计算机科学
COMPUTER SCIENCE

**基于随机加权三重 Q 学习的异策略最大熵强化学习算法**

范静宇，刘全

引用本文

范静宇，刘全. 基于随机加权三重 Q 学习的异策略最大熵强化学习算法[J]. 计算机科学, 2022, 49(6): 335-341.

FAN Jing-yu, LIU Quan. Off-policy Maximum Entropy Deep Reinforcement Learning Algorithm Based on RandomlyWeighted Triple Q -Learning[J]. Computer Science, 2022, 49(6): 335-341.

---

**相似文章推荐（请使用火狐或 IE 浏览器查看文章）**

**Similar articles recommended (Please use Firefox or IE to view the article)**

基于解耦-检索-生成的图像风格化描述生成模型

Stylized Image Captioning Model Based on Disentangle-Retrieve-Generate

计算机科学, 2022, 49(6): 180-186. https://doi.org/10.11896/jsjkx.211100129

基于特征注意力融合网络的遥感变化检测研究

Remote Sensing Change Detection Based on Feature Fusion and Attention Network

计算机科学, 2022, 49(6): 193-198. https://doi.org/10.11896/jsjkx.210500058

基于样本分布损失的图像多标签分类研究

Study on Multi-label Image Classification Based on Sample Distribution Loss

计算机科学, 2022, 49(6): 210-216. https://doi.org/10.11896/jsjkx.210300267

多分支 RA 胶囊网络及在图像分类中的应用

Multi-branch RA Capsule Network and Its Application in Image Classification

计算机科学, 2022, 49(6): 224-230. https://doi.org/10.11896/jsjkx.210400087

机器学习在金融资产定价中的应用研究综述

Application of Machine Learning in Financial Asset Pricing:A Review

计算机科学, 2022, 49(6): 276-286. https://doi.org/10.11896/jsjkx.210900127

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210300081

# 基于随机加权三重Q学习的异策略最大熵强化学习算法

范静宇[1]　刘　全[1,2,3,4]
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
(20185227066@stu.suda.edu.cn)

**摘　要**　强化学习是机器学习中一个重要的分支,随着深度学习的发展,深度强化学习逐渐发展为强化学习研究的重点。因应用广泛且实用性较强,面向连续控制问题的无模型异策略深度强化学习算法备受关注。同基于离散动作的Q学习一样,类行动者-评论家算法会受到动作值高估问题的影响。在类行动者-评论家算法的学习过程中,剪切双Q学习可以在一定程度上解决动作值高估的问题,但同时也引入了一定程度的低估问题。为了进一步解决类行动者-评论家算法中的高低估问题,提出了一种新的随机加权三重Q学习方法。该方法可以更好地解决类行动者-评论家算法中的高低估问题。此外,将这种新的方法与软行动者-评论家算法结合,提出了一种新的基于随机加权三重Q学习的软行动者-评论家算法,该算法在限制Q估计值在真实Q值附近的同时,通过随机加权方法增加Q估计值的随机性,从而有效解决了学习过程中对动作值的高低估问题。实验结果表明,相比SAC算法、DDPG算法、PPO算法与TD3算法等深度强化学习算法,SAC-RWTQ算法可以在gym仿真平台中的多个Mujoco任务上获得更好的表现。

**关键词**:Q学习;深度学习;异策略强化学习;连续动作空间;最大熵;软行动者-评论家算法
**中图法分类号**　TP181

# Off-policy Maximum Entropy Deep Reinforcement Learning Algorithm Based on Randomly Weighted Triple Q-Learning

FAN Jing-yu[1] and LIU Quan[1,2,3,4]
1 School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
2 Provincial Key Laboratory for Computer Information Processing Technology,Soochow University,Suzhou,Jiangsu 215006,China
3 Key Laboratory of Symbol Computation and Knowledge Engineering of Ministry of Education,Jilin University,Changchun 130012,China
4 Collaborative Innovation Center of Novel Software Technology and Industrialization,Nanjing 210000,China

**Abstract**　Reinforcement learning is an important branch of machine learning. With the development of deep learning,deep reinforcement learning research has gradually developed into the focus of reinforcement learning research. Model-free off-policy deep reinforcement learning algorithms for continuous control attract everyone's attention because of their strong practicality. Like Q-learning,algorithms based on actor-critic suffer from the problem of overestimations. To a certain extent,clipped double Q-learning method solves the effect of the overestimation in actor-critic algorithms,but it also introduces underestimation to the learning process. In order to further solve the problems of overestimation and underestimation in the actor-critic algorithms,a new learning method,randomly weighted triple Q-learning method is proposed. In addition,combining the new method with the soft actor critic algorithm,a new soft actor critic algorithm based on randomly weighted triple Q-learning is proposed. This algorithm not only limits the Q estimation value near the real Q value,but also increases the randomness of the Q estimation value through

到稿日期:2021-03-08　返修日期:2022-01-21
基金项目:国家自然科学基金(61772355,61702055,61502323,61502329);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);江苏省高校优势学科建设工程资助项目
This work was supported by the National Natural Science Foundation of China(61772355,61702055,61502323,61502329),Jiangsu Province Natural Science Research University Major Projects(18KJA520011,17KJA520004),Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education,Jilin University(93K172014K04,93K172017K18),Suzhou Industrial Application of Basic Research Program Part(SYG201422) and Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.
通信作者:刘全(quanliu@suda.edu.cn)

randomly weighted method, so as to solve the problems of overestimation and underestimation of action value in the learning process. Experiment results show that, compared to the SAC algorithm and other currently popular deep reinforcement learning algorithms such as DDPG, PPO and TD3, the SAC-RWTQ algorithm has better performance on several Mujoco tasks on the gym simulation platform.

