doi:10.11959/j.issn.1000-436x.2017089

增量式双自然策略梯度的行动者评论家算法

章鹏[1]，刘全[1,2,3]，钟珊[1]，翟建伟[1]，钱炜晟[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，江苏 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，吉林 长春 130012）

摘 要：针对强化学习中已有连续动作空间算法未能充分考虑最优动作的选取方法和利用动作空间的知识，提出一种对自然梯度进行改进的行动者评论家算法。该算法采用最大化期望回报作为目标函数，对动作区间上界和下界进行加权来求最优动作，然后通过线性函数逼近器来近似动作区间上下界的权值，将最优动作求解转换为对双策略参数向量的求解。为了加快上下界的参数向量学习速率，设计了增量的 Fisher 信息矩阵和动作上下界权值的资格迹，并定义了双策略梯度的增量式自然行动者评论家算法。为了证明该算法的有效性，将该算法与其他连续动作空间的经典强化学习算法在 3 个强化学习的经典测试实验中进行比较。实验结果表明，所提算法具有收敛速度快和收敛稳定性好的优点。

关键词：强化学习；自然梯度；行动者评论家；连续空间

中图分类号：TP181 **文献标识码**：A

Actor-critic algorithm with incremental dual natural policy gradient

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

(1. School of Computer Science and Technology, Soochow University, Suzhou 215006, China;
2. Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210000, China;
3. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China)

Abstract: The existed algorithms for continuous action space failed to consider the way of selecting optimal action and utilizing the knowledge of the action space, so an efficient actor-critic algorithm was proposed by improving the natural gradient. The objective of the proposed algorithm was to maximize the expected return. Upper and the lower bounds of the action range were weighted to obtain the optimal action. The two bounds were approximated by linear function. Afterward, the problem of obtaining the optimal action was transferred to the learning of double policy parameter vectors. To speed the learning, the incremental Fisher information matrix and the eligibilities of both bounds were designed. At three reinforcement learning problems, compared with other representative methods with continuous action space, the simulation results show that the proposed algorithm has the advantages of rapid convergence rate and high convergence stability.

Key words: reinforcement learning, natural gradient, actor-critic, continuous space

1 引言

强化学习（RL，reinforcement learning）又称为增强学习或激励学习，是从状态到动作的映射学习，其目的是通过最大化累计奖赏来实现状态到动作的映射[1,2]。强化学习采用马尔可夫决策过程

收稿日期：2016-11-04；**修回日期**：2017-03-03

基金项目：国家自然科学基金资助项目(No.61272005, No.61303108, No.61373094, No.61472262, No.61502323, No.61502329)；江苏省自然科学基金资助项目（No.BK2012616）；江苏省高校自然科学研究基金资助项目（No.13KJB520020）；吉林大学符号计算与知识工程教育部重点实验室基金资助项目（No.93K172014K04）；苏州市应用基础研究计划工业部分基金资助项目（No.SYG201422, No.SYG201308）

Foundation Items: The National Natural Science Foundation of China (No.61272005, No.61303108, No.61373094, No.61472262, No.61502323, No.61502329), The Natural Science Foundation of Jiangsu Province (No.BK2012616), The High School Natural Foundation of Jiangsu Province (No.13KJB520020), The Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (No.93K172014K04), Suzhou Industrial Application of Basic Research Program (No.SYG201422, No.SYG201308)

(MDP, Markov decision processes)作为基本框架。当一个强化学习问题满足马尔可夫决策过程时，通常可以使用动态规划（DP，dynamic programming）、蒙特卡洛（MC，Monte Carlo）、时间差分（TD，temporal-difference）等方法进行求解。

在离散的状态和动作空间中，通常使用查询表的方式来存储状态值函数或状态—动作值函数。该存储方式简单有效，并且能够快速获取最优策略，但是该方法只适用于一些简单的任务。在大规模空间中，为了获得精确的策略，需要对大量的值函数进行存储和计算，因此在大规模空间中使用查询表的方法不符合实际。为了解决这些问题，可以将传统的强化学习方法和函数逼近方法相结合，使学习到的经验泛化到整个状态和动作空间中。该方法可以在不需要大量计算量和存储量的情况下，依然能够获得有效的策略和较为准确的值函数[3,4]。

连续动作空间的强化学习研究在近年来开始增多，逐渐成为强化学习领域的热点。Sutton 等[5]首次将强化学习动作表示方法和策略梯度方法相结合，提出了强化学习策略梯度函数逼近方法。Peter 等[6,7]将自然梯度方法运用在函数逼近中，并且结合了强化学习的时间差分最小二乘算法，提出自然梯度行动者评论家算法（NAC, natural actor-critic）。Hasselt 等[8]利用动作差值更新策略参数，使用时间差分误差评价执行动作的好坏，提出了连续空间的行动者评论家自动学习机（CACLA，continuous actor-critic learning automaton）算法。Wierstra 等[9,10]将自然梯度方法和进化策略方法同时运用在策略更新中，提出了自然进化策略（NES，natural evolutionary strategies）方法。Busoniu 等[2,11,12]利用交叉熵优化基函数的位置和形状，提出了交叉熵优化（cross-entropy optimization）方法。Martin 等[13]利用 k 最近邻分类方法，对空间进行离散化，提出了基于 k 最近邻分类的时间差分算法。Lillicrap 等[14]在深度强化学习中使用策略梯度进行学习，提出了深度确定性策略梯度算法。Gu 等[15]在连续空间中使用模型学习方法来提高收敛速度，提出了基于模型加速的连续空间深度 Q 学习算法。Khamassi 等[16]使用元学习方法运用在连续空间行动者评论家的探索参数更新当中，提出了基于元学习的行动者评论家强化学习方法。

目前，已有的连续动作空间算法很少对最优动作值的获取方式进行研究，也未能充分利用动作空间本身的知识。最优动作的选取方式对策略的获取和算法的收敛效果都会产生很大的影响。动作空间与值函数空间通常不相同，值函数空间在学习之前通常是未知的，而动作空间大多数情况下是已知的。在最优策略的学习当中，动作空间的知识是可以被利用的。而当前连续空间的强化学习算法中将动作空间的约束作为学习知识使用得不多。当今连续空间强化学习中的策略参数更新大多使用随机梯度方法。使用该方法的数据利用率不高，总体收敛速度偏低。

为了解决以上问题，本文提出增量式双自然策略梯度的行动者评论家（IDNPG-AC，actor-critic algorithm with incremental dual natural policy gradient）算法。IDNPG-AC 算法使用增量自然梯度方法更新策略参数向量，利用动作加权将动作空间的知识运用在最优动作的选择中，使用自然梯度方法更新策略参数，再通过策略资格迹来加快策略参数的收敛速度。最后以 3 个经典的连续空间强化学习问题：Pole Balancing 问题、Mountain Car 问题和 Puddle World 问题作为基准实验，将 IDNPG-AC 算法与经典的连续空间算法：CACLA 算法、INAC 算法和 INAC-S 算法进行比较。实验结果表明 IDNPG-AC 算法在不同的环境下均有较快的收敛速度和稳定的收敛效果。

2　背景知识

2.1　马尔可夫决策过程

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[1,2,4]。马尔可夫决策过程问题可以用四元组 $M=<X,U,\rho,f>$ 表示。X 表示状态集合，x_t 表示在第 t 时间步时 agent 所处的状态。U 表示动作集合，u_t 表示第 t 时间步时 agent 采取的动作。$\rho: X\times U\times X\to\mathbb{R}$ 表示奖赏函数。$r(x_t,u_t,x_{t+1})$ 表示 agent 在状态 x_t 时执行动作 u_t 后转移到 x_{t+1} 得到的立即奖赏。$f: X\times U\times X\to(0,1)$ 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式为

