计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190500143

基于双估计器的改进 Speedy Q-learning 算法

郑 帅 罗 飞 顾春华 丁炜超 卢海峰
华东理工大学信息科学与工程学院 上海 200237
(2503522167@qq.com)

摘 要 Q-learning 算法是一种经典的强化学习算法,更新策略由于保守和过估计的原因,存在收敛速度慢的问题。Speedy Q-learning 算法和 Double Q-learning 算法是 Q-learning 算法的两个变种,分别用于解决 Q-learning 算法收敛速度慢和过估计的问题。文中基于 Speedy Q-learning 算法 Q 值的更新规则和蒙特卡洛强化学习的更新策略,通过理论分析及数学证明提出了其等价形式,从该等价形式可以看到,Speedy Q-learning 算法由于将当前 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。针对该问题,文中基于 Double Q-learning 算法中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。其通过双估计器,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。在不同规模的格子世界中进行实验,分别采用线性学习率和多项式学习率,来对比 Q-learning 算法及其改进算法在迭代初期的收敛速度和整体收敛速度。实验结果表明,Double Speedy Q-learning 算法在迭代初期的收敛速度快于 Speedy Q-learning 算法,且其整体收敛速度明显快于对比算法,其实际平均奖励值和期望奖励值之间的差值最小。

关键词:Q-learning;Double Q-learning;Speedy Q-learning;强化学习
中图法分类号 TP181

Improved Speedy Q-learning Algorithm Based on Double Estimator

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

Abstract Q-learning algorithm is a classical reinforcement learning algorithm. However,due to overestimation and the conservative updating strategy,there exists a problem of slow convergence. Speedy Q-learning algorithm and Double Q-learning algorithm are two variants of the Q-learning algorithm which are used to solve the problems of slow convergence and over-estimation in Q-learning algorithm respectively. Based on the updating rule of Q value in Speedy Q-learning algorithm and the updating strategy of Monte Carlo reinforcement learning,the equivalent form of the updating rule of Q value is proposed through theoretical analysis and mathematical proof. According to the equivalent form,Speedy Q-learning algorithm takes the estimation function of current Q value as the estimation of the historical Q value. Although the overall convergence speed of the agent is improved,Speedy Q-learning also has the problem of overestimation,which leads to a slow convergence at the beginning of iterations. In order to solve the problem of slow convergence at the initial stage of iterations in the Speedy Q-learning algorithm,an improved algorithm named Double Speedy Q-learning is proposed based on the fact that the double estimator in the Double Q-learning algorithm can improve the convergence speed of agents. By using double estimator,the selection of optimal action and maximum Q value is separated,so that the learning strategy of Speedy Q-learning algorithm in the initial iteration period can be improved and the overall convergence speed of Speedy Q-learning algorithm can be improved. Through grid world experiments of different scales,linear learning rate and polynomial learning rate are used to compare the convergence speed of Q-learning algorithm and its improved algorithm in the initial iteration stage and the overall convergence speed. The results show that the convergence speed of the Double Speedy Q-learning algorithm is faster than that of Speedy Q-learning algorithm in the initial iteration stage and its overall convergence speed is significantly faster than that of comparison algorithms. Its difference between the actual average reward value and the expected reward value is also the smallest.

Keywords Q-learning,Double Q-learning,Speedy Q-learning,Reinforcement learning

到稿日期:2019-05-27 返修日期:2019-08-26 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(61472139);华东理工大学 2017 年教育教学规律与方法研究项目(ZH1726107)
This work was supported by the National Natural Science Foundation of China (61472139) and Research Project of Education and Teaching Law and Method of East China University of Science and Technology in 2017 (ZH1726107).
通信作者:罗飞(luof@ecust.edu.cn)

1 引言

强化学习主要用于解决序列决策问题，通过最大化长期奖励找到最优策略[1]。智能体根据当前学习到的策略执行最优动作，进入下一状态，环境根据智能体的行为给出反馈，智能体根据环境反馈学习，然后根据学习到的策略继续执行最优动作，进入下一状态，智能体通过与环境的不断交互学习到最优策略。根据是否有状态转移和奖励模型，可将强化学习算法分为基于模型[2]的强化学习算法和不基于模型的强化学习算法。

Q-learning算法是经典的不基于模型的强化学习算法[3]，被用于解决马尔可夫决策过程(Markov Decision Processes，MDPs)中的最优控制问题[4]，在人机对话[5]、机器人导航[6]、生产调度[7]、交通控制[8]等领域有广泛应用。当状态动作数目有限时，Q-learning算法被证明能够收敛到最优策略[9]，但是当衰减因子接近1时，Q-learning算法的收敛速度会变得很慢[10]。在Q-learning算法中，需要考虑如何估计最大期望Q值[11]，一个好的估计器可以使算法避免陷入局部最优并更快地收敛。目前，最常用的方法是利用样本中的最大值作为最大期望值的估计，但这会导致大范围的过估计问题，使得算法不能快速地收敛。

很多研究者基于Q-learning的估计函数，对Q-learning算法进行了改进。Double Q-learning算法[11]通过使用两个估计器的方式避免了过估计的问题，一个估计器每次选择最优动作，另一个估计器根据选择的最优动作选择Q值，由于分离了最优动作和最大Q值的选择，使得每次选取的估计Q值小于或等于实际最大期望Q值。Double Q-learning算法虽然避免了过估计问题，但存在欠估计问题，随着状态空间的增加，Double Q-learning算法的收敛速度也变得十分缓慢。