**Keywords** Q-learning, Deep learning, Off-policy reinforcement learning, Continuous action space, Maximum entropy, Soft actor critic algorithm

## 1 引言

强化学习(Reinforcement Learning, RL)是一种无监督的机器学习方法[1]。目前,强化学习已经在一些具有挑战性的领域中(如机器人控制[2]和博弈游戏[3-4]等)得到了广泛的应用。强化学习通常使用马尔可夫决策过程(Markov Decision Processes, MDPs)作为随机条件下的理论框架,通常采用最大化累计奖赏的手段来求解最优策略[5]。

强化学习分为基于模型强化学习和模型无关强化学习。模型无关强化学习作为强化学习算法的一个分支,因其实用性较强而备受关注。Watkins 等[6]于 1989 年提出了模型无关强化学习算法 Q 学习(Q-Learning, QL)。Q 学习是很多现代模型无关强化学习算法的基础,该算法在当时取得了良好的效果。Q 学习有一个致命的缺点,即其对动作值的高估,其在动作评估过程中包含一个最大化操作,这个操作会导致算法每次学习到的动作值比真实值高。由于 TD 学习(Time Difference Learning, TDL)的更新公式[7]会对学习到的动作值进行二次利用,对动作值的过高估计会因此累积,这直接导致了 Q 学习算法对动作值的高估问题。

传统的强化学习中,如何表达高维度的状态空间和动作空间一直是一个难以解决的问题。近年来,深度学习(Deep Learning, DL)[8]的发展在一定程度上解决了此问题。卷积神经网络(Convolutional Neural Networks, CNN)使得抽取图像特征作为状态变得可行,深度 Q 网络(Deep Q-Network, DQN)也应运而生[9]。深度 Q 网络成功地结合了深度神经网络(Deep Neural Network, DNN)和 Q 学习,也是第一种将 Q 学习和非线性函数逼近技术成功结合的算法。深度学习和强化学习相结合得到的深度强化学习(Deep Reinforcement Learning, DRL)使得传统的强化学习算法可以在一些复杂的环境中取得更好的效果。深度 Q 网络已经被证明能够在 Atari 2600 的许多个游戏中训练得到可媲美人类操纵水平的控制策略。尽管深度神经网络可以较好地逼近动作值,但是 DQN 中仍然使用了最大值操作来进行更新,因此 DQN 算法还是会极大地高估动作值。

为了解决 DQN 中的高估问题,Hasselt 等[10]于 2016 年提出了双深度 Q 网络(Double Deep Q-Network, DDQN)。DDQN 在 DQN 的基础上进行了改进,有效地缓解了 DQN 的高估问题。并且,在部分 Atari 2600 游戏中,DDQN 取得了优于 DQN 的表现。这也证明了解决高估问题可以影响算法训练后所获得策略的好坏。

DQN 等基于 Q 学习的算法在离散动作空间任务中具有良好的效果,这类方法大多通过 ε-greedy 策略来选择动作。ε-greedy 策略每次选择动作时有 $1-\varepsilon$ 的概率来选择动作值最大的动作,从而利用已经学习到的动作值。另外,以 ε 的概率进行探索,从而学习新的动作值。在连续动作空间求解动作值最大的动作的代价较高,因此 DQN 等基于 Q 学习的算法在连续动作空间任务上的效果并不能令人满意,而基于策略梯度的方法可以较好地解决此问题。基于策略梯度的方法可以分为随机策略梯度(Stochastic Policy Gradients, SPG)方法和确定性策略梯度(Deterministic Policy Gradients, DPG)方法。确定性策略梯度方法在选择动作时,策略输出一个确定的动作,无须求解动作值最大的动作,因此确定性策略梯度方法可以被较好地应用到连续控制任务中。深度确定性策略梯度(Deep Deterministic Policy Gradients, DDPG)算法[11]是确定性策略梯度算法中比较有代表性的算法。

DDPG 算法的样本学习率很高,但其对超参数十分敏感,因而很难达到较好的效果。Fujimoto 等基于双延迟深度确定性策略梯度(Twin Delayed Deep Deterministic policy gradient, TD3)算法[12]提出了剪切双 Q 学习(Clipped Double Q-learning, CDQ),该方法取两个 Q 估计值中的最小值作为更新目标。软行动者-评论家(Soft Actor-Critic, SAC)算法[13]中也采用了剪切双 Q 学习。虽然剪切双 Q 学习可以较好地解决高估问题,但会引入巨大的低估偏差。

本文改进了剪切双 Q 学习方法,并在其基础上提出了一种新的随机加权三重 Q 学习(Randomly Weighted Triple Q-learning, RWTQ)方法。该方法可以有效减少 Q 值的高估与低估问题,在一些实验环境下取得了良好的效果。

本文的主要贡献包括 3 个方面:

(1)提出了一种新的随机加权三重 Q 学习方法,通过限制 Q 估计值的选择范围以及增加随机性来有效地降低高估与低估带来的影响。

(2)结合随机加权三重 Q 学习方法和 SAC 算法,提出了一种新的基于随机加权三重 Q 学习的软行动者-评论家算法(Soft Actor Critic with Randomly Weighted Triple Q-learning, SAC-RWTQ)。

(3)在 Mujoco 的 4 个环境中,将 SAC-RWTQ 算法与其他算法的性能进行了对比。实验结果表明,相比其他对比算法,使用随机加权三重 Q 学习的改进算法 SAC-RWTQ 的性能获得了一定程度的提升。