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

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。

状态值函数 $V^h(x)$表示状态根据策略 h 获得的累计奖赏。连续空间在策略 h 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，$r=\rho(x,u,x')$。

若存在某一策略 h^*，对于任意状态 x 和策略 h，使 $V^{h^*}(x)\geqslant V^h(x)$ 恒成立，则此时值函数被称为最优 V 值函数，h^* 被称为最优策略。最优 V 值函数 V^* 的表示为

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

2.2 行动者评论家方法

行动者评论家方法是一种用单独存储结构来明确表示、策略独立于值函数的时间差分方法。行动者和评论家 2 个部分组成了行动者评论家的结构。行动者用于选择动作，评论家对动作的好坏进行评价。行动者不是依据当前的值函数选择动作，而是依据存储的策略选择动作。评论家的评论一般采用时间差分误差的形式。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学习[17,18]。行动者评论家算法结构如图 1 所示。

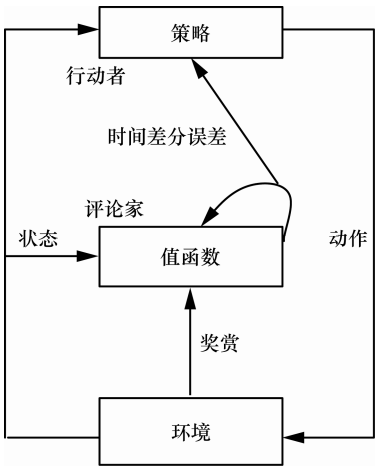


图 1 行动者评论家方法结构

传统的行动者评论家方法主要用于离散状态和动作空间中。行动者根据查询表来选择当前状态下应执行的动作。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。

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

其中，$p(x,u)$ 表示状态 x 动作 u 的偏好度。根据偏好度可以计算出当前状态下每个动作选择的概率。一种常见的调整偏好度方法如式(5)所示。

$$p(x_t,u_t)\leftarrow p(x_t,u_t)+\beta\delta_t \quad (5)$$

其中，β 表示偏好度步长参数，$0<\beta<1$。

时间差分误差是用来评价动作选择好坏的一个重要指标。评论家会根据当前的值函数计算出时间差分误差值。时间差分误差值的计算如式(6)所示。

$$\delta_t=r_{t+1}+\gamma V(x_{t+1})-V(x_t) \quad (6)$$

行动者评论家方法可以分离值函数的计算和策略的获取，并可同时学习值函数和最优策略。和普通的值函数方法相比，行动者评论家方法选择动作时的计算量较少，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策略有较大幅度的改变。

3 值函数和策略的表示和更新方法

3.1 最优动作表示方法

在连续动作空间中常用线性方法表示当前状态的最优动作值。近似最优动作的选择方法为

$$\mu(x)=\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{\phi}(x)=\sum_{i=1}^{n}\psi_i\phi_i(x) \quad (7)$$

其中，$\boldsymbol{\psi}$ 表示策略参数向量，$\boldsymbol{\phi}(x)$ 表示状态 x 的特征向量，n 表示特征维数。

使用线性方法逼近最优动作具有计算量小和存储量少等优点。但是该方法逼近效果不理想，收敛速度较慢，需要较多的情节才能得到较优的效果。通常情况下，动作空间是已知的，因此，动作空间的约束也可以作为学习的知识使用。定义动作空间为 $[u_{\min},u_{\max}]$，状态 x 对应的最优动作可以通过对 2 个策略权值($\boldsymbol{w}_1(x)$ 和 $\boldsymbol{w}_2(x)$)来表示，即

$$\mu(x)=\boldsymbol{w}_1(x)u_{\min}+\boldsymbol{w}_2(x)u_{\max} \quad (8)$$

为了在动作空间中满足状态 x 的最优动作 $\mu(x)$，

且每一个最优动作 $\mu(x)$ 都只有一组向量组 $w_i(x)$ 相对应，策略权值必须满足条件 $0 \leqslant w_1(x) \leqslant 1$, $0 \leqslant w_2(x) \leqslant 1$ 和 $w_1(x)+w_2(x)=1$。可以使用线性计算方法对策略权值进行求解。

$$\begin{cases} w_1(x)=\boldsymbol{\psi}_1^{\mathrm{T}}\boldsymbol{\phi}(x) \\ w_2(x)=\boldsymbol{\psi}_2^{\mathrm{T}}\boldsymbol{\phi}(x) \end{cases} \tag{9}$$

其中，$\boldsymbol{\psi}_1$、$\boldsymbol{\psi}_2$ 表示策略权值向量组。$\boldsymbol{\phi}(x)$ 表示状态 x 的特征向量。

为了保证向量组 $w_i(x)$ 满足条件 $w_1(x)+w_2(x)=1$，需要对线性方法求得的策略权值组进行归一化计算，如式（10）所示。

$$\begin{cases} w_1(x)=\dfrac{\boldsymbol{\psi}_1^{\mathrm{T}}\boldsymbol{\phi}(x)}{\boldsymbol{\psi}_1^{\mathrm{T}}\boldsymbol{\phi}(x)+\boldsymbol{\psi}_2^{\mathrm{T}}\boldsymbol{\phi}(x)} \\ w_2(x)=\dfrac{\boldsymbol{\psi}_2^{\mathrm{T}}\boldsymbol{\phi}(x)}{\boldsymbol{\psi}_1^{\mathrm{T}}\boldsymbol{\phi}(x)+\boldsymbol{\psi}_2^{\mathrm{T}}\boldsymbol{\phi}(x)} \end{cases} \tag{10}$$

使用动作加权方法求得的最优动作不会出现在动作空间以外的情况，能更有效地利用了交互获得的信息，提高了收敛的速度。尤其是当实验交互数据很少的时候，使用动作权值表示的收敛效果远好于其他算法。

3.2　值函数参数更新方法

线性方法是常用的值函数近似的方法。当算法满足在策略时，使用线性方法求得的解将收敛于使用 *TD*(0)算法的解。用线性函数逼近的值函数求解方法如式(11)所示。

$$\widehat{V}(x,\boldsymbol{\theta})=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(x)=\sum_{i=1}^{n}\theta_i\phi_i(x) \tag{11}$$

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，n 表示特征维数。

值均方误差（MSVE，mean squared value error）是常见的评价值函数参数向量好坏的方式。值均方误差的表示方法如下所示。

$$MSVE(\boldsymbol{\theta})=\sum_{x}d(x)\left[V^h(x)-\widehat{V}(x,\boldsymbol{\theta})\right]^2 \tag{12}$$

其中，$d(x)$ 表示状态 x 的分布权重，满足条件 $0 \leqslant d(x) \leqslant 1$ 且 $\int_{x\in X} d(x)\mathrm{d}x=1$。

值均方误差表示近似值函数与真实值函数的差距。因此，值均方误差可以作为值函数参数向量的评价标准。值均方误差越小，说明值函数参数越为准确。强化学习中常用梯度下降法来优化值函数参数。在线学习的线性方法表示中，值函数状态参数向量的更新如式(13)所示。

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\alpha\delta_t\boldsymbol{\phi}(x_t) \tag{13}$$

