DOI:10.7544/issn1000-1239.2019.20170812
56(2):262-280,2019

基于 TD-error 自适应校正的深度 Q 学习主动采样方法

白辰甲　刘　鹏　赵　巍　唐降龙
(哈尔滨工业大学计算机科学与技术学院模式识别与智能系统研究中心　哈尔滨　150001)
(bai_chenjia@stu.hit.edu.cn)

Active Sampling for Deep Q-Learning Based on TD-error Adaptive Correction

Bai Chenjia, Liu Peng, Zhao Wei, and Tang Xianglong
(*Pattern Recognition and Intelligent System Research Center, School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)

Abstract　Deep reinforcement learning (DRL) is one of research hotspots in artificial intelligence. Deep Q-learning is one of the representative achievements of DRL. In some fields, its performance has met or exceeded the level of human expert. It is necessary for training deep Q-learning to acquire lots of samples. These samples are obtained by the interaction between agent and environment. However, it is usually computationally intensive and sometimes impossible to keep away from interaction risk. We propose an active sampling method based on TD-error adaptive correction in order to solve sample efficiency problem in deep Q-learning. In various deep Q-learning methods, the updating of storage priority in experience memory lags behind the updating of Q-network parameters. It causes that a lot of samples are not selected to apply in Q-network training because the storage priority cannot reflect the true distribution of TD-error in experience memory. The TD-error adaptive correction active sampling method proposed in this paper uses the replay periods of samples and Q-network state to establish a priority bias model to estimate the real priority of each sample in experience memory during the Q-network iteration. The samples are selected from experience memory according to the corrected priority and the bias model parameters are adaptively updated by a segmented form. We analyze the complexity of the algorithm and the relationship between learning performance and the order of polynomial feature and updating period of model parameters. Our method is verified on the platform of Atari 2600. The experimental results show that proposed method improves the learning speed and reduces the number of interaction between agent and environment. Meanwhile, it ameliorates the quality of optimal policy.

Key words　sample priority; TD-error correction; adaption; active sampling; deep Q-learning; reinforcement learning

摘　要　强化学习中智能体与环境交互的成本较高.针对深度 Q 学习中经验池样本利用效率的问题,提出基于 TD-error 自适应校正的主动采样方法.深度 Q 学习训练中样本存储优先级的更新滞后于 Q 网络参数的更新,存储优先级不能准确反映经验池中样本 TD-error 的真实分布.提出的 TD-error 自适应校正主动采样方法利用样本回放周期和 Q 网络状态建立优先级偏差模型,估计经验池中样本的真实优先级.在 Q 网络迭代中使用校正后的优先级选择样本,偏差模型在学习过程中分段更新.分析了 Q 网络

收稿日期:2017-10-27;**修回日期**:2018-04-25
基金项目:国家自然科学基金项目(61671175,61672190)
This work was supported by the National Natural Science Foundation of China (61671175, 61672190).
通信作者:赵巍(zhaowei@hit.edu.cn)

学习性能与偏差模型阶数和模型更新周期之间的依赖关系,并对算法复杂度进行了分析.方法在 Atari 2600 平台进行了实验,结果表明,使用 TD-error 自适应校正的主动采样方法选择样本提高了智能体的学习速度,减少了智能体与环境的交互次数,同时改善了智能体的学习效果,提升了最优策略的质量.

关键词 样本优先级;TD-error 校正;自适应;主动采样;深度 Q 学习;强化学习

中图法分类号 TP181

强化学习[1](reinforcement learning, RL)是机器学习的重要分支.智能体在与环境的交互过程中根据当前状态选择动作,执行动作后转移到下一个状态并获得奖励.从智能体执行动作到获得奖励的过程产生了一条"经验",智能体从大量经验中学习,不断改进策略,最大化累计奖励.传统强化学习方法适用于离散状态空间,不能有效处理高维连续状态空间中的问题.线性值函数近似或非线性值函数近似法将状态与值函数之间的关系进行参数化表示[2],可以处理高维状态空间中的问题.其中,线性值函数近似有良好的收敛性保证,但往往需要人为提取状态特征,且模型表达能力有限;非线性值函数近似没有严格的收敛性证明,但具有强大的模型表达能力.Tesauro[3-4]将时序差分法(temporal difference, TD)与非线性值函数近似相结合,提出了 TD-Gammon 算法.该算法用前馈神经网络对状态值函数进行估计,在西洋双陆棋中达到了人类顶尖水平.

近年来,深度学习[5-6]与强化学习相结合[7]的非线性值函数近似方法在大规模强化学习中表现出良好性能.谷歌 DeepMind 团队提出了"深度 Q 网络"[8-9](deep Q-network, DQN),DQN 以图像序列作为输入,输出对动作值函数的估计,策略表示为卷积神经网络的参数.DQN 在 Atari 2600 的大多数视频游戏上的性能超过了人类专家水平.

对经验样本的存储和采样是 DQN 的关键问题.DQN 在训练中使用误差梯度反向传播算法,只有训练样本集合满足或近似满足独立同分布时才能正常收敛[10].然而,智能体与环境交互产生的样本前后具有强相关性.为了消除样本相关性,DQN 使用经验池来存储和管理样本,在训练中从经验池中随机取样.DQN 需要容量巨大的经验池(约 10^6),为了填充经验池,智能体需要频繁地与环境进行交互.然而,智能体与环境交互的代价非常昂贵的,不仅表现在时间的消耗上,更表现在安全性、可控性、可恢复性等诸多方面[11].因此,研究如何对样本进行高效采样、减少智能体与环境的交互次数、改善智能体的学习效果,对深度强化学习研究有重要意义.