Speedy Q-learning算法[12]通过将Q-learning算法中的一个更新准则的学习率更换为一个较大的学习率，使得算法的收敛速度得到提升。本质上，Speedy Q-learning算法将历史Q值的估计函数更换为当前Q值的估计函数，选择了一个更有效的估计器，提升了Q-learning算法的收敛速度。但是，该算法由于一开始就将历史Q值的估计更换为当前Q值的估计，因此同样存在过估计问题，在迭代初期，Speedy Q-learning算法的收敛速度表现较差。

针对上述问题，本文提出了Speedy Q-learning算法Q值更新公式的等价变形，分析了Speedy Q-learning算法在迭代初期收敛速度慢的原因，同时根据Double Q-learning中的双估计器可以改善智能体收敛速度的特性，将Speedy Q-learning算法中的最优动作和Q值选择分离，从而改善智能体的收敛速度。

2 相关工作

基于Q-learning算法收敛速度慢的问题，研究者提出了很多改进算法。Bias-correction Q-learning算法[13]使用了偏差更正策略，使其可以渐进收敛至最优策略，但是该算法只能用于动作值的个数不少于5的情况。Delayed Q-learning算法[14]改变了Q值的更新策略，Q-learning算法是每步都更新Q值，而Delayed Q-learning算法是遇到某状态动作m次后才更新该状态动作值。Weighted Q-learning算法[15]使用了一个权重估计器，把所有样本均值的加权平均值作为期望动作值的估计，但是该算法由于使用高斯分布作为样本均值的近似分布，计算量比较大。Double Q-learning将样本随机均分为两个部分，通过两个估计器估计最大期望Q值。通过使用神经网络[16]，可以将Double Q-learning应用于连续状态空间[17]，以提升深度强化学习的收敛速度。Zhang等[18]针对Q-learning算法的过估计问题和Double Q-learning算法的欠估计问题，提出了基于过估计和欠估计的Weighted Double Q-learning算法，该算法通过平衡过估计和欠估计来改善智能体的收敛速度。Speedy Q-learning算法通过使用连续两个状态动作Q值的估计，将其中一个保守更新准则修正为冒进更新准则，使得智能体的收敛速度得到较大提升。

综上所述，针对Q-learning收敛速度较慢的问题，当前研究的一个重点是如何估计最大期望Q值，从而提升智能体的收敛速度。本文将当前Q值的估计函数作为历史Q值的估计，同时利用双估计器可以改善智能体收敛速度的特性，提出了结合两者更新策略的新的估计策略，并给出了相关的证明，得到了改进算法Double Speedy Q-learning。

3 Q-learning及其衍生算法

本节介绍了Q-learning算法及其3种衍生算法Double Q-learning，Weighted Double Q-learning和Speedy Q-learning。Double Q-learning算法侧重于解决Q-learning算法中存在的过估计问题，以提升智能体的收敛速度。Weighted Double Q-learning通过为单估计器和双估计器赋予不同的权重值来获得新的估计策略，以获取更为准确的估计值。Speedy Q-learning算法通过改变Q-learning算法的更新准则，使用冒进更新准则来提升智能体的收敛速度。

3.1 Q-learning

Q-learning算法的描述如算法1所示。Q-learning算法把离开当前状态的及时奖励值r和下一状态s'的最大Q值$\max_{a'}Q(s',a')$作为完整的马尔可夫决策序列的收获，通过贝尔曼[19]最优方程更新Q值：

$$Q(s,a)=(1-\alpha(s,a))Q(s,a)+\alpha(s,a)(r+\gamma\max_{a'}Q(s',a')) \tag{1}$$

其中，$\alpha(s,a)\in[0,1]$是学习率，且与状态动作相关；$\gamma\in[0,1]$是用于平衡及时奖励和长期奖励的权重。

算法1 Q-learning

1. Initialize Q, s
2. Repeat:
3. Choose action a from state s based on Q and some exploration strategy(e.g., ε-greedy))
4. Take action a, observe r, s'
5. $a^* \leftarrow \arg\max_{a'} Q(s',a')$
6. $\delta \leftarrow r+\gamma Q(s',a^*)-Q(s,a)$
7. $Q(s,a) \leftarrow Q(s,a)+\alpha(s,a)\delta$
8. $s \leftarrow s'$

3.2 Double Q-learning

Double Q-learning算法的描述如算法2所示。该算法使

用表 Q^A 和表 Q^B 更新 Q 值，两个表使用随机均分的两个独立样本集进行学习。以相同的概率，随机更新 Q^A 和 Q^B 表中的 Q 值。当更新表 Q^A 时，首先从表 Q^A 中选取下一状态 s' 中最大 Q 值对应的动作 a^*，然后表 Q^B 根据表 Q^A 中选择的动作 a^* 选取下一状态的 Q 值 $Q^B(s',a^*)$，用于更新表 Q^A。更新表 Q^B 的流程和更新表 Q^A 的过程相同。

算法 2　Double Q-learning

1. Initialize Q^A, Q^B, s
2. Repeat:
3. Choose action a from state s based on Q^A and Q^B (e. g. , ε-greedy in Q^A+Q^B)
4. Take action a, observe r, s′
5. Choose(e. g. random) whether to update Q^A or Q^B
6. If update(A):
7. $a^* \leftarrow \arg\max_a Q^A(s',a)$
8. $\delta \leftarrow r+\gamma Q^B(s',a^*)-Q^A(s,a)$
9. $Q^A(s,a) \leftarrow Q^A(s,a)+\alpha^A(s,a)\delta$
10. Else if update(B):
11. $a^* \leftarrow \arg\max_a Q^B(s',a)$
12. $\delta \leftarrow r+\gamma Q^A(s',a^*)-Q^B(s,a)$
13. $Q^B(s,a) \leftarrow Q^B(s,a)+\alpha^B(s,a)\delta$
14. $s \leftarrow s'$