## 2 相关工作

### 2.1 强化学习

传统的强化学习建立在马尔可夫决策过程模型之上,以

最大化累积回报为目标。马尔可夫决策过程由四元组 $(\mathcal{S},\mathcal{A},p,r)$ 决定，其中 $\mathcal{S}$ 代表状态空间；$\mathcal{A}$ 代表动作空间；$p$ 代表未知状态迁移概率 $p:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,\infty)$，表示在当前状态 $s_t\in\mathcal{S}$ 下采取动作 $a_t\in\mathcal{A}$ 到达下一状态 $s_{t+1}\in\mathcal{S}$ 的概率密度；在每一个迁移过后，环境都会给出一个奖赏 $r:\mathcal{S}\times\mathcal{A}\to[r_{\min},r_{\max}]$。强化学习的目标是求解最优策略，使累积回报最大化。agent 的策略，即在状态 $s_t$ 下采取动作 $a_t$ 的概率，用 $\pi:\mathcal{S}\times\mathcal{A}\to[0,1]$ 表示。在策略 $\pi(a_t|s_t)$ 下，用 $\rho_\pi(s_t)$ 表示状态的概率密度，用 $\rho_\pi(s_t,a_t)$ 表示状态动作对的概率密度。强化学习的目标就是找到可以使期望奖赏和 $J(\pi)=\sum_{t=0}^{T}\mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t)]$ 最大化的最优策略 $\pi^*$。

### 2.2 基于最大熵模型的强化学习算法

许多同策略深度强化学习算法（如 TRPO 算法[14]、PPO 算法[15]和 A3C 算法[16]等）样本的利用效率都较低，这是由于每一次梯度更新都需要新的样本数据。在任务复杂度较高时，每次梯度更新需要的样本数量可能会非常大。DDPG 算法很好地解决了这个问题，但其对超参数非常敏感，难以应用。软行动者-评论家算法通过在奖赏函数中加入一个最大熵（Maximum Entropy）正则项来修改强化学习目标，加入最大熵正则项后的期望奖赏和 $J(\pi)$ 可表达为：

$$J(\pi)=\sum_{t=0}^{T}\mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t)+\alpha\mathcal{H}(\pi(\cdot|s_t))] \tag{1}$$

其中，$\mathcal{H}(\pi(\cdot|s_t))$ 代表状态 $s_t$ 在策略 $\pi$ 下的熵；$\alpha$ 是温度系数，温度系数直接决定了熵正则项与奖赏的比重，因此可以通过其来控制最优策略的随机性。

### 2.3 高估误差

高估指在对状态动作对的 Q 值进行评估时，其估值相比真实 Q 值偏高的一种情况。高估误差最早在 Q 学习中被发现，而 Q 学习几乎是目前所有基于值的 RL 算法的原型。在 Q 学习中，每次选取动作时会执行一个取最大值的操作。最大值操作会倾向于高估真实的 Q 值，而这种高估误差会被时序差分学习的更新公式进一步扩大。在时序差分学习中，计算一个 Q 值的估计值会用到下一序列状态的 Q 值，这会累积高估误差，进而形成更大的高估误差。

深度 Q 网络等深度强化学习算法采用深度神经网络来估计 Q 值。尽管深度神经网络可以很好地逼近 Q 值，但即使在确定性算法中仍然存在高估现象。Fujimoto 等[12]于 2018 年发现在行动者-评论家强化学习算法中也存在高估问题，如确定性策略梯度和深度确定性策略梯度等算法。

为了改善标准 Q 学习中的高估情况，Hasselt[17]于 2010 年提出了双 Q 学习（Double Q-Learning，DQL）。双 Q 学习可以把目标网络中的最大值操作解耦为动作选择和评估两部分，其中一个 Q 网络用于确定贪婪策略，另一个网络用于确定 Q 值，从而获得无偏的估计。DDQN 是深度学习和双 Q 学习的结合体，其在 DQN 的目标 Q 网络的基础上提供了第二个 Q 网络，在一定程度上缓解了高估问题。但是，以上两种方法都只能在离散动作环境下获得较好的效果。Fujimoto 等[12]基于标准双深度 Q 网络提出了一种面向连续动作空间的行动者-评论家算法变体，但该算法和 DDPG 一样也会受到高估问题的影响。最终，Fujimoto 等提出了剪切双 Q 学习[12]。剪切双 Q 学习通过在两个 Q 估计值中选取较小的一个作为 Q 目标值来解决高估问题。双延迟深度确定性策略梯度和软行动者-评论家算法均采用了此方法来解决高估问题。但是，剪切双 Q 学习在缓解高估问题的同时也会引入巨大的低估误差。

在 Q 学习中，假设 Q 估计值是由 Q 函数 $Q_\theta(s,a)$ 使用参数 $\theta$ 近似得来的。Q 目标值为 $y_\theta=\mathbb{E}[r]+\gamma E_{s'}[\max_{a'}Q_\theta(s',a')]$，$Q_\theta(s,a)$ 可以通过使用梯度下降等方法[18-19]最小化损失函数 $(y_\theta-Q_\theta(s,a))^2/2$ 来更新。参数更新公式如下：

$$\theta_{\text{new}}=\theta+\beta(y_\theta-Q_\theta(s,a))\nabla_\theta Q_\theta(s,a) \tag{2}$$

其中，$\beta$ 是学习率。在实际应用中，Q 估计值 $Q_\theta(s,a)$ 通常包含一些随机误差，这些误差一般是由系统噪声和函数近似误差导致的。$\widetilde{Q}$ 表示当前真实的 Q 值，可以假设：

$$Q_\theta(s,a)=\widetilde{Q}(s,a)+\varepsilon_Q \tag{3}$$

假设在随机误差 $\varepsilon_Q$ 的平均值为 0 且与状态动作对 $(s,a)$ 无关的情况下，式(2)的右侧很可能不准确。$\theta_{\text{true}}$ 表示根据真实 Q 值 $\widetilde{Q}$ 更新后获得的参数，即：

$$\theta_{\text{true}}=\theta+\beta(\widetilde{y}-\widetilde{Q}(s,a))\nabla_\theta Q_\theta(s,a) \tag{4}$$

其中，目标值 $\widetilde{y}=\mathbb{E}[r]+\gamma\mathscr{E}_{s'}[\max_{a'}\widetilde{Q}(s',a')]$。

