值.

比较的结果如图 6 所示. 可见, 异策略评估方面采用资格迹的 TD 算法的收敛速率也都得到了加速. 需要特别指出的是异策略评估算法中 HTD 的收敛速率最快, 但是牺牲了准确度. ATDMC 方法在保证收敛速率不变的条件下, 提高了 HTD 算法的准确度.

3.3 控制

山地车 (Mountain car) 和平衡杆 (Cart pole) 问题是强化学习控制方面的经典问题, 许多算法都

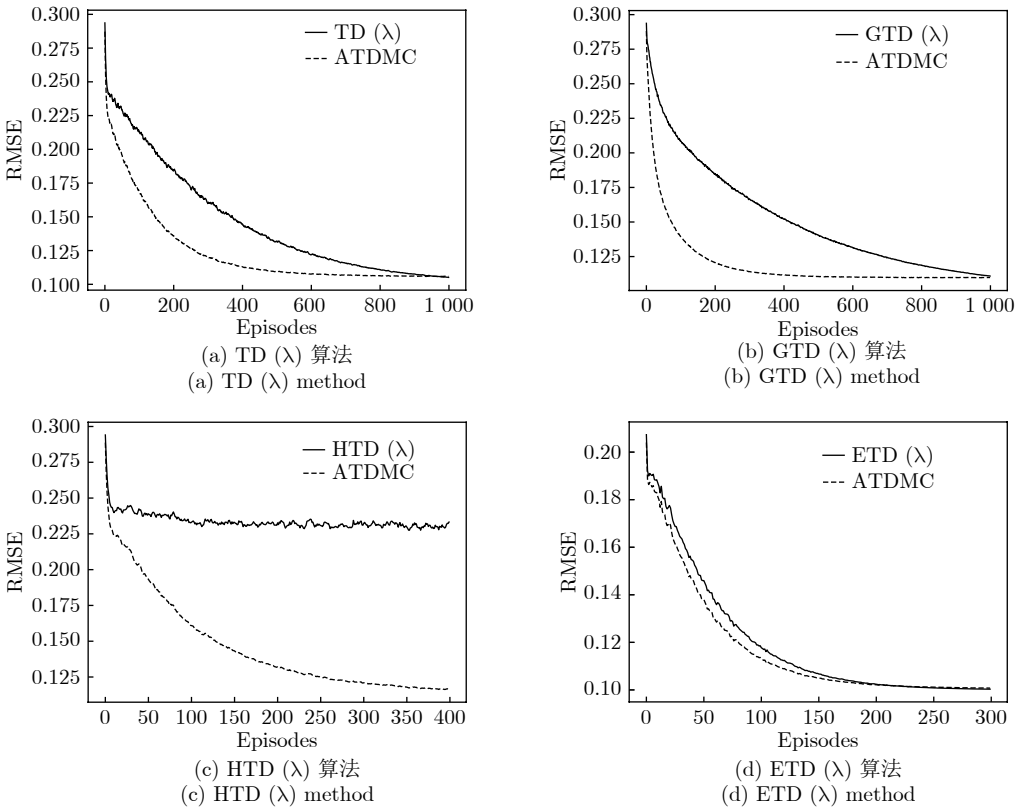


图 6　异策略评估

Fig. 6　Off-policy estimation

是在这两个问题上进行实验. 关于这两个问题的细节在许多文献中有着详细的描述. 另外, 选用这两个问题还因为这两者的目标刚好相反, 山地车问题的目标是小车尽可能快地到达山顶, 平衡杆问题是尽可能久地保持杆子的平衡, 一个是让一个情节的时间步数尽可能的少, 而另一个让时间步数尽可能的多.

山地车问题采用的是 sarsa (λ) 算法[26] 来解决. 为了体现算法收敛速率的快慢, 实验只关心前 50 个情节的时间步数. 前 50 个情节的平均时间步数越短, 说明收敛速率越快. 实验采用了瓦片编码 (Tile coding), 选择动作的 ϵ-greedy 算法的参数 ϵ 设为 0.

为了展示相同步长参数 β 和不同的 α 对加速效果的影响, 第一个实验固定 $\beta = 0.0001$, 而 sarsa (λ) 算法采取不同的步长参数 α, 实验结果如图 7 所示. 纵坐标的步数是前 50 个情节的时间步数的平均值. 从图中看出当 α 较大时, β 设为 0.0001 没有加速, 反而减慢了速率. 当然在 α 较大时, sarsa (λ) 算法的效果也比较差, 所以需要调整步长参数. 当 α 较小时, 虽然 α 不同但是都得到了加速.

