一种采用模型学习和经验回放加速的正则化自然行动器评判器算法

钟 珊[1),2),3),4)] 刘 全[1),3),5)] 傅启明[3),4),6)] 龚声蓉[1),2)] 董虎胜[1)]

[1)](苏州大学计算机科学与技术学院 江苏 苏州 215006)

[2)](常熟理工学院计算机科学与工程学院 江苏 常熟 215500)

[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)

[4)](苏州科技大学江苏省建筑智慧节能重点实验室 江苏 苏州 215006)

[5)](软件新技术与产业化协同创新中心 南京 210000)

[6)](苏州科技大学电子与信息工程学院 江苏 苏州 215006)

摘 要 行动器评判器(Actor Critic,简称 AC)算法是强化学习连续动作领域的一类重要算法,其采用独立的结构表示策略,但更新策略时需要大量样本导致样本效率不高.为了解决该问题,提出了基于模型学习和经验回放加速的正则化自然 AC 算法(Regularized Natural AC with Model Learning and Experience Replay,简称 RNAC-ML-ER).RNAC-ML-ER 将 Agent 与环境在线交互产生的样本用于学习系统动态性对应的线性模型和填充经验回放存储器.将线性模型产生的模拟样本和经验回放存储器中存储的样本作为在线样本的补充,实现值函数、优势函数和策略的更新.为了提高更新的效率,在每个时间步,仅当模型的预测误差未超过阈值时才利用该模型进行规划,同时根据 TD-error 从大到小的顺序对经验回放存储器中的样本进行回放.为了降低策略梯度估计的方差,引入优势函数参数向量对优势函数进行线性近似,在优势函数的目标函数中加入 ℓ_2-范数进行正则化,并通过优势函数参数向量来对策略梯度更新,以促进优势函数和策略的收敛.在指定的两个假设成立的条件下,通过理论分析证明了所提算法 RNAC-ML-ER 的收敛性.在 4 个强化学习的经典问题即平衡杆、小车上山、倒立摆和体操机器人中对 RNAC-ML-ER 算法进行实验,结果表明所提算法能在大幅提高样本效率和学习速率的同时保持较高的稳定性.

关键词 行动器评判器算法;模型学习;经验回放;最优策略;正则化;自然梯度

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2019.00532

A Regularized Natural AC Algorithm with the Acceleration of Model Learning and Experience Replay

ZHONG Shan[1),2),3),4)] LIU Quan[1),3),5)] FU Qi-Ming[3),4),6)] GONG Sheng-Rong[1),2)] DONG Hu-Sheng[1)]

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

[2)](*School of Computer Science and Engineering, Changshu Institute of Technology, Changshu, Jiangsu* 215500)

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

[4)](*Jiangsu Province Key Laboratory of Intelligent Building Energy Efficiency, Suzhou University of Science and Technology, Suzhou, Jiangsu* 215006)

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

[6)](*College of Electronic & Information Engineering, Suzhou University of Science and Technology, Suzhou, Jiangsu* 215006)

Abstract Actor Critic (AC) algorithm serves as an important method for solving problems with continuous action space in reinforcement learning (RL), where the actor corresponds the policy

收稿日期:2016-12-02;在线出版日期:2017-12-29.本课题得到国家自然科学基金项目(61772355,61702055,61303108,61373094,61472262,61502323,61502329)、江苏省自然科学基金(BK2012616)、江苏省高校自然科学研究项目(13KJB520020)、江苏省高校自然科学研究面上项目(16KJD520001)、江苏省科技计划项目(BK2015260)、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(93K172014K04,93K172017K18)、苏州市应用基础研究计划工业部分(SYG201422,SYG201308)资助.**钟 珊**,女,1983 年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为强化学习和深度学习.E-mail:sunshine-620@163.com.**刘 全**(通信作者),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处理、自动推理和机器学习.E-mail:quanliu@suda.edu.cn.**傅启明**,男,1985 年生,博士,讲师,主要研究方向为强化学习和贝叶斯推理.**龚声蓉**,男,1966 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为机器学习和图像处理.**董虎胜**,男,1981 年生,博士研究生,讲师,主要研究方向为机器学习和图像处理.

and the critic refers to the value function. However, this separate representation structure of the policy results in that enormous samples are required to achieve the convergence for policy. To address this problem, a regularized natural AC algorithm with model learning and experience replay, called RNAC-ML-ER, is proposed, where the value function, the advantageous function and the policy are updated in on-line learning, planning and experience replaying so that the optimal policy can be found as soon as possible. The linear model with respect to the system dynamics is learned and the memory of experience replay is filled in on-line learning, via the samples collected from the interaction between Agent and environment. After the linear model is learned, it can be used to generate amount of simulated samples. The actual samples generated during learning, the samples stored in the memory as well as the simulated samples corporate together so as to update the value function, the advantageous function and the policy further. In order to improve the updating efficiency, the prediction error the model is computed at each time step, but it is used for planning only when the prediction error does not exceed the threshold. Furthermore, the samples in the memory are replayed according to their TD-errors. To reduce the variance of the estimated gradient and accelerate the convergence of the policy, two tricks are employed here. One is that the advantageous function is also linearly approximated, where the ℓ_2-regularization as the smooth method is introduced to the goal function of optimizing process, and the other is that the policy gradient is update by using the learned parameters of the advantageous function. Theoretically, RNAC-ML-ER is analyzed from two aspects such as the time and space complexities analysis and the convergence analysis. The time and the space complexities are $O(SW(T^E+T^M+1)d)$ and $O(M)$, where S, W, T^E, T^M, d and M represent the number of episodes, the maximal steps in every episode, the updating times of the samples in the memory, the planning times, the dimension of the parameters and the capacity of the memory, respectively. The convergence of RNAC-ML-ER is analyzed by proving three theorems under the predefined two assumptions. RNAC-ML-ER is implemented on four typical benchmarks such as the pole balancing problem, the mountain car problem, the inverted pendulum problem and the acrobat problem. RNAC-ML-ER is not only compared in performance with the discrete and continuous methods, but also compared with the non-linear deep network models. The performance mainly concerns in the convergence rate, the sample efficiency and the stability. The experimental results show that RNAC-ML-ER has the best performance compared the other methods nearly in all operated experiments. It also demonstrates that the application of the model outperforms the method without using model in sample efficiency. Therefore, one of the next work will be that introducing the linear model to the deep-network-based approximation to speed the learning of the value function and the policy.

Keywords　actor critic algorithm; model learning; experience replay; optimal policy; regularization; natural gradient

1　引　言

自然界的序贯决策问题可以被建模为马尔可夫决策过程(Markov Decision Processes,简称 MDPs),并通过动态规划或线性规划来迭代求解[1].然而,当模型即状态迁移函数和奖赏函数未知时,MDPs 无法通过这类传统的规划方法进行直接求解[2-3].强化学习(Reinforcement Learning,简称 RL)是一种基于样本学习的方法,可以在模型未知的情况下,通过与控制器交互获得样本,实现在线或离线学习,从而求解出 MDPs 的最优值函数和最优策略.

近年来,RL 已经成为人工智能和机器学习的一个关键领域[4-7],其大致可以分为两类算法:精确 RL 算法和近似 RL 算法.精确 RL 算法采用表格存储值函数和策略,仅适合小规模离散状态和动作空

间问题的求解.随着状态和动作空间规模的增大,采用这类方法求解会面临无法表征的问题甚至导致"维数灾".近似 RL 方法采用线性或非线性函数逼近器来近似表示值函数和策略,这类方法较传统的精确学习方法具有更好的泛化能力,因此可以满足连续状态动作空间的需要.

当状态空间连续而动作空间离散时,可采用解决离散动作空间问题的近似 RL 方法来寻求最优值函数[8],然后在最优值函数上采用贪心方法来求最优策略.由于值函数是近似表示的,因此仍然无法保证策略得到改进.此外,值函数的较小误差将会引起策略的较大变化,反过来又会使值函数发生改变,导致算法出现剧烈震荡甚至无法收敛.在实际应用中,状态和动作空间往往是大规模或连续的.如果采用仅近似值函数的离散动作方法来进行求解,必须面对合理离散化动作空间的问题,即确定哪部分动作空间更重要、选择离散粒度以及泛化最优策略到整个动作空间等问题.

策略梯度算法直接近似策略,能满足大规模或连续动作空间的需要,同时能保证收敛性,是近似强化学习的一个重要分支[9-10].策略梯度算法对策略进行随机近似并从样本中学习来实现策略梯度的无偏估计,因此,能克服基于值函数学习算法的收敛性难以得到保证的缺陷.但是从理论分析和实验性能的角度来说,这类算法在梯度估计时的较大方差会导致收敛速度慢和样本利用率低.行动器评判器算法(Actor Critic,简称 AC)是策略梯度算法[11-13]的一个子类,这类方法是对动态规划算法中的策略迭代的近似.AC 主要包含两个部分即行动器(actor)和评判器(critic).Actor 表示动作选择策略,通过概率分布实现状态到动作的映射,目标是实现控制;评判器相当于传统的状态值函数,即从初始状态出发得到的期望累积回报,目标是评估当前的策略.评估的主要方法有时间差分(Temporal Difference,简称 TD)算法[14-15]、Sarsa(λ)算法[16]、Q 学习算法[17]、最小二乘时间差分(Least Squares Temporal Difference,简称 LSTD)算法[18-19]和蒙特卡罗(Monte Carlo,简称 MC)算法[20].在 AC 算法中,采用值函数的时间差分误差(Temporal Difference error,简称 TD-error)来更新行动器的策略,可以大幅降低策略梯度的方差,从而提高算法的收敛速度.

随着深度学习的发展,结合深度学习和强化学习的深度强化学习方法已经成为近年来机器学习领域的一个研究热点.Mnih 等人[21]结合深度学习与 Q 学习算法,构建多层深度神经网络来逼近动作值函数,提出了深度 Q 网络(Deep Q-Network,简称 DQN).DQN 能在采用原始像素为输入的许多 Atari 视频游戏中取得人类水平的效果.为了进一步提高算法性能,Mnih 等人[22]采用多核 CPU 来代替 GPU,通过异步并行执行多个 Agent 来降低硬件要求和提高稳定性,使得该算法在 Atari 视频游戏中获得当时最先进的性能.为了使其应用于连续动作空间,Schulman 等人[23]提出了一种信任区域策略优化方法(Trust Region Policy Optimization,简称 TRPO),并证明了在采用代理损失函数和合适步长参数的情况下,策略能不断得到改进.Lillicrap 等人[24]结合确定性策略梯度(Deterministic Policy Gradient,简称 DPG)和 DQN,提出了深度确定性策略梯度方法 DDPG(Deep Deterministic Policy Gradient,简称 DDPG).与 DQN 相同的是,DDPG 也引入经验回放存储器来对深度网络进行离策略训练,以降低样本关联,同时采用时间差分算法确定的一致目标来训练当前网络参数.然而,DDPG 在利用目标网络时,采用软目标更新代替直接更新权值,并采用批量正则化方法来统一不同的物理变量单位和环境域,同时还引入一个独立的探索过程来添加噪声,从而增加算法的效率和鲁棒性.

以上基于深度非线性神经网络的方法在解决一些复杂问题,尤其是以原始像素作为输入的问题域中获得了突破性的进展.然而,当样本数量有限时,相对于基于线性函数逼近器的方法,多层的深度网络模型可能面临欠拟合问题,同时在训练网络模型时由于参数数量较多,导致其训练速度较线性模型慢.

在实际应用中,在线近似强化学习方法必须满足两个相互冲突的条件:

(1)样本效率高,即在与系统接触很短的时间后,就能获得一个较好的控制性能;

(2)计算效率高以适应实时系统的需要.

为了提高样本利用率和算法收敛速率,本文结合 AC 框架和线性函数逼近器并提出了一种采用模型学习和经验回放[25-26]加速最优策略学习的正则化自然 AC 算法(Regularized Natural AC with Model Learning and Experience Replay,简称 RNAC-ML-ER).本文的主要工作为:

(1)采用线性函数逼近器近似状态分量和立即奖赏,实现系统动态性的建模.每个状态分量对应一

个状态迁移向量，每个状态对应一个奖赏向量.状态迁移向量和奖赏向量均利用 Agent 与环境交互产生的样本进行更新.当每个时间步的模型误差满足约束时，采用状态迁移向量和奖赏向量进行规划，实现值函数、优势函数和策略的更新；

(2)将经验回放机制引入 AC 框架中，以提高数据利用率和计算效率.经验回放存储器中保存了与当前时刻较近的 N 个样本.在每个时间步，选择 TD-error 较大的一些样本用于更新值函数、优势函数和策略；

(3)提出了一种基于优势函数的自然梯度[27-31]更新方法，并通过正则化来降低策略梯度的方差，从而保证策略学习的稳定性；

(4)在指定的两个假设成立的条件下，通过理论分析证明了 RNAC-ML-ER 的收敛性；

(5)在 4 个经典的 RL 基准实验中将 RNAC-ML-ER 与离散动作空间方法以及连续动作空间方法进行对比，验证了所提算法的优越性.

2　相关工作

采用模型学习来加快算法收敛的方法具体可以分为离散状态动作方法、连续状态离散动作方法以及连续状态动作方法 3 种情形.

2.1　离散状态动作方法

Sutton 首次将模型学习与 Q 学习算法相结合，提出了一种解决离散状态动作空间问题的模型学习方法 Dyna-Q[32].Dyna-Q 采用在线获取的样本来学习精确的状态迁移函数和奖赏函数，模型在每个时间步都产生模拟样本进行规划，用于学习最优状态动作值函数.Peng 等人[33]和 Moore 等人[34]同时提出了优先级扫描方法 Prioritized sweeping，即在 Dyna-Q 的基础上，采用优先级队列存储 TD-error 不为 0 的状态动作；规划时仅更新优先级队列中存储的状态动作对，并将更新状态对应的 TD-error 不为 0 的前驱状态动作加入优先级队列中.为了学习一个更为精确的模型，Santos 等人[35]提出了一种针对角色扮演游戏的启发式模型学习算法 Dyna-H，启发式函数选择的动作是所有动作中下一个状态与目标状态欧式距离最大的动作，实现对状态空间更充分地探索.

2.2　连续状态离散动作方法

Sutton 等人提出了一种针对连续状态和离散动作空间的 Dyna 算法(Dyna algorithm based on Linear Function Approximation and Prioritized Sweeping，简称 Dyna-LFA-PS)[36].Dyna-LFA-PS 基于线性函数逼近器和优先级扫描技术，对每个离散的动作近似对应系统动态性的特征迁移矩阵和立即奖赏向量；规划阶段利用学习的模型和优先级队列实现值函数参数向量的更新.启发式 Dyna 优化算法(a Heuristic Dyna optimization algorithm using Approximate Model Representation，简称 HDyna-AMR)[37]在 Dyna-LFA-PS 的基础上引入状态动作值函数资格迹，加快状态动作值函数的收敛，同时通过启发式探索实现对状态空间更充分的探索，实现对模型更精确的估计；在优先级队列中对某个特征的所有前驱结点进行批量更新，以提高算法的样本效率和加快算法的收敛.

2.3　连续状态动作方法

Grondman 等人[38]提出了一种基于标准行动器评判器算法(Standard-Actor-Critic，简称 SAC)[39]并采用模型更新策略梯度的行动器评判器方法(Model Learning Actor-Critic，简称 MLAC).MLAC 采用存储器中与当前样本最近邻的 K 个样本，学习基于局部线性回归(Local Linear Regression，简称 LLR)的近似模型，并采用链式法则将值函数对下一个状态以及模型中下一个状态对动作的微分相乘用于更新策略梯度.同年，Grondman 等人[40]又提出了一种基于径向基函数(Radical Basis Functions，简称 RBFs)的模型学习行动器评判器方法 MLAC-RBFs，其采用 RBFs 表示特征，但实验结果表明 MLAC-RBFs 的收敛速度较 MLAC 慢.Costa 等人[41]结合 Dyna 和 MLAC 提出了 Dyna-MLAC 算法.Dyna-MLAC 采用 LLR 局部模型来改进策略梯度和进行模型规划以促进算法收敛.Cheng 等人[42]利用 LSTD 算法存储样本的性质，提出了两种行动器评判器算法，即基于资格迹的递推 rLSTD-AC 和增量的 iLSTD-AC.Gu 等人[43]针对以图片作为输入的控制任务提出了一个连续的 Q 学习算法，并采用迭代重拟合的线性函数近似局部模型以加速算法的收敛.Tamar 等人[44]提出了一种值迭代网络模型，利用观察的信息学习基于多层卷积神经网络表示的模型即迁移函数和奖赏函数，并利用学习的模型来规划，促进值函数收敛.

