计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201200173

基于逐次超松弛技术的 Double Speedy Q-Learning 算法

周 琴 罗 飞 丁炜超 顾春华 郑 帅
华东理工大学信息科学与工程学院 上海 200237
(zhouqin0822@163.com)

摘 要 Q-Learning 是目前一种主流的强化学习算法,但其在随机环境中收敛速度不佳,之前的研究针对 Speedy Q-Learning 存在的过估计问题进行改进,提出了 Double Speedy Q-Learning 算法。但 Double Speedy Q-Learning 算法并未考虑随机环境中存在的自循环结构,即代理执行动作时,存在进入当前状态的概率,这将不利于代理在随机环境中学习,从而影响算法的收敛速度。针对 Double Speedy Q-Learning 中存在的自循环结构,利用逐次超松弛技术对 Double Speedy Q-Learning 算法的 Bellman 算子进行改进,提出基于逐次超松弛技术的 Double Speedy Q-Learning 算法(Double Speedy Q-Learning based on Successive Over Relaxation,DSQL-SOR),进一步提升了 Double Speedy Q-Learning 算法的收敛速度。通过数值实验将 DSQL-SOR 与其他算法的实际奖励和期望奖励之间的误差进行对比,实验结果表明,所提算法比现有主流的算法 SQL 的误差低 0.6,比逐次超松弛算法 GSQL 低 0.5,这表明 DSQL-SOR 算法的性能较其他算法更优。实验同时对 DSQL-SOR 算法的可拓展性进行测试,当状态空间从 10 增加到 1000 时,每次迭代的平均时间增长缓慢,始终维持在 10^{-4} 数量级上,表明 DSQL-SOR 的可拓展性较强。

关键词:强化学习;Q-Learning;马尔可夫决策过程;逐次超松弛迭代法;自循环结构
中图法分类号 TP181

Double Speedy Q-Learning Based on Successive Over Relaxation

ZHOU Qin,LUO Fei,DING Wei-chao,GU Chun-hua and ZHENG Shuai
School of Information Science and Engineering,East China University of Science and Technology,Shanghai 200237,China

Abstract Q-Learning is a mainstream reinforcement learning algorithm atpresent,but its convergence speed is poor in random environment. Previous studies have improved the overestimation problem of Spee-dy Q-Learning,and have proposed Double Speedy Q-Learning algorithm. However,the Double Speedy Q-Learning algorithm does not consider the self-loop structure existing in the random environment,that is,the probability of entering the current state when the agent performs an action,which will not be conducive to the agent's learning in the random environment,thereby affecting the convergence speed of the algorithm. Aiming at the self-loop structure existing in Double Speedy Q-Learning,the Bellman operator of Double Speedy Q-Learning algorithm is improved by using successive over-relaxation technology,and the Double Speedy Q-Learning algorithm based on successive over relaxation (DSQL-SOR) is proposed to further improve the convergence speed of the Double Speedy Q-Learning algorithm. By using numerical experiments to compare the error between the actual rewards and expected rewards of DSQL-SOR and other algorithms,the experimental results show that the proposed algorithm has a lower error of 0.6 than the existing mainstream algorithm SQL,which is lower than the successive over-relaxation algorithm GSQL 0.5,indicating that the performance of the DSQL-SOR algorithm is better than other algorithms. The experiment also tests the scalability of the DSQL-SOR algorithm. When the state space is increased from 10 to 1000,the average time of each iteration increases slowly,always maintaining at the magnitude of 10^{-4},indicating that DSQL-SOR has strong scalability.

Keywords Reinforcement learning,Q-Learning,Markov decision process(MDP),Successive over relaxation(SOR),Self-loop structure

到稿日期:2020-12-20 返修日期:2021-05-09
基金项目:国家自然科学基金(61472139);上海汽车工业科技发展基金会产学研课题(1915)
This work was supported by the National Natural Science Foundation of China(61472139) and Shanghai Automotive Industry Science and Technology Development Foundation Industry-University-Research Project(1915).
通信作者:罗飞(luof@ecust.edu.cn)

1 引言

近年来，强化学习[1]受到了广泛的关注，在博弈论、运筹学、信息论、控制理论、统计等各个领域都有成功的应用。强化学习是一类强大的学习算法，通过与环境交互来学习最优策略，它将智能体与环境之间的相互作用建模为一个 MDP 无限折扣的奖励，并根据这个奖励来预测最优策略。Q-Learning[2]是一种无模型的强化学习算法，也可以被视为一种异步动态规划(Dynamic Programing，DP)方法[3]，它使得代理在不需要建立环境地图的情况下具有学习在马尔可夫环境中通过一系列动作的经验值执行最优动作的能力。Q-Learning 的学习过程与时间差分(Temporal Differences，TD)方法[4]相似：代理在特定状态尝试执行一个动作，并评估它收到的及时奖励或惩罚和状态值估计，通过反复尝试所有状态的所有动作，代理就能从长期折扣奖励中判断出总体上最优的动作。

在随机环境中，强化学习算法 Q-Learning 的性能表现非常差，这是因为 Q-Learning 使用最大行动值作为最大预期行动值的近似值，所以引入了正偏差，从而导致了过估计，影响了算法的整体的性能[5]。为此，许多研究者对 Q-Learning 算法进行了改进，我们之前的研究对 Speedy Q-Learning[6]算法进行分析，Speedy Q-Learning 算法利用一个冒进的学习率对 Q-Learning 算法进行改进，但由于其将当前 Q 值的估计函数作为历史 Q 值的估计，虽然整体上提升了智能体的收敛速度，但是同样存在过估计问题，使得算法在迭代初期的收敛速度较慢。对于这个问题，之前的研究基于 Double Q-Learning[7]算法中的双估计器可以改善智能体收敛速度的特性，提出了一种改进算法 Double Speedy Q-Learning。

Double Speedy Q-Learning[8]算法虽然提升了强化学习算法 Q-Learning 在随机环境中的整体性能，但并未针对代理在环境中的特殊情况进行分析改进。Q-Learning 算法在随机环境中的性能差的原因是，随机环境中环境的不确定性与随机性。本文针对 Double Speedy Q-Learning 在随机环境中存在的自循环结构，即代理在当前状态执行动作准备进入下一状态时，存在一定的停留在当前状态的概率，这将不利于代理在随机环境中的学习与推理，从而影响算法的整体性能。

