连续空间中的一种动作加权行动者评论家算法

刘 全[1),2),3)] 章 鹏[1)] 钟 珊[1)] 钱炜晟[1)] 翟建伟[1)]

[1)](苏州大学计算机科学与技术学院 江苏 苏州 215006)

[2)](软件新技术与产业化协同创新中心 南京 210000)

[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)

摘 要 经典的强化学习算法主要应用于离散状态动作空间中.在复杂的学习环境下,离散空间的强化学习方法不能很好地满足实际需求,而常用的连续空间的方法最优策略的震荡幅度较大.针对连续空间下具有区间约束的连续动作空间的最优控制问题,提出了一种动作加权的行动者评论家算法(Action Weight Policy Search Actor-Critic,AW-PS-AC).AW-PS-AC算法以行动者评论家为基本框架,对最优状态值函数和最优策略使用线性函数逼近器进行近似,通过梯度下降方法对一组值函数参数和两组策略参数进行更新.对两组策略参数进行加权获得最优策略,并对获得的最优动作通过区间进行约束,以防止动作越界.为了进一步提高算法的收敛速度,设计了一种改进的时间差分算法,即采用值函数的时间差分误差来更新最优策略,并引入了策略资格迹调整策略参数.为了证明算法的收敛性,在指定的假设条件下对AW-PS-AC算法的收敛性进行了分析.为了验证AW-PS-AC算法的有效性,在平衡杆和水洼世界实验中对AW-PS-AC算法进行仿真.实验结果表明AW-PS-AC算法在两个实验中均能有效求解连续空间中近似最优策略问题,并且与经典的连续动作空间算法相比,该算法具有收敛速度快和稳定性高的优点.

关键词 强化学习;连续空间;函数逼近;行动者评论家;梯度下降;人工智能

中图法分类号 TP18 **DOI号** 10.11897/SP.J.1016.2017.01252

An Improved Actor-Critic Algorithm in Continuous Spaces with Action Weighting

LIU Quan[1),2),3)] ZHANG Peng[1)] ZHONG Shan[1)] QIAN Wei-Sheng[1)] ZHAI Jian-Wei[1)]

[1)](*School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu* 215006)

[2)](*Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing* 210000)

[3)](*Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun* 130012)

Abstract Classic reinforcement learning algorithms mainly aim at the discrete state and action spaces. For the complex environment or the more applicable continuous spaces, the methods for the discrete spaces cannot satisfy the requirement. One feasible method is to discretize the state and action spaces, then the methods applied in discrete spaces can solve these problems with continuous state and action spaces. However, the reasonable discretization for the state and action spaces is not an easy problem. The methods applicable in continuous spaces do not have to discretize the state or action spaces, but most of them did not consider the constraint of the action range, additionally, the fluctuations of the optimal action were heavily. To be more applicable in continuous action spaces, we propose an actor-critic algorithm for continuous action space based

收稿日期:2016-04-15;在线出版日期:2016-12-06.本课题得到国家自然科学基金(61472262,61502323,61502329)、江苏省自然科学基金(BK2012616)、江苏省高校自然科学研究项目(13KJB520020)、吉林大学符号计算与知识工程教育部重点实验室基金项目(93K172014K04)、苏州市应用基础研究计划工业部分(SYG201422,SYG201308)资助.**刘 全**,男,1969年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为智能信息助理、自动推理和机器学习.E-mail:quanliu@suda.edu.cn.**章 鹏**,男,1992年生,硕士研究生,主要研究方向为连续空间强化学习.**钟 珊**,女,1983年生,博士研究生,主要研究方向为机器学习和深度学习.**钱炜晟**,男,1992年生,硕士研究生,主要研究方向为部分可观察马尔可夫模型.**翟建伟**,男,1992年生,硕士研究生,主要研究方向为深度强化学习和深度学习.

on weighting of the actions by considering the constraint of the action range and decreasing the fluctuation, called AW-PS-AC. AW-PS-AC is designed in the framework of the actor-critic which is a classic method for the continuous space. The action exploration policy takes the Gaussian distribute by using the optimal action as the mean value, so that the selective action is the action with a small exploration factor. The optimal state value function and the optimal policy are approximated by linear function approximation, where the gradient descent method is utilized to update one set of the value function parameter and two sets of the policy parameters. The two sets of the policy parameters are weighted to obtain the optimal policy to constraint the optimal action, so that the optimal action will not surpass the action range and the optimal policy will not fluctuate significantly. The weighting for the actions can satisfy the constraint of the action range. Moreover, the samples can be utilized more comprehensively, resulting in a better performance under only a small amount of the data. To speed the convergence rate, an improved temporal difference algorithm is designed, where the temporal difference error (TD-error) of the value function are employed to update the optimal policy and the policy eligibility trace is introduced to improve the convergence rate for the algorithm. To prove the convergence of this proposed method, under the three given assumptions, AW-PS-AC is analyzed theoretically and its convergence is proved. On two classic benchmarks of the classic reinforcement learning benchmarks which have the nonlinear system dynamics, pole-balancing problem and puddle world problem, AW-PS-AC is compared with the representative methods which are representative in continuous spaces, namely, continuous actor-critic learning automaton (CALAC), continuous-action on Q-learning (CAQ) and incremental natural actor-critic with scaling gradient (INAC-S), and they are implemented on them. The results show that the AW-PS-AC algorithm performs well in the two experiments. The good performances in the two experiments demonstrate that the AW-PS-AC algorithm can solve the approximated-optimal problems effectively in continuous space. Compared with the state-of-the-art algorithms, AW-PS-AC outperforms them not only in convergence but also in stability. From the experiments, it is clearly that AW-PS-AC algorithm can converge only after only a few episodes, moreover, it can be stable all the time after it is converged.

Keywords　reinforcement learning; continuous spaces; function approximation; actor-critic; gradient descent; artificial intelligence

1　引　言

强化学习(Reinforcement Learning,RL),又名激励学习、增强学习,是指从状态到动作映射的一种学习,其目标是通过最大化累计奖赏来寻找最优策略[1].近年来,强化学习在大规模空间上获得极大进展,在实际生产和生活中的运用也日益广泛,在人工智能、自动控制、运筹学、心理学、统计学和遗传算法等多个研究领域中均引起了广泛的关注.

在强化学习中常用表格式存储方式保存策略,这种存储方式具有简单有效的优点,但在状态动作空间规模较大或状态动作空间连续时,会出现"维数灾难"的问题.为了解决该问题,可以将传统的强化学习算法与函数逼近方法相结合,通过函数逼近方法来近似表示值函数和策略,使学习到的经验能够泛化到整个状态和动作空间[1-4].近年来,连续动作空间的强化学习研究逐渐增多,成为强化学习领域中的热点之一.Sutton 等人[5]首次将策略梯度和强化学习动作表示相结合,提出了强化学习策略梯度函数逼近方法.Peters 等人[6,7]将自然梯度运用在函数逼近中,并与强化学习中时间差分最小二乘算法相结合,提出了基于自然梯度的行动者评论家算法(Natural Actor-Critic,NAC).Millán 等人[8]将离散动作空间中获取的 Q 值函数直接运用在动作选择中,并利用资格迹获得最优动作,提出连续动作空间 Q 学习算法(Continuous-Action on Q-learning,CAQ).van Hasselt 等人[9]提出了针对连续空间的

行动者评论家自动学习机算法(Continuous Actor-Critic Learning Automaton,CACLA).该算法通过时间差分误差的符号决定更新是否执行,尽可能避免策略向更差的方向更新,从而提高算法效率.Martin 等人[10]提出了使用 K 最近邻分类的时间差分算法.该方法使用 K 最近邻分类对状态空间进行离散化,使用状态对应类别的分类器进行加权求和获得最优动作.

然而,以上连续动作空间强化学习算法依然存在以下问题:

(1)常见的连续空间算法多用常规的线性方法来选择动作,其具有每轮迭代的计算量和存储量少、运行速度快的优点,然而依然存在总体逼近效果较差、逼近所需情节数较多的问题.

(2)通常情况下动作空间具有上下界约束,而常用的动作逼近方法不考虑动作空间的区间约束.

(3)大多数连续动作空间算法并不考虑优化策略表示方式,而好的策略表示方式可以大幅减少与环境交互所需要的情节数,从而提高收敛速度.