目前针对 DQN 样本采样问题的研究主要有 2 个方面:1)并行采样和训练.使用多个智能体并行采样和训练,用异步梯度法更新 Q 网络[12-13],其思想是通过并行采样和训练来打破样本之间的相关性.2)主动采样.强调样本之间的差异性[14],为每个样本设定优先级,按优先级进行采样.Schaul 等人[11]提出了优先经验回放法(prioritized experience replay, PER),经验池中样本被采样的概率与样本的存储优先级成正比,存储优先级由经验池中样本上次参与训练的 TD-error 决定.TD-error 是样本在使用 TD 算法更新时目标值函数与当前状态值函数的差值,其中目标值函数是立即奖励与下一个状态值函数之和.尽管 PER 算法在一定程度上提高了样本的利用效率,但在包含大量样本的经验池中,每个时间步仅有少量样本可以参与 Q 网络迭代,其余样本无法参与训练,这些样本的 TD-error 没有跟随 Q 网络的更新而变化,导致样本的存储优先级无法准确反映经验池中 TD-error 的真实分布.

为了进一步提高样本的采样效率,应当以更准确的采样优先级在经验池中选择样本.本文研究发现,经验池中样本距离上次被采样的时间间隔越长,样本的存储 TD-error 偏离真实值越远.由于长时间未被采样的样本数量巨大,导致存储优先级与真实优先级的偏差较大.本文提出一种基于 TD-error 自适应校正的主动采样模型(active sampling method based on adaptive TD-error correction, ATDC-PER).该模型基于样本回放周期和网络状态在样本与优先级偏差之间建立映射关系,实现对存储优先级的校正,在 Q 网络每次训练中都使用校正后的优先级在经验池中选择样本.本文提出的偏差模型参数是自适应的,在训练中模型始终跟随 Q 网络的变化,对采样优先级的估计更为合理.实验结果表明,校正后的优先级符合经验池中样本 TD-error 的分布,利用校正后的优先级选择样本提高了样本的利用效率,减少了智能体与环境的交互,同时改善了智能体的学习效果,获得了更大的累计奖励.

1 相关工作

1.1 强化学习

强化学习的目标是最大化累计奖励.智能体通过从经验中学习,不断优化状态与动作之间的映射关系,最终找到最优策略(policy).智能体与环境的交互过程可以用马尔可夫决策过程[15](Markov decision processes, MDP)来建模.在周期型任务中,MDP包括一系列离散的时间步0,1,2,…,t,…,T,其中T为终止时间步.在时间步t,智能体观察环境得到状态的表示S_t,根据现有策略π选择动作A_t并执行.执行动作后MDP到达下一个时间步$t+1$,智能体收到奖励R_{t+1}并转移到下一个状态S_{t+1}.回报定义为折扣奖励之和,智能体通过调整策略来最大化回报.动作状态值函数是指智能体处于状态s时执行动作a,随后按照策略π与环境交互直到周期结束所获得的期望回报,记为$Q_\pi(s,a)$.

最优策略求解一般遵循"广泛策略迭代"的思想,包含策略评价和策略提升[1].策略评价是已知策略计算值函数的过程,策略提升是已知值函数选择最优动作的过程.策略评价中值函数$Q_\pi(s,a)$的求解可以使用贝尔曼方程[15]转换为递归形式

$$\sum_{s',r} p(s',r \mid s,a)[r+\gamma \max_{a'} Q_\pi(s',a')],$$

其中,r是立即奖励,γ是折扣因子.值函数定义了在策略空间上的偏序关系,存在最优策略π^*优于(或等同于)其他所有策略.

贝尔曼方程中$p(s',r|s,a)$代表状态转移概率,由智能体所处的环境决定.如果已知状态转移概率,可以使用动态规划[16]、树搜索[17-18]等求解最优策略,此类方法为基于模型(model-base)的强化学习.然而,在多数问题中无法获取环境的状态转移概率,智能体需要通过与环境交互进行学习,此类方法为模型无关(model-free)的强化学习,主要包括策略梯度法和Q学习2种类型.策略梯度法[19-22]对策略π进行建模,奖励均值的期望作为策略π的评价函数,策略π跟随策略梯度的方向更新来最大化评价函数.Q学习法主要包括蒙特卡洛法[23](Monte Carlo, MC)、时序差分法[24](temporal difference, TD)等.其中,MC方法需在一个周期结束后更新值函数,而TD方法在交互过程的每个时间步均可更新值函数,因而适用范围更广.TD方法包括在策略(on-policy)和离策略(off-policy)2种类型.Q-learning[24]是一种典型的离策略算法,使用智能体与环境交互产生的样本s_t,a_t,r_{t+1},s_{t+1}进行值函数更新.离散状态空间下Q-learning算法中TD-error是目标动作值函数与当前动作值函数之差,其中目标动作值函数是智能体获得的立即奖励与下一个状态的最大动作值函数之和,目标动作值函数表示智能体在未来的期望回报.TD-error用$r_{t+1}+\gamma \max_a Q(s_{t+1},a)-Q(s_t,a_t)$表示,其中$\gamma$是折扣因子.