假设在 $\beta$ 足够小的情况下，更新后的 Q 函数可以使用泰勒展开式很好地近似：

$$Q_{\theta_{\text{true}}}(s,a)\approx Q_\theta(s,a)+\beta(\widetilde{y}-\widetilde{Q}(s,a))\parallel\nabla_\theta Q_\theta(s,a)\parallel_2^2 \tag{5}$$

$$Q_{\theta_{\text{new}}}(s,a)\approx Q_\theta(s,a)+\beta(y_\theta-Q_\theta(s,a))\parallel\nabla_\theta Q_\theta(s,a)\parallel_2^2 \tag{6}$$

更新后的 Q 估计值 $Q_{\theta_{\text{new}}}(s,a)$ 的估计偏差如下：

$$\begin{aligned}\Delta(s,a)&=\mathbb{E}_{\varepsilon_Q}[Q_{\theta_{\text{new}}}(s,a)-Q_{\theta_{\text{true}}}(s,a)]\\&\approx\beta(\mathbb{E}_{\varepsilon_Q}[y_\theta]-\widetilde{y})\parallel\nabla_\theta Q_\theta(s,a)\parallel_2^2\end{aligned} \tag{7}$$

其中，$\mathbb{E}_{\varepsilon_Q}[\max_{a'}(\widetilde{Q}(s',a')+\varepsilon_Q)]-\max_{a'}\widetilde{Q}(s',a')\geqslant 0$[20]。若把 $\mathscr{E}_{s'}[\mathscr{E}_{\varepsilon_Q}[\max_{a'}Q_\theta(s',a')]-\max_{a'}\widetilde{Q}(s',a')]$ 定义为 $\delta$，则偏差 $\Delta(s,a)$ 可以被重写为：

$$\Delta(s,a)\approx\beta\gamma\delta\parallel\nabla_\theta Q_\theta(s,a)\parallel_2^2\geqslant 0 \tag{8}$$

由式(8)可知，$\Delta(s,a)$ 是一个大于等于零的偏差。事实上，任何估计误差都会因为最大化操作而含有一个大于等于零的高估误差。虽然个别的高估误差是合理的，但是这些高估误差可能会被 TD 学习，并将其进一步扩大，进而导致巨大的高估误差和次优策略更新。

## 3 基于随机加权的三重 Q 学习

本节主要介绍一种新的基于随机加权的三重 Q 学习方法与一种新的基于随机加权的三重 Q 学习方法的异策略强化学习算法。

### 3.1 软行动者-评论家算法

软行动者-评论家算法是一种面向连续空间的异策略

强化学习算法，它在通过异策略方法优化随机策略的同时，也把随机策略优化和类 DDPG 方法联系了起来。

软行动者-评论家算法最重要的特点是熵正则化(Entropy Regularization)，该算法在奖赏函数的基础上引入熵正则项来增加算法的探索性。

软行动者-评论家算法同时学习 4 个网络参数 $\phi$，$\theta_1$，$\theta_2$ 和 $\psi$，它们分别表示策略网络 $\pi$、两个 Q 值网络以及 V 值网络的参数。

如图 1 所示，软行动者-评论家算法在梯度更新的过程中，首先在经验回放池中取一个样本集合 B；然后使用此样本集合来计算 Q 值网络和 V 值网络的目标值，根据计算所得的目标值进行梯度更新，更新完毕再对 V 值网络参数进行软拷贝，更新 V 目标值网络；最后通过更新过的 Q 值网络取样更新策略网络 $\pi$。

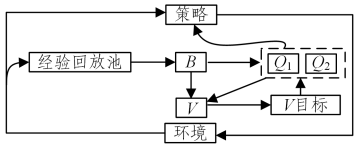



图 1 SAC 算法示意图

Fig. 1 Schematic diagram of soft actor critic algorithm

### 3.2 剪切双 Q 技巧学习

在剪切双 Q 学习方法被提出之前已经出现了一些减小高估偏差的方法，但这些方法在行动者-评论家算法框架下的效果都不尽人意。Fujimoto 等[12]提出了双 Q 学习方法的剪切式变体，即剪切双 Q 学习方法，这种方法可以较好地减小在各种行动者-评论家算法框架中出现的高估偏差。剪切双 Q 学习算法的更新公式如下：

$$y=r+\gamma \min_{i=1,2} Q_{\theta_i'}(s',\pi_\phi(s')) \tag{9}$$

如图 2 所示，CDQ 方法在两个 Q 值网络中取最小值 $Q_{\min}$ 作为 Q 函数的真实目标值并进行计算。

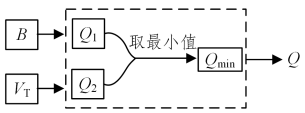



图 2 剪切双 Q 学习方法示意图

Fig. 2 Schematic diagram of clipped double Q-learning method

使用剪切双 Q 学习不会引入任何高估偏差，但会引入一些低估偏差。相比高估偏差，低估偏差不会被传播，因此其对学习效果的影响远小于高估对学习效果的影响。

### 3.3 随机加权三重 Q 学习方法

在一些特殊情况下，取两个 Q 估计值的最小值并不能很好地起到降低高估的效果，甚至还会造成低估的情况。

针对此问题，本文提出了一些改进方法。由于高估情况会被传播，因此我们在剪切双 Q 学习的基础上扩大选择 Q 值的范围，从而增加算法的探索性，间接降低高估传播现象出现的概率。同时，将 Q 值的选择范围限制在一个比较合理的范围内，从而提高算法的利用效率。

为解决上述问题，本文提出了随机加权三重 Q 学习方法。RWTQ 方法使用了 3 个 Q 值函数来进一步解决算法学习中的高低估问题，其更新目标计算式如下：

$$\begin{aligned} Q_{\min} &= \min_{i=1,2,3} Q_{\theta_i'}(s',\pi_\phi(s')) \\ Q_{\text{average}} &= \frac{1}{3}\sum_{i=1,2,3} Q_{\theta_i'}(s',\pi_\phi(s')) \\ y &= r+\gamma(\rho Q_{\min}+(1-\rho)Q_{\text{average}}) \end{aligned} \tag{10}$$