为了解决以上问题,本文提出了一种动作加权行动者评论家算法.该算法使用动作加权法选取当前最优动作,解决了选择的动作不在动作空间的问题.通过策略参数直接获取最优策略,无需求解状态动作值函数就可以直接获取当前状态下应选择的动作,减少了存储量和计算量.使用改进的梯度下降算法来更新策略参数向量,将时间差分误差作为策略参数更新的重要依据,并用动作资格迹对策略进行优化以加速其更新.最后选择了两个经典的强化学习基准实验,平衡杆和水洼世界对该算法进行验证.并将 AW-PS-AC 算法和 3 种经典的连续动作空间强化学习算法 CACLA,INAC-S,CAQ 进行比较,实验结果表明 AW-PS-AC 算法具有收敛速度快和实验效果稳定的优点.

2 背景知识

2.1 马尔可夫决策过程

将满足马尔可夫性质的强化学习任务称为马尔可夫决策过程(Markov Decision Processes,MDP)[1,11].马尔可夫决策过程问题可以用四元组 $M=\langle X,U,\rho,f\rangle$ 表示.

(1) X 表示状态集合,x_t 表示 agent 在第 t 时间步时所处的状态.

(2) U 表示动作集合,u_t 表示 agent 在第 t 时间步时采取的动作.

(3) $\rho:X\times U\times X\to\mathbb{R}$ 表示奖赏函数.$r(x_t,u_t,x_{t+1})$ 表示 agent 在状态 x_t 执行动作 u_t 后转移到 x_{t+1} 得到的立即奖赏.

(4) $f:X\times U\times X\to[0,1]$ 表示状态转移函数,$f(x_t,u_t,x_{t+1})$ 表示 agent 在状态 x_t 执行动作 u_t 时转移到状态 x_{t+1} 的概率.

Agent 通过策略 h 来选择第 t 时间步的动作 u_t,$u_t=h(x_t)$.给定奖赏函数 ρ,状态转移函数 f,第 t 时间步的状态 x_t 和动作 u_t,就可确定下一时间步的状态 x_{t+1} 和立即奖赏 r_{t+1},这被称为马尔可夫性质.

强化学习的目标是寻找最优策略 h^*,即最大化以任一状态为初始状态的累计奖赏.折扣累计奖赏的计算公式如式(1)所示.

$$R^h(x_0)=\sum_{k=0}^{\infty}\gamma^k r_{k+1}=\sum_{k=0}^{\infty}\gamma^k\rho(x_k,h(x_k)) \quad (1)$$

其中 γ 为折扣因子,$\gamma\in(0,1)$.折扣因子可以表示 agent对未来奖赏的考虑程度.

V 值函数,又称状态值函数,指的是从状态 x 开始根据策略 h 获得的累计奖赏.连续空间下 V 函数的计算方法如式(2)所示.

$$V^h(x)=\int_{u\in U}h(x,u)\int_{x'\in X}f(x,u,x')[r+\gamma V^h(x')]\mathrm{d}x'\mathrm{d}u \quad (2)$$

其中 $r=\rho(x,u,x')$.

对于任意状态 x 和策略 h,存在某策略 h^*,使得 $V^{h^*}(x)\geq V^h(x)$ 恒成立,则称此时的 h^* 为最优策略,此时的 V 值函数为最优 V 值函数,记为 V^* 值函数.最优 V 值函数如式(3)所示.

$$V^*(x)=\max_h V^h(x) \quad (3)$$

2.2 行动者评论家方法

根据 agent 选择动作方法的不同,可以把强化学习方法分为三大类:行动者方法(Actor-only)、评论家方法(Critic-only)和行动者评论家方法(Actor-Critic).行动者评论家方法是由行动者和评论家两个部分构成.行动者用于选择动作,评论家评论选择动作的好坏.行动者选择的动作不是依据当前的值函数计算得出,而是直接从存储的策略中获得.评论家一般采用时间差分误差的形式进行评论.这个信号是评论家的唯一输出,并且驱动了行动者和评论家之间的所有学习[12,13].行动者评论家算法结构如图 1 所示.

传统的行动者评论家方法多用于离散状态空间

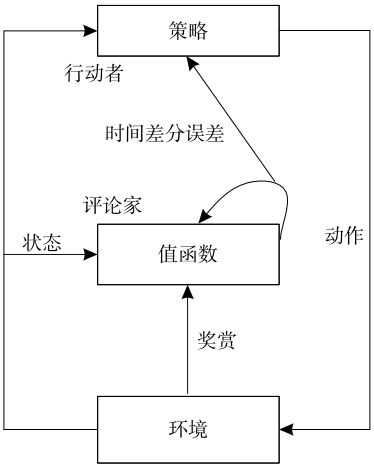

图 1　行动者评论家方法结构图

和离散动作空间中. 行动者根据查询表选择当前状态下应执行的动作. 查询表中存储每个状态动作对所对应的偏好度 $p(x,u)$. 根据偏好度可以计算出当前状态下每个动作被选择的概率. 常用的动作选择方法为 Gibbs 软最大化方法. Gibbs 软最大化方法如式(4)所示.

$$h(x,u)=\frac{e^{p(x,u)}}{\sum_{u'\in U}e^{p(x,u')}} \tag{4}$$

评论家通常用时间差分误差来评论动作选择的好坏. 评论家得到动作时，根据值函数计算出时间差分误差. 时间差分误差的计算公式如式(5)所示.

$$\delta_t=r_{t+1}+\gamma V(x_{t+1})-V(x_t) \tag{5}$$

一种常见的调整偏好度方法如式(6)所示.

$$p(x_t,u_t)\leftarrow p(x_t,u_t)+\beta\delta_t \tag{6}$$

其中 β 表示调整步长参数.

使用行动者评论家方法，可以将值函数的计算和策略的获取进行剥离，可以同时对值函数和策略进行学习. 和普通值函数方法相比，行动者评论家方法动作选择的计算量少，策略变化较为平滑，时间复杂度低，不会因为某一次值函数的更新使得策略有较大幅度的改变.

3　连续空间中的一种动作加权行动者评论家算法

AW-PS-AC 算法以行动者评论家方法为基本结构. 在大规模空间中，使用线性函数逼近器逼近最优值函数，使用动作加权法逼近最优策略，将资格迹运用在值函数和策略的更新中. 利用梯度下降方法最小化状态均方误差函数和动作均方误差函数，并利用时间差分误差优化更新公式，最终得到最优参数向量组. 该算法在连续动作空间下，能够又快又准地得到最优策略.

3.1　状态值函数表示

线性方法是常用的近似状态值函数的方法. 用线性函数逼近值函数的方法如式(7)所示.

$$V(x)=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(x)=\sum_{i=1}^{n}\theta_i\cdot\phi_i(x) \tag{7}$$

其中：$\boldsymbol{\theta}$ 表示值函数参数向量；$\boldsymbol{\phi}(x)$ 表示状态 x 的特征向量. 线性函数逼近方法能够保证所有收敛或接近局部最优的 $\boldsymbol{\theta}$ 一定是收敛或者接近全局最优，并且当算法满足在策略(On-Policy)时，使用线性方法求得的解将与使用 TD(0)算法得到的解相同[1]. 因此，大多数函数逼近强化学习系统均使用线性函数逼近方法.

最小化值均方误差(Mean Squared Value Error，MSVE)可以评价当前值函数参数向量的好坏. MSVE 越小，说明通过值函数参数向量求得的值函数与真实值函数的差距越小，即说明当前值函数越准确. 当值函数在线性表示下，对值均方误差使用梯度下降得到的值函数参数向量更新公式如式(8)所示.

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\alpha\delta\boldsymbol{\phi}(x) \tag{8}$$

其中：α 表示步长参数；δ 表示时间差分误差.

使用资格迹可以加快时间差分算法的收敛速度. 资格迹可以当作一步时间差分算法和蒙特卡罗算法之间的桥梁. 时间差分算法经过资格迹的扩展，将原本对一步更新的时间差分误差的计算优化为对平均 n 步更新值函数增量的计算，使得 agent 能够向前看到未来所有的奖赏并把未来奖赏更好地进行结合.