其中，α 表示值函数步长参数，$0<\alpha<1$。

使用资格迹方法可以进一步提高值函数参数向量的收敛速度。资格迹通常可以当作一步时间差分算法和蒙特卡罗算法之间的桥梁，使 agent 能够向前看到所有的奖赏并把奖赏更好地进行结合。常用的资格迹包括累加迹、替代迹和荷兰迹 3 种。在连续空间中通常使用累加迹作为资格迹。连续空间下累加迹的表示方法如式(14)所示。

$$\boldsymbol{e}_{t+1}=\gamma\lambda\boldsymbol{e}_t+\varphi(x_t) \tag{14}$$

其中，λ 为迹衰减(trace-decay)参数，用来表示 agent 对未来奖赏值的重视程度，满足条件 $0 \leqslant \lambda < 1$，$\varphi(x)=\dfrac{\partial V(x)}{\partial\theta}$。

使用资格迹后的值函数参数向量的更新如式(15)所示。

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\alpha\delta_t\boldsymbol{e}_{t+1} \tag{15}$$

3.3　策略参数更新方法

在连续动作空间中对策略的评价方法很多，本文通过状态值函数来评价当前策略的好坏。在强化学习中，当新的策略优于原策略时，新策略对应的所有状态的值函数都不会比原来策略对应的值函数小。对于最优策略，所有状态的值函数均取到最大值。因此，在确定的环境中，值函数越大说明当前策略越好。通过最大化值函数期望的方法获得当前环境下的最优策略。

连续动作空间下的状态值函数表示为

$$V^h(x)=\int P(T\mid x,\boldsymbol{\psi})E\left\{\sum_{t=0}^{\infty}\gamma^t r_{t+1}\mid T\right\}\mathrm{d}T \tag{16}$$

其中，T 表示轨迹序列，即一个情节中所有状态动作对组成的序列，$T=\{x_0,u_0,x_1,u_1,\cdots\}$。$P(T\mid x,\boldsymbol{\psi})$ 表示在策略参数 $\boldsymbol{\psi}$ 下，以 x 为起始状态的轨迹 T 出现的概率。

梯度上升法是机器学习中常见的函数最大化的参数更新方法。因此，可以通过梯度上升方法优化策略向量使值函数最大化。使用随机梯度上升法的策略参数更新为

$$\boldsymbol{\psi}_{t+1}=\boldsymbol{\psi}_t+\beta\nabla_{\boldsymbol{\psi}}V^h(x_t) \tag{17}$$

根据式(16)，可以推出$\nabla_{\boldsymbol{\psi}}V^h(x)$的表示方法为

$$\begin{aligned}\nabla_{\boldsymbol{\psi}}V^h(x)&=\int\nabla_{\boldsymbol{\psi}}P(T|x,\boldsymbol{\psi})\mathrm{E}\left\{\sum_{t=0}^{\infty}\gamma^t r_{t+1}|T\right\}\mathrm{d}T\\&=\int P(T|x,\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}}\ln P(T|x,\boldsymbol{\psi})\mathrm{E}\left\{\sum_{t=0}^{\infty}\gamma^t r_{t+1}|T\right\}\mathrm{d}T\\&=\mathrm{E}\left\{\nabla_{\boldsymbol{\psi}}\ln P(T|x,\boldsymbol{\psi})\mathrm{E}\left\{\sum_{t=0}^{\infty}\gamma^t r_{t+1}|T\right\}|x,T\right\}\end{aligned} \tag{18}$$

式(18)将对值函数的梯度求解转化为对轨迹概率的梯度求解。$\nabla_{\boldsymbol{\psi}}\ln P(T|x,\boldsymbol{\psi})$的求解如式(19)所示。

$$\begin{aligned}&\nabla_{\boldsymbol{\psi}}\ln P(T|x,\boldsymbol{\psi})\\&=\nabla_{\boldsymbol{\psi}}\left(\ln P(x_0=x)+\sum_{t=0}^{\infty}\ln h(x_t,u_t,\boldsymbol{\psi})+\sum_{t=0}^{\infty}P(x_t,u_t,x_{t+1})\right)\\&=\sum_{t=0}^{\infty}\nabla_{\boldsymbol{\psi}}\ln h(x_t,u_t,\boldsymbol{\psi})\end{aligned} \tag{19}$$

其中，$P(x_0=x)$表示x为起始状态时的概率，$P(x_t,u_t,x_{t+1})$表示x_t状态选择动作u_t时到达状态x_{t+1}的概率，$h(x_t,u_t,\boldsymbol{\psi})$表示在策略参数$\boldsymbol{\psi}$下，状态$x_t$选择动作$u_t$的概率。

$P(x_0=x)$和$P(x_t,u_t,x_{t+1})$与环境有关，与策略参数$\boldsymbol{\psi}$无关，所以$P(x_0=x)$和$P(x_t,u_t,x_{t+1})$对$\boldsymbol{\psi}$的偏导数均为0。因此$\nabla_{\boldsymbol{\psi}}V^h(x)$如式(20)所示。

$$\nabla_{\boldsymbol{\psi}}V^h(x_t)=\mathrm{E}\left\{R(x_t)\left(\sum_{j=t}^{\infty}\nabla_{\boldsymbol{\psi}}\ln h(x_j,u_j,\boldsymbol{\psi})\right)\right\} \tag{20}$$

其中，$R(x_t)$表示以x_t为起始状态的折扣累积奖赏。

在行动者评论家方法中，评论家通过时间差分误差值来评价当前选择动作的好坏，并优化当前策略。因此，可以将时间差分误差值运用在策略参数的更新中，使行动者和评论家之间可以通过时间差分进行交流。结合上述方法，策略参数的更新式为

$$\boldsymbol{\psi}_{t+1}=\boldsymbol{\psi}_t+\beta\delta_t\nabla_{\boldsymbol{\psi}}\ln h(x_t,u_t,\boldsymbol{\psi}_t) \tag{21}$$

式(21)为最大化值函数的策略向量更新。然而，该方法是在参数空间中进行梯度更新操作的。在策略参数空间中使用传统的随机梯度法，每次更新会使策略参数有较小的变化。然而，策略参数的变化不同于策略的变化。在一定条件下，策略参数较小的变化可能会导致策略产生极大的改变，此时，学习后的策略可能会变得很差。在强化学习中策略是一种概率分布。2 个分布之间概率属性距离可以通过 Kullback-Leibler 距离来表示。在按照 Kullback-Leibler 距离度量下使用梯度方法决定的梯度被称为自然梯度。自然梯度和随机梯度相比有更快的收敛速度和更加稳定的性能。Kullback-Leibler 距离度量下的梯度与普通欧式空间下梯度的关系如式(22)所示。

$$\tilde{\nabla}J(\boldsymbol{\psi})=\boldsymbol{G}^{-1}(\boldsymbol{\psi})\nabla J(\boldsymbol{\psi}) \tag{22}$$

其中，$\boldsymbol{G}$表示为 Fisher 信息矩阵。

在策略更新中的 Fisher 信息矩阵的表示方法为

$$\boldsymbol{G}(\boldsymbol{\psi})=\mathop{\mathrm{E}}_{x\in X,u\in U}\left[\nabla_{\boldsymbol{\psi}}\ln h(x,u,\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}}^{\mathrm{T}}\ln h(x,u,\boldsymbol{\psi})\right] \tag{23}$$

