基于优化子目标数的 Option-Critic 算法

刘成浩[1)] 朱 斐[1),2)] 刘 全[1),2)]

[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)

摘 要 时间抽象是分层强化学习中的重要研究方向,而子目标是时间抽象形成的核心元素.目前,大部分分层强化学习需要人工给出子目标或设定子目标数量.然而,在很多情况下,这不仅需要大量的人工干预,而且所作设定未必适合对应场景,在动态环境未知的指导下,这一问题尤为突出.针对此,提出基于优化子目标数的 Option-Critic 算法(Option-Critic algorithm based on Sub-goal Quantity Optimization,OC-SQO),增加了智能体对环境的探索部分,通过与环境的简单交互,得到适用于应用场景的初始子目标数量估值,并在此基础上识别子目标,然后利用通过策略梯度生成对应的抽象,使用初态、内部策略和终止函数构成的三元组表示,以此进行训练,根据交互得到的抽象改变当前状态,不断迭代优化.OC-SQO 算法可以在任意状态下开始执行,不要求预先指定子目标和参数,在执行过程中使用策略梯度生成内部策略、抽象间策略和终止函数,不需要提供内部奖赏信号,也无需获取子目标的情况,尽可能地减少了人工干预.实验验证了算法的有效性.

关键词 分层深度强化学习;时间抽象;子目标;强化学习;Option

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2021.01922

Option-Critic Algorithm Based on Sub-Goal Quantity Optimization

LIU Cheng-Hao[1)] ZHU Fei[1),2)] LIU Quan[1),2)]

[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)

Abstract Reinforcement learning has been extensively studied as a branch of machine learning, where an agent keeps interacting with the environment with the goal of getting maximal long-term return, making it prominent in areas such as control and optimal scheduling. Deep reinforcement learning (DRL) is designed to handle large-scale high-dimensional data such as video and image by extracting the abstract representation, and learning an optimal1 policy through reinforcement learning component. Deep reinforcement learning has become a research hotspot in artificial intelligence and a lot of algorithms have been developed. For example, deep Q Network (DQN) is one of the most famous models in deep reinforcement learning, which is based on convolutional neural network (CNN) and Q-learning algorithm and has been used to learn policy in complex environments with high dimensional inputs. However, the DQN failed to perform well in sparse reward environment or with large-scale state space. Hierarchical reinforcement learning was introduced to solve the aforementioned problems where the initial problem space is decomposed into several sub-problem spaces, and the initial large problem is solving by meaning of dealing with each sub-problem individually. However, hierarchical reinforcement learning tends to be

收稿日期:2020-08-02;在线发布日期:2021-04-25.本课题得到国家自然科学基金项目(61303108,61772355)、江苏省高校自然科学研究项目重大项目(17KJA520004)、苏州市重点产业技术创新-前瞻性应用研究项目(SYG201804)、江苏高校优势学科建设工程资助项目(PAPD)资助.**刘成浩**,硕士研究生,主要研究方向为深度强化学习.E-mail:20185227050@stu.suda.edu.cn.**朱 斐**(通信作者),博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为强化学习、文本挖掘和模式识别.E-mail:zhufei@suda.edu.cn.**刘 全**,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为自动推理和机器学习.

effective in tasks with discrete state/action space. The idea of hierarchical deep reinforcement learning, by combining hierarchical reinforcement learning with deep learning, is similar to that of hierarchical reinforcement learning, where it solves sub-problems through the neural network. Time abstraction is an important concept in hierarchical reinforcement learning, and the sub-goal is the key for producing time abstraction. Time abstraction, as one of the most promising areas of hierarchical reinforcement learning, requires the notion of sub-goal as the prerequisite. At present, however, sub-goals or the number of sub-goals must be manually specified, which is in short of automation and generalization across different scenarios. To solve the problem, we propose Option-Critic algorithm based on Sub-goal Quantity Optimization (OC-SQO) that enables the agent to explore the environment, decide the initial sub-goal number, identify sub-goals, and derive the corresponding abstraction through policy gradient. The abstraction is denoted as a triple consisting of the initial state, the internal policy and the terminate function, by which the agent optimizes the policy. The advantage is that the OC-SQO requires no particular initial state, sub-goals or parameters; moreover, it doesn't require internal reward signal since the abstraction is produced during the learning process as mentioned. Based on the above process, OC-SQO reduces a lot of manual intervention compared with the OC method, and achieves better results with only a small amount of manual intervention. The experiment results indicate the effectiveness of OC-SQO algorithm.

Keywords　hierarchical deep reinforcement learning; time abstraction; sub-goal; reinforcement learning; Option

《计算机学报》

1　引　言

深度强化学习是一种强化学习和深度学习结合了的接近人类思维方式的人工智能方法. 在深度 Q 网络(Deep Q Network,DQN)提出之后[1],出现了多种改进的算法. 但是,很多强化学习方法固有的不足仍未得到有效地解决,如难以处理大规模任务和稀疏奖赏任务[2]. 分层强化学习[3]的抽象机制[4]可以用于处理这些问题. 自动发现抽象动作是分层强化学习研究热点. Kulkarni 等人将深度学习与分层强化学习结合,提出了分层深度强化学习[5],通过构建层次式的决策模型和设立"小目标"的形式来缓解稀疏奖赏的问题. Florensa 等人提出的针对分层强化学习的随机网络将任务分成两层[6],在预先训练的环境中学习到较好技能,并将其应用在分层任务中. 而上述方法都是建立在最优价值的基础上,关键在于对价值函数的估计,然后通过价值函数来获得对应的策略[7],其存在后期围绕最优价值函数持续小幅度震荡而不收敛、价值函数计算复杂等问题. 还有一种基于策略的方法,通过直接计算策略可能更新的方向来学习策略,如经典策略梯度算法直接通过梯度来更新策略[8]. Konda 等人在策略梯度的基础之上,提出了 Actor-Critic(AC)算法[9],其中 Actor 部分使用策略梯度方法来负责生成动作并与环境交互,Critic 部分采用价值函数的方法来负责评估 Actor 的表现,并指导 Actor 下一阶段的动作. Schulman 等人提出的置信区域策略优化(Trust Region Policy Optimization,TRPO)算法可以确保策略模型在优化时单调提升[10]. 之后又提出了一种近端策略优化(Proximal Policy Optimization,PPO)算法,通过化简目标函数并且使用梯度下降法来加快学习速度[11]. Vezhnevets 等人提出了用于学习宏动作的策略注意力作家(Strategic Attentive Writer,STRAW)模型,设计了一个深度循环神经网络体系结构,通过与强化学习环境中的环境交互以端对端的方式学习构建隐式策略[12].

虽然基于抽象的方法在解决大规模的任务时表现较好[13],但也存在着很多难点需要解决,如何自动发现子目标就是其中之一. 基于策略的方法具有直接学习策略等方面的优势,但是在解决大规模问题时有可能会失效. Option-Critic(OC)算法结合了这两类算法[14],该方法无需知道子目标的具体情况,也无需获取额外的辅助奖赏就可以学习到

Option. 在 OC 算法中，Option 由两个重要元素所组成，一个是 Option 的内部策略，另一个是 Option 的终止函数，二者均通过策略梯度方法来学习得到. OC 算法也存在不足，如 Option 数量的设定问题：合适的抽象数量可以提高算法性能，但需要预先给定抽象的数量值，而经典 OC 算法并没有指出如何确定最佳的 Option 数量，这就导致 OC 算法无法发挥最优性能.

