计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100084

基于最小二乘的双权重学习法

李 斌[1] 刘 全[1,2,3,4]
1 苏州大学计算机科学与技术学院 江苏 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 江苏 苏州 215006
3 软件新技术与产业化协同创新中心 南京 210000
4 吉林大学符号计算与知识工程教育部重点实验室 长春 130012
(2314073669@qq.com)

摘 要 强化学习是人工智能领域中的一个研究热点。在求解强化学习问题时,传统的最小二乘法作为一类特殊的函数逼近学习方法,具有收敛速度快、充分利用样本数据的优势。通过对最小二乘时序差分算法(Least-Squares Temporal Difference,LSTD)的研究与分析,并以该方法为基础提出了双权重最小二乘 Sarsa 算法(Double Weights With Least Squares Sarsa,DWLS-Sarsa)。DWLS-Sarsa 算法将两权重通过一定方式进行关联得到目标权重,并利用 Sarsa 方法对时序差分误差进行控制。在算法训练过程中,两权重会因为更新样本的不同而产生不同的值,保证了算法可以有效地进行探索;两权重也会因为样本数据的分布而逐渐缩小之间的差距直到收敛至同一最优值,确保了算法的收敛性能。最后将 DWLS-Sarsa 算法与其他强化学习算法进行实验对比,结果表明 DWLS-Sarsa 算法具有较优的学习性能与鲁棒性,可以有效地处理局部最优问题并提高算法收敛时的表现效果。

关键词:强化学习;函数逼近;最小二乘;时序差分;Sarsa
中图法分类号 TP181

Double Weighted Learning Algorithm Based on Least Squares

LI Bin[1] and LIU Quan[1,2,3,4]
1 School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
2 Provincial Key Laboratory for Computer Information Processing Technology,Soochow University,Suzhou,Jiangsu 215006,China
3 Collaborative Innovation Center of Novel Software Technology and Industrialization,Nanjing 210000,China
4 Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry Education,Jilin University,Changchun 130012,China

Abstract Reinforcement Learning is one of the most challenging and difficult concerns in the field of artificial intelligence. Least-squares method is one of the advanced function approximate methods that can be used to solve the problem of reinforcement learning. It has advantages of fast convergence rate and sufficient utilization of sample data. After the study and analysis of least squares temporal diffe-rence algorithm (LSTD),this paper proposes a double weights with least-squares Sarsa algorithm (DWLS-Sarsa) based on the LSTD algorithm. DWLS-Sarsa combines two weights in a certain way and takes control of temporal difference error with Sarsa methods. During the training process,two weights will produce different values because of the difference in the updated samples and will gradually narrow the gap between the two weights until they converge to the same optimal value duo to the distribution of the sample data. So that the exploration performance and convergence of the algorithm will be ensured. Finally,DWLS-Sarsa algorithm is applied to the experiment and compared with other reinforcement learning algorithms. The experimental results show that DWLS-Sarsa algorithm can deal with local optimum problems effectively to achieve more precise

到稿日期:2019-11-11 返修日期:2020-03-24 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(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)
This work was supported by the 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) and Suzhou Industrial Application of Basic Research Program Part (SYG201422).
通信作者:刘全(quanliu@suda.edu.cn)

convergence value and has better learning performance and robustness.

Keywords　Reinforcement learning，Function approximation，Least-squares，Temporal difference，Sarsa

1　引言

强化学习是一种重要的机器学习方法，可广泛应用于游戏学习、复杂系统控制、自主机器人以及多任务系统等多个领域[1-3]，已成为机器学习和智能控制领域的研究热点之一[4-5]。强化学习主要通过 Agent 与环境之间的交互，并利用交互过程中产生的反馈信息对当前的行为策略进行评估改进，从而求解出能使期望累计奖赏最大化的策略[6]。

利用强化学习解决实际问题时，传统的基于查询表(Lookup-Table)的强化学习方法仅适用于小规模、离散状态或动作空间的问题，对于大规模、连续状态或动作空间的强化学习任务则会面临“维数灾”(Curse of Dimensionality)问题。因此，可以使用函数逼近方法对状态值函数或动作值函数进行近似评估，从而求解出近似最优策略。在函数逼近方法中，最小均方差方法(Least-Mean-Square，LSM)是各种增量梯度下降算法的基础。该方法可以有效地更新权重，调整对状态的评估值，但存在收敛速度慢、易陷入局部最优困境的问题。Degris 等[7]对 LSM 进行了改进，并引入了资格迹方法，进一步加快了梯度求解的收敛速度。LSTD 算法是一种特殊的强化学习算法，其从全局最优的角度出发对梯度下降算法进行了优化。通过 Nedić 等[8]的深入研究，该算法可以更充分地利用样本数据，并且拥有更快的收敛速度。

利用函数逼近求解状态值时，可以使用特征构造方法对状态进行特征表示，比如粗糙编码(Coarse Coding)方法、Tile 编码(Tile Coding)方法和傅立叶基数(Fourier Basis)构造法等。这些特征构造方法可以有效表示当前的状态信息，降低当前状态表示的误差；同时，也可以将连续状态空间进行离散化，来降低状态之间的关联性。近期，Williams 等[9]针对连续动作空间问题提出了快速特征选择法。Wookey 等[10]提出了正则特征选择法，进一步完善了对状态的特征表示，提高了状态值评估的精确性。

LSTD 算法将最小二乘思想与时序差分结合，可更充分地利用样本数据信息，将其与 Tile 编码结合，能高效地处理大规模、连续状态空间的强化学习问题。经过众多学者的研究与大量的实验证明，该算法拥有更快的收敛速度和更好的学习性能。为了进一步提高算法的稳定性，Lagoudakis 等[11]提出最小二乘策略迭代算法(Least-Squares Policy Iteration，LSPI)。Jung 等[12]采用增量式计算方式，在基于动作值函数的基础上提出了最小二乘策略评估算法(Least-Squares Policy Evaluation for Q-functions，LSPE-Q)。为了降低样本的复杂度，Wang 等[13]提出平行最小二乘策略迭代算法(Parallel Least-Squares Policy Iteration，PLSPI)。然而，LSTD 算法仍然存在许多不足之处，比如算法的复杂度较高、批处理技术的引入并不能高效地提高算法的运行速度[14]。探索与利用之间的矛盾是 LSTD 算法的一个瓶颈问题。若偏重于利用则会减少对状态的探索，从而无法得到准确的状态分布，此时算法无法收敛至准确值并会陷入局部最优的困境；若偏重于探索，则算法在收敛过程中会产生较大的波动，不利于算法的稳定。在 LSTD 算法的训练过程中所产生的样本数据信息均存储于算法的正定矩阵中。随着算法的更新，Agent 会不断地访问较优的状态，避免访问或直接不再访问较差的状态。此时样本分布会不断偏离真实分布值，较优状态的分布值会不断增加，而较差状态的分布值则会不断减少以至于趋近于 0。无论是采用期望计算的方式还是采用增量计算的方式对矩阵进行更新，均会造成矩阵中的某些元素值发散或者趋近于 0，从而导致算法失效，不利于算法的长期更新。