接着, 为了展现不同的 β 和同一 α 加速效果, α

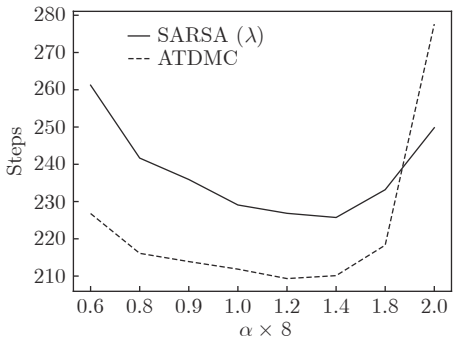
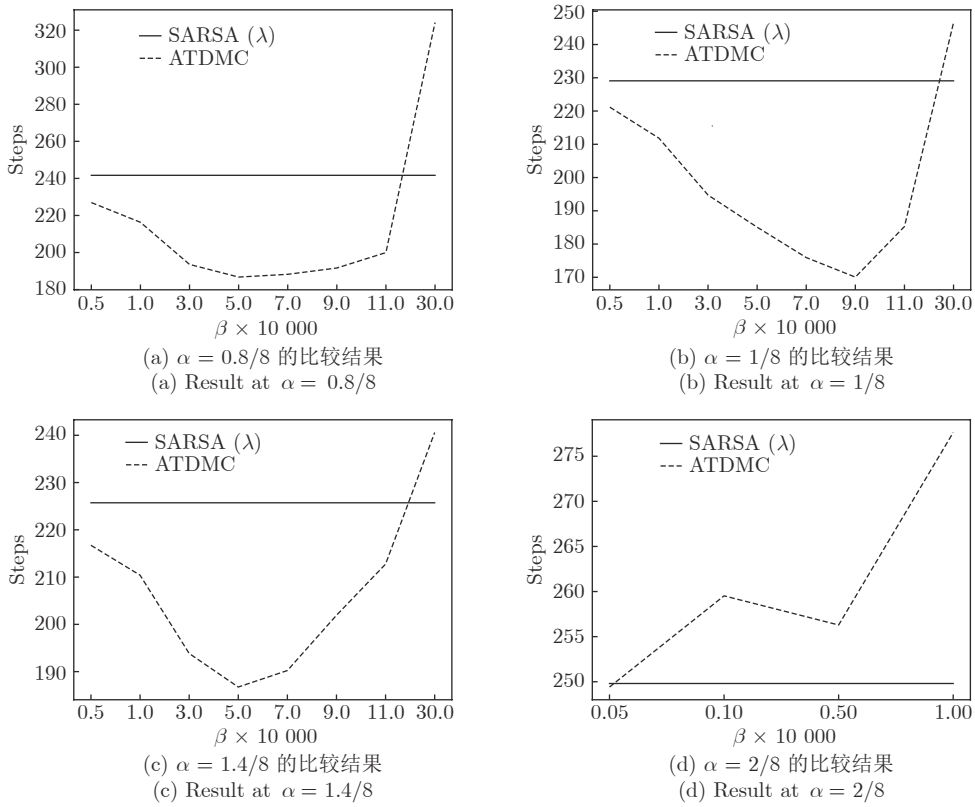


图 7　山地车问题 (β 固定)

Fig. 7　Mountain car (fixed β)

分别取了 0.8/8、1/8、1.4/8 和 2/8. 结果如图 8 所式. 当 α 较小时, 随着 β 逐渐变大, 加速效果由逐渐变好再逐渐变坏. 当 α 较大时, ATDMC 方法的并没有加速效果. 这一点与其他强化学习算法一致, 过大的步长参数将会使算法效果变差.

第二个实验是平衡杆问题. 平衡杆的每个情节的步长最大值为 200, 当连续 100 个情节的平均步长大于 195 视为该问题得到解决. 易知, 解决问题所花费的情节数越短说明算法收敛得越快. 实验设计和山地车问题是一致的, 但是采用的是一步 sar-



(a) $\alpha = 0.8/8$ 的比较结果
(a) Result at $\alpha = 0.8/8$

(b) $\alpha = 1/8$ 的比较结果
(b) Result at $\alpha = 1/8$

(c) $\alpha = 1.4/8$ 的比较结果
(c) Result at $\alpha = 1.4/8$

(d) $\alpha = 2/8$ 的比较结果
(d) Result at $\alpha = 2/8$

图 8 山地车问题 (α 固定)

