带最大熵修正的行动者评论家算法

姜玉斌[1] 刘 全[1),2),3),4)] 胡智慧[1]

[1](苏州大学计算机科学与技术学院 江苏 苏州 215006)

[2](苏州大学江苏省计算机信息处理技术重点实验室 江苏 苏州 215006)

[3](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](软件新技术与产业化协同创新中心 南京 210000)

摘 要 在行动者评论家算法中,策略梯度通常使用最大熵正则项来提高行动策略的随机性以保证探索.策略的随机使 Agent 能够遍历所有动作,但是会造成值函数的低估并影响算法的收敛速度与稳定性.针对策略梯度中最大熵正则项带来的低估问题,提出最大熵修正(Maximum-Entropy Correction,MEC)算法.该算法有两个特点:(1)利用状态值函数与策略函数构造一种状态动作值函数的估计,构造的状态动作值函数符合真实值函数的分布;(2)将贝尔曼最优方程与构造的状态动作值函数结合作为 MEC 算法的目标函数.通过使用新的目标函数,MEC 算法可以解决使用最大熵正则项带来的性能下降与不稳定.为了验证算法的有效性,将该算法与近似策略优化算法以及优势行动者评论家算法在 Atari 2600 游戏平台进行比较实验.实验结果表明,MEC 在改进性能的同时提高了算法的稳定性.

关键词 强化学习;深度学习;行动者评论家算法;最大熵;策略梯度

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2020.01897

Actor-Critic Algorithm with Maximum-Entropy Correction

JIANG Yu-Bin[1] LIU Quan[1),2),3),4)] HU Zhi-Hui[1]

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

[2](*Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou, Jiangsu* 215006)

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

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

Abstract In recent years, Deep Reinforcement Learning (DRL) combines deep learning with reinforcement learning, and has become one of the research hotspots in the field of artificial intelligence. Deep learning combines intensive learning with amazing results in robot control, Atari 2600 games and more. The first algorithm that combines reinforcement learning with deep learning is TD-Gammon, which surpasses professional chess players in the problem of backgammon. Then there is a series of algorithms from Deep Q-Network (DQN), which has achieved extraordinary performance in Atari 2600 games. But limited by the value function method, DQN cannot be applied to continuous motion tasks. Therefore, the Actor-Critic (AC) algorithm is widely used in DRL because of its excellent scalability. It consists of two parts: the actor and the critic. Usually, the critic uses value function methods to evaluate the action, and the actor uses policy gradient methods for the action generation. This means that the AC method can be applied to continuous

收稿日期:2018-11-28;在线发布日期:2019-11-04.本课题得到国家自然科学基金项目(61772355,61702055,61472262,61502323,61502329)、江苏省高等学校自然科学研究重大项目(18KJA520011,17KJA520004)、吉林大学符号计算与知识工程教育部重点实验室资助项目(93K172014K04,93K172017K18)、苏州市应用基础研究计划工业部分(SYG201422)、江苏高校优势学科建设工程资助项目资助.**姜玉斌**,硕士研究生,主要研究方向为强化学习、深度强化学习.E-mail:562058113@qq.com.**刘 全**(通信作者),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.**胡智慧**,硕士研究生,主要研究方向为强化学习、深度强化学习.

motion and discrete motion tasks. Researchers have proposed many AC algorithms, such as Asynchronous Advantage Actor-Critic (A3C) using asynchronous updates, Advantage Actor-Critic (A2C) using multi-agent updates, and Proximal Policy Optimization (PPO) with guaranteed policy improvement. One of the core challenges in reinforcement learning (RL) is how to balance the exploration and exploitation. In order to ensure a high return, the algorithm needs to find those actions that expect high returns in the experience explored in the past. The Immediate rewards for certain actions are high but their expected return is low, and the Agent needs to explore all untraversed actions to prevent the policy from entering local optimum. In other words, the Agent must use the previous experience to obtain higher expectations, and on the other hand must explore to find better action options. In the policy gradient, the maximum entropy regularity term is usually used to increase the randomness of the policy to ensure exploration. The randomness of the policy enables the Agent to traverse all the actions but it will cause the underestimation of the value function and affect the convergence speed and stability of the algorithm. In order to solve the underestimation problem caused by the maximum entropy regular term in the policy gradient, the Maximum-Entropy Correction (MEC) algorithm is proposed. The algorithm has two characteristics: (1) constructing a state action value function estimation using state value function and policy function, the constructed state action value function satisfies the distribution of the real value function; (2) The Berman optimal equation is combined with the constructed state action value function as the objective function of the MEC algorithm. The performance degradation and instability caused delete u the maximum entropy regular term can be solved by using the new objective function MEC algorithm. In order to verify the effectiveness of the algorithm, the algorithm was compared with PPO and A2C on the seven Atari 2600 games: BeamRide, Breakout, Enduro, Pong, Qbert, Seaquest and SpaceInvaders. Experimental results show that MEC improves the stability of the algorithm while improving performance.

Keywords reinforcement learning; deep learning; actor-critic algorithm; maximum entropy; policy gradient

1 引 言

近年来,深度强化学习[1](Deep Reinforcement Learning,DRL)将深度学习[2]与强化学习相结合,成为人工智能领域的研究热点之一.深度强化学习可以追溯到 TD-Gammon[3],该算法首次将神经网络与 TD(λ)[4]结合,并在西洋双陆棋问题上取得了超过专家的表现.DeepMind 公司的 Mnih 等人[5]提出了深度 Q 网络(Deep Q-Network,DQN),通过输入原始图像,使用卷积神经网络(Convolutional Neural Network,CNN)提取图像特征,利用提取特征构建 Q 值网络以计算与更新值函数.经过训练,DQN 在 Atari 2600 平台中大部分游戏的表现超过了人类水平.在这个基础上,研究者们提出了许多改进算法,例如优先级经验重放[6]、竞争深度 Q 网络[7](Dueling-DQN)、双层深度 Q 网络(Double-DQN)[8]等.

DQN 的一系列算法在 Atari 2600 游戏上有着出色的表现,但是由于值函数网络的限定,DQN 不能运用到连续动作空间的任务当中.研究者们使用行动者-评论家(Actor-Critic,AC)算法来解决这一问题.行动者-评论家算法通过行动策略与策略评估分离的方式进行学习,行动策略通常使用策略梯度方法,能够应用于连续动作或者离散动作任务中.Lillicrap 等人[9]在确定性策略梯度[10](Deterministic Policy Gradient,DPG)算法的基础上提出了深度确定性策略梯度(Deep Deterministic Policy Gradient,DDPG)算法,通过具有确定性策略的行动者和状态动作值函数结合深度神经网络,学习大规模连续动作空间任务.Wang 等人[11]利用重要性权重将 AC 与

经验重放结合，提出了(Actor-Critic with Experience Replay，ACER)算法，提高了算法的样本利用率.但是使用经验重放生成样本会显著增加存储空间并提高计算复杂度.因此 Mnih 等人[12]提出了异步优势行动者评论家(Asynchronous Advantage Actor-Critic，A3C)算法，将异步方法应用于深度强化学习中，在脱离经验重放机制的情况下在 Atari 2600 游戏上进行训练并获得了更好的表现. Wu 等人[13]指出可以使用多组 Agent 进行交互替代异步方法的优势行动者评论家算法(Advantage Actor-Critic，A2C)能够达到 A3C 同样的效果，并且可以更好地利用 GPU 的加速效果，并在此基础上提出了(Actor Critic using Kronecker-factored Trust Region，ACKTR)算法，利用分布式 Kronecker 因子分解提高算法的样本效率和可扩展性. Schulman 等人[14]提出置信域策略优化(Trust Region Policy Optimization，TRPO)算法，证明了该算法在一般随机梯度下改进策略的单调性. Schulman 等人[15]又将 TRPO 进一步简化，提出了近似策略优化(Proximal Policy Optimization，PPO)，使得计算复杂度大大降低，可以应用于大规模的问题中. Fujita 等人[16]提出 CAPG(Clipped Action Policy Gradient)，通过构造新的策略函数解决了高斯分布边界动作溢出的问题，降低了值函数的方差，提高了算法的稳定性.