然而，直接求解$\boldsymbol{G}(\boldsymbol{\psi})$需要大量的计算，并且在大部分的强化学习问题中，环境信息几乎无法事先获得，难以直接求解$\boldsymbol{G}(\boldsymbol{\psi})$。这些原因导致准确的 Fisher 信息矩阵很难直接获得。由于本算法是在线学习算法，因此可以将在线获得的数据直接运用在 Fisher 信息矩阵的更新中，这样可以大幅减少计算量。增量方法的 Fisher 信息矩阵的更新式为

$$\boldsymbol{G}_{t+1}=(1-\zeta)\boldsymbol{G}_t+\zeta\nabla_{\boldsymbol{\psi}}\ln h(x_t,u_t,\boldsymbol{\psi}_t)\nabla_{\boldsymbol{\psi}}^{\mathrm{T}}\ln h(x_t,u_t,\boldsymbol{\psi}_t) \tag{24}$$

其中，ζ表示 Fisher 信息矩阵更新步长，满足$0\leqslant\zeta\leqslant1$。

为了提高策略参数的更新效率，可以类似于状态资格迹的方法，将策略资格迹e_u运用在策略参数的更新中。策略资格迹可以将在线学习中实际经历过的动作作为知识，并用于策略参数更新中。策略资格迹的更新方法为

$$e_u=\gamma\lambda_u e_u+\nabla_{\boldsymbol{\psi}}\ln h(x,u,\boldsymbol{\psi}) \tag{25}$$

其中，λ_u表示策略迹衰减参数，$0<\lambda_u<1$。

以上方法均是用单策略参数来表示策略时的策略更新方法。在 3.2 节提出的最优动作表示方法下，策略参数为 2 个。可以将动作加权方法中的策略权值分别用在策略向量组的更新中。使 2 个策略向量分别在不同的方向上进行更新。最终策略参数的更新方法为

$$\begin{cases}\boldsymbol{\psi}_1=\boldsymbol{\psi}_1+\beta\delta\boldsymbol{G}_1^{-1}\boldsymbol{e}_{u\_1}\\\boldsymbol{\psi}_2=\boldsymbol{\psi}_2+\beta\delta\boldsymbol{G}_2^{-1}\boldsymbol{e}_{u\_2}\end{cases} \tag{26}$$

策略资格迹组的更新方法为

$$\begin{cases} \boldsymbol{e}_{u\_1} = \gamma\lambda\boldsymbol{e}_{u\_1} + w_1(x)\nabla_{\psi_1}\ln h(x,u,\boldsymbol{\psi}) \\ \boldsymbol{e}_{u\_2} = \gamma\lambda\boldsymbol{e}_{u\_2} + w_2(x)\nabla_{\psi_2}\ln h(x,u,\boldsymbol{\psi}) \end{cases} \tag{27}$$

3.4　算法描述

完整的增量式双自然策略梯度的行动者评论家算法如算法 1 所示。

算法 1　增量式双自然策略梯度的行动者评论家算法

1) 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 $\boldsymbol{e}_x$、$\boldsymbol{e}_{u\_1}$、$\boldsymbol{e}_{u\_2}$，Fisher 信息矩阵 $\boldsymbol{G}_1$、$\boldsymbol{G}_2$；

2) Repeat　(对于每个情节)

3) $x \leftarrow x_0$, x_0 为初始状态；

4) 向量 $\boldsymbol{e}_x$、$\boldsymbol{e}_{u\_1}$、$\boldsymbol{e}_{u\_2}$ 清零，矩阵 $\boldsymbol{G}_1$、$\boldsymbol{G}_2$ 清零；

5) Repeat　(对于每个时间步)

6) w_1：　$w_1 = \boldsymbol{\psi}_1^{-1}\boldsymbol{\phi}(x)$；

7) w_2：　$w_2 = \boldsymbol{\psi}_2^{-1}\boldsymbol{\phi}(x)$；

8) $\mu(x) = \dfrac{u_{\min}w_1(x) + u_{\max}w_2(x)}{w_1(x) + w_2(x)}$；

9) 根据最优动作 $\mu(x)$ 得到策略 h；

10) 根据策略 h 得到状态 x 下选择的动作 u；

11) 在状态 x 执行动作 u，得到奖赏 r 和状态 x'；

12) $\boldsymbol{e}_x = \gamma\lambda_x\boldsymbol{e}_x + \boldsymbol{\phi}(x)$；

13) $\delta = r + \gamma\boldsymbol{\phi}(x')^{-1}\boldsymbol{\theta} - \boldsymbol{\phi}(x)^{-1}\boldsymbol{\theta}$；

14) $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}_x$；

15) 更新 Fisher 信息矩阵组

$$\boldsymbol{G}_1 = (1-\zeta)\boldsymbol{G}_1 + \zeta\nabla_{\psi_1}\mathrm{lb}h(x,u,\boldsymbol{\psi})\nabla_{\psi_1}^{\mathrm{T}}\ln h(x,u,\boldsymbol{\psi})$$

$$\boldsymbol{G}_2 = (1-\zeta)\boldsymbol{G}_2 + \zeta\nabla_{\psi_2}\mathrm{lb}h(x,u,\boldsymbol{\psi})\nabla_{\psi_2}^{\mathrm{T}}\ln h(x,u,\boldsymbol{\psi})$$

16) 更新策略资格迹分量 $\boldsymbol{e}_u$

$$\boldsymbol{e}_{u\_1} = \gamma\lambda_u\boldsymbol{e}_{u\_1} + \frac{w_1(x)}{w_1(x)+w_2(x)}\nabla_{\psi_1}\ln h(x,u,\boldsymbol{\psi})$$

$$\boldsymbol{e}_{u\_2} = \gamma\lambda_u\boldsymbol{e}_{u\_2} + \frac{w_2(x)}{w_1(x)+w_2(x)}\nabla_{\psi_2}\ln h(x,u,\boldsymbol{\psi})$$

17) 更新策略参数向量分量 $\boldsymbol{\psi}$

$$\boldsymbol{\psi}_1 = \boldsymbol{\psi}_1 + \beta\delta\boldsymbol{G}_1^{-1}\boldsymbol{e}_{u\_1}$$

$$\boldsymbol{\psi}_2 = \boldsymbol{\psi}_2 + \beta\delta\boldsymbol{G}_2^{-1}\boldsymbol{e}_{u\_2}$$

18) $x = x'$；

19) Until　x 为终止状态

20) Until　到达最大情节数

连续动作空间中，高斯分布是连续空间中的一种常用动作探索方法。该方法将最优动作值作为均值，将探索的宽度值作为标准差。使用高斯分布时状态 x 执行动作 u 的概率为

$$h(x,u) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(u-\mu(x))^2}{2\sigma^2}\right) \tag{28}$$

其中，σ 表示标准差。

根据式(28)可得，用高斯分布表示策略时，最优动作周围的动作有较大概率被选到，并且远离最优动作的动作依然有较低的概率被选择，满足探索的要求。策略的探索幅度受到标准差 σ 的影响。当最优动作 $\mu(x)$ 使用普通线性方法进行表示时，即 $\mu(x) = \boldsymbol{\psi}^{-1}\boldsymbol{\phi}(x)$ 时，$\nabla_{\psi}\ln h(x,u,\boldsymbol{\psi})$ 的计算方法如下所示。

$$\nabla_{\psi}\ln h(x,u,\boldsymbol{\psi}) = \frac{1}{\sigma^2}(u-\mu(x))\boldsymbol{\phi}(x) \tag{29}$$

使用动作加权法求最优动作 $\mu(x)$ 时，可以得到 $\nabla_{\psi_1}\ln h(x,u,\boldsymbol{\psi}) = \nabla_{\psi_2}\ln h(x,u,\boldsymbol{\psi})$。此时，在每一轮迭代中求得的 2 个 Fisher 信息矩阵 $\boldsymbol{G}_1^{-1}$、$\boldsymbol{G}_1^{-2}$ 均相同，则可以将 2 个矩阵合并为一个矩阵进行更新，从而减少了计算量、存储量和运算时间。

