基于认知行为模型的启发加速深度 Q 网络

李嘉祥 陈 浩 黄 健 张中杰
(国防科技大学智能科学学院 湖南 长沙 410073)

摘 要 由于状态-动作空间的扩大或奖励回报稀疏,强化学习智能体在复杂环境下从零开始学习最优策略将更为困难。由此提出基于智能体认知行为模型的启发加速深度 Q 网络,将符号化的规则表示融入学习网络,动态引导智能体策略学习,解决有效加速智能体学习的问题。该算法将启发知识建模为基于 BDI(Belief-Desire-Intention)的认知行为模型,用于产生认知行为知识引导智能体策略学习,设计启发策略网络在线引导智能体的动作选择。GYM 典型环境与星际争霸 2 环境下实验表明,该算法可以根据环境变化动态提取有效的认知行为知识,并借助启发策略网络加速智能体策略收敛。

关键词 强化学习 认知行为模型 启发加速深度 Q 网络

中图分类号 TP391 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.09.022

HEURISTIC ACCELERATED DEEP Q NETWORK BASED ON COGNITIVE ACTION MODEL

Li Jiaxiang Chen Hao Huang Jian Zhang Zhongjie
(*College of Artificial Intelligence*, *National University of Defense Technology*, *Changsha 410073*, *Hunan*, *China*)

Abstract Due to the expansion of the state-action space or sparse rewards of the complex environment, it is more difficult for reinforcement learning agents to learn an optimal policy from scratch. Therefore, a cognitive behavior model-based heuristic accelerated deep Q network is proposed. It incorporated symbolic rules into the learning network and guided policy learning dynamically, which solved the problem of effectively accelerating agents learning. The algorithm modeled the heuristic knowledge as a BDI-based cognitive behavior model, which was used to generate cognitive behavior knowledge to guide the agents' strategy learning. The heuristic strategy network was designed to guide the agent's action selection online. Experiments in GYM's typical environment and StarCraft II environment show that the algorithm can dynamically extract effective cognitive behavior knowledge according to environmental changes, and accelerate the agent strategy convergence with the help of heuristic strategy network.

Keywords Reinforcement learning Cognitive behavior model Heuristic accelerated deep Q network

0 引 言

当前深度强化学习(Deep Reinforcement Learning)技术已在 Atari 视频游戏和计算机围棋博弈等多种应用中取得成功[1-3]。但采样效率(Sample Efficiency)问题一直以来制约着强化学习算法在复杂问题中的应用[4]。强化学习中,智能体通过试错学习与环境交互,因此往往需要大量交互样本才能完成对状态-动作空间的充分探索,从而收敛到最优策略。特别是面对复杂任务(如高维、连续状态空间或环境奖励稀疏)时,强化学习智能体采样低效的问题尤为突出。

利用合适的先验知识或迁移已学习到的策略模型是加速强化学习智能体策略收敛的有效手段。一种可行的思路是直接重用已学到的策略,例如 Reinforcement Learning with Context Detection (RL-CD)[5]、Bayes-Pepper[6]、Deep BPR +[7] 和 Context-Aware Policy Reuse (CAPS)[8] 等。这些算法维护了一个策略库,用于记录

收稿日期:2021-04-22。国家自然科学基金项目(61906202)。**李嘉祥**,硕士生,主研领域:任务规划技术。**陈浩**,博士生。**黄健**,研究员。**张中杰**,讲师。

之前已经学习到的策略。在策略重用阶段,智能体检测环境或其他参与者的策略是否发生改变,并直接重用策略库中的策略或在此基础上继续学习。另一种思路是智能体从零学起,通过选择合适的启发知识用于策略探索,从而加速策略收敛[9]。例如,π-reuse 探索策略[10]、启发加速 Q(λ)(HA-Q(λ))[11]和 π-selection[12]算法等。以上两种思路中,启发知识在其所覆盖的状态空间直接引导智能体进行动作选择。然而,启发知识一般通过手工设计或基于案例推理获得。这些针对具体任务设计的静态知识模型,知识表示方式不统一、可迁移性和泛化性能差、灵活性低,难以动态引导学习过程[13]。

信念-愿望-意图(BDI)模型是一种通用的智能体模型架构[14],BDI 模型将人的感知、信念、意图、事件、愿望和行动等功能要素进行模块化构建,并有机融入到智能体模型中,构建出逻辑行为智能体。基于 BDI 模型的面向智能体编程(Agent-Oriented Programming, AOP)[15]是自主智能系统发展的重要方向之一。利用基于 BDI 模型的 AOP 方法,构建智能体认知行为模型,能够使智能体具有感知环境,根据内部知识库、规则库和信念库进行动态行为决策并作用于环境的能力,形成动态的可迁移的认知行为知识来源。但 AOP 智能体只进行内部逻辑推理,不具备学习能力,无法自主提升智能体的行为能力。