本文针对上述提到的随机环境中的自循环结构问题，利用逐次超松弛技术[9]对 Double Speedy Q-Learning 的 Bellman 算子进行改进，提出了基于逐次超松弛技术的 Double Speedy Q-Learning 算法，进一步提升了 Double Speedy Q-Learning 在随机环境中的整体性能。

2 相关工作

Q-Learning 是最具有代表性的强化学习算法之一，许多研究者将它应用到强化学习和人工智能问题中，但早期的 Q-Learning 存在许多的缺陷，应用范围有限[10]，为此许多研究者对 Q-Learning 的缺陷进行了分析及改进。本文将改进的算法分为两类：单智能体 Q-Learning 改进算法和多智能体 Q-Learning 改进算法。在单智能体 Q-Learning 改进算法中，最优 Q 值的概念定义为一个智能体在随机环境下自己预期收益的最大值，在多智能体 Q-Learning 改进算法中，最优 Q 值取决于其他智能体的策略[11]。

单智能体 Q-Learning 改进算法的典型代表主要有 Deep Q-Learning，Incremental multistep Q-Learning，Hierarchical Q-Learning，Double Q-Learning 和 Speedy Q-Learning 等。其中，Deep Q-Learning 是最流行的单智能体 Q-Learning 改进算法。深度强化学习模型[12]由 Vnih 等于 2013 年首次提出，其利用深度学习能够从原始观测数据中提取高级特征的能力，提出了第一个能够利用强化学习直接从高维感官输入成功学习控制策略的深度学习模型，弥补了强化学习在高维观测数据领域性能表现不稳定的缺陷，AlphaGo[13]在 2017 年战胜了许多顶端围棋高手，引起了学术界的轰动。Incremental multistep Q-Learning[14]算法由 Peng 等提出，通过将基于动态规划的强化学习方法 Q-Learning 与 TD(λ)回报估计过程相结合，参数(λ)用于在整个动作序列中分配积分，从而提高学习速度，这有助于缓解粗状态空间量化的非马尔可夫效应。此外，由于 Q-Learning 使用最大行动值作为最大预期行动值的近似值，因此引入了正偏差，从而导致了过估计。为了解决 Q-Learning 的过估计问题，Van 等[7]通过将双估计器应用到 Q-Learning 中，提出了一种新的离线策略强化学习算法——Double Q-Learning，并在网格世界中进行实验，实验结果表明，就平均回报而言，Double Q-Learning 的表现更好。

多智能体 Q-Learning 改进算法的典型代表有 Nash Q-Learning、合作多代理 Q-Learning、加权政策 Q-Learning、基于种群的 Q-Learning 等。其中，Nash Q-Learning[15]是一种非合作的多智能体 Q-Learning 改进算法，在该算法中，代理从随机动作开始试图学习一个平衡 Q 值。为了达到这个目的，Nash Q-Learning 代理必须维护其他代理的 Q 值模型(Q 值为代理在每种状态下采取行动的期望)，并使用该信息来更新自己的 Q 值。与 Nash Q-Learning 不同，文献[16]提出了一种合作多代理 Q-Learning 算法，其同时也是一种基于种群的 Q-Learning 算法。该算法利用奖励机制和人工蚂蚁来确定全局最优路由选择，用于解决水下无线传感器网络(Underwater Wireless Sensors Networks，UWSNs)中的能效问题和链路不稳定问题。合作多代理 Q-Learning 算法的优势在于，在一个多智能体的环境中，由于智能体之间是相互合作的，因此可以有效地发现并评价某个特定的行动。

本文将针对单智能体 Q-Learning 算法——Double Speedy Q-Learning 在随机环境中存在的自循环结构问题，利用逐次超松弛技术对其 Bellman 算子进行改进，从而提升 Double Speedy Q-Learning 算法的整体性能。

3 关键技术

本文利用逐次超松弛技术来解决 Double Speedy Q-Learning 算法在随机环境中的自循环问题。本节首先对 Double Speedy Q-Learning 算法进行介绍，然后介绍了 Double Speedy Q-Learning 在随机环境中存在的自循环问题，最后介绍逐次超松弛技术以及如何通过逐次超松弛技术来解决 Double Speedy Q-Learning 算法在随机环境中存在的自循环问题。

3.1 Double Speedy Q-Learning 算法

Double Speedy Q-Learning(DSQL)算法[8]由 Zheng 等提出，其目标是解决 Speedy Q-Learning 算法存在的过估计问题。该算法基于 Double Q-Learning 的双估计器可以改善智能体收敛速度的特性，将双估计器应用到 Speedy Q-Learning 中，利用两个 Q 表(Q^A和Q^B)分离最优动作和最大 Q 值的选择，改善了 Speedy Q-Learning 算法在迭代初期的学习策略，提升了 Speedy Q-Learning 算法的整体收敛速度。DSQL 算法对应的 Bellman 公式为：

$$Q(s_t,a_t)=\gamma\sum_{t=1}^{M}P(s_{t+1}|s_t,a_t)\max_{a_{t+1}\in A}Q(s_{t+1},a_{t+1})+R(s_t,a_t) \tag{1}$$