4　实验结果分析

为了说明 IDNPG-AC 算法的实验效果，本文将 IDNPG-AC 算法与 CACLA 算法、INAC 算法和 INAC-S 算法进行对比。实验环境分别为 Pole Balancing[19]、Mountain Car[20]和 Puddle World[21]。

4.1　Pole Balancing 实验

Pole Balancing 示意如图 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，长度为 l =1 m 的杆子。杆子和竖直方向的角度为 ω，实验目的是使 ω 的范围为 $\left[-\dfrac{\pi}{8},\dfrac{\pi}{8}\right]$。为了保持杆子平衡，每 Δt =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 N,50 N]（右方向为正方向）。施加作用力 F 的同时，水平方向还会受到大小为 [−10N,10N] 的噪声扰动（右方向为正方向）。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(30)所示。

$$\ddot{\omega}=\frac{\mathrm{g}\sin\omega+\cos\omega\left(\dfrac{-F-ml\dot{\omega}^2\sin\omega}{m_\mathrm{p}+M_\mathrm{c}}\right)}{l\left(\dfrac{4}{3}-\dfrac{m_\mathrm{p}\cos^2\omega}{m_\mathrm{p}+M_\mathrm{c}}\right)} \tag{30}$$

其中，g 表示重力加速度，为 9.811 m/s^2。

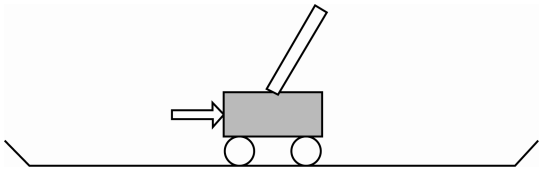


图 2 Pole Balancing 示意

杆子角速度计算式为 $\dot{\omega}=\dot{\omega}+\ddot{\omega}\Delta t$，角度计算式为 $\omega=\omega+\dot{\omega}\Delta t$。奖赏值设置恒为 1。初始状态为 ω=0，$\dot{\omega}$ =0。当该状态执行动作后状态的角度大于 $\frac{\pi}{8}$ 或执行步数到达 3 000 步时，该情节结束。实验的目的是保持杆子能在 3 000 步中能与竖直方向夹角小于 $\frac{\pi}{8}$。

在实验中均使用高斯径向基函数作为逼近函数，在状态空间 ω 分量上的中心点取值为−0.6、−0.4、−0.2、0、0.2、0.4、0.6，该分量上宽度为 0.2。在 $\dot{\omega}$ 分量上中心点取值为−2、0、2，该分量上宽度为 2。以求得的最优动作作为均值，以固定值为方差的高斯分布来表示策略。折扣因子 γ 为 0.9。在值函数更新中，步长参数为 $\alpha=0.9$，策略更新中步长参数为 $\beta=0.2$。在 IDNPG-AC 算法中值函数资格迹为 0.9，策略资格迹为 0.1，策略选择中方差取值为 0.3，动作值上界取值为 50，动作值下界取值为−50。CACLA 中方差取值为 2.5。INAC 和 INAC-S 的方差取值为 2。

表 1 中首次成功表示第一次情节步数到达 3 000 步时的情节数。成功率表示步数到达 3 000 步的情节数与总情节数的比值。首次成功的情节数越小，说明收敛速度越快，成功率越高，平均步数越多，说明实验效果越稳定。图 3 中横坐标表示执行的情节数，纵坐标表示该情节数下小车直到情节完成时的总步数。在某个情节中，时间步数越多，说明小车能够平衡的时间越长，即该算法的效果越好。图 3 和表 1 中的所有数据均为 5 次实验的平均结果。Pole Balancing 实验中通常将杆子和竖直方向的角度设置为 $[-\frac{\pi}{4},\frac{\pi}{4}]$。当设置为 $[-\frac{\pi}{8},\frac{\pi}{8}]$ 时，会使杆子更难以达到实验条件。而从图 3 中可以看出，由于杆子环境设置使杆子平衡更困难，所有算法均需要 200 个以上情节才能保持杆子基本平衡。从收敛结果来看，除了 CACLA 算法外，其他算法经过一定数量的情节数后均能使杆子保持平衡。CACLA 算法在如此苛刻的条件下已经很难获得稳定的效果。INAC-S 和 INAC 算法虽然能最终能够达到最优效果，但是可以发现 2 个算法的图上均有很明显的“平台”，即在较长的情节数中出现平均情节步数没有太大变化的情况。这是由于这 2 个算法在 5 次实验中达到收敛时所需情节数的差别很大。因为图 3 中的情节步数为 5 次实验步数的平均值，所以当某个情节下部分实验已经达到最优（即情节步数为 3 000），而其他几次实验中实验效果依然不理想的情况下，对多次实验的步数直接求平均，就会出现“平台”的情况。“平台”现象说明这 2 种算法虽然最终能够得到最优解，但是算法受样本的影响较大，由于探索等原因偶尔选择到较差样本时，收敛所需要的情节数会大大增多，总体收敛效果不稳定。IDNPG-AC 算法中就几乎看不到“平台”的情况，这说明该算法收敛效果很稳定，不会由于偶尔较差的样本导致收敛情节数的突变，这与使用了动作加权方法有关。并且从图 3 和表 1 中可以很明显看出，IDNPG-AC 算法在收敛速度上和收敛后的稳定性方面均远好于其他算法，说明在 Pole Balancing 实验中，IDNPG-AC 算法性能是高效的。

表 1 Pole Balancing 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
IDNPG-AC	245.4	85.33%	2 601.5
CACLA	809.8	12.38%	444.5
INAC	643.0	63.68%	1 939.5
INAC-S	510.2	70.10%	2 129.7

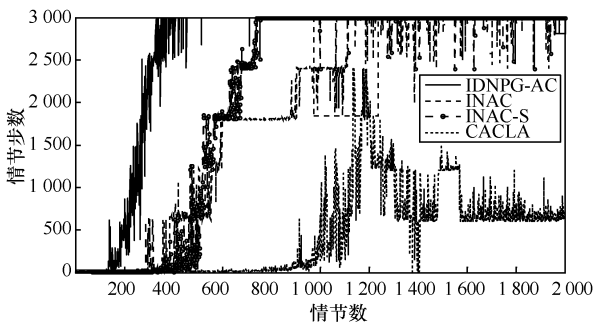


图 3 Pole Balancing 实验中不同算法实验效果比较

4.2　Mountain Car 实验

Mountain Car 实验环境如图 4 所示。一辆小车行驶在山路上。由于动力不足无法直接开到终点，小车只能通过先爬到相反方向的斜坡上，然后加足油门，才能够通过惯性到达终点。实验目标是小车能在尽可能少的时间步中到达终点。Mountain Car 的物理模型如图 4 所示。

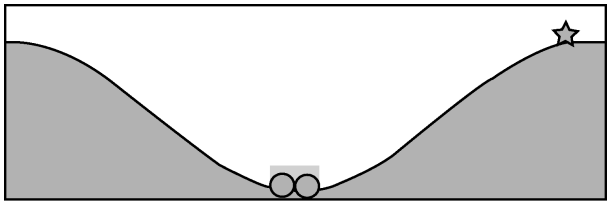


图 4　Mountain Car 的物理模型

$$\begin{cases} p_{t+1} = bound\left[p_t + \dot{p}_{t+1}\right] \\ \dot{p}_{t+1} = bound\left[\dot{p}_t + 0.001A_t - 0.0025\cos\left(3p_t\right)\right] \end{cases} \tag{31}$$