在大多数实际问题中无法用表格化的方式穷举所有的(s,a)组合,因此使用参数化方式对值函数进行表达,记为$Q(s,a;\boldsymbol{\theta}_t)$,其中$\boldsymbol{\theta}_t$为参数向量.值函数近似条件下Q-learning算法中TD-error为$r_{t+1}+\gamma \max_a Q(s_{t+1},a;\boldsymbol{\theta}_t)-Q(s_t,a_t;\boldsymbol{\theta}_t)$,参数向量按照$\boldsymbol{\theta}_t=\boldsymbol{\theta}_t+\alpha\delta\ \nabla_{\boldsymbol{\theta}_t}Q(s_t,a_t,\boldsymbol{\theta}_t)$进行更新.

1.2 深度Q网络

深度Q网络[9](DQN)遵循Q-learning算法的更新法则,与线性值函数近似的不同之处在于DQN使用Q网络来表示动作值函数,进而表示策略.Q网络是由卷积层和全连接层组成的卷积神经网络(CNN).DQN的主要特点有2个:

1)使用2个独立的Q网络.分别使用$\boldsymbol{\theta}$和$\boldsymbol{\theta}^-$代表Q网络和目标Q网络的参数,每隔L个时间步将Q网络的参数复制到目标Q网络中,即$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$,随后$\boldsymbol{\theta}^-$在$L$个时间步内保持不变.DQN中TD-error定义为

$$\delta_t^{\mathrm{DQN}}=r_{t+1}+\gamma \max_a Q(s_{t+1},a;\boldsymbol{\theta}_t^-)-Q(s_t,a_t;\boldsymbol{\theta}_t). \tag{1}$$

使用2个独立的Q网络可以使学习的目标值函数分段保持稳定,使训练过程更加鲁棒.Q网络的损失函数为平方误差损失,$L(\boldsymbol{\theta}_t)=\mathrm{E}_{s,a}[(\delta_t^{\mathrm{DQN}})^2]$,用误差反向传播算法迭代更新Q网络的参数,直到Q网络收敛.

2)使用经验池存储和管理样本,使用经验回放[25]选择样本.样本存储于经验池中,从经验池中随机抽取批量样本训练Q网络.使用经验回放可以消除样本之间的相关性,使参与训练的样本满足或近似满足独立同分布.

在DQN的基础上,Hasselt等人[26-27]指出,DQN在计算TD-error时使用相同的Q网络来选择动作和计算值函数会导致值函数的过高估计,进而提出了DDQN(double DQN)算法.该算法用Q网络来选择动作,用目标Q网络来估计值函数,从而消除了对值函数的过高估计,并提升了智能体的

累计奖励. DDQN 中 TD-error 由式(2)计算:

$$\delta_t^{\mathrm{DDQN}}=r_{t+1}+\gamma Q(s_{t+1},\arg\max_a Q(s_{t+1},a;\boldsymbol{\theta}_t);\boldsymbol{\theta}_t^-)-Q(s_t,a_t;\boldsymbol{\theta}_t). \tag{2}$$

此外,一些研究从不同角度提出了 DQN 的改进方法,主要思路可以分为:1)从探索和利用的角度[28-31],将传统强化学习中使用的探索与利用策略扩展到深度强化学习中,改进 DQN 中的 ε-贪心策略. 2)从网络结构的角度,使用竞争式网络结构[32],分别估计动作值函数和优势函数. 3)从并行训练的角度[12-13],消除样本之间的相关性,同时加速训练. 4)从减少训练中奖励方差的角度[33],使用多个迭代步的延迟估计 Q 值,使其更加稳定. 5)从模型之间知识迁移和多任务学习的角度[34-35],扩展 DQN 的使用范围. 6)从视觉注意力的角度,使智能体在训练中将注意力集中于有价值的图像区域[36]. 7)从主动采样,减少与环境交互次数的角度[11],使用基于 TD-error 的优先经验回放来提高样本的利用效率.

1.3 优先经验回放

优先经验回放是一种有效的管理经验池的方法. 原始的 DQN 算法在训练时不考虑经验池中样本之间的差异,随机抽取样本进行训练,导致样本的利用效率较低,智能体需要频繁地与环境进行交互. 针对这一问题,Schaul 等人[11]提出了优先经验回放法(PER)来估计每个样本的优先级,按照优先级选择样本训练 Q 网络. 样本优先级由样本的 TD-error 分布决定. 样本的 TD-error 绝对值越大,在训练中造成的 Q 网络损失越大,表明该样本给 Q 网络带来的信息量越大,在 Q 网络后续迭代中应"优先"选择该样本参与训练. PER 算法的主要特点有 3 个:

1) 样本优先级基于样本上次参与训练时的 TD-error 值. 长时间未参与训练的样本在经验池中存储的 TD-error 长时间未更新,这些样本的 TD-error 无法跟随 Q 网络的变化. 只有上一个时间步参与训练的批量样本的 TD-error 值被更新,然而这部分样本仅占经验池中样本的少数.

2) 采样时在优先级的基础上引入随机性. 完全按照优先级进行采样会降低样本的多样性,因此 PER 算法中引入随机性,使样本被抽取的概率与优先级成正比,同时所有样本都有机会被采样.

3) 优先级的引入相对于均匀抽样改变了样本的分布,引入了误差. PER 算法使用重要性采样权重对误差进行补偿. 在计算损失函数梯度时,需在原有梯度的基础上乘以重要性采样权重,按补偿后的梯度进行更新.

PER 算法[11]具有高度的灵活性,可与离策略的深度强化学习方法进行广泛结合. 例如,文献[37-38]将 PER 算法和策略梯度法结合,提高了确定性策略梯度算法[39-40](deep deterministic policy gradient, DDPG)在仿真机器人控制中的数据利用效率;PER 算法与深度 Q 学习的结合提高了经验池中样本的利用效率,显著减少了智能体与环境的交互次数,提升了智能体的最优策略得分[11].