在强化学习中，核心挑战之一是如何在探索与利用之间进行平衡[17].为了保证获得高回报，算法需要在过去探索到的经验中找到那些期望回报高的动作[18].某些动作的即时奖赏高但是期望奖赏低，Agent 需要探索所有未遍历的动作以防止策略进入局部最优.也就是说，Agent 一方面要利用之前的经验获得更高的期望奖赏，另一方面必须进行探索以寻找更好的动作选择.尽管研究者在探索与利用的平衡问题上努力了数十年，却仍然没有一个明确的解决方案.在以值函数为基础的算法中，通常使用例如 ε-greedy 的软贪心算法来保证策略的探索.在行动者评论家算法中，策略梯度通常采用最大熵正则项来防止策略变成确定性策略.这些方式虽然保证了策略的探索性，但生成的策略倾向于随机，会造成值函数的低估，可能导致性能的损失.

为了解决这一问题，文本提出一种最大熵修正算法，用状态值函数和策略函数设计一种状态动作值函数的估计，使用构造的状态动作值函数通过贝尔曼最优方程构造新的目标函数.新的目标函数可以提高算法的期望回报和算法的收敛速度.为了验证 MEC 的稳定性和有效性，在 Atari 2600 游戏平台的 7 款游戏中进行比较实验.实验结果表明，本文提出的 MEC 算法能够弥补最大熵正则项带来的性能下降并提高算法稳定性.

本文第 2 节介绍行动者评论家算法以及最大熵正则项；第 3 节阐述使用的网络模型、策略优化方法以及最大熵修正项的构造；第 4 节通过对比实验验证算法的有效性并分析算法的优点以及存在的问题；第 5 节进行总结以及对未来研究方向的展望.

2　相关工作

2.1　强化学习

强化学习不同于监督学习与非监督学习，其利用 Agent 与环境进行交互获得的反馈信息进行自我学习.强化学习依赖于马尔科夫决策过程[19](Markov Decision Process，MDP).本文使用一个无限有折扣的 MDP，它由五部分组成(S,A,P,R,γ)，S 是所有状态的集合，A 是所有动作的集合，$P:S\times A\times S$ 是状态转移概率，$P(s_t,a_t,s_{t+1})$表示在状态 s_t 作出动作 a_t 到达下一状态 s_{t+1} 的概率.R 表示奖赏函数，$\gamma\in(0,1)$是折扣因子，表示对未来奖赏的重视程度，γ 越大代表越重视长期回报.

在强化学习中，Agent 在当前状态 $s_t\in S$ 根据策略 π 选择动作 $a_t\in A$ 并与环境交互获得奖赏 $r(s_t)\in R$ 和下一状态 s_{t+1}.Agent 到达下一状态 s_{t+1} 并重复上述步骤，则可获得当前轨迹的累计折扣奖赏 $F_t=\sum_{k=0}^{\infty}\gamma^k r(s_{t+k})$，定义状态值函数 $V_\pi(s)$为状态 s 在策略 π 下获得的累计折扣奖赏的期望：

$$V_\pi(s)=\mathbb{E}[F_t\mid s_t=s] \tag{1}$$

$V_\pi(s)$的大小能够衡量在当前策略 π 下不同状态的优劣，但是在状态转移概率未知的情况下很难从 $V_\pi(s)$中得到更好的策略，因此通过定义状态动作值函数 $Q_\pi(s,a)$来确定动作的好坏，以获得新的策略：

$$\begin{aligned}Q_\pi(s,a)&=\mathbb{E}[r(s_t)+\gamma F_{t+1}\mid s_t=s,a_t=a]\\&=\mathbb{E}[r(s_t)+\gamma V_\pi(s_{t+1})\mid s_t=s,a_t=a]\end{aligned} \tag{2}$$

强化学习的目标是找到一个最优策略 π^*，使得期望奖赏最大，那么 π^* 的状态值函数一定满足等式 $V_{\pi^*}(s)=\max_a Q(s,a)$，由此得到状态动作值函数的贝尔曼最优方程：

$$Q_\pi(s,a)=\mathbb{E}\left[r(s_t)+\gamma\max_{a^*}Q_\pi(s_{t+1},a^*)\mid s_t=s,a_t=a\right] \tag{3}$$

利用贝尔曼最优方程对值函数进行更新，从而获得更好的策略.当值函数满足式(3)时，当前的值函数即为最优值函数，从最优值函数中获得的策略即为最优策略[20].

2.2 行动者评论家算法

强化学习可以分为基于值函数方法与基于策略梯度方法，行动者评论家[21]算法则将这两种方法结合，可以有效运用到大规模连续动作任务中.行动者评论家算法将值函数方法与策略梯度方法结合，一方面弥补了策略梯度难以判断动作好坏的问题，另一方面扩大了值函数方法的应用范围.使用值函数的算法通常应用于离散动作任务中，使用策略梯度作为动作选择函数能够弥补这一缺陷，使得值函数能够应用于连续动作任务中.行动者评论家算法示意图如图1.

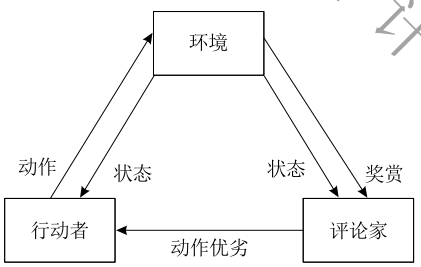


图1 行动者评论家算法

图1中环境提供当前状态以及在当前状态执行动作得到的奖赏和下一个状态.行动者使用策略梯度方法，通过参数化网络决定当前状态执行的动作.评论家使用值函数方法，通过立即奖赏和下一状态更新自身并对行动者在当前状态选择的动作进行评估.评论家将评估动作的优劣反馈给行动者，行动者通过评论家的反馈对策略进行调整.