DSQL 算法的最优策略可表示为：

$$\pi(s_t)=\arg\max_{a\in A}Q(s_t,a_t) \tag{2}$$

DSQL 算法的更新公式如下：

$$\delta_1=R(s,a)+\gamma Q_{k-1}^B(s',a_A^*) \tag{3}$$

$$\delta_2=\gamma(Q_k^B(s',a_A^*)-Q_{k-1}^B(s',a_A^*)) \tag{4}$$

$$Q_k^A(s,a)=(1-\alpha)Q_k^A(s,a)+\alpha\delta_1+(1-\alpha)\delta_2 \tag{5}$$

其中，δ_1表示上一迭代步骤采样下一状态的经验 Bellman 最优算子，δ_2表示当前迭代步骤采样下一状态的经验 Bellman 最优算子与上一迭代步骤采样下一状态的经验 Bellman 最优算子之差。$Q_k^A(s,a)$表示第 k 次迭代时，状态-动作对(s,a)在Q^A表中对应的 Q 值，$Q_k^B(s,a)$表示第 k 次迭代时，状态-动作对(s,a)在Q^B表中对应的 Q 值。$a_A^*=\arg\max_{a'}Q^A(s',a')$，$s'$表示下一状态，$k$ 表示迭代次数，$R(s,a)$表示在状态 s 时采取动作 a 获得的奖励，a 表示学习率，γ 表示折扣因子。Double Speedy Q-Learning 算法的伪代码如算法 1 所示。

算法 1　Double Speedy Q-Learning 算法

输入：Q_0，γ，K，P

输出：Q_K

1. 初始化：$Q_{-1}\leftarrow Q_0$ / * 初始化 Q 表 * /；
2. for k←0 to K−1 do
3.　$\alpha_k\leftarrow 1/(k+1)$
4.　for each (i,a)∈S×A do
5.　　generate next-state sample s_
6.　　if update Q^A
7.　　$a^*\leftarrow\arg\max_a Q_k^A(s\_,a)$
8.　　$\delta_1\leftarrow R+\max_a Q_{k-1}^B(s,a^*)-Q_k^A(s,a)$
9.　　$\delta_2\leftarrow\gamma(\max_a Q_k^B(s,a^*)-\max_a Q_{k-1}^B(s,a^*))$
10.　　$Q_{k+1}^A(s,a)\leftarrow Q_k^A(s,a)+\alpha\delta_1+(1-\alpha)\delta_2$
11.　if update Q^B
12.　　$a^*\leftarrow\arg\max_a Q_k^B(s\_,a)$
13.　　$\delta_1\leftarrow R+\max_a Q_{k-1}^A(s,a^*)-Q_k^B(s,a)$
14.　　$\delta_2\leftarrow\gamma(\max_a Q_k^A(s,a^*)-\max_a Q_{k-1}^A(s,a^*))$
15.　　$Q_{k+1}^B(s,a)\leftarrow Q_k^B(s,a)+\alpha\delta_1+(1-\alpha)\delta_2$

3.2 马尔可夫决策过程和自循环结构

在介绍 Double Speedy Q-Learning 在随机环境中的自循环结构之前，先介绍马尔可夫决策过程(Markov Decision Process，MDP)。一个 MDP 可以用一个 5 元组$(S,A,\boldsymbol{P},\boldsymbol{R},\gamma)$来表示，其中，$S$ 表示状态集，A 表示动作集，$\boldsymbol{P}$ 表示转移概率矩阵，$\boldsymbol{P}(j|i,a)$表示从状态 i 采取动作 a 到状态 j 的概率，$\boldsymbol{R}$ 表示奖励矩阵，$\boldsymbol{R}(i,a)$表示在状态 i 采取动作 a 时获得的奖励，γ 表示折扣因子，且 $\gamma\in(0,1)$。一个强化学习问题可以用 MDP 的框架进行建模，在折现奖励马尔可夫决策过程(MDP)中，目标是寻找最优价值函数，即与最优策略对应的价值函数。该问题可以理解为利用值迭代求解如下的 Bellman 方程：

$$E_{\pi(s_t)}[\sum_{t=0}^{\infty}\gamma^t R(s_t,s_{t+1})|s_0=i] \tag{6}$$

其中，s_t为 t 时刻时代理所处的状态，$\pi(s_t)$表示状态 i 中最大化折扣奖励目标所要采取的行动。这样，当模型信息已知时，我们可以通过数值计算最优值函数。在许多实际应用中，我们无法访问模型信息，但是我们可以使用所访问的状态和奖励样本找到最佳价值函数和相应的策略，Q-Learning 及其改进算法是从样本计算最佳策略和价值函数的算法之一。

在 MDP 中，代理从状态 i 采取动作 a 到下一状态时，存在使得代理的下一状态仍然为 i 的概率，即存在 $\boldsymbol{P}(j|i,a)>0$，使得代理从状态 i 采取动作 a 回到状态 i。自循环结构的存在，使得代理在训练过程中学习到了一定的无效经验，这些无效经验对代理任务的完成没有任何作用，降低了算法的整体性能。

3.3 逐次超松弛技术

逐次超松弛(Successive Over Relaxation，SOR)迭代法是一种经典的迭代算法。它是为了解决大规模系统的线性等式而提出来的，在高斯法基础上为提高收敛速度，采用加权平均而得到的新算法[17]。在逐次超松弛迭代法中进行迭代时，没有对a_{ii}部分进行计算，如式(7)所示。利用这一点，我们可以用它来解决 MDP 中存在的自循环结构问题。在 SOR 理论的发展中，研究人员一直在研究能够使得 SOR 收敛的超松弛因子 ω 的范围，并在可能的情况下求出 ω 的最优值[9]。使用超松弛迭代法的关键在于选取合适的松弛因子，如果松弛因子选取合适，则会大大缩短计算时间。

$$x_i^{k+1}=x_i^k+\frac{\omega}{a_{ii}}(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{k+1}-\sum_{j=i+1}^{n}a_{ij}x_j^k) \tag{7}$$

文献[18]证明了 SOR 可以用于求解一个折现马尔可夫决策问题，并确定了 MDP 模型的最优超松弛因子ω^*(如式(8)所示)和最小收缩因子 $1-\omega^*+\gamma\omega^*$，最小收缩因子 $1-\omega^*+\gamma\omega^*$最终至多等于折扣因子 γ(见定理 1)。

$$\omega^*=\min_{i,j}\frac{1}{1-\gamma p(i|i,j)} \tag{8}$$

其中，γ 表示折扣因子($0<\gamma\leqslant 1$)，$\boldsymbol{P}(i|i,j)$表示在状态 i 采取动作 j 回到状态 i 的概率($0<\boldsymbol{P}(i|i,j)<1$)，因此可以推断$\omega^*>1$。