资格迹主要分为累加迹和替代迹两种. 式(9)为累加迹(Accumulating Trace)的定义.

$$\boldsymbol{e}_{t+1}=\gamma\lambda\boldsymbol{e}_t+\boldsymbol{\varphi}(x_t) \tag{9}$$

其中：γ 是折扣系数；λ 是迹-衰减(Trace-Decay)参数，满足条件 $0\leq\lambda<1$. $\boldsymbol{\varphi}(x)=\partial V(x)/\partial\boldsymbol{\theta}$. 资格迹一般用来提高强化学习算法的运算效率. 在使用较少的存储量和计算量的情况下，使用资格迹方法可以大幅减少总迭代次数，加快收敛速度.

3.2　策略表示

探索是强化学习问题中的重要元素. 如何平衡探索和利用的问题是强化学习中的需要重点研究的课题之一. 如果仅利用当前得到的最优策略去指导 agent 行动，当学习不够充分时，算法容易陷入局部最优. 而探索可以使得 agent 在寻找最优策略时，考虑除了最优动作以外的其它动作，让算法不易陷入

局部最优. agent 通过探索可能会发现比当前策略更好的策略. 由于在连续动作空间下,难以估计所有状态动作对的偏好度,因此难以将式(4)所示的 Gibbs 软最大化方法作为探索策略. 连续空间中常用的探索策略方法是高斯策略探索方法,即在近似最优动作的周围进行高斯探索. 假设 $A_t(x_t)$ 为在第 t 次迭代下状态 x_t 处的近似最优动作,则此时状态 x_t 下每个动作被选择的概率如式(10)所示.

$$h_t(x_t,u)=\frac{1}{\sqrt{2\pi}\sigma}e^{-(u-A_t(\boldsymbol{x}_t))^2/2\sigma^2} \tag{10}$$

其中 σ 表示探索的标准差.

在式(10)所示的动作选择概率中,第 t 次迭代时实际的选择动作 u_t 满足 $u_t\sim N(A_t(x_t),\sigma^2)$. 其中 $N(A_t(x_t),\sigma^2)$ 为均值为 $A_t(x_t)$,标准差为 σ 的正态分布.

从式(10)中可得出当动作在 $A_t(x_t)$ 周围时,被选择概率较大,而远离最优动作的动作依然会以较小的概率被探索到,探索程度受标准差参数 σ 影响. 通过高斯分布获取的策略,可以直接得到当前状态应选择的动作. 与常用的动作离散化相比,无需计算和存储大量的状态动作值函数,减少了存储量,加快了收敛速度.

在行动者评论家方法中时间差分误差影响策略的更新. 假设 agent 当前到达的状态为 x,执行两个不同的动作 u_1 和 u_2,得到的立即奖赏分别为 r_1 和 r_2,对应的时间差分误差为 $\delta(x,u_1)$ 和 $\delta(x,u_2)$. 如果时间差分误差满足 $\delta(x,u_1)>\delta(x,u_2)$. 根据时间差分误差定义可得式(11).

$$r_1+\gamma V(x_1)-V(x)>r_2+\gamma V(x_2)-V(x) \tag{11}$$

式(11)变形后可以得到式(12).

$$r_1+\gamma V(x_1)>r_2+\gamma V(x_2) \tag{12}$$

其中 x_1,x_2 分别表示在状态 x 处执行动作 u_1 和 u_2 得到的下一个状态.

由式(12)和值函数定义可得到此时执行动作 u_1 优于执行动作 u_2. 若 u_1 表示实际选择的动作,u_2 表示在状态 x 处根据当前策略得到的最优动作,那么当 $\delta(x,u_1)>\delta(x,u_2)$ 时,则希望当前策略向动作 u_1 方向调整. 当 $\delta(x,u_1)<\delta(x,u_2)$ 时,则希望当前策略向远离动作 u_1 的方向调整.

3.3 最优动作表示

在连续动作空间中,常用的近似策略方法是先使用线性函数逼近器获取当前近似最优动作,再通过平衡探索和利用来获取当前近似策略. 类似于式(7),通常近似最优动作的选择方法可由式(13)表示.

$$A(x)=\boldsymbol{\psi}^{\mathrm{T}}\cdot\boldsymbol{\phi}(x)=\sum_{i=1}^{n}\psi_i\cdot\phi_i(x) \tag{13}$$

其中:$\boldsymbol{\phi}(x)$ 表示状态 x 的特征向量;$\boldsymbol{\psi}$ 表示策略参数向量.

使用线性逼近器直接近似最优策略是常规的获取最优动作的方法,如式(13)所示. 这类方法具有计算量小、存储量少的优点. 当策略向量的维数为 m 时,其在求解策略参数时计算和存储开销仅为 $O(m)$. 然而,这些方法需要较多的情节数才能实现收敛,从而限制了其在实际中的进一步应用. 因此,如何改善最优动作逼近方法从而提高算法收敛速度,是强化学习中亟待解决的一个关键问题. 通常情况下,值函数范围是未知的,而动作空间的范围大多是已知的,因此,可以利用动作空间的知识来求解最优策略. 为了充分地利用已知的动作空间,本文提出了动作加权法,利用两个策略权值求解当前最优动作,使得值函数的逼近方法和最优动作的逼近方法产生区别,大幅减少收敛所需的情节数,提高算法的收敛速度. 由于动作加权法仅需要额外存储和计算一个 m 维的参数向量,因此,在策略参数求解过程中的计算和存储开销仍为 $O(m)$. 从以上内容可知,动作加权法能在计算和存储开销不变的情况下,充分利用已知的动作空间以提高算法的收敛速度,从而满足实际需求.

假设动作空间为 $[u_{\min},u_{\max}]$,通过策略权值方法可以保证选择的动作在动作空间中. 定义状态 x 的策略权值分别为 $w_1(x)$ 和 $w_2(x)$,此时状态 x 的最优动作表示如式(14)所示.

$$A(x)=w_1(x)\cdot u_{\min}+w_2(x)\cdot u_{\max}$$
$$\text{s.t. } 0\le w_1(x),w_2(x)\le 1,w_1(x)+w_2(x)=1 \tag{14}$$

通过对策略权值 $w_1(x)$ 和 $w_2(x)$ 的范围控制,可以保证状态 x 的最优动作 $A(x)$ 在动作空间 $[u_{\min},u_{\max}]$ 中. 使用线性方法可以对策略权值 w 进行计算. 如式(15)所示.

$$\begin{cases}w_1(x)=\boldsymbol{\psi}_1^{\mathrm{T}}\cdot\boldsymbol{\phi}(x)\\ w_2(x)=\boldsymbol{\psi}_2^{\mathrm{T}}\cdot\boldsymbol{\phi}(x)\end{cases} \tag{15}$$

式(15)与式(13)类似,将原本直接对最优动作的计算改为对策略权值的计算. 最终需要用归一化方法保证 $w_1(x)+w_2(x)=1$,确保求得的动作在动作空间中. 策略权值的更新如式(16)所示.

$$\begin{cases}w_1(x)=\dfrac{\boldsymbol{\psi}_1^{\mathrm{T}}\cdot\boldsymbol{\phi}(x)}{\boldsymbol{\psi}_1^{\mathrm{T}}\cdot\boldsymbol{\phi}(x)+\boldsymbol{\psi}_2^{\mathrm{T}}\cdot\boldsymbol{\phi}(x)}\\ w_2(x)=1-w_1(x)\end{cases} \tag{16}$$

在实际的动作选择中需要一定的探索来保证当

前策略不会陷入局部最优. 可以将式(10)的探索机制加入到式(15)当中，然后再进行归一化操作. 这样可以确保根据策略选择的动作与最优动作有一定的差值. 使用动作加权法选择的最优动作只会以较少的频率出现 $w_i(x)<0$ 或 $w_i(x)>1$ 的情况，此时可将策略权值定义为边界值，即当根据式(14)计算出的最优动作 $A(x)$ 小于 $u_{\min}$ 或大于 $u_{\max}$ 时，令其等于 $u_{\min}$ 或 $u_{\max}$. 由此，AW-PS-AC 算法中选择的动作不在动作空间中的频率远小于使用一般线性方法选择动作出现该问题的频率，从而可以更有效地利用实验数据，和其它算法相比，当情节数较少时有更好的收敛效果.