2.3 策略梯度

策略梯度[22]的目标是寻找最优策略 π 使得期望回报 $\eta(\pi)$ 最大：

$$\eta(\pi)=\mathbb{E}_{s_0,a_0,\cdots}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t)\right] \tag{4}$$

其中 $s_0\sim\rho_0(\cdot)$ 表示初始状态的分布，$a_t\sim\pi(\cdot\mid s_t)$，$s_{t+1}\sim P(s_{t+1}\mid s_t,a_t)$.策略梯度算法的梯度可以表示为如下形式：

$$g=\mathbb{E}_s\left[\mathbb{E}_a\left[\tau_t\nabla\log\pi_\theta(a_t\mid s_t)\right]\right] \tag{5}$$

在状态空间巨大的大规模任务中，数组形式的值函数无法覆盖整个状态空间，因此使用近似函数表示值函数与策略函数.式(5)中 θ 为近似策略函数 π 的参数，τ_t 为目标函数，可以用状态值函数 $V_\pi(s)$，状态动作值函数 $Q_\pi(s,a)$ 或者优势函数 $A_\pi(s,a)$ 替代：

$$A_\pi(s,a)=Q_\pi(s,a)-V_\pi(s) \tag{6}$$

获得梯度后使用梯度下降方式对策略函数进行更新并改进策略，但是在实际运用中很难获得策略函数准确的梯度，通常使用三种方式计算梯度：

(1)批量梯度(Batch Gradient).使用所有获得的数据计算梯度然后对参数进行更新，优点是实现简单并且保证收敛，缺点是数据量多时需要分批计算并且无法进行在线更新；

(2)随机梯度(Stochastic Gradient).与批量梯度相反，拿到一个数据后就进行梯度计算并更新参数，优点是收敛速度快并且支持在线更新，缺点是无法使用矩阵操作加速计算；

(3)最小批量梯度(Mini-batch Gradient)，该方法在前两种方法中折中，每次使用固定大小的数据计算梯度并更新，保证在线更新的同时可以利用矩阵计算进行加速.

本文的策略梯度使用最小批量梯度方法来计算，除了能够在线更新和利用矩阵加速，还能够一定程度保持策略的稳定性，防止学习中发生剧烈抖动导致算法性能的下降.

2.4 最大熵正则项

为了保证策略持续探索，通常在计算梯度时加入最大熵正则项[23]，通过增加策略的信息熵以防止策略变为确定性策略，保证所有动作都能被遍历.此时梯度变成如下形式：

$$g=\mathbb{E}_s\left[\mathbb{E}_a\left[\tau_t\nabla\log\pi_\theta(a_t\mid s_t)\right]+\beta\nabla_\theta H_\pi(s_t)\right] \tag{7}$$

其中 $H_\pi(s_t)=-\sum_a\pi(s_t,a)\log\pi(s_t,a)$ 表示策略 π 在状态 s_t 下的信息熵，$\beta>0$ 表示正则项的惩罚参数.信息熵用于衡量信息所带信息量的大小.以策略为例，当一个策略为确定性策略(确定某个动作为最优动作，不再选择其他动作)，那么这个策略的信息熵最小；若一个策略是随机策略(每个动作选取的概率相同)，那么这个策略的信息熵最大.在梯度中加入最大熵正则项会使得改进后的策略分布倾向于熵更大的状态，即倾向于随机策略，有利于保持策略的探索性，最大熵正则项对策略的影响如图2所示.

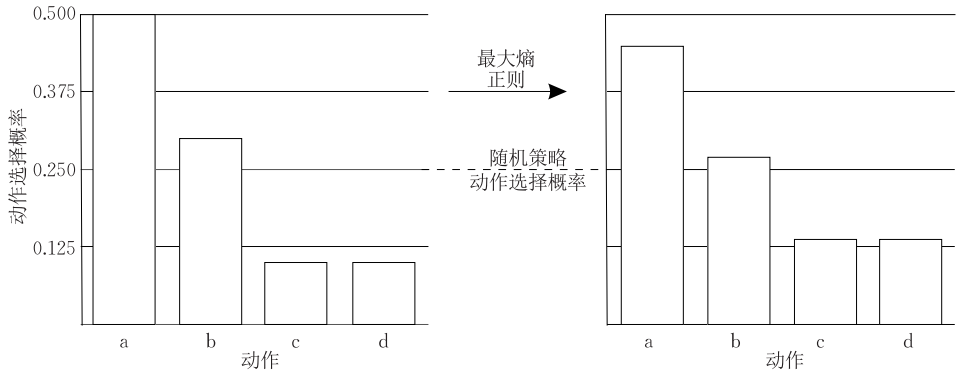


图 2　最大熵正则项对策略的影响

3　最大熵修正

3.1　网络结构

在深度强化学习中，行动者评论家算法的网络结构从更新方式可以分为两种：共享权值和目标网络. 共享权值是指网络中选取动作与策略评估的网络共享一套权重，通过权值共享，将策略改进与策略评估更好地联系在一起，通常应用于不使用经验重放的算法. 使用目标网络的网络结构将策略改进与策略评估分开，当前网络负责策略改进和动作选择，在每一次更新中都会进行策略改进与值函数更新；目标网络负责策略评估，在当前网络选择动作后对动作进行评估，目标网络不随当前网络的更新发生改变，只在多轮更新后进行同步操作. 这样做保证了使用经验重放的情况下网络的稳定性. 使用经验重放提高了样本利用率，但是当前网络生成的策略不能及时得到评估，所以推迟目标网络的更新以保证学习的稳定性.

这两种结构各有优劣，目标网络使用经验重放，样本利用率高，目标网络的更新频率低，学习到的策略更加稳定，缺点是内存占用大，无法进行在线更新. 共享权重不使用经验重放，占用内存小，策略能够通过在线更新快速得到改进与评估，可以快速地进行学习，但是在具有高度随机奖赏的环境中，性能的稳定性不足.

本文使用与 A2C 一致的共享权重网络结构，具体结构如图 3. 从图中可以看出，输入首先进入特征提取网络，常用的特征提取有多层神经网(Multilayer Perceptron，MLP)、卷积神经网(CNN)等[24]，提取的特征通过全连接得到值函数，策略函数在全连接后再进行归一化，保证所有动作概率和为 1. 使用共享权重有两个优点：(1) 学习效率高. 使用共享权重的网络相较于使用目标网络的网络，训练时间大大减少；(2) 策略函数与值函数对应. 使用目标网络所得到的值函数与策略函数在未同步之前并不统一，而共享权重网络策略不使用经验重放并且只有一套权重，策略函数与值函数相互对应，有利于理论研究的展开与验证.

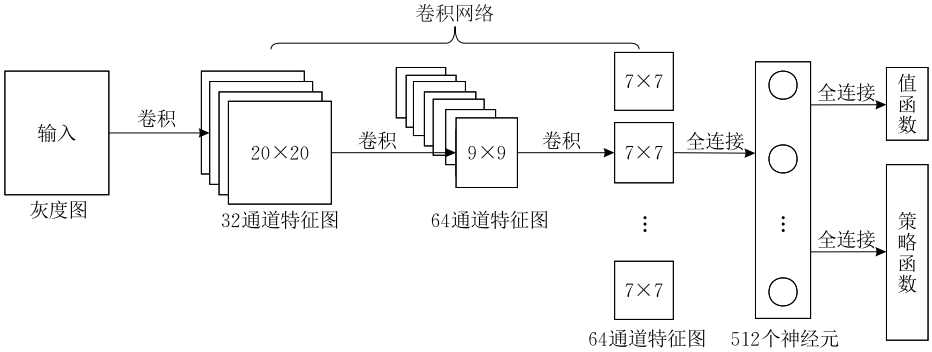


图 3　共享权重网络结构[13]

3.2　策略优化方法

策略优化算法可以分为三类：(1) 策略迭代方法，通过策略改进与策略评估交替进行来对值函数进行估计；(2) 策略梯度方法，通过采样获得的轨迹计算策略的梯度从而更新策略函数；(3) 梯度无关算法，例如交叉熵算法.