定理 1　$1-\omega^*+\gamma\omega^*\leqslant\gamma$

证明：

$1-\omega^*+\gamma\omega^*\leqslant\gamma$

$<=>1-\omega^*+\gamma\omega^*-\gamma\leqslant 0$

$<=>(\gamma-1)(\omega^*-1)\leqslant 0$

$\because 0<\gamma\leqslant 1,\therefore\gamma-1\leqslant 0$

$\because \omega^* > 1, \therefore \omega^* - 1 \geqslant 0$

$\therefore (\gamma - 1)(\omega^* - 1) \leqslant 0$

$\therefore 1 - \omega^* + \gamma\omega^* \leqslant \gamma$

证毕。

4 基于逐次超松弛技术的 Double Speedy Q-Learning 算法

4.1 Double Speedy Q-Learning 算法的概述

基于 Double Speedy Q-Learning 算法，考虑存在这样一种特殊结构的 MDP，即对于动作空间中的每个动作，状态空间中的每个状态都有正的自循环概率。这种结构可以利用对 Bellman 操作算子的逐次超松弛技术来开发。对于具有这种结构的 MDP，可以选择一个超松弛因子 ω，这样在有限时间内该算法的性能优于 DSQL。本文利用逐次超松弛技术，对 Double Speedy Q-Learning 的 Bellman 方程进行改进，提出了一种新的算法，基于逐次超松弛技术的 Double Speedy Q-Learning 算法(DSQL-SOR)，进一步提升了 Double Speedy Q-Learning 算法的性能。改进的 Bellman 方程为：

$$Q(s,a) = \omega\tau + (1-\omega)\max_{c\in A} Q(s,c) \tag{9}$$

$$\tau = \boldsymbol{R}(s,a) + \alpha \sum_{s'=1}^{M} \boldsymbol{P}(s'|s,a)\max_{b\in A} Q(s',b) \tag{10}$$

DSQL-SOR 的更新公式如下(当更新Q^A时)：

$$\delta_1 = \omega\boldsymbol{R} + (1-\omega+\omega\gamma)\max_{a} Q_{k-1}^{B}(s,a^*) - Q_k^A(s,a) \tag{11}$$

$$\delta_2 = \max_{a\in A} Q_k^B(s,a^*) - \max_{a\in A} Q_{k-1}^B(s,a^*) \tag{12}$$

$$Q_{k+1}^A(s,a) = Q_k^A(s,a) + \alpha\delta_1 + (1-\alpha)(1-\omega+\omega\gamma)\delta_2 \tag{13}$$

其中，δ_1表示上一迭代步骤采样下一状态的经验 Bellman 最优算子，δ_2表示当前迭代步骤采样下一状态的经验 Bellman 最优算子与上一迭代步骤采样下一状态的经验 Bellman 最优算子之差，$\omega(0<\omega<\omega^*)$表示超松弛因子，$R$ 表示在当前迭代步骤获得的奖励，$1-\omega^*+\gamma\omega^*$表示最小收缩因子，$\max_a Q_k^A(s,a)$表示第 k 次迭代时，状态-动作对(s,a)在Q^A表中对应的最大 Q 值，$Q_k^A(s,a)$表示第 k 次迭代时，状态-动作对(s,a)在Q^A表中对应的 Q 值，$a^* = \arg\max_a Q_k^A(s\_,a)$，$s_$表示下一状态。

Double Speedy Q-Learning based on Successive Over Relaxation 算法的伪代码如算法 2 所示。算法 2 开始前先对 Q 表 Q_0、折扣因子 γ、超松弛因子 ω、迭代次数 K、概率矩阵 $\boldsymbol{P}$ 以及奖励矩阵 $\boldsymbol{R}$ 进行初始化。随后，根据式(8)计算出最优超松弛因子ω^*，ω^*在算法中只需计算一次，由于ω^*只与概率矩阵 $\boldsymbol{P}$ 和折扣因子 γ 有关，而概率矩阵 $\boldsymbol{P}$ 和折扣因子 γ 初始化后在整个算法执行过程中保持不变，因此可以保证ω^*在算法的所有迭代中都是最优的。该算法采用线性学习率 $\alpha = 1/(1+k)$。每次迭代开始时，对每个$(s,a)\in S\times A$生成下一个状态 $s\_$。该算法采用两个 Q 表(Q^A和Q^B)将最优动作与 Q 值的选择分离，每次更新 Q 值时，随机选择一个 Q 表进行更新。若对Q^A表进行更新，则根据Q^A选择最优动作a^*，并使用Q^B的 Q 值对Q^A表的 Q 值进行更新；若对Q^B表进行更新，则根据Q^B选择最优动作a^*，并使用Q^A的 Q 值对Q^B表的 Q 值进行更新。

算法 2 Double Speedy Q-Learning based on Successive Over Relaxation 算法

输入：$Q_0, \gamma, \omega, K, \boldsymbol{P}$

输出：Q_K

1. 初始化：$Q_{-1} \leftarrow Q_0$ / * 初始化 Q 表 * /；
2. compute the optimal ω^*
3. for k←0 to K−1 do
4. $\quad \alpha_k \leftarrow 1/(k+1)$
5. $\quad$ for each $(i,a)\in S\times A$ do
6. $\quad\quad$ generate next-state sample s_
7. $\quad\quad$ if update Q^A
8. $\quad\quad\quad a^* \leftarrow \arg\max_a Q_k^A(s\_,a)$
9. $\quad\quad\quad \delta_1 \leftarrow \omega^* R + (1-\omega^*+\omega^*\gamma)\max_a Q_{k-1}^B(s,a^*) - Q_k^A(s,a)$
10. $\quad\quad\quad \delta_2 \leftarrow (1-\omega^*+\omega^*\gamma)\max_a Q_k^B(s,a) - \max_k Q_{k-1}^B(s,a^*)$
11. $Q_{k+1}^A(s,a) \leftarrow Q_k^A(s,a) + \alpha\delta_1 + (1-\alpha)\delta_2$
12. $\quad\quad$ if update Q^B
13. $\quad\quad\quad a^* \leftarrow \arg\max_a Q_k^B(s\_,a)$
14. $\quad\quad\quad \delta_1 \leftarrow \omega^* \mathbf{R} + (1-\omega^*+\omega^*\gamma)\max Q_{k-1}^A(s,a^*) - Q_k^B(s,a)$
15. $\quad\quad\quad \delta_2 \leftarrow (1-\omega^*+\omega^*\gamma)\max_a Q_k^A(s,a) - \max_a Q_{k-1}^A(s,a^*)$
16. $\quad\quad\quad Q_{k+1}^B(s,a) \leftarrow Q_k^B(s,a) + \alpha\delta_1 + (1-\alpha)\delta_2$