其中，p_t 表示第 t 次迭代时小车的水平位置，$\dot{p}_t$ 表示第 t 次迭代时小车的速度。A_t 表示第 t 次迭代时小车的加速度，即 agent 选择的动作。在连续动作空间中，动作 A_t 的选择范围为 $[-1,1]$。*bound* 操作会强制控制 p 和 $\dot{p}$ 满足 $-1.2 \leqslant p \leqslant 0.5$，$-0.07 \leqslant \dot{p} \leqslant 0.07$。当小车到达最左边界时，其速度强制定义为 0。小车到达最右边界时，定义小车到达最终目标。情节的起点是 $p=-0.5$、$\dot{p}=0$。

在实验中每个情节的最大步数为 3 000，均使用高斯径向基函数作为值函数逼近函数和策略逼近函数。将状态空间的 p 分量平均分为 15 份，将 $\dot{p}$ 分量平均分为 10 份作为基函数的中心点坐标。折扣因子 γ 设置为 0.95。值函数更新中步长参数 $\alpha=0.9$，策略更新中步长参数 $\beta=0.2$。使用以最优动作值为均值，固定值为方差的高斯正态分布表示策略。IDNPG-AC 算法中方差为 0.3，状态资格迹 λ_x 为 0.9，策略资格迹 λ_u 为 0.2，动作值上界取 1，动作值下界取−1。CACLA 中方差取值为 4。INAC 和 INAC-S 中方差取值为 1。INAC 算法中策略资格迹 λ_u 取 0.2。图 5 中的总步数均为对应情节数下 10 次实验的平均步数。

Mountain Car 在大多数状态下的最优动作值为 −1 和 1，并且经常出现动作突变的情况，即存在 2 个状态的欧氏距离很近，然而动作选择差别较大情况。所以在相同函数逼近器下能否更好更快地学习到动作边界值即为各个算法在该实验中性能好坏的关键。图 5 中横坐标表示情节数，纵坐标表示在该情节下小车从起点到达终点所需要的总步数。当步数越少时，表示小车到达终点越快，也说明该算法的效果越好。从图 5 中可以看出，IDNPG-AC 从收敛速度和收敛稳定性上均好于其他算法，经过多次实验，达到收敛时绝大多数都能保持在时间步在 110 左右，并且在 20 个情节下就能找到较好的策略，可以很快的达终点。在 Pole Balancing 实验中实验效果并不理想的 CACLA 算法，在 Mountain Car 实验中却有较好的表现。由于 CACLA 算法在一些时间步中放弃了对策略参数更新，导致了总体收敛速度较慢，但是达到收敛时的小车到达终点的步数较小，依然能够达到实验要求。INAC 算法和 INAC-S 算法的性能相似，虽然能够很快达到收敛，但是收敛后步数的振动幅度较大。从实验数据中发现，INAC 算法和 INAC-S 算法达到收敛时情节步数最少约为 110，最多约为 300，10 次实验的平均总步数的震动幅度较大，这说明了 INAC 算法和 INAC-S 算法对边界动作值的学习都不太理想。

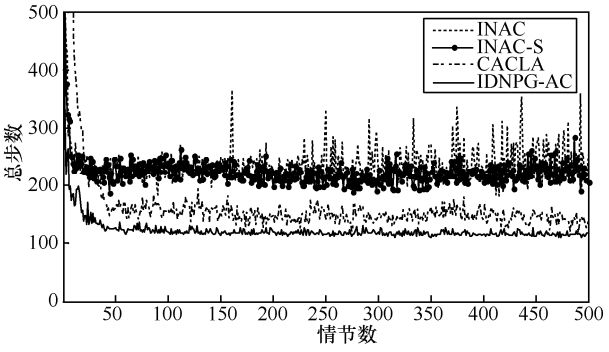


图 5　Mountain Car 实验中不同算法实验效果比较

图 6 为 IDNPG-AC 算法在 Mountain Car 实验中不同情节数下使用 IDNPG-AC 算法求得的最优动作的分布。图 6 中横坐标表示小车所在的水平位置，纵坐标表示小车的速度。对于某一状态，颜色越深代表动作值越接近−1，即说明此时策略中小车向左的加速度越大。颜色越接近白色，代表加速度越接近 1，即说明此时策略中小车向右的加速度越大。为了更加准确地计算出小车行驶的策略，实验环境中每个情节的起始状态定义为满足实验条件的任意状态。使用更改后的起始状态可以使较差状态有更多的可能性被访问到，但是同时会使一个情节下的平均步数减少，学习的信息量减少。