本文使用策略梯度方法中的近似策略优化(PPO)算法进行策略的更新. 该方法使用优势函数

A_π作为目标函数，通过满足如下信任域来保证策略改进的单调性：

$$\text{最大化 } \mathbb{E}_t[\rho_t(\pi)A_t]$$

$$\text{且服从 } \mathbb{E}_t[CE[\pi(\cdot|s_t),\pi'(\cdot|s_t)]]\leq\sigma \quad (8)$$

其中π为更新后的策略，π'为更新前的策略，$\rho_t(\pi)=\frac{\pi(a_t|s_t)}{\pi'(a_t|s_t)}$为重要性采样率，用于修正异策略更新带来的偏差，$CE(x,y)=-\sum_a x(a)\cdot\log y(a)$为交叉熵，用于计算两个策略分布的距离.使用上述信任域进行策略优化能够保证单调的改进，但是需要计算二次导数，实现困难并且计算复杂度高.PPO通过构造近似函数来降低计算复杂度：

$$L(\theta)=\mathbb{E}_t[\min(\rho_t(\pi)A_t,clip(\rho_t(\pi),1-\varepsilon,1+\varepsilon)A_t)] \quad (9)$$

其中ε为超参数，通常在0.1～0.2之间，$clip$函数将重要性采样率限制在$(1-\varepsilon,1+\varepsilon)$之间，通过这样的限制可以防止策略更新幅度过大，从而满足式(8).

3.3 修正项构造

最大熵正则项在保证探索的同时也带来了低估问题.设原策略为π，使用最大熵正则项后的策略为π_e，从上可知$\max_a\pi(a|s)\geq\max_a\pi_e(a|s)$.若$\pi$满足$\forall a,b\in A$，如果$Q_\pi(s,a)\geq Q_\pi(s,b)$，则$\pi(a|s)\geq\pi(b|s)$，则有：

$$\eta(\pi)\geq\eta(\pi_e) \quad (10)$$

证明.

$$\begin{aligned}\eta(\pi)&=\sum_a\pi(a|s_0)Q_\pi(s_0,a)\\&=\sum_a\pi_e(a|s_0)Q_\pi(s_0,a)+\\&\quad[\pi(a|s_0)-\pi_e(a|s_0)]Q_\pi(s_0,a)\\&\geq\sum_a\pi_e(a|s_0)Q_\pi(s_0,a)+\\&\quad\sum_a[\pi(a|s_0)-\pi_e(a|s_0)]\max_bQ_\pi(s_0,b)\\&=\sum_a\pi_e(a|s_0)Q_\pi(s_0,a)\\&=\eta(\pi_e).\end{aligned}$$

证毕.

这意味着使用最大熵正则项虽然让策略保持一定的探索，但会使得策略的期望回报降低从而导致学习效率降低.

本文的主要贡献是用状态值函数与策略函数设计一种状态动作值函数的估计，该估计满足真实状态动作值函数的分布，并使用该估计通过贝尔曼最优方程构造新的目标函数.这里使用优势函数A_π构造目标函数，其中$Q_\pi(s,a)$被$r(s)+\gamma V_\pi(s')$替代，式(6)改写为

$$A_\pi(s_t,a_t)=\mathbb{E}_{s'\in P(s'|s_t,a_t)}[r(s_t)+\gamma V_\pi(s')-V_\pi(s_t)] \quad (11)$$

使用共享权重并且利用采样得到的轨迹进行实时的更新，会使得同一状态在一次更新中只会更新一个动作，其他动作无法得到更新.这会导致其他动作的值不准确，因为更新值函数权重的同时也会更新特征提取网络的权重，其他动作的权重不更新必然会导致值函数不稳定，以至于影响学习的稳定性.策略函数通常使用Softmax或者Gaussian分布进行归一化，这使得所有动作能够得到更新.当一个动作被认为是好的动作，那么它被选择的概率就会提高，相应其他的动作被选择的概率会按比例降低以保持所有动作的概率和为1.本文将状态值函数与策略函数结合，构造一个状态动作值函数的估计：

$$\hat{Q}_\pi(s,a)=\beta(\log\pi(s|a)+H_\pi(s))+V_\pi(s) \quad (12)$$

其中$\beta>0$与式(6)中一致.使用式(3)贝尔曼最优方程将目标函数改成如下形式：

$$\begin{aligned}\hat{A}_\pi(s_t,a_t)&=\mathbb{E}_{s'\in P(s'|s_t,a_t)}[r(s_t)+\gamma\max_a\hat{Q}_\pi(s',a)-V_\pi(s_t)]\\&=A_\pi(s_t,a_t)+\gamma\beta C(s_t,a_t)\end{aligned} \quad (13)$$

其中$C(s_t,a_t)$为最大熵修正项，具体形式为$\mathbb{E}_{s'\in P(s'|s_t,a_t)}[\max_a\log\pi(s'|a)+H_\pi(s')]$.$\hat{A}_\pi(s_t,a_t)$的计算复杂度由$A_\pi(s_t,a_t)$与$C(s_t,a_t)$两部分组成，$A_\pi(s_t,a_t)$的计算复杂度为$O(1)$，$C(s_t,a_t)$需要计算最大值与信息熵，复杂度为$O(N)$，$N$为离散动作的数量，通常小于20.但是算法的总复杂度来自于深度网络的向后传播，$\hat{A}_\pi(s_t,a_t)$在每轮mini-batch中被计算好，会在网络中经过多轮的训练，因此对训练时长的影响很小.$\hat{A}_\pi(s_t,a_t)$由于使用的是已有的网络输出进行计算，所以无需增加额外的空间.

MEC算法如下：

算法1. MEC算法.

输入：$\theta,\gamma,\beta,\varepsilon,\alpha$

输出：θ

1. 初始化$\theta,\gamma,\beta,\varepsilon,\alpha$
2. REPEAT(每轮更新)
3. 利用策略网络与环境交互获得mini-batch
4. REPEAT(mini-batch中的每组(s,a))
5. 使用式(13)计算策略函数目标值
6. 计算$r(s)+\gamma\max_a\hat{Q}_\pi(s',a)$作为值函数目标值
7. END REPEAT
8. 使用式(9)对计算策略误差
9. 使用式(7)对策略函数网络进行更新
10. 使用$\nabla_\theta(r(s)+\gamma\max_a\hat{Q}_\pi(s',a)-V_\pi(s))^2$对值函数网络进行更新
11. END REPEAT