DSQL-SOR 算法与 DSQL 算法的主要区别在于，在每次迭代开始之前，DSQL-SOR 算法需要根据式(8)计算出超松弛因子 ω 的最优值，迭代开始之后，根据改进的 Bellman 算子进行 Q 表的更新。

4.2 DSQL-SOR 算法的收敛性分析

本节将对 DSQL-SOR 的收敛性进行分析。首先，本文提出引理 1，得出$H_w Q^* = Q^*$；其次根据得出的引理 1 及定理 3 对定理 2 进行了证明，从而得出结论：DSQL-SOR 算法迭代 Q_n 几乎可以肯定收敛到最优 Q 值 Q^*。

引理 1 $H_w: \mathbb{R}^{S\times A} \to \mathbb{R}^{S\times A}$是一个最大收缩范数，并且$Q^*$是$H_w$的唯一不动点。其中，$H_w Q(s,a) = \omega[\boldsymbol{R}(s,a) + \gamma\sum_{s'=1}^{M} P(s'|s,a)\max_{b\in A} Q(s',b)] + (1-\omega)\max_{c\in A} Q(s,c)$，$Q^*(s,a) = \omega[\boldsymbol{R}(s,a) + \gamma\sum_{s'=1}^{M} \boldsymbol{P}(s'|s,a)\max_{b\in A} Q^*(s',b)] + (1-\omega)\max_{c\in A} Q^*(s,c)$。

证明：

$$\begin{aligned}
&|(H_w\boldsymbol{P} - H_w Q)(s,a)| \\
&= |\omega\gamma\sum_{s'=1}^{M} \boldsymbol{P}(s'|s,a)(\max_{b\in A} \boldsymbol{P}(s',b) - \max_{b\in A} Q(s',b)) + \\
&\quad (1-\omega)(\max_{c\in A} \boldsymbol{P}(s,c) - \max_{c\in A} Q(s,c))| \\
&= |\omega\gamma\sum_{s'=1,s'\neq s}^{M} \boldsymbol{P}(s'|s,a)(\max_{b\in A} \boldsymbol{P}(s',b) - \max_{b\in A} Q(s',b)) + \\
&\quad (1-\omega+\omega\gamma\boldsymbol{P}(s|s,a))(\max_{c\in A} \boldsymbol{P}(s,c) - \max_{c\in A} Q(s,c))| \\
&\leqslant \omega\gamma\sum_{s'=1,s'\neq s}^{M} \boldsymbol{P}(s'|s,a)|(\max_{b\in A} \boldsymbol{P}(s',b) - \max_{b\in A} Q(s',b)| + \\
&\quad (1-\omega+\omega\gamma\boldsymbol{P}(s|s,a))|(\max_{c\in A} \boldsymbol{P}(s,c) - \max_{c\in A} Q(s,c)| \\
&\leqslant \omega\gamma\sum_{s'=1,s'\neq s}^{M} \boldsymbol{P}(s'|s,a)|(\max_{b\in A} \boldsymbol{P}(s',b) - \max_{b\in A} Q(s',b)| + \\
&\quad (1-\omega+\omega\gamma\boldsymbol{P}(s|s,a))|(\max_{b\in A} \boldsymbol{P}(s,b) - \max_{b\in A} Q(s,b))| \\
&\leqslant (\omega\gamma+1-\omega)\,\|\boldsymbol{P}-Q\| \\
&\therefore |(H_w\boldsymbol{P} - H_w Q)(s,a)| \leqslant (\omega\gamma+1-\omega)\,\|\boldsymbol{P}-Q\|
\end{aligned}$$

因此，H_w 在收缩因子为 $\omega\gamma+1-\omega$ 时是一个最大范数收缩，并且 Q^* 是唯一的收敛点。

我们应用文献[19]的定理(即定理 2)来证明 DSQL-SOR 算法迭代对最优 Q 值的收敛性(见定理 3)。

定理 2　随机迭代过程 $\Delta_{n+1}(x)=(1-\alpha_n(x))\Delta_n(x)+\alpha_n(x)F_n(x)$ 在以下 3 个条件下，当 $n\to\infty$ 时以概率 1 收敛至 0。

(1) $0\leqslant\alpha_n(x)\leqslant1$，$\sum_{n=1}^{\infty}\alpha_n(x)=\infty$，$\sum_{n=1}^{\infty}\alpha_n(x)^2<\infty$

(2) $\|E[F_n]\|\leqslant\beta\|\Delta_n\|$，$\beta<1$

(3) $var\{F_n(x)\}\leqslant C(1+\|\Delta_n\|^2)$，$C>0$

定理 3　给定一个有限 $MDP(S,A,P,r,\gamma)$，并且 $r\leqslant\boldsymbol{R}$，$\forall s,a,s'\in(S\times A\times S)$，DSQL-SOR 的更新公式为：

$$Q_{n+1}(s,a)=(1-\alpha_n(s,a))Q_n(s,a)+\alpha_n(s,a)(\omega r(s,a,s')+\gamma'\max Q_{n-1}(s,a'))+(1-\alpha_n(s,a))\gamma'[\max Q_n(s,a)-\max Q_{n-1}(s,a')],\gamma'=\omega\gamma+1-\omega$$

以概率 1 收敛至最优 Q 值，只要满足对于所有的 (s,a)，$\sum_{n=1}^{\infty}\alpha_n(s,a)=\infty$，$\sum_{n=1}^{\infty}\alpha_n(s,a)^2<\infty$。