针对此，本文提出了基于优化子目标数的 Option-Critic（Option-Critic algorithm based on Sub-goal Quantity Optimization，OC-SQO）算法. 通过与环境进行交互得到初始的目标数，以此生成对应数量的 Option；这些 Option 与环境交互，得到奖赏值，反馈给算法进而优化子目标数量，求得最优解或近似最优解所对应的子目标数. 本文方法仅需简单地设定子目标数量的上限值，而无需其他设定，解决了经典 OC 算法无法确定合适抽象数量的问题，也改善了效果、提升了算法性能.

2 背景知识

2.1 深度强化学习

强化学习以试错式的学习方式与环境交互，目标是寻求能得到最大长期回报的策略，其包括状态、动作、奖赏以及环境等基本元素. 强化学习方法可以用马尔可夫决策过程（Markov Decision Process，MDP）来建模[15]，由状态集 S、动作集 A、转移函数 $P:S\times A\rightarrow(S\rightarrow[0,1])$ 和立即奖赏函数 $r:S\times A\rightarrow\mathbb{R}$ 所组成. 强化学习的智能体与环境进行交互，接收环境中的状态，然后根据当前状态采取动作，进入下一个状态. 一个马尔可夫策略是给定状态下的动作的概率分布，$\pi:S\times A\rightarrow[0,1]$. 强化学习方法常用价值函数或状态动作值函数来评估智能体表现. 在使用带折扣累计奖赏方式计算时，状态 s 下的策略 π 的价值函数 V 定义为

$$V_\pi(s)=E_\pi\Big[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\,|\,s_0=s\Big] \tag{1}$$

状态 s 下采取动作 a 的状态动作值函数 Q 为

$$Q_\pi(s,a)=E_\pi\Big[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\,|\,s_0=s,a_0=a\Big] \tag{2}$$

其中，$\gamma\in[0,1)$ 是折扣因子，E 表示期望，r 为立即奖赏.

如果策略 $\pi(s,a)>0$ 且仅当 $a=\arg\max_{a'}Q(s,a')$ 时，对于给定的状态动作值函数 Q 而言，π 是贪心策略. 在离散的 MDP 中，至少包含一个最优策略[16]. 由于强化学习的目标是最大化长期回报期望，上述方法与寻找最优策略目的一致. 这样，找到了最优策略及可得到最大化长期回报期望.

大多数经典强化学习算法面向离散空间任务，所能处理的动作空间和状态空间都不大，一旦将相关算法扩展到连续空间中，效果难以保证. 近年来，研究人员把深度学习中的高维输入与强化学习结合起来，提出了深度强化学习，改善了强化学习在连续空间任务中的表现. 由 DeepMind 提出的深度 Q 网络（Deep Q Network，DQN）是深度强化学习的基本方法[1]，其采用一个深度网络的方式来代表价值函数，根据强化学习中的 Q-learning 算法来为深度网络提供目标值，对网络不断更新直至收敛. 深度强化学习在 DQN 的基础之上进行不断改进，涌现了诸多方法，也取得了很多成果.

深度强化学习在多个方面有着应用. DeepMind AlphaStar 打败了星际争霸人类高手；DeepMind 在一款多人抢旗游戏中达到了人类玩家水平；OpenAI Five 打败了人类选手；OpenAI 训练了类人机器手 Dactyl，用于灵活地操纵实物；DeepMind 模拟人形机器人，掌握高难度的运动技能；谷歌云的自动机器学习提供了自动优化神经网络结构设计的服务；谷歌研发了基于强化学习的 YouTube 视频推荐算法；亚马逊与英特尔合作，发布了一款强化学习实体测试平台 AWS DeepRacer；滴滴出行则把强化学习应用于派单等业务.

2.2 策略梯度

Andrychowicz 等人通过执行随机梯度下降来优化给定参数化的随机策略 π_θ 的性能目标[17]，从而逐渐得到较优策略. 策略梯度求解给出了平均奖赏和折扣奖赏目标相对于 θ 的梯度的表达式. 在带有着折扣的设置中，目标的定义是由开始的状态或状态分布 s_0：$J(\theta,s_0)=E_{\pi_\theta}\Big[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\,|\,s_0\Big]$ 所决定的. 策略梯度求解方法为

$$\frac{\partial P(\theta,s_0)}{\partial\theta}=\sum_s\mu_{\pi_\theta}(s\,|\,s_0)\sum_a\frac{\partial\pi_\theta(a,s)}{\partial\theta}Q_{\pi_\theta}(s\,|\,a) \tag{3}$$

其中，$\mu_{\pi_\theta}(s\,|\,s_0)=\sum_{t=0}^{\infty}P(s_t=s\,|\,s_0)$ 是从 s_0 开始沿轨道的状态的折扣权重.

在实际应用中，策略梯度是基于策略平稳分布

《计算机学报》

的样本估计获得[18]，其在某组参数的基础上，使用独立的函数逼近器来近似求解随机策略. 例如，策略可以使用一个神经网络来表示，神经网络的输入是状态的表示，输出的是动作选择概率，权值是策略的参数[19]. 策略梯度算法广泛应用于具有连续动作空间的强化学习问题[20]. 在某个状态下根据参数 θ 随机选择动作 a. 策略梯度算法是通过对这种随机策略来进行抽样，并根据累计收益的增加来调整策略参数. 策略梯度方法分为两步：第一步计算梯度，第二步更新参数 θ 以提高选择对的动作的概率.

2.3　Option 框架

基于时间抽象的 Option 框架[21]是现阶段分层强化学习领域中常见的一种框架. 在使用马尔可夫表示一个 Option $o\in\Omega$ 时，使用三元组 (I_o,π_o,β_o)，其中 $I_o\in S$ 是初始状态集，π_o 是内部的 Option 策略，终止函数 $\beta_o:S\rightarrow[0,1]$ 用来表示在当前状态下 Option 终止的概率[22]. 在使用 Option 时，有如下前提：$\forall s\in S,\forall o\in\Omega:s\in I_o$，即 Option 可以在任意一个状态下开始.

Gasparetto 提出了具有一组 Option 的 MDP 的方法，即半马可科夫决策过程(Semi-Markov Decision Process，SMDP)，以及关于 Options 的值函数 $V_\Omega(s)$ 和 Option-值函数 $Q_\Omega(s,o)$. MDP 的学习和规划算法在环境中都有对应的算法[23]，而底层 MDP 的存在提供了学习不同 Option 的可能. Option o 的执行过程如下：一个智能体在 $I_o\in S$ 处选择 o，然后按照 o 的策略 π_o 来执行动作，直到满足终止条件 $\beta_o=1$ 为止. 如果执行的 o 结束了，就可以选择其他的 o，直至完成任务. 这种抽象不仅可以利用先验知识来加快学习速度[24]，也可以加速学习到相关任务的迁移.