3.4　函数逼近方法

本文以最小化动作均方误差为目标，采用梯度下降方法来解决函数逼近的问题. 通常采用式(17)来表示动作均方误差 *MSAE*(Mean-Square Action Error).

$$MSAE(\boldsymbol{\psi})=\int_{x\in X} d^h(x)\int_{u\in U} p(u|x)(A(x)-u)^2\mathrm{d}u\mathrm{d}x \tag{17}$$

其中，$d^h(x)$ 表示在策略 h 下状态 x 的概率分布. $p(u|x)$ 表示在状态 x 下动作 u 被选择的概率. $A(x)-u$ 表示在状态 x 下当前最优动作 $A(x)$ 与实际选择动作 u 的差值.

由于在强化学习中动作选择必须要存在探索，所以 $MSAE>0$. 当执行选择的动作后，均方误差越小表示最优动作和实际选择动作更加接近，该情况表明当前策略越好. 因此，函数逼近的目标之一是使得动作均方误差最小化.

然而，绝大多数强化学习问题中状态概率分布 $d^h(x)$ 难以获得，而且直接使用式(17)会产生巨大的计算量和复杂的积分计算，在实际情况中很难实现. 然而当样本足够多，每个状态访问的次数足够多时，根据大数定律可得此时每个状态出现的频率会与 $d^h(x)$ 相一致. 并且当每个状态的动作选择满足当前策略时，最终得到的某一状态下动作选择的频率也会和 $p(u|x)$ 相近. 因此，当样本足够多时则无需考虑 $d^h(x)$ 和 $p(u|x)$，可直接将所有样本得到的动作差值进行平方求和作为评价策略参数的标准. 这不仅降低了对环境信息的依赖性，而且大幅减少了计算量，同时可以保证计算的准确性.

使用梯度下降法可以有效地减少动作均方误差. 通过对策略参数向量组分别进行梯度下降可求解策略参数向量的最优值. 使用梯度下降后策略参数的更新如式(18)所示.

$$\begin{cases}\boldsymbol{\psi}_1=\boldsymbol{\psi}_1+\beta\delta_u w_1(x)\cdot\boldsymbol{\phi}(x)\\ \boldsymbol{\psi}_2=\boldsymbol{\psi}_2+\beta\delta_u w_2(x)\cdot\boldsymbol{\phi}(x)\end{cases} \tag{18}$$

其中 β 表示 $\boldsymbol{\psi}_1$ 和 $\boldsymbol{\psi}_2$ 更新的步长参数. δ_u 表示选择动作 u 和最优动作 $A(x)$ 的差值.

为了提高收敛速度，可以在动作空间中使用资格迹来调整梯度下降的方向. 使用资格迹可使得当前动作的选择受到之前选择过的动作的影响. 动作空间中资格迹的更新如式(19)所示.

$$\begin{cases}\boldsymbol{e}_{u\_1}=\gamma\lambda\boldsymbol{e}_{u\_1}+w_1(x)\cdot\boldsymbol{\phi}(x)\\ \boldsymbol{e}_{u\_2}=\gamma\lambda\boldsymbol{e}_{u\_2}+w_2(x)\cdot\boldsymbol{\phi}(x)\end{cases} \tag{19}$$

当使用高斯分布来选择动作时，δ_u 越大表示探索程度越大，此时选择的动作有很大概率是较差动作. 当 δ_u 较大时使用该动作更新策略参数向量，会使得参数向量沿着梯度方向移动较长的距离. 对于当前策略，如果由于探索的存在，选择了较差动作，这会最终导致学习到的策略参数向量会比原来的差. 在 3.2 节中提到时间差分误差 δ 的大小可以表示当前选择动作的好坏，因此可以将 δ 运用在动作选择上. 可以使用时间差分误差 δ 表示沿着值函数梯度下降的长度，代替原本的动作差值 δ_u. 最终优化后的 $\boldsymbol{\psi}$ 的更新如式(20)所示.

$$\begin{cases}\boldsymbol{\psi}_1=\boldsymbol{\psi}_1+\beta\delta\boldsymbol{e}_{u\_1}\\ \boldsymbol{\psi}_2=\boldsymbol{\psi}_2+\beta\delta\boldsymbol{e}_{u\_2}\end{cases} \tag{20}$$

在式(20)中，若 $\delta>0$，则 $\boldsymbol{\psi}$ 向梯度下降方向移动，动作被选取概率增加. 若 $\delta<0$，说明选择的动作较差，则 $\boldsymbol{\psi}$ 向梯度下降方向的相反方向移动，动作被选取概率降低. 在式(18)中梯度下降的长度仅仅与动作差值有关，而式(20)中 $\boldsymbol{\psi}$ 受到可以评价动作好坏的 δ 影响，从而评论家的评论可以优化行动者动作的选择，符合行动者评论家方法的要求. 根据式(19)和(20)可以看出，策略权值可以控制策略向量向梯度方向更新的幅度，会使策略更新对策略步长参数的依赖减少. 减少参数依赖也是本算法的一个优点.

3.5　算法描述和相关比较

AW-PS-AC 算法是从行动者评论家算法发展而来的，行动者使用当前策略权值通过动作加权法获取最优动作，再利用高斯概率分布得到当前策略和实际执行动作. 评论家通过计算时间差分误差来评价当前选择动作的好坏，并且将动作的好坏表现在策略参数的更新上. AW-PS-AC 算法与其它经典的连续动作行动者评论家算法有以下两点不同：

(1) 最优动作表达形式不同. 动作空间作为容

易获得的先验知识,如果能够充分利用,则可以减少算法运行前期选择的动作不在动作空间的情况,加快算法收敛.动作加权法充分利用了动作空间的知识,并由此改进了当前由线性逼近器直接近似动作的方法,得到了新的最优动作获得方法和策略获取方法.

(2)策略参数的更新形式不同.AW-PS-AC 算法以最小化动作均方误差为目标,将时间差分误差直接运用于策略权值的更新公式中,利用策略权值调整策略参数的更新幅度,并与资格迹相结合,获得了新的策略参数更新形式.在策略参数的更新中,充分利用了每一次行动的数据,而不会像 CACLA 算法[9]中只使用了时间差分误差大于 0 的数据进行更新.因此 AW-PS-AC 算法使得数据的利用率更高,从而加快收敛速度.

完整的 AW-PS-AC 算法如算法 1 所示.

算法 1 一种动作加权的行动者评论家算法.

输入:折扣因子 γ、状态迹-衰减参数 λ_v、动作迹-衰减参数 λ_u、动作选取标准差 σ、动作上界 $u_{\max}$、动作下界 $u_{\min}$、值函数参数更新步长 α、策略参数更新步长 β;

输出:策略参数向量 $\boldsymbol{\psi}_1$、$\boldsymbol{\psi}_2$;

初始化:值函数参数向量 $\boldsymbol{\theta}$,策略参数向量 $\boldsymbol{\psi}_1$、$\boldsymbol{\psi}_2$,资格迹向量组 $\boldsymbol{e}_v$、$\boldsymbol{e}_{u\_1}$、$\boldsymbol{e}_{u\_2}$;