2 基于 TD-error 校正的主动采样模型

PER 算法的问题在于,经验池中存储的 TD-error 值相对于 Q 网络来说更新不及时,无法准确反映样本的优先级. 在每次迭代中 Q 网络都会更新自身参数,而经验池仅可更新参与训练的批量样本的 TD-error 值,此类样本的数量约占经验池容量的

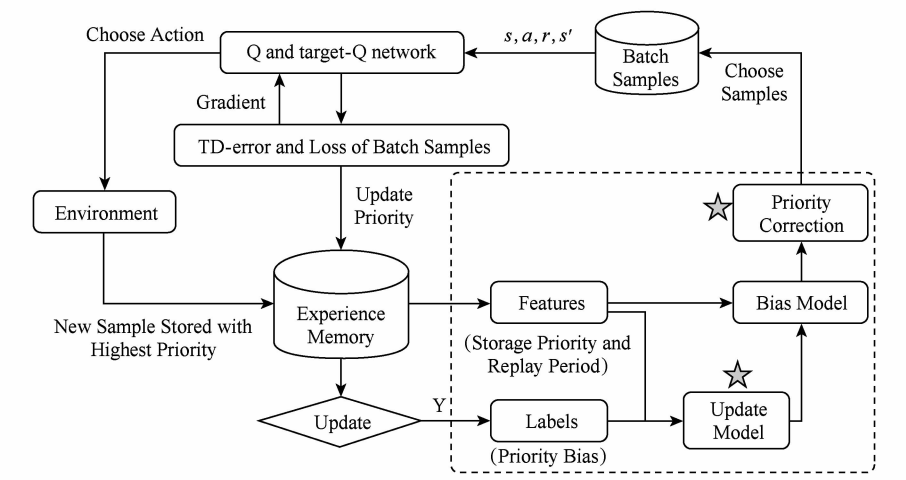


Fig. 1 The main architecture of ATDC-PER model

图 1 本文方法(ATDC-PER)的总体结构

0.1%. 这导致大多数样本的存储 TD-error 无法跟随 Q 网络的变化,并进一步导致样本在经验池中的存储优先级与"真实"优先级之间存在偏差. 本文分析了这种偏差对学习的影响,并阐述了依据真实或逼近真实的 TD-error 分布产生的优先级来选择样本能够使深度 Q 网络以更高的效率收敛到最优策略.

本文提出基于 TD-error 自适应校正的主动采样模型(ATDC-PER)来校正样本的存储优先级,使用校正后的优先级分布进行主动采样,能够提高经验池中样本的利用效率. 图 1 是本文方法 ATDC-PER 的总体结构,主要包括"偏差模型"和"优先级校正"两部分. 其中,优先级校正需在所有时间步进行,偏差模型则每隔 D 个时间步更新 1 次. 2.1 节使用引例说明 TD-error 及样本优先级分布对学习的作用,2.2 节阐述 ATDC-PER 的建模过程,2.3 节给出算法描述和复杂度分析.

2.1 TD-error 偏差分析

本节用强化学习中的经典问题平衡车[41](CartPole)为引例①,说明 PER 算法中经验池的存储 TD-error 与真实 TD-error 之间存在偏差,并分析 TD-error 分布对学习的作用.

CartPole 的状态表示为 4 维向量,包括杆的角度、杆运动的角速度、车的位置和车运动的速度,如图 2 所示:

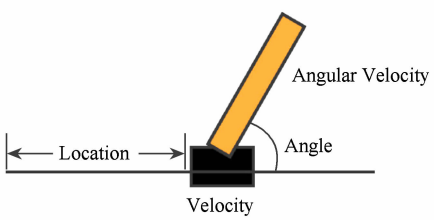


Fig. 2 The state of CartPole

图 2 倒立摆的状态表示

智能体通过向小车施加+1 或−1 的力来尽力保持杆的平衡. 在时间步 t,如果杆与垂直方向的角度不超过 15 度且小车与中心的距离不超过 2.4 个单元,则奖励为+1,否则奖励为−1 同时周期结束. 设 1 个周期的时间步不超过 200 步,则智能体在 1 个周期内所获得的最大奖励不超过 200. 用基于 DDQN 的 PER 算法来训练 Q 网络,Q 网络是包含 64 个隐含层节点的 3 层全连接网络,经验池容量为 50 000.

① https://gym.openai.com/envs/Cartpole-v0

PER 算法中经验池用样本集合 $E=\{e^{(1)},e^{(2)},\cdots,e^{(m)}\}$ 表示,其中序号为 i 的样本 $e^{(i)}=\langle s^{(i)},a^{(i)},r^{(i)},s'^{(i)}\rangle\in E$. PER 算法在每次迭代时抽取 E 中的 1 个批量样本进行训练. 本文将样本 $e^{(i)}$ 上次参与训练的 TD-error 值称为样本的存储 TD-error,记为 $\delta^{(i)}$. 设当前时间步为 t,$e^{(i)}$ 上次参与训练的时间步为 $t-\tau^{(i)}$,则 $\delta^{(i)}$ 由式(3)计算[27]:

$$\delta^{(i)}=r^{(i)}+\gamma Q(s^{(i)},\arg\max_a Q(s'^{(i)},a;\boldsymbol{\theta}_{t-\tau^{(i)}});\boldsymbol{\theta}^-_{t-\tau^{(i)}})-Q(s^{(i)},a^{(i)};\boldsymbol{\theta}_{t-\tau^{(i)}}),\tag{3}$$

其中 $\boldsymbol{\theta}_{t-\tau^{(i)}}$ 和 $\boldsymbol{\theta}^-_{t-\tau^{(i)}}$ 分别是 $t-\tau^{(i)}$ 时间步的 Q 网络和目标 Q 网络参数.