本文使用 Tile 编码方式将状态空间离散化，对状态进行特征表示；同时，对 LSTD 算法进行改进，在动作值函数的基础上利用 Sarsa 方法对时序差分误差(Temporal Difference Error，TD Error)进行控制，进而提出 DWLS-Sarsa 算法。该算法由两个权重构成，且两个权重使用独立的更新矩阵。两权重独立更新时所得的权重值也会不同，因此算法对动作的选择更具多样性。然后该算法将两权重进行关联使其可以相互配合、相互修正，直到收敛至同一值。同时，双矩阵可以延缓矩阵中元素的变化速率，有利于算法的长期更新。值迭代更新可以在一个情节结束前及时地更新权重，优化当前策略。本文将 DWLS-Sarsa 算法应用到实验中，通过算法对比得出：DWLS-Sarsa 算法可以有效地增强探索，解决局部最优的问题，且拥有较优的学习性能，即使在变化的状态空间中，该算法也能获得较好的收敛效果，具有较强的鲁棒性。

2　理论基础

2.1　马尔可夫决策过程

强化学习方法可用于解决复杂的决策问题。如图 1 所示，强化学习系统包含 Agent 与环境两部分。Agent 以获得最大期望累计奖赏为目标，采用“试错”的方式与环境不断地进行交互，并利用交互所得的奖赏对动作进行评估与指导。

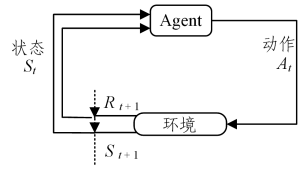

图 1　强化学习示意图

Fig. 1　Diagram of reinforcement learning

在求解强化学习问题时，通常使用马尔可夫决策过程(Markov Decision Process，MDP)对问题进行建模[15]。MDP 模型可以用一个五元组表示，$M=\langle S,A,P,R,\gamma\rangle$。其中，$S$ 是一组有限的状态集合；A 是一组有限的动作集合；P 是状态迁移函数，$P:S\times A\times S\to[0,1]$，$P(s,a,s')$ 表示在状态 $s\in S$ 下采取动作 $a\in A$ 后，转移到后续状态 $s'\in S$ 时的概率；R 为奖赏函数，$R:S\times A\to\mathbb{R}$，$R(s,a)$ 表在状态 s 下采取动作 a 后得到的立即奖赏；$\gamma\in[0,1]$是折扣因子，体现了未来收益对当前时刻的重要性。

强化学习算法根据策略 π 进行决策，策略 π 可以定义为

从状态空间到动作空间的一个映射，即 $\pi: S \rightarrow A$，$a=\pi(s)$ 表示在状态 s 时所采用的动作为 a。

在 MDP 中，通过迭代求解值函数可获得最优策略[16]。在给定策略 π 的情况下，Agent 在状态 s 选择动作 a 得到奖赏 r，并到达下一状态 s'，由 Bellman 公式可以得到当前状态的值函数 V^{π} 以及动作值函数 Q^{π}：

$$V^{\pi}(s)=\sum_{a\in A}\pi(s,a)[R(s,a)+\gamma\sum_{s'\in S}P(s,a,s')V^{\pi}(s')] \tag{1}$$

$$Q^{\pi}(s,a)=R(s,a)+\gamma\sum_{s'\in S}P(s,a,s')\sum_{a'\in A}\pi(s',a')Q^{\pi}(s',a') \tag{2}$$

在最优策略 π^* 下，最优状态值函数 V^* 和最优动作值函数 Q^* 遵循 Bellman 最优方程：

$$V^*(s)=\max_{a\in A}\{R(s,a)+\gamma\sum_{s'\in S}P(s,a,s')V^*(s')\} \tag{3}$$

$$Q^*(s,a)=R(s,a)+\gamma\sum_{s'\in S}P(s,a,s')\{\max_{a'\in A}Q^{\pi}(s',a')\} \tag{4}$$

2.2 最小二乘时序差分法

时序差分(Temporal Difference，TD)方法是一类重要的强化学习方法。利用函数逼近可以解决大规模强化学习问题，同时降低算法对存储资源的消耗[17]。LSTD 算法在 TD 方法的基础上进行优化，利用矩阵计算函数值，摆脱了对参数设定的困扰，也提高了算法的学习性能。

当面临大规模或连续状态空间的强化学习问题时，Agent 无法访问环境中的所有状态，也无法为每一个状态设立独立的变量存储其评估值，因此可以利用函数逼近器对每个状态进行近似表示。函数逼近器为每个状态设立一个共享的 d 维权重向量 $\boldsymbol{w}$ 对其值函数进行近似评估，实现从参数空间到目标函数空间的映射：$\hat{V}(s,\boldsymbol{w})\approx V_{\pi}(s)$。逼近器使用 d 个基函数(Basis Function，BFs)近似表示状态 s：$\phi_1(s)$，$\phi_2(s)$，…，$\phi_d(s)$。近似值函数表示公式如下：

$$\hat{V}(s,\boldsymbol{w})=\boldsymbol{\phi}(s)^{\mathrm{T}}\boldsymbol{w}=\sum_{i=1}^{d}\phi_i(s)\boldsymbol{w}_i \tag{5}$$

其中，$\boldsymbol{\phi}(s)=[\phi_1(s),\phi_2(s),\cdots,\phi_d(s)]^{\mathrm{T}}$ 是状态 s 关于基函数的 d 维特征向量。

基于线性函数逼近的 TD 方法通常使用随机梯度下降的方法对权重 $\boldsymbol{w}$ 进行更新。当算法收敛时，权重会收敛于一个值，称该值为 TD 最优权重 $\boldsymbol{w}_{\mathrm{TD}}$。LSTD 算法直接利用样本数据对权重进行期望求解。记 $\boldsymbol{b}=\mathbb{E}[R(s_t,a_t)\boldsymbol{\phi}(s_t)]$，$\boldsymbol{A}=\mathbb{E}[\boldsymbol{\phi}(s_t)(\boldsymbol{\phi}(s_t)-\gamma\boldsymbol{\phi}(s_{t+1}))^{\mathrm{T}}]$。则 LSTD 算法的求解公式如下：

$$\boldsymbol{w}_{\mathrm{TD}}=\boldsymbol{A}^{-1}\boldsymbol{b} \tag{6}$$

LSTD 算法中，权重的求解只需要利用当前所得的样本数据，不再依赖上一时刻的权重值。大量的实验研究表明，LSTD 算法可以更充分、有效地利用样本数据，以计算消耗为代价提高算法的收敛速度。