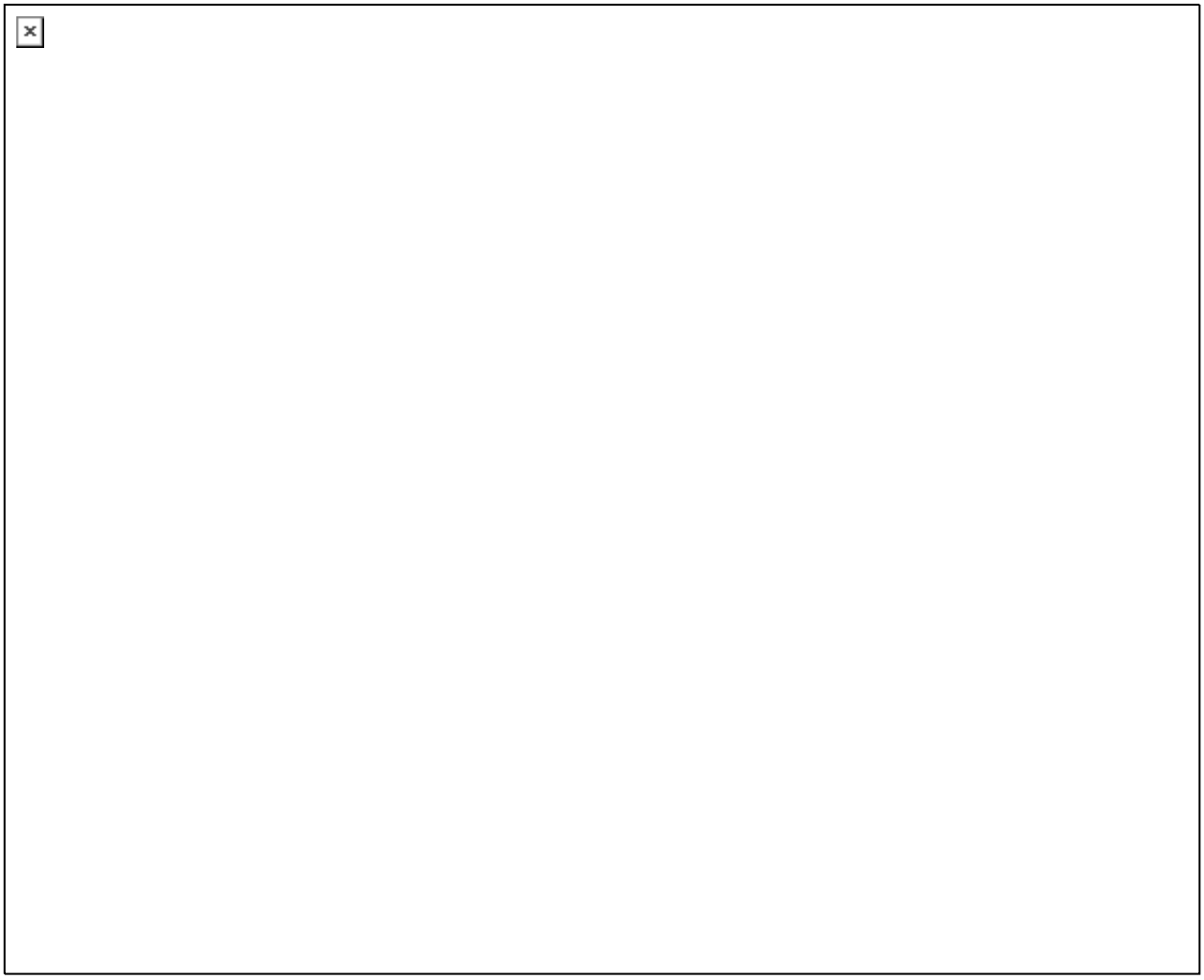


图 6　Mountain Car 实验中不同情节数下 IDNPG-AC 算法的策略分布

从图 6 中情节数为 50 的策略分布图可以看出，当情节数较少时就已经能够学会较为优秀的策略，即绝大多数状态下，动作值设为 1，即最右加速度最大，在水平位置−0.5 左右，小车在某些速度下选择动作值为−1，即小车在最低点附近，速度在一定条件下，小车选择向左加速度最大。本问题中动作空间为连续空间，然而从图 6 中可看出小车在绝大多数状态中动作值均选择−1 或 1，并且学习速度很快，这说明 IDNPG-AC 算法在最优动作多为临界值的环境下能获得较快的收敛速度和较好的实验效果。经过情节数的增加，小车的动作取值也越来越精准。当水平位置在−0.5 左右处的动作值选择也越来越接近−1，这说明小车基本学习到的策略为：当小车在最低点处周围时，加速度多选择为−1，其余加速度多选择为 1。

在 Mountain Car 实验中使用 IDNPG-AC 算法在 1 000 情节数下部分状态的动作选择如表 2 所示。表 2 中每一行中小车的位置相同，每一列中小车的速度相同，格子中的数值表示小车在该位置和速度下选择的加速度大小。从表 2 可以看出，小车速度−0.042 时动作值多选择−1，即小车在最低点位置周围多选择向左最大加速度。其余位置下多选择向右最大加速度。结合表 2 和图 6 可以看出，当小车速度较大且水平位置大于−0.2 时，IDNPG-AC 算法学到的动作值为 0.5 左右。这是由于当小车速度足够大且小车在上坡的路径中时，加速度对小车的速度和小车到终点总时间的影响较小，在较小的加速度下就能保证小车能够平稳的到达终点。

4.3　Puddle World 实验

Puddle World 实验的状态空间为边长为 1 的正方形，如图 7 所示。在正方形的状态空间中存在一部分 Puddle 作为障碍物，阻挡 agent 的前进。本次实验中 Puddle 定义为 2 条线段，2 条线段的顶点位置分别为 (0.1,0.75)，(0.45,0.75) 和 (0.45,0.4)，

表 2　IDNPG-AC 算法下 Mountain Car 实验部分状态的最优动作表示

小车水平位置	小车速度为−0.07 时的加速度	小车速度为−0.042 时的加速度	小车速度为−0.014 时的加速度	小车速度为 0.014 时的加速度	小车速度为 0.042 时的加速度
−1.20	1	1	1	1	1
−1.03	−1	1	1	1	1
−0.86	1	−1	−1	1	1
−0.69	1	−1	1	1	1
−0.52	1	−1	−1	1	1
−0.35	1	−1	−1	1	1
−0.18	1	−1	−1	1	1
−0.01	1	−1	−1	1	1
0.16	1	−1	−1	−1	1
0.33	1	1	1	1	0.56

(0.45,0.8)。如果执行动作后到达的状态离 Puddle 的最短距离 d 小于 0.1，则奖赏值 r 定义为 $-1-400(0.1-d)$，其余情况下奖赏值 r 均为−1。由于 agent 到达 Puddle 时会带来极大的负奖赏，于是 agent 必须试图绕过 Puddle 到达最终目标。实验的目标是以最大化奖赏为目标的情况下使 agent 能够较快地到达目的地。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。本实验中起点定义为 $(0,0)$，终点为 (x,y) 满足 $x+y>1.9$。

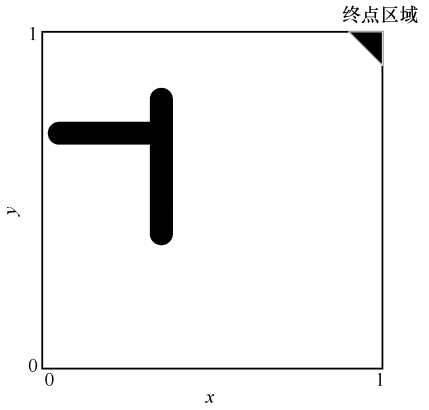


图 7　Puddle World 示意

对于 Puddle World 实验，4 种算法均使用高斯径向基线性函数进行线性逼近。基函数中心点定义为 $(0.1x,0.1y)$，其中，$x\in\{0,1,\cdots,10\}$，$y\in\{0,1,\cdots,10\}$，总共 121 个中心点。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 1 000。在 Puddle World 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。在该实验中动作空间并不是有限空间，在 IDNPG-AC 算法中，将动作下界定义为 $-\pi$，动作上界定义为 π。不同算法实验效果对比结果如图 8 所示。图 8 中的平均回报值为 20 次实验的平均值。

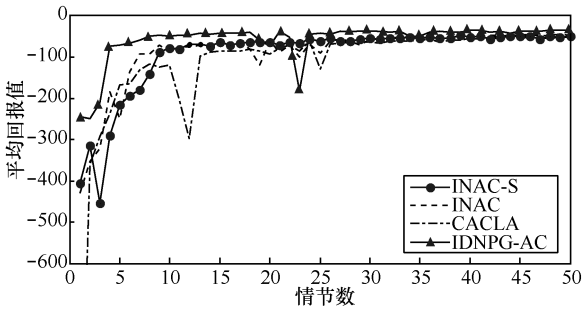


图 8　Puddle World 实验中不同算法性能对比

Puddle World 的实验环境较为简单，所有算法在 15 个情节之前就能使平均回报大于−150 且平均回报能基本保持稳定。Puddle World 实验与上述 2 个实验不同。Puddle World 每一时间步的奖赏值是可变的。从图 8 中可以看出，IDNPG-AC 算法在可变奖赏值的环境下依然有较好的实验效果，在第 4 个情节时就能使平均回报值大于−100，在 20 次实验中均能找到近似最优解。而且在实验前期，即情节数较少的时候，IDNPG-AC 算法的平均回报值远远大于其他算法的平均回报值，这表明在 Puddle World 实验中 IDNPG-AC 算法的收敛速度远快于其他算法。虽然 IDNPG-AC 算法在收敛后存在一些波

动，但是总体来说比较稳定，并且当所有算法的平均回报达到稳定时，IDNPG-AC 算法的平均回报值也比其他算法的平均回报值大 10 左右，说明 IDNPG-AC 算法在 Puddle World 环境中收敛速度快，平均回报值大，可以求得比其他算法更好的策略。

图 9 是 Puddle World 实验中在较少中心点时的算法性能。图 8 和图 9 除了中心点的位置和数量不同，其余参数均相同。图 9 中基函数中心点满足定义 $(0.2x,0.2y)$，其中，$x\in\{0,1,\cdots,5\}$，$y\in\{0,1,\cdots,5\}$，总共 36 个中心点。在新的实验中，中心点大幅减少，使要学习的参数的分量个数大幅减少，数据存储量和数据计算量同时大幅降低，但这会使函数逼近器难以对复杂环境进行泛化，逼近效果会受到一定的影响。