其中，$Q_{\min}$ 表示 3 个 Q 估计值中的最小值，$Q_{\text{average}}$ 表示 3 个 Q 估计值的平均值，参数 $\rho$ 是一个加权参数，符合一个均匀分布，即 $\rho\in U(0,1)$。

如图 3 所示，RWTQ 方法在每一次计算更新目标时，都会对 3 个 Q 估计值的最小值和平均值进行随机加权。随机加权后的结果会均匀地分布在 3 个 Q 估计值的平均值和最小值之间。随机加权方法的随机性使得每一个动作值都有一定的概率会被小范围可控地高估或者低估，增大了每一个动作被选择的概率，极大地降低了高估传播现象出现的概率。

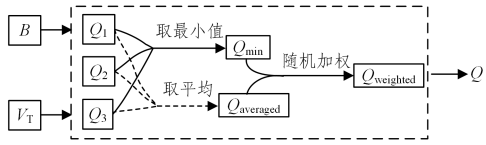



图 3 随机加权三重 Q 学习方法示意图

Fig. 3 Schematic diagram of randomly weighted triple Q-learning method

CDQ 方法较好地解决了高估问题，但同时引入了一系列的低估问题，所以 CDQ 方法所选择的两个 Q 估计值的最小值较为接近真实 Q 值。我们用 $Q_1$，$Q_2$ 和 $Q_3$ 来代表 3 个 Q 估计值。这 3 个 Q 估计值中的最小值一定小于等于其中任意两个 Q 估计值中的最小值，即：

$$\min_{i=1,2,3} Q_i \leqslant \min(Q_j,Q_k) \tag{11}$$

其中，$j,k\in\{1,2,3\}$ 且 $j\neq k$。因此，$Q_{\min}\leqslant Q_{\text{true}}$。实际上，3 个 Q 估计值的平均值与单个 Q 估计值没有区别。相比真实的 Q 值，由于存在高估问题，3 个 Q 估计值的平均值大于等于真实 Q 值，即 $Q_{\text{average}}\geqslant Q_{\text{true}}$，因此 3 个 Q 估计值的平均值与最小值之间的范围可以很好地近似真实 Q 值，即 $Q_{\min}\leqslant Q_{\text{true}}\leqslant Q_{\text{average}}$。

在此范围内，通过均匀分布随机加权的方式计算 Q 值可以较好地对状态动作空间进行探索，从而提升算法的探索效率，也可以充分利用已有数据，获取最优策略。

### 3.4 SAC-RWTQ 算法描述

SAC 算法也采用了 CDQ 方法来减少更新过程中出现的高估现象。将 RWTQ 方法与 SAC 算法相结合可以在利用最大熵的优秀学习效果的同时，进一步减小高低估现象对算法学习效果的影响。本文结合 RWTQ 和 SAC 算法提出了基于随机加权三重 Q 学习的软行动者-评论家算法，该算法的效果优于当前大部分基于连续状态空间和连续动作空间的深度强化学习算法。

**算法 1** SAC-RWTQ 算法

输入：最大时间步 M，每步更新次数 N，初始化策略网络参数 $\varphi$，Q 函数参数 $\theta_1$，$\theta_2$，$\theta_3$，V 函数参数 $\psi$，经验回放池 D

输出：策略网络参数 $\varphi$

1. 初始化目标网络参数：$\psi_{\text{targ}}\leftarrow\psi$

2. for m←1 to M do：
3. 先观察当前状态 s，然后根据策略选择动作 a～π(·|s)
4. 在环境中执行动作 a
5. 观察下一状态 s′、奖赏 r 和表示 s′状态是否为结束状态信号 d
6. 将元组(s,a,r,s′,d)存储到经验池 $\mathcal{D}$ 中
7. 如果状态 s′为终止状态，重置环境状态
8. if 经验池 $\mathcal{D}$ 中有足够的经验：
9. for n←1 to N do：
10. 从 $\mathcal{D}$ 中随机取样一小批过程 B={(s,a,r,s′,d)}
11. 计算 Q 值与 V 值的目标值：

$$y_q(r,s',d)=r+\gamma(1-d)V_{\psi_{targ}}(s')$$

$$Q_{min}=\min_{i=1,2,3}Q_{\theta_i'}(s',\pi_\phi(s'))$$

$$Q_{average}=\frac{1}{3}\sum_{i=1,2,3}Q_{\theta_i'}(s',\pi_\phi(s'))$$

$$Q_{weighted}=\rho Q_{min}+(1-\rho)Q_{average}$$

$$y_v(s)=Q_{weighted}-\alpha\log\pi_\phi(\tilde{a}|s)\ \tilde{a}\sim\pi_\varphi(\cdot|s),\ \rho\sim U(0,1)$$

12. 通过一步梯度下降更新 Q 函数：

$$\nabla_{\theta_i}\frac{1}{|B|}\sum_{(s,a,r,s',d)\in B}(Q_{\theta_i}(s,a)-y_q(r,s',d))^2,\text{for i}=1,2,3$$

13. 通过一步梯度下降更新 V 值函：

$$\nabla_\psi\frac{1}{|B|}\sum_{s\in B}(V_\psi(s)-y_v(s))^2$$

14. 通过一步梯度上升更新策略参数：

$$\nabla_\varphi\frac{1}{|B|}\sum_{s\in B}(Q_{\theta_1}(s,\tilde{a}_\phi(s))-\alpha\log\pi_\phi(\tilde{a}_\varphi(s)|s))$$