3.3　Weighted Double Q-learning

Weighted Double Q-learning 算法的描述如算法 3 所示。针对 Q-learning 算法的过估计问题和 Double Q-learning 算法的欠估计问题，Weighted Double Q-learning 算法通过为单估计器和双估计器分配不同的权重方式，来减少估计误差。

$$Q^A_{\max}=\beta^A Q^A(s',a^*)+(1-\beta^A)Q^B(s',a^*) \tag{2}$$

其中，$a^*=\arg\max_a Q^A(s',a)$，$Q^A_{\max}$ 表示更新表 Q^A 时最大期望 Q 值的估计。$\beta^A\in[0,1]$ 是与参数 c 有关的变量：

$$\beta^A=|Q^B(s',a^*)-Q^B(s',a_L)|/(c+|Q^B(s',a^*)-Q^B(s',a_L)|) \tag{3}$$

其中，$a_L=\arg\min_a Q^A(s',a)$，c 是一个可以根据实验环境自由调整的参数，也可以自适应变化。

算法 3　Weighted Double Q-learning

1. Initialize Q^A, Q^B, s
2. Repeat:
3. Choose action a from state s based on Q^A and Q^B (e. g. , ε-greedy in Q^A+Q^B)
4. Take action a, observe r, s′
5. Choose(e. g. random) whether to update Q^A or Q^B
6. If update(A):
7. $a^* \leftarrow \arg\max_a Q^A(s',a)$
8. $a_L \leftarrow \arg\min_a Q^A(s',a)$
9. $\beta^A=|Q^B(s',a^*)-Q^B(s',a_L)|/(c+|Q^B(s',a^*)-Q^B(s',a_L)|)$
10. $\delta \leftarrow r+\gamma[\beta^A Q^A(s',a^*)+(1-\beta^A)Q^B(s',a^*)]-Q^A(s,a)$
11. $Q^A(s,a) \leftarrow Q^A(s,a)+\alpha^A(s,a)\delta$
12. Else if update(B):
13. $a^* \leftarrow \arg\max_a Q^B(s',a)$
14. $a_L \leftarrow \arg\min_a Q^B(s',a)$
15. $\beta^B=\dfrac{|Q^A(s',a^*)-Q^A(s',a_L)|}{c+|Q^A(s',a^*)-Q^A(s',a_L)|}$
16. $\delta \leftarrow r+\gamma[\beta^B Q^B(s',a^*)+(1-\beta^B)Q^A(s',a^*)]-Q^B(s,a)$
17. $Q^B(s,a) \leftarrow Q^B(s,a)+\alpha^B(s,a)\delta$
18. $s \leftarrow s'$

3.4　Speedy Q-learning

设 $M(s',a')=\max_{a'} Q(s',a')$，$M\_(s',a')$ 表示当上一迭代步更新状态动作 (s,a) 的 Q 值时下一状态的最大 Q 值。则 Speedy Q-learning 算法的 Q 值更新公式为：

$$Q(s,a)=(1-\alpha(s,a))Q(s,a)+\alpha(s,a)(r+\gamma M\_(s',a'))+(1-\alpha(s,a))\gamma(M(s',a')-M\_(s',a')) \tag{4}$$

基于式(4)，Q-learning 算法的更新公式可以改写为：

$$Q(s,a)=(1-\alpha(s,a))Q(s,a)+\alpha(s,a)(r+\gamma M\_(s',a'))+\alpha(s,a)\gamma(M(s',a')-M\_(s',a')) \tag{5}$$

根据式(4)和式(5)可以看到，Q-learning 算法和 Speedy Q-learning 算法的主要区别在于 $(M(s',a')-M\_(s',a'))$ 前的学习率不同，Speedy Q-learning 算法使用更为冒进的学习率 $(1-\alpha(s,a))$，提升了智能体的收敛速度。

为了便于算法描述，对式(4)进行多项式合并，得到式(6)：

$$Q(s,a)=(1-\alpha(s,a))(Q(s,a)+\gamma M(s',a'))+\alpha(s,a)r+(2\alpha(s,a)-1)\gamma M\_(s',a') \tag{6}$$

Speedy Q-learning 算法的描述如算法 4 所示。

算法 4　Speedy Q-learning

1. Initialize Q(s,a)=M(s,a), s
2. Repeat:
3. Choose action a from state s based on Q and some exploration strategy (e. g. , ε−greedy)
4. Take action a, observe r, s′
5. $a^* \leftarrow \arg\max_{a'} Q(s',a')$
6. temp←(2α(s,a)−1)γM(s,a)
7. M(s,a)←Q(s′,a*)
8. Q(s,a)←(1−α(s,a))(Q(s,a)+γM(s,a))+α(s,a)r+temp
9. s←s′

4　Speedy Q-learning 的收敛性分析

本节提出了 Speedy Q-learning 算法 Q 值更新公式的等价变形，基于该等价变形分析了 Speedy Q-learning 算法收敛速度慢的原因。

4.1　Speedy Q-learning 更新公式的等价形式

设 $Q_n(s,a)$ 为第 n 次迭代时的 Q 值，则 Speedy Q-learning 算法的 Q 值更新公式为：

$$Q_{n+1}(s,a)=(1-\alpha_n)Q_n(s,a)+\alpha_n(R_n+\gamma M_{n-1}(s',a'))+(1-\alpha_n)\gamma(M_n(s',a')-M_{n-1}(s',a')) \tag{7}$$