2.4　Option-Critic(OC)框架

2.4.1　框架介绍

研究人员提出了 Option-Critic(OC)框架[14]，无需识别发现子目标也可对任务进行分层处理. Option-Critic 框架将所有可用的经验提炼到组成框架的每一个部分，包括值函数、Option 间的策略、Option 内部策略和 Option 终止函数. Option 的执行模型如下：智能体根据 Options 间的策略 π_Ω 选择一个 Option o，然后根据 Option 的内部策略 π_o 选择 Option 的内部状态，直到满足 Option 的终止条件，重复上述步骤. OC 框架使用 $\pi_{o,\theta}$ 表示参数化后的 Option 内部策略，使用 $\beta_{o,\vartheta}$ 表示参数化后的 Option 终止条件，使用策略梯度方法来优化参数化后的内部策略和终止条件. 假设目标是直接优化带有折扣的回报[25]，则期望所有的轨迹从一个确定的状态 s_0 和 Option o_0 开始，于是有：

$$\rho(\Omega,\theta,\vartheta,s_0,o_0)=E_{\Omega,\theta,o}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\mid s_0,o_0\right]\tag{4}$$

根据式(4)可知，该期望依赖于 Option 间的策略，以及 Option 内部的参数和终止条件的参数. 于是目标就转换成了关于参数 θ 和 ϑ 的梯度的求解问题. OC 框架提出了 Option 价值函数计算方法：

$$Q_\Omega(s,o)=\sum_a\pi_{o,\theta}(a\mid s)Q_U(s,o,a)\tag{5}$$

其中，$Q_U:S\times\Omega\times A\rightarrow R$ 是在某个状态、选择某个 Option 后，采取动作之后产生的总收益，其计算方法为

$$Q_U(s,o,a)=r(s,a)+\gamma\sum_{s'}P(s'\mid s,a)U(o,s')\tag{6}$$

其中，(s,o) 是增广状态空间. 函数 $U:\Omega\times S\rightarrow R$ 是到达某个状态后，执行 Option 的值. 在进入状态 s' 后执行 Option o 的值为

$$U(o,s')=(1-\beta_{o,\vartheta}(s'))Q_\Omega(s',o)+\beta_{o,\vartheta}(s')V_\Omega(s')\tag{7}$$

Q_U 和 U 分别依赖于参数 θ 和 ϑ. OC 框架用来推导策略梯度需要的一个因素是马尔可夫链，它是在增广状态空间下定义的链. 如果 Option o_t 在 t 时刻、状态 s_t 下已经被启动或者正在被执行，然后在一步中转变到新的状态 (s_{t+1},o_{t+1}) 的概率为

$$P(s_{t+1},o_{t+1}\mid s_t,o_t)=\sum_a\left(\pi_{o_t,\theta}(a\mid s_t)P(s_{t+1}\mid s_t,a)\left((1-\beta_{o_t,\vartheta}(s_{t+1}))1_{o_t=o_{t+1}}+\beta_{o_t,\vartheta}(s_{t+1})\pi_\Omega(o_{t+1}\mid s_{t+1})\right)\right)\tag{8}$$

记 $P(s_{t+1},o_{t+1}\mid s_t,o_t)$ 为 $P_\gamma^{(1)}(s_{t+1},o_{t+1}\mid s_t,o_t)$，则从 (s_t,o_{t-1}) 转移到 (s_{t+1},o_t) 的概率为

$$P_\gamma^{(1)}(s_{t+1},o_t\mid s_t,o_{t-1})=\left((1-\beta_{o_{t-1}}(s_t))1_{o_t=o_{t-1}}+\beta_{o_{t-1}}(s_t)\pi_\Omega(o_t\mid s_t)\right)\cdot\sum_a\pi_{o_t}(a\mid s_t)\gamma P(s_{t+1}\mid s_t,a)\tag{9}$$

根据式(8)和式(9)可以推出在增广状态下的 k 步转移概率为

$$P_\gamma^{(k)}(s_{t+1},o_{t+k}\mid s_t,o_t)=\sum_{s_{t+1}}\sum_{o_{t+1}}\left(P_\gamma^1(s_{t+1},o_{t+1})P_\gamma^{(k-1)}(s_{t+k},o_{t+k}\mid s_{t+1},o_{t+1})\right)\tag{10}$$

$$P_\gamma^{(k)}(s_{t+k},o_{t+k-1}\mid s_t,o_{t-1})=\sum_{s_{t+1}}\sum_{o_t}\left(P_\gamma^{(1)}(s_{t+1},o_t\mid s_t,o_{t-1})P_\gamma^{(k-1)}(s_{t+k},o_{t+k-1}\mid s_{t+1},o_t)\right)\tag{11}$$

然后，推导目标函数对参数 θ 的梯度[26]. 假设其随机且可微，根据前面的两个公式，便可以推

计算机学报

导出：

$$\frac{\partial Q_\Omega(s,o)}{\partial_\theta}=\left(\sum_a \frac{\partial \pi_{o,\theta}(a|s)}{\partial\theta}Q_U(s,o,a)\right)+\sum_a \pi_{o,\theta}(a|s)\sum_{s'}\gamma P(s'|s,a)\frac{\partial U(o,s')}{\partial\theta} \quad (12)$$

等式右边可以用式(6)和式(7)展开，从而得到内部策略梯度求解方法.

2.4.2 内部策略的策略梯度求解

给定参数为 θ 的随机内部策略的马尔可夫 Options. 对于参数 θ 和初始条件 (s_0, o_0) 的期望折扣奖赏[27]的梯度是：

$$\sum_{s,o}\mu_\Omega(s,o|s_o,o_o)\cdot\sum_a \frac{\partial \pi_{o,\theta}(a|s)}{\partial\theta}Q_U(s,o,a) \quad (13)$$

其中，$\mu_\Omega(s,o|s_0,o_0)$ 是从 (s_0, o_0) 开始沿轨迹的 state-Option 对的折扣权重：

$$\mu_\Omega(s,o|s_0,o_0)=\sum_{t=0}^{\infty}\gamma^t P(s_t=s,o_t=o|s_0,o_0) \quad (14)$$

该梯度描述了原始水平上的局部变化对全局的期望折扣奖赏的影响.

然后可以计算终止函数[28]的梯度，即终止函数对参数 ϑ 的梯度. 从前面的公式中，可以推导：

$$\frac{\partial U(o,s')}{\partial\vartheta}=\sum_a \pi_{o,\theta}(a|s)\sum_{s'}\gamma P(s'|s,a)\frac{\partial(o.s')}{\partial\vartheta} \quad (15)$$

从公式中可以看出，关键的量是 U 的梯度，可以进一步的展开为

$$\frac{\partial U(o,s')}{\partial\vartheta}=-\frac{\partial\beta_{o,\vartheta}(s')}{\partial\vartheta}A_\Omega(s',o)+\gamma\sum_{o'}\sum_{s''}P(s'',o'|s',o)\frac{\partial U(o',s'')}{\partial\vartheta} \quad (16)$$

其中，A_Ω 是 Options 上的优势函数[29]，$A_\Omega=Q_\Omega(s',o)-V_\Omega(s')$. 递归地展开 $\frac{\partial U(o,s')}{\partial\vartheta}$ 得到与 Option 内部策略求解的公式类似，其中的 state-Option 对的权重根据马尔可夫链 $\mu_\Omega(s_{t+1},o_t|s_t,o_{t-1})$ 移动了一个时间步长.