3 双权重最小二乘 Sarsa 算法

DWLS-Sarsa 算法以 LSTD 算法为基础构造线性函数逼近器来评估动作值函数，并利用 Sarsa 方法控制 TD 误差，调整策略，以实现对目标函数的更新。算法最终收敛时，其评估值与真实值的误差平方和满足该策略分布下的最小值，如式(7)所示：

$$\boldsymbol{w}=\arg\min_{\boldsymbol{w}\in\mathbb{R}^d}\|\boldsymbol{Q}^{\pi}-\boldsymbol{Q}(\boldsymbol{w})\|_{\mu}^{2}=\arg\min_{\boldsymbol{w}\in\mathbb{R}^d}\sum_{s\in\boldsymbol{S},a\in\boldsymbol{A}}\mu(s,a)(\boldsymbol{Q}^{\pi}(s,a)-\hat{\boldsymbol{Q}}(s,a,\boldsymbol{w}))^2 \tag{7}$$

其中，$\boldsymbol{Q}^{\pi}$ 为 π 策略下所有状态的真实值函数向量集；$\boldsymbol{Q}(\boldsymbol{w})$ 为所有状态的评估值函数向量集；$\mu(s,a)$ 是对状态动作对 (s,a) 的关注程度，关注度越高，在实际情况中到达 s 状态并采用 a 动作的概率也越高。当策略确定时，考虑到该状态下的所有动作，可以得到该策略下的状态分布 $\mu(s)$。当任何状态均可以被无限次地访问时，样本中的实际分布与该策略下的真实数据分布相同。

3.1 算法推导

在特殊的环境中，如 Cliff Walking 实验，一个状态会同时拥有最优和最差两个动作。在无模型环境问题中，无法直接得到当前状态的后续状态，若直接对状态值进行评估，则无法有效地选择该状态下的最优动作。DWLS-Sarsa 算法将对状态值函数的近似评估转换为对动作值函数的近似评估，保留了状态的动作信息。将动作值函数表示为 $\hat{Q}(s,a,\boldsymbol{w})$，可得下式：

$$\hat{Q}(s,a,\boldsymbol{w})=\boldsymbol{\phi}(s,a)\boldsymbol{w}=\sum_{i=0}^{d}\phi_i(s,a)\boldsymbol{w}_i \tag{8}$$

此时，算法对动作的选择可以直接利用当前状态所具备的动作信息，不再依赖其后续状态。算法可以采用在线更新的方式对权重进行更新，无须设置模型对环境进行存储，减少了对存储资源的消耗。

在函数逼近中，使用随机梯度下降方法对权重进行更新，其更新公式如下：

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\alpha(U_t-\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{\phi}_t)\boldsymbol{\phi}_t \tag{9}$$

其中，$\boldsymbol{\phi}_t$ 表示 t 时刻的特征向量 $\boldsymbol{\phi}(s,a)$，U_t 表示此时所得的立即回报值。令 T 为情节结束时间，R_t 为 t 时刻所得的即时奖赏，则 $U_t=\sum_{t}^{T}R_t$。由于对 U_t 直接求解需要耗费大量的计算资源和存储资源，因此可以使用半梯度更新方式，将时序差分的思想应用于式(9)中，并利用 Sarsa 方法对时序差分误差进行控制。最终可得如下公式：

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\alpha(R_t+\gamma\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{\phi}_{t+1}-\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{\phi}_t)\boldsymbol{\phi}_t \tag{10}$$

式(10)用 $R_t+\gamma\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{\phi}_{t+1}$ 替换 U_t，并将其视为一个整体。因此算法在求导时，并未将下一状态纳入考虑。引入 LSTD 算法的思想，对式(10)进行化简，并对其误差进行操作，写成权重的期望形式：

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\alpha(R_t\boldsymbol{\phi}_t-[(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t]^{\mathrm{T}}\boldsymbol{w}_t) \tag{11}$$

$$\mathbb{E}[\boldsymbol{w}_{t+1}|\boldsymbol{w}_t]=\boldsymbol{w}_t+\alpha(\mathbb{E}[R_t\boldsymbol{\phi}_t]-\mathbb{E}[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}]\boldsymbol{w}_t) \tag{12}$$

由式(12)可知，在已知权重 $\boldsymbol{w}_t$ 值的情况下，在任意 t 时刻都可以直接求出下一时刻 $\boldsymbol{w}_{t+1}$ 的值，且收敛时，$\boldsymbol{w}_t=\boldsymbol{w}_{t+1}$。将该算法收敛时所得到的权重值称为 TD 最优权重，记为 $\boldsymbol{w}_{\mathrm{TD}}$。根据上述条件可得 $\boldsymbol{w}_{\mathrm{TD}}$ 的最终求解公式为：

$$\mathbb{E}[R_t\boldsymbol{\phi}_t]-\mathbb{E}[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}]\boldsymbol{w}_{TD}=0 \tag{13}$$

$$\boldsymbol{w}_{\mathrm{TD}}=\mathbb{E}[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}]^{-1}\mathbb{E}[R_t\boldsymbol{\phi}_t] \tag{14}$$

此时，权重的求解受样本数据驱动，并不依赖于上一时刻的权重。称该方法为最小二乘 Sarsa 算法(Least-Squares Sarsa，LS-Sarsa)，权重的详细求解公式如下：

$$\boldsymbol{w}_t=(\sum_{k=0}^{t-1}\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}+\varepsilon\boldsymbol{I})^{-1}(\sum_{k=0}^{t-1}R_t\boldsymbol{\phi}_t) \tag{15}$$

在进行矩阵的逆运算时，并不能确保矩阵 $\boldsymbol{A}$ 可逆，因此引入了单位矩阵 $\boldsymbol{I}$ 和一个常数 ε。当 ε 足够小时，它不仅可以确保矩阵 $\boldsymbol{A}$ 为满秩矩阵，也能降低对算法结果的影响。最后将算法写为如下增量形式：

$$\boldsymbol{w}_t=\boldsymbol{A}_t^{-1}\boldsymbol{b}_t \tag{16}$$

$$\begin{aligned}\boldsymbol{w}_t&=\sum_{k=0}^{t}\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}+\varepsilon\boldsymbol{I}\\&=\begin{cases}\boldsymbol{A}_{t-1}+\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}, & t\neq0\\ \varepsilon\boldsymbol{I}, & t=0\end{cases}\end{aligned} \tag{17}$$

$$\begin{aligned}\boldsymbol{b}_t&=\sum_{k=0}^{t}R_t\boldsymbol{\phi}_t\\&=\begin{cases}\boldsymbol{b}_{t-1}+R_t\boldsymbol{\phi}_t, & t\neq0\\ \vec{0}, & t=0\end{cases}\end{aligned} \tag{18}$$

当 $t=0$ 时，初始权重 $\boldsymbol{w}$ 为 0 向量。该算法为 LS-Sarsa 增量算法，在求解过程中，通过不断地采样对矩阵 $\boldsymbol{A}_t$ 以及向量 $\boldsymbol{b}_t$ 进行累加计算，最终可求解出最优 TD 权重 $\boldsymbol{w}_{\mathrm{TD}}$。