1. REPEAT(对于每个情节):
2. $x \leftarrow x_0$,x_0 为初始状态;
3. $\boldsymbol{e}_v$,$\boldsymbol{e}_{u\_1}$,$\boldsymbol{e}_{u\_2}$ 清零;
4. REPEAT (对于每个时间步):
5. 获得策略权值 w_1:
$$w_1(x) \sim N(\boldsymbol{\psi}_1^{\mathrm{T}} \cdot \boldsymbol{\phi}(x), \sigma^2);$$
6. 获得策略权值 w_2:
$$w_2(x) \sim N(\boldsymbol{\psi}_2^{\mathrm{T}} \cdot \boldsymbol{\phi}(x), \sigma^2);$$
7. 计算最优动作 u:
$$u = \frac{u_{\min} \cdot w_1(x) + u_{\max} \cdot w_2(x)}{w_1(x) + w_2(x)};$$
8. IF $u > u_{\max}$
THEN $u = u_{\max}$;
9. IF $u < u_{\min}$
THEN $u = u_{\min}$;
10. 在状态 x 执行动作 u 后得到奖赏 r 和状态 x';
11. 更新值函数资格迹 $\boldsymbol{e}_v$:
$$\boldsymbol{e}_v = \gamma\lambda_v \boldsymbol{e}_v + \boldsymbol{\phi}(x);$$
12. 计算时间差分误差 δ:
$$\delta = r + \gamma\boldsymbol{\phi}^{\mathrm{T}}(x')\boldsymbol{\theta} - \boldsymbol{\phi}^{\mathrm{T}}(x)\boldsymbol{\theta};$$
13. 更新值函数参数向量 $\boldsymbol{\theta}$:
$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}_v;$$
14. 更新动作资格迹分量 $\boldsymbol{e}_{u\_1}$:
$$\boldsymbol{e}_{u\_1} = \gamma\lambda_u \boldsymbol{e}_{u\_1} + \boldsymbol{\phi}(x) \cdot \frac{w_1(x)}{w_1(x)+w_2(x)};$$
15. 更新动作资格迹分量 $\boldsymbol{e}_{u\_2}$:
$$\boldsymbol{e}_{u\_2} = \gamma\lambda_u \boldsymbol{e}_{u\_2} + \boldsymbol{\phi}(x) \cdot \frac{w_2(x)}{w_1(x)+w_2(x)};$$
16. 更新策略参数向量分量 $\boldsymbol{\psi}_1$:
$$\boldsymbol{\psi}_1 = \boldsymbol{\psi}_1 + \beta\delta\boldsymbol{e}_{u\_1};$$
17. 更新策略参数向量分量 $\boldsymbol{\psi}_2$:
$$\boldsymbol{\psi}_2 = \boldsymbol{\psi}_2 + \beta\delta\boldsymbol{e}_{u\_2};$$
18. $x = x'$;
19. UNTIL x 为终止状态.
20. UNTIL 到达最大情节数.

为了保证算法收敛,评论家中的步长参数 α_t 和行动者中的步长参数 β_t 应满足条件 $\sum_t \alpha_t = \sum_t \beta_t = \infty$,$\sum_t \alpha_t^2 < \infty$,$\sum_t \beta_t^2 < \infty$,$\lim_{t\to\infty} \frac{\beta_t}{\alpha_t} = 0$[12].然而在实际使用中若使用该条件得到的步长参数进行实验,实验的收敛速度会很慢,必须经过大量的参数调整才能获得较好的收敛速度,并且在不稳定的环境下使用渐变的步长参数会使得实验效果不理想[1].因此在实际运用中为了保持较快的收敛速度,通常会将步长参数 α,β 设定为较小的正常数,并且满足条件 β 远小于 α.

3.6 收敛性分析

文献[14]详细分析了在 3 个假设成立的情况下,使用线性函数逼近器和时间差分算法在策略学习算法中的收敛性证明.本文正是采用线性函数逼近器的策略学习方法,同时采用时间差分算法来实现值函数学习,因此,文中算法如果满足以下 3 个假设,则能以概率 1 收敛.

假设 1. 马尔可夫链是不可约的且是非周期的,立即奖赏和值函数有界.

假设 2. 基函数向量组满秩,基函数有界.

假设 3. 步长参数 α_t 是正的,非增的,已决定的(在算法执行之前就已经被选择的),并且满足公式 $\sum_{t=0}^{\infty} \alpha_t = \infty$ 和 $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

以下 3 个引理即为证明本算法满足文献中的假设.

引理 1. AW-PS-AC 算法依赖的马尔可夫链具有不可约性和非周期性,且立即奖赏值和值函数有界.

证明. 如果一个马尔可夫过程中任意两个状态可以相互转移,则该过程具有不可约性.本算法为解决马尔可夫决策问题中的强化学习问题,对于任

意状态 x 和 x'，必定存在一个转移函数 f，使得 $f(x,x')>0$. 因此状态 x 可以转移为状态 x'. 不可约性得证. 对于不可约的马尔可夫链，若某一个状态具有非周期性，则整个马尔可夫链具有非周期性. 若某状态具有自回归性，则可证明该状态具有非周期性. 因此，对马尔可夫链非周期性证明，即为对某状态自回归性的证明. 在本算法中，对于状态 x，必定存在 1 个 f 满足 $f(x,x)>0$，即证明该状态 x 具有自回归性，由此可得该算法具有非周期性.

本算法中立即奖赏有界，即存在正实数 C，对于任意状态 x，x' 和动作 u，均使得 $|\rho(x,u,x')|<C$ 成立. 根据值函数定义可得 $V(x)=E\left\{\sum_{i=0}^{\infty}\gamma^i\cdot\rho_i\right\}$，$0<\gamma<1$. 根据以上公式可得 $V(x)\leqslant\max\left(\sum_{i=0}^{\infty}\gamma^i\cdot\rho_i\right)\leqslant\sum_{i=0}^{\infty}\gamma^i C=\frac{C}{1-\gamma}$. 根据该不等式可得值函数 $V(x)$ 存在上界. 同理可以求得 $V(x)\geqslant\min\left(\sum_{i=0}^{\infty}\gamma^i\cdot\rho_i\right)\geqslant-\sum_{i=0}^{\infty}\gamma^i C=-\frac{C}{1-\gamma}$，根据该不等式可得值函数 $V(x)$ 有下界. 因此，值函数有界性得证.　　证毕.

引理 2.　AW-PS-AC 算法中基函数有界，且由 agent 访问的状态构建的基函数向量组线性无关.

证明.　本算法中值函数的计算为线性算法，基函数使用的是归一化的高斯径向基函数. 该基函数运算是以自然对数为底的指数运算，因此基函数向量所有分量均大于 0. 基函数进行了归一化操作，因此所有分量均小于等于 1. 于是基函数有界得证.

假设使用高斯径向基函数的部分状态的特征向量线性相关，且满足公式 $\boldsymbol{\alpha}=k_1\boldsymbol{\beta}_1+k_2\boldsymbol{\beta}_2+\cdots+k_n\boldsymbol{\beta}_n$. 对于状态空间中的所有点，总存在某个点，当该点作为中心点时使得该式不成立. 将该点加入中心点集合中，即可以使得原本线性相关的向量变为线性无关. 因此，按照该方法增加若干中心点后，此时就能保证访问状态构建的基函数向量组线性无关. 在实际任务中，若当前基函数向量组线性相关，则说明基函数组成的矩阵不可逆. 可以通过岭回归方法在原矩阵的对角线上加一个较小的常数. 此时构成的新的基函数向量组可逆，而且还可以确保基函数向量组的稳定性. 因此该引理得证.　　证毕.

引理 3.　AW-PS-AC 算法的步长参数 α_t 为非增正数，且满足公式 $\sum_{t=0}^{\infty}\alpha_t=\infty$ 和 $\sum_{t=0}^{\infty}\alpha_t^2<\infty$.

证明.　步长参数常用的设定方法为 $\alpha_t=1/(t+1)$，其中 t 为时间步. 很显然步长参数为正数且非增的. 牛顿幂级数展开 $\sum_{t=0}^{\infty}\alpha_t$ 可以得到式(21).

$$\sum_{t=0}^{\infty}\alpha_t=1+1/2+\cdots+1/n+\cdots=\lim_{n\to\infty}\ln(n+1)+r \tag{21}$$

其中 r 为欧拉常数，函数 $\ln n$ 为无上界的单调递增函数. 因此，$\sum_{t=0}^{\infty}\alpha_t=\infty$.

$$\sum_{t=0}^{\infty}\alpha_t^2=\lim_{t\to\infty}(1^2+(1/2)^2+\cdots+(1/t)^2)<2 \tag{22}$$

当 $n\to\infty$ 时，满足 $\sum_{t=0}^{n}\alpha_n^2<\infty$. 因此，该命题得证.　　证毕.

由以上引理可以得出该算法满足引理[14]的 3 个假设，因此 AW-PS-AC 算法收敛.　　证毕.

4　实验结果分析

4.1　实验描述

为了验证 AW-PS-AC 算法的可行性，对两个经典的连续动作强化学习问题，平衡杆问题[15]和水洼世界问题[16]进行实验. 并且与连续动作强化学习算法：CACLA 算法[9]、CAQ 算法[8]和 INAC-S[17]算法进行对比.