本文所提算法 RNAC-ML-ER 也是一种解决连续状态动作空间问题并通过模型学习加速收敛的方

《计算机学报》

法,其区别于以上模型学习方法主要表现在:

(1)近似模型表示方式. 对每个状态分量和奖赏直接进行线性近似,并采用梯度下降法对状态分量参数和奖赏参数进行更新;

(2)模型资格迹. RNAC-ML-ER 首次将模型资格迹引入模型学习中,以加快模型学习;

(3)模型误差阈值:在每个时间步,只有当模型的预测误差小于模型误差阈值时,才启动模型规划.

3 背景知识

3.1 马尔可夫决策过程

MDPs 可以被定义为四元组,即 $M=\langle X,U,f,u\rangle$,其中,$X\in\mathbb{R}^k$ 表示状态空间,$U\in\mathbb{R}^n$ 表示动作空间,$f:X\times U\times X\to[0,+\infty]$ 为状态迁移函数,$r:X\times U\times X\to R$ 为奖赏函数. 在某个时间步 t,状态为 $\boldsymbol{x}_t$,根据控制策略 $h:X\times U\to[0,1]$,选择动作 u_t,在动作 u_t 被执行完后,根据 $f(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1})$ 迁移至下一个状态 $\boldsymbol{x}_{t+1}$,并给定相应的立即奖赏 $r(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1})$. 在确定性环境中,转移的下一个状态和立即奖赏都是确定的,即 $f(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1})=1$,$r(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1})r_{t+1}$. Agent 根据策略 h 与环境进行交互,通过最大化期望折扣累积奖赏或期望平均奖赏来寻求最优策略.

假设初始状态为 $\boldsymbol{x}_0$,Agent 优化的目标为最大化从初始状态到目标状态即一个情节的长期回报. 长期回报可以分为期望折扣累积奖赏和期望平均奖赏这两种情形:

(1)期望折扣累积奖赏适合于情节式的任务,目标函数 $J(h)$ 可表示为

$$J(h)=E_{\boldsymbol{x}_{t+1}\sim f(\boldsymbol{x}_t,h(\boldsymbol{x}_t),\cdot)}\left[\sum_{t=1}^{T}\gamma^{t-1}r(\boldsymbol{x}_t,h(\boldsymbol{x}_t),\boldsymbol{x}_{t+1})\right] \tag{1}$$

其中,$0\leq\gamma<1$ 表示折扣因子,$T\geq1$ 表示情节的终止点.

(2)期望平均奖赏适合于与环境进行连续交互的无限水平 MDP,其目标函数 $J(h)$ 可表示为

$$J(h)=\lim_{T\to\infty}\frac{1}{T}E_{\boldsymbol{x}_{t+1}\sim f(\boldsymbol{x}_t,h(\boldsymbol{x}_t),\cdot)}\left[\sum_{t=1}^{T}r(\boldsymbol{x}_t,h(\boldsymbol{x}_t),\boldsymbol{x}_{t+1})\right] \tag{2}$$

其中,$T\geq1$ 表示 MDP 运行的终止时间.

在采用平均奖赏和折扣累积奖赏的情况下,状态动作值函数 $Q^h(\boldsymbol{x},u)$,简称动作值函数,即

$$Q^h(\boldsymbol{x},u)=E_{\boldsymbol{x}_{t+1}\sim f(\boldsymbol{x}_t,h(\boldsymbol{x}_t),\cdot)}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r(\boldsymbol{x}_t,h(\boldsymbol{x}_t),\boldsymbol{x}_{t+1})-R(h)\,|\,\boldsymbol{x}=\boldsymbol{x}_0,u=u_0\right] \tag{3}$$

其中,u_0 表示在初始状态 $\boldsymbol{x}_0$ 处执行的动作.

式(3)所示的 $Q^h(\boldsymbol{x},u)$ 可以表示两种情况:

① 平均奖赏. 折扣因子 $\gamma=0$ 且 $R(h)$ 为式(2)所示的 $J(h)$;

② 折扣累积奖赏. 折扣因子 $0<\gamma\leq1$ 且 $R(h)=0$.

最优状态动作值函数 $Q^*(\boldsymbol{x},u)$ 可表示为

$$Q^*(\boldsymbol{x},u)=\max_h Q^h(\boldsymbol{x},u) \tag{4}$$

最优策略 $h^*(\boldsymbol{x})$ 表示状态 $\boldsymbol{x}$ 对应的最优 Q 值的动作,即

$$h^*(\boldsymbol{x})=\arg\max_u Q^*(\boldsymbol{x},u) \tag{5}$$

3.2 行动器评判器方法

实际问题的状态和动作空间往往是大规模或连续的,值函数和策略无法被精确的表示并通过精确 RL 方法求解,只能通过近似强化学习方法求解. 在采用线性函数逼近器的情况下,动作值函数可以近似表示为

$$\hat{Q}(\boldsymbol{x},u,\boldsymbol{\theta})=\sum_{i=1}^{d}\phi_i(\boldsymbol{x},u)\theta_i=\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x},u)\boldsymbol{\theta} \tag{6}$$

其中,$\boldsymbol{\phi}=[\phi_1,\phi_2,\cdots,\phi_d]^{\mathrm{T}}$ 为状态动作的特征向量,$d=pm$ 为状态动作特征空间的维数,p 为状态特征空间的维数,m 为动作特征空间的维数,$\boldsymbol{\theta}=[\theta_1,\theta_2,\cdots,\theta_d]^{\mathrm{T}}$ 为状态动作值函数的参数向量.

状态值函数简称值函数,可以近似表示为

$$\hat{V}(\boldsymbol{x},\boldsymbol{v})=\sum_{i=1}^{p}\varphi_i(\boldsymbol{x})v_i=\boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{v} \tag{7}$$

其中,$\boldsymbol{\varphi}=[\varphi_1,\varphi_2,\cdots,\varphi_p]^{\mathrm{T}}$ 表示状态特征向量,$\boldsymbol{v}=[v_1,v_2,\cdots,v_p]^{\mathrm{T}}$ 表示值函数参数向量.

近似最优策略可以表示为

$$\hat{h}^*(\boldsymbol{x},\boldsymbol{\beta})=\sum_{i=1}^{p}\varphi_i(\boldsymbol{x})\beta_i=\boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{\beta} \tag{8}$$

其中,$\boldsymbol{\beta}=[\beta_1,\beta_2,\cdots,\beta_p]^{\mathrm{T}}$ 为策略参数.

连续动作空间的 AC 算法是一种近似强化学习方法. 由于动作值函数在连续动作空间中无法被表征,因此,通常仅近似值函数并通过 TD(λ)方法评估值函数. 在 TD(λ)中,首先需要计算当前状态的 TD-error. 在时刻 t 时,平均奖赏和折扣累积奖赏的 TD-error 可以统一表示为

$$\delta_t=r_{t+1}+\gamma V(\boldsymbol{x}_{t+1},\boldsymbol{v}_t)-V(\boldsymbol{x}_t,\boldsymbol{v}_t)-R(h) \tag{9}$$

其中,折扣因子 γ 和平均奖赏 $R(h)$ 的设置均与式(3)相同.

资格迹是 Sutton 和 Barto[16] 基于前向和后向等价的观点提出的一种加快算法收敛速度的技巧. 采用资格迹可以将当前状态的 TD-error 分配到访问过的历史状态上,以提高值函数的学习速率. 值函

数的资格迹在时刻 $t+1$ 时可被更新为

$$\boldsymbol{e}_{t+1}^{C} \leftarrow \lambda^{C}\gamma \boldsymbol{e}_{t}^{C}+\boldsymbol{\varphi}(\boldsymbol{x}_{t}) \tag{10}$$

其中，$0\leq\lambda^{C}<1$ 是值函数衰减因子.

在采用资格迹的情况下，值函数的参数向量 $\boldsymbol{v}_t$ 可以被更新为

$$\boldsymbol{v}_{t+1} \leftarrow \boldsymbol{v}_{t}+\alpha^{C}\delta_{t}\boldsymbol{e}_{t+1}^{C} \tag{11}$$

其中，$0\leq\alpha^{C}<1$ 为值函数的学习率.

3.3　策略梯度

采用式(3)统一平均奖赏和折扣累积奖赏后，设置目标函数为最大化动作值函数

$$J(h)=\int_{X} d^{h}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\int_{U} h(u|\boldsymbol{x})Q(\boldsymbol{x},u)\mathrm{d}u \tag{12}$$

其中，$h(u|\boldsymbol{x})$ 为动作探索策略，即在状态 $\boldsymbol{x}$ 采取动作 u 的概率. $d^{h}(\boldsymbol{x})$ 表示在策略 h 下状态 x 的分布：

$$d^{h}(\boldsymbol{x})=\begin{cases}\lim\limits_{t\to\infty} f(\boldsymbol{x}_{t}=\boldsymbol{x}|\boldsymbol{x}_{0},h), & \text{平均奖赏}\\ \sum\limits_{t=0}^{\infty}\gamma^{t} f(\boldsymbol{x}_{t}=\boldsymbol{x}|\boldsymbol{x}_{0},h), & \text{折扣累积奖赏}\end{cases} \tag{13}$$

采用式(12)对策略参数 $\boldsymbol{\beta}$ 求导可以得到策略梯度 $\nabla_{\boldsymbol{\beta}}J(h)$ 为

$$\nabla_{\boldsymbol{\beta}}J(h)=\int_{\boldsymbol{x}\in X} d^{h}(\boldsymbol{x})\int_{u\in U}\nabla_{\boldsymbol{\beta}}h(u|\boldsymbol{x})Q(\boldsymbol{x},u)\mathrm{d}u\mathrm{d}\boldsymbol{x} \tag{14}$$

根据 $\int_{U}\nabla_{\boldsymbol{\beta}}h(u|\boldsymbol{x})\mathrm{d}u=0$ 可知：任意与状态无关的基准函数 $b(\boldsymbol{x})$ 均满足 $\int_{U}\nabla_{\boldsymbol{\beta}}h(u|\boldsymbol{x})b(\boldsymbol{x})\mathrm{d}u=0$. 因此可采用仅与状态相关的基准函数 $b(\boldsymbol{x})$ 来降低梯度估计的方差. Bhatnagar 等人[45]已证明：当基准函数为状态值函数时，即 $b(\boldsymbol{x})=V(\boldsymbol{x})$ 时，策略梯度的方差最小，此时式(14)可以转换为

$$\nabla_{\boldsymbol{\beta}}J(h)=\int_{\boldsymbol{x}\in X} d^{h}(\boldsymbol{x})\int_{u\in U}\frac{\nabla_{\boldsymbol{\beta}}h(u|\boldsymbol{x})}{h(u|\boldsymbol{x})}A(\boldsymbol{x},u)\mathrm{d}u\,\mathrm{d}\boldsymbol{x} \tag{15}$$

其中，$A(\boldsymbol{x},u)=Q(\boldsymbol{x},u)-V(\boldsymbol{x})$ 称为优势函数. $\dfrac{\nabla_{\boldsymbol{\beta}}h(u|\boldsymbol{x})}{h(u|\boldsymbol{x})}=\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})$ 表示兼容型特征，即表示策略梯度的向量与表示策略分布的向量是兼容的. 此时，将式(15)可以进一步转换为期望形式：

$$\nabla_{\boldsymbol{\beta}}J(h)=E_{\boldsymbol{x}\sim d^{h}(\cdot),u\sim h(\cdot|\boldsymbol{x})}[\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})A(\boldsymbol{x},u)] \tag{16}$$

由于连续动作空间无法直接表示 $Q(\boldsymbol{x},u)$，因此优势函数也无法被显示表示. Bhatnagar 等人[45]已经证明式(9)所示的 TD-error 的期望是优势函数，因此，采用式(9)的期望来代替优势函数 $A(\boldsymbol{x},u)$，可以将式(16)所示的策略梯度转换为

$$\nabla_{\boldsymbol{\beta}}J(h)=E_{\boldsymbol{x}\sim d^{h}(\cdot),u\sim h(\cdot|\boldsymbol{x})}[\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})\delta] \tag{17}$$

4　正则化自然 AC 算法

4.1　动作选择策略

RL 中的动作探索策略有 3 种：ε-贪心策略、吉布斯探索策略和高斯探索策略. ε-贪心策略和吉布斯探索策略需要计算每个动作的选择概率，主要被应用于离散动作空间. RNAC-ML-ER 采用高斯探索策略，在任意状态 $\boldsymbol{x}_t$ 处，动作 u 被选择的概率为

$$h(\boldsymbol{x}_{t},u)=\frac{1}{\sqrt{2\pi}\sigma_{\tau}}\mathrm{e}^{-(u-u^{*})^{2}/2\sigma_{\tau}^{2}} \tag{18}$$

其中，σ_τ 表示探索方差，方差取值越大表示探索的程度越大，反过来则表示利用最优动作的程度更高. 本文采用自适应的方差更新方式，实现算法初期对动作空间充分的探索而后期对最优动作更多的利用，如下所示：

$$\sigma_{\tau+1}=\frac{\sigma_{\tau}}{\sqrt{\tau}} \tag{19}$$

其中，τ 表示当前迭代次数或情节数.

高斯探索策略对应的兼容型特征 $\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})\in\mathbb{R}^{p}$ 可以表示为

$$\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})=(u-\hat{h}^{*}(\boldsymbol{x},\boldsymbol{\beta}))\boldsymbol{\beta}/\sigma_{\tau}^{2} \tag{20}$$

在自然梯度框架下，式(8)中的最优动作 $\hat{h}^{*}(\boldsymbol{x},\boldsymbol{\beta})$ 可以近似为 $\hat{h}^{*}(\boldsymbol{x},\boldsymbol{\beta})=\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})\boldsymbol{\beta}$.

连续动作必须满足区间 $[u_{\min},u_{\max}]$ 约束，如下所示：

$$u=\begin{cases}u_{\max}, & u\geq u_{\max}\\ u_{\min}, & u\leq u_{\min}\end{cases} \tag{21}$$

4.2　正则化的自然梯度

自然梯度算法(Natural Gradient Actor-Critic，简称 NAC)[45]是依据监督学习中的 Fisher 信息度量对应的最陡峭上升方向的原理，通过采用 Fisher 信息矩阵逆线性地将传统的策略梯度转换为自然梯度. 自然梯度是在普通策略梯度的基础上左乘 Fisher 信息矩阵的逆[46]，即

$$\tilde{\nabla}_{\boldsymbol{\beta}}J(h)=\boldsymbol{G}^{-1}(\boldsymbol{\beta})\nabla_{\boldsymbol{\beta}}J(h) \tag{22}$$

其中，Fisher 信息矩阵 $\boldsymbol{G}(\boldsymbol{\beta})\in\mathbb{R}^{p\times p}$ 可以表示为

$$\begin{aligned}\boldsymbol{G}(\boldsymbol{\beta})&=E_{\boldsymbol{x}\sim d^{h}(\cdot),u\sim h(\cdot|\boldsymbol{x})}(\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})^{\mathrm{T}})\\ &=\int_{\boldsymbol{x}\in X} d^{h}(\boldsymbol{x})\int_{u\in U} h(u|\boldsymbol{x})\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})\cdot\\ &\quad\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})^{\mathrm{T}}\mathrm{d}u\,\mathrm{d}\boldsymbol{x}\end{aligned} \tag{23}$$

由于自然梯度需计算 Fisher 信息矩阵 $\boldsymbol{G}(\boldsymbol{\beta})$ 的逆，而求逆的时间复杂度为 $O(p^3)$，因此采用式(22)

《计算机学报》

计算自然梯度的时间复杂度为 $O(p^3)$. 在采用如 Shermon-Morrila 等递推公式进行求解时,计算自然梯度的时间复杂度可以降低到 $O(p^2)$.

基于自然梯度的策略参数更新可以表示为

$$\boldsymbol{\beta}_{t+1}\leftarrow\boldsymbol{\beta}_t+\alpha^A\boldsymbol{G}^{-1}(\boldsymbol{\beta})\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})\delta_t \tag{24}$$

其中,$0\leq\alpha^A\leq1$ 为策略学习率.