虽然该算法可以有效地利用样本数据，以较快的速度收敛到 TD 最优值。但收敛速度的提升需要以计算资源的消耗为代价，它的时间复杂度为 $O(d^3)$。当强化学习问题的环境规模变大时，算法难以取得令人满意的效果。因此，需要对矩阵 $\hat{\boldsymbol{A}}_t$ 进行进一步优化，降低其时间复杂度，提高算法的学习速度。由 Sherman-morrison 公式可知：$\boldsymbol{\phi}_t,(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})\in\mathbb{R}^d$ 为列向量；$\boldsymbol{A}_{t-1},\boldsymbol{A}_t$ 为可逆矩阵，$\boldsymbol{A}_{t-1}+\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}$ 可逆当且仅当 $1+(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}\boldsymbol{A}_t^{-1}\boldsymbol{\phi}_t\neq0$，因此可以使用 Sherman-morrison 公式对 $\boldsymbol{A}_t$ 进行如下逆运算操作：

$$\begin{aligned}\boldsymbol{A}_t^{-1}&=(\boldsymbol{A}_{t-1}+\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}})^{-1}\\&=\boldsymbol{A}_{t-1}^{-1}-\frac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}\boldsymbol{A}_{t-1}^{-1}}{1+(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}\boldsymbol{A}_{t-1}^{-1}\boldsymbol{\phi}_t}\end{aligned} \tag{19}$$

此时，算法可以直接存储逆矩阵 $\boldsymbol{A}_t^{-1}$，并利用该逆矩阵进行下一步计算求解权重值。算法的时间复杂为 $O(d^2)$。

修改后的算法为 Sarsa 控制下的增量算法，将其作为 DWLS-Sarsa 算法的基础对权重进行求解。取正定矩阵 $\boldsymbol{C}=\mathbb{E}[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}]$，向量 $\boldsymbol{d}=\mathbb{E}[R_t\boldsymbol{\phi}_t]$。其计算方式如式(17)、式(18)所示。为算法设置两个权重 $\boldsymbol{w}_1$ 和 $\boldsymbol{w}_2$，令 $\boldsymbol{w}_1=\boldsymbol{A}_t^{-1}\boldsymbol{b}_t,\boldsymbol{w}_2=\boldsymbol{C}_t^{-1}\boldsymbol{d}_t$。算法使用两个权重求解最优策略，且两权重使用不同的正定矩阵和向量。因此两权重的配合需满足一定条件。前期，权重 $\boldsymbol{w}_1$ 与 $\boldsymbol{w}_2$ 取不同的值以确保算法有足够的探索能力。在样本更新过程中，$\boldsymbol{w}_1$ 与 $\boldsymbol{w}_2$ 会因为样本而收敛到同一值，从而确保了算法的收敛性。因此可以将最终的权重 $\boldsymbol{w}$ 写成如下形式：

$$\boldsymbol{w}=\begin{cases}(1-\alpha)\boldsymbol{w}+\alpha\boldsymbol{w}_1 & (20)\\ \boldsymbol{w}_2 & (21)\end{cases}$$

权重 $\boldsymbol{w}$ 最终由式(20)－式(21)得出。在算法初期，由于两权重不同，两种更新方式所求的权重也会不同，因此会产生较大的波动，有利于算法的探索。式(20)对权重的更新可以将两权重进行关联，减小了权重 $\boldsymbol{w}$ 更新时的变化幅度，有利于算法的稳定。式(21)可以确保权重值在变化的过程中能始终回归于该权重，为算法后期的收敛提供了保障。

3.2　算法分析

根据以上推导，利用值迭代对算法进行更新，给出如下详细的算法流程。

算法 1　DWLS-Sarsa 算法

输入：步长参数 α，更新偏向 β，基函数 $\boldsymbol{\phi}$，折扣因子 γ，探索因子 ε

输出：权重向 $\mathbf{w}$

初始化：$\mathrm{t}=0,\mathbf{A}=\varepsilon\mathbf{I},\mathbf{b}=0,\mathbf{C}=\varepsilon\mathbf{I},\mathbf{d}=0$

1. Repeat1(对于每个情节)
2. 环境初始状态 s
3. 环境初始动作 a，$\mathrm{a}\in\mathbf{A}$
4. Repeat2(对于情节中的每一步)
5. 执行动作 a，得到立即奖赏 r，以及下一状态 s′
6. 如果 s′为终止状态，结束 Repeat2
7. $\mathrm{t}=\mathrm{t}+1$
8. 选择动作 a′，$\mathrm{a}'\in\mathbf{A}$
9. if random $<\beta$：

$$\mathbf{A}=\mathbf{A}-\frac{\mathbf{A}\boldsymbol{\phi}(\mathrm{s},\mathrm{a})(\boldsymbol{\phi}(\mathrm{s},\mathrm{a})-\gamma\boldsymbol{\phi}(\mathrm{s}',\mathrm{a}'))^{\mathrm{T}}\mathbf{A}}{1+(\boldsymbol{\phi}(\mathrm{s},\mathrm{a})-\gamma\boldsymbol{\phi}(\mathrm{s}',\mathrm{a}'))^{\mathrm{T}}\mathbf{A}\boldsymbol{\phi}(\mathrm{s},\mathrm{a})}$$

$$\mathbf{b}=\mathbf{b}+\mathrm{r}\boldsymbol{\phi}(\mathrm{s},\mathrm{a})$$

$$\mathbf{w}_1=\mathbf{A}\mathbf{b}$$

$$\mathbf{w}=(1-\alpha)\mathbf{w}+\alpha\mathbf{w}_1$$

else：

$$\mathbf{C}=\mathbf{C}-\frac{\mathbf{C}\boldsymbol{\phi}(\mathrm{s},\mathrm{a})(\boldsymbol{\phi}(\mathrm{s},\mathrm{a})-\gamma\boldsymbol{\phi}(\mathrm{s}',\mathrm{a}'))^{\mathrm{T}}\mathbf{C}}{1+(\boldsymbol{\phi}(\mathrm{s},\mathrm{a})-\gamma\boldsymbol{\phi}(\mathrm{s}',\mathrm{a}'))^{\mathrm{T}}\mathbf{C}\boldsymbol{\phi}(\mathrm{s},\mathrm{a})}$$

$$\mathbf{d}=\mathbf{d}+\mathrm{r}\boldsymbol{\phi}(\mathrm{s},\mathrm{a})$$

$$\mathbf{w}_2=\mathbf{C}\mathbf{d}$$

$$\mathbf{w}=\mathbf{w}_2$$

10. $\mathrm{s}\leftarrow\mathrm{s}',\mathrm{a}\leftarrow\mathrm{a}'$
11. 直到达到收敛要求，结束 Repeat1
12. Return $\mathbf{w}$

DWLS-Sarsa 算法可以对探索进行有效地控制，平衡探索与利用之间的关系。其中，参数 β 决定了 $\boldsymbol{w}_1$ 与 $\boldsymbol{w}_2$ 的更新比重，参数 α 将 $\boldsymbol{w}_1$ 与 $\boldsymbol{w}_2$ 进行了关联。参数 α 与 β 共同决定了算法对探索的控制和对当前权重值的利用。若 α 与 β 不为 0 为 1 时，该算法即为 LS-Sarsa 算法。在其情况下，若 α 与 β 取值同时偏高或偏低，算法更倾向于使用较为准确的权重，因此更注重利用；反之，算法更注重于探索。