当智能体经历的采样序列数目可计算时，可以将学习率 α_n 更换为准确学习率[20] $\dfrac{1}{n(s,a)}$，则 Speedy Q-learning 的更新公式为：

$$Q_{n+1}(s,a)=(1-\frac{1}{n(s,a)})Q_n(s,a)+\frac{1}{n(s,a)}(R_n+\gamma M_{n-1}$$

$$(s',a'))+(1-\frac{1}{n(s,a)})\gamma(M_n(s',a')-M_{n-1}(s',a')) \tag{8}$$

基于蒙特卡洛采样[20]的 Q-learning 算法的更新公式为：

$$Q_{n+1}(s,a)=\frac{1}{n(s,a)}\sum_{i=1}^{n}(R_i+\gamma M_i(s',a')) \tag{9}$$

其中，$n(s,a)\geqslant 1$，$M_0(s',a')=0$，$Q_1(s,a)=0$。

对于任意 $n(s,a)\geqslant 1$，基于式(9)对式(8)进行改写。

定理 1 式(8)等价于式(10)：

$$Q_{n+1}(s,a)=\frac{1}{n(s,a)}\sum_{i=1}^{n}(R_i+\gamma(1-\frac{1}{n(s,a)})M_n(s',a')) \tag{10}$$

证明：

当 $n=1$ 时，

$Q_2=R_1+\gamma M_0(s',a')=R_1$

假设当 $n=k$ 时式(10)成立，则有：

$$Q_k(s,a)=\frac{1}{k(s,a)-1}\sum_{i=1}^{k-1}(R_i+\gamma(1-1/(k(s,a)-1))M_{k-1}(s',a'))$$

当 $n=k+1$ 时，则有：

$$Q_{k+1}(s,a)=(1-\frac{1}{k(s,a)})Q_k(s,a)+\frac{1}{k(s,a)}(R_k+\gamma M_{k-1}(s',a'))+(1-\frac{1}{k(s,a)})\gamma(M_k(s',a')-M_{k-1}(s',a'))$$

进一步可得：

$$Q_{k+1}(s,a)=(1-\frac{1}{k(s,a)})\frac{1}{k(s,a)-1}\sum_{i=1}^{k-1}(R_i+\gamma(1-\frac{1}{k(s,a)-1})M_{k-1}(s',a'))+\frac{1}{k(s,a)}(R_k+\gamma M_{k-1}(s',a'))+(1-\frac{1}{k(s,a)})\gamma(M_k(s',a')-M_{k-1}(s',a'))$$

整理可得：

$$Q_{k+1}(s,a)=\frac{1}{k(s,a)}\sum_{i=1}^{k-1}R_i+(\frac{1}{k(s,a)}(k(s,a)-1)-(1-\frac{1}{k(s,a)}))\gamma M_{k-1}(s',a')+(1-\frac{1}{k(s,a)})\gamma M_k(s',a')$$

最终可得：

$$Q_{k+1}(s,a)=\frac{1}{k(s,a)}\sum_{i=1}^{k}R_i+(1-\frac{1}{k(s,a)})\gamma M_k(s',a')=\frac{1}{k(s,a)}\sum_{i=1}^{k}(R_i+\gamma(1-\frac{1}{k(s,a)})M_k(s',a'))$$

证毕。

4.2 Speedy Q-learning 的收敛性问题

根据式(9)和式(10)可以看到，Speedy Q-learning 将 $M_i(s')$更换为 $M_n(s')$，即把历史 Q 值的估计函数更换为当前 Q 值的估计函数，提升了算法的收敛速度。由于智能体根据学习到的经验不断优化策略，当前估计函数对期望 Q 值的估计会随着策略的改善而优于历史估计函数，因此通过将历史估计函数更换为当前估计函数可以提升智能体的收敛速度。

由于智能体在更新 Q 值时根据贪心策略来选取下一状态的最大 Q 值，并将其作为最大期望 Q 值的估计，与 Q-learning 算法一样存在过估计问题，因此在训练初期，智能体更新 Q 值的策略较差，使用当前估计函数作为历史估计函数会产生较大的误差，使得 Speedy Q-learning 算法在初始训练过程中学习到的经验较差。针对 Speedy Q-learning 算法在迭代初期策略改善慢的问题，本文基于双估计器可以避免累积正向误差，从而提升智能体的收敛速度的特性，将双估计器用于 Speedy Q-learning 算法，以此避免直接选取最大 Q 值作为最大期望 Q 值的估计，从而造成智能体收敛速度慢的问题。

5 Double Speedy Q-learning 算法

与 Double Q-learning 算法相同，Double Speedy Q-learning 算法利用两个表 Q(表 Q^A 和表 Q^B)将最优动作和 Q 值的选择分离。每次更新 Q 值时，随机更新一个表的 Q 值。当更新表 Q^A 时，从 Q^A 中选取下一状态的最优动作 $a^*=\arg\max_a Q^A(s',a)$，然后表 Q^B 根据表 Q^A 选出的动作，选择下一状态的 Q 值 $Q^B(s',a^*)$，把 $Q^B(s',a^*)$作为对最大期望 Q 值的估计。与 Double Q-learning 算法不同，Double Speedy Q-learning 算法增加了一个辅助函数 $M^A(s,a)$，用于存储前一次迭代时得到的对最大期望 Q 值的估计，通过使用当前估计 $Q^B(s',a^*)$和前一次的估计 $M^A(s,a)$更新 $Q^A(s,a)$。在更新 Q^B 时也一样，用 $M^B(s,a)$记录前一次迭代时得到的对最大期望 Q 值的估计。