证明：令 $\Delta_n(s,a)=Q_n(s,a)-Q^*(s,a)$，则更新公式可重写为：

$$\Delta_{n+1}(s,a)=(1-\alpha_n(s,a))\Delta_n(s,a)+\alpha_n(s,a)[\omega r(s,a,s')+\gamma'\max Q_{n-1}(s,a')]+(1-\alpha_n(s,a))\gamma'[\max Q_n(s,a)-\max Q_{n-1}(s,a')]-Q^*(s,a)$$

令

$$F_n(s,a)=\alpha_n(s,a)[\omega r(s,a,s')+\gamma'\max Q_{n-1}(s,a')]+(1-\alpha_n(s,a))\gamma'[\max Q_n(s,a)-\max Q_{n-1}(s,a')]-Q^*(s,a)$$

则 $E[F_n(s,a)]=\sum_{s'=1}^{M}p(s'|s,a)[\alpha_n(s,a)(\omega r(s,a,s')+\gamma'\max Q_{n-1}(s,a'))+(1-\alpha_n(s,a))\gamma'(\max Q_n(s,a)-\max Q_{n-1}(s,a'))]-Q^*(s,a)=(H_wQ_n)(s,a)-Q^*(s,a)$。

由引理 1 可知：$H_wQ^*=Q^*$，因此：

$$|E[F_n(s,a)]|\leqslant(\omega\gamma+1-\omega)\|Q_n-Q^*\|=(\omega\gamma+1-\omega)\|\Delta_n\|$$

最后，

$$\begin{aligned}
&var[F_n(s,a)]\\
&=E[(F_n(s,a)-(H_wQ_n)(s,a)+Q^*(s,a))^2]\\
&=E[(\alpha_n(s,a)[\omega r(s,a,s')+\gamma'\max Q_{n-1}(s,a')]+(1-\alpha_n(s,a))\gamma'[\max Q_n(s,a)-\max Q_{n-1}(s,a')]-(H_wQ_n)(s,a))^2]\\
&\leqslant E[(\alpha_n(s,a)[\omega r(s,a,s')+\gamma'\max Q_{n-1}(s,a')]+(1-\alpha_n(s,a))\gamma'[\max Q_n(s,a)-\max Q_{n-1}(s,a')])^2]\\
&\leqslant 3(\alpha_n(s,a)^2\omega^2r(s,a,s')^2+\alpha_n(s,a)^2\gamma'^2\max Q_{n-1}(s,a')^2+(1-\alpha_n(s,a))^2\gamma'^2(\max Q_n(s,a)-\max Q_{n-1}(s,a'))^2)\\
&=3(\alpha_n(s,a)^2\omega^2r(s,a,s')^2+\alpha_n(s,a)^2\gamma'^2\|Q_{n-1}\|^2+(1-\alpha_n(s,a))^2\gamma'^2(\|Q_n\|-\|Q_{n-1}\|)^2)\\
&=3(\alpha_n(s,a)^2\omega^2r(s,a,s')^2+\alpha_n(s,a)^2\gamma'^2[\|Q^*\|+\|\Delta_n\|]^2+(1-\alpha_n(s,a))^2\gamma'^2(\|\Delta_n\|-\|\Delta_{n-1}\|)^2)\\
&=3(\alpha_n(s,a)^2\omega^2r(s,a,s')^2+\alpha_n(s,a)^2\gamma'^2[\|Q^*\|+\|\Delta_n\|]^2-2(1-\alpha_n(s,a))^2\|\Delta_n\|\|\Delta_{n-1}\|+(1-\alpha_n(s,a))^2\gamma'^2\|\Delta_{n-1}\|^2+(1-\alpha_n(s,a))^2\|\Delta_n\|^2)\\
&\leqslant C(1+\|\Delta_n\|^2)
\end{aligned}$$

其中，$C=\max\{(\alpha_n(s,a)^2\omega^2r(s,a,s')^2+\alpha_n(s,a)^2[\|Q^*\|+\|\Delta_n\|]-2(1-\alpha_n(s,a))^2\gamma'^2\|\Delta_n\|\|\Delta_{n-1}\|+(1-\alpha_n(s,a))^2\gamma'^2\|\Delta_{n-1}\|^2)/(1-\alpha_n(s,a))^2\}$

$\therefore var[F_n(s,a)]\leqslant C(1+\|\Delta_n\|^2)$

综上，根据定理 2 和定理 3，Δ_n 以概率 1 收敛至 0，即 DSQL-SOR 算法迭代 Q_n 几乎可以肯定收敛到最优 Q 值 Q^*。

5　实验设计与结果分析

本节对 DSQL-SOR 算法进行实验评估。首先对本实验的环境及配置进行说明，其次将所提算法与主流强化学习算法和逐次超松弛算法进行对比。接下来通过增大空间状态基数对所提出算法的可扩展性进行测试。最后比较了不同超松弛因子 ω 下 DSQL-SOR 算法的性能。

5.1　实验环境及配置

本实验通过将算法在随机的 MDPs 模型上进行数值比较来检验算法的性能，评价指标为算法随迭代次数变化的平均误差，其定义为 100 个 MDPs 最优值函数与其基于不同算法给出的 Q 函数的估计之间的差值，即：

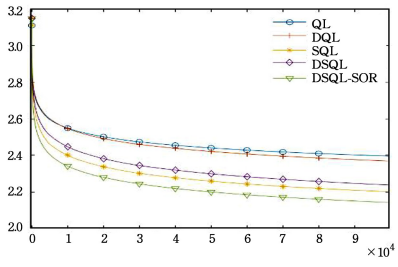
$$E_n=\frac{1}{100}\sum_{1}^{100}|V_k^*-\max_a Q_{k,n}(.,a)| \tag{14}$$