Fig. 8 Mountain car (fixed α)

sa 算法, 选择动作的 ϵ -greedy 算法的参数 ϵ 设为 0.08. 首先也是比较相同的 β 和不同的 α 的对加速效果的影响, 固定 β 为 0.0005. 实验结果如图 9 所示. 接着也比较不同 β 和相同 α 的影响. α 分别等于 0.2/16、0.4/16、0.6/16 和 0.8/16. 实验结果如图 10 所示. 图中纵座标是总共进行的情节数, 其中最后 100 个情节是验证问题是否解决, 前面的情节是学习花费的情节. 和山地车问题的实验结果相比, 虽然实验内容不同, 但是加速效果类似.

从这两个实验可以看出在 TD 算法步长参数 α 过大时, ATDMC 方法并没有加速效果. 而当步长参数 α 较小时, 可以实现不同程度的加速, 但是还没有达到最好效果. 只有当步长参数 α 适中时, ATDMC 方法的步长参数 β 也适中时, 才能达到算法的最好效果.

4 结论和后续工作方向

针对加速特定 TD 算法收敛速率的方法匮乏的问题, 本文提出利用 MC 算法无偏的性质来加速 TD 算法收敛的 ATDMC 方法. 通过分析 TD 算法的收敛速率得出可以通过减小 $e^1, e^2, \cdots$ 来加速收敛, 而 MC 算法的目标可以在学习初期减小这些值, 进一步

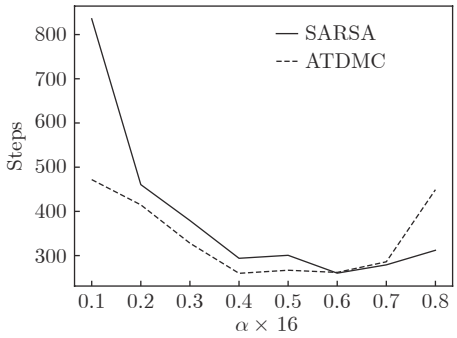
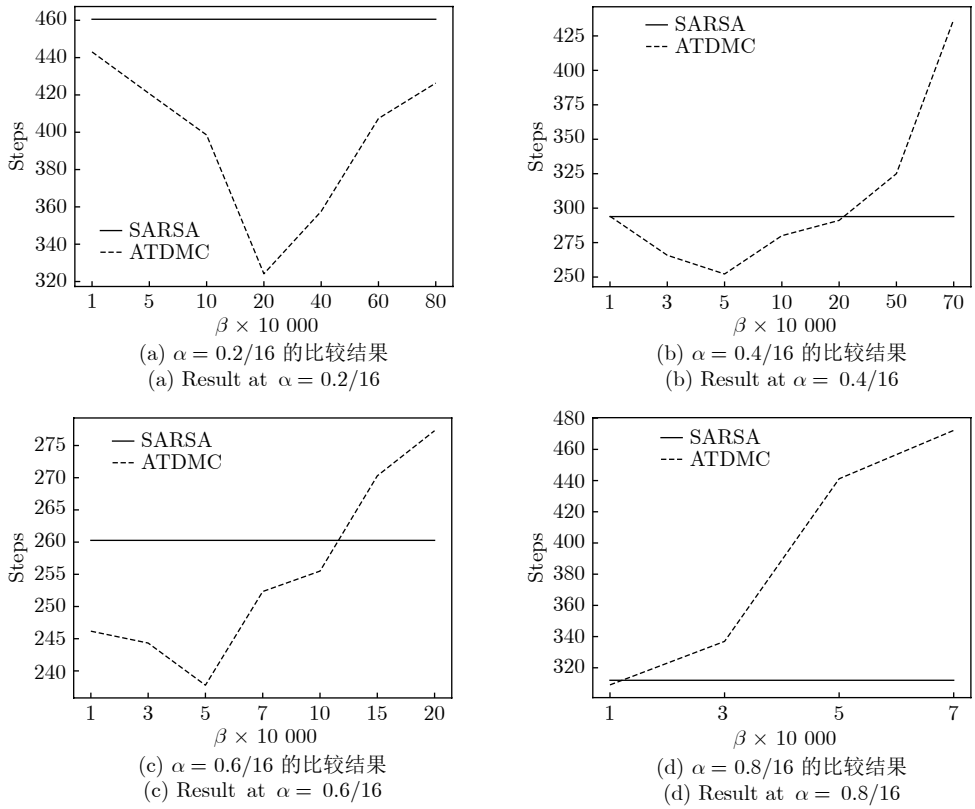


图 9 平衡杆问题 (β 固定)

Fig. 9 Cart pole (fixed β)

推导得出了最终的方法. 从推导过程可知, 加速的实现并不需要改变原 TD 算法, 也即 TD 算法的在线实现的方式不会被改变. 同时, 只引入了 3 个额外的变量, 并没有增加算法的空间复杂度. 从实验结果可以看出, 本文提出的方法可以有效地加速各类 TD 算法, 恰当的步长参数使得算法达到最优的加速效果.