由式(3)可知,存储 TD-error 计算中使用的参数 $\boldsymbol{\theta}_{t-\tau^{(i)}}$ 不是当前 Q 网络参数,而是 $\tau^{(i)}$ 个时间步之前的参数. 在 $\tau^{(i)}$ 个时间步内 Q 网络经历了 $\tau^{(i)}$ 次参数迭代,而存储 TD-error 值却保持不变. 这导致样本的存储 TD-error 与当前 Q 网络不匹配,且 $\tau^{(i)}$ 的值越大,这种不匹配程度越高. 新样本加入经验池时 $\tau^{(i)}=1$,每经历 1 个时间步,参与该时间步训练的样本 $\tau^{(i)}$ 置 1,未参与该时间步训练的样本 $\tau^{(i)}$ 加 1. 由于经验池满后新样本会覆盖老样本,因此所有样本 $\tau^{(i)}$ 的取值范围均在 1 到经验池样本总量之间. 图 3 显示了 CartPole 在训练周期为 420 时经验池样本的 $\tau^{(i)}$ 的分布,样本按 $\tau^{(i)}$ 由小到大排序,经验池中样本总数为 50 000. 图 3 表明,经验池中有大量样本的 $\tau^{(i)}$ 值较大,这些样本长期没有参与训练,$\delta^{(i)}$ 长期未更新. 因此,存储优先级与当前 Q 网络的不匹配程度较高.

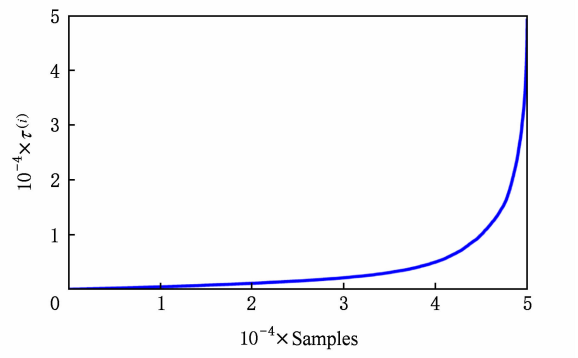


Fig. 3 Distribution of $\tau^{(i)}$ in experience memory when training episode is 420

图 3 训练周期为 420 时经验池中样本的 $\tau^{(i)}$ 的分布

引入一种理想的 TD-error 计算方式,称为样本"真实"的 TD-error,记为 $\delta^{(i)}_{\text{real}}$. 计算 $\delta^{(i)}_{\text{real}}$ 的分布需在

当前时间步将经验池的全部样本送入 Q 网络和目标 Q 网络,更新包括上次参与训练的样本在内的经验池中全部样本的 TD-error 值. 在这种计算方式下,经验池中任意样本的 TD-error 均由最新的 Q 网络和目标 Q 网络计算,均与当前最新的网络参数 $\boldsymbol{\theta}_t$ 和 $\boldsymbol{\theta}_t^-$ 相对应,如式(4)所示:

$$\delta_{\text{real}}^{(i)}=r^{(i)}+\gamma Q(s^{(i)},\arg\max_a Q(s'^{(i)},a;\boldsymbol{\theta}_t);\boldsymbol{\theta}_t^-)-Q(s^{(i)},a^{(i)};\boldsymbol{\theta}_t). \tag{4}$$

用 $p^{(i)}$ 和 $p_{\text{real}}^{(i)}$ 分别表示样本在经验池中的存储优先级和真实优先级,分别与存储 TD-error 值 $\delta^{(i)}$ 和真实 TD-error 值 $\delta_{\text{real}}^{(i)}$ 相对应,用式(5)(6)计算.

$$p^{(i)}=(|\delta^{(i)}|+\varepsilon)^{\alpha}, \tag{5}$$

$$p_{\text{real}}^{(i)}=(|\delta_{\text{real}}^{(i)}|+\varepsilon)^{\alpha}, \tag{6}$$

其中 ε 和 α 均为常数. 样本 TD-error 的绝对值越大,在采样中样本的优先级就越高.

真实优先级 $p_{\text{real}}^{(i)}$ 与当前最新的 Q 网络相对应,而存储优先级 $p^{(i)}$ 相对于当前 Q 网络滞后了 $\tau^{(i)}$ 个时间步,二者的分布存在差异. 图 4 比较了 CartPole 训练周期为 180 时 $p^{(i)}$ 和 $p_{\text{real}}^{(i)}$ 的分布,样本按 $p^{(i)}$ 进行排序. 图 4 中坐标系右侧的 $p^{(i)}$ 和 $p_{\text{real}}^{(i)}$ 的分布偏差较大,表明在 PER 算法中用于选择样本的存储优先级并没有反映经验池中样本的真实情况.