DWLS-Sarsa 算法通过对参数 α 和 β 的调整，提高了算法的收敛速度和学习性能，即使在探索率 ε 很小的情况下，算法依旧可以利用自身的优势进行合理的探索，得到更精确的分布值，能有效地解决局部最优问题并求解出最优权重。

3.3　收敛性证明

本节主要分析 DWLS-Sarsa 算法的收敛性能。将算法中的正定矩阵和向量进行如下操作：

$$\begin{aligned}\boldsymbol{A}&=\mathbb{E}[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t-\gamma\boldsymbol{\phi}_{t+1})^{\mathrm{T}}]\\&=\sum_s\mu(s)\sum_a\pi(a\mid s)\sum_{s'}P(s'\mid s,a)\sum_{a'}\pi(a'\mid s')\boldsymbol{\phi}(a,s)\\&\quad(\boldsymbol{\phi}(a,s)-\gamma\boldsymbol{\phi}(a',s'))^{\mathrm{T}}\\&=\sum_{a,s}\mu(s,a)\sum_{a',s'}p(a',s'\mid s,a)\boldsymbol{\phi}(a,s)\\&\quad(\boldsymbol{\phi}(a,s)-\gamma\boldsymbol{\phi}(a',s'))^{\mathrm{T}}\\&=\sum_{a,s}\mu(s,a)p(a',s'\mid s,a)\boldsymbol{\phi}(a,s)\\&\quad(\boldsymbol{\phi}(a,s)-\sum_{a',s'}\gamma\boldsymbol{\phi}(a',s'))^{\mathrm{T}}\end{aligned} \tag{22}$$

$$\begin{aligned}\boldsymbol{b}&=\mathbb{E}[R_t\boldsymbol{\phi}_t]\\&=\sum_s\mu(s)\sum_a\pi(a|s)\boldsymbol{\phi}(a,s)R(s,a)\\&=\sum_{s,a}\mu(s,a)\boldsymbol{\phi}(a,s)R(s,a)\end{aligned}\tag{23}$$

其中，$p(a',s'|s,a)$表示当前状态动作对(s,a)到下一状态动作对(a',s')的概率。令 $\boldsymbol{D}$ 为关于$\mu(s,a)$的对角矩阵，$\boldsymbol{P}$ 为关于 $p(a',s'|s,a)$的迁移函数矩阵，$\boldsymbol{\phi}$ 表示状态动作对的特征向量矩阵，$\boldsymbol{R}$ 为关于奖赏函数 R 的奖赏矩阵，则可以将式(22)、式(23)转换为如下矩阵形式：

$$\boldsymbol{A}=\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{\phi}-\gamma\boldsymbol{P}\boldsymbol{\phi})=\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{I}-\gamma\boldsymbol{P})\boldsymbol{\phi}\tag{24}$$

$$\boldsymbol{b}=\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{R}\tag{25}$$

因此可得最终收敛权重为：

$$\begin{aligned}\boldsymbol{w}&=\boldsymbol{A}^{-1}\boldsymbol{b}\\&=(\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{I}-\gamma\boldsymbol{P})\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{R}\end{aligned}\tag{26}$$

权重值与状态动作对的分布相关，因此，所求得的权重为当前分布下的最优权重。当获得足够的样本数据时，算法实际所得的分布矩阵会逐渐逼近 $\boldsymbol{D}$。由于矩阵 $\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{I}-\gamma\boldsymbol{P})\boldsymbol{\phi}$ 可逆，则 $\boldsymbol{w}=(\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{I}-\gamma\boldsymbol{P})\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{R}$ 必然成立。在同一个策略中更新矩阵 $\boldsymbol{C}$ 与 $\boldsymbol{d}$ 和 $\boldsymbol{A}$ 与 $\boldsymbol{b}$，当样本足够时，会得到相同的状态分布，因此可得如下公式：

$$\boldsymbol{w}=\boldsymbol{C}^{-1}\boldsymbol{d}=\boldsymbol{A}^{-1}\boldsymbol{b}\tag{27}$$

由此可知，该算法的两部分均可收敛，且能收敛到同一值。

4 实验及结果分析

为验证算法的有效性，本文以强化学习中经典的 A-Presplit 反例、Mountain Car 和 Random Walk 为例进行实验。其中，A-Presplit 反例用于验证算法收敛时的准确性；Mountain Car 实验用于验证算法的收敛性能，体现算法对局部最优问题的处理能力；Random Walk 实验用于测试算法在变化空间中的学习性能，验证其鲁棒性。

4.1 A-Presplit 反例

A-Presplit 反例是强化学习中的一个经典问题，由 A-split 反例改进而来。利用该实验可以更好地验证算法收敛的准确性。

如图 2 所示，该问题包含 4 个状态，情节可以从状态 A_1 开始，经过状态 B 后到达终点，并得到 0 和 1 的奖赏。情节也可以从状态 A_2 出发，经过状态 C 后到达终点，所得奖赏均为 0。两情节发生的概率相同且状态 A_1 和状态 A_2 的特征表示也相同。

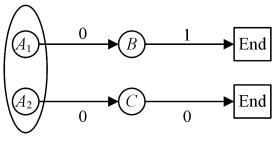

图 2 A-Presplit 问题示意图

Fig. 2 Diagram of A-Presplit

可将图 2 中的状态 A_1 和状态 A_2 视为一个状态 A。状态 B 和状态 C 的真实值为 1 和 0，状态 A 得到 1 或 0 的奖赏的概率均为 0.5，所以其真实值为 0.5。本文的目标是对动作值函数进行评估，在对状态动作对进行表示时，假设状态 A_1 到状态 B 的动作与状态 A_2 到状态 C 的动作的特征表示相同，其真实值均为 0.5。状态 B 和状态 C 所采取的动作的真实值分别为 1 和 0。当状态空间较大时，由于特征维度的限制，一些状态动作对的特征表示会相同或者线性相关，导致算法无法收敛到真实值。该实验中设置两个相同的状态动作对用于模拟这种情况，算法最终的收敛结果如表 1 所列。

表 1 各算法收敛时所得期望动作值

Table 1 Expected convergence value of different algorithms

Algorithm	$Q(A_1,B)$	$Q(A_2,B)$	$Q(B,\mathrm{End})$	$Q(C,\mathrm{End})$
Semi-gradient Sarsa	0.5	0.5	1	0
LS-Sarsa	0.5	0.5	1	0
Naive Residual-Gradient	0.5	0.5	0.75	0.25
Residual-Gradient	0.5	0.5	0.75	0.25
GTD(0)	0.5	0.5	1	0
DWLS-Sarsa	0.5	0.5	1	0