针对深度强化学习算法在高维、连续状态空间或稀疏奖励的环境中学习效率低下的问题,本文将启发知识建模为基于 BDI 模型的认知行为模型(Agents Cognitive Behavior Model, ACB)。用于在学习过程中动态生成认知行为知识,在线引导强化学习智能体策略收敛。为将符号化的认知行为知识与深度强化学习算法有机融合,本文设计一个启发策略网络用于在线引导智能体的动作选择。在此基础上,将启发策略网络引入深度 Q 学习网络,提出基于智能体认知行为模型的启发加速深度 Q 网络(CB-HADQN)算法,并设计启发策略网络的同步更新方式。

本文在 OpenAI GYM[16]典型连续空间强化学习环境中验证 CB-HADQN 可以高效利用认知行为知识加速策略收敛,有效缓解环境奖励稀疏对强化学习的影响,从而加速智能体学习,同时在 Deepmind 星际争霸 2 PySC2[17]强化学习环境下验证算法应对复杂问题的有效性。

1　BDI 模型与强化学习

1.1　BDI 模型

BDI 架构模仿人类行为的驱动因素,将智能体内部描述为信念、目标与计划。信念表示智能体从环境中得到的信息;目标表示智能体一切行为的目的,包括执行动作、达到状态和保持状态等;计划表示智能体对事件进行响应的手段,计划包括头部和主体,头部由条件规则组成,用以判断并触发行为,主体包含了计划实施的各种细节,包括对自身状态以及环境的影响。

智能体的运行需要解释器与执行器两个关键功能模块。解释器用以处理环境信息,同时管理并连接智能体内部状态,保证智能体运行;执行器是智能体预定执行动作的序列,同样受智能体解释器管理。BDI 模型的整体运行机制如图 1 所示。

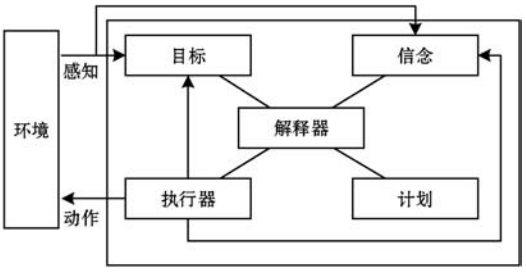


图 1　BDI 模型及运行机制

1.2　深度强化学习

强化学习问题通常可以描述为离散时间、有限的状态和动作集合的马尔可夫决策过程(Markov Decision Process, MDP)[18]问题,一般采用 $<S,A,P,R>$ 四元组描述该模型:

S 为状态空间(State Space),表示智能体可能感知到的有限的状态集。

A 为动作空间(Action Space),表示智能体在每个状态能采取的有限的动作集。

$P:S\times A\rightarrow S'$ 为状态转移函数,表示在状态 s 下执行动作 a 而转移到状态 s' 这一过程的转移概率 $P(s'|s,a)$。

$R:A\times S\rightarrow R$ 为回报函数,表示在状态 s 下执行动作 a 时,获得的环境给予的立即回报 $R(s,a)$。

智能体通过与环境的不断试错、迭代,在交互中优化自身策略。在每一步迭代中智能体感知到环境当前状态 s,执行动作 a,转移到状态 s' 并得到环境对所执行动作的反馈 r 作为强化信号,最终目的是学到最优策略。

深度 Q 网络(Deep Q Network, DQN)[1]是一种将神经网络和 Q 学习结合的强化方法,抛弃了传统的 Q 学习用表格方式记录状态和动作对应的 Q 值的方式,采用深度神经网络估计状态动作值函数,借助经验回放机制(Experience Replay)提升了强化学习算法对于复杂状态空间的适应性。DQN 内部状态如图 2 所示。

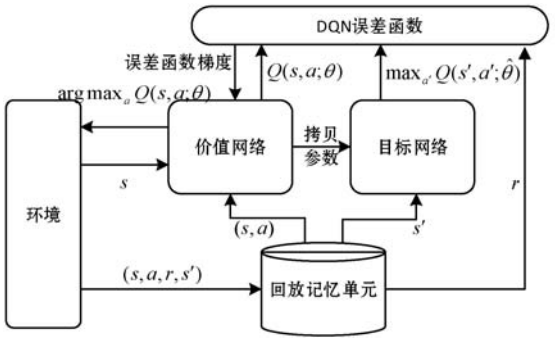


图 2 DQN 结构与训练流程

DQN 算法构建了两个 Q 网络,即价值网络 θ 与目标网络 $\hat{\theta}$,形成离线策略学习(Off-policy Learning)方式,防止算法陷入局部最优。价值网络随训练参数实时更新,目标网络间隔一定步数后获得价值网络参数并更新。训练过程中,DQN 算法计算价值网络输出与评估 Q 值的误差,并将误差反向传播,采用梯度下降方式更新整个网络参数。

2 基于认知行为知识的启发加速深度强化学习算法

ACB-DHQN 算法架构与学习流程如图 3 所示。ACB-DHQN 算法以 BDI 模型为基础,利用通用的智能体编辑语言和逻辑知识表示语言构建智能体认知行为模型,认知行为模型能够根据状态信息动态地为学习模型提供认知行为知识。为有效利用认知行为知识,算法在强化学习模块中设计启发策略网络,利用行为知识计算启发策略函数值,进而引导深度 Q 网络的学习过程。