其中，$\tilde{a}_\varphi(s)$是从策略 a～π(·|s)取样而来，而这个策略在使用重参数化技巧的情况下对 φ 是可微的

15. 使用软更新来更新目标值网络参数：

$$\psi_{targ}\leftarrow\rho\psi_{targ}+(1-\rho)\psi$$

16. end for
17. end if
18. end for

SAC-RWTQ 算法的流程如图 4 所示。

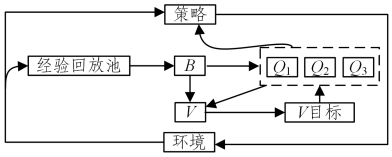



图 4 SAC-RWTQ 算法的示意图

Fig. 4 Schematic diagram of SAC-RWTQ algorithm

## 4 实验与结果

本节将本文提出的 SAC-RWTQ 算法应用到 Mujoco 的一些环境中。结果表明，在这些环境中，SAC-RWTQ 算法都取得了优于对比算法的结果。

### 4.1 实验设置

我们首先取 5 个随机种子，然后记录每 1 000 步的奖赏值，最后根据 5 个随机实验的平均值绘制曲线，以反映算法的学习速度和效果。

各种算法在对应 Mujoco 任务中执行的步数如表 1 所列。其中 Humanoid-v2 任务需要 1 000 万步左右的训练才趋于收敛。

表 1 Mujoco 任务下的执行步数

Table 1 Number of steps executed under Mujoco tasks

| Mujoco Task | Steps |
|---|---|
| HalfCheetah-v2 | 1 000 000 |
| Hopper-v2 | 1 000 000 |
| Walker2d-v2 | 4 000 000 |
| Humanoid-v2 | 10 000 000 |

SAC 算法和 SAC-RWTQ 算法的参数如表 2 所列。Humanoid-v2 任务中的温度系数 $\alpha$ 为 0.05，其他任务中的温度系数 $\alpha$ 均为 0.2。

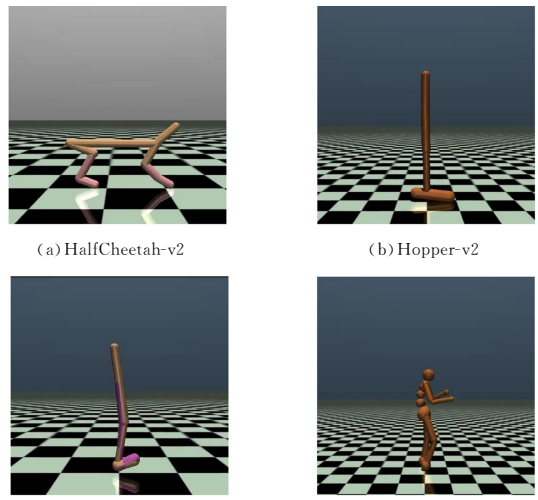
表 2 SAC 算法和 SAC-RWTQ 算法的参数

Table 2 Parameters of SAC and SAC-RWTQ

| Parameters | Value |
|---|---|
| Discount Factor $\gamma$ | 0.99 |
| Target Smoothing Coefficient $\tau$ | 0.005 |
| Learning Rate | 0.000 3 |
| Temperature Parameter $\alpha$ | 0.2/0.05 |
| Batch Size | 256 |
| Replay Buffer Size | 1 000 000 |

### 4.2 实验环境

本节在 gym[21] 实验平台 Mujoco[22] 模拟环境中的 HalfCheetah-v2，Walker2d-v2，Humanoid-v2 和 Hopper-v2 任务上进行实验。如图 5(a)所示，HalfCheetah-v2 任务是让一个二维的猎豹机器人跑得尽可能快，此任务可以较为稳定地展示各种算法之间的学习效果的差异；如图 5(b)所示，Hopper-v2 任务是让一个二维的单腿机器人跑得尽可能快；如图 5(c)所示，Walker2d-v2任务是让一个二维的双腿机器人尽可能快地向前走。如图 5(d)所示，Humanoid-v2 任务是让一个三维的双腿机器人在不摔倒的情况下尽可能快地向前走。这 4 个任务中，Humanoid-v2 任务最难，一般来说需要的训练步数最多。



(a) HalfCheetah-v2 (b) Hopper-v2

(c) Walker2d-v2 (d) Humanoid-v2

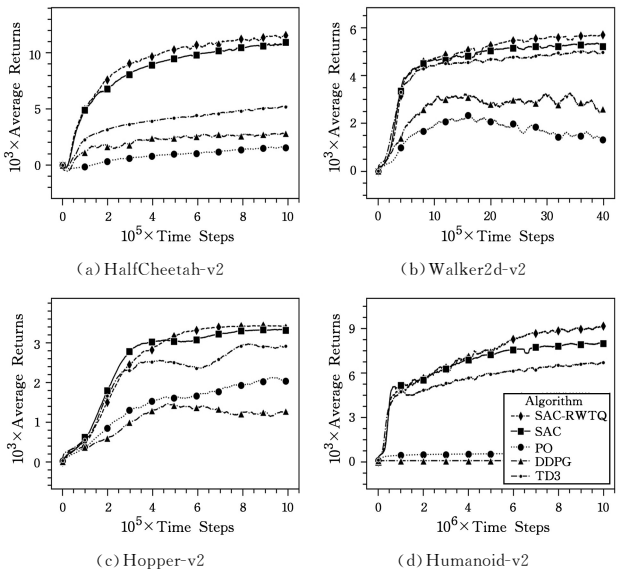
图 5 Mujoco 环境中的 4 个任务

Fig. 5 Four tasks in Mujoco environment

### 4.3 实验结果

为了验证 SAC-RWTQ 算法的可行性和高效性，将实验分为两组。

第一组实验是将 SAC-RWTQ 算法的学习效果与 SAC 算法、DDPG 算法、PPO 算法以及 TD3 算法的学习效果进行对比，实验结果如图 6 所示。图中的实验性能曲线已经过平滑处理。