Double Speedy Q-learning 算法的 Q 值的更新规则为：

$$Q_n^A(s,a)=(1-\alpha_n^A(s,a))Q_n^A(s,a)+\alpha_n^A(s,a)(R_n+\gamma Q_{n-1}^B(s',a_A^*))+(1-\alpha_n^A(s,a))\gamma(Q_n^B(s',a_A^*)-Q_{n-1}^B(s',a_A^*)) \tag{11}$$

其中，$a_A^*=\arg\max_{a'}Q^A(s',a')$，$s'$表示下一状态，$n$ 表示遍历更新状态动作对(s,a)的次数，R_n 表示获得的及时奖励值，$\alpha_n^A(s,a)$为学习率。

为了便于算法描述，对式(11)进行多项式合并，得到：

$$Q_n^A(s,a)=(1-\alpha_n^A(s,a))(Q_n^A(s,a)+\gamma Q_n^B(s',a_A^*))+\alpha_n^A(s,a)R_n+(2\alpha_n^A(s,a)-1)\gamma Q_{n-1}^B(s',a_A^*) \tag{12}$$

通过使用两个表 Q 的方式选择 Q 值，可以避免在迭代初期由单估计器造成智能体过估计的问题，从而提升智能体的收敛速度。Double Speedy Q-learning 算法的描述如算法 5 所示。

算法 5 Double Speedy Q-learning

1. Initialize $Q^A(s,a)=M^A(s,a)$, $Q^B(s,a)=M^B(s,a)$, s
2. Repeat
3. Choose a from s based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$(e.g., ε-greedy in $Q^A(s,\cdot)+Q^B(s,\cdot)$)
4. Take action a, observe r, s'
5. Choose(e.g. random) either UPDATE(A) or UPDATE(B)
6. If UPDATE(A) then
7. Define $a^*=\arg\max_a Q^A(s',a)$
8. temp←$(2\alpha^A(s,a)-1)\gamma M^A(s,a)$
9. $M^A(s,a)$←$Q^B(s',a^*)$

10. $Q^A(s,a)\leftarrow(1-\alpha^A(s,a))(Q^A(s,a)+\gamma M^A(s,a))+\alpha^A(s,a)$ r+ temp
11. If UPDATE(B) then
12. Define $b^*=\arg\max_a Q^B(s',a)$
13. temp$\leftarrow(2\alpha^B(s,a)-1)\gamma M^B(s,a)$
14. $M^B(s,a)\leftarrow Q^A(s',b^*)$
15. $Q^B(s,a)\leftarrow(1-\alpha^B(s,a))(Q^B(s,a)+\gamma M^B(s,a))+\alpha^B(s,a)$r +temp
16. End if
17. s$\leftarrow$s′
18. Until end

算法 5 和算法 2 主要的不同之处在于：算法 2 的第 8－9 行被算法 5 的第 8－10 行替换，算法 2 的第 12－13 行被算法 5 的第 13－15 行替换，即更换为 Speedy Q-learning 算法的更新策略。同时，算法 5 的第 1 行引入两个辅助函数 $M^A(s,a)$ 和 $M^B(s,a)$，分别用于存储更新表 Q^A 和表 Q^B 时，前一次遍历状态动作对(s,a)时对最大期望 Q 值的估计。对表 Q^A 和表 Q^B 的对应 Q 值求和取均值后，依据贪心策略选取最大 Q 值对应的动作，智能体执行根据该策略选取出来的最优动作进入下一状态。

6　实验及结果分析

Speedy Q-learning 和 Double Speedy Q-learning 算法收敛的一个条件是连续动作 Q 值估计的差值为 0，此时采用冒进更新准则可以加快算法的收敛速度。但是，在连续状态空间，需要使用近似函数拟合整个状态空间，会导致连续动作 Q 值估计的差值始终不为 0，算法不容易收敛。由于 Speedy Q-learning 算法和 Double Q-learning 算法的收敛性已经得到证明[11-12]，而 Double Speedy Q-learning 算法的更新策略基于以上两种算法，因此其也是可以收敛的。由于本文的关注点在算法的收敛速度上，因此选择在离散环境中展开实验。本节在格子世界环境下比较 Q-learning 算法及其改进算法的收敛速度。其中，衰减因子 γ 统一为 0.95，每次使用贪心策略选取最优动作，$\varepsilon(s)=1/\sqrt{n(s)}$，$n(s)$是状态的访问次数。

6.1　格子世界

$n\times n$ 的格子世界一共有 n^2 个状态，每一个格子代表一个状态。智能体的起始状态位于格子世界的左下角，终止状态位于格子世界的右上角。智能体在每个格子中可以有上、下、左、右 4 个动作，根据执行的动作以 100% 的概率进入临近格子。当智能体想离开格子世界时，智能体停留在当前状态。为了增加环境的随机性，智能体在非终止状态离开当前状态时，将以相同的概率获得－12 或者 10 的奖励值，当离开终止状态时，将获得 5 的奖励值。

6.2　评价指标

当智能体收敛到最优策略时，每个回合的最优路径经历的步数为$(2n-1)$，因此每一步获得的期望奖励值为：

$$\mathbb{E}(r_{ave})=(\mathbb{E}(r_{ter})+2(n-1)\times\mathbb{E}(r))/(2n-1) \tag{13}$$