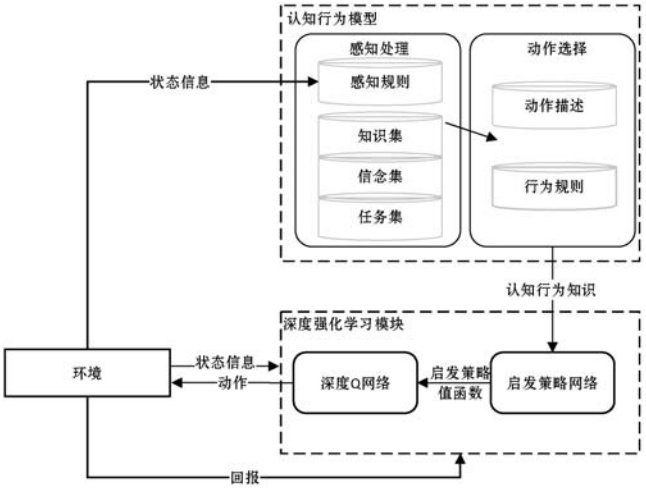


图 3 CB-HADQN 算法架构与学习流程

算法运行时,智能体的认知行为模型感知当前状态信息,通过感知规则更新认知行为模型的信念集和任务集。更新后的认知行为模型综合内部状态及其行为规则作出决策,向深度强化学习模块推荐动作。与此同时,深度强化学习模块也从环境中获取当前状态信息,输入深度强化学习网络。结合认知行为模型推荐的动作,深度强化学习模块选择在当前状态下执行的动作,从环境中获得回报并达到新的状态。

2.1 基于 BDI 架构的认知行为知识模型

本文基于 BDI 智能体模型架构,构建智能体的认知行为模型(ACB),基于逻辑编程语言 PROLOG[19] 编辑实现智能体内部状态及知识的描述,同时基于 AOP 语言最新成果之一 GOAL 语言[20],具体实现认知行为模型的功能模块。ACB 模型设计以下功能模块:

(1) 知识集:表示永久为真的客观的现实公理,知识集一般不发生改变,如声明实体及属性参数个数:

:- dynamic car/3. #car(Name,Type,Speed)

表示汽车实体,及其具备三个属性变量。声明状态事实:

:- empty(Pin) :- not(on(_, Pin)).

表示空的 Pin 指无物体位于 Pin 上。

(2) 信念集:表示智能体对于环境的理解,能够随动作执行与环境感知而改变,信念集初始为空,仅需构建无内容的信念集即可。

(3) 目标集:表示智能体期望达到的环境状态,如:

on(a,b), on(b,c), on(c,table).

表示智能体期望达成以 a、b、c、table 为顺序的环境状态。

(4) 动作集:表示智能体可执行动作库,定义可执行动作,如:

define move(FromPin, ToPin)

定义将物体从 FromPin 移动到 ToPin 的动作。

(5) 行为规则集:表示智能体选择动作的规则条件,即行为的触发规则,如:

If bel(disc(Disk)), goal(empty(ToPin)) then move(Disk, ToPin).

表示信念条件和目标条件共同影响下,决策执行 move 动作。

针对具体任务环境,ACB 模型具体模块构建方式如图 4 所示,在将问题背景分成任务、环境和能力的基础上,分别进行相应模块设计,其中:分析任务,将目标进行时序先后分解并进行逻辑表示,构建目标集;分析环境,对固定的现实公理进行描述,形成知识集,同时设置初始化环境认知,形成信念集;分析能力,确定可执行行为,构建动作集,同时由状态-动作关系确定动作触发条件,形成行为规则集。

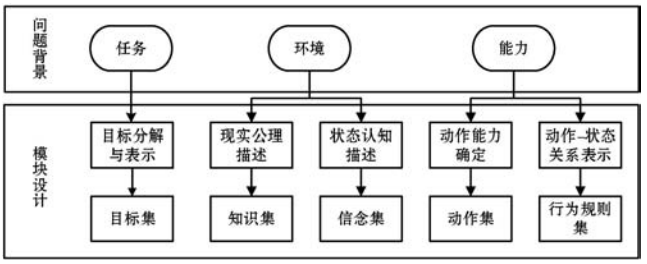


图 4　ACB 功能模块构建流程

ACB 模型中,信念集内容均通过感知来获取而不进行预置,保证智能体信念集的准确性。同时行动只通过改变环境中智能体的行为来间接改变信念集内容,不能够直接影响信念集,突出环境交互的作用。

智能体与环境交互过程如图 5 所示,智能体首先由环境感知状态信息,对信念集进行初始化或更新,而后知识集与信念集相结合得到状态认知,形成智能体对内部状态与环境状态的总体认识,并以此作为行动选择的基础。行动选择受状态与目标双重驱动影响,智能体以信念集为依据,以目标集为导向,以动作集为选项,最后以行为规则集为约束,进行行动选择并作用于环境。模型循环这一过程直至实现目标集内容,即完成智能体最终目标。