Shermon-Morrila 将优势函数参数化为 $A^h(\boldsymbol{x},u)=\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\omega}$,通过最小化近似值 $\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\omega}$ 与真实值 $A^h(\boldsymbol{x},u)$ 的均方误差来求优势函数参数向量 $\boldsymbol{\omega}$. 为了尽可能地降低策略方差,引入 ℓ_2-范数正则项并建立目标方程:

$$\varepsilon^h(\boldsymbol{\omega})=E_{\boldsymbol{x}\sim d^h(\cdot),u\sim h(\boldsymbol{x},\cdot)}[(\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\omega}-A^h(\boldsymbol{x},u))^2]+\ell\|\boldsymbol{\omega}\|^2 \tag{25}$$

其中,$\ell>0$ 为惩罚项.

采用式(25)对参数 $\boldsymbol{\omega}$ 求偏导数并令兼容特征 $\boldsymbol{\psi}(\boldsymbol{x},u)=\nabla_{\boldsymbol{\beta}}\log h(u|\boldsymbol{x})$ 可得

$$\nabla_{\boldsymbol{\omega}}\varepsilon^h(\boldsymbol{\omega})=2\int_{\boldsymbol{x}\in X}d^h(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\int_{u\in U}h(u|\boldsymbol{x})\cdot[\boldsymbol{\psi}(\boldsymbol{x},u)(\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x},u)\boldsymbol{\omega}-A^h(\boldsymbol{x},u))+\ell\boldsymbol{\omega}]\mathrm{d}u \tag{26}$$

文献[45]已证明 δ 的期望为 $A^h(\boldsymbol{x},u)$,因此 $\nabla_{\boldsymbol{\omega}}\varepsilon^h(\boldsymbol{\omega})$ 的近似值 $\hat{\nabla}_{\boldsymbol{\omega}}\varepsilon^h(\boldsymbol{\omega})$ 可以表示为

$$\hat{\nabla}_{\boldsymbol{\omega}}\varepsilon^h(\boldsymbol{\omega})=2[\boldsymbol{\psi}(\boldsymbol{x},u)\boldsymbol{\psi}(\boldsymbol{x},u)^{\mathrm{T}}\boldsymbol{\omega}-\boldsymbol{\psi}(\boldsymbol{x},u)\delta+\ell\boldsymbol{\omega}] \tag{27}$$

为加快优势函数收敛,引入优势函数资格迹:

$$\boldsymbol{e}^A_{t+1}\leftarrow\gamma\lambda^C\boldsymbol{e}^A_t+\boldsymbol{\psi}(\boldsymbol{x},u) \tag{28}$$

令 $\boldsymbol{\psi}_t=\boldsymbol{\psi}(\boldsymbol{x}_t,u_t)$,优势函数参数 $\boldsymbol{\omega}$ 可以更新为

$$\boldsymbol{\omega}_{t+1}\leftarrow\boldsymbol{\omega}_t+\alpha^C(-\boldsymbol{\psi}_t\boldsymbol{\psi}_t^{\mathrm{T}}\boldsymbol{\omega}_t+\boldsymbol{e}^A_{t+1}\delta_t-\ell\boldsymbol{\omega}_t) \tag{29}$$

在更新优势函数参数向量的基础上,策略参数可被更新为

$$\boldsymbol{\beta}_{t+1}\leftarrow\boldsymbol{\beta}_t+\alpha^A\boldsymbol{\omega}_{t+1} \tag{30}$$

值函数、优势函数和策略的更新如过程 1 所示. 由于 $\boldsymbol{\psi}_t^{\mathrm{T}}\boldsymbol{\omega}_t$ 的值为一个数值,因此式(29)和式(30)的最小计算复杂度均为 $O(p)$,低于传统自然梯度算法对应的 $O(p^3)$ 以及采用递推方法的 $O(p^2)$,具有计算效率高的优点. 过程 1 主要包括 TD-error 的计算、兼容性特征的计算、资格迹的更新(值函数和优势函数资格迹),参数的更新(值函数参数、优势函数参数和策略参数).

过程 1. Update 过程.

输入:$\boldsymbol{x}_t,u_t,r_{t+1},\boldsymbol{x}_{t+1},\boldsymbol{e}^C,\boldsymbol{e}^A,\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{\beta}$

输出:$\boldsymbol{e}^C,\boldsymbol{e}^A,\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{\beta}$

1. 更新值函数资格迹:$\boldsymbol{e}^C_{t+1}\leftarrow\gamma\lambda^C\boldsymbol{e}^C_t+\boldsymbol{\varphi}(\boldsymbol{x}_t)$
2. 更新 TD-error:$\delta_t=r_{t+1}+\gamma V(\boldsymbol{x}'_{t+1},\boldsymbol{v}_t)-V(\boldsymbol{x}_t,\boldsymbol{v}_t)$
3. 更新值函数参数:$\boldsymbol{v}_{t+1}\leftarrow\boldsymbol{v}_t+\alpha^C\delta_t\boldsymbol{e}^C_{t+1}$
4. 计算兼容型特征:$\boldsymbol{\psi}_t=\nabla_{\boldsymbol{\beta}}\log h(u_t|\boldsymbol{x}_t)$
5. 更新优势函数资格迹:$\boldsymbol{e}^A_{t+1}\leftarrow\gamma\lambda^C\boldsymbol{e}^A_t+\boldsymbol{\psi}_t$
6. 更新优势函数参数:$\boldsymbol{\omega}_{t+1}\leftarrow\boldsymbol{\omega}_t+\alpha^C(-\boldsymbol{\psi}_t\boldsymbol{\psi}_t^{\mathrm{T}}\boldsymbol{\omega}_t+\boldsymbol{e}^A_{t+1}\delta_t-\ell\boldsymbol{\omega}_t)$
7. 更新策略参数:$\boldsymbol{\beta}_{t+1}\leftarrow\boldsymbol{\beta}_t+\alpha^A\boldsymbol{\omega}_{t+1}$

4.3 模型学习与规划

强化学习的模型包括状态迁移函数和奖赏函数. 当模型已知时,通过模型规划可以大幅提高算法的收敛效率. RL 中的模型通常是未知的,为了进行规划,需要利用 Agent 与环境交互产生的数据来学习一个近似的模型. 由于线性函数近似器具有形式简单和计算效率高的优点,因此采用其对状态迁移函数和奖赏函数进行线性近似.

在时刻 t,Agent 与环境进行交互产生的样本为 $(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1},r_{t+1})$,$\boldsymbol{x}_t=\{x_{t,1},x_{t,1},\cdots,x_{t,k}\}$ 为当前状态,$\boldsymbol{\eta}_i\in\mathbb{R}^d(1\leq i\leq k)$ 为对下一个状态的第 i 个分量进行预测的参数向量,$\varsigma\in\mathbb{R}^d$ 为对奖赏进行预测的参数向量,则下一状态和奖赏可以预测为

$$\begin{cases}x'_{t+1,1}=\boldsymbol{\eta}_1^{\mathrm{T}}\boldsymbol{\phi}(x_{t,1},x_{t,2},\cdots,x_{t,k},u_t)\\x'_{t+1,2}=\boldsymbol{\eta}_2^{\mathrm{T}}\boldsymbol{\phi}(x_{t,1},x_{t,2},\cdots,x_{t,k},u_t)\\\quad\vdots\\x'_{t+1,k}=\boldsymbol{\eta}_k^{\mathrm{T}}\boldsymbol{\phi}(x_{t,1},x_{t,2},\cdots,x_{t,k},u_t)\\r'_{t+1}=\varsigma^{\mathrm{T}}\boldsymbol{\phi}(x_{t,1},x_{t,2},\cdots,x_{t,k},u_t)\end{cases} \tag{31}$$

将目标函数定义为最小化状态预测值 $x'_{t+1,i}$ 与实际值 $x_{t+1,i}$ 之间的均方误差,以及最小化奖赏预测值 r'_{t+1} 与实际值 r_{t+1} 之间的均方误差,其中,$1\leq i\leq k$ 且 $1\leq t\leq T$.

模型参数 $\boldsymbol{\eta}_i(1\leq i\leq k)$ 和 ς 可以被更新为

$$\begin{cases}\boldsymbol{\eta}_{t+1,1}\leftarrow\boldsymbol{\eta}_{t,1}+\alpha^M(x_{t+1,1}-x'_{t+1,1})\boldsymbol{\phi}_t\\\boldsymbol{\eta}_{t+1,2}\leftarrow\boldsymbol{\eta}_{t,2}+\alpha^M(x_{t+1,2}-x'_{t+1,2})\boldsymbol{\phi}_t\\\quad\vdots\\\boldsymbol{\eta}_{t+1,k}\leftarrow\boldsymbol{\eta}_{t,k}+\alpha^M(x_{t+1,k}-x'_{t+1,k})\boldsymbol{\phi}_t\\\varsigma_{t+1}\leftarrow\varsigma_t+\alpha^M(r_{t+1}-r'_{t+1})\boldsymbol{\phi}_t\end{cases} \tag{32}$$

其中,$\boldsymbol{\phi}_t=\boldsymbol{\phi}(x_{t,1},x_{t,2},\cdots,x_{t,k},u_t)$ 为时刻 t 对应的状态动作特征,$0\leq\alpha^M\leq1$ 为模型学习率.

为了加快模型的学习速率,在模型更新的过程中引入资格迹. 模型资格迹 $\boldsymbol{e}^M\in\mathbb{R}^d$ 的更新如下所示:

$$\boldsymbol{e}^M_{t+1}\leftarrow\gamma\lambda^M\boldsymbol{e}^M_t+\boldsymbol{\phi}_t \tag{33}$$

其中,$0\leq\lambda^M<1$ 是模型衰减因子.

在引入模型资格迹后,模型可被更新为

$$\begin{cases}\boldsymbol{\eta}_{t+1,1}\leftarrow\boldsymbol{\eta}_{t,1}+\alpha^M(x_{t+1,1}-x'_{t+1,1})\boldsymbol{e}^M_{t+1}\\\boldsymbol{\eta}_{t+1,2}\leftarrow\boldsymbol{\eta}_{t,2}+\alpha^M(x_{t+1,2}-x'_{t+1,2})\boldsymbol{e}^M_{t+1}\\\quad\vdots\\\boldsymbol{\eta}_{t+1,k}\leftarrow\boldsymbol{\eta}_{t,k}+\alpha^M(x_{t+1,k}-x'_{t+1,k})\boldsymbol{e}^M_{t+1}\\\varsigma_{t+1}\leftarrow\varsigma_t+\alpha^M(r_{t+1}-r'_{t+1})\boldsymbol{e}^M_{t+1}\end{cases} \tag{34}$$

《计算机学报》

每个时间步均需计算模型预测误差. 如果预测误差大于误差阈值，则意味着模型误差较大，在当前时间步不进行规划.

时刻 t 对应的模型预测误差为

$$Er_t=\max\left\{\left|\frac{x'_{t+1,1}-x_{t+1,1}}{x_{t+1,1}}\right|,\cdots,\left|\frac{x'_{t+1,k}-x_{t+1,k}}{x_{t+1,k}}\right|,\left|\frac{r'_{t+1}-r_{t+1}}{r_{t+1}}\right|\right\}\tag{35}$$

模型规划如算法 1 所示.

算法 1.　Planning 算法.

输入：$T^M,\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{\beta},\boldsymbol{\eta}_1,\boldsymbol{\eta}_2,\cdots,\boldsymbol{\eta}_k,\varsigma,\boldsymbol{x}$

输出：$\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{\beta}$

1. 初始化：$\boldsymbol{e}^C\leftarrow\boldsymbol{0},\boldsymbol{e}^A\leftarrow\boldsymbol{0},\boldsymbol{x}_1\leftarrow\boldsymbol{x},t\leftarrow1$
2. REPEAT（T^M次）
3.　根据式(18)和式(21)选择动作 u_t
4.　根据式(31)来预测 $x'_{t+1,i}(1\leqslant i\leqslant k)$和 r'_{t+1}
5.　调用 Update 过程
6.　根据式(19)更新策略探索方差 σ_{t+1}
7.　$t\leftarrow t+1$
8. END REPEAT

在算法 1 中，通过模型进行规划，即根据第 4 行来预测下一个状态和奖赏并组合当前状态和动作构成模拟样本，进一步更新值函数、优势函数和策略，以促进算法的收敛和提高样本效率.

4.4　经验回放

经验回放如算法 2 所示，经验回放是将在线学习获得的样本$(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1},r_{t+1})$存储在容量为 M 的经验回放存储器 D 中，并采用 RL 方法实现值函数更新.

算法 2.　Experience-replay 算法.

输入：$T^E,\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{\beta},D$

输出：$\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{\beta}$

1. 初始化：$\boldsymbol{e}^C\leftarrow\boldsymbol{0},\boldsymbol{e}^A\leftarrow\boldsymbol{0},t\leftarrow1$
2. LOOP（T^E次）
3.　从 D 中取出 δ 最大的样本$(\boldsymbol{x},u,r',\boldsymbol{x}',\delta)$
4.　调用 Update 过程
5.　$t\leftarrow t+1$
6. END LOOP

本文的经验回放方式区别于已有的经验回放方法主要体现在 4 个方面：

(1) 存储的样本形式. 样本形式由$(\boldsymbol{x}_t,u_t,r_{t+1},\boldsymbol{x}_{t+1})$变换为$(\boldsymbol{x}_t,u_t,r_{t+1},\boldsymbol{x}_{t+1},\delta_t)$，即加入样本的 TD-error 值 δ_t；

(2) 样本集的更新. 当 D 中存储的样本数量小于 M 时，将新样本直接存入 D 中. 否则，只有当新样本的 TD-error 大于存储器中所有样本的最小 TD-error 时，才能替代存储器中的最小 TD-error 的样本. 在每个情节初始时刻，经验回放存储器中的样本都会被清空；

(3) 资格迹的更新. 维护值函数和优势函数资格迹，以提高值函数和策略的学习速率；

(4) 参数向量的更新. 经验回放过程利用存储器中的样本来更新值函数参数、优势函数参数和策略参数.

4.5　RNAC-ML-ER 算法描述

RNAC-ML-ER 如算法 3 所示. 该算法主要包含 3 个部分：

(1) 在线学习. 采用 Agent 与环境在线交互获得样本$(\boldsymbol{x}_t,u_t,r_{t+1},\boldsymbol{x}_{t+1})$作为输入，调用过程 1 对值函数、优势函数、策略和模型进行在线学习，同时根据经验回放存储器容量和样本的 δ_t 来确定是否采用$(\boldsymbol{x}_t,u_t,r_{t+1},\boldsymbol{x}_{t+1},\delta_t)$填充数据集. 为了加快算法的收敛，在值函数、优势函数、策略和模型的学习过程中均引入资格迹；

(2) 模型规划（算法 1）. 当且仅当每个时间步的预测误差不超过误差阈值时，才采用学习的模型进行规划，即调用过程 1 实现值函数、优势函数和策略的更新；

(3) 经验回放（算法 2）. 经验回放过程即根据 TD-error 的值从大到小依次取出样本，并采用过程 1 来更新值函数、优势函数和策略.

在线学习、模型规划和经验回放均调用过程 1 对值函数、优势函数和策略进行多重更新. 多重更新的目的是通过实际样本、模拟样本以及存储样本的相互协作来加快算法收敛和提高样本效率.

算法 3.　RNAC-ML-ER 算法.