此外，本实验使用 python 的 mdptoolbox 来生成 100 个独立的 MDPs，每个 MDP 具有 S(10/20/50/100/500/1 000)个状态以及 5 个动作，并满足条件 $\boldsymbol{P}(i|i,a)>0$($\boldsymbol{P}(i|i,a)>0$ 确保了本实验环境中 MDPs 存在自循环结构)。本实验中所有比较算法保持相同的步长，并运行相同的迭代次数(即 100 000 次)。在迭代过程中，所有算法使用相同的参数。其中，学习率 $\alpha=1/(k+1)$，$k=1,2,\cdots,100\,000$，折扣因子 $\gamma=0.6$。为了更好地对比各种算法之间的性能，各种算法使用同一个始化概率矩阵 $\boldsymbol{P}$ 和奖励矩阵 $\boldsymbol{R}$。

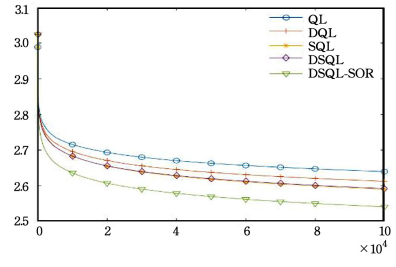
5.2　DSQL-SOR 与主流强化学习算法的对比

本节将 DSQL-SOR 与另外 4 种传统强化学习算法 Q-Learning(QL)，Double Q-Learning(DQL)，Speedy Q-Learning(SQL)和 Double Speedy Q-Learning(DSQL)在随机构造的 MDPs 上进行比较，并考虑了在不同空间状态基数($S=10$ 和 $S=20$)上，所提算法 DSQL-SOR 与其他算法的性能比较。

如图 1 所示，在状态数为 10 和 20 的情况下，随着迭代次数的增加，各种算法的平均误差均在减小。如图 1(a)所示，所提算法在整个迭代步上的误差始终保持最小(即2.14)，而 QL，DQL，SQL，DSQL 算法的误差分别为 2.40，2.37，2.20，2.24。对比图 1(a)和图 1(b)可知，空间状态基数越大，DSQL-SOR 算法的优势就越明显。这是因为本文考虑 MDPs 过程的自循环结构，避免了代理学习无用的经验，从而提升了算法的整体性能。



(a)$S=10$



(b)$S=20$

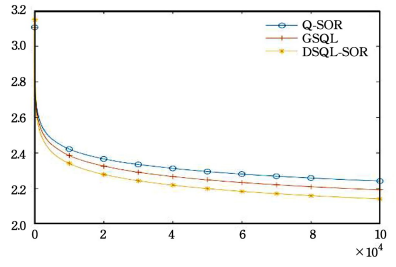
图 1 DSQL-SOR 和传统 Q-Learning 算法的平均误差

Fig. 1 Average error of DSQL-SOR and other traditional Q-Learning algorithm

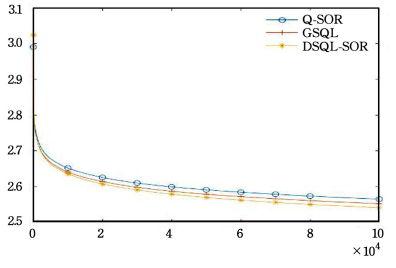
5.3 DSQL-SOR 与逐次超松弛算法的对比

为了将 DSQL-SOR 算法与同类逐次超松弛算法进行对比，本节将所提算法 DSQL-SOR 与 Generalized Speedy Q-Learning(GSQL)[20] 和 Q-Learning SOR(Q-SOR)[21] 在随机 MDPs 上进行对比，评估 DSQL-SOR 在逐次超松弛算法中的优越性。同样考虑了在不同空间状态基数($S=10$ 和 $S=20$)上所提算法 DSQL-SOR 与其他算法的性能比较。

如图 2 所示，随着迭代次数的增加，Q-SOR，GSQL 和 DSQL-SOR 算法的误差均在减小，但无论是在状态数为 10 或 20 的情况下，本文提出的 DSQL-SOR 算法的误差始终保持最小。



(a)$S=10$



(b)$S=20$

图 2 DSQL-SOR 和逐次超松弛算法的平均误差

Fig. 2 Average error of DSQL-SOR and successive over relaxation algorithm

如图 2(a)所示，所提算法在整个迭代步骤上的误差始终保持最小(即 2.14)，而 Q-SOR 和 GSQL 算法的误差分别为 2.24，2.19。这是由于本文是基于 DSQL 算法考虑了自循环结构，DSQL 是基于 QL 以及 SQL 算法，考虑了收敛速度慢以及过估计等缺陷的改进算法，因此 DSQL 算法的性能在 QL 和 SQL 的基础上得到了提升。

5.4 DSQL-SOR 算法的可扩展性测试

本节通过考虑状态空间基数为 10，50，100，500 和 1 000 的 MDPs，评估了所提算法的可扩展性。

表 1 列出了对于不同状态空间基数的 MDPs 和 DSQL-SOR 算法每次迭代的平均时间，可以看到，当状态空间从 10 增加到 1000 时，DSQL-SOR 算法每次迭代的平均时间增长缓慢，始终维持在10^{-4}数量级上，可见 DSQL-SOR 的可拓展性较强。这是因为基于 DSQL-SOR 算法的改进提升了算法的性能，使得其在较大空间状态基数的 MDPs 上也能较快地收敛。因此，可以推断出 DSQL-SOR 算法的可拓展性较高。

表 1 DSQL-SOR 算法的拓展性

Table 1 DSQL-SOR scalability

State Number	Average Time per iteration /s
10	3.71×10^{-4}
50	3.94×10^{-4}
100	4.16×10^{-4}
500	6.39×10^{-4}
1 000	9.25×10^{-4}

5.5 不同超松弛因子 ω 下 DSQL-SOR 算法的性能

为了检验最优超松弛因子对算法性能的影响，本节测试了根据式(8)计算出的超松弛因子ω^*和其他超松弛因子下 DSQL-SOR 算法的性能。

图 3 给出了对于选择不同超松弛因子 ω，($0<\omega<\omega^*$)时，DSQL-SOR 算法的性能。本实验中计算得出的最优超松弛因子$\omega^*=1.3$，我们测试了 $\omega=1.3$，$\omega=1.1$，$\omega=0.9$，$\omega=0.7$ 时，DSQL-SOR 算法的性能。图 3 中，ω 值越接近ω^*，DSQL-SOR 算法的平均误差就越小。可以得知，根据式(8)计算得出的超松弛因子能够使得 DSQL-SOR 算法的性能得到最大提升。