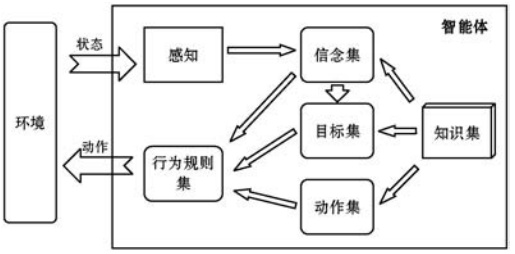


图 5　ACB 模型架构

2.2　引入启发策略网络的启发加速深度 Q 学习

为了有效利用 ACB 模型,加速强化学习,本文设计引入启发策略网络的深度 Q 学习,通过启发策略网络,引导 DQN 学习过程中的探索行为。

算法运行时初始化构建三个神经网络,即价值网络 θ_q、目标网络 $\theta_{\hat{q}}$和启发策略网络 θ_h。价值网络即算法训练的目标模型,完成由状态到动作的有效映射,价值网络在训练过程中实时更新,逐步提升模型性能;目标网络的设立旨在一定程度上分离探索策略与价值网络,形成离线策略学习方式,目标网络与价值网络结构相同,每隔固定周期获得价值网络参数并随之更新自身参数;启发策略网络完成由知识、状态到启发值的映射,启发策略网络与价值网络同时作用,决定智能体学习过程中的行为选择,启发策略网络的更新与价值网络同步进行。

训练过程中,智能体由初始状态开始进行探索,探索策略为:

$$\pi(s_t|\theta_{\hat{q}},\theta_h)=\begin{cases}\arg\max\limits_{a_t\in A}[Q(s_t,a_t|\theta_q)+\xi H_t(s_t,a_t|\theta_h)] & p>\varepsilon\\ a_{\text{random}} & \text{其他}\end{cases}\tag{1}$$

式中:$Q(s_t,a_t|\theta_q)$与 $H_t(s_t,a_t|\theta_h)$分别为价值网络与启发策略网络输出,两者线性相加决定动作选择策略;ξ 表示启发权重,用于描述启发策略网络的影响程度;p 为[0,1]中概率一致的随机值;$\varepsilon(0\leqslant\varepsilon\leqslant1)$为探索率,$\varepsilon$ 越大表示随机选择的概率越大。选择动作过程中,ACB 模型经过内部推演向智能体输出启发动作 a_{cog}。由于 a_{cog}并不一定完备,存在特定状态下无启发知识的情况,算法设置标签 p_{adding},用以记录是否存在启发知识。p_{adding}为 0 表示不存在启发知识,此时将 a_{cog}设置为 $\arg\max\limits_{a_t\in A}Q(s_t,a_t|\theta_q)$,$p_{\text{adding}}$为 1 表示启发知识存在。

启发策略网络仅作用于行动选择过程,该算法只影响学习过程中的探索方式,且能够保证状态空间的完全探索,不影响原学习算法收敛性。

智能体执行动作后,环境反馈相应的奖励 r 以及新的状态 s_{t+1},每个时间步 t 将状态转移序列$(s_t,a_t,r,s_{t+1},a_{\text{cog}},p_{\text{adding}})$储存在回放记忆单元 D 中。

算法使用最小化损失函数的梯度下降方法进行训练,分别设计价值网络和启发策略网络的损失函数,计算目标值与估计值的误差。价值网络目标值为:

$$y_q=\begin{cases}r & s_{t+1}\text{为终止状态}\\ r+\gamma\max\limits_{a_{t+1}\in A}\hat{Q}(s_{t+1},a_{t+1}|\theta_q) & \text{其他}\end{cases}\tag{2}$$

以目标值与网络输出值的均方误差作为损失函数,相应的 Q 网络损失函数为:

$$L_q(\theta_q)=\frac{1}{m}\sum_{j=1}^{m}(y_q-Q(s_t,a_t|\theta_q))^2\tag{3}$$

式中:m 表示一次误差计算采样的样本数。

启发策略网络描述了当前状态 s_t下对动作 a_t的推荐程度,本文将 $h(s_t,a_t)$作为该网络拟合的目标值。此外,为保证拟合的启发策略网络能够有效引导行动的选择,$h(s_t,a_{\text{cog}})$必须高于其 Q 估计值 $Q(s_t,a_{\text{cog}}|\theta_q)$与当前状态下最高的 Q 估计值 $\arg\max\limits_{a_t\in A}A(s_t,a_t|\theta_q)$之间的差值,$h(s_t,a_t)$形式化描述为:

$$h(s_t,a_t)=\begin{cases}\max\limits_{a_i\in A}Q(s_t,a_i|\theta_q)-Q(s_t,a_t|\theta_q)+\eta & a_t=a_{\text{cog}}\\ 0 & \text{其他}\end{cases}\tag{4}$$

式中:η 表示启发幅度,通常为较小的数值以尽量减小对 Q 值估计的影响。

基于此,启发策略网络损失函数定义为:

$$L_h(\theta_h) = \left\{\frac{1}{k}\sum_{j=1}^{k}\left[\frac{1}{n}\sum_{i=1}^{n}(h(s_j,a_i) - H(s_j,a_i \mid \theta_h))^2 p_{\text{adding}}\right]\right. \quad (5)$$