另外, 需要指出的是 ATDMC 方法只适用于情节式任务. 对于连续任务, 有两种解决方案: 一种是引入可变的折扣因子 γ , 在恰当的时间步设置 γ 等于 0 (相当于主动结束任务), 从而转换为一个情节



(a) $\alpha = 0.2/16$ 的比较结果
(a) Result at $\alpha = 0.2/16$

(b) $\alpha = 0.4/16$ 的比较结果
(b) Result at $\alpha = 0.4/16$

(c) $\alpha = 0.6/16$ 的比较结果
(c) Result at $\alpha = 0.6/16$

(d) $\alpha = 0.8/16$ 的比较结果
(d) Result at $\alpha = 0.8/16$

图 10　平衡杆问题 (α 固定)
Fig. 10　Cart pole (fixed α)

任务, 但并不影响对应 MDP 的流程; 另外一种是采用平均奖赏设定 (Average-reward setting), MC 方法的目标需要重新用奖赏和奖赏的平均值的差值来定义. 更多的细节需要进一步研究.

References

1　Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 2018.

2　Liu Nai-Jun, Lu Tao, Cai Ying-Hao, Wang Shuo. A review of robot manipulation skills learning methods. *Acta Automatica Sinica*, 2019, **45**(3): 458−470
(刘乃军, 鲁涛, 蔡莹皓, 王硕. 机器人操作技能学习方法综述. 自动化学报, 2019, **45**(3): 458−470)

3　Polydoros A S, Nalpantidis L. Survey of model-based reinforcement learning: applications on robotics. *Journal of Intelligent & Robotic Systems*, 2017, **86**(2): 153−173

4　Wang Ding. Research progress on learning-based robust adaptive critic control. *Acta Automatica Sinica*, 2019, **45**(6): 1031−1043
(王鼎. 基于学习的鲁棒自适应评判控制研究进展. 自动化学报, 2019, **45**(6): 1031−1043)

5　Wang Fei-Yue, Zheng Nan-Ning, Cao Dong-Pu, Clara M M, Li Li, Liu Teng. Parallel driving in CPSS: A unified approach for transport automation and vehicle intelligence. *IEEE/CAA Journal of Automatica Sinica*, 2017, **4**(4): 577−587

6　Du S S, Chen Jian-Shu, Li Li-Hong, Xiao Lin, Zhou Deng-Yong. Stochastic variance reduction methods for policy evaluation. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. Sydney, NSW, Australia: PMLR, 2017. 1049–1058

7　Lyu Dao-Ming, Liu Bo, Geist M, Dong Wen, Biaz S, Wang Qi. Stable and efficient policy evaluation. *IEEE Transactions on Neural Networks and Learning Systems*, 2019, **30**(6): 1831−1840

8　Dabney W C. Adaptive step-sizes for reinforcement learning [Ph. D. dissertation], University of Massachusetts Amherst, 2014

9　Young K, Wang Bao-Xiang, Taylor M E. Metatrace actor-critic: Online step-size tuning by meta-gradient descent for reinforcement learning control. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. Macau, China: ijcai.org, 2019. 4185–4191

10　Szepesvári C. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2010, **4**(1): 1−103

11　Busoniu L, Babuska R, De Schutter B, Ernst D. Reinforcement Learning and Dynamic Programming Using Function Approximators. Florida: CRC Press, 2017.