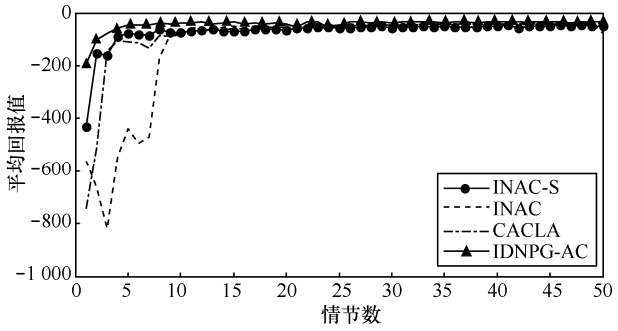


图 9 Puddle World 实验在较少中心点时不同算法性能对比

结合图 8 和图 9 可以看出 INAC-S 算法在不同中心点的实验中性能差距不明显。而 INAC 算法和 CACLA 算法在新实验初期的平均回报远小于原实验的平均回报值，这与中心点个数大幅度减少有关，这说明 INAC 算法和 CACLA 算法本身受到基函数的影响较大。当基函数本身的定义较差时，算法性能可能会大幅降低。而在情节数较多时，INAC 算法和 CACLA 算法在新实验的平均回报值略高于原先实验，这可能与在情节数较少时 INAC 算法和 CACLA 算法学习的信息量较多有关，同时也和具体实验环境和中心点的数量和位置有一定关系。IDNPG-AC 算法中新实验的平均回报值略大于旧实验，这可以说明 IDNPG-AC 算法受到函数逼近器本身的影响并不大，在不同的基函数下 IDNPG-AC 算法均能有着较好的实验效果。并且从图 9 可以看出，IDNPG-AC 算法的收敛速度和收敛稳定性均好于其他算法，IDNPG-AC 算法的平均回报值也高于其他算法。从以上几点可以说明 IDNPG-AC 算法在不同基函数环境下都会能保持着较好的实验效果。

5 结束语

为了解决连续动作空间下求解最优策略效率较低的问题，提出了增量式双自然策略梯度的行动者评论家算法。该算法以行动者评论家方法作为框架，在策略空间中以最大化值函数为目标，使用增量自然梯度方法更新策略参数。并且使用了动作资格迹，将轨迹中经历过的动作选择作为学习信息运用在当前策略参数的更新中，加快了收敛速度。使用动作加权的方法表示当前策略下的最优动作，提高算法前期的收敛速度和算法收敛时的稳定性。将该算法与 CACLA 算法、INAC 算法和 INAC-S 算法在 3 个经典的连续空间的强化学习问题进行比较。从比较结果可以看出，IDNPG-AC 算法收敛速度快，收敛稳定性好，在一些较差的样本时不会导致性能发生较大变化，能够较好地满足实验要求。

本文主要解决的是在线算法下的连续动作空间问题，即可以通过实时交互获得的数据进行学习的。在离线状态下获得大量的状态、动作、奖赏值、下一个状态组成的四元组时，如果继续使用在线方法学习，会在优化参数的计算中耗费大量的时间，导致整体学习效率低下。并且大多数在线学习算法着重对当前获得的数据进行学习，而对离线获得的数据进行学习则不适合使用该方法。如何在连续空间中快速、高效地对现有的大量的数据进行批量处理，在较短的时间内获得较为准确的值函数和策略，是值得继续研究的课题。

参考文献：

[1] SUTTON R S, BARTO A G. Reinforcement learning: an introduction[M]. Cambridge Massachusetts: MIT press, 1998.

[2] BUSONIU L, BABUSKA R, SCHUTTER B D, et al. Reinforcement learning and dynamic programming using function approximators[M]. Florida: CRC Press, 2010.

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

[4] WIERING M, VAN O M. Reinforcement learning: STATE-OF-THE-Art[M]. Berlin Heidelberg: Springer. 2014.

[5] SUTTON R S, MCALLESTER D A, SINGH S P, et al. Policy gradient methods for reinforcement learning with function approximation[C]//NIPS. 1999, 99: 1057-1063.

[6] PETERS J, SCHAAL S. NATURAL A C [J]. Neurocomputing, 2008, 71(7-9):1180-1190.

[7] PETERS J, VIJAYAKUMAR S, SCHAAL S. Reinforcement learning for humanoid robotics[J]. Autonomous Robot, 2003, 12(1):1-20.

[8] VAN H H. Reinforcement learning in continuous state and action spaces[M]//Reinforcement Learning. Springer Berlin Heidelberg, 2012: 207-251.

[9] WIERSTRA D, SCHAUL T, PETERS J, et al. Natural evolution strategies[C]//2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). 2008: 3381-3387.

[10] SUN Y, WIERSTRA D, SCHAUL T, et al. Efficient natural evolution strategies[C]//The 11th Annual Conference on Genetic and Evolutionary Computation. 2009: 539-546.

[11] RUBINSTEIN R Y, KROESE D P. The cross-entropy method[J]. Information Science & Statistics, 2008, 50(1):92-92.

[12] BOTEV Z I, KROESE D P, RUBINSTEIN R Y, et al. The cross-entropy method for optimization[J]. Machine Learning: Theory and Applications, Chennai: Elsevier BV, 2013, 31: 35-59.

[13] MARTIN H J A, DE LOPE J. x<α>: an effective algorithm for continuous actions reinforcement learning problems[C]//The IEEE Industrial Electronics Society. 2009:2063-2068.

[14] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2015, 8(6):A187.

[15] GU S, LILLICRAP T, SUTSKEVER I, et al. Continuous deep Q-learning with model-based acceleration[J]. arXiv Preprint arXiv: 1603.00748, 2016.

[16] KHAMASSI M, TZAFESTAS C. Active exploration in parameterized reinforcement learning[J]. arXiv preprint arXiv:1610.01986, 2016.

[17] BHATNAGAR S, GHAVAMZADEH M, LEE M, et al. Incremental natural actor-critic algorithms[C]//Advances in neural information processing systems. 2007: 105-112.

[18] VIJAY R, KONDA, JOHN N. Tsitsiklis. actor-critic algorithms[J]. Siam Journal on Control & Optimization, 2001, 42(4):1008-1014.

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

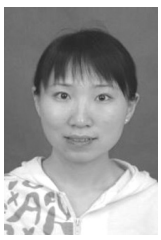
[20] SINGH S P, SUTTON R S. Reinforcement learning with replacing eligibility traces[J]. Machine Learning, 1996, 22(1-3):123-158.

[21] SUTTON R S. Generalization in reinforcement learning: successful examples using sparse coarse coding[J]. Neural Information Processing Systems, 1996:1038-1044.

作者简介：

章鹏（1992-），男，江苏仪征人，苏州大学硕士生，主要研究方向为连续空间强化学习。

刘全（1969-），男，内蒙古牙克石人，苏州大学教授、博士生导师，主要研究方向为强化学习、智能信息处理和自动推理。



钟珊（1983-），女，湖南双峰人，苏州大学博士生，主要研究方向为机器学习和深度学习。

翟建伟（1992-），男，江苏盐城人，苏州大学硕士生，主要研究方向为深度学习和深度强化学习。

钱炜晟（1992-），男，江苏常熟人，苏州大学硕士生，主要研究方向为部分可观察马氏问题的近似规划方法。