4.1.1　平衡杆

平衡杆问题示意图如图 2 所示. 在水平轨道上放置一辆质量为 $M_c=1$ kg 的小车. 在小车上固定一根质量为 $m_p=0.1$ kg 的杆子，杆子长度 $l=1$ m. 杆子和竖直方向成一定角度，目的是使得杆子和竖直方向的角度在 $[-\pi/4,\pi/4]$ 范围内. 为了保持杆子平衡，每 $\Delta t=0.1$ s 都需要对小车施加水平方向作用力 F，作用力 F 的范围为 $[-50\text{ N},50\text{ N}]$（右方向为正方向）. 施加力 F 的同时，水平方向还会受到随机噪声的扰动，扰动大小范围为 $[-10\text{ N},10\text{ N}]$（右方向为正方向）.

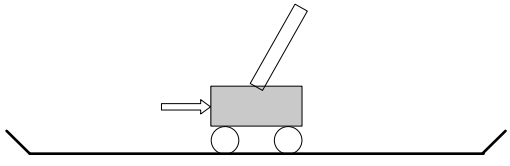

图 2　平衡杆示意图

该问题的状态由 ω 和 $\dot{\omega}$ 表示. ω 表示杆子与竖直方向的夹角. $\dot{\omega}$ 表示杆子的角速度. 杆子的角加速

度 $\ddot{\omega}$ 计算如式(23)所示.

$$\ddot{\omega}=\frac{g\sin\omega+\cos\omega\left(\frac{-F-ml\dot{\omega}^2\sin\omega}{m_p+M_c}\right)}{l\left(\frac{4}{3}-\frac{m_p\cos^2\omega}{m_p+M_c}\right)} \tag{23}$$

其中 g 为重力加速度,取为 9.811 m/s^2.

杆子角速度计算公式为 $\dot{\omega}=\dot{\omega}+\ddot{\omega}\Delta t$,角度计算公式为 $\omega=\omega+\dot{\omega}\Delta t$. 奖赏值设置为当下一个状态的角度大于 $\pi/4$ 时,奖赏值为-1;当角度小于等于 $\pi/4$ 时,奖赏值为 1. 开始状态定义为 $\omega=0,\dot{\omega}=0$. 当该状态执行动作后角度大于 $\pi/4$ 或者执行步数到达 3000 步时,该情节结束. 实验的目的是保持杆子能在 3000 步中不会倒下并且与竖直方向夹角小于 $\pi/4$.

4.1.2 水洼世界

图 3 为连续空间的水洼世界问题的示意图. 图中状态空间为边长为 1 的正方形. 图中两条线段表示水洼所在的位置. 两条线段的顶点位置分别为(0.1,0.75),(0.45,0.75)和(0.45,0.4),(0.45,0.8). 如果执行动作后到达的状态离水洼的最短距离 d 小于 0.1,则奖赏值 r 为$-1-400\times(0.1-d)$,其余情况下奖赏值 r 均为-1. 在每个状态下 agent 都可以向任意方向进行移动,每次移动的距离固定为 0.05. 每次移动的过程中在 x 轴和 y 轴方向都会受到噪声的影响,噪声的大小满足均值为 0、标准差为 0.01 的正态分布. 如果某次移动后的状态超过边界,则该状态停留在边界上. 水洼世界实验的目的是找到起点到终点的最短路径且行走的路径能够有效地绕过水洼. 本实验中起点定义为(0,0),终点(x,y)满足公式 $x+y>1.9$.

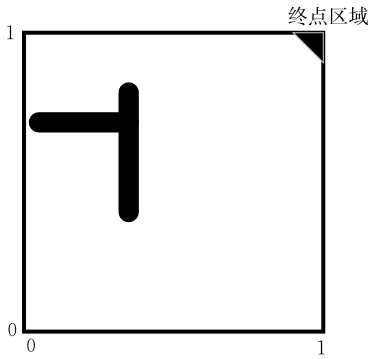

图 3 水洼世界示意图

4.2 实验结果

4.2.1 平衡杆

本实验中 4 种算法均使用高斯径向基函数来进行线性函数逼近. 在状态空间的 ω 分量上的中心点的取值为$-0.6,-0.4,-0.2,0,0.2,0.4,0.6$,该分量上宽度为 0.2. 在 $\ddot{\omega}$ 分量上中心点取值为$-2,0,2$,该分量上宽度为 2. 值函数更新中步长参数 $\alpha=0.5$. 策略更新中步长参数 $\beta=0.2$. 折扣因子 $\gamma=0.9$,值函数参数向量 $\boldsymbol{\theta}$ 初始定义为每个分量均为 0. 一个情节最大步数定义为 3000,总情节数为 3000.

在平衡杆实验中,每个情节的累计奖赏和步数成正比关系,所以可以通过比较每个情节的步数来说明不同算法在平衡杆实验中收敛效果的好坏. 图 4 为 AW-PS-AC 算法,CACLA 算法[9],CAQ[8]算法和 INAC-S[17]算法的算法性能对比图. 所有算法均实现 30 次,纵坐标为每个情节的步数平均值. AW-PS-AC 算法中 $\lambda_x=0.9,\lambda_u=0.1$,动作选取过程中方差取值为 0.4. 在 CACLA 算法的动作选取过程中方差取值为 3. INAC-S 算法动作资格迹中 $\lambda=0.2$,动作选取过程中方差取值为 3. 在 CAQ 算法中,基础总动作数为 5 个,将分别为-50N,-25N,0N,25N,50N. 资格迹中 $\lambda=0.4$,探索因子 $\varepsilon=0.01$.

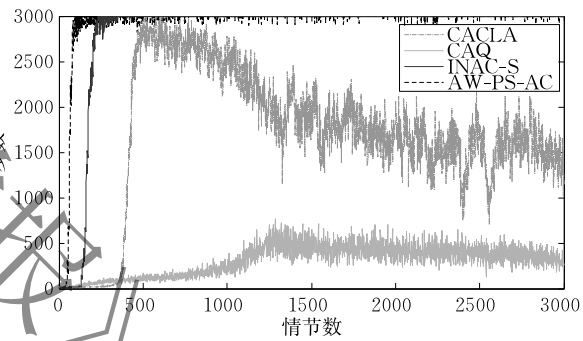

图 4 平衡杆实验中不同算法步数比较

平衡杆实验的动作空间为连续闭区间,与本算法要求的动作空间一致. 图 4 中可以看出使用 AW-PS-AC 算法得到的实验结果收敛速度最快,尤其是在情节数小于 200 时,该算法中步数随着情节数增长的速度远高于其它算法,而且收敛后能保持长期稳定,在 4 个算法中收敛效果最好. 与本算法类似的 CACLA 算法大约在 500 情节数时步数才能到达最高值,并且情节数过多后性能会缓慢的降低,无法持续维持较好的效果. INAC-S 算法总体收敛效果也很好,大约 300 情节数就能保证步数达到最高值,但是收敛速度依然略慢于 AW-PS-AC 算法,收敛后效果比较稳定. CAQ 算法中虽然也有个别情节能够到达步数为 3000 步,但总体成功率较低,同时收敛效果较差,远远达不到实验要求.

4 种算法的收敛效果比较如表 1 所示. 表中首次成功表示第 1 次情节步数到达 3000 步时的情节数. 成功率表示步数到达 3000 步的情节数与总情节

数的比值.表中所有数据均为 30 次实验的平均数据.通过表 1 可以很明显地看出 AW-PS-AC 算法的收敛效果均远远好于其它算法.综合图 4 和表 1 可看出,以平衡杆为例,AW-PS-AC 算法在动作空间是连续闭区间的情况下收敛速度和收敛后的稳定性均好于同类型的连续动作空间的强化学习算法.

表 1　平衡杆实验 4 种算法的收敛效果比较

算法名称	首次成功	成功率/%	平均步数
AW-PS-AC	66	97.17	2927.2
CACLA	416	45.09	1683.1
CAQ	1355	0.33	305.0
INAC-S	175	93.98	2825.7

AW-PS-AC 算法利用了动作空间的知识优化了当前的最优动作选择,同时增加了部分计算量和存储量.对于存储量,AW-PS-AC 算法将常用的单个策略参数向量的存储改为两个策略参数向量组的存储,更新时对两个策略参数向量组进行更新,因此,其计算和存储复杂度仍然为 $O(m)$,其中 m 为策略参数向量维数.