2.4.3 终止条件策略梯度求解

给定一组参数为 ϑ 的马尔可夫 Options，这些 Options 的随机终止函数是可微的，对于参数 ϑ 和初始条件 (s_1, o_0) 的期望折扣回报的梯度为

$$-\sum_{s',o}\mu_\Omega(s',o|s_1,o_0)\frac{\partial\beta_{o,\vartheta}(s')}{\partial\vartheta}A_\Omega(s',o) \quad (17)$$

其中，$\mu_\Omega(s',o|s_1,o_0)$ 是从 (s_1,o_0) 开始的 state-Option 对的折扣权重：

$$\mu_\Omega(s',o|s_1,o_0)=\sum_{t=0}^{\infty}\gamma^t P(s_{t+1}=s,o_t=o|s_1,o_0) \quad (18)$$

优势函数通常出现在策略梯度方法之中，用来形成一个基线来减少梯度估计的方差. 在这里的优势函数主要与算法的设计相关. 当选择到的 Option 比其他所有的 Option 的预期收益的期望值低时，则优势函数是负的，然后就会产生一个终止的概率增高的梯度，当 Option 终止之后，通过外部的策略 π_Ω 来选择一个更好的 Option.

3 算法描述与分析

3.1 算法描述

针对 OC 框架中需要人为指定子目标数量的问题，本文提出了基于优化子目标数的 Option-Critic (OC-SQO)算法. 通过与环境进行交互得到子目标[30]数量，优化该数量从而得到一个最优情况下的数量，然后生成对应抽象，使得算法性能得到提升. 算法在对应离散问题和连续问题时，分别采用不同的方法估计得到子目标的大概数量. 此估计值不一定是一个完全精确的子目标的数量，将共用于执行 OC 算法，根据 OC 算法得到的结果，优化该估计值，得到最适合的子目标数量，并以此为条件进行子目标的自动识别，使 OC 算法得到一个较优的效果.

基于最优子目标数量的 Option-Critic 模型的框架，如图 1 所示.

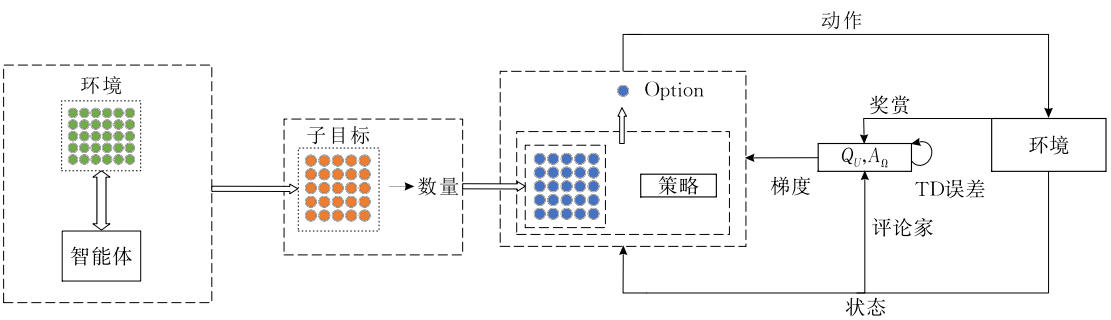

图 1 基于最优子目标数量的 Option-Critic 模型框架

《计算机学报》

从图 1 可知，智能体需要与环境进行交互，然后得到一个子目标的数量 m，OC 框架对应生成 m 个 Option. OC-SQO 算法根据优势函数以及 Actor-Critic 算法优化每个 Option，智能体在采用这些 Option 的情况下再次与环境进行交互，针对不同的问题，考虑当前的 m 个 Option 是否能够高效快速地解决问题，如果可以，则保留当前值为最优数值并改变 m 的值，否则，直接改变 m 的值. 通过改变 m 的值来判断当前的 m 的值是否更优，如当前得到的效果值比之前更好，更新最优的数量，反之，最优值在历史取值之中. 从而找到一个最佳的 m 值用来处理对应的问题，其中 c 值根据环境设定.

3.2　算法分析

OC-SQO 算法根据 Option 内部策略的策略梯度求解方法和 Option 的终止条件策略梯度求解方法，采用随机梯度下降的方法学习 Option. 在学习 Option 时，使用较快的速率更新 Option 的内部策略，使用较慢的速率更新 Option 的终止函数. OC 算法类似于 Actor-Critic 算法，其中，Option 的内部策略、终止函数和外部策略是系统中的 Actor 部分，而 Q_u 和 A_Ω 则是系统中的 Critic 部分. OC 算法并没有指出 Option 之间的策略 π_Ω 是如何得到的，如果 π_Ω 是关于 Option 的贪心策略[31]，则 Q_u 的公式可以推导出对应的一步异策略更新目标 g_t：

$$g_t = r_{t+1} + \gamma\Big((1-\beta_{o_t,\vartheta}(s_{t+1}))\cdot \sum_a \pi_{o_t,\theta}(a|s_{t+1})Q_U(s_{t+1},o_t,a) + \beta_{o_t,\vartheta}(s_{t+1})\cdot \max_o \sum_a \pi_{o,\theta}(a|s_{t+1})Q_U(s_{t+1},o,a)\Big) \tag{19}$$

式(19)是 Option 内部策略的 Q-learning 算法的更新目标，α、α_θ 和 α_ϑ 分别是 Critic、Option 内部策略和终止条件的学习率.

从图 1 中可以发现，确定子目标大致数量的过程需要智能体与环境进行交互得到. 在交互过程中，并不需要额外的工作来加快智能体到达目标点的速度，经典的强化学习中所固有的智能体与环境的交互即可满足条件，使其到达目标点，循环执行此过程，依次到达子目标点，最后到达终点. 智能体与环境交互到达终点的过程，重复此过程若干次，在交互过程中可以发现，每次完成任务的过程中，有些点是必须经过的，这些点的数量即对应子目标的数量. 由于此过程的处理较为粗糙，得到的子目标数量和真实子目标个数存在偏差，因此用这个数值来表示子目标的数量的估计值，而非确切的子目标数量值. 基于优化子目标数的 Option-Critic 算法(Option-Critic algorithm based on Sub-goal Quantity Optimization，OC-SQO)的处理流程如算法 1 所示.

算法 1.　基于优化子目标数的 Option-Critic 算法(Option-Critic algorithm based on Sub-goal Quantity Optimization，OC-SQO).