式中：n 为状态对应的动作空间大小；k 为 m 个采样样本中 p_{adding} 不为 0 的样本个数。

学习过程中，在记忆单元 D 存储一定数量的状态转移序列后，每个时间步使用最小化误差的梯度下降方法分别更新价值网络和启发策略网络参数，直至训练结束。

CB-HADQN 算法流程可以表述为算法 1。

算法 1 CB-HADQN

Initialize：

经验回放器 D，价值网络参数 θ_q，目标网络参数 $\theta_{\hat{q}} = \theta_q$，启发策略网络参数 θ_h，探索率 ε，最大步数 T，训练幕数 M，采样样本数 m。

Build ACB model：

编写知识文件

创建目标模块

设置信念集文件（初始为空）

创建初始化模块，确定智能体与环境的初始化交互机制

设置事件模块，确定智能体执行过程中的环境感知规则

创建动作模块，确定可执行动作内容

创建智能体模块，确定动作执行规则

编辑智能体主程序，加载智能体功能模块

Learning：

for $i \in \{1,2,\cdots,M\}$ do

$t=0$，初始化状态 s_t

Repeat

$p_{\text{adding}} = 0$

ACB 模型由状态 s_t 输出启发动作 a_{cog}

if a_{cog} = None

$a_{\text{cog}} = \text{argmax}Q_{a_t \in A}(s_t, a_t \mid \theta_q)$

else $p_{\text{adding}} = 1$

以式(1)选择动作

从环境获得奖励 r 并得到下一个状态 s_{t+1}

将经验 $(s_t, a, r, s_{t+1}, a_{\text{cog}}, p_{\text{adding}})$ 存入 D

计算当前目标 Q 值 y_q：

$$y_q = \begin{cases} r & s_{t+1} \text{ is end} \\ r + \gamma \max\limits_{a_{t+1} \in A} \hat{Q}(s_{t+1}, a_{t+1} \mid \theta_{\hat{q}}) & \text{其他} \end{cases}$$

$s_t \leftarrow s_{t+1}$

$t \leftarrow t+1$

最小化 $L_q(\theta_q) = \frac{1}{m}\sum_{j=1}^{m}(y_q - Q(s_t, a_t \mid \theta_q))^2$ 以更新 θ_q

最小化

$L_h(\theta_h) = \left\{\frac{1}{k}\sum_{j=1}^{k}\left[\frac{1}{n}\sum_{i=1}^{n}(h(s_j,a_i) - H(s_j,a_i \mid \theta_h))^2 p_{\text{adding}}\right]\right.$ 以更新 θ_h

每隔 N 步更新 $\theta_{\hat{q}} \leftarrow \theta_q$

Until s_t 为终止状态 or 到达最大步数 T

end for

3 实验与结果分析

本文设置两组实验，分别为实验一和实验二。实验一环境为 OpenAI GYM 强化学习环境 MountainCar，验证算法有效性。实验二环境为 Deepmind PySC2 星际争霸 2 强化学习环境 BuildMarines，验证算法在复杂学习环境下适应性。

3.1 实验一环境与具体实施

MountainCar 典型环境如图 6 所示。

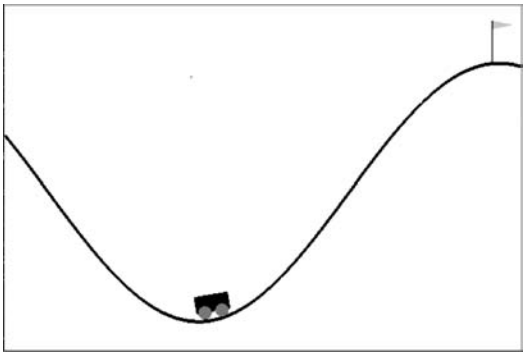


图 6 MountainCar 环境

MountainCar 环境下，小车智能体从模拟山谷谷底随机位置出发，目标是达到右侧山顶。在任何一个位置，都可选择执行向左加速、不加速和向右加速三种动作。当小车达到右侧山顶或步数超过 200 步时，一个幕结束，分别认为任务成功和任务失败。运动过程的观测值为小车位置、速度和当前步数。

实施过程中，首先构建小车智能体认知行为模型。以直观认知分析，在运动初始阶段，小车先向右加速，在动力不足后向左加速，以此积累小车势能与动能，达成目标概率更大。因此基于此项经验知识，构建如图 7 所示 MountainCar 小车 ACB 模型。

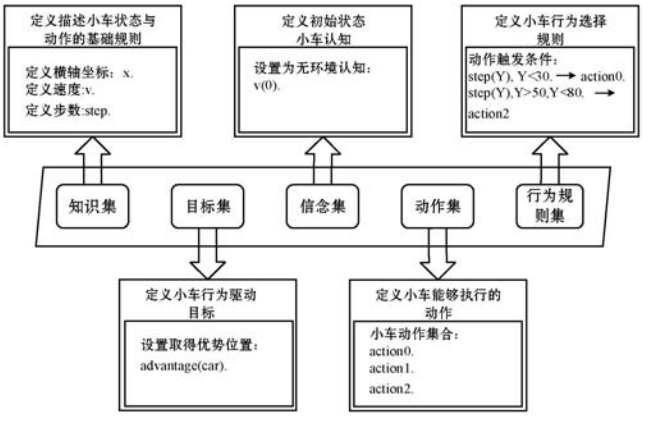


图 7 MountainCar 小车 ACB 模型

而后将构建好的 ACB 模型应用到 CB-HADQN 算法中,进行智能体训练。为验证算法可行性与高效性,进行 CB-HADQN 算法与 DQN 算法的对照实验。为验证算法在奖励稀疏条件下的适应性,另设奖励稀疏条件下的对照实验。两种条件下奖励设计为:

(1) 奖励为非稀疏奖励,奖励值与小车自身高度正相关,即越远离谷底奖励值越大,奖励函数设计为 $r = \mathrm{abs}(p_{\mathrm{osition}} + g_{\mathrm{oal\_position}})$,达到目标位置时奖励为 10,其中:$p_{\mathrm{osition}}$为当前位置;$g_{\mathrm{oal_position}}$为目标位置。

(2) 奖励为稀疏奖励,小车只有在达到目标位置时能够获得奖励 10,其他状态下奖励为 0。

在两种奖励设置下分别对 CB-HADQN 算法和 DQN 算法分别进行 30 轮实验,记录训练过程中的 20 幕奖励均值,统计两者 30 轮实验结果的数据均值及其置信区间(95% 置信度),并计算全过程成功次数及成功率。实验一学习网络为双层 32 单元全连接网络,参数设置如表 1 所示。

表 1　实验一参数设置

参数名	参数值
学习率	0.01
折扣系数	0.9
探索率	0.1
记忆单元容量	500
最大步数	200
启发权重	1
启发幅度	1

3.2　实验一结果分析

奖励非稀疏条件下得到实验结果,如图 8 和表 2 所示。

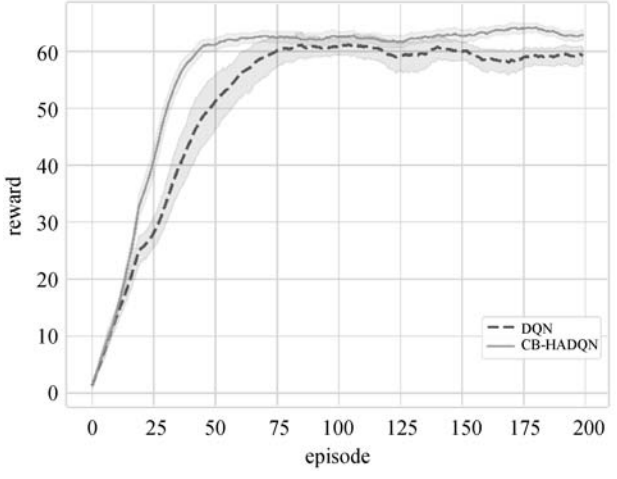


图 8　实验一连续奖励条件下 20 幕均值奖励统计

表 2　实验一连续奖励条件下训练成功次数及成功率统计

算法	平均成功次数	成功率/%
DQN	127.53	63.93
ACB-HADQN	138.47	69.23

对比实验结果,两种算法均能够训练出有效的学习模型,完成小车爬山任务,但 CB-HADQN 算法在多方面体现出其优势。首先,CB-HADQN 算法学习速度更快,在约 40 幕训练后,即能够达到平均 60 的奖励,而 DQN 则需要约 75 幕,才能达到相同的效果,CB-HADQN 算法训练速度有极大的提升。其次,CB-HADQN 算法整体训练过程中波动幅度较小,而 DQN 波动幅度大,这说明前者训练稳定性更好,受随机种子等系统因素影响小。最后,CB-HADQN 训练后期整体奖励值更高,说明模型的整体训练效果强于 DQN。

奖励稀疏条件下进行同样实验并统计实验结果,得到结果如图 9 所示。

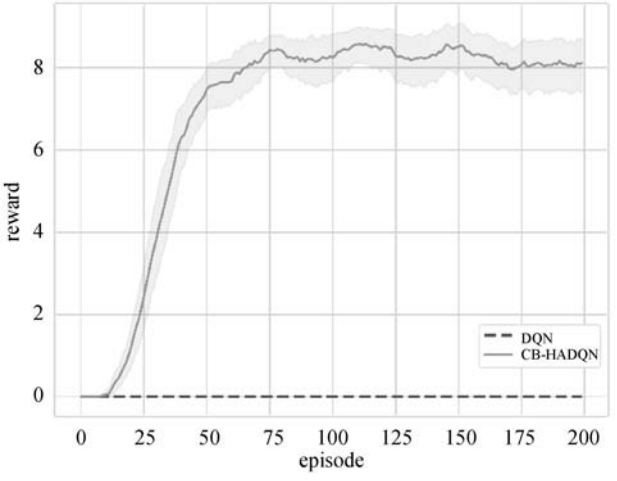


图 9　实验一稀疏奖励条件下 20 幕均值奖励统计

训练过程中,CB-HADQN 算法能够有效利用 ACB 模型提供的知识,成功地训练出高效的价值网络和启发策略网络,能够成功训练出有效的小车智能体。DQN 算法在 30 轮实验中,整体未呈现学习趋势,未能学出有效结果,由表 3 可得,DQN 算法在该情况下未能完成任务,成功率为 0。CB-HADQN 算法训练全过程成功率达 70.37%,这一成功率表明该算法能够有效胜任奖励稀疏环境下的学习问题,ACB 能够有效引导智能体学习过程,提高学习效率,完成 DQN 无法完成的特定任务。