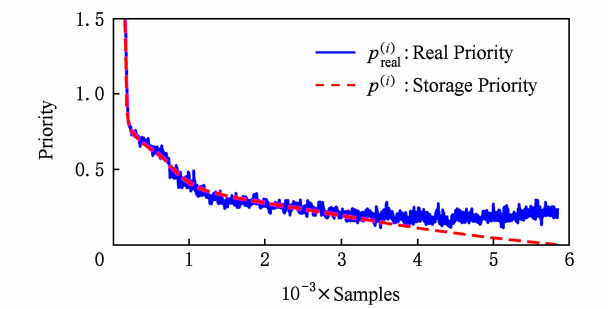


Fig. 4 Distribution of $p^{(i)}$ and $p_{\text{real}}^{(i)}$ in CartPole training when training episode is 180

图 4 CartPole 训练周期为 180 时 $p^{(i)}$ 和 $p_{\text{real}}^{(i)}$ 分布图

设 $\Delta p^{(i)}$ 为真实优先级 $p_{\text{real}}^{(i)}$ 与存储优先级 $p^{(i)}$ 之间的优先级偏差:

$$\Delta p^{(i)}=p_{\text{real}}^{(i)}-p^{(i)}. \tag{7}$$

图 5 显示了优先级偏差的绝对值 $|\Delta p^{(i)}|$ 的分布,样本仍按照 $p^{(i)}$ 由大到小排序.

图 4 和图 5 表明,PER 算法中使用基于 $\delta^{(i)}$ 的存储优先级进行主动采样存在 2 个问题:

1) 约 1/3 样本的优先级被显著降低. 如图 4 所示,$p^{(i)}$ 和 $p_{\text{real}}^{(i)}$ 的差异主要集中在坐标系右侧约占全体样本 1/3 的区域. 经统计,按照 $p^{(i)}$ 采样时,该区域

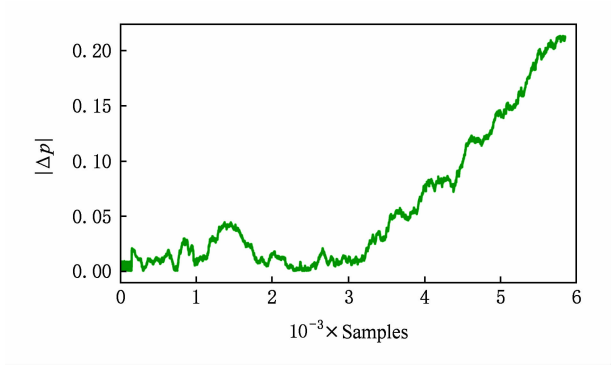


Fig. 5 Distribution of $|\Delta p^{(i)}|$ in CartPole training when training episode is 180

图 5 CartPole 训练周期为 180 时 $|\Delta p^{(i)}|$ 的分布图

样本优先级之和占全体样本的比例为 6.9%,而按照 $p_{\text{real}}^{(i)}$ 采样时这一比例可以达到 19.5%,提高近 3 倍. 因此,按照存储优先级进行采样显著降低了约 1/3 的样本被采样的概率.

2) $\delta^{(i)}$ 的分布不能准确反映经验池的真实情况. 其中,当 $|\delta^{(i)}|$ 较大时,样本被抽取的概率高,$\tau^{(i)}$ 值较小,期间 Q 网络变化不明显,优先级偏差较小,此时可以使用存储优先级作为真实优先级的近似估计;但当 $|\delta^{(i)}|$ 较小时,$\tau^{(i)}$ 值较大,Q 网络在 $\tau^{(i)}$ 个时间步内参数变化较大,导致存储优先级偏离真实优先级的分布,此时基于 $\delta^{(i)}$ 的存储优先级不能准确反映经验池的情况.

对 CartPole 问题分别使用基于存储优先级 $p^{(i)}$ 和真实优先级 $p_{\text{real}}^{(i)}$ 的 PER 算法进行训练. 图 6 对比了 2 种优先级下的训练曲线. 其中,横轴代表迭代时间步,每个时间步智能体与环境交互一次,纵轴代表智能体在一个周期内获得的累计奖励.

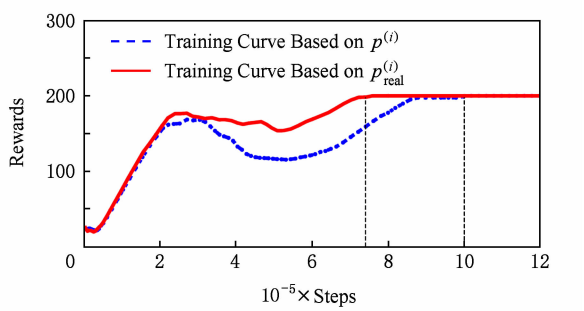


Fig. 6 Training curve comparison using experience replay with $p^{(i)}$ and $p_{\text{real}}^{(i)}$

图 6 使用 $p^{(i)}$ 和 $p_{\text{real}}^{(i)}$ 作为优先级的训练曲线对比

图 6 表明,使用真实优先级 $p_{\text{real}}^{(i)}$ 进行主动采样的效果更好. 具体体现为:

1) 收敛性. 使用存储优先级 $p^{(i)}$ 和使用真实优先级 $p_{\text{real}}^{(i)}$ 进行采样都可以收敛到最大奖励值.

2) 智能体与环境的交互次数. 使用真实优先级 $p_{\text{real}}^{(i)}$ 采样达到最大奖励需约 74 000 个时间步;而使用存储优先级 $p^{(i)}$ 则需约 100 000 个时间步,且学习过程中波动明显.

以上数据和分析表明,使用真实优先级 $p_{\text{real}}^{(i)}$ 进行主动采样可以提高样本的利用效率,显著减少智能体与环境的交互次数.