其中，$\mathbb{E}(r_{ter})$表示终止状态获得的奖励值，$\mathbb{E}(r)$表示非终止状态获得的奖励值。

当智能体收敛到最优策略时，智能体从起点到终点的路径为最短路径，当智能体的迭代步数 step$\rightarrow\infty$时，每一步获得的平均奖励值 $r_{ave}\rightarrow\mathbb{E}(r_{ave})$。因此，评价指标为每一步的平均奖励值与每一步获得的期望奖励值之间的差值：

$$\Delta=\mathbb{E}(r_{ave})-r_{ave} \tag{14}$$

Δ 越小，说明智能体策略改善得越快，即收敛速度越快。

6.3　学习率

不同的学习率会使算法的收敛速度不同，根据经验可知，线性学习率和多项式学习率能够使算法获得较快的收敛速度[10]，因此本文实验比较了 Q-learning(Q)，Double Q-learning(DQ)，Weighted Double Q-learning(WDQ)，Speedy Q-learning(SQ)以及 Double Speedy Q-learning(DSQ)在线性学习率和多项式学习率下的收敛速度。其中，Weighted Double Q-learning 算法选择自适应参数时 $k=1$，选择固定参数时 $c=10$。线性学习率 $\alpha_n(s,a)=1/(n(s,a)+1)$，多项式学习率 $\alpha_n(s,a)=1/(n(s,a)+1)^{0.8}$，其中 $n(s,a)$表示 $Q(s,a)$的遍历次数。对于 Double Q-learning，Weighted Double Q-learning 和 Double Speedy Q-learning，当学习率为线性学习率时，分别使用 $a_n^A(s,a)=1/(n^A(s,a)+1)$和 $a_n^B(s,a)=1/(n^B(s,a)+1)$作为表 Q^A 和表 Q^B 更新 Q 值时的学习率。当学习率为多项式学习率时，分别使用 $a_n^A(s,a)=1/(n^A(s,a)+1)^{0.8}$ 和 $a_n^B(s,a)=1/(n^B(s,a)+1)^{0.8}$作为表 Q^A 和表 Q^B 更新 Q 值时的学习率。其中，$n^A(s,a)$和 $n^B(s,a)$分别表示 $Q^A(s,a)$和 $Q^B(s,a)$的遍历次数。

6.4　实验结果分析

图 1 和图 2 给出了迭代 10000 步时整个迭代过程中各种算法的收敛速度；图 3 和图 4 给出了是迭代 1000 步(3×3 至 4×4 格子)及 2000 步(6×6 格子)时各种算法在迭代初期的收敛速度。

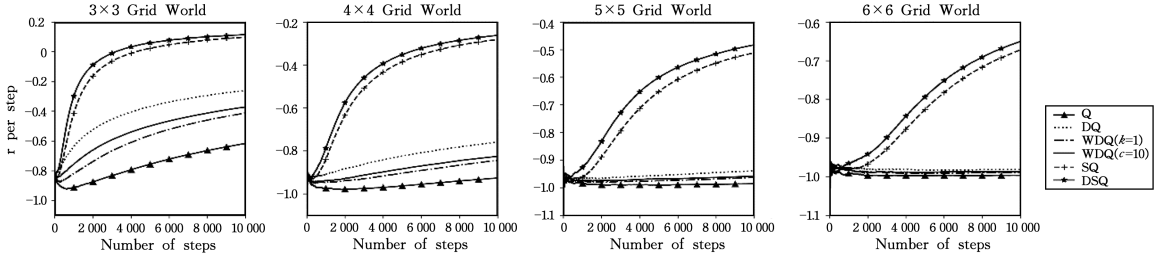


图 1　线性学习率下，Q，DQ，WDQ(k=1)，WQD(c=10)，SQ 和 DSQ 在格子世界中每步得到的 10000 次实验结果的平均奖励值

Fig. 1　Average reward values of results of 10000 experiments of Q，DQ，WDQ(k=1)，WQD(c=10)，SQ and DSQ at each step in grid world under linear learning rate

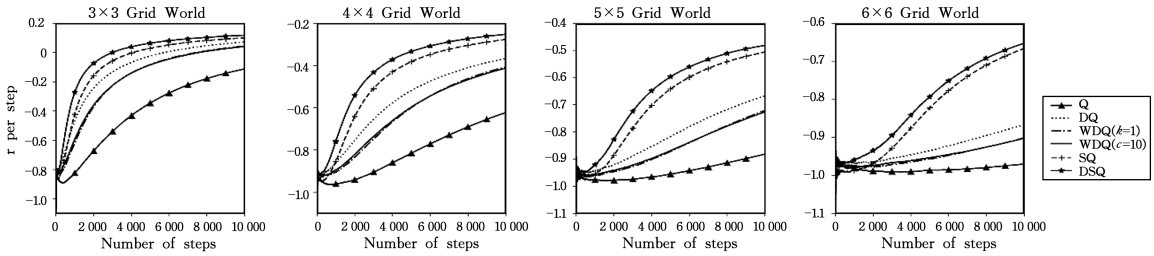
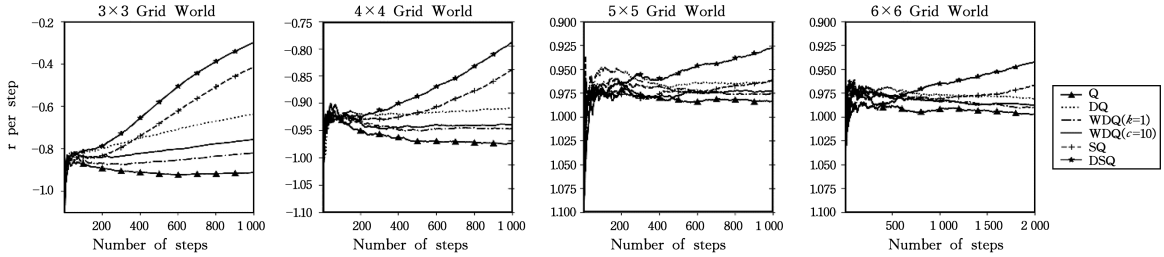