为了比较不同算法在时间性能上的差距,将 4 种算法进行 30 次实验,得到了平均每次实验所需总时间和总步数,并计算出每十万步所需时间.由于平衡杆实验的特殊性,不适合将每次实验的总时间作为时间性能衡量标准.于是,本实验中以每十万步所需时间为度量标准,评价不同算法中每一轮迭代所需时间的长短.以此为度量方法,对不同算法的时间性能进行比较.平衡杆实验的时间性能比较如表 2 所示.

表 2　平衡杆实验 4 种算法时间性能比较

算法名称	总时间/s	总步数	每十万步所需时间/s
AW-PS-AC	85.692	8787391	0.9752
CACLA	45.641	3958355	1.1530
CAQ	6.716	968100	0.6937
INAC-S	49.911	7207523	0.6925

通过表 2 可以看出,AW-PS-AC 算法虽然增加了计算量,然而每一轮迭代时间依然在可接受的范围内.CAQ 算法虽然每一轮迭代时间较短,但收敛速度较慢,到达收敛时需要的情节数也较多,同时该算法在选取不同的离散动作数下,存储量和迭代次数的差距也十分大,在复杂的实验环境下效果并不好.CACLA 算法在主动放弃一些不好的实验数据的前提下,每一轮迭代时间依然是 4 种算法中最长的.INAC-S 算法在 4 种算法中,每一轮迭代时间最短,并且收敛速度也较快,但总体收敛时间依然比 AW-PS-AC 算法长.

AW-PS-AC 算法在平衡杆实验中使用不同的动作累加迹的实验结果如图 5 所示.图 5 中状态资格迹的迹-衰减参数 $\lambda_x=0.9$.

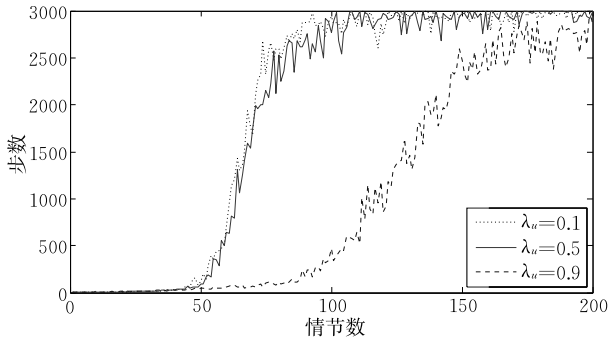

图 5　平衡杆问题不同动作资格迹下步数比较

从图 5 可看出,该实验中随着动作资格迹的迹-衰减参数减小,每个情节达到 3000 步的速度会加快,并且迹-衰减参数的值对实验的收敛速度有着一定的影响.因此,在算法中选用合适的迹-衰减参数能够大幅提高实验的收敛速度.在该实验中动作迹-衰减参数越大效果越差的原因,可能是因为在平衡杆实验中相邻两步选择的动作关联性不强,新的动作无需采用最近选择的动作做参考.

4.2.2　水洼世界

对于水洼世界实验,4 种算法均使用高斯径向基线性函数进行线性逼近.将状态的两个维度各平分 10 份,每一个顶点均为基函数的中心点.中心点宽度取值为 0.05,折扣率 $\gamma=0.95$,状态更新步长 $\alpha=0.1$.情节最大步数定义为 1000.在水洼世界实验中通常使用每个情节的累计奖赏来衡量策略的好坏.回报值越大,说明该策略越好.不同算法实验效果对比结果如图 6 和表 3 所示.图 6 和表 3 中的回报值均为每个实验执行 20 次后得到的平均回报值.

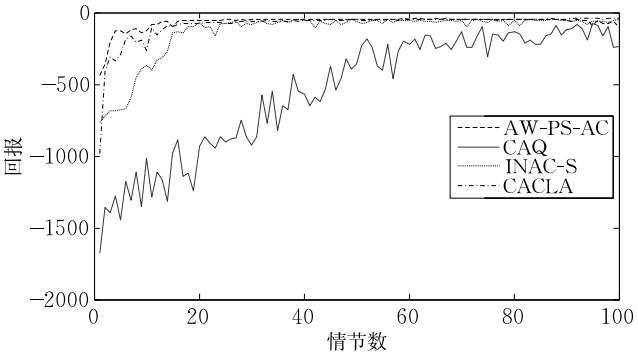

图 6　水洼世界问题不同算法下回报值比较

AW-PS-AC 算法中 $\lambda_x=\lambda_u=0.9$.策略更新中步长参数 $\beta=0.01$.动作选取过程中方差取值为

0.05. 在 CACLA 算法的策略更新中步长参数 $\beta=0.02$，动作选取过程中方差取值为 0.5. INAC-S 算法动作资格迹中 $\lambda=0.9$，动作方差取值为 0.5. 在 CAQ 算法中，基础总动作数为 5 个，将动作空间平均划分，资格迹 $\lambda=0.4$，探索因子 $\varepsilon=0.01$.

表 3 水洼世界实验不同情节数下回报值比较

算法名称	前 20 步	前 40 步	前 60 步
AW-PS-AC	−129.1	−89.47	−74.82
CACLA	−213.9	−138.80	−109.60
CAQ	−1215.0	−988.30	−786.10
INAC-S	−400.6	−238.60	−180.70

水洼世界的动作空间是整个实数空间，并且动作值每相差 2π 代表的实际动作相同，而 AW-PS-AC 算法针对的是连续闭区间，两个对应动作空间有一定差别. 可以经过简单的变换使得该水洼世界动作空间满足 AW-PS-AC 算法的要求：AW-PS-AC 算法策略权值的参数设置为最大动作 $u_{\max}=\pi$，最小动作 $u_{\min}=-\pi$. 若动作值大于 π 或小于 $-\pi$，将转化为等价的、在$[-\pi,\pi]$空间中的动作.

根据图 6 和表 3 可得，在水洼世界实验中 AW-PS-AC 算法的效果依然远优于其它同类算法. 尤其在情节数小于 10 时，AW-PS-AC 算法收敛速度很快. CACLA 算法和 INAC-S 算法在收敛速度和稳定性方面均略微差一些，但依然能够保证在较少情节以后实现收敛. CAQ 算法在 4 种算法中效果最差，大约需要 300 个情节才能使回报值基本稳定. 可见，即使在非连续闭区间情况下，AW-PS-AC 算法依然会有较好的实验效果.

表 4 呈现的是在水洼世界实验中不同算法的时间性能的比较. 根据每十万步所需时间来评价 4 种算法每次迭代直至收敛所需时间. 由于水洼世界算法的实验环境简单，模拟环境所需要的时间较少，所以水洼世界实验中每十万步所需要的时间均远少于平衡杆实验中所需要的时间.

表 4 水洼世界实验时间性能比较

算法名称	总时间/s	总步数	每十万步所需时间/s
AW-PS-AC	2.094	152798	0.1370
CACLA	2.090	170034	0.1229
CAQ	1.633	784355	0.0208
INAC-S	1.788	182262	0.0981

通过表 4 可以看出虽然 AW-PS-AC 算法在水洼世界实验中每十万步所需时间最长，但是与类似的在动作空间中使用函数逼近方法的算法相比，例如 CACLA 算法和 INAC-S 算法，时间性能差距并不大，而且结合图 6 和表 4 中的收敛效果可以看出总体收敛时间依然是 AW-PS-AC 算法最短. CAQ 算法在简单环境下每次迭代所需要的时间远远少于其它算法. 虽然 CAQ 算法收敛速度依然较慢，但也能在较短时间内学习到较优的路径. CACLA 算法忽略部分较差数据的特性在水洼世界实验中产生了优势，收敛所需时间和收敛所需情节数均比平衡杆实验中的表现好，但总体性能依然弱于 AW-PS-AC. INAC-S 算法虽然每十万步所需时间依然较短，但是优势已经没有在平衡杆实验中那么明显，并且收敛效果也比 CACLA 算法差，但总体来说，也能保持在较短时间和较少情节数下达到收敛效果.

图 7 中显示了在 AW-PS-AC 算法中动作资格迹对实验效果的影响. 从图 7 中可以看出，在该实验中迹-衰减参数越大，收敛速度越快. 该实验中动作资格迹对该算法的影响和在平衡杆实验中的影响恰好相反. 可见动作资格迹的具体取值是受到实验环境影响的. 该实验中动作资格迹对实验有正向影响的原因可能是该实验中每一步动作的选择和前几步动作的选择有所关联，使得 AW-PS-AC 算法中学到的路径和其它算法中学到的路径相比具有更强的动作连续性.