输入：$\lambda^M,\lambda^C,\alpha^A,\alpha^C,\alpha^M,M,\sigma,T^E,T^M,\ell$

输出：$\boldsymbol{v},\boldsymbol{\beta}$

1. 初始化：$\boldsymbol{\beta},\boldsymbol{v},\boldsymbol{\eta}_1,\boldsymbol{\eta}_2,\cdots,\boldsymbol{\eta}_k,\varsigma,\tau\leftarrow0,D\leftarrow\Phi$
2. REPEAT（每个情节）
3.　$\boldsymbol{x}_1\leftarrow\boldsymbol{x}_0,t\leftarrow1,\tau\leftarrow\tau+1,\boldsymbol{e}^M,\boldsymbol{e}^A,\boldsymbol{e}^C\leftarrow\boldsymbol{0}$
4.　REPEAT（每个时间步）
5.　根据式(18)和式(21)选择动作 u_t
6.　执行 u_t，观察 r_{t+1} 和 $\boldsymbol{x}_{t+1}=\{x_{t+1,1},\cdots,x_{t+1,k}\}$
7.　更新模型资格迹：$\boldsymbol{e}_{t+1}^M\leftarrow\gamma\lambda^M\boldsymbol{e}_t^M+\boldsymbol{\phi}_t$
8.　根据式(34)更新模型并根据式(35)计算 Er_t
9.　IF（$Er_t<0.1$）
10.　调用 Planning 算法
11.　END IF
12.　调用 Update 过程
13.　IF（$t\leqslant M$）
14.　将样本$(\boldsymbol{x}_t,u_t,r_{t+1},\boldsymbol{x}_{t+1},\delta_t)$插入 D 中
15.　ELSE IF（δ_t大于 D 中样本的最小 TD-error）

《计算机学报》

16. 采用($\boldsymbol{x}_t$,u_t,r_{t+1},$\boldsymbol{x}_{t+1}$,δ_t)替换 D 中此样本
17. END IF
18. 调用 Experience-replay 算法
19. $t \leftarrow t+1$
20. END REPEAT
21. 根据式(19)更新方差 σ_{t+1}
22. END REPEAT

5 理论分析

5.1 算法复杂度分析

5.1.1 时间复杂度分析

RNAC-ML-ER 算法融合了在线学习、规划和经验重放. 当状态特征空间维数为 p,动作特征空间维数为 m,状态动作特征空间维数为 $d=pm$ 时,算法复杂度可以从以下 3 个角度进行分析:

(1)在线学习过程. 在线学习过程中的每个时间步更新值函数、优势函数、策略和模型,对应的计算复杂度分别为 $O(p)$、$O(d)$、$O(p)$和 $O(d)$,总的计算复杂度为 $O(d)$;

(2)模型规划. 在线过程中的每个时间步调用规划过程,进行 T^M次规划,规划过程中每个时间步更新值函数、优势函数和策略,它们的计算复杂度分别为 $O(T^Mp)$、$O(T^Md)$和 $O(T^Mp)$,总的计算复杂度为 $O(T^Md)$;

(3)经验回放. 经验回放过程的计算开销包含两个部分:

① 经验回放存储器中样本实时更新. 由于本算法中的新样本不用于替换最旧样本,而是当新样本的 TD-error 大于存储器中所有样本的相应值时,才用于替代 TD-error 最小的样本,因此,在算法运行的初始阶段,样本实时更新的开销为 $O(1)$. 随着值函数、优势函数和策略慢慢趋向收敛,样本实时更新的开销渐渐趋向于 0;

② 值函数、优势函数和策略参数向量的更新. 在线过程中的每个时间步调用经验回放算法,抽取其中 T^E个样本来更新值函数、优势函数和策略,它们的计算复杂度分别为$O(T^Ep)$、$O(T^Ed)$和 $O(T^Ep)$,总的计算复杂度为 $O(T^Ed)$.

由于算法包含 S 个情节,每个情节包含 W 个时间步,则算法总的计算复杂度为 $O(SW(T^E+T^M+1)d)$.

5.1.2 空间复杂度分析

在线学习算法主要需存储值函数、优势函数、策略和模型对应的参数向量,存储开销分别为 $O(p)$、$O(d)$、$O(p)$和 $O(d)$,在引入经验重放后,增加的存储开销为 $O(M)$,由于 $M\geq d\geq p$,因此算法总的存储开销为 $O(M)$.

5.2 收敛性证明

Bhatnagar 等人[45]已证明采用 TD(λ)学习值函数的自然 AC 算法的收敛性,即在满足引理 1～3 的情况下,当优势函数有界时,自然 AC 算法能收敛. RNAC-ML-ER 也采用 TD(λ)学习值函数,并在行动器中应用自然梯度 RNAC-ML-ER 对文献[45]的策略梯度主要进行了两点改进:正则化和优势函数资格迹. 因此,证明过程分为两个部分展开:

(1)证明 RNAC-ML-ER 算法满足引理 1～引理 3 所规定的条件.

(2)证明优势函数的参数向量有界.

在证明前,先作下面 2 个假设:

假设 1. 值函数学习率 $a_t^C=1/t^{0.7}$和策略学习率 $a_t^A=1/t$,其中,t 表示当前情节.

假设 2. 状态动作特征 $\boldsymbol{\phi}_i(1\leq i\leq d)$和状态特征 $\boldsymbol{\varphi}_i(1\leq i\leq p)$均满足线性无关性.

引理 1. 在任意策略 h 下,RNAC-ML-ER 算法产生的 MDP 具有不可约性和非周期性.

证明. 不可约性是指 MDP 中的任意两个状态之间具有相互转移性. 在 RNAC-ML-ER 算法中,任意两个状态 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$之间必定存在一个转移函数 f,使得 $f(\boldsymbol{x},\boldsymbol{x}')>0$,所以任意状态 $\boldsymbol{x}$ 可以转移到状态 $\boldsymbol{x}'$,RNAC-ML-ER 算法的不可约性可以得证. 对于不可约 MDP,如果该 MDP 的任意一个状态具有非周期性,则该 MDP 具有非周期性. 因此,当某个状态具有自回归性时,它的非周期性就可以得证. 在 RNAC-ML-ER 算法中,对于任意状态 $\boldsymbol{x}$,必定存在一个状态使得 $f(\boldsymbol{x},\boldsymbol{x}')>0$ 成立,因此该状态具有自回归性,从而该 MDP 具有非周期性. 因此,引理 1 得证. 证毕.

引理 2. 值函数学习率 a_t^C和策略学习率 a_t^A满足 $0\leq a_t^C,a_t^A<1$,$\sum_{t=1}^{\infty}a_t^C=+\infty$,$\sum_{t=1}^{\infty}(a_t^C)^2<+\infty$,$\sum_{t=1}^{\infty}a_t^A=+\infty$,$\sum_{t=1}^{\infty}(a_t^A)^2<+\infty$且 $a_t^A=O(a_t^C)$,其中,t 表示当前情节.

证明. 如果 $f(t)$是$[1,+\infty]$上的非负函数,那么对于任意正数 L,$f(t)$在$[1,L]$上可积分,从而可以得到

$$f(n)\leq\int_{n-1}^{n}f(t)\mathrm{d}t\leq f(n-1),\ n=2,3,\cdots.$$

依次相加可以得到

$$\sum_{n=2}^{m} f(n) \leq \int_{1}^{m} f(t)\mathrm{d}t \leq \sum_{n=2}^{m} f(n-1) = \sum_{n=1}^{m-1} f(n).$$

如果反常积分 $\int_{1}^{+\infty} f(t)\mathrm{d}t$ 收敛，则对于任意整数 m：

$$S_m = \sum_{n=2}^{m} f(n) \leq f(1) + \int_{1}^{m} f(t)\mathrm{d}t \leq f(1) + \int_{1}^{+\infty} f(t)\mathrm{d}t.$$

由于正项级数 $\sum_{n=1}^{m} f(n)$ 收敛的充分条件是：部分和数列有界，则存在某个正数 Y，对于一切正整数 m 都有 $S_m \leq Y$，$\sum_{n=1}^{m} f(n)$ 存在一个上界即

$$Y = f(1) + \int_{1}^{m} f(t)\mathrm{d}t \leq f(1) + \int_{1}^{+\infty} f(t)\mathrm{d}t.$$

由此，当 $\int_{1}^{m} f(t)\mathrm{d}t$ 收敛时，$\sum_{n=1}^{m} f(n)$ 是收敛的.

令 $f(t)=1/t^p$，则有

$$\int_{1}^{m} 1/t^p \mathrm{d}t = \begin{cases} \dfrac{1}{1-p}(m^{1-p}-1), & \text{如果 } p \neq 1 \\ \ln m, & \text{其他} \end{cases}.$$

又因为

$$\lim_{m \to +\infty} \int_{1}^{m} 1/t^p \mathrm{d}t = \begin{cases} \dfrac{1}{p-1}, & \text{如果 } p>1 \\ +\infty, & \text{其他} \end{cases}.$$

因此，当 $p>1$ 时，$\int_{1}^{m} f(t)\mathrm{d}t$ 收敛且收敛解为 $1/(p-1)$，而 $p \leq 1$ 时发散于 $+\infty$.

假设 1 中的值函数学习率为 $a_t^C = 1/t^{0.7}$，策略学习率为 $a_t^A = 1/t$，由于 $p \leq 1$ 所以 $\sum_{t=0}^{\infty} a_t^C$ 和 $\sum_{t=0}^{\infty} a_t^A$ 发散到 $+\infty$，而 $(a_t^C)^2 = 1/t^{1.4}$ 和 $(a_t^A)^2 = 1/t^2$，因此 $\sum_{t=0}^{\infty} (a_t^A)^2$ 和 $\sum_{t=0}^{\infty} (a_t^C)^2$ 发散到 $+\infty$，同时，由于 $\lim_{t \to +\infty} a_t^A / a_t^C = 1/t^{0.3} = 0$. 因此引理 2 得证.　　证毕.

引理 3.　对于 MDP 中的任意状态动作对 $(\boldsymbol{x},u)$，$h(\boldsymbol{x},u)$ 在策略参数 $\boldsymbol{\beta}$ 上是连续可微的.

证明.　由 $h(\boldsymbol{x}_t,u) = \dfrac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(u-\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x}_t,u_t)\boldsymbol{\beta})^2/2\sigma^2}$ 可以看出 $h(\boldsymbol{x}_t,u)$ 在策略参数 $\boldsymbol{\beta}$ 上处处连续并可导，因此引理 3 得证.　　证毕.

定理 1.　当 $t \to +\infty$ 时，优势函数参数向量 $\boldsymbol{\omega}_t$ 以概率 1 收敛到 $\boldsymbol{\omega}_t \to \boldsymbol{\omega}^h$，其中 $\boldsymbol{\omega}^h$ 是根据式(29)得到的优势函数参数.

证明.　根据式(29)并以最小化 TD-error 为目标时，$\boldsymbol{\omega}$ 的微分方程可以转换为

$$\dot{\boldsymbol{\omega}} = E_{\boldsymbol{x} \sim d^h(\cdot), u \sim h(\boldsymbol{x},\cdot)}[-\boldsymbol{\psi}(\boldsymbol{x},u)\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x},u)\boldsymbol{\omega} + \boldsymbol{e}^A \delta^h - \ell\boldsymbol{\omega}] \tag{36}$$

令 $g(\boldsymbol{\omega}) = -\boldsymbol{\psi}(\boldsymbol{x},u)\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x},u)\boldsymbol{\omega} + \boldsymbol{e}^A\delta^h - \ell\boldsymbol{\omega}$，$g(\boldsymbol{\omega})$ 是关于 $\boldsymbol{\omega}$ 利普希茨连续的，则 $g_\infty(\boldsymbol{\omega})$ 可以表示为

$$g_\infty(\boldsymbol{\omega}) = \lim_{t \to \infty} \frac{g(r\boldsymbol{\omega})}{r} = -\boldsymbol{\psi}(\boldsymbol{x},u)\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x},u)\boldsymbol{\omega} - \ell\boldsymbol{\omega}.$$

对于微分系统 $\dot{\boldsymbol{\omega}} = -\boldsymbol{\psi}(\boldsymbol{x},u)\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x},u)\boldsymbol{\omega} - \ell\boldsymbol{\omega}$. 根据文献[45]，在假设 2 成立的情况下，Fisher 信息矩阵 $\boldsymbol{G}(\boldsymbol{x},u) = \boldsymbol{\psi}(\boldsymbol{x},u)\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x},u)$ 是正定的. ℓ 是一个很小的正数，因此 $\dot{\boldsymbol{\omega}}$ 的积分形式是一个渐进的稳定平衡态 Lyapunov 函数 $F(\boldsymbol{\omega}) = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}/2$.

定义 $M(t)$ 为

$$M(t) = (-\boldsymbol{G}_0(\boldsymbol{x},u)\boldsymbol{\omega} + \boldsymbol{e}^A\delta^h - \ell\boldsymbol{\omega}) + E[\boldsymbol{G}_0(\boldsymbol{x},u)\boldsymbol{\omega} - \boldsymbol{e}^A\delta^h + \ell\boldsymbol{\omega} \mid N(t)] \tag{37}$$

其中，$N(t) = \tau(\boldsymbol{\omega}_r, M(t), r \leq t)$.

由式(37)可知：存在一个常数 $C_0 < \infty$，使得

$$E[\|M(t+1)\|^2 \mid N(t)] \leq C_0(1 + \|\boldsymbol{\omega}\|^2).$$

定义 $\bar{F}(\boldsymbol{\omega}) = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}/2$，然后对式(36)积分可以得到

$$\bar{F}(\boldsymbol{\omega}) = ((\boldsymbol{\omega} + (\boldsymbol{G}_0(\boldsymbol{x},u) + \ell I)^{-1}E[\boldsymbol{e}^A\delta^h])^{\mathrm{T}} (\boldsymbol{\omega} + (\boldsymbol{G}_0(\boldsymbol{x},u) + \ell I)^{-1}E[\boldsymbol{e}^A\delta^h]))/2 \tag{38}$$

采用 $\bar{F}(\boldsymbol{\omega})$ 对 $\boldsymbol{\omega}$ 先求导，再由 $\boldsymbol{\omega}$ 对 t 求导，根据链式法则可以得到

$$\begin{aligned} \frac{\mathrm{d}\bar{F}(\boldsymbol{\omega})}{\mathrm{d}t} &= \nabla_{\boldsymbol{\omega}}\bar{F}(\boldsymbol{\omega})^{\mathrm{T}}\dot{\boldsymbol{\omega}} \\ &= -(\boldsymbol{\omega} + (\boldsymbol{G}_0(\boldsymbol{x},u) + \ell I)^{-1}E[\boldsymbol{e}^A\delta^h]) (\boldsymbol{G}_0(\boldsymbol{x},u)\boldsymbol{\omega} + E[\boldsymbol{e}^A\delta^h] + \ell\boldsymbol{\omega}) \\ &< 0 \end{aligned}$$

$$\text{s.t. } \boldsymbol{\omega} \neq -(\boldsymbol{G}_0(\boldsymbol{x},u) + \ell I)^{-1}E[\boldsymbol{e}^A\delta^h]$$

由文献[47]可知，$\boldsymbol{\omega} = -(\boldsymbol{G}_0(\boldsymbol{x},u) + \ell I)^{-1}E[\boldsymbol{e}^A\delta^h]$ 是式(29)具有稳定解. 此时再根据文献[48]中的定理 2.2 可知，在引理 1～3 成立的条件下，$\boldsymbol{\omega}$ 以概率 1 收敛到 $\boldsymbol{\omega}^h$.　　证毕.

定理 2.　在满足假设 1～2 以及引理 1～3 成立的条件下，由优势函数参数 $\boldsymbol{\omega}$ 收敛以及策略参数的更新公式 $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \alpha^A\boldsymbol{\omega}_{t+1}$ 可知，当 $t \to \infty$ 时，RNAC-ML-ER 所得的 $\boldsymbol{\beta}$ 以概率 1 收敛.

证明.　由于定理 1 已经证明优势函数参数向量 $\boldsymbol{\omega}$ 在 $t \to \infty$ 时以概率 1 收敛到 $\boldsymbol{\omega}^h$，即 $\boldsymbol{\omega}^h = \lim_{t \to \infty} \boldsymbol{\omega}_t$，因此优势函数参数向量是有界的；同时由于 RNAC-ML-ER 满足引理 1～引理 3，则根据文献[45]可知策略必然收敛.

计算机学报

根据假设 1,策略参数的学习率 α^A 要小于优势函数参数的学习率 α^C,因此优势函数参数将会先于策略参数收敛.根据 $\boldsymbol{\beta}$ 的更新公式 $\boldsymbol{\beta}=\lim_{t\to\infty}\boldsymbol{\beta}_t=\boldsymbol{\beta}_{t-1}+\alpha^A\boldsymbol{\omega}_t$ 可知:当 $\boldsymbol{\omega}_t$ 收敛时,$\boldsymbol{\beta}$ 以概率 1 收敛.证毕.