图 2 多项式学习率下，Q，DQ，WDQ($k=1$)，WQD($c=10$)，SQ 和 DSQ 在格子世界中每步得到的 10000 次实验结果的平均奖励值

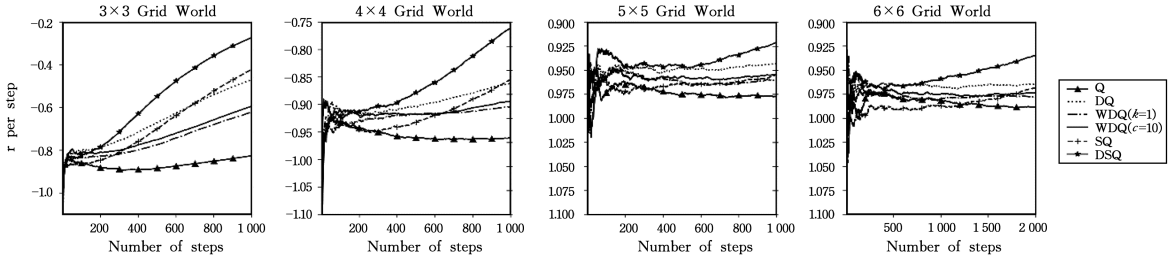
Fig. 2 Average reward values of results of 10000 experiments of Q, DQ, WDQ($k=1$), WDQ($c=10$), SQ and DSQ at each step in grid world under polynomial learning rate



注：3×3 格子世界至 5×5 格子世界每次实验迭代 1000 步，6×6 格子世界每次实验迭代 2000 步

图 3 线性学习率下，Q，DQ，WDQ(k=1)，WQD($c=10$)，SQ 和 DSQ 在格子世界中每步得到的 10000 次实验结果的平均奖励值

Fig. 3 Average reward values of results of 10000 experiments of Q, DQ, WDQ($k=1$), WQD($c=10$), SQ and DSQ at each step in grid world under linear learning rate



注：3×3 格子世界至 5×5 格子世界每次实验迭代 1000 步，6×6 格子世界每次实验迭代 2000 步

图 4 多项式学习率下，Q，DQ，WDQ($k=1$)，WQD($c=10$)，SQ 和 DSQ 在格子世界中每步得到的 10000 次实验结果的平均奖励值

Fig. 4 Average reward values of results of 10000 experiments of Q, DQ, WDQ($k=1$), WQD($c=10$), SQ and DSQ at each step in grid world under linear learning rate

图 1 给出格子世界大小分别为 3×3 到 6×6 的实验结果，学习率为线性学习率。可以看到，由于估计器不同，不同算法的收敛速度差别很大。Q-learning 算法由于过估计，在迭代初期没有合理地评估各个状态动作的价值，收敛速度十分缓慢。Double Q-learning 使用了两个估计器，因此避免了过估计问题，收敛速度比 Q-learning 快，但是随着状态空间的增大，收敛速度也变得很慢，这是因为 Double Q-learning 存在欠估计的问题。Weighted Double Q-lear-ning 算法无论是选择固定参数还是自适应参数，都难以选择到最优参数来获得准确的 Q 值，因此收敛速度也随着状态空间的增加变得十分缓慢。从图 1 可以看到，随着状态空间的增加，Speedy Q-learning 算法的收敛速度受到的影响较小，收敛速度始终较快。但是，从图 3 和图 4 可以看出，Speedy Q-learning 算法由于在一开始就将当前 Q 值的估计函数作为历史 Q 值的估计，而开始时存在过估计的问题，因此在迭代初期表现较差，特别是在多项式学习率下，初期收敛速度最慢，这限制了 Speedy Q-learning 算法的整体收敛速度。Double Speedy Q-learning 算法由于使用双估计器，避免了智能体在迭代初期产生过估计的问题，而欠估计又能较好地改善智能体的收敛速度，因此在迭代初期收敛速度明显快于 Speedy Q-learning 算法。

图 2 给出了多项式学习率下，各种算法在不同大小的格子世界中的实验结果。可以看到，Double Speedy Q-learning 算法受到学习率的影响较小，在这两种学习率下的收敛速度没有显著区别，而其他算法则受到学习率的影响较大，在多项式学习率下，Q-learning，Double Q-learning，Weighted Double Q-learning 3 种算法的收敛速度都有明显提升，但是表现依然不如 Double Speedy Q-learning 算法和 Speedy Q-learning 算法，而且随着状态空间的增加，Double Speedy Q-learning 算法的优势更加明显。

表 1 和表 2 对比了各种算法迭代 10 000 步后，每一步的平均奖励值和期望获得奖励值之间的差值。可以看到，Double Speedy Q-learning 算法的 Δ 都是最小的，这说明 Double Speedy Q-learning 收敛到最优策略的速度比其他算法快。