3.4 最大熵修正项的理论分析

构造的状态动作值函数的估计式(12)满足：

若 $\pi(a|s)\geq\pi(b|s)$ 则 $\hat{Q}_\pi(s,a)\geq\hat{Q}_\pi(s,b)$，且

$$\begin{aligned}\sum_a\pi(a|s)\hat{Q}_\pi(s,a)&=\sum_a\pi(a|s)\log\pi(a|s)+\\&\quad H_\pi(s)+V_\pi(s)\\&=-H_\pi(s)+H_\pi(s)+V_\pi(s)\\&=V_\pi(s).\end{aligned}$$

由上式可知构造的状态动作值函数 $\hat{Q}_\pi(s,a)$ 的均值仍是 $V_\pi(s)$ 且其分布符合当前的策略函数的分布. 虽然 $\hat{Q}_\pi(s,a)$ 不是准确的状态动作值函数，但是其最优动作与准确的值函数一致，利用这一特性本文构造了新的目标函数 $\hat{A}_\pi(s_t,a_t)$. 对于新的目标函数，若 π 满足 $\forall a,b\in A$，如果 $Q_\pi(s,a)\geq Q_\pi(s,b)$，则 $\pi(a|s)\geq\pi(b|s)$，且有：

$$\begin{aligned}\hat{\eta}(\pi_e)-\eta(\pi_e)&=\sum_a\pi_e(a|s_0)Q^*_{\pi_e}(s_0,a)-Q_{\pi_e}(s_0,a)\\&=\sum_a\pi_e(a|s_0)r(s_0)+\gamma\max_a\hat{Q}_{\pi_e}(s',a)-\\&\quad(r(s_0)+\gamma V_{\pi_e}(s'))\\&=\sum_a\pi_e(a|s_0)r(s_0)+\gamma\max_a\hat{Q}_{\pi_e}(s',a)-\\&\quad V_{\pi_e}(s_t)-(r(s_0)+\gamma V_{\pi_e}(s')-V_{\pi_e}(s_t))\\&=\sum_a\pi_e(a|s_0)\hat{A}_{\pi_e}(s_0,a)-A_{\pi_e}(s_0,a)\\&=\sum_a\gamma\beta C(s_0,a)\end{aligned}\tag{14}$$

由式(13)定义可知 $C(s_t,a_t)\geq0$，那么从上式可得 $\hat{\eta}(\pi_e)\geq\eta(\pi_e)$，这意味着使用修正项进行更新，可以在保证策略探索性的同时弥补使用最大熵正则项带来的低估问题. 由于无法获得准确的状态动作值函数，过大的修正会带来稳定性的下降，因此使用 β 参数能够对修正的幅度进行调整以获得更好的收敛速度. 另一方面，使用最大熵修正项能够打破使用 Softmax 软贪心算法带来的平衡态. 假设两个状态 s_1 与 s_2 的值函数一致，但是 s_1 的最大动作值大于 s_2 的最大动作值，这意味 s_1 相较于 s_2 更优. 使用下一个状态最大动作的估计能够一定程度地打破这样的僵局，提高收敛速率.

由于目标函数通过估计值进行构造，所以在一些情况下会有性能损失：(1) 学习初期值函数和策略函数与真实值相差较大，使用式(13)会造成值函数不稳定从而导致学习效率下降. 本文使用的 β 较小，对于学习初期有一定的影响，但是随着值函数与策略函数的逐渐稳定并满足上述条件，这种影响会逐渐消失；(2) 当奖赏函数有阶段性，使得策略函数长期不满足 $\forall a,b\in A$，如果 $Q_\pi(s,a)\geq Q_\pi(s,b)$，则 $\pi(a|s)\geq\pi(b|s)$ 时，那么式(14)不成立，这时 $\hat{Q}_\pi(s,a)$ 不符合当前状态动作值函数的分布，导致使用 $\hat{A}_\pi(s_t,a_t)$ 作为目标函数进行更新不能对最大熵正则项进行修正，同时也会引入误差造成性能损失.

4 实　验

本文使用 Gym 平台中的 Atari 2600 游戏作为实验环境. Gym 是 OpenAI 公司开发的环境平台，其中涵盖了多种多样的实验环境，例如强化学习环境 Mountain Car、平衡杆、Atari 2600 游戏环境、MuJoCo 控制机器人环境等. 本文针对离散动作任务中的最大熵正则项进行修正，因此选择 7 个 Atari 2600 游戏：BeamRider、Breakout、Enduro、Pong、Qbert、Seaquest、SpaceInvaders 进行测试. 这 7 种实验涵盖了奖赏稀疏、强策略性、强随机性等多种性质，能够较为完善地测试算法在不同环境中的表现，在 OpenAI 的 baselines① 中作为基准实验. 由于本文使用共享权重网络和异步更新，所以本文选取 PPO 算法、ACKTR 算法、A2C 算法与 MEC 算法在上述环境中进行对比实验.

4.1 实验参数设置

为了保证实验的有效性，本文使用的所有算法均使用相同的参数和 Adam 梯度下降方法. 实验的输入为 80×80 分辨率的 RGB 三通道图像，特征提取网络使用 3 层卷积网络，第一层为 32 通道 8×8 的卷积核，步长为 4，生成 32 通道 20×20 的特征图，第二层为 64 通道 4×4 的卷积核，步长为 2，第三层为 64 通道 3×3 的卷积核，步长为 1，最终得到 64 通道 7×7 的特征图. 特征图与 512 个神经元全连接后作为值函数网络与策略函数网络的输入，值函数网络通过全连接至单个神经元作为状态值函数的输出，策略函数网络输出神经元数量与动作数对应，所有神经元与输入全连接后经过 Softmax 归一化作为策略函数的输出.

MEC 在 PPO 的基础上构造了新的目标函数，因此对比实验使用的参数参照 PPO，设置如下：MDP 环境折扣因子 γ 设为 0.99，最大熵正则项的惩罚参数 β 设为 0.01，PPO 超参数 ε 为 0.1，梯度下降更新步长 α 为 2.5×10^{-4}. 本文使用较小的惩罚参数 β，只会对策略进行细微的调整，这种调整不会导致最优动作发生改变，最优动作的选择概率会略

① Openai baselines. https://github.com/openai/baselines.

微降低以保证其他动作可以被遍历到.使用这种软贪心策略,既保证了大部分情况下选择的是最优动作,又能够遍历到所有的动作,防止策略陷入局部最优当中.

网络模型的参数使用最小批量梯度方法进行更新,为了保证模型学习性能的稳定性,使用 8 个行动者同时与环境交互,每个行动者进行交互的环境参数相同但是相互独立,每个环境收集 32 个时间步的数据,每次更新使用 8×32 时间步的数组作为一个最小批次.每个实验训练总步长为 10 000 000 步,使用测试集上 100 个情节获得奖赏的均值作为性能指标.

4.2 超参数对比实验

为了探究超参数对算法性能的影响,将 MEC 算法在 Qbert 游戏环境中使用不同的 β 进行对比实验,此时 β 同时作用于最大熵正则项与最大熵修正项,结果如图 4 所示.