6 实验结果分析

为了验证模型学习和经验回放的功能,将基于模型学习加速的正则化自然 AC 算法 RNAC-ML(regularized natural AC with model learning)和 RNAC-ML-ER 在 4 个基准实验即平衡杆、小车上山、倒立摆和体操机器人中进行仿真.4 个实验中一些共同的参数设置为:折扣因子 $\gamma=0.9$、值函数和优势函数衰减因子 $\lambda^C=0.9$、模型衰减因子 $\lambda^M=0.4$、探索方差初始值 $\sigma_0=0.9$、模型规划次数 $T^M=30$、经验回放次数 $T^E=50$ 以及经验回放存储器的容量 $M=100$. 4 个实验中的状态特征和状态动作特征均采用 RBFs 进行编码,并进行归一化使得特征分量位于[0,1].

采用模型进行加速学习时,模型学习率是满足(0,1)约束的,对于某些简单的问题域,模型学习率取值过大时,会导致模型过拟合,不满足误差阈值的需求,同理,如果模型学习率取值过小,会使得模型欠拟合,也不能满足误差阈值的需求,两种情况都使得大部分时间步都无法有效启动规划过程.因此,在所有实验中,都将模型学习率 α^M 设置为 0.5.

6.1 平衡杆

平衡杆如图 1 所示.

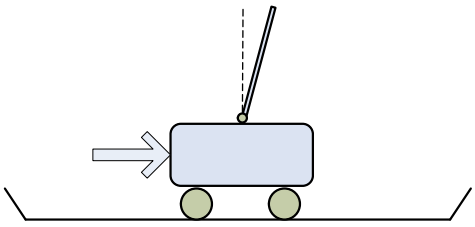


图 1 平衡杆示意图

在水平方向上有一辆可以左右移动的小车,小车质量为 $M_c=1\,\mathrm{kg}$,小车上放置了一根质量为 $m_p=0.1\,\mathrm{kg}$ 的杆子,杆子长度为 $l=1\,\mathrm{m}$.当杆子与竖直方向的角度满足 $[-\pi/4,\pi/4]$ 时,杆子被认为是平衡的.由于杆子一端固定在一辆左右移动的小车上,因此,在没有外力的作用下,杆子无法持续保持平衡.该实验的目标就是通过学习一个策略,即如何对小车施加水平向左或向右的作用力 F,使杆子保持平衡.F 的施加频率是 $\Delta t=0.1\,\mathrm{s}$,其范围为 $[-50,50]\mathrm{N}$(正力的方向向右,反之向左).任意时刻 F 会受水平方向的 $[-10,10]\mathrm{N}$ 范围内的噪声干扰.

该问题的状态是两维向量 $(\boldsymbol{\omega},\dot{\boldsymbol{\omega}})$,其中,$\boldsymbol{\omega}$ 表示杆子与竖直方向的夹角,而 $\dot{\boldsymbol{\omega}}$ 表示杆子的角速度.角加速度 $\ddot{\boldsymbol{\omega}}$ 的计算可以表示为

$$\ddot{\boldsymbol{\omega}}=\frac{g\sin\boldsymbol{\omega}+\cos\boldsymbol{\omega}\left(\dfrac{-F-ml\dot{\boldsymbol{\omega}}^2\sin\boldsymbol{\omega}}{m_p+M_c}\right)}{l\left(\dfrac{4}{3}-\dfrac{m_p\cos^2\boldsymbol{\omega}}{m_p+M_c}\right)} \tag{39}$$

其中 $g=9.811\,\mathrm{m/s^2}$ 为重力加速度.

杆子在每个时间步的角度 $\boldsymbol{\omega}_t\in[-\pi/2,\pi/2]$ 和角速度 $\dot{\boldsymbol{\omega}}_t\in[2,2]$ 可以分别根据 $\boldsymbol{\omega}_t=\boldsymbol{\omega}_{t-1}+\dot{\boldsymbol{\omega}}_t\Delta t$ 和 $\dot{\boldsymbol{\omega}}_t=\dot{\boldsymbol{\omega}}_{t-1}+\ddot{\boldsymbol{\omega}}_t\Delta t$ 进行更新.奖赏函数设置为:角度 $\boldsymbol{\omega}$ 大于 $\pi/4$ 时奖赏为 -1,否则为 1.情节结束的条件为:当杆子与竖直方向的角度大于 $\pi/4$ 或杆子平衡 3000 个时间步.

初始状态定义为 $(\boldsymbol{\omega},\dot{\boldsymbol{\omega}})=(0,0)$.算法共运行 400 个情节,每个情节的最大时间步为 3000,每个时间步的仿真时间为 0.1 s.本文所有的实验在采用 RBF 对状态特征和状态动作编码时,采用相同的设置方式.以平衡杆为例,由于状态包含角度和角速度,而动作是施加的力,角度的范围为 $[-\pi/2,\pi/2]$,角速度的范围为 $[-2,2]$.当角度方向的特征空间维数为 6 时,取角度中心点的最小值为 -1.5,可以得到角度的中心点为 $\{-1.5,-1,-0.5,0,0.5,1,1.5\}$;选择角速度的最小中心点为 -1.5 时,根据中心点个数的设置值为 3 时,可以得到角速度的中心点为 $\{-1.5,0,1.5\}$.同理,当动作的边界最小值为 -45 时,由于中心点的个数设置为 10 时,可以得到动作中心点为 $\{-45,-35,-25,-15,-5,5,15,25,35,45\}$.方差为区域最大值与最小值之差与中心点个数之商的平方,如角度方差为 $(\pi/(2\times7))^2\approx0.05$,角速度的方差为 0.44,动作方差为 25.

为了使算法在具有较快收敛速度的同时具有较高的稳定性,需对正则化惩罚因子进行合理设置.正则化惩罚因子的值越大,算法更侧重稳定性,即寻优的参数变化更缓慢,防止陷入局部最优;而当正则化惩罚因子的值越小,则更侧重学习速度,即以尽可能快的速度寻求最优解.在平衡杆实验中,当正则化惩罚因子固定在(0,1)之间时,可选值为 $\{0.01,0.05,0.1,0.4,0.8\}$,RNAC-ML-ER 对应的收敛性曲线如图 2(a)、(b)、(c)、(d)和(e)所示.

《计算机学报》

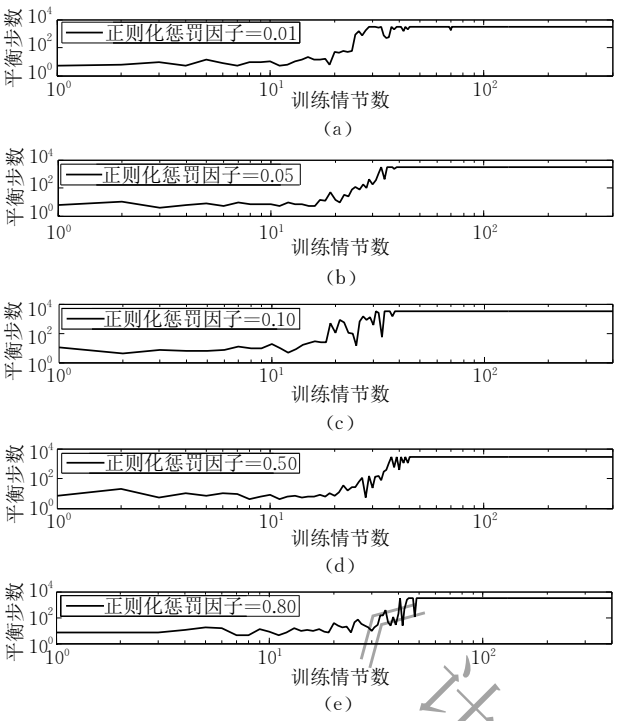


图 2　不同正则化惩罚因子的收敛性曲线

从图 2 中可以看出，当正则化惩罚因子为 0.01(图 2(a))时，在前 20 个情节中收敛速度最快，然而在第 40 个情节后仍有剧烈的震荡；当正则化惩罚因子为 0.05(图 2(b))时，虽然在前 20 个情节中，收敛速度较 0.01 时慢，但在 20 个情节后较其它设置值具有更快的收敛速度，同时收敛后能一直保持稳定；当正则化惩罚因子为 0.1(图 2(c))时，在前 20 个情节中，学习速度较慢；而正则化惩罚因子为 0.4(图 2(d))和 0.8(图 2(e))时，学习速度均较慢，且分别于第 46 和 49 个情节才趋于收敛. 综合上述，正则化惩罚因子取 0.05 时，算法具有更好的收敛性能.

为了对 RNAC-ML 和 RNAC-ML-ER 进行验证，将其分别与离散动作空间算法 Sarsa(λ)[16]、Dyna-LFA-PS[36]和 HDyna-AMR[37]以及连续动作空间算法 SAC[39]、NAC[45]和 MLAC[38]进行比较. 离散动作算法将动作空间[−50,50]N 离散化为−50 N、0 和 50 N. 8 种算法的平衡步数随着训练情节数增加的变化如图 3 所示.

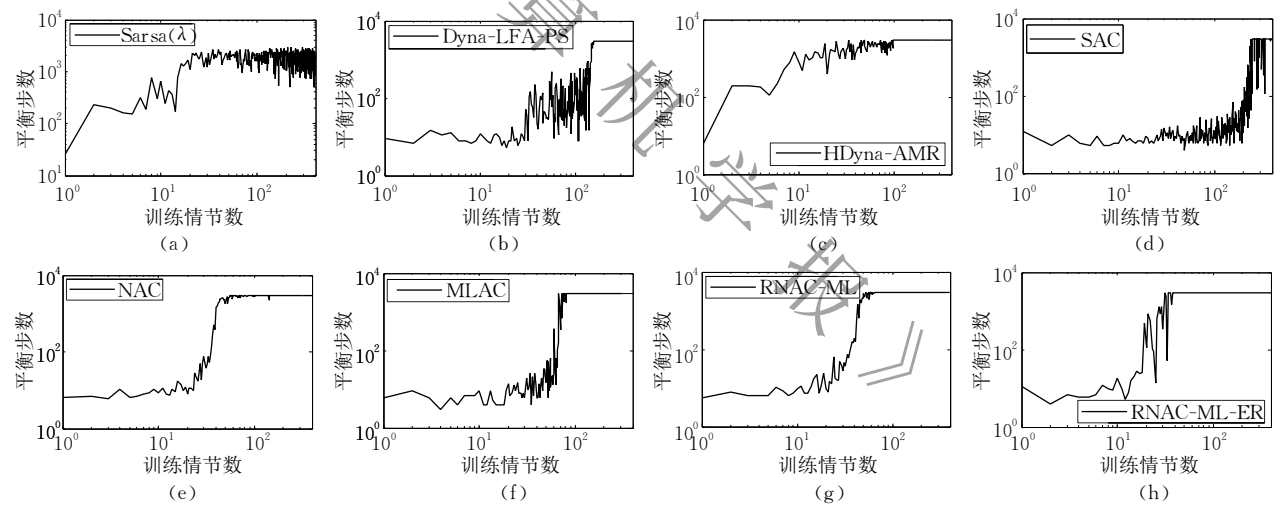


图 3　RNAC-ML 和 RNAC-ML-ER 与连续和离散动作算法的平衡步数比较

从图 3(h)可以发现，RNAC-ML-ER 在第 38 个情节就开始收敛且能平衡 3000 步，因此它的收敛效果最好. 与离散的动作算法 Sarsa(λ)(图 3(a))、Dyna-LFA-PS(图 3(b))和 HDyna-AMR(图 3(c))相比，RNAC-ML 和 RNAC-ML-ER 具有更好的收敛效果. 在前 30 个情节中，Sarsa(λ)和 HDyna-AMR 的学习速率相对其它方法更快. HDyna-AMR 在第 99 个情节处趋于收敛，而 Sarsa(λ)在整个仿真期间一直剧烈震荡，只在极少数的情节上能平衡 3000 个时间步. Dyna-LFA-PS 在前 146 个情节中学习速率较慢且平衡时间步少于 900，在此之后开始能平衡 2000 个时间步，最终在第 153 个情节时趋于收敛. 与 Sarsa(λ)相比，基于模型的方法 Dyna-LFA-PS 和 HDyna-AMR 均能收敛. 由于 HDyna-AMR 采用了启发式的探索机制，所以在仿真期间收敛速率优于 Dyna-LFA-PS.

与连续的动作算法 SAC(图 3(d))、NAC(图 3(e))和 MLAC(图 3(f))相比，RNAC-ML 和 RNAC-ML-ER 在学习过程中震荡幅度较小，这主要是由于正则化方法、模型规划和经验回放使得学习过程具有更好的稳定性. NAC 在前 40 个情节中学习速率更快，但 RNAC-ML 在第 58 个情节时趋于收敛，因

此 RANC-ML 的收敛速度较 NAC 更快. NAC 在第 63 个情节首次实现平衡 3000 个时间步,并在后来的大部分情节中都能保持杆子平衡,但在某些情节处平衡步数仍出现大幅震荡. 3 种基于自然梯度的方法即 RNAC-ML-ER、RNAC-ML 和 NAC 较 SAC 和 MLAC 收敛速度更快,表明了自然梯度较普通的策略梯度收敛效果更好. MLAC 在第 81 个情节后就趋于收敛,且其收敛性能在仿真期间一直能保持稳定. SAC 算法在第 243 个情节时首次实现平衡 3000 个时间步,但在此之后一直剧烈震荡,直到第 334 个情节时才趋于收敛. RNAC-ML-ER 和 RNAC-ML 的收敛速度较 NAC 更快,同时 MLAC 的收敛速度较 SAC 更快,显示了基于模型学习的方法具有更好的收敛性. RNAC-ML-ER 较 RNAC-ML 的收敛效果更好意味着经验回放能在模型学习的基础上通过储存的样本进一步促进值函数和策略的收敛.

将 RNAC-ML 和 RNAC-ML-ER 与基于深度神经网络的 TRPO 和 DDPG 进行比较,TRPO 和 DDPG 均采用 2 个 32×32 的隐藏层分别学习优势函数和动作值函数,TRPO 和 DDPG 的参数设置与开源平台[49]保持一致,但将 TRPO 和 DDPG 中的奖赏函数更改为与本实验设置相同,4 种方法得到的平衡步数如图 4 所示,从图 4 中可以看出,在初始的 16 个情节中,TRPO 算法的学习速度最快,但之后一直剧烈震荡始终不能收敛. 4 种算法中收敛速度最快的是 DDPG,在第 26 个情节时就已首次趋于收敛,但是在情节 186～279 之间又开始剧烈震荡,直到 280 个情节后开始真正收敛;RNAC-ML-ER 和 RNAC-ML 分别收敛于第 38 个情节和 58 个情节. 虽然 RNAC-ML 和 RNAC-ML-ER 的收敛速度较 DDPG 慢,但收敛后一直能保持稳定. 这可能是由于 RNAC-ML 和 RNAC-ML-ER 通过引入模型学习和正则化,均能平衡在线学习过程中由于探索而带来的不稳定性,尽可能地实现算法稳定.

为了对样本效率进行验证,将 8 种不同算法在是否收敛、首次平衡所需样本、收敛所需总样本、收敛时对应的情节和总训练时间上进行比较,如表 1 所示. 从表 1 中可以看出,除了 Sarsa(λ)和 TRPO 无法收敛外,其它 8 种算法均能收敛 RNAC-ML-ER 在第 38 个情节开始收敛,且收敛时所需的样本数 22753 远少于其它方法. 按收敛所需样本递增对所有算法进行排序,依次为 RNAC-ML-ER、RNAC-ML、MLAC、HDyna-AMR、Dyna-LFA-PS、NAC、DDPG 和 SAC. 显然,前 5 种算法由于采用了模型学习因而具有更好的样本效率. 此外,RNAC-ML-ER 在样本效率和收敛效果上均优于 RNAC-ML,这表明经验回放能在一定程度上改善算法的性能. 从总训练时间的比较中可以看出,基于深度神经网络近似器的方法 TRPO 和 DDPG 远远超过基于线性函数近似器的方法即 RNAC-ML-ER、RNAC-ML、MLAC、NAC、SAC、Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR. 这是因为即使它们仅采用 2 个隐含层的简单网络模型,但其学习的参数数量仍然远远超过线性函数逼近器需要学习的参数数量,因此导致训练效率较低.