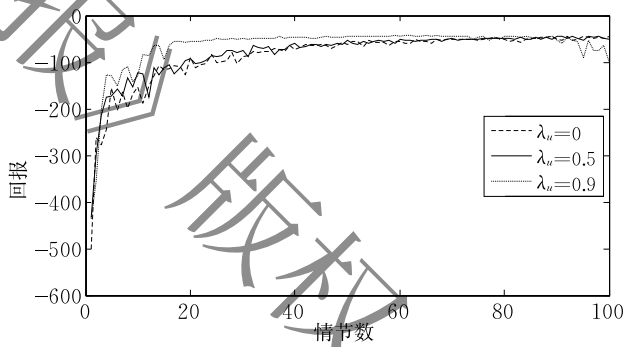

图 7 水洼世界不同动作资格迹下回报值比较

5 相关工作

在本文中 AW-PS-AC 算法的主要对比算法为 CAQ 算法，CACLA 算法和 INAC-S 算法.

CAQ 算法是一个经典的连续动作空间离散化的算法. 该算法把动作空间分为若干个离散动作，根据离散化的动作求出所有的状态动作值函数并对其进行加权，最终求出当前最优动作，并按照权值更新当前资格迹，从而更新当前所有的状态动作值函数. 该算法中离散的动作数依赖于个人经验，空间存储

量很大，而且在某些环境下效果会很差. 但是 CAQ 算法具有计算量少和计算简单的优点，适用于一些简单的环境.

CACLA 算法，INAC-S 算法和 AW-PS-AC 算法思路相似，都属于行动者评论家算法，在动作空间中均使用函数逼近方法近似最优动作，使用线性方法计算最优动作. 然而 3 个算法也都有各自的特点.

AW-PS-AC 算法和 CACLA 算法相似，都通过梯度下降法来更新策略参数. 不同的是，CACLA 算法利用动作差值进行更新，同时放弃了所有时间差分误差小于 0 的数据. 这样会使得策略参数不会向更差的方向更新，但同时会浪费大量的实验数据. 而 AW-PS-AC 算法直接利用时间差分误差更新策略参数，使得所有数据都能得到充分利用.

AW-PS-AC 算法和 INAC-S 算法均使用资格迹优化策略参数向量的更新，从而提高收敛速度. 不同的是，AW-PS-AC 算法是利用常规的梯度下降方法更新策略参数，而 INAC-S 算法则是求出自然梯度，将自然梯度运用在梯度下降中更新策略参数.

AW-PS-AC 算法与常用的连续空间强化学习算法最大的区别是，常用的连续空间算法不考虑动作计算的优化，使用常规的线性方法计算当前最优动作. 而 AW-PS-AC 算法利用了动作空间的知识，改进了当前的最优动作的计算方法，并且通过实验可以看出该逼近方法有效且收敛效果好.

6　结　论

为了解决传统的强化学习方法在连续动作空间学习最优策略时效率低的问题，以行动者评论家方法为基础，使用动作加权方法计算最优动作，使用高斯分布直接获取策略，以最小化策略函数均方误差为目的，使用梯度下降方法得到策略权值的更新公式，使用资格迹和时间差分误差优化权值更新公式，提出了动作加权行动者评论家算法. 并将动作加权行动者评论家算法与 CACLA 算法、CAQ 算法和 INAC-S 算法 3 种经典连续动作空间算法进行比较. 从实验结果可以看出，AW-PS-AC 算法的收敛速度快，收敛后稳定性好，适用于解决各种类型的问题. 该算法使用较少的存储量，最终获得的动作有较好的连续性，满足实际需求.

以后的工作主要分为两个方面：(1) 本文中使用梯度下降法更新策略权值. 梯度下降法计算量小，但同时需要的样本数量依然较多，样本利用率不高. 如何在样本数有限的情况下较快地实现收敛效果，这是以后可以研究的问题；(2) 动作加权的权值可以从 2 个增加到多个. 如何在策略权值个数发生变化后依然保证又快又好的收敛效果，也将作为下一步研究的方向.

参　考　文　献

[1] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge Massachusetts, UK: MIT Press, 1998

[2] Busoniu L, Babuska R, De Schutter B, et al. Reinforcement Learning and Dynamic Programming Using Function Approximators. Boca Raton, Florida, USA: CRC Press, 2010

[3] Lee D, Seo H, Jung M W. Neural basis of reinforcement learning and decision making. Annual Review of Neuroscience, 2012, 35(5): 287-308

[4] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529(7587): 484-489

[5] Sutton R S, Mcallester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems, 2000, 12: 1057-1063

[6] Peters J, Schaal S. Natural actor-critic. Neurocomputing, 2008, 71(7-9): 1180-1190

[7] Peters J, Vijayakumar S, Schaal S. Reinforcement learning for humanoid robotics. Autonomous Robot, 2003, 12(1): 1-20

[8] Millán J D R, Posenato D, Dedieu E. Continuous-action Q-learning. Machine Learning, 2002, 49(2-3): 247-265

[9] van Hasselt H, Wiering M A. Reinforcement learning in continuous action spaces//Proceedings of the Approximate Dynamic Programming and Reinforcement Learning. Honolulu, USA, 2007: 272-279

[10] Martin H J A, De Lope J. Ex(α): An effective algorithm for continuous actions Reinforcement Learning problems//Proceedings of the the 35th IEEE Annual Conference on Industrial Electronics Society. Oporto, Portugal, 2009: 2063-2068

[11] Wiering M, van Otterlo M. Reinforcement Learning: State-of-the-Art. Berlin Heidelberg: Springer, 2014

[12] Bhatnagar S, Sutton R S, Ghavamzadeh M, et al. Incremental natural actor-critic algorithms//Proceedings of the Conference on Neural Information Processing Systems. Vancouver, Canada, 2007: 2471-2482

[13] Konda V R, Tsitsiklis J N. Actor-critic algorithms. SIAM Journal on Control & Optimization, 2001, 42(4): 1008-1014

[14] Tsitsiklis J N, van Roy B. An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control, 1997, 42(5): 674-690

[15] Berenji H R, Khedkar P. Learning and tuning fuzzy logic controllers through reinforcements. IEEE Transactions on Neural Networks, 1992, 3(5): 724-740

[16] Sutton R S. Generalization in reinforcement learning: Successful examples using sparse coarse coding//Proceedings of the Neural Information Processing Systems. Denver, USA, 1995: 1038-1044

[17] Degris T, Pilarski P M, Sutton R S. Model-free reinforcement learning with continuous action in practice. American Control Conference, 2012, 50(6): 2177-2182

LIU Quan, born in 1969, Ph. D., professor, Ph. D. supervisor. His main research interests include intelligence information processing, automated reasoning and machine learning.

ZHANG Peng, born in 1992, M. S. candidate. His main research interest is continuous space reinforcement learning.

ZHONG Shan, born in 1983, Ph. D. candidate, lecturer. Her research interests include machine learning and deep learning.

QIAN Wei-Sheng, born in 1992, M. S. candidate. His main research interests include partially observable Markov decisions processes.

ZHAI Jian-Wei, born in 1992, M. S. candidate. His main research interests include deep learning and deep reinforcement learning.

Background

Reinforcement learning is a class of learning methods that try to obtain the mappings from states to actions, via maximizing the cumulative numerical rewards. In allusion to the problem of poor convergence in the traditional reinforcement learning methods, this paper proposed a new algorithm, namely, Action Weight Policy Search Actor-Critic algorithm (AW-PS-AC). AW-PS-AC uses Actor-Critic as the framework and approximates the optimal value function, and then utilizes the knowledge of the action space to improve the gradient descent in conventional algorithms. AW-PS-AC solves the problem with continuous state and action spaces. Simulation evaluation illustrates that AW-PS-AC behaves effectively, with rapid converge rate and satisfactory stability.

This paper is supported by the National Natural Science Foundation of China (61472262, 61502323, 61502329), the Natural Science Foundation of Jiangsu (BK2012616), the High School Natural Foundation of Jiangsu (13KJB520020), the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (93K172014K04), and the Suzhou Industrial Application of Basic Research Program Part (SYG201422, SYG201308).