输入：最多子目标数 c，最多交互次数 d

1. 初始化：全局数组 *sum* 的所有位置上的值为 0，交互次数 *count* 为 0
2. While $count < d$：
3. $s \leftarrow s_0$
4. While s 不是终态：
5. 根据策略 $\pi(a|s)$ 选择一个动作 a
6. 执行动作 a，观测状态 s'
7. If s' 不是终态：
8. $s \leftarrow s'$
9. 根据状态出现的次数改变 *sum* 对应位置上的值
10. $count \leftarrow count + 1$
11. 根据 *sum* 得到子目标数 m(*sum* 的最大值的个数)
12. If m 未超过子目标数上限：
13. $m \leftarrow 2$
14. Else：
15. $m \leftarrow c - m$
16. For $i \leftarrow m$ To c：
17. $s \leftarrow s_0$
18. 根据策略 $\pi_\Omega(s)$ 选择一个 Option o
19. Repeat：
20. 根据内部策略 $\pi_{o,\vartheta}(a|s)$ 选择动作 a
21. 在状态 s 处选择动作 a，观测 s' 和 r
22. $\delta \leftarrow r - Q_U(s,o,a)$
23. If s' 不是终态
24. $\delta \leftarrow \delta + \gamma\Big((1-\beta_{o,\vartheta}(s'))Q_\Omega(s',o) + \beta_{o,\vartheta}(s')\max_{\bar{o}} Q_\Omega(s',\bar{o})\Big)$
25. $Q_U(s,o,a) \leftarrow Q_U(s,o,a) + \alpha\delta$
26. $\theta \leftarrow \theta + \alpha_\theta \dfrac{\partial \log \pi_{o,\theta}(a|s)}{\partial \theta} Q_U(s,o,a)$
27. $\vartheta \leftarrow \vartheta - \alpha_\vartheta \dfrac{\partial \beta_{o,\vartheta}(s')}{\partial \vartheta}(Q_\Omega(s',o) - V_\Omega(s'))$
28. IF $\beta_{o,\vartheta}$ 在 s' 处终止
29. 根据 $\pi_\Omega(s')$ 选择一个新的 o
30. $s \leftarrow s'$
31. Until s' 是终态且 i 个 Option 均已完成学习
32. 根据累计折扣奖赏或者达到目标点需要花费的最小步数得到最优 Option 数

3.3　复杂度分析

首先，对离散环境下算法的时间复杂度进行分析. 算法第 2 步到第 15 步是用于得到子目标数的算法，时间复杂度受到交互次数的影响，同时也受到智能体到达环境中终点的影响. 假设最好情况下智能体到达终点的时间复杂度为 $O(D)$，其中 D 为智能

计算机学报

体到达终点所耗费的时间，这部分的时间复杂度为 $O(d*D)$. 算法第 19 步到第 31 步为学习 Option 以及优化子目标数的过程，其时间复杂度与智能体和环境交互所花费的时间相关，即为 $O(D)$. 在最多有 c 个子目标的情况下，智能体到达最后的目标点的时间复杂度为 $O(c*D)$. 综上可知，时间复杂度由智能体到达终点所花费的时间决定，整体的时间复杂度为 $O((c+d)*D)$.

然后，对连续环境下算法的时间复杂度进行分析. 在连续环境下，由于神经网络的加入，其时间复杂度主要由神经网络中的卷积层和全连接层决定，为 $O(c*B*T)$，其中 c 为最多的子目标数量，B 为经验池大小，T 为网络的复杂度.

4 实验与分析

本文分别在离散环境下和连续环境下进行实验. 在离散环境下，将 OC-SQO 算法与 Q-learning 算法、SARSA 算法和 OC 算法进行对比，其中 Q-learning 算法和 SARSA 算法是强化学习中的经典算法，而 OC 算法是 OC-SQO 算法的基础. 在连续环境下，将 OC-SQO 算法与 DQN 算法、PPO 算法和 OC 算法进行对比，其中 DQN 算法是深度强化学习算法的起源，PPO 是 DQN 算法的改进.

4.1 离散控制问题

四房间问题是分层强化学习中的一个经典离散控制问题[32]. 经典四房间问题的终点是固定的，本文实验增加了问题的难度，从右下方的房间中随机选择一个位置作为终点. 如图 2 所示，智能体从 s 处出发，探索经过右上角的房间或者左下角的房间，到达终点.

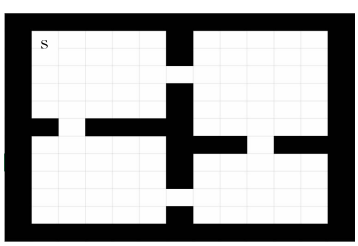

图 2 四房间问题，s 为起点，终点为右下方某个随机位置

智能体在开始的时候，移动的方向是随机的，在训练的过程中，逐渐改变各方向移动的概率. 折扣因子为 0.99，到达目标的奖赏为 1，没有到达目标的奖赏为 0. Option 间的策略采用的是内部 Option Q-learning 方式. 本文将 OC-SQO 算法和 Q-learning 算法、SARSA 算法以及 Option-Critic 算法进行比较. 在实验中，所有的权重初始化为 0. 由于在四房间问题中，这 4 种算法均可以到达最后的目标点，在这种情况下对比 4 种算法在对应情节下得到的累计奖赏值，不太容易能看出各种算法的区别，而通过对比到达目标点所需的步数，可以看出各种算法的表现差异显著.

图 3 给出了 1000 个情节内各算法下终止时所需最多步数. 与其余 3 个算法相比，OC-SQO 算法的收敛效果更好. OC 的算法仅次于 OC-SQO 算法，SARSA 算法和 Q-learning 算法的效果欠佳，主要原因是 Q-learning 算法的值函数在更新的时候选取的是最大化 Q 的方向，而 SARSA 选择的是一条比较安全的路. 所以在没有危险的情况下，Q-learning 算法比 SARSA 算法更具有优势. OC 算法在策略梯度的情况下学习到比较好的 Option 后，可以得到一个很好的效果，而 OC-SQO 算法是在 OC 算法的基础之上进行优化，因此效果会更好. 在 Option 的设置中，Option 的数量是一个很重要的因素，针对上述的问题域，子目标的数量决定了算法的优劣，图 3 也可以验证这点. 因此，找到合适的 Option 数量，可以使算法达到一个更好的效果.

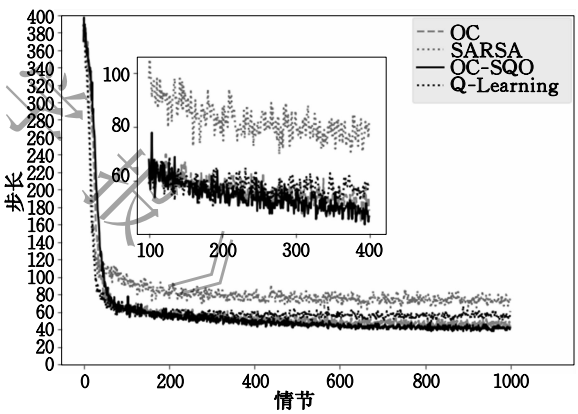

图 3 1000 情节内各算法到达终点所需要的步数(参与对比的 4 种算法分别对应四种不同的曲线. 各算法可以达到收敛的效果. 图中放大的部分为曲线重叠的部分)

4.2 连续控制问题

采用深度神经网络来估计 Critic 及 Option 的内部策略和 Option 的终止条件函数，其深度神经网络结构图如图 4 所示.

从图 4 可以看出，网络结构与 DQN 类似，采用三层的卷积网络. 第一层是 32 个大小为 8×8、步长为 4 的卷积滤波器，第二层是 64 个大小为 3×3、步长为 2 的滤波器，第三层是 64 个大小为 3×3，步长为 1 的滤波器，接着将输出结果输入到一个 512 个神经元组成的共享层.