虽然 $p_{\text{real}}^{(i)}$ 是一种更为理想的优先级分布,但仅在小规模问题(如 CartPole)中可以使用. 在大规模强化学习中,计算 $p_{\text{real}}^{(i)}$ 的分布需在每个时间步将经验池的全部样本送入 Q 网络和目标 Q 网络,由于迭代时间步较多且经验池容量较大,计算会造成大量的时间成本和存储成本. 本文提出一种 TD-error 自适应校正方法来估计真实优先级和存储优先级偏差的分布,进而得到真实优先级的估计 $\hat{p}_{\text{real}}^{(i)}$. 本文并不直接计算真实优先级 $p_{\text{real}}^{(i)}$,能够以极小的代价获得跟随 Q 网络更新的采样优先级.

2.2 ATDC-PER 模型

本文提出的基于 TD-error 自适应校正的主动采样算法(ATDC-PER)包括"偏差模型"和"优先级校正"2 部分,使用分段更新策略将二者结合. 偏差模型是经验池中样本与真实优先级和存储优先级的偏差之间的回归模型. 在时刻 t 提取当前经验池的样本特征和优先级偏差求解偏差模型的参数,作为时刻 t 的偏差模型,在时刻 $t\sim(t+D)$ 内均使用时刻 t 的偏差模型对当前经验池样本的存储优先级进行校正. 校正后得到样本真实优先级的估计 $\hat{p}_{\text{real}}^{(i)}$,在 Q 网络训练时使用 $\hat{p}_{\text{real}}^{(i)}$ 选择样本. 在时刻 $t+D$ 再一次使用该时刻的经验池求解偏差模型的参数,偏差模型被更新,在 $(t+D)\sim(t+2D)$ 时间步内使用时刻 $t+D$ 求解的偏差模型进行优先级校正. 以此类推,偏差模型将在时刻 $t+3D$, $t+4D$, $t+5D$, … 更新,在所有时间步均根据最新的偏差模型进行优先级校正,D 为偏差模型的更新周期. ATDC-PER 算法在存储优先级的基础上估计真实优先级,既使得优先级可以跟随 Q 网络更新,又避免了直接计算真实优先级所需的巨大开销.

真实优先级和存储优先级的偏差分布具有稳定性,所以 ATDC-PER 方法对偏差模型更新周期 D 的选取是不敏感的. 真实优先级与存储优先级的不同之处在于计算时使用的 Q 网络参数不同,真实优先级使用时刻 t 的 Q 网络参数,而存储优先级使用 $t-\tau^{(i)}$ 时刻的 Q 网络参数,二者的区别在于 $\tau^{(i)}$ 的分布. 如图 3 所示,$\tau^{(i)}$ 在经验池中呈现长尾分布,最小值为 1,最大值为经验池样本总量. 在训练中经验池填充满后容量不再发生变化,$\tau^{(i)}$ 分布具有稳定性,因此偏差模型不需要频繁更新.

ATDC-PER 算法在训练过程中偏差模型分段更新的示意图如图 7 所示,横轴代表训练时间步,纵轴代表智能体在周期内的累计奖励. 偏差模型更新后的 D 个时间步内均使用该模型进行优先级校正,可以极大地减少时间开销.

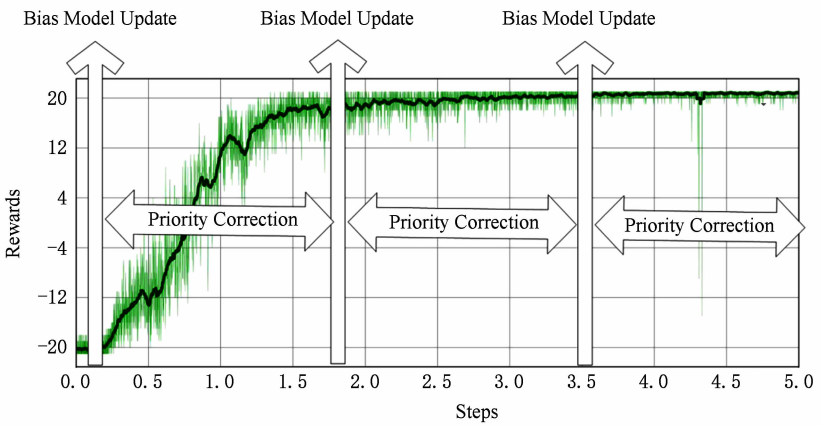


Fig. 7 Interval update policy in ATDC-PER algorithm

图 7 ATDC-PER 算法的分段更新策略示意图

2.2.1 偏差模型

从经验池中提取的样本特征是偏差模型的输入,真实优先级和存储优先级之间的优先级偏差作为样本标签,是偏差模型的输出. 使用线性回归模型对样本特征与样本标签之间的关系进行建模,通过最小化平方损失函数来求解模型参数 $\boldsymbol{w}$.