(a) HalfCheetah-v2 (b) Walker2d-v2

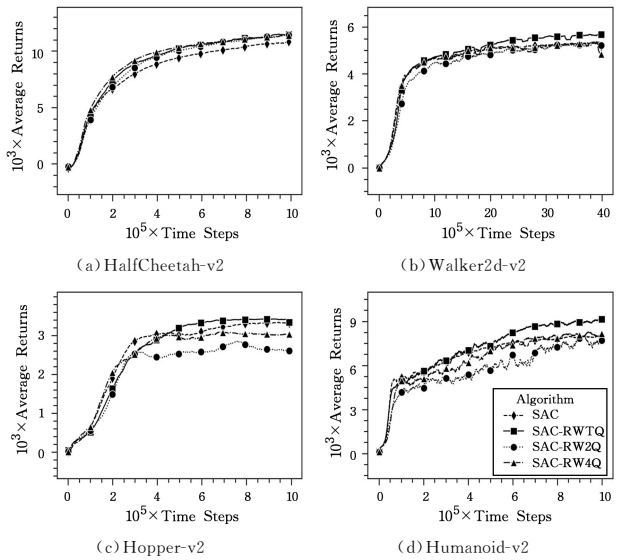
(c) Hopper-v2 (d) Humanoid-v2

图 6 SAC-RWTQ 算法和其他算法在 Mujoco 任务上的性能曲线

Fig. 6 Performance curves of SAC-RWTQ and other algorithms on Mujoco tasks

可以看出，4 个 Mujoco 任务中的每一种算法的平均回报均已收敛，SAC-RWTQ 算法在 4 个任务中均获得了最优的收敛效果，在相同的时间步的情况下，SAC-RWTQ 算法的平均回报的收敛值均优于次优算法 SAC。

第二组实验是将 SAC-RWTQ 算法的学习效果与 SAC 算法、SAC-RW2Q 算法、SAC-RW4Q 算法的学习效果进行比较，实验结果如图 7 所示。其中 SAC-RW2Q 算法采用了随机加权二重 Q 学习方法的软行动者-评论家算法。随机加权二重 Q 学习方法中，目标值由 2 个 Q 估计值的最小值与 2 个 Q 估计值的平均值进行随机加权计算。SAC-RW4Q 算法是采用了随机加权四重 Q 学习方法的软行动者-评论家算法。随机加权四重 Q 学习方法中，目标值由 4 个 Q 估计值的最小值与平均值进行随机加权计算。




(a) HalfCheetah-v2 (b) Walker2d-v2

(c) Hopper-v2 (d) Humanoid-v2

图 7 SAC，SAC-RWTQ，SAC-RW2Q 以及 SAC-RW4Q 在 Mujoco 任务上的性能曲线

Fig. 7 Performance curves of SAC, SAC-RWTQ, SAC-RW2Q and SAC-RW4Q on Mujoco tasks

4 个 Mujoco 任务中，SAC-RWTQ 算法最后的收敛效果均优于原 SAC 算法，说明 SAC-RWTQ 可以有效地解决在 SAC 算法中出现的高低估问题，并提高 SAC 算法的学习效果。

在 HalfCheetah-v2 任务中，3 种随机加权算法均优于 SAC 算法，这说明随机加权方法可以提高对环境的探索与利用水平，从而提升了算法的效果。在 Humanoid-v2 任务中，SAC-RWTQ 算法的效果明显优于其他 3 种算法，说明基于随机加权三重 Q 学习方法优于两重 Q 以及四重 Q 的随机加权方法。

在 4 个任务中，SAC-RWTQ 算法的收敛效果均优于 SAC-RW2Q 与 SAC-RW4Q 算法，说明相比随机加权二重 Q 学习与随机加权四重 Q 学习，随机加权三重 Q 学习方法可以更好地逼近真实 Q 值，从而进一步缓解算法存在的高低估问题，获得更好的效果。

表 3 列出了 SAC-RWTQ 算法与其他对比算法在 4 个 Mujoco 任务上平均回报的最大平均值以及最大值。

表 3 Mujoco 任务上的实验效果

Table 3 Experimental data on Mujoco tasks

| Task | Algorithm | Mean Value | Maximum Value |
|---|---|---|---|
| HalfCheetah-v2 | SAC | 10709.93 | 11237.76 |
| | SAC-RWTQ | **11380.58** | **11786.80** |
| | DDPG | 2713.98 | 3116.91 |
| | PPO | 1506.34 | 1715.42 |
| | TD3 | 5063.66 | 5342.82 |
| | SAC-RW2Q | 11322.30 | 11697.48 |
| | SAC-RW4Q | 11299.84 | 11601.23 |
| Walker2d-v2 | SAC | 5284.16 | 5487.08 |
| | SAC-RWTQ | **5633.46** | **5809.92** |
| | DDPG | 2876.67 | 4969.16 |
| | PPO | 1482.81 | 3230.07 |
| | TD3 | 4959.61 | 5292.69 |
| | SAC-RW2Q | 5257.34 | 5652.01 |
| | SAC-RW4Q | 5222.29 | 5493.32 |
| Hopper-v2 | SAC | 3323.42 | 3481.39 |
| | SAC-RWTQ | **3372.52** | **3559.88** |
| | DDPG | 1217.13 | 2648.66 |
| | PPO | 2079.46 | 3048.79 |
| | TD3 | 2891.64 | 3440.67 |
| | SAC-RW2Q | 2634.42 | 3405.89 |
| | SAC-RW4Q | 3046.30 | 3570.60 |
| Humanoid-v2 | SAC | 7938.33 | 8246.74 |
| | SAC-RWTQ | **8975.51** | **9646.28** |
| | DDPG | 72.17 | 81.34 |
| | PPO | 579.93 | 645.80 |
| | TD3 | 6575.25 | 7069.56 |
| | SAC-RW2Q | 7566.84 | 9307.77 |
| | SAC-RW4Q | 8096.29 | 8896.04 |