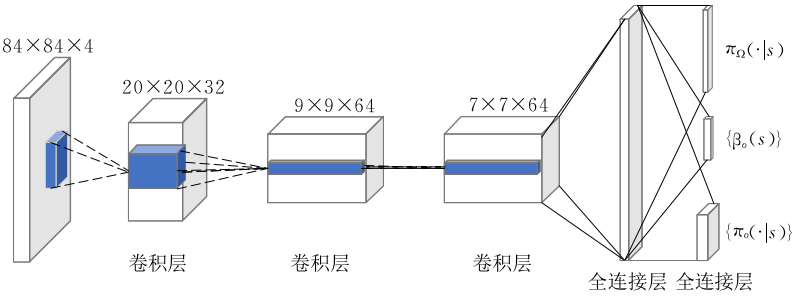

图 4　深度神经网络结构图

本文采用的实验环境是 Atari 游戏平台. 为了和其他的几个算法进行比较，所有算法的参数设置都是一致的. 由于在 Atari 游戏平台中，每个游戏可以获得的奖赏可能会很高，各算法在每个情节下所获得的奖赏值有差异，这与离散环境下每个算法获得的奖赏值一致有所不同. 本文记录了实验在每次情节下花费若干步长取得的奖赏，最后再将取得的奖赏根据情节取一个平均值. 从实验结果上看，OC 算法和 OC-SQO 算法都学习到了 Option，对比二者可以发现，OC-SQO 算法效果更佳.

4.2.1　Asterix 实验及分析

在 Asterix 实验环境中，智能体的任务是通过上下移动来获得高奖赏. 如图 5 所示，智能体上下移动，吃掉有奖赏的物体就会增加智能体的得分，从而使奖赏变的更大，而如果碰到移动的障碍物，便会失败. 在这个实验环境下，智能体的任务是尽可能少失败的前提下，获得更好的奖赏.

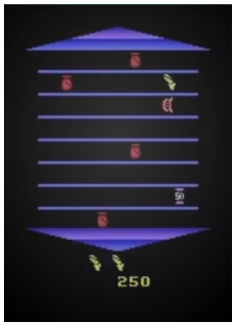

图 5　Asterix 环境示意图

DQN 算法和 PPO 算法直接通过训练得到结果，不需要产生 Option，而另外两种算法都是需要产生 Option. 在 OC 算法中，生成 Option 的数量均为 8 个，而在 OC-SQO 中，形成 Option 的数量和通过探索后得到的最优的子目标数量相同，所对应的实验结果要优于 OC 算法的. 图 6 给出了 4 种算法的实验效果.

从图 6 中可以看出，具有 Option 的两个算法 OC-SQO 和 OC 效果明显优于 DQN 算法和 PPO 算法，这是因为 Option 的产生加速了学习，使得智能体的表现变得更好. 这与前文所分析一致：适合数量的 Option 有助于算法获得好的效果，而生成 Option 的数量需要根据具体环境来确定. DQN 算法在表现上不如其他 3 种算法；PPO 算法是 DQN 算法的改进，其表现一直很好；OC-SQO 算法和 OC 算法相对于 DQN 来说也做了许多改进，效果也很好. 由于 OC-SQO 算法得到的 Option 数量和只有固定 Option 数量的 OC 算法相比，OC-SQO 得到了更优的 Option 数量，相比之下，OC 算法指定的 Option 的数量在效果上就会差很多，因此 OC-SQO 算法比 OC 算法的效果更佳. 综上可知，无论在何种环境中，OC-SQO 算法都得到了比较可靠的结果.

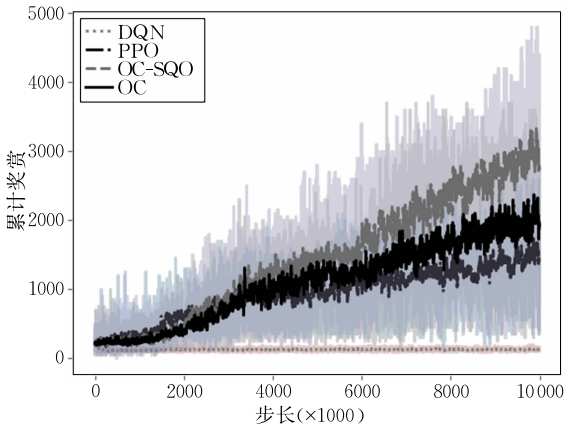

图 6　Asterix 实验效果图(纵坐标为四种算法下的智能体在对应步长下所获得的奖赏值. PPO、OC 和 OC-SQO 都稳步上升. 在有好的 Option 的情况下，OC-SQO 会更好的效果)

4.2.2　Breakout 实验及分析

在 Breakout 实验环境中，通过移动下端的挡板来控制智能体，使智能体打掉上方的方块. 如果打掉了方块，智能体就会获得分数，从而使奖赏变的更高，如果智能体没有打掉方块就下来的话，就需要让挡板来挡住智能体，如果没有挡住智能体，使智能体死亡，则游戏结束，反之，就会使智能体一直打掉方块. 本实验下的任务是智能体打掉尽可能多的方块从而获得更高的奖赏. Breakout 实验环境如图 7 所示.

计算机学报

图 7 Breakout 实验环境示意图

图 8 给出了 Breakout 实验中 4 种算法的效果. 从图中可以看出 OC 算法和 OC-SQO 算法效果接近;虽然 DQN 效果比较稳定但一直不如其他 3 种算法;PPO 算法则是处于一个稳定上升的状态,而且 PPO 算法的稳定性较好,在前 8000 个情节中,PPO 算法优于另外 3 种算法的,表明了 PPO 算法是一个较为优秀的算法;OC 算法和 OC-SQO 算法在当前的实验环境下处于一个相接近的水平,从图 8 中可以分析得到,这两个算法在前 6000 个情节中表现一致,但是从 6000 个情节后,二者表现开始有所差别,这说明两个算法都学习到了 Option,学习速度开始变快,两个算法迅速上升并且都超过了 PPO. 总体而言,OC-SQO 算法是略优于 OC 算法的,并且 OC 算法的波动性比 OC-SQO 高. 综上可以看出,OC-SQO 算法总体表现优于其他 3 种算法.