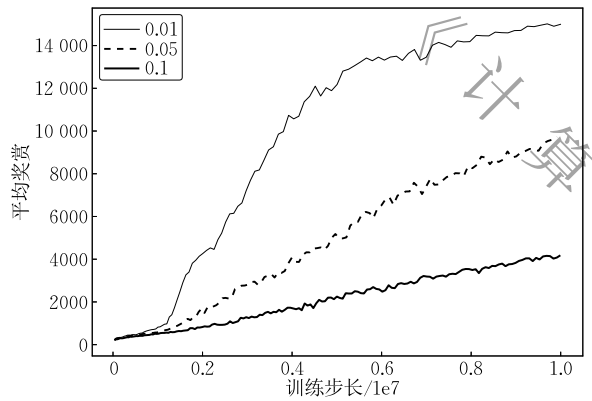


图 4 最大熵正则项 β 超参数对比实验

从图 4 中可以看出,超参数 β 对 MEC 算法有着较大的影响.为了探究是最大熵正则项还是最大熵修正项导致性能显著波动,本文将最大熵正则项的超参数固定为性能最好的 0.01,改变最大熵修正项的超参数再次进行对比实验,实验结果如图 5 所示.

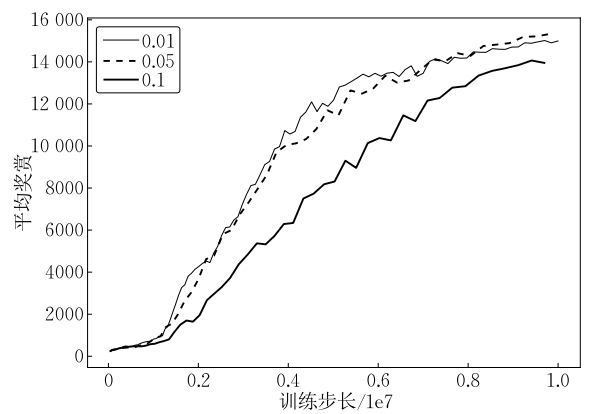


图 5 MEC 算法 β 超参数对比实验
(最大熵正则项超参数固定为 0.01)

结合图 5 与图 4 可以得知最大熵正则项对于超参数比较敏感,这是因为 β 越大,最大熵正则项的影响就越强,策略越趋向于随机策略.策略过于随机就会导致最优动作被选择的频率降低,值函数被低估使得收敛速度下降.从图 5 中可以看出最大熵修正项在不同 β 下表现稳定,在训练前期 β 越小收敛速度越快但是 β 为 0.05 时最终结果最好.这意味着:(1)最大熵修正项与正则项的超参数不一定要统一,使用不同的超参数可能会得到更好的结果;(2)最大熵修正项在训练的不同阶段有着不同的效果,通过构造自适应的超参数可能会得到更好的训练效果.

4.3 稳定性对比实验

文献[25]表明:深度强化学习中,使用不同的随机种子进行训练会影响算法的收敛速度和性能;算法对随机种子的敏感程度体现了算法的稳定性,算法的稳定性决定了其在不同环境下的收敛性和性能.稳定性较差的算法在不同环境和随机种子下的表现会有巨大的差别,这意味着:(1)使用某一组随机种子的训练结果作为实验数据并不足信,应当选取多组随机种子的训练结果来衡量算法有效性;(2)算法稳定性决定了算法在不同环境中性能是否稳定,一个稳定的算法有利于其他研究者复现并改进算法以及将算法应用于实践中,因此算法的稳定性应当作为衡量算法好坏的标准之一.为了验证算法的稳定性,本节选取 PPO 作为对比算法使用六组随机种子在 Breakout 环境中进行训练.选取 PPO 的原因是 MEC 算法采用了与 PPO 相同的策略优化算法,能够客观地反映算法的稳定性.训练曲线如图 6 所示.

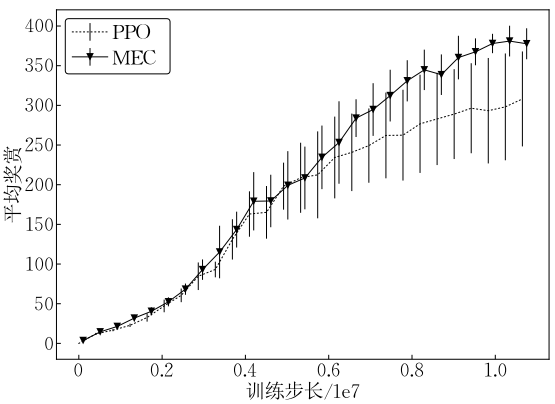


图 6 Breakout 对比实验方差曲线图

图 6 直观展示在两种算法在学习过程中的稳定性,可以看出在学习的前中期,两者的方差近似,但是随着时间步的增加,MEC 算法的方差在逐渐减少,而 PPO 的方差在逐渐增加.这是因为使用最大

熵会使策略趋向于随机，过多的随机会影响算法的稳定性，并且随着学习步长的增加影响逐步扩大. 使用最大熵修正项能够保证策略探索性的同时减少随机动作对于学习的影响.

为了验证训练得到的模型的有效性与稳定性，本文将训练后的模型在测试集中取 100 个情节的奖赏的均值作为结果，如表 1 所示.

表 1　不同随机种子训练得到模型测试集结果

随机种子	PPO	MEC
0	357.17	396.46
10	243.64	410.05
50	347.57	372.64
100	379.34	408.63
500	234.22	362.91
1000	255.01	382.62

从表 1 所示数据可以看出 MEC 算法不仅在最终的奖赏上好于 PPO 算法，而且在稳定性上也优于 PPO. PPO 上下界相差 145.12 而 MEC 算法上下界则只相差 47.14，验证了 MEC 算法相较于 PPO 算法更加稳定.

4.4　性能对比实验

为了验证本文提出的 MEC 算法的有效性，本文评估了 MEC、PPO、ACKTR 和 A2C 算法在 BeamRide、Enduro、Pong、Qbert、Seaquest、SpaceInvaders 上的性能表现，每个环境使用了 6 组不同的随机种子进行训练，最终结果由所有随机种子取平均得到，曲线图中横坐标为训练阶段，纵坐标为当前训练阶段最近 100 个情节获得奖赏的平均值. 结果如图 7 所示.