表 1　线性学习率下实际奖励值与期望奖励值之间的差值

Table 1　Difference between actual reward value and expected reward value under linear learning rate

$\Delta=\mathbf{E}(r_{ave})-r_{ave}$	3×3	4×4	5×5	6×6
Q	0.8169	0.7832	0.6523	0.5420
DQ	0.4603	0.6166	0.6056	0.5273
WDQ(c=10)	0.6124	0.7012	0.6297	0.5341
WDQ(k=1)	0.5716	0.6822	0.6246	0.5327
SQL	0.1021	0.1388	0.1787	0.2179
DSQ	**0.0848**	**0.1191**	**0.1498**	**0.1965**

表 2　多项式学习率下实际奖励值与期望奖励值之间的差值

Table 2　Difference between actual reward value and expected reward value under polynomial learning rate

$\Delta=\mathbf{E}(r_{ave})-r_{ave}$	3×3	4×4	5×5	6×6
Q	0.3126	0.4804	0.5486	0.5159
DQ	0.1280	0.2239	0.3346	0.4125
WDQ(c=10)	0.1553	0.2642	0.3881	0.4472
WDQ(k=1)	0.1593	0.2701	0.3932	0.4479
SQL	0.1000	0.1342	0.1728	0.2120
DSQ	**0.0821**	**0.1094**	**0.1483**	**0.1983**

结束语　本文根据蒙特卡洛强化学习 Q 值的更新公式，对 Speedy Q-learning 算法的更新公式做了变形，分析了 Speedy Q-learning 算法可以改善收敛速度的原因以及存在的问题，结合 Double Q-learning 算法，将双估计器应用在 Speedy Q-learning 算法中，利用双估计器可以改善智能体收敛速度的特性，提出了 Double Speedy Q-learning 算法。实验结果表明，改进算法可以提升智能体的收敛速度。针对 Speedy Q-learning 算法和 Double Speedy Q-learning 算法在连续状态空间中并不易收敛的问题，未来将进一步研究其在连续状态空间中的优化。

参 考 文 献

[1] MAO H, ALIZADEH M, MENACHE I, et al. Resource Management with Deep Reinforcement Learning[C]// ACM Workshop on Hot Topics in Networks. ACM, 2016: 50-56.

[2] BRAFMAN R I, TENNENHOLTZ M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning[J]. Journal of Machine Learning Research, 2002, 3(Oct): 213-231.

[3] WATKINS C J C H, DAYAN P. Technical Note: Q-Learning [J]. Machine Learning, 1992, 8(3/4): 279-292.

[4] BERTSEKAS D P. Stable optimal control and semicontractive dynamic programming[J]. SIAM Journal on Control and Optimization, 2018, 56(1): 231-252.

[5] SCHEFFLER K, YOUNG S. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning [C]// Proceedings of the second international conference on Human Language Technology Research. San Francisco: Morgan Kaufmann Publishers Inc, 2002: 12-19.

[6] YANG G S, CHEN E K, AN C W. Mobile robot navigation using neural Q-learning[C]// Proceedings of 2004 International Conference on Machine Learning and Cybernetics. New York: IEEE, 2004: 48-52.

[7] WANG Y C, USHER J M. Application of reinforcement learning for agent-based production scheduling[J]. Engineering Applications of Artificial Intelligence, 2005, 18(1): 73-82.

[8] DJONIN D V, KRISHNAMURTHY V. Q-Learning Algorithms for Constrained Markov Decision Processes With Randomized Monotone Policies: Application to MIMO Transmission Control [J]. IEEE Transactions on Signal Processing, 2007, 55(5): 2170-2181.

[9] EVEN-DAR E, MANSOUR Y. Learning rates for Q-learning [J]. Journal of Machine Learning Research, 2003, 5(Dec): 1-25.

[10] SZEPESVÁRI C. The asymptotic convergence - rate of Q-learning[C]// Advances in Neural Information Processing Systems. MIT Press, 1998: 1064-1070.

[11] HASSELT H V. Double Q-learning[C]// Advances in Neural Information Processing Systems. MIT Press, 2010: 2613-2621.

[12] GHAVAMZADEH M, KAPPEN H J, AZAR M G, et al. Speedy Q-learning[C]// Advances in Neural Information Processing Systems. MIT Press, 2011: 2411-2419.

[13] LEE D, POWELL W B. An intelligent battery controller using bias-corrected Q-learning[C]// Twenty-Sixth AAAI Conference on Artificial Intelligence. AAAI Press, 2012: 316-322.

[14] STREHL A L, LI L, WIEWIORA E, et al. PAC model-free reinforcement learning[C]// Proceedings of the 23rd International Conference on Machine Learning. ACM, 2006: 881-888.

[15] D'ERAMO C, RESTELLI M, NUARA A. Estimating maximum expected value through gaussian approximation[C]// International Conference on Machine Learning. 2016: 1032-1040.

[16] LE CUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553): 436.

[17] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning[C]// Thirtieth AAAI Conference on Artificial Intelligence. 2016: 2094-2100.

[18] ZHANG Z, PAN Z, KOCHENDERFER M J. Weighted Double Q-learning[C]// International Joint Conferences on Artificial Intelligence Organization. 2017: 3455-3461.

[19] BELLMAN R. Dynamic programming [J]. Science, 1966, 153 (3731): 34-37.

[20] SUTTON R S, BARTO A G. Reinforcement learning: An introduction[M]. MIT Press, 1998: 107-127.

ZHENG Shuai, born in 1994, postgraduate. His main research interests include reinforcement learning and so on.

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