表 1 中，记 $Q(x,y)$为从状态 x 到状态 y 所选择的动作的评估值。该实验将随机梯度下降方法与半梯度方法作比较，以验证算法收敛时的准确性。在随机梯度下降算法中，算法的求导更为完整。其中，Naive Residual-Gradient 算法[6]以最小化 TD 误差为目标进行函数逼近；Residual-Gradien 算法[6]以最小化 Bellman 误差为目标进行函数逼近，并以二次采样的方式进行优化。由于 TD 误差和 Bellman 误差的不可学习性，算法在求解过程中容易收敛到错误值，因此无法对动作值进行准确评估。GTD(0)算法[6]将 Bellman 误差进行投影操作，并以二次采样的方式进行求解。该算法可以有效利用样本数据求解出精确的函数值，但算法操作复杂，需要人为设定两个独立且合理的参数。半梯度 TD 方法虽然无法有效应用于异策略函数逼近问题，易产生发散的问题，但该方法可以在同策略函数逼近问题中取得较好的效果，并收敛至正确值。DWLS-Sarsa 和 LS-Sarsa 对半梯度方法进行了改进，且更新过程由样本驱动，可高效地利用样本数据，均能收敛至正确值。

4.2 Mountain Car 实验

为了将算法应用到更为复杂的问题中，本文以 Mountain Car 问题为例，其实验示意图如图 3 所示。

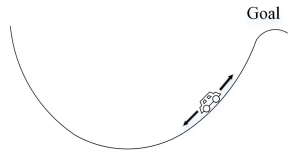

图 3 Mountain Car 实验示意图

Fig. 3 Diagram of Mountain Car

实验中，动力不足的小车从山坡左侧出发，在两坡间不断徘徊以获得足够动力，从而到达右侧山顶。环境中，小车的特征由坐标 p，车速 v 以及动作 a 构成。t 时刻小车的坐标 p_t 和车速 v_t 满足如下条件：

$$p_{t+1}=bound[p_t+v_t]$$

$$v_{t+1}=bound[v_t+0.001a_t-0.0025\cos(3p_t)]$$

其中，$bound$ 为限界函数，规定了 $-1.2\leqslant p_t\leqslant 0.5$，$-0.07\leqslant v_t\leqslant 0.07$。小车拥有 3 个动作：$a_t$ 为 1 时表示向右加速，a_t 为 0 时表示不加速，a_t 为 -1 时表示向左加速。小车从坐标为 $p_0=-0.5$ 的位置开始运行，起始车速 $v_0=0$。在运行过程

中，小车每一时刻均可获得值为 0 的立即奖赏。当小车到达目标，即运行到最右侧坡顶时，获得值为 1 的立即奖赏，且一个情节结束。

Mountain Car 问题中状态的特征表示维度为 3，环境中含有大量线性相关的状态，算法在求解过程中会产生较大的误差。本文采用 Tile 编码[18]方法将连续状态空间转换为离散状态空间，并将状态的特征表示的维度升至 1024 维。Tile 编码方法中 tiling 数为 8，且 tiling 中每一个 tile 的大小均为 1。最后分析算法在实验中的学习性能，并与其他算法进行对比。

图 4 为 DWLS-Sarsa 算法在不同参数下运行 100 次且每次运行 300 个情节时所得的平均步数。实验中算法的探索率 $\varepsilon=0$。该实验可用于分析 DWLS-Sarsa 算法在不同参数下的效果。

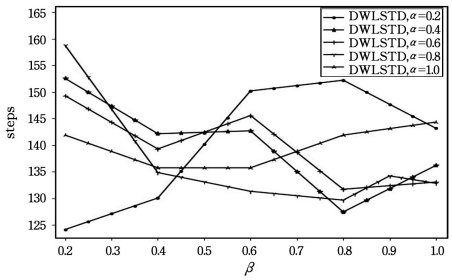

图 4　DWLS-Sarsa 算法在不同参数下到达目标所需的时间步

Fig. 4　Average steps of DWLS-Sarsa with different parameters

图 4 可以反映算法在不同参数下对局部最优问题的处理效果，当 $\alpha=0.2$，$\beta=0.2$ 时，该算法取得的效果最好。此时算法以 $\boldsymbol{w}_2$ 为主导，以 $\boldsymbol{w}_1$ 为辅。在实验运行初期，$\boldsymbol{w}_1$ 因为样本数据不足以及更新数据的不同，与 $\boldsymbol{w}_2$ 存在较大的误差。此时 α 取值较小，使得目标权重 $\boldsymbol{w}$ 的变化较小。因此算法在选择动作时，不仅可以有效地利用 $\boldsymbol{w}_2$，也可以根据 $\boldsymbol{w}_1$ 进行有效的探索。当 $\alpha=0.8$，$\beta=0.8$ 时，算法可以取得同样较优的学习效果。在其他情况下，α 与 β 的调整会使算法更偏向于探索，虽然可以访问更多的状态，但却无法充分利用主权重，因此算法的实际表现性能有所降低。若 α 与 β 值相差过大，则会影响算法的性能，无法取得理想的效果。

图 5 和图 6 为实验运行 50 次后所得的平均步数。其中，图 5 记录了前 100 个情节算法的平均步数，用于分析算法的收敛速度；图 6 记录了后 200 个情节算法平均步数，用于分析算法的收敛性。实验中，算法的参数取值均能使算法取得较优的学习性能。该实验中算法的探索率 $\varepsilon=0$，用于表现 DWLS-Sarsa 算法自身所具备的探索优势。

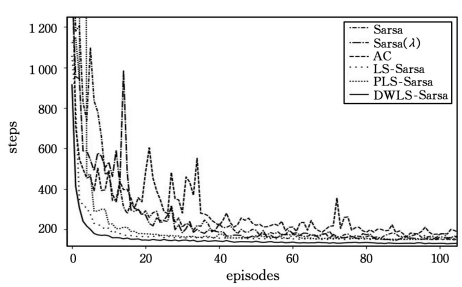

图 5　不同算法在前 100 个情节下的时间步

Fig. 5　Average steps of different algorithms in first hundred episodes

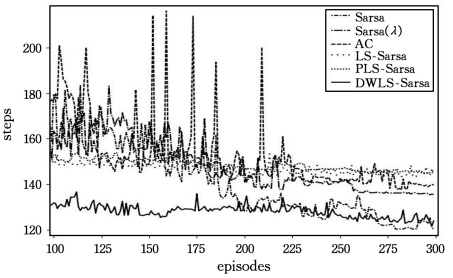

图 6　不同算法在后 200 个情节下的时间步

Fig. 6　Average steps of different algorithms in last two hundred episodes

从图 5 与图 6 可以看出，DWLS-Sarsa 算法的学习性能优于其他算法。其中半梯度 Sarsa 算法中步长参数 $\alpha=0.1$。该算法虽然可以取得良好的效果，但是收敛速度过慢。提高步长参数 α 的值虽然可以提升收敛速度，但是也会导致收敛时权重值不稳定，从而影响算法的平均性能。Sarsa(λ)算法[19]在 Sarsa 基础上引入资格迹，其中 $\alpha=0.1$，$\lambda=0.1$。资格迹的引入使该算法可以更好地利用样本路径信息，同时也提高了算法的收敛速度，后期收敛时算法也相对稳定。但该算法依旧未能取得最优的表现效果。行动者-评论家算法(Actor-Critic，AC)中评论家部分所用参数为 $\alpha=0.1$，行动者部分所用参数 $\beta=0.05$，其学习性能与 Sarsa(λ)算法相似。LS-Sarsa 算法收敛速度相对较快，且收敛时权重也更加稳定，但过快的收敛导致了算法所经历的状态较少，所得到的状态分布并不准确，因此收敛时效果并非最好。PLS-Sarsa 算法[13]对 PLS-PI 算法进行了改进，在基于动作值函数的基础上利用值迭代方式更新权重。该算法中参数 $\alpha=2$，设置了 2 个独立的工作者求解最优策略。由图 5 和图 6 可知，该算法虽然收敛速度较快，但其实际表现效果并没有得到明显提升。DWLS-Sarsa 算法中 $\alpha=0.2$，$\beta=0.2$。该算法对 LS-Sarsa 算法进行了改进，算法的收敛速度得到了提升。双权重的配合使得算法能有效地利用权重，也能更合理地进行探索。因此该算法可以有效地处理局部最优问题。