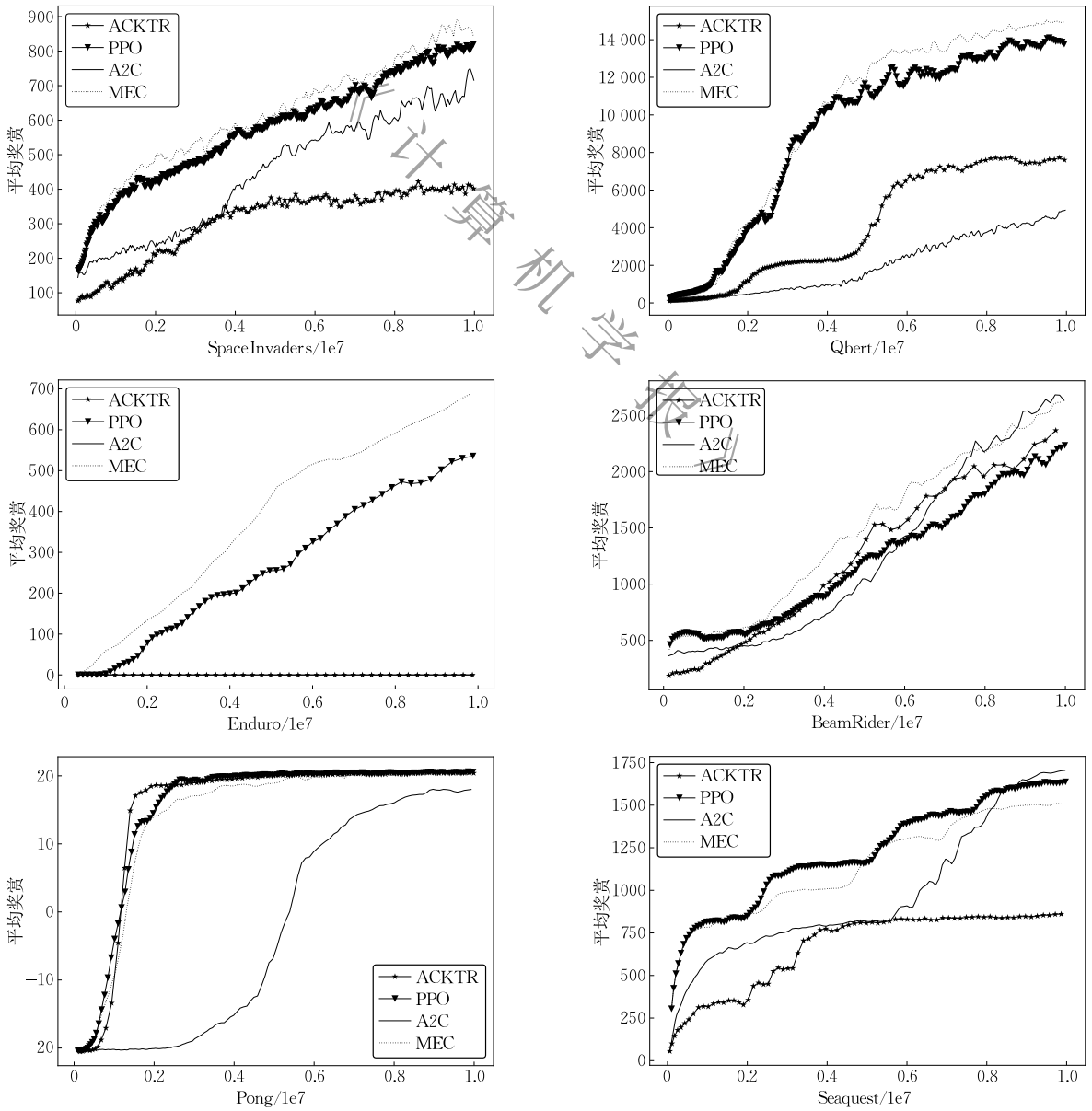


图 7　最大熵修正算法与 PPO、A2C、ACKTR 的对比实验结果(横坐标表示训练步长，纵坐标为最近 100 个情节的平均奖赏)

从图 7 中可以看出，MEC 算法在大部分游戏环境中的表现都优于 PPO 和 A2C，具有更快的收敛速度. 下面选取一些实验环境分析算法性能提升的原因以及存在的一些问题：

(1) 选取 Qbert 环境进行分析. 在 Qbert 中，玩家控制人物在阶梯上移动并躲避怪物，人物遍历完阶梯上的所有台阶，则进入下一关，当人物从阶梯边缘掉落或者碰到怪物则情节结束. 从图 7 中可以看出 MEC 算法在最终收敛结果上最好，而 A2C 与 ACKTR 算法的收敛速度明显低于其他两种算法. 在学习的前期，MEC 算法与 PPO 算法的性能没有明显的差距，原因有两点：① 在训练前期值函数不准确且初始化时策略函数中所有动作的概率相等，这时的策略不满足 $\forall a,b\in A$，如果 $\pi(a|s)\geq\pi(b|s)$，则 $Q_\pi(s,a)\geq Q_\pi(s,b)$，因此修正项的作用并没有发挥出来；② 本文使用的最大熵正则项的惩罚系数较小，在训练前期对值函数产生的影响并不明显，即使正则项发挥作用也不会有明显的差距.

但是随着时间步与采样轨迹的增加，值函数逐渐向真实值靠拢，策略函数分布开始符合值函数的分布，最大熵正则项在策略函数中的影响逐渐扩大. 此时最大熵修正项开始发挥作用，通过贝尔曼最优方程抵消使用最大熵带来的低估问题. 图 7 中的结果符合 3.4 节中的分析，虽然使用的是估计值，但只要值函数与策略函数满足 $\forall a,b\in A$，如果 $\pi(a|s)\geq\pi(b|s)$，则 $Q_\pi(s,a)\geq Q_\pi(s,b)$，使用修正项就能够在弥补低估问题的同时打破使用最大熵正则项带来的僵局，在训练后期带来更好的学习效率以及性能表现.

(2) 选取 Pong 进行分析，实验结果在图 7. Pong 可以看做乒乓的简化，玩家控制横板接球，电脑同样控制横板接球. 某一方未接住球，则另一方得一分. 当某一方得分到达 21 分时，情节结束，最终奖赏由双方比分差距决定. 从图 7 中可以看出，虽然在训练最后阶段四种算法都收敛了，但 A2C 算法收敛较慢且最终结果低于 ACKTR、PPO 与 MEC 算法. MEC 算法与 PPO 最终收敛结果接近，但是 PPO 比 MEC 算法收敛更快. 产生这一现象的原因是 MEC 使用了贝尔曼最优方程，贝尔曼最优方程选取最大值作为目标函数，意味着 Agent 总是乐观估计当前状态，认为当前环境下的最优策略总是能够得到最好的结果. 这种做法在 Pong 的对抗环境中会将对手的失误放大，认为对手的失误是必然的，因此在进行动作选择时过于自信，从而跳入对手的陷阱中. 这种情况在训练前期采样较少时经常出现，所以在训练前期 PPO 的学习效果比使用最大熵修正的算法好. 随着采样的增加，对于动作的高估被矫正，值函数逐渐向真实值接近，此时 MEC 算法逐渐追平 PPO 算法.

(3) 选取 Seaquest 进行分析. 在 Seaquest 中，玩家控制潜水艇救人，在水中潜水艇能够使用子弹击杀敌人或者靠近潜水员拯救他们，潜水艇在水中会消耗氧气. 当潜水艇接触到敌人、没有氧气或者浮出水面补充氧气时潜水艇中没有潜水员，情节结束. 从图 7 中可以看出，MEC 算法性能较 PPO 和 A2C 有一定差距. 主要原因是奖赏存在阶梯性，Seaquest 与其他 5 种环境不同，奖赏主要来源于将潜水员送上岸. 这使得值函数波动较大，同时策略也会有较大波动，从 Qbert 实验分析中得知在值函数波动的情况下，使用最大熵正则项的效果并不显著. 在 Seaquest 中，击杀敌人同样有较少的奖赏，但是消耗氧气得不偿失，从 Pong 实验中得知使用贝尔曼最优方程式会乐观估计当前情况从而导致学习性能的下降. 在这两点的共同作用下，MEC 算法的性能表现要劣于 PPO 和 A2C 算法.

为了验证训练得到的模型的有效性，将训练后的模型在测试集中进行测试. 在测试集上取 6 组随机种子下各 100 个情节获得的最终奖赏的平均值作为结果，如表 2 所示.