表 3　实验一稀疏奖励条件下训练成功次数及成功率统计

算法	平均成功次数	成功率/%
DQN	0	0
ACB-HADQN	140.73	70.37

综合两种奖励设置下的实验结果,CB-HADQN 算法在学习效率、学习结果和稀疏奖励适应性等方面较 DQN 算法都有更好的表现,表明 CB-HADQN 算法能够有效利用知识模型引导学习过程,完成知识与学习的高效融合,提升智能体的学习能力,从而加速智能体学习。

3.3 实验二环境与具体实施

实验二 PySC2 星际争霸 2 BuildMarines 环境如图 10 所示。

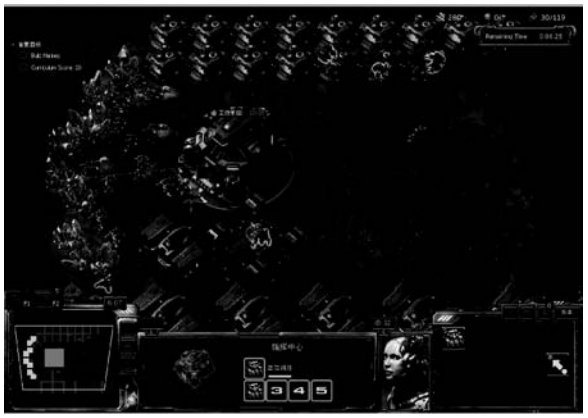


图 10 PySC2 星际争霸 2 BuildMarines 环境

BuildMarines 环境是初始拥有 12 个 SCV、1 个指挥中心和 8 个矿场的星际争霸 2 游戏。智能体通过控制 SCV 完成矿物收集,矿物用于建造补给站和营房,补给充足且具有营房时可用于建造陆战队员,并通过建造更多陆战队员获得更大奖励,每建造一个陆战队员获得奖励 1,以限定时间内(15 分钟)成功制造陆战队员总数为最终得分。BuildMarines 是 PYSC2 Minigames 中最具策略性和挑战性的实验环境[17],因此更能充分验证本文算法有效性。

实验实施过程中,首先进行 BuildMarines 智能体的 ACB 模型的构建。BuildMarines 环境下人类玩家经验为:实时判断是否需要补给站,补给充足时优先制造一定数量 SCV,提升采矿能力并进行采矿,条件允许时建造一定数量兵营,而后集中制造能力制造陆战队员,军队数量过多时,可手动杀死已生产的陆战队员,在不影响奖励的情况下减少需要的补给站数量。因此,ACB 模型的目标集由多个不同优先顺序的子目标构成,且补给充足为实时目标,即模型运用过程中,所有动作的执行均以此目标为驱动之一。

BuildMarines 智能体 ACB 模型其余各模块构建内容与 MountainCar 小车 ACB 模型类似,其中动作集确定了制造补给站、制造兵营、制造 SCV、制造陆战队员和击杀陆战队员五个动作,行为规则集依据人类玩家经验进行动作触发条件构建。

而后应用构建好的 ACB 模型进行智能体训练。与实验一相同,实验二同样进行 CB-HADQN 算法与 DQN 算法对照实验,并统计两者 30 轮奖励均值及奖励置信区间(95% 置信度)。实验二网络为双层 256 单元全连接网络,参数设置如表 4 所示。

表 4 实验二参数设置

参数名	参数值
学习率	0.01
折扣系数	0.99
探索率	0.1
记忆单元容量	500
启发权重	1
启发幅度	1

3.4 实验二结果分析

实验二的统计结果如图 11 所示。

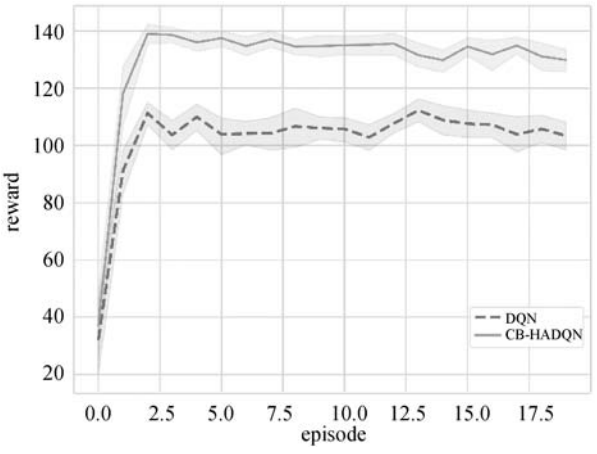
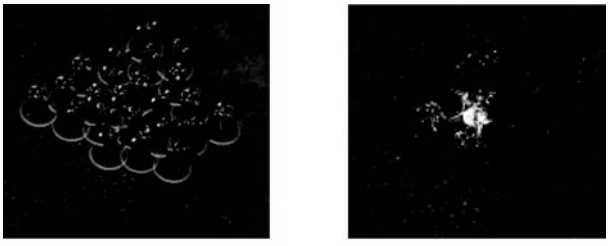