图 7 和图 8 为加入探索因子 $\varepsilon=0.01$ 时，算法运行 100 次后所得的平均结果。其中，图 7 记录了前 100 个情节的平均步数，图 8 记录了后 200 个情节算法收敛时的平均步数。Sarsa 算法中 $\alpha=0.1$；Sarsa(λ)算法中 $\alpha=0.1$，$\lambda=0.1$；AC 算法中 $\alpha=0.1$，$\beta=0.01$；PLS-Sarsa 算法中 $\alpha=2$；GTD(0)算法中 $\alpha=0.001$，$\beta=0.01$；DWLS-Sarsa 算法中 $\alpha=0.6$，$\beta=0.8$。这些参数取值均能使算法取得较优的效果。该实验用于分析算法在探索情况下对算法稳定性的控制能力。

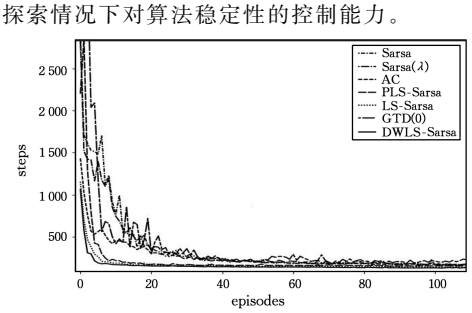

图 7　不同算法在前 100 个情节下的时间步

Fig. 7　Average steps of different algorithms in first hundred episodes

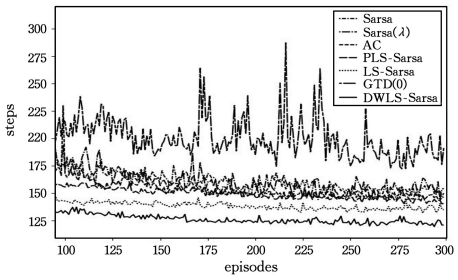

图 8 不同算法在后 200 个情节下的时间步

Fig. 8 Average steps of different algorithms in last two hundred episode

在运算过程中，算法进行探索虽然有利于发现更有效的路径，但选择错误的动作所得的数据也会用于更新权重。受这些错误数据的影响，Sarsa 算法与 Sarsa(λ)算法的稳定性较差，在收敛过程中波动性较大。而 AC 算法、LS-Sarsa 算法、PLS-Sarsa 算法的稳定性较好，但依旧无法取得理想的表现性能。然而，DWLS-Sarsa 算法不仅拥有较快的收敛速度，也能得到较优的收敛值，它的表现性能是所有算法中最好的。

由上述实验可得：DWLS-Sarsa 算法在收敛速度和收敛值方面均有较强的优势。算法本身具备良好的探索能力，可以利用两权重的配合进行合理的探索，有效地防止算法陷入局部最优的困境。

4.3 Random Walk 实验

Random Walk 实验用于在变化的状态空间中验证算法的鲁棒性。该实验包含 1000 个状态，且两端各有一个终点，呈线性排列。实验示意图如图 9 所示。

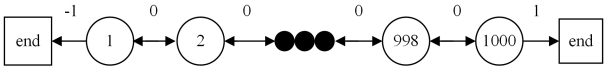

图 9 Random Walk 问题示意图

Fig. 9 Diagram of Random Walk

情节从中间位置 $s_0=500$ 出发，随机向左或向右前行。若到达最左端终点状态，则获得值为 -1 的奖赏；若到达最右端终点状态，则获得值为 1 的奖赏；其他情况下取得的奖赏为 0。在 t 时刻，记动作选择向右时 $a_t=1$，动作选择向左时 $a_t=-1$，其状态迁移满足如下公式：

$$s_{t+1}=\max(\min(s_t+a_t(\text{random}(100)+1),1001),0)$$

DWLS-Sarsa 算法的实验效果如图 10 所示，图中的数据为算法在不同的参数下运行 100 次，且每次运行 1000 个情节之后的状态值标准差(Root Mean Square，RMS)的平均值。

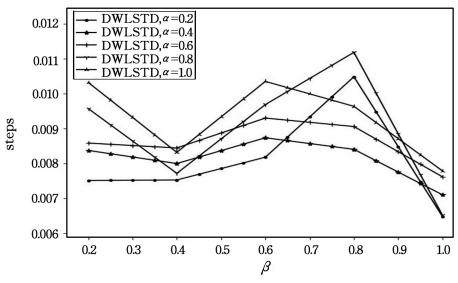

图 10 DWLS-Sarsa 在不同参数下收敛时的状态值标准差

Fig. 10 *RMS* of DWLS-Sarsa when it convengences

由图 10 可知，当更新偏向 $\beta=1$ 时，算法所取得的效果最好，拥有较低的 *RMS*。此时算法只含有一个权重，矩阵中所存储的样本数据更为完整，可以有效地降低算法的 *RMS*。而 $\beta\neq1$ 时，在更新过程中算法需要存储两个正定矩阵，导致短期内两个矩阵所存储的样本数据较少，因此算法的 *RMS* 偏高，学习性能较差。

图 11 和图 12 为 Sarsa 算法、Sarsa (λ)算法、LS-Sarsa 算法以及 DWLS-Sarsa 算法在实验中运行 100 次后的实验结果图。其中，图 11 记录了前 100 个情节的平均 *RMS*，图 12 记录了后 200 个情节的平均 *RMS*。Sarsa 算法中 $\alpha=0.1$；Sarsa (λ)算法中 $\alpha=0.1,\beta=0.1$；AC 算法中 $\alpha=0.01,\beta=0.01$；PLS-Sarsa 算法中 $\alpha=3$；DWLS-Sarsa 算法中 $\alpha=0.8,\beta=1$。各算法在这些参数下均能取得良好的性能。