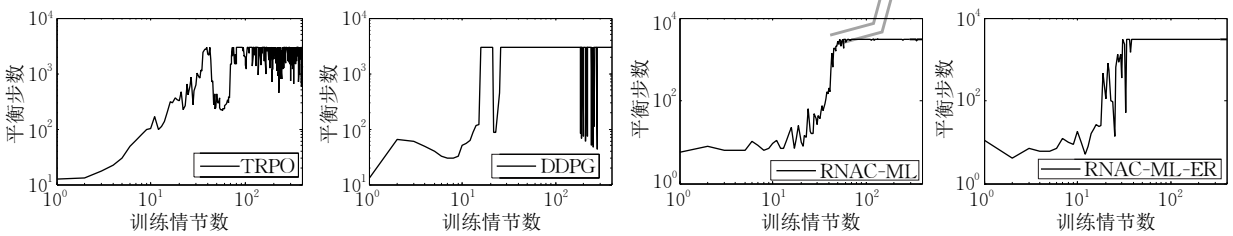


图 4 RNAC-ML 和 RNAC-ML-ER 与深度网络模型 TRPO 和 DDPG 的平衡步数比较

表 1 不同算法在平衡杆实验中的性能比较

算法名称	是否收敛(Y/N)	首次平衡所需样本	收敛所需总样本	收敛对应情节	总训练时间/min
Sarsa(λ)	N	71	19795	89	5.6
Dyna-LFA-PS	Y	43989	43989	153	2.6
HDyna-AMR	Y	39140	40564	99	2.7
SAC	Y	5807	246159	334	0.5
NAC	Y	43336	193716	110	1.5
MLAC	Y	4580	35535	81	8.6
TRPO	N	54143	—	—	17.0
DDPG	Y	3838	737656	280	90.0
RNAC-ML	**Y**	**21831**	**24549**	**58**	**4.2**
RNAC-ML-ER	**Y**	9856	**22753**	**38**	**1.1**

6.2　小车上山

小车上山如图 5 所示，它是强化学习中一个经典的情节式任务.

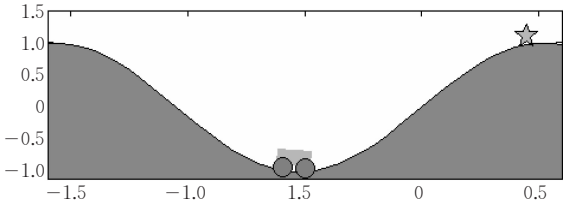


图 5　小车上山实验

在每个情节的初始时刻，小车位于两座山的坡底，而在情节的结束时刻，小车或是已经达到目标位置(五角星所示)或是已达到每个情节所允许的最大时间步. 小车在坡底时，由于动力不足，无法通过油门直接加速到达坡顶.

小车在时刻 t 时对应的状态采用二维向量 $\boldsymbol{x}_t=(p_t,v_t)$ 来表示，$p_t\in[-1.2,+0.5]$ 为小车在水平方向的位置，$v_t\in[-0.07,+0.07]$ 表示小车在水平方向的速度. 小车可以执行的动作 $u_t\in[-1,-1]$，在执行了动作 u_t 后，小车状态被更新为

$$\begin{cases}v_{t+1}=v_t+0.001u_t+g\cos(3p_t)\\ p_{t+1}=p_t+v_{t+1}\end{cases}\tag{40}$$

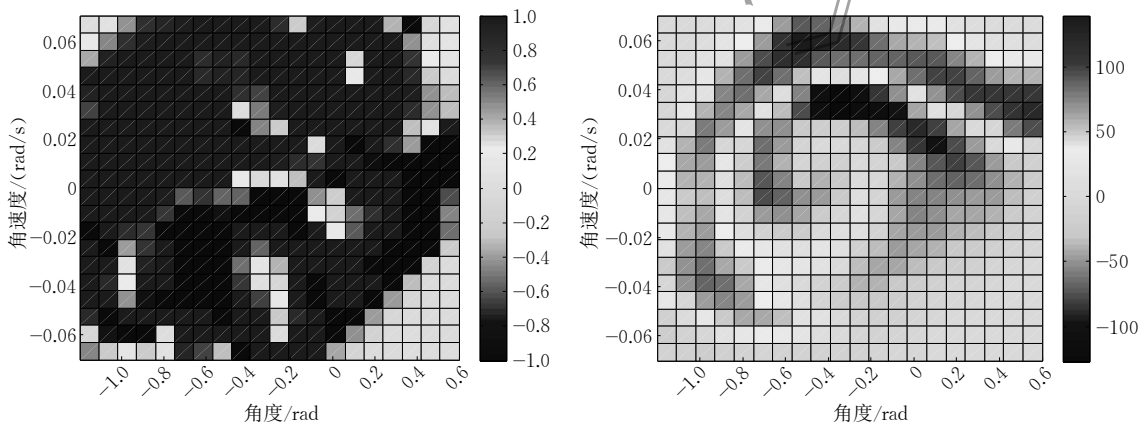
其中，$g=-0.0025$ 为重力加速度.

动作 u_t 被执行后，如果小车达到目标位置，则得到的立即奖赏为 0，否则为 −1，立即奖赏函数可以表示为

$$\rho_t(\boldsymbol{x}_t,u_t,\boldsymbol{x}_{t+1})=\begin{cases}0, & p_{t+1}=0.5\\ -1, & -1.2<p_{t+1}<0.5\end{cases}\tag{41}$$

状态特征中心点取自二维网格{−1.05，−0.9，−0.75，−0.6，−0.45，−0.3，−0.15，0，0.15，0.3，0.45}×{−0.058，−0.046，−0.034，−0.022，−0.01，0.002，0.014，0.026，0.038，0.05}. 位置和速度的方差分别取 0.006 和 0.000 05. 状态动作特征的中心点取自三维空间{−1.05，−0.9，−0.75，−0.6，−0.45，−0.3，−0.15，0，0.15，0.3，0.45}×{−0.058，−0.046，−0.034，−0.022，−0.01，0.002，0.014，0.026，0.038，0.05}×{−0.6，−0.2，0.2，0.6}，动作方差为 0.06.

初始状态定义为 $\boldsymbol{x}_0=(p_0,v_0)=(-0.5,0)$. 算法共运行 500 个情节，每个情节的最大时间步为 3000，每个时间步的仿真时间为 0.01 s. 将 RNAC-ML-ER 算法应用于该实验中，训练得到的最优策略和最优值函数如图 6 所示，从图 6(a)中可以看出绝大多数状态对应的最优策略为 1、0 和 −1，其中，最优策略为 1 的状态最多，几乎占据了整个状态空间的 1/2. 从图 6(b)中可以看出，当小车状态为 $p\in[-0.6,0]$ 且 $v\in[0.02,0.04]$ 时而小车状态为 $p\in[0,0.3]$ 且 $v\in[0,0.02]$ 时，训练得到的最优值函数几乎为最小值. 对于左边坡顶的一些状态如 $p\in[-1.4,-0.8]$ 且 $v\in[0,0.02]$ 时，虽然这些状态距离目标状态较远，但由于它们能使小车在下坡过程中获得足够的动量，因此，这些状态对应的值函数反而较高约为 50.



(a) 训练得到的最优策略　　(b) 训练得到的最优值函数

图 6　训练得到的最优策略和最优值函数

将 RNAC-ML 和 RNAC-ML-ER 分别与离散动作空间算法 Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR 以及连续动作空间算法 SAC、NAC 和 MLAC 进行比较. RNAC-ML 和 RNAC-ML-ER 中的正则化惩罚因子设置为 $\ell=0.1$. 离散动作算法将动作空间[−1,1]离散化为 −1、0 和 1. 8 种算法的平衡步数随着训练情节数增加的变化如图 7 所示，8 种算法的收敛情况分别为：RNAC-ML 在第 44 个情节收敛到 115 个时间步；RNAC-ML-ER 在第 8 个情节收敛到 110；Sarsa(λ)在第 89 个情节收敛到 115；

Dyna-LFA-PS 在第 86 个情节处收敛到 121；HDyna-AMR 在第 72 个情节收敛到 92；SAC 在第 256 个情节处收敛到 160；NAC 在第 225 个情节处收敛到 148；MLAC 在第 45 情节处收敛到 148. 因此，收敛最快的算法是 RNAC-ML-ER，其次是 RNAC-ML，但是收敛精度最高的是离散动作空间算法 HDyna-AMR. 此外，从实验结果发现，离散动作算法无论是在收敛速度和精度上均表现更好. 结合图 7 可发现，由于 RNAC-ML-ER 求解的最优策略最终也基本属于{−1,0,1}，因此，离散动作算法反而具有更好的收敛解.

将 RNAC-ML 和 RNAC-ML-ER 与 TRPO 和 DDPG 进行比较，TRPO 和 DDPG 的参数设置与开源平台保持一致，4 种方法得到的平衡步数如图 8 所示，在最大时间步为 1000 的情况下，DDPG 在此实验中始终无法到达目标，无法得到有效的时间差分误差而实现策略的学习，最终未能收敛并获得最优策略. TRPO 初始学习速率较慢，在第 33 个情节时才开始探索到达目标的路径，并在第 58 个情节时才收敛到 93，其表现仅次于离散动作算法 HDyna-AMR，但优于连续动作算法 SAC、NAC、MLAC、RNAC-ML 和 RNAC-ML-ER.

为了进一步对 8 种算法的性能进行全面比较，在算法是否收敛、达到目标小于 120 个时间步的首个情节、收敛所需的总样本数、收敛对应的情节、收敛解、总训练时间多个指标上进行比较，得到的结果如表 2 所示，从表 2 中可以发现，所有算法中收敛速

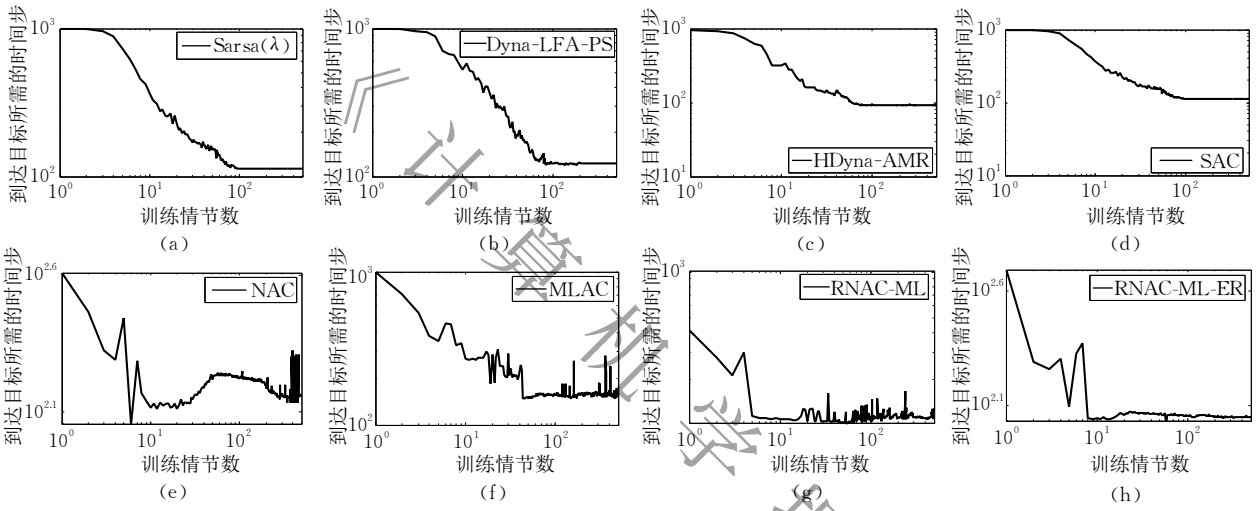


图 7 RNAC-ML 和 RNAC-ML-ER 与连续和离散动作算法的到达目标所需时间步比较

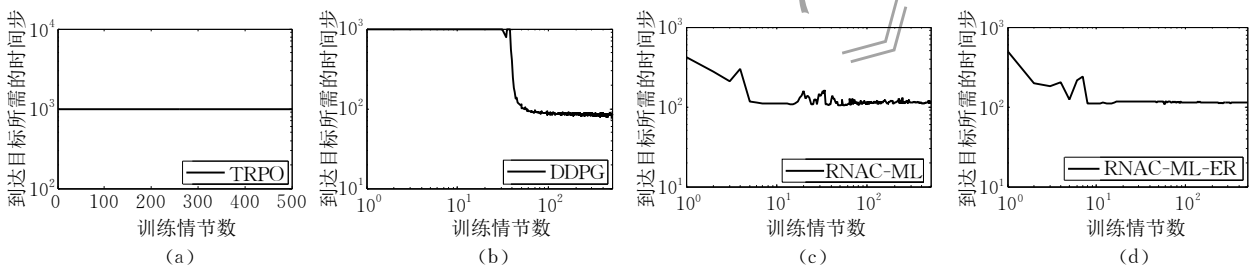


图 8 RNAC-ML 和 RNAC-ML-ER 与深度网络模型的到达目标所需时间步比较

表 2 不同算法在小车上山实验中的性能比较

算法名称	是否收敛(Y/N)	达到目标小于 120 时间步的首个情节	收敛所需总样本	收敛对应情节	收敛解	总训练时间/min
Sarsa(λ)	Y	71	19795	89	115	3.4
Dyna-LFA-PS	Y	—	24425	86	121	3.5
HDyna-AMR	Y	48	14640	72	92	2.3
SAC	Y	—	49688	256	160	0.4
NAC	Y	—	36821	225	148	0.5
MLAC	Y	—	12977	45	148	0.8
TRPO	Y	46	40755	58	93	17.2
DDPG	N	—	—	—	—	40.3
RNAC-ML	**Y**	**8**	**6056**	**44**	**115**	1.1
RNAC-ML-ER	**Y**	**8**	**1756**	**9**	**110**	**1.5**

度最快的是 RNAC-ML-ER，其它依次是 RNAC-ML 和 MLAC，分别在第 44 个情节和第 45 个情节收敛. 收敛解最优的是 HDyna-AMR 和 TRPO，其次是 RNAC-ML-ER 和 RNAC-ML. 在该实验中，仅有 DDPG 无法收敛和学习到一个近似的最优策略或局部最优策略. 样本效率最高的是 RNAC-ML-ER，其次是 RNAC-ML，收敛所需的样本分别为 1756 和 6056. 显然，RNAC-ML-ER 和 RNAC-ML 在样本效率上较其它算法具有明显优势. 与平衡杆实验相同，在此实验中，基于深度神经网络的方法所需的总训练时间远超于基于线性函数逼近器的方法.

6.3　倒立摆

倒立摆是一个富有挑战性的非线性控制问题，如图 9 所示.

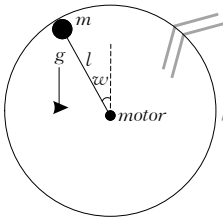


图 9　倒立摆示意图

倒立摆下面黑色小圆是一个可以施加 $[-3,3]$V 电压的电机. 倒立摆实验的目标就是通过施加电压使得倒立摆与竖直方向的角度尽可能地接近 0. 仅通过施加电压无法使得倒立摆到达竖直方向，因此需要通过倒立摆的左右摇摆使其获得足够的动量.

初始时刻倒立摆与竖直方向的夹角为 π. 系统的状态为二维向量$(\boldsymbol{a},\dot{\boldsymbol{a}})$，其中，$\boldsymbol{a}\in[-\pi,\pi]$为倒立摆与竖直方向的角度和 $\dot{\boldsymbol{a}}\in[-8\pi,8\pi]$为倒立摆运动的角速度. 动作为施加的电压 $u\in[-3,3]$ V. 该系统的动力学方程为

$$J\ddot{\boldsymbol{a}}=Mgl\sin(\boldsymbol{a})-\left(\boldsymbol{b}+\frac{K^2}{R}\right)\dot{\boldsymbol{a}}+\frac{K}{R}u \tag{42}$$

其中，$J=1.91\cdot10^{-4}\,\mathrm{kgm^2}$ 是惯性因子、$M=5.5\cdot10^{-2}\,\mathrm{kg}$ 为倒立摆的质量、$g=9.81\,\mathrm{m/s^2}$ 为重力加速度、$l=4.20\cdot10^{-2}\,\mathrm{m}$ 为倒立摆的长度、阻尼系数 $\boldsymbol{b}=3\cdot10^{-6}\,\mathrm{Nms}$、力矩常数 $K=5.36\cdot10^{-2}\,\mathrm{Nm/A}$ 和转子电阻 $R=9.5\Omega$.

奖赏函数表示为

$$r_k(\boldsymbol{x}_{k-1},u_{k-1})=-\boldsymbol{x}_{k-1}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_{k-1}-Pu_{k-1}^2 \tag{43}$$

其中，$\boldsymbol{Q}=[5,0;0,0.1]$，$P=1$.