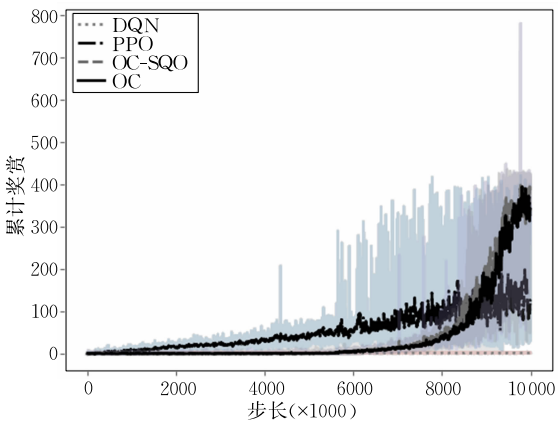

图 8 Breakout 实验效果(纵坐标为四种算法的智能体在对应步长下所获奖赏值. DQN 和 PPO 算法曲线较平缓上升,而 OC 和 OC-SQO 在得到对应的 Option 后效果迅速上升)

另外,从图 8 中还可以分析得出,具有 Option 的两个算法 OC-SQO 和 OC 效果仍然明显优于 DQN 算法和 PPO 算法,原因在于 Option 的产生加速了学习,使得智能体的表现变得更好. DQN 算法表现不令人满意,而 PPO 算法前期效果略好,但后期明显差了很多. 而 OC 算法和 OC-SQO 算法相比,效果很接近,然而 OC 算法的波动比较大,但并不影响 OC 算法的性能. 这是因为在此环境下,Option 数量的差别仅会造成收敛过程中的波动,不会影响最终效果.

4.2.3 Assault 实验及分析

在 Assault 实验环境中,控制智能体左右移动躲避上方敌人的攻击,还可以通过控制智能体发射子弹来击杀敌人获得奖赏. 如果智能体发射的子弹击落了上方的敌方飞船,就会获得一个奖赏. 如果智能体被上方发射的子弹击杀,则智能体死亡. 智能体的实验次数是 3,如果全部用完,游戏结束. 在本实验中,通过控制智能体击杀敌人来获得一个高奖赏值,如果智能体彻底死亡,则游戏结束. 当前实验下的任务是让智能体在当前环境下能够获得一个高奖赏值. Assault 实验环境如图 9 所示.

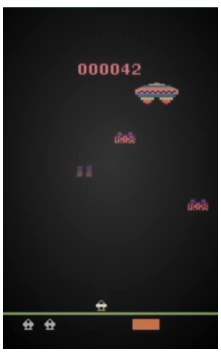

图 9 Assault 实验环境示意图

图 10 给出了 Assault 实验的 4 种算法的效果图. 从图 10 可以看出,两个 OC 算法的实验效果明显要比 DQN 算法和 PPO 算法优越很多. OC 算法和 OC-SQO 算法的实验效果接近,但从图中曲线可以看出,OC-SQO 算法的实验效果要略优于 OC 算法的实验效果,且 OC 算法效果的波动大于 OC-SQO 算法,因此可知 OC-SQO 算法的稳定性优于 OC 算法. 由于利用了 Option 机制,将若干单步动作封装成了一个多步动作,使智能体不需要每一步都要进行探索,只要根据 Option 内部策略即可,可以加速学习的效果,故而,基于 Option 的 OC 算法和 OC-SQO 算法的实现效果要好于其他两个算法更好. DQN 算法效果一般,PPO 算法优于 DQN 算法. 虽然在前期,PPO 算法好于 OC 算法和 OC-SQO 算法,但随着训练的持续开展,PPO 算法效果明显不如这两个算法. OC 算法和 OC-SQO 算法效果接近,但 OC 算法的波动较大. 在此实验环境下,由于 Option 数量的影响,OC 算法没有很强的增长;OC-SQO 算法是优于 OC 算法,且更为稳定.

《计算机学报》

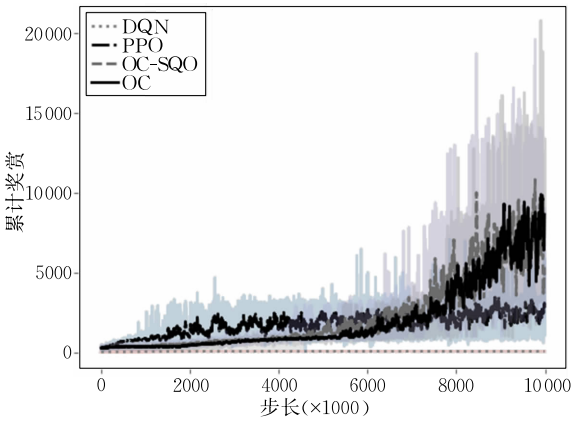

图 10　Assault 的实验效果图(纵坐标为 4 种算法对应的智能体在对应步长下所获得的奖赏值. 在 OC 和 OC-SQO 的曲线中，OC-SQO 的波动性比 OC 要好一些)

4.2.4　连续控制问题分析

表 1 给出了各模型在 Atari 游戏中的测试结果.

表 1　各模型在 Atari 游戏中的测试结果

游戏名	模型	平均奖赏	最大奖赏
Asterix	DQN	127.21	270
	PPO	936.79	2900
	OC-SQO	**1450.06**	**4800**
	OC	1107.02	3700
Breakout	DQN	2.62	6
	PPO	55.20	422
	OC-SQO	**44.85**	**781**
	OC	39.68	436
Assault	DQN	53.00	70
	PPO	1768.00	7157
	OC-SQO	**2292.00**	19606
	OC	2027.00	**20801**

从表 1 中可以看出，OC-SQO 方法模型的测试效果最佳，但平均奖赏在 Breakout 游戏中低于 PPO 方法. OC-SQO 算法在 OC 算法的基础之上得到了进一步的调整，找到一个更为恰当的 Option 数量，从而提高算法的效果. 虽然在 Assault 实验环境中，OC-SQO 算法的最大奖赏略低于 OC 算法，但 OC-SQO 的平均值高于 OC 算法，且 OC 算法的波动性更大. 总体而言，OC-SQO 算法优于其他算法.

4.2.5　子目标数量分析

表 2 给出了各个实验环境中所获得的最佳子目标数. 可以看出，随着环境的改变，最佳子目标数量是不一致的. 因此，预先设定子目标数量的方式是存在固有缺陷的. 本文算法自动获得实验环境所对应最优子目标数或近似最优子目标数，从而确定Option 数量，无需通过人工设置.

表 2　不同环境对应的最佳子目标数量

类型	环境	最佳子目标数
离散	四房间问题	5
连续	Asterix	4
	Breakout	5
	Assault	7

和其他算法相比，OC-SQO 算法具备的优越性. 在 Atari 平台下，OC-SQO 的适用范围与其他 3 种算法一致的，且效果更好. 虽然与人工指定 Option 数量的方法相比，OC-SQO 算法计算复杂度有所提高，但一旦获得最优数量后，其效果最佳. 其他方式，如人工指定 Option 数量、根据图像来获取等方法，都存在局限性. 例如，通过人工指定的方式来获取，很难确定最优 Option 数量；通过图像来获取的方法会在离散域下失效. 综上分析可知，基于优化子目标数的 OC-SQO 方法，通过与环境交互，求得最佳子目标数或近似最优子目标数，以增加计算的代价获得更好的效果，是一种较好的方法.

5　结　语

OC 算法是一种基于策略梯度和分层深度强化学习的算法，不需要额外的辅助奖赏和精确的子目标划分，只需指定需要生成的 Option 数量即可. 通过使用策略梯度对 Option 中的内部策略和终止函数进行学习. 但是 OC 算法所需要的 Option 的数量并不确定. 人工指定 Option 数量的方式会影响最终效果. 针对这个问题，本文提出了一种基于子目标数量的 Option-Critic 算法，解决了一般分层强化学习需要精确划分子目标的情况，同时也解决了 OC 算法无法在最佳 Option 数量上执行的问题.