在 4 个 Mujoco 任务上，SAC-RWTQ 算法均取得了最好的成绩，说明随机加权三重 Q 学习方法确实解决了 SAC 算法中的高低估问题，并且提升了算法的探索能力。在 Hopper-v2 任务上，虽然 SAC 算法在前 40 万步的学习效果略优于 SAC-RWTQ 算法，但是收敛效果劣于 SAC-RTWQ 算法。在 HalfCheetah-v2 任务中，所有的随机加权算法相比原 SAC 算法均有更好的收敛效果，这说明随机加权学习方法可以更好地解决高低估问题，并且可以提升算法的探索能力，从而获得更优的策略。在 Hopper-v2，Walker2d-v2 和 Humanoid-v2

这 3 个任务中,三重 Q 网络的随机加权方法的效果明显优于两重 Q 网络以及四重 Q 网络的随机加权方法。在 HalfCheetah-v2 任务中,使用随机加权方法的算法收敛效果相当,均好于 SAC 算法。在 4 个任务中,SAC-RWTQ 算法在学习过程中获得的奖赏最大值均为最优值,说明随机加权三重 Q 学习方法确实解决了 SAC 算法中的高低估问题,并且提升了算法的探索能力。

综上所述,相比对比算法,SAC-RWTQ 算法表现出了更快的收敛速度以及更好的收敛效果。

**结束语** 本文提出了一种新的基于随机加权三重 Q 学习的连续动作空间强化学习优化方法,并将其与最大熵异策略强化学习算法相结合,提出了一种新的基于随机加权三重 Q 学习方法的软行动者-评论家强化学习算法。在连续动作空间任务中进行实验,结果表明,本文提出的基于随机加权三重 Q 方法的最大熵异策略强化学习算法在收敛速度以及收敛效果方面均优于对比算法,因此本文方法是可行且有效的。

## 参 考 文 献


[1] SUTTON R S,BARTO A G. Reinforcement Learning:An Introduction[M]. Massachusetts:MIT Press,2018.

[2] HUA J,ZENG L,LI G,et al. Learning for a Robot:Deep Reinforcement Learning,Imitation Learning,Transfer Learning[J]. Sensors,2021,21(4):1278.

[3] SILVER D,SCHRITTWIESER J,SIMONYAN K,et al. Mastering the Game of Go without Human Knowledge[J]. Nature,2017,550(7676):354-359.

[4] ARTHUR L,SAMUE L. Some Studies in Machine Learning Using the Game of Checkers [J]. IBM Journal of Research and Development,2000,44(1/2):206-226.

[5] CHEN J P,ZOU F,LIU Q,et al. A Reinforcement Learning Algorithm Based on Generative Adversarial Networks[J]. Theoretical Computer Science,2019,46(10):265-272.

[6] WATKINS C,DAYAN P. Technical Note Q-Learning[J]. Machine Learning,1992,8:279-292.

[7] SUTTON R S. Learning to Predict by the Method of Temporal Differences[J]. Machine Learning,1988,3(1):9-44.

[8] GOODFELLOW I,BENGIO Y,COURVILLE A. Deep Learning [M]. Massachusetts:MIT Press,2016.

[9] MNIH V,KAVUKCUOGLU K,SILVER D,et al. Playing Atari with Deep Reinforcement Learning [J]. arXiv: 1312. 5602, 2013.

[10] HASSELT H V,GUEZ A,SILVER D. Deep Reinforcement Learning with Double Q-Learning[C]// Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, 2016:2094-2100.

[11] LILLICRA T P,HUNT J J,PRITZEL A,et al. Continuous Control with Deep Reinforcement Learning[C]// Proceedings of the 4th International Conference on Learning Representations. ICLR,2016.

[12] FUJIMOTO S,HOOF H V,MEGER D. Addressing Function Approximation Error in Actor-Critic Methods [C]// International Conference on Machine Learning. PMLR, 2018: 1587-1596.

[13] HAARNOJA T,ZHOU A,ABBEEL P,et al. SOFT ACTOR-CRITIC: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor[C]// Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018: 1856-1865.

[14] SCHULMAN J,LEVINE S,ABBEEL P,et al. Trust Region Policy Optimization[C]// International Conference on Machine Learning. PMLR,2015:1889-1897.

[15] SCHULMAN J,WOLSKI F,DHARIWAL P,et al. Proximal Policy Optimization Algorithms [J]. arXiv:/1707. 06347,2017.

[16] MNIH V,BADIA A P,MIRZA M,et al. Asynchronous Methods for Deep Reinforcement Learning[C]// International Conference on Machine Learning. PMLR,2016:1928-1937.

[17] HASSELT H. Double Q-learning [J]. Advances in Neural Information Processing Systems,2010,23:2613-2621.

[18] RUDER R. An Overview of Gradient Descent Optimization Algorithms[J]. arXiv:1609. 04747,2016.

[19] KINGMA D P,BA J. Adam:A Method for Stochastic Optimization[J]. arXiv:1412. 6980,2014.

[20] THRUN S,SCHWARTZ A. Issues in using Function Approximation for Reinforcement Learning [C]// Proceedings of the Fourth Connectionist Models Summer School. Erlbaum, 1993: 255-263.

[21] BROCKMAN G,CHEUNG V,PETTERSSON L,et al. Openai Gym[J]. arXiv:1606. 01540,2016.

[22] TODOROV E,EREZ T,TASSA Y. MuJoCo:A Physics Engine for Model-based Control[C]// Intelligent Robots and Systems. IEEE,2012:5026-5033.


**FAN Jing-yu**, born in 1995, postgraduate. His main research interests include deep reinforcement learning and so on.

**LIU Quan**, born in 1969, Ph.D, professor, is a member of China Computer Federation. His main research interests include deep reinforcement learning and automated reasoning.

(责任编辑:李亚辉)