表 2 Atari 2600 游戏测试集结果

实验环境	BeamRide	Enduro	Pong	Qbert	Seaquest	SpaceInvaders
MEC	2765.59	**726.89**	**20.74**	**15178.12**	1514.53	**886.02**
PPO	2209.97	556.07	20.63	14035.66	1642.13	848.56
A2C	2826.63	0	18.15	5692.30	**1721.84**	735.59
ACKTR	**3116.18**	0	20.39	868.16	868.16	461.64

从表 2 的结果能够看出 MEC 在大部分游戏中都有着性能的提升. 这说明本文提出的最大熵修正算法在大部分任务中相较于其他算法有一定的优势，能够提高收敛速度与学习效率. 上述结果使用 6 组随机种子，同时验证该算法能够提供稳定的性能提升，不会受到随机种子的影响.

5 总结语

为了解决行动者评论家算法使用最大熵正则项

带来的低估问题，本文提出了最大熵修正算法，通过已有的状态值函数和策略函数构造状态动作值函数的估计并利用贝尔曼最优公式得到新的目标函数，相较于原目标函数多了一项修正项.使用该修正项能够在保证策略探索性的同时弥补最大熵带来的低估，提高算法的学习效率.为了验证该算法的稳定性与有效性，本文在 7 个 Atari 游戏上进行了对比实验.在 Breakout 的 6 组随机种子稳定性对比实验中可以看出，MEC 算法与 PPO 相比在稳定性方面有着较大的提升且平均奖赏更高，这种稳定性的提升随着时间步的增加越发明显，这说明最大熵修正算法在改进性能的同时也能够提高算法的稳定性.在 6 组性能对比实验中可以看出，MEC 算法在大部分环境中有着较明显的性能提升，这种提升通常发生在学习的中后期，这说明当值函数与策略函数满足条件时，最大熵修正项就能够发挥作用提高算法性能.

但是在某些随机性较强或者奖赏具有阶段性的问题中，最大熵修正算法未能起作用，这与构造目标函数的机制有着一定的联系.未来的研究方向可以分为两类：(1)使用新的机制构造目标函数，本文使用的构造方式较为简单，之后的研究可以使用不同的机制进行构造以避免上述问题；(2)使用自适应的惩罚参数，本文使用固定的惩罚参数，在处理问题时不够灵活.通过研究自适应的惩罚参数，使得该算法能够应用到更多环境中.

参　考　文　献

[1] Liu Quan, Zhai Jian-Wei, Zhang Zong-Zhang, et al. A survey on deep reinforcement learning. Chinese Journal of Computers, 2018, 41(1): 1-27(in Chinese)
(刘全，翟建伟，章宗长等. 深度强化学习综述. 计算机学报，2018，41(1)：1-27)

[2] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444

[3] Tesauro G. Temporal difference learning and TD-Gammon. Communications of the ACM, 1995, 38(3): 58-68

[4] Seijen H V, Sutton R. True online TD(Lambda)//Proceedings of the 31st International Conference on Machine Learning. Beijing, China, 2014: 692-700

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

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

[7] Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning//Proceedings of the 33nd International Conference on Machine Learning. New York, USA, 2016: 692-700

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

[9] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning//Proceedings of the 4th International Conference on Learning Representations. San Juan, Puerto Rico, 2016: 110-123

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

[11] Wang Z, Bapst V, Heess N, et al. Sample efficient actor-critic with experience replay//Proceedings of the 5th International Conference on Learning Representations. Toulon, France, 2017: 344-360

[12] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning//Proceedings of the 33rd International Conference on Machine Learning. New York, USA, 2016: 1928-1937

[13] Wu Y, Mansimov E, Grosse R B, et al. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation//Proceedings of the Neural Information Processing Systems. California, USA, 2017: 5279-5288

[14] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization//Proceedings of the 32nd International Conference on Machine Learning. Lille, France, 2015: 1889-1897

[15] Wang Y, He H, Tan X. Truly proximal policy optimization//Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence. Tel Aviv, Israel, 2019: 131-144

[16] Fujita Y, Maeda S. Clipped action policy gradient//Proceedings of the 35th International Conference on Machine Learning. Stockholm, Sweden, 2018: 1592-1601

[17] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. USA, Cambridge: MIT Press, 2018

[18] Busoniu L, Babuska R, De Schutter B, et al. Reinforcement Learning and Dynamic Programming Using Function Approximators. USA, Boca Raton: CRC Press, 2017

[19] Puterman M L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. USA, Hoboken: John Wiley & Sons, 2014

[20] Modares H, Nageshrao S P, Lopes G A D, et al. Optimal model-free output synchronization of heterogeneous systems using off-policy reinforcement learning. Automatica, 2016, 71: 334-341

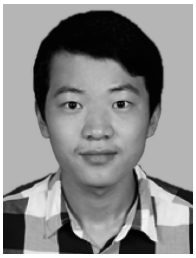
[21] Ghavamzadeh M, Engel Y, Valko M. Bayesian policy gradient and actor-critic algorithms. The Journal of Machine Learning Research, 2016, 17(1): 2319-2371

[22] Grondman I, Busoniu L, Lopes G A D, et al. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6): 1291-1307

[23] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor//Proceedings of the 35th International Conference on Machine Learning. Stockholm, Sweden, 2018: 1856-1865

[24] Schmidhuber J. Deep learning in neural networks: An overview. Neural networks, 2015, 61: 85-117

[25] Henderson P, Islam R, Bachman P, et al. Deep reinforcement learning that matters//Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans, USA, 2018: 3207-3214



JIANG Yu-Bin, M. S. candidate. His main research interests include reinforcement learning, deep reinforcement learning.

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

HU Zhi-Hui, M. S. candidate. Her main research interests include reinforcement learning, deep reinforcement learning.

Background

Deep reinforcement learning based on value function includes deep Q-Network, dueling deep Q-Network, double deep Q-Network and more. These methods are limited to value functions and cannot be applied to continuous motion tasks. To extend the application of deep reinforcement learning, deep actor-critic algorithms such as A3C, A2C, PPO, etc. use the policy gradient method as the policy function. These algorithms have achieved great success in problem domains such as robot control and Atari 2600 games. To ensure the exploratory nature of the policy, these algorithms typically use maximum entropy regularization terms to prevent the policy from becoming a fixed policy. But the maximum entropy regular term will cause the policy to be biased towards randomness. The stochastic policy will make the value function be underestimated. Underestimation of the value function leads to a slower convergence of the algorithm and inappropriate randomness can lead to unstable algorithm performance. Aiming at the underestimation problem caused by the maximum entropy regular term in the policy gradient, the Maximum-Entropy Correction (MEC) algorithm is proposed. The state action function and the objective function of MEC are constructed by the existing state value function and the strategy function. Experimental results show that MEC performs better than A2C and PPO and is more stable on seven Atari 2600 tasks.

This paper is supported by the National Natural Science Foundation of China(61772355,61702055,61472262,61502323,61502329), the Jiangsu Province Natural Science Research University Major Projects (18KJA520011, 17KJA520004), 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), A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. These projects aim to expand the theory of reinforcement learning and deep reinforcement learning.