参　考　文　献

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

[2] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge, USA: MIT Press, 2018: 47-71

[3] Nachum O, Gu S S, Lee H, et al. Data-efficient hierarchical reinforcement learning//Advances in Neural Information Processing Systems. Montreal, Canada, 2018: 3303-3313

[4] Sutton R S, Precup D, Singh S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 1999, 112(1-2): 181-211

[5] Kulkarni T D, Narasimhan K, Saeedi A, et al. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation//Advances in Neural Information Processing Systems. Barcelona, Spain, 2016: 3675-3683

《计算机学报》

[6] Florensa C, Duan Y, Abbeel P, et al. Stochastic neural networks for hierarchical reinforcement learning//Proceedings of the International Conference on Learning Representations. Toulon, France, 2017: 1-17

[7] Baxter J, Bartlett P L. Infinite-horizon policy-gradient estimation. Journal of Artificial Intelligence Research, 2019, 15(1): 319-350

[8] Luo Y, Chiu C, Jaitly N, et al. Learning online alignments with continuous rewards policy gradient//Proceedings of the International Conference on Acoustics, Speech, and Signal Processing. New Orleans, USA, 2017: 2801-2805

[9] Konda V R, Tsitsiklis J N. On Actor-Critic algorithms. SIAM Journal on Control and Optimization, 2003, 42(4): 1143-1166

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

[11] Liu B, Cai Q, Yang Z, et al. Neural trust region/proximal policy optimization attains globally optimal//Advances in Neural Information Processing Systems. Vancouver, Canada, 2019: 10565-10576

[12] Vezhnevets A, Mnih V, Agapiou J, et al. Strategic attentive writer for learning macro-actions//Advances in Neural Information Processing Systems. Barcelona, Spain, 2016: 3486-3494

[13] Al-Emran M. Hierarchical reinforcement learning: A survey. International Journal of Computing and Digital Systems, 2015, 4(2): 137-143

[14] Bacon P, Harb J, Precup D, et al. The Option-critic architecture //Proceedings of the National Conference on Artificial Intelligence. San Francisco, USA, 2017: 1726-1734

[15] Hansen T M, Chong E K P, Suryanarayanan S, et al. A partially observable Markov decision process approach to residential home energy management. IEEE Transactions on Smart Grid, 2016, 9(2): 1271-1281

[16] Gobet E, Turkedjiev P. Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions. Mathematics of Computation, 2016, 85(299): 1359-1391

[17] Andrychowicz M, Denil M, Gomez S, et al. Learning to learn by gradient descent by gradient descent//Advances in Neural Information Processing Systems. Barcelona, Spain, 2016: 3981-3989

[18] Tsurumine Y, Cui Y, Uchibe E, et al. Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. Robotics and Autonomous Systems, 2019, 112: 72-83

[19] Dong C, Loy C C, Tang X. Accelerating the super-resolution convolutional neural network//Proceedings of the European Conference on Computer Vision. Amsterdam, The Netherlands: Springer, 2016: 391-407

[20] Haarnoja T, Tang H, Abbeel P, et al. Reinforcement learning with deep energy-based policies//Proceedings of the International Conference on Machine Learning. Sydney, Australia, 2017: 1352-1361

[21] Machado M C, Bellemare M G, Bowling M. A Laplacian framework for Option discovery in reinforcement learning//Proceedings of the International Conference on Machine Learning. Sydney, Australia, 2017: 2295-2304

[22] Zhang S, Whiteson S. DAC: The double Actor-Critic architecture for learning Options//Advances in Neural Information Processing Systems. Vancouver, Canada, 2019: 2012-2022

[23] Gasparetto A, Boscariol P, Lanzutti A, et al. Path planning and trajectory planning algorithms: A general overview//Carbone G, Gomez-Bravo F, eds. Motion and Operation Planning of Robotic Systems. Berlin, Germany: Springer, 2015: 3-27

[24] Jiang Y, Gu S S, Murphy K P, et al. Language as an abstraction for hierarchical deep reinforcement learning//Advances in Neural Information Processing Systems. Vancouver, Canada 2019: 9419-9431

[25] Bozkurt A K, Wang Y, Zavlanos M M, et al. Control synthesis from linear temporal logic specifications using model-free reinforcement learning//Proceedings of the 2020 IEEE International Conference on Robotics, Control and Automation (ICRA). Seoul, Korea, 2020: 10349-10355

[26] Lamaazi H, Benamar N. A comprehensive survey on enhancements and limitations of the RPL protocol: A focus on the objective function. Ad Hoc Networks, 2020, 96: 102001

[27] Miryoosefi S, Brantley K, Daume III H, et al. Reinforcement learning with convex constraints//Advances in Neural Information Processing Systems. Vancouver, Canada, 2019: 14093-14102

[28] Zhang Z, Zhuang L, Zhou W, et al. Dynamic cascaded regression network with reinforcement learning for robust face alignment//Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME). Shanghai, China, 2019: 598-603

[29] Peng B, Li X, Gao J, et al. Adversarial advantage Actor-Critic model for task-completion dialogue policy learning//Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Seoul, South Korea, 2018: 6149-6153

[30] Raza S J A, Lin M. Constructive policy: Reinforcement learning approach for connected multi-agent systems//Proceedings of the 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Vancouver, Canada, 2019: 257-262

[31] Efroni Y, Merlis N, Ghavamzadeh M, et al. Tight regret bounds for model-based reinforcement learning with greedy policies//Advances in Neural Information Processing Systems. Vancouver, Canada, 2019: 12224-12234

[32] Arumugam D, Karamcheti S, Gopalan N, et al. Grounding natural language instructions to semantic goal representations for abstraction and generalization. Autonomous Robots, 2019, 43(2): 449-468

《计算机学报》

LIU Cheng-Hao, M. S. candidate. His main research interest is deep reinforcement learning.

ZHU Fei, Ph. D., associate professor. His main research interests include reinforcement learning, text mining, and pattern recognition.

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

Background

Deep reinforcement learning is currently a research focus in the field of artificial intelligence. However, as the environment gets more complex, standard deep reinforcement learning method turns to be ineffective in some cases. The hierarchical deep reinforcement learning emerges as the times require, where the Option that uses prior knowledge to speed up learning, is a very important concept.

In order to generate good Options intelligently, this paper proposes a novel method to generate Options based on Option-Critic, referred as Option-Critic algorithm based on Sub-goal Quantity Optimization method (OC-SQO). OC-SQO method obtains the most appropriate number of Options By interacting with the environment, which avoids the problem of manually setting the number of sub-goals in OC method that could affect the performance of the algorithm. According to the experiment results, OC-SQO outperforms its opponents not only in overall performance, but also in stability. Overall, OC-SQO method can achieve the best results without setting the number of sub-goals manually.

This paper is supported by the National Natural Science Foundation of China (61303108, 61772355), the Jiangsu College Natural Science Research Key Program (17KJA520004), the Suzhou Key Industries Technological Innovation-Prospective Applied Research Project (SYG201804), A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. These projects aim to enrich the reinforcement learning theory and develop efficient approximate algorithms to expand the power and applicability of reinforcement learning on large scale problems.

《计算机学报》