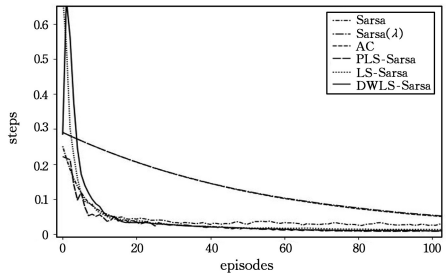

图 11 不同算法前 100 个情节时的标准差

Fig. 11 *RMS* of different algorithms in first hundred episodes

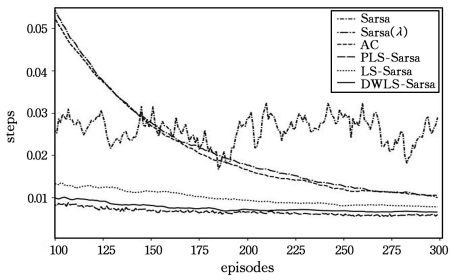

图 12 不同算法后 200 个情节时的标准差

Fig. 12 *RMS* of different algorithms in last two hundred episodes

由图 11 与图 12 可知，Sarsa 算法虽然可以快速收敛，但是收敛时波动较大，且 *RMS* 偏高。Sarsa (λ)算法与 AC 算法的效果相似，虽然收敛速度慢，但是相对稳定，可逐步降低 *RMS*。LS-Sarsa 算法和 DWLS-Sarsa 算法前期 *RMS* 偏大，但均可以快速降低 *RMS*，且 DWLS-Sarsa 算法的整体性能优于 LS-Sarsa 算法。在所有算法中，PLS-Sarsa 算法所取得的效果最优，拥有较快的收敛速度和最低的 *RMS*。

通过上述实验可以得出：在变化的状态空间中，DWLS-Sarsa 算法并不会因为两权重值的不同而产生算法不稳定的情况，并且算法不仅拥有较快的速度和较好的收敛效果，还拥有较强的鲁棒性。同时，可以借鉴 PLS-Sarsa 算法的思想，为 DWLS-Sarsa 算法设置多个工作者，以进一步降低该算法的 *RMS*。

结束语 本文以最小二乘时序差分算法为基础，提出了双权重最小二乘 Sarsa 算法(DWLS-Sarsa)。DWLS-Sarsa 算法中两权重相互配合以寻找最优策略，直至算法收敛。算法所选择的动作既可以有效利用当前数据，也可以进行合理的

探索。同时，两个正定矩阵能存储比原来更多的样本信息，降低了单个矩阵中元素的变化速度，有利于算法的长期更新。实验表明，DWLS-Sarsa 算法拥有较快的收敛速度，并快速收敛至最优权重。该算法可以利用自身的探索优势进行更多合理的探索，能有效处理因状态规模过大而产生的局部最优问题。在变化的状态空间中，DWLS-Sarsa 算法依旧拥有良好的学习性能，具有较强的鲁棒性。

DWLS-Sarsa 算法虽然对计算资源的消耗偏高，但对初始参数的依赖较低，拥有较优的学习性能。在下一步工作中，可以考虑采用多线程的方式同步计算权重，并利用批量策略迭代方法的思想[21]来优化算法，提高算法的计算速度。未来的研究内容可以考虑引入异策略思想形成异策略函数逼近算法，或者从深度强化学习的角度出发，将 DWLS-Sarsa 算法与神经网络相结合，以解决最小二乘方法的反向传递问题，形成深度强化学习算法。

参 考 文 献

[1] MOERLAND T M, BROEKENS J, JONKER C M. Emotion in reinforcement learning agents and robots: a survey [J]. Machine Learning, 2018, 107(2): 4480.

[2] LIU T, TIAN B, CAO D, et al. Parallel Reinforcement Learning: A Framework and Case Study [J]. IEEE/CAA Journal of Automatica Sinica, 2018, 5(4): 65-73.

[3] DU W, DING S F. Overview on Multi-agent Reinforcement Learning [J]. Computer Science, 2019, 46(8): 1-8.

[4] ZHAO X Y, DING S F. Research on Deep Reinforcement Learning [J]. Computer Science, 2018, 45(7): 1-6.

[5] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540): 529-533.

[6] SUTTON R S, BARTO A. Reinforcement Learning: An Introduction [M]. MIT Press, 2019.

[7] DEGRIS T. PILARSKI P M. SUTTON R S. Model-free reinforcement learning with continuous action in practice[C]// Proceedings of 2012 American Control Conference. Montreal, QC, Canada, 2012: 2177-2182.

[8] NEDIĆ A, BERTSEKAS D. Convergence Rate of Incremental Subgradient Algorithms [J]. Stochastic Optimization: Algorithms and Applications, 2001, 54: 223.

[9] LI L, WILLIAMS J D, BALAKRISHNAN S. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection[C]// Proceedings of the 10th Annual Conference of the International Speech Communation Association. Brighton, UK, 2009.

[10] WOOKEY D S, KONIDARIS G D. Regularized feature selection in reinforcement learning [J]. Machine Learning, 2015, 100(2/3): 655-676.

[11] LAGOUDAKIS M G, PARR R. Least-Squares Policy Iteration [J]. Journal of Machine Learning Research, 2004, 4(6): 1107-1149.

[12] JUNG T, POLANI D. Least squares SVM for least squares TD learning[C]// Procedings of the 17th European Conference on Artificial Intelligence. Riva del Garda, Italy, 2006.

[13] WANG J K, LIN S D. Parallel Least-Squares Policy Iteration [C]// 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). 2016: 166-173.

[14] ZHOU X, LIU Q, FU Q M, et al. Batch Least-squares Policy Iteration [J]. Computer Science, 2014, 41(9): 232-238.

[15] GEORGE J A, SHALABH B. An online prediction algorithm for reinforcement learning with linear function approximation using cross entropy method [J]. Machine Learning, 2018, 107(8/9/10): 1385-1429.

[16] GEIST M, PIETQUIN O. Parametric value function approximation: A unified view[C]// Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning. Piscataway, USA, 2011.

[17] BUSONIU L BRUINB T D, TOLICD, et al. Reinforcement Learning for Control: Performance, Stability, and Deep Approximators[J]. Annual Reviews in Control, 2018, 46: 8-28.

[18] JIN Y J, ZHU W W, FU Y C, et al. Actor-Critic Algorithm Based on Tile Coding and Model Learning[J]. Computer Science, 2014, 41(6): 239-242, 249.

[19] VAN SEIJEN H, MAHMOOD A R, PILARSKI P M, et al. True Online Temporal-Difference Learning [J]. Journal of Machine Learning esearch, 2015, 17(1): 5057-5096.

[20] GRONDMAN I, BUSONIU L, LOPES G A D, et al. A Survey of ActorCritic Reinforcement Learning: Standard and Natural Policy Gradients [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6): 1291-1307.

[21] GHORBANI F, DERHAMI V, AFSHARCHI M. Fuzzy Least Square Policy Iteration and Its Mathematical Analysis [J]. International Journal of Fuzzy Systems, 2017, 19(3): 849-862.

LI Bin, born in 1994, master candidate. His main research interests include reinforcement learning and deep reinforcement learning.

LIU Quan, born in 1969, Ph.D, professor, Ph.D supervisor, is a member of China Computer Federation. His main research interests include reinforcement learning, intelligence information processing and automated reasoning.