状态特征中心点取自二维网格$\{-3,-2.5,-2.0,-1.5,-1,-0.5,0,0.5,1.0,1.5,2,2.5,3\}\times\{-25,-20,-15,-10,-5,0,5,10,15,20,25\}$，角度的方差为 0.06，角速度的方差为 5.21. 状态动作特征的中心点取自三维空间$\{-3,-2.5,-2.0,-1.5,-1,-0.5,0,0.5,1.0,1.5,2,2.5,3\}\times\{-25,-20,-15,-10,-5,0,5,10,15,20,25\}\times\{-2.8,-2.1,-1.4,-0.7,0,0.7,1.4,2.1,2.8\}$，动作的方差为 0.11. 整个仿真过程共运行 34 min，包含 500 个情节，每个情节运行 800 个时间步，每个时间步的仿真时间为 0.005 s.

将 RNAC-ML 方法和 RNAC-ML-ER 与离散动作空间方法即 Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR 以及连续动作算法 SAC、NAC 和 MLAC 进行比较. Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR 将动作空间$[-3,3]$ V 离散化为 −3 V、0 和 3 V. RNAC-ML 和 RNAC-ML-ER 中正则化惩罚因子 $\ell=0.03$. 8 种算法得到的折扣累积奖赏如图 10 所示. 从图 10 中可以看出，RNAC-ML(图 10(g))和

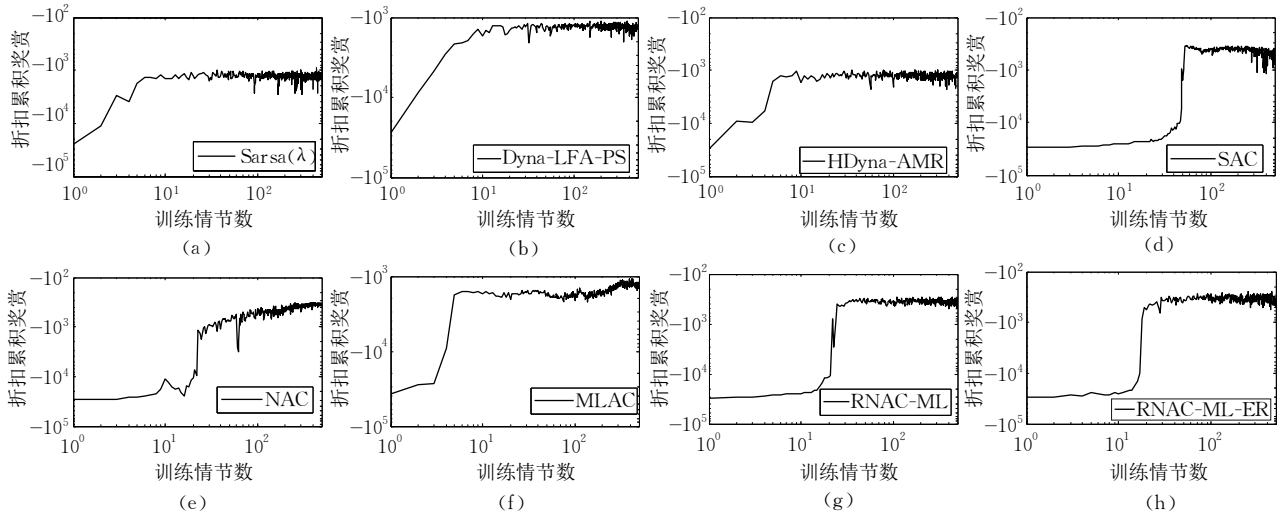


图 10　RNAC-ML 和 RNAC-ML-ER 与连续和离散动作算法的折扣累积奖赏比较

计算机学报

RNAC-ML-ER(图 10(f))得到的折扣累积奖赏最稳定.RNAC-ML-ER 和 RNAC-ML 分别在第 21 个情节和 25 个情节处趋于收敛,最终得到的累积奖赏分别约为－345.34 和－360.42.在 3 种离散动作算法 Sarsa(λ)(图 10(a))、Dyna-LFA-PS(图 10(b))和 HDyna-AMR(图 10(c))中,HDyna-AMR 震荡的幅度最小且表现最稳定.显然,RNAC-ML 和 RNAC-ML-ER 较 Sarsa(λ)(图 10(a))、Dyna-LFA-PS(图 10(b))和 HDyna-AMR(图 10(c))均具有更好的收敛性能.在连续算法 SAC(图 10(d))、NAC(图 10(e))和 MLAC)(图 10(f))中,初始学习率最快的是 MLAC 方法.MLAC 采用 LLR 近似值函数、策略和模型,虽然它在第 5 个情节后就趋于收敛,但收敛的累积奖赏值较小约为－1515.7,且其在 150 个情节后累积奖赏在[－1700,－1500]之间有轻微的震荡.NAC 方法在前 27 个情节中的学习率与 RNAC-ML 和 RNAC-ML-ER 相当,但在此之后一直在[－1000,－400]之间震荡,直到 212 个情节后趋于收敛,收敛的累积奖赏大约为－402.31.SAC 方法在前 53 个情节中学习率最慢,在第 53 个情节后累积奖赏能稳定在－401 附近,但在 299 个情节后又开始震荡.因此,SAC 方法无论从稳定性还是从学习率上都落后于其它 4 种方法.

倒立摆实验表明:基于线性函数逼近器和模型学习的方法即 RNAC-ML 和 RNAC-ML-ER 算法,在收敛效果上优于基于 LLR 近似的 MLAC 方法,同时也优于未使用模型学习的方法 SAC 和 NAC.

由于离散动作算法的性能与动作空间的离散粒度关系密切,因此,以 Dyna-LFA-PS 为例,将动作空间分别划分为 3 个动作(－3,0,3)、5 个动作(－3,－1.5,0,1.5,3)、7 个动作(－3,－2,－1,0,1,2,3)和 9 个动作(－3,－2.25,－1.5,－0.75,0,0.75,1.5,2.25,3),然后将它们与 RNAC-ML 和 RNAC-ML-ER 算法进行比较,最终得到的累积奖赏如图 11 所示.从图 11 中可以看出,Dyna-LFS-PS 除了 3 个动作对应的情况无法探索到最优策略外,其它情况均能探索到与 RNAC-ML 和 RNAC-ML-ER 相同的近似最优解.然而,这些离散化方法在最优解附近仍剧烈震荡,这可能是因为离散化虽然提高了值函数精度,但是仍不具备泛化能力,所以遇到一些没有经历过的状态时会发生剧烈的震荡.相比之下,RNAC-ML 和 RNAC-ML-ER 的表现更为稳定.

将 8 种不同算法(离散方法的动作空间离散为 3)在是否收敛、收敛对应的累积奖赏、累积奖赏首次大于－400 所需样本、收敛所需总样本以及收敛对应的情节等性能指标上进行比较,如表 3 所示.

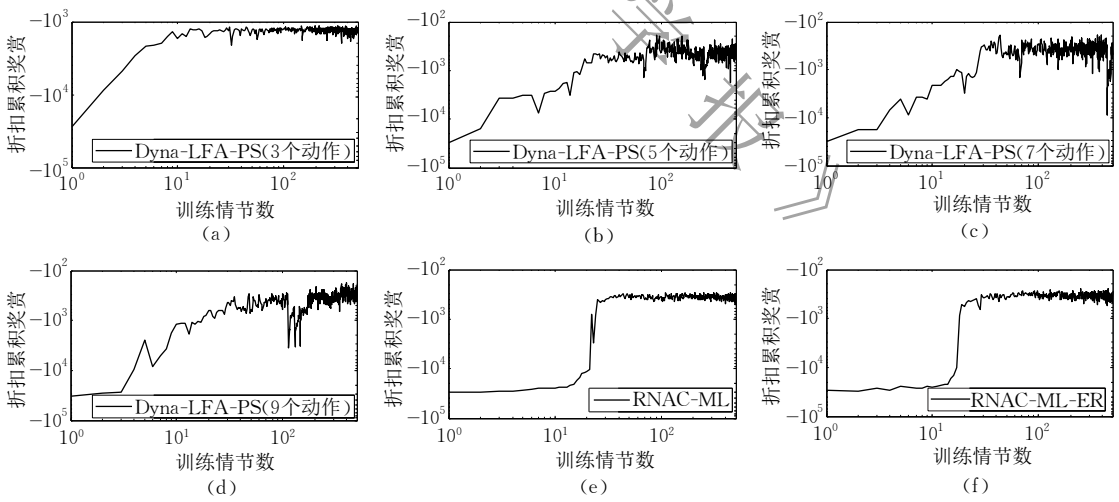


图 11 RNAC-ML 和 RNAC-ML-ER 与不同粒度离散算法的折扣累积奖赏比较

表 3 不同算法在倒立摆问题中的性能比较

算法名称	是否收敛(Y/N)	收敛对应的累积奖赏	累积奖赏首次大于－400 所需样本	收敛所需总样本	收敛时对应的情节
Sarsa(λ)	Y	－1200.97	—	21600	17
Dyna-LFA-PS	Y	－1256.72	—	47200	59
HDyna-AMR	Y	－1089.45	—	18400	23
SAC	Y	－454.49	43200	259200	324
NAC	Y	－402.31	120000	169600	213
MLAC	Y	－1515.70	—	4000	5
RNAC-ML	**Y**	**－360.42**	**20000**	**20000**	**25**
RNAC-ML-ER	**Y**	**－345.34**	**16800**	**16800**	**21**

从表 3 可以看出，收敛速度最快且所需样本最少的是 MLAC，但收敛到了局部最优解 −1515.7. RNAC-ML-ER 和 HDyna-AMR 样本效率次之，它们的折扣累积奖赏分别为 −345.34 和 −1089.45. 表 3 也表明了连续动作空间中基于模型学习的方法比未采用模型的算法具有更高的样本效率.

6.4　体操机器人

体操机器人是一个较倒立摆问题更复杂的控制问题，如图 12 所示. 体操机器人是一个在垂直平面运动的双链（OA 和 AB）倒立摆，但仅仅只在连接处 A 有一个电机，而 O 是一个无法施加力的固定点. 控制目标就是通过摆动体操机器人并通过施加 $[-1,1]$之间的力矩，使得点 B 能从垂直向下的位置超过目标点 D. 系统状态是一个四维向量 $(\boldsymbol{a},\dot{\boldsymbol{a}},\boldsymbol{b},\dot{\boldsymbol{b}})$，其中 $\boldsymbol{a}\in[-\pi,\pi]$和 $\dot{\boldsymbol{a}}\in[-4\pi,4\pi]$分别表示链 OA 的角度和角速度，$\boldsymbol{b}\in[-\pi,\pi]$和 $\dot{\boldsymbol{b}}\in[-9\pi,9\pi]$分别表示链 AB 的角度和角速度，动作 $u\in[-1,1]$.

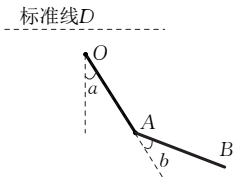


图 12　体操机器人示意图

立即奖赏的设置为：当 B 点超过 D 点时得到一个值为 1 的立即奖赏，其它情况下的每个时间步的立即奖赏均为 −1. 每个情节开始时初始化状态为 $(\boldsymbol{a},\dot{\boldsymbol{a}},\boldsymbol{b},\dot{\boldsymbol{b}})=(0,0,0,0)$. 情节结束当且仅当目标状态已经到达或者是当前时间步为情节允许的最大时间步. 状态特征的中心点取自四维空间 $\{-3,-1.5,0,1.5,3\}\times\{-12,-6,0,6,12\}\times\{-3,-1.5,0,1.5,3\}\times\{-26,-13,0,13,26\}$，状态在 4 个维度上的方差分别为 0.39、6.31、0.39 和 25.24. 状态动作特征的中心点来自五维空间 $\{-3,-1.5,0,1.5,3\}\times\{-12,-6,0,6,12\}\times\{-3,-1.5,0,1.5,3\}\times\{-26,-13,0,13,26\}\times\{-0.5,0,0.5\}$，动作的方差为 0.11. 算法共运行 400 个情节，每个情节的最大时间步为 3000，每个时间步的仿真时间为 0.05 s.

体操机器人的系统动态性方程可以描述为

$$\begin{cases}\ddot{\boldsymbol{a}}=-(d_2\ddot{\boldsymbol{b}}+\varphi_1)/d_1\\ \ddot{\boldsymbol{b}}=(\tau+d_2\varphi_1/d_1-\varphi_2)\end{cases}\tag{44}$$

其中：

$$d_1=m_1l_{c1}^2+m_2(l_1^2+l_{c2}^2+2l_1l_{c2}\cos\boldsymbol{b})+I_1+I_2,$$

$$d_2=m_2(l_{c2}^2+l_1l_{c2}\cos\boldsymbol{b})+I_2,$$

$$\varphi_1=-m_2l_1l_{c2}\dot{\boldsymbol{b}}^2\sin\boldsymbol{b}-2m_2l_1l_{c2}\dot{\boldsymbol{a}}\dot{\boldsymbol{b}}\sin\boldsymbol{b}+(m_1l_{c1}+m_2l_1)g\cos(a-\pi/2)+\varphi_2,$$

$$\varphi_2=m_2l_{c2}g\cos(a+\boldsymbol{b}-\pi/2),$$

其中，$\ddot{\boldsymbol{a}}$ 和 $\ddot{\boldsymbol{b}}$ 分别为链 OA 和链 AB 的角加速度. m_i、l_i、I_i和 l_{ci} 分别表示链 OA（$i=1$）和 AB（$i=2$）的质量、长度和惯性矩以及质心的长度. 部分参数设置为：$m_1=m_2=1\,\mathrm{kg}$，$l_{c1}=l_{c2}=0.5\,\mathrm{m}$，$l_1=l_2=1\,\mathrm{m}$，$g=9.8\,\mathrm{m/s^2}$，$D$ 与 O 点的垂直距离设置为 0.1 m.

将 RNAC-ML 和 RNAC-ML-ER 与离散动作算法 Sarsa(λ)（图 13(a)）、Dyna-LFA-PS（图 13(b)）和 HDyna-AMR（图 13(c)）以及连续动作算法 SAC（图 13(d)）、NAC（图 13(e)）和 MLAC（图 13(f)）进行比较. 离散动作算法 Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR 将动作空间离散化为 −1、0 和 1. RNAC-ML 和 RNAC-ML-ER 中正则化惩罚因子 $\ell=0.09$. 随着训练情节数的增加，各算法到达目标所需要的时间步如图 13 所示. 从图 13 中可以看出 RNAC-ML 和 RNAC-ML-ER 在体操机器人上表现相差无几，但在前 17 个情节中 RNAC-ML-ER 比 RNAC-ML 学习的更快. 总体而言，RNAC-ML-ER 震荡幅度更小. 两种算法最终得到的到达目标所需时间步约为 55～62. RNAC-ML 和 RNAC-ML-ER 算法到达目标所需的时间步远远低于离散算法 Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR. 3 种离散方法中表现最好的是 HDyna-AMR，其不仅稳定度较高且收敛时到达目标所需时间步约 103～115. 相对 Sarsa(λ)而言，Dyna-LFA-PS 震荡幅度较小，在排除一些振幅较大的点后，两种算法达到目标所需时间步约为 120～150. Dyna-LFA-PS 和 HDyna-AMR 算法的稳定性较 Sarsa(λ)更好，进一步说明了通过模型学习与规划可以较大地改善算法的稳定性. 3 种连续方法中震荡幅度最大的是 MLAC，其次是 SAC. MLAC 由于引入 LLR 局部模型，因此在前 13 个情节中学习速率比 SAC 和 NAC 快，但随着情节数的增加，MLAC 不仅不能得到一个较好的解反而一直大幅地震荡. 这是由于 MLAC 采用 LLR 作为函数近似器，难以在四维状态空间中精确地近似模型. 当采用不精确的模型来更新策略梯度时，反而会使得算法难以收敛到较好的解. NAC 算法震荡相对较小，但仍然比 RNAC-ML 和 RNAC-ML-ER 要大，且到达目标所需要的时间步大约为 105～128.