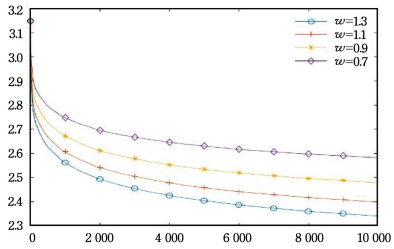


图 3 不同 ω 值下 DSQL-SOR 算法的平均误差

Fig. 3 Average error of DSQL-SOR with different value of ω

结束语 首先对 Q-Learning 及其衍生算法进行研究并分析其缺陷与优势。Q-Learning 由于具有能够与环境交互这一特性，在许多领域中得到了广泛的应用，但是它在随机环境中的性能不佳，因此提出对 Q-Learning 算法进行进一步改进的需求。然后充分利用逐次超松弛迭代法收敛速度快的

特点，通过超松弛因子 ω 对 Double Speedy Q-Learning 算法的 Bellman 算子进行改进，提出了一种改进的 Q-Learning 算法，即 Double Speedy Q-Learning based on Successive Over Rlaxation。最后，我们通过数值实验将 DSQL-SOR 算法与其他改进算法的性能进行比较，实验结果表明，本文提出的 DSQL-SOR 算法的性能表现最好，并且可拓展性强。在本文算法的迭代过程中，概率矩阵保持不变，因此最优超松弛因子 ω 也保持不变，为了更好地适应环境，未来考虑在迭代过程中对概率矩阵进行更新，从而使超松弛因子能够自适应环境，进一步提升算法的性能。

参 考 文 献

[1] XIONG H，DIAO X. Safety robustness of reinforcement learning policies：A view from robust control[J]. Neurocomputing，2021，422：12-21.

[2] WATKINS C，DAYAN P. Technical note：Q-learning[J]. Machine Learning，1992，8314：279-292.

[3] SONG Z，HOFMANN H，LI J，et al. Optimization for a hybrid energy storage system in electric vehicles using dynamic programing approach[J]. Applied Energy，2015，139：151-162.

[4] DANN C，NEUMANN G，PETERS J. Policy evaluation with temporal differences：A survey and comparison[J]. Journal of Machine Learning Research，2014，15(1)：809-883.

[5] SCHILPEROOT J，MAK I，DRUGAN M，et al. Learning to Play Pac-Xon with Q-Learning and Two Double Q-Learning Variants[C]// Institute of Electrical and Electronics Engineers Inc. IEEE Symposium Series on Computational Intelligence (SSCI). Bangalore，India，2018：1151-1158.

[6] GHESHLAGHI A M，MUNOS R，GHAVAMZADEH M，et al. Speedy Q-learning[C]// Curran Associates Inc. Advances in Neural Information Processing Systems (NIPS). Granada，Spain，2011：2411-2419.

[7] VAN H H. Double Q-learning[C]// Curran Associates Inc. Advances in Neural Information Processing Systems(NIPS). Canada，2010：2613-2621.

[8] ZHENG S，LUO F，GU C H，et al. Double Speedy Q Learning [J]. Computer Science，2020，47(7)：179-185.

[9] HADJIDIMOS A. Successive overrelaxation (SOR) and related methods[J]. Journal of Computational and Applied Mathematics，2000，123(1)：177-199.

[10] SAITO M，MASUDA K，KURIHARA K. A flexible Q-relearning method to accelerate learning under the change of environments by reusing a portion of useful policies[C]// Society of Instrument and Control Engineers (SICE). Akita，Japan，2012：1223-1227.

[11] JANG B，KIM M，HARERIMANA G，et al. Q-Learning Algorithms：A Comprehensive Classification and Applications[J]. IEEE Access，2019，7：133653-133667.

[12] MNIH V，KAVUKCUOGLU K，SILVER D，et al. Playing Atari with Deep Reinforcement Learningar[J]. arXiv：1312. 5602，2013.

[13] WANG F，ZHANG J J，ZHENG X，et al. Where does AlphaGo go：From church-turing thesis to AlphaGo thesis and beyond [J]. IEEE/CAA Journal of Automatica Sinica，2016，3(2)：113-120.

[14] PENG J，WILLIAMS R J. Incremental multi-step Q-learning [J]. Machine Learning，1996，22(1/2/3)：283.

[15] HU J，WELLMAN M P. Nash Q-learning for general-sum stochastic games[J]. Journal of Machine Learning Research，2004，4(6)：1039-1069.

[16] FANG Z，WANG J，JIANG C，et al. QLACO：Q-learning Aided Ant Colony Routing Protocol for Underwater Acoustic Sensor Networks[C] // Republic of：Institute of Electrical and Electronics Engineers Inc. IEEE Wireless Communications and Networking Conference (WCNC). Seoul，Korea，2020：1-6.

[17] CUI R，WEIM. Research and application of Successive Over-Relaxation Iterative Algorithm[C]// Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology. 2011：3856-3858.

[18] REETZ D. Solution of a Markovian decision problem by successive overrelaxation [J]. Zeitschrift für Operations Research，1973，17(1)：29-32.

[19] TOMMI J，MICHAEL I，SATINDER P. Convergence of Stochastic Iterative Dynamic Programming Algorithms [J]. Neural Computation，1994，6：1185-1201.

[20] JOHN I，KAMANCHI C，BHATNAGAR S. Generalized Speedy Q-Learning[J]. IEEE Control Systems Letters，2020，4(3)：524-529.

[21] KAMANCHI C，DIDDIGI R B，BHATNAGAR S. Successive Over-Relaxation Q-Learning [J]. IEEE Control Systems Letters，2020，4(1)：55-60.

ZHOU Qin，born in 1996，postgraduate. Her main research interests include reinforcement learning and so on.

LUO Fei，born in 1978，Ph.D，associate professor，is a member of China Computer Federation. His main research interests include distributed computing，cloud computing and reinforcement learning.

（责任编辑：喻藜）