图 11 实验二学习奖励统计

对比实验结果,由于一轮游戏内训练步数较多(约 800 步),两种算法均能在两至三轮游戏后完成网络的初步训练并取得一定的训练效果。但 CB-HADQN 相较于 DQN 算法在以下几个方面均具有优势:首先,CB-HADQN 的训练速度更快,达到 DQN 的训练效果仅需其约 1/2 的训练步数。其次,CB-HADQN 算法整体训练结果较 DQN 更为稳定,学习曲线置信区间整体小于后者。最后,CB-HADQN 算法有着更好的训练结果,在 BuildMarines 环境下,DeepMind 人类玩家的奖励均值为 138,星际争霸大师水平得分标准为 133[17],CB-HADQN 算法训练后得分已超越星际争霸大师水平标准,与人类玩家水平接近,这表明训练得到的模型较为成功,而 DQN 算法训练结果只能体现一定程度的策略性,平均得分较低。

此外,利用两者训练模型进行测试可发现,DQN 智能体对已制造的陆战队员处理不够灵活,易堆积

并占用大量补给,如图 12(a)所示。而 CB-HADQN 智能体能够动态地调整陆战队员数量,适当降低补给占用以争取最大得分,如图 12(b)所示。这进一步表明 CB-HADQN 算法训练的模型具备更高的策略性。



(a)　　(b)

图 12　陆战队员处理方式对比

综合以上分析,CB-HADQN 算法在复杂环境下具备良好的表现,能够有效利用 ACB 模型的启发指导作用,引导智能体进行高效训练,提升强化学习智能体策略性。

4　结　语

本文基于 BDI 模型构建动态的 ACB 模型,形成引导智能体学习的有效启发知识来源。此外,将启发策略网络引入到深度强化学习过程中,与 DQN 算法相结合,形成 CB-HADQN 强化学习算法。该算法提升强化学习算法在连续状态空间及奖励稀疏环境等复杂学习环境下的学习速度与学习效果,加速智能体学习,并在 MountainCar 环境与 BuildMarines 环境中得到充分验证。

本文算法在一定程度上实现了知识与学习、认知与感知的融合,是利用人的经验知识或现有的学习成果引导学习的有效方法,为加速智能体学习、提升学习效率提供可行手段。

参　考　文　献

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 51(7540):529 – 533.

[2] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7216):484 – 489.

[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] Wu Y H, Mansimov E, Liao S, et al. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation[C]//31st International Conference on Neural Information Processing Systems, 2017:5285 – 5294.

[5] Silva B C, Basso E W, Bazzan A L, et al. Dealing with non-stationary environments using context detection[C]//23rd International Conference on Machine Learning, 2006: 217 – 224.

[6] Hernandez-Leal P, Kaisers M. Towards a fast detection of opponents in repeated stochastic games[C]//1st Workshop on Transfer in Reinforcement Learning, 2017:319 – 324.

[7] Zheng Y, Meng Z P, Hao J Y, et al. A deep Bayesian policy reuse approach against non-stationary agents[C]//32nd International Conference on Neural Information Processing Systems, 2018:962 – 972.

[8] Li S Y, Gu F D, Zhu G X, et al. Context-aware policy reuse [EB]. arXiv:1806.03793, 2018.

[9] 李晨溪,曹雷,张永亮,等. 基于知识的深度强化学习研究综述[J]. 系统工程与电子技术, 2017, 39(11):2603 – 2613.

[10] Fernández F, Veloso M. Probabilistic policy reuse in a reinforcement learning agent[C]//5th International Joint Conference on Autonomous Agents and Multiagent Systems, 2006: 720 – 727.

[11] Bianchi R A C, Ribeiro C H C, Costa A H R. Accelerating autonomous learning by using heuristic selection of actions [J]. Journal of Heuristics, 2008, 14(2):135 – 168.

[12] Li S, Zhang C. An optimal online method of selecting source policies for reinforcement learning [EB]. arXiv: 1709.08201, 2017.

[13] 姜强,药文静,赵蔚,等. 面向深度学习的动态知识图谱建构模型及评测[J]. 电化教育研究, 2020, 41(3):85 – 92.

[14] Bratman M E, Israel D J, Pollack M E. Plans and resource-bounded practical reasoning[J]. Computational Intelligence, 2010, 4(3):349 – 355.

[15] Bordini R H, Seghrouchni A, Hindriks K, et al. Agent programming in the cognitive era[J]. Autonomous Agents and Multi-Agent Systems, 2020, 34(2):9453 – 9461.

[16] Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym [EB]. arXiv:1606.01540, 2016.

[17] Vinyals O, Ewalds T, Bartunov S, et al. StarCraft II: A new challenge for reinforcement learning [EB]. arXiv: 1708.04782, 2017.

[18] Baxter L A. Markov decision processes: Discrete stochastic dynamic programming[J]. Technometrics, 1995, 37(3): 353.

[19] Shapiro E, Garrett R. The art of prolog[J]. IEEE Expert, 1994, 2(2):106 – 107.

[20] Hindriks K V, Dix J. GOAL: A multi-agent programming language applied to an exploration game[M]//Agent-Oriented Software Engineering. Springer, 2014.