《计算机学报》

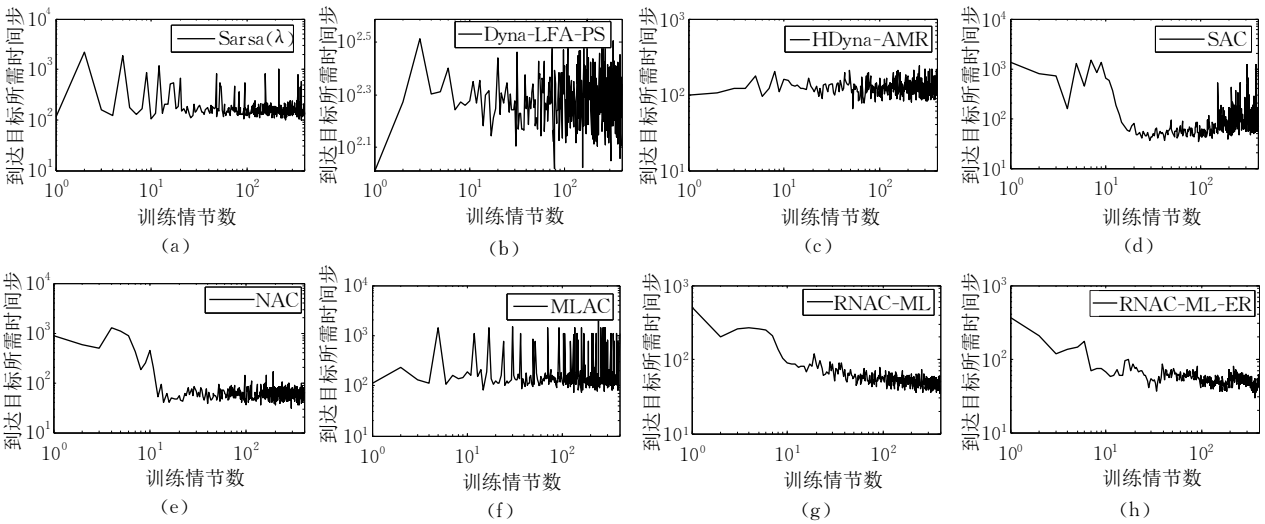


图 13 RNAC-ML 和 RNAC-ML-ER 与连续和离散动作算法的到达目标所需时间步比较

为了对样本效率进行验证,以相邻的 20 轮迭代中学习曲线振幅小于 15%作为收敛的条件,将 8 种不同算法在是否收敛、到达目标所需时间步小于 100 的首个情节、收敛所需总样本以及收敛时对应的情节进行比较,如表 4 所示. 从表 4 中可以发现,8 种算法中最终能收敛的只有 HDyna-AMR、NAC、RNAC-ML 和 RNAC-ML-ER. 样本效率高的算法为模型相关的算法 HDyna-AMR、RNAC-ML 和 RNAC-ML-ER. 虽然 NAC 算法能够收敛,但收敛时到达目标所需的时间步最小值仍大于 100,同时它的样本效率较低. 其它 4 种算法 Sarsa(λ)、Dyna-LFA-PS、SAC 和 MLAC 的震荡幅度较大而未能收敛. 显然,RNAC-ML-ER 具有最好的样本效率.

表 4 不同算法在体操机器人问题中的性能比较

算法名称	是否收敛(Y/N)	到达目标所需时间步少于 100 的首个情节	收敛所需总样本数	收敛时对应的情节
Sarsa(λ)	N	—	—	—
Dyna-LFA-PS	N	—	—	—
HDyna-AMR	Y	23	3576	28
SAC	N	15	—	—
NAC	Y	15	19526	77
MLAC	N	12	—	—
RNAC-ML	**Y**	**11**	**3122**	**20**
RNAC-ML-ER	**Y**	**7**	**2085**	**19**

7 结 论

连续动作空间的最优策略求解是 RL 的一个重要研究方向,尤其是在样本难以获取的实时系统中快速地学习最优策略并保持较高的样本效率是一个极富挑战性的难题. 针对该问题,本文提出了基于模型学习和经验回放加速的正则化自然 AC 算法——RNAC-ML-ER 算法. RNAC-ML-ER 算法在 AC 算法的基础上通过改进自然梯度并对其进行正则化来提高策略更新效率和降低策略方差. 为了提高样本效率,采用在线学习得到的真实样本既用于学习近似模型进行规划又用于经验回放.

将 RNAC-ML-ER 应用于 4 个 RL 基准实验中,结果表明 RNAC-ML-ER 在收敛速度和样本效率上普遍高于离散动作空间的方法如 Sarsa(λ)、Dyna-LFA-PS 和 HDyna-AMR,同时也高于连续动作空间的方法如 SAC、NAC、MLAC. 相对于基于深度神经网络的方法 TRPO 和 DDPG,RNAC-ML-ER 和 RNAC-ML 具有稳定性强和样本效率高的优点. TRPO 和 DDPG 在不同的问题域中表现不同,在平衡杆实验中 DDPG 表现更优;而在小车上山实验中 TRPO 表现更优,其不仅能有效收敛,而且收敛到了一个较好的解. 同时实验结果也表明:采用模型学习的算法的样本效率普遍高于未采用模型的同类算法,如 Dyna-LFA-PS 和 HDyna-AMR 高于 Sarsa(λ),RNAC-ML 和 RNAC-ML-ER 高于 SAC 和 NAC. MLAC 作为一种基于 LLR 的连续动作方法,在倒立摆和体操机器人实验中效果均不如模型无关的连续动作算法 SAC 和 NAC. 这是因为 MLAC 学习了一个不够精确局部模型并用于更新策略梯度,当局部模型出现较大误差时容易引起策略的剧烈波动. 在平衡杆和倒立摆实验中,RNAC-ML-ER 在收敛速度和样本效率上都较 RNAC-ML

有大幅提升，而在体操机器人实验中的性能提升却非常有限.由于体操机器人实验中的数据集存储的大量样本都是离目标状态较远的样本，这些样本的TD-error几乎为0，使得值函数的大量更新都是无效的，因而经验回放带来的改进有限.

本文下一步的工作主要从两个方面考虑：(1)为了对模型进行更为精确地近似，对状态空间和动作空间分段来分别近似基于状态分段和动作分段的模型.基于状态分段的模型直接用于规划，而基于动作分段的模型通过结合优先级队列来更新值函数；(2)在经验回放存储器中加入一些目标附近的先验样本来提高样本的均衡性，以促进算法的更快收敛；(3)建立基于深度神经网络逼近的模型近似方法，并与行动器评判器算法结合应用于连续状态动作空间.

参　考　文　献

[1] Puterman M L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. New Jersey, USA: John Wiley & Sons Inc., 2014

[2] Guo X X, Singh S, Lee H, et al. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning//Proceedings of the Conference on Neural Information Processing Systems (NIPS 2014). Montréal, Canada, 2014: 3338-3346

[3] van Seijen H, Sutton R S. Efficient planning in MDPs by small backups//Proceedings of the 30th International Conference on Machine Learning (ICML 2013). Atlanta, USA, 2013: 361-369

[4] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529-533

[5] Caicedo J C, Lazebnik S. Active object localization with deep reinforcement learning//Proceedings of the IEEE International Conference on Computer Vision (ICCV 2015). Santiago, Chile, 2015: 2488-2496

[6] He H, Boyd-Graber J, Kwok K, Daumé III H. Opponent modeling in deep reinforcement learning//Proceedings of the 33rd International Conference on Machine Learning (ICML 2016). New York, USA, 2016: 1804-1813

[7] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning//Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016). Phoenix, USA, 2016: 2094-2100

[8] Fu Qi-Ming, Liu Quan, Wang Hui, et al. A novel off $Q(\lambda)$ algorithm based on linear function approximation. Chinese Journal of Computers, 2014, 37(3): 677-686(in Chinese)
(傅启明，刘全，王辉等. 一种基于线性函数逼近的离策略$Q(\lambda)$算法. 计算机学报，2014，37(3)：677-686)

[9] Sutton R S, McAllester D A, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation//Proceedings of the Conference on Neural Information Processing Systems (NIPS 1999). Denver, USA, 1999: 1057-1063

[10] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms//Proceedings of the 31st International Conference on Machine Learning (ICML 2014). Beijing, China, 2014: 387-395

[11] Barto A G, Sutton R S, Anderson C W. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transaction on System, Man, Cybernetics, 1983, 5(SMC-13): 834-846

[12] Grondman I, Busoniu L, Lopes G A D, Babuska R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transaction on Systems, Man, and Cybernetics, 2012, 42(6): 1291-1307

[13] Ghavamzadeh M, Engel Y, Valko M. Bayesian policy gradient and actor-critic algorithms. Journal of Machine Learning Research, 2016, 17(66): 1-53

[14] Hasselt H V, Mahmood A R, Sutton R S. Off-policy TD(λ) with a true online equivalence//Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI 2014). Quebec, Canada, 2014

[15] Seijen H V, Sutton R. True online TD(λ)//Proceedings of the International Conference on Machine Learning (ICML 2014). Beijing, China, 2014: 692-700

[16] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Massachusset, USA: MIT Press, 1998

[17] Watkins C, Dayan P. Q-learning. Machine Learning, 1992, 3-4(8): 279-292

[18] Boyan J A. Technical update: Least-square temporal difference learning. Machine learning, 2002, 2-3(49): 233-246

[19] Tagorti M, Scherer B. On the rate of the convergence and error bounds for LSTD(λ)//Proceedings of the International Conference on Machine Learning (ICML 2015). Lille, France, 2015: 528-536

[20] Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search//Proceedings of the 5th International Conference on Computers and Games. Turin, Italy, 2006: 72-83

[21] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning//Proceedings of the Neural Information Processing Systems Deep Learning Workshop (NIPS 2013). Nevada, USA, 2013: 201-220

[22] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning//Proceedings of the International Conference on Machine Learning (ICML 2016). New York, USA, 2016: 1928-1937

《计算机学报》

[23] Schulman J, Levine S, Moritz P, et al. Trust region policy optimization//Proceedings of the International Conference on Machine Learning (ICML 2015). Lille, France, 2015: 1889-1897

[24] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning//Proceedings of the International Conference on Learning Representation (ICLR 2016). San Juan, Puerto Rico, 2016

[25] Lin L J. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning, 1992, 8(3-4): 293-321

[26] Adam S, Busoniu L, Babuska R. Experience replay for real-time reinforcement learning control. IEEE Transaction on Systems, Man, and Cybernetics, 2012, 42(2): 201-212

[27] Amari S I. Natural gradient works efficiently in learning. Neural Computation, 1998, 10(2): 251-276

[28] Bagnell J, Schneider J. Covariant policy search//Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2003). Acapulco, Mexico, 2003: 1019-1024

[29] Peters J, Vijayakumar S, Schaal S. Reinforcement learning for humanoid robotics//Proceedings of the 3rd IEEE-RAS International Conference on Humanoid Robots. Karlsruhe, Germany, 2003: 1-20

[30] Grosse R, Salakhudinov R. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. Journal of Machine Learning Research, 2015, 37: 2304-2313

[31] Thomas P S, Dabney W C, Giguere S, Mahadevan S. Projected natural actor-critic//Proceedings of the Conference on Neural Information Processing Systems (NIPS 2013). Nevada, USA, 2013: 2337-2345

[32] Sutton R S. Integrated architecture for learning, planning and reacting based on approximating dynamic programming//Proceedings of the International Conference on Machine Learning (ICML1990). Texas, USA, 1990: 216-224

[33] Peng J, Williams R J. Efficient learning and planning within the Dyna framework. Adaptive Behavior, 1993, 4(1): 437-454

[34] Moore A W, Atkeson C G. Prioritized sweeping: Reinforcement learning with less data and less real time. Machine Learning, 1993, 1(13): 103-130

[35] Santos M, Lopez V, Botella G. Dyna-H: A heuristic planning reinforcement learning algorithm applied to role-playing game strategy decision systems, Knowledge-Based Systems, 2012, 1(32): 28-36

[36] Sutton R S, Szepesvári C, Geramfard A, Bowling M. Dyna-style planning with linear function approximation and prioritized sweeping.//Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI 2008). Helsinki, Finland, 2008: 528-536

[37] Zhong Shan, Liu Quan, Fu Qi-Ming, Zhang Zong-Zhang, et al. A heuristic Dyna optimizing algorithm using approximation model representation. Journal of Computer Research and Development, 2015, 52(12): 2764-2775 (in Chinese)
(钟珊, 刘全, 傅启明等. 一种近似模型表示的启发式 Dyna 优化算法. 计算机研究与发展, 2015, 52(12): 2764-2775)

[38] Grondman I, Vaandrager M, Busoniu L, Schuitema E. Efficient model learning methods for actor-critic control systems. IEEE Transaction on Man, and Cybernetics, 2012, 42(3): 591-602

[39] Busoniu L, Babuška R, Schutter B D, Ernst D. Reinforcement Learning and Dynamic Programming Using Function Approximators. New York, USA: CRC Press, 2010

[40] Grondman I, Busoniu L, Babuska R. Model learning actor-critic algorithms: Performance evaluation in a motion control task//Proceedings of the 51st IEEE Annual Conference on Decision and Control (CDC 2012). Hawaii, USA, 2012: 5272-5277

[41] Costa B, Caarls W, Menasche D S. Dyna-MLAC: Trading computational and sample complexities in actor-critic reinforcement learning//Proceedings of the 2015 Brazilian Conference on Intelligent Systems. Natal, Brazil, 2015: 37-42

[42] Cheng Y H, Huang F, Wang X S. Efficient data use in incremental actor-critic algorithms. Neurocomputing, 2013, 116: 346-354

[43] Gu S, Lillicrap T, Sutskever I, Levine S. Continuous deep Q-learning with model-based acceleration//Proceedings of the 33rd International Conference on Machine Learning (ICML 2016). New York, USA, 2016: 2829-2838

[44] Tamar A, Wu Y, Thomas G, et al. Value iteration networks //Proceedings of the Conference on Neural Information Processing Systems (NIPS 2016). Barcelona, Spain, 2016: 2154-2162

[45] Bhatnagar S, Sutton R S, Ghavamzadeh M, Lee M. Natural actor-critic algorithm. Automatica, 2009, 45(11): 2471-2482

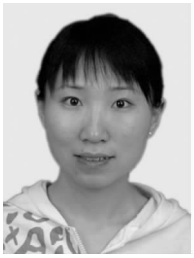
[46] Peters J, Schaal S. Natural actor-critic. Neurocomputing, 2008, 71(7): 1180-1190

[47] Kushner H J, Clark D S. Stochastic Approximation Algorithms and Applications. New York, USA: Springer-Verlag, 1997

[48] Borkar V S, Meyn S P. The O. D. E. method for convergence of stochastic approximation and reinforcement learning. SIAM Journal of Control and Optimization, 2000, 38(2): 447-469

[49] Duan Y, Chen X, Houthooft R, et al. Benchmarking deep reinforcement learning for continuous control//Proceedings of the 33rd International Conference on Machine Learning (ICML 2016). New York, USA, 2016: 1329-1338

《计算机学报》



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

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

FU Qi-Ming, born in 1985. Ph. D. His main research interests include reinforcement learning, Bayesian method and genetic algorithm.

GONG Sheng-Rong, born in 1966, Ph. D., professor, Ph. D. supervisor. His main research interests include machine learning and image processing.

DONG Hu-Sheng, born in 1981, Ph. D. candidate, lecturer. His main research interests include machine learning and image processing.

Background

Reinforcement learning aims at finding an optimal policy, mapping from states to actions, via maximizing the cumulative numerical rewards. As a class of important model-free reinforcement learning (RL) methods, the actor-critic (AC) algorithm and its variants have weakness in the efficient use of samples. This paper proposes a novel method, called RNAC-ML-ER, to improve AC using real samples, simulated samples and stored samples. These samples work together to learn the value function, the advantageous function and the policy, so that the convergence rate and the sample efficiency can be enhanced significantly. The simulated samples are generated from the learned model while the stored sample are fetched from the memory of the experience replay. Simulation evaluation illustrates that RNAC-ML-ER behaves effectively, with rapid converge rate and satisfactory stability.

This paper is supported by the National Natural Science Foundation of China (61772355, 61702055, 61303108, 61373094, 61472262, 61502323, 61502329), the Natural Science Foundation of Jiangsu (BK2012616), the Science Technology Program of Jiangsu (BK2015260), the High School Natural Foundation of Jiangsu (13KJB520020), the General Project of University Natural Science Research of Jiangsu Province (16KJD520001), the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (93K172014K04, 93K172017K18), the Suzhou Industrial Application of Basic Research Program Part (SYG201422, SYG201308).

《计算机学报》