12　Boyan J A. Least-squares temporal difference learning. In: Proceedings of the 16th International Conference on Machine Learning. Bled, Slovenia: Morgan Kaufmann, 1999. 49–56

13　Sutton R S, Szepesvári C, Maei H R. A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in Neural Information Processing Systems*, 2008, **21**(21): 1609−1616

14　Sutton R S, Maei H R, Precup D, Bhatnagar S, Silver D,

Szepesvári C, et al. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: Proceedings of the 26th International Conference on Machine Learning. Montreal, Quebec, Canada: ACM, 2009. 993–1000

15 Dann C, Neumann G, Peters J. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 2014, **15**(1): 809−883

16 Van Seijen H, Sutton R S. True online TD (lambda). In: Proceedings of the 31st International Conference on Machine Learning. Beijing, China: JMLR.org, 2014. 692–700

17 Precup D, Sutton R S, Dasgupta S. Off-policy temporal-difference learning with function approximation. In: Proceedings of the 18th International Conference on Machine Learning. Williams College, Williamstown, MA, USA: Morgan Kaufmann, 2001. 417–424

18 Maei H R, Sutton R S. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In: Proceedings of the 3rd Conference on Artficial General Intelligence. Lugano, Switzerland: Atlantis Press, 2010. 91–96

19 Precup D, Sutton R S, Singh S. Eligibility traces for off-policy policy evaluation. In: Proceedings of the Seventeenth International Conference on Machine Learning. Stanford, CA, USA: Morgan Kaufmann, 2000. 759–766

20 Sutton R S. Learning to predict by the method of temporal differences. *Machine Learning*, 1988, **3**(1): 9−44

21 Van Seijen H, Mahmood A R, Pilarski P M, Machado M C, Sutton R S. True online temporal-difference learning. *The Journal of Machine Learning Research*, 2016, **17**(1): 5057−5096

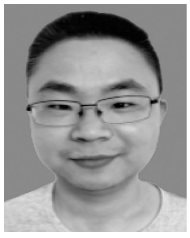
22 Maei H R. Gradient temporal-difference learning algorithms [Ph. D. dissertation], University of Alberta, 2014

23 White A, White M. Investigating practical linear temporal difference learning. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems. Singapore, Singapore: ACM, 2016. 494–502

24 Sutton R S, Mahmood A R, White M. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 2016, **17**(1): 2603−2631

25 Yu H. Weak convergence properties of constrained emphatic temporal-difference learning with constant and slowly diminishing stepsize. *The Journal of Machine Learning Research*, 2016, **17**(1): 7745−7802

26 Rummery G A. Problem solving with reinforcement learning [Ph. D. dissertation], University of Cambridge, 1995

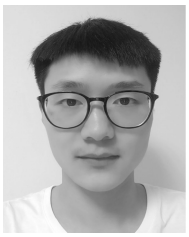


何 斌 苏州大学计算机科学与技术学院硕士研究生. 主要研究方向为强化学习.
E-mail: hebinyc@gmail.com
(**HE Bin** Master student at the School of Computer Science and Technology, Soochow University. His main research interest is reinforcement learning.)

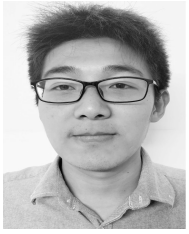
刘 全 苏州大学计算机科学与技术学院教授. 2004 年获得吉林大学博士学位. 主要研究方向为强化学习,深度学习,深度强化学习. 本文通信作者.
E-mail: quanliu@suda.edu.cn
(**LIU Quan** Professor at the School of Computer Science and Technology, Soochow University. He received his Ph. D. degree from Jilin University in 2004. His research interest covers reinforcement learning, deep learning and deep reinforcement learning. Corresponding author of this paper.)

张琳琳 苏州大学计算机科学与技术学院硕士研究生. 主要研究方向为强化学习.
E-mail: 20175227007@stu.suda.edu.cn
(**ZHANG Lin-Lin** Master student at the School of Computer Science and Technology, Soochow University. Her main research interest is reinforcement learning.)

时圣苗 苏州大学计算机科学与技术学院硕士研究生. 主要研究方向为深度强化学习.
E-mail: 20175227045@stu.suda.edu.cn
(**SHI Sheng-Miao** Master student at the School of Computer Science and Technology, Soochow University. His main research interest is deep reinforcement learning.)



陈红名 苏州大学计算机科学与技术学院硕士研究生. 主要研究方向为深度强化学习.
E-mail: 20174227007@stu.suda.edu.cn
(**CHEN Hong-Ming** Master student at the School of Computer Science and Technology, Soochow University. His main research interest is deep reinforcement learning.)



闫 岩 苏州大学计算机科学与技术学院硕士研究生. 主要研究方向为深度强化学习.
E-mail: 20165222006@stu.suda.edu.cn
(**YAN Yan** Master student at the School of Computer Science and Technology, Soochow University. His main research interest is deep reinforcement learning.)