1) 样本特征. 样本特征是针对单个样本而言的. 特征在训练中的经验池中保存且与优先级偏差相关. 样本特征包括:

① 样本在经验池中的存储优先级. 由图 4 可知,

总体上,样本的存储优先级 $p^{(i)}$ 越低,优先级偏差越大. 为了使模型在不同环境和不同训练阶段具有通用性,将存储优先级的定义在式(5)的基础上进行改造,用经验池中样本优先级的最大值进行规约,记为 $\tilde{p}^{(i)}$,则

$$\tilde{p}^{(i)}=\frac{p^{(i)}}{\max\limits_{k} p^{(k)}}. \tag{8}$$

② 样本的回放周期. 样本的回放周期用样本距离上次参与训练的时间步 $\tau^{(i)}$ 来表示,$\tau^{(i)}$ 在经验池中的分布形式如图 3 所示. 训练中样本不断填充经验池,新样本和刚刚参与训练的样本 $\tau^{(i)}=1$,所有样本的 $\tau^{(i)}$ 不超过经验池样本总量,因此所有样本 $\tau^{(i)}$ 的取值范围均在 1 到经验池样本总量之间. 类似地,将 $\tau^{(i)}$ 用全部样本的最大值作规约,定义为回放周期,记为 $\tilde{\tau}^{(i)}$,则

$$\tilde{\tau}^{(i)}=\frac{\tau^{(i)}}{\max\limits_{k} \tau^{(k)}}. \tag{9}$$

2) 样本标签. 样本标签为样本的优先级偏差,是真实优先级和存储优先级之差. 其中真实优先级的计算基于 TD-error 值 $\delta^{(i)}_{\text{real}}$,按式(4)在当前时间步将经验池中所有样本送入 Q 网络和目标 Q 网络进行计算. 将优先级偏差重新记为 $\Delta\tilde{p}^{(i)}$,则

$$\Delta\tilde{p}^{(i)}=\tilde{p}^{(i)}_{\text{real}}-\tilde{p}^{(i)}, \tag{10}$$

其中,$\tilde{p}^{(i)}$ 是存储优先级,由式(8)计算. $\tilde{p}^{(i)}_{\text{real}}$ 是真实优先级,其值是在式(6)的 $p^{(i)}_{\text{real}}$ 的基础上规约得到的:

$$\tilde{p}^{(i)}_{\text{real}}=\frac{p^{(i)}_{\text{real}}}{\max\limits_{k} p^{(k)}_{\text{real}}}. \tag{11}$$

Atari 游戏 Pong① 和 Breakout② 在训练时间步为 10^5 时经验池中所有样本的样本特征 $\tilde{p}^{(i)}$,$\tilde{\tau}^{(i)}$ 和样本标签 $\Delta\tilde{p}^{(i)}$ 的分布如图 8 所示,样本按照存储优先级排序,具体的设置将在实验部分介绍. 图 8 表明,样本的存储优先级 $\tilde{p}^{(i)}$ 和回放周期 $\tilde{\tau}^{(i)}$ 都与优先级偏差 $\Delta\tilde{p}^{(i)}$ 密切相关.

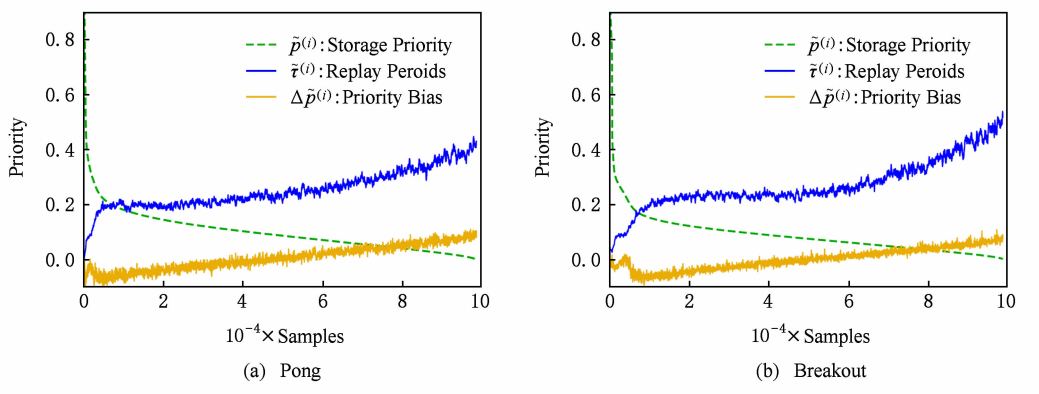


Fig. 8 Distribution of feature $\tilde{p}^{(i)}$, $\tilde{\tau}^{(i)}$ and label $\Delta\tilde{p}^{(i)}$ in Pong and Breakout when $t=10^5$

图 8 Pong 和 Breakout 在 $t=10^5$ 时样本特征 $\tilde{p}^{(i)}$,$\tilde{\tau}^{(i)}$ 和样本标签 $\Delta\tilde{p}^{(i)}$ 的分布

3) 偏差模型参数求解. 使用回归模型来对样本特征和样本标签之间的关系进行建模. 由于样本特征与样本标签之间的关系明显,因此使用线性回归模型. 同时,线性回归模型在当前的输入输出条件下存在闭式解,不需要使用梯度法迭代求解,时间开销很小.

在特征 $\tilde{p}^{(i)}$ 和 $\tilde{\tau}^{(i)}$ 的基础上提取 K 阶多项式特征作为偏差模型的输入. 以 $K=2$ 为例,特征向量 $\boldsymbol{x}^{(i)}$ 为 6 维,表示为

$$(\boldsymbol{x}^{(i)})^{\mathrm{T}}=(1,\tilde{p}^{(i)},\tilde{\tau}^{(i)},(\tilde{p}^{(i)})^2,(\tilde{\tau}^{(i)})^2,\tilde{p}^{(i)}\tilde{\tau}^{(i)}). \tag{12}$$

权重向量 $\boldsymbol{w}$ 的维度与特征 $\boldsymbol{x}^{(i)}$ 的维度相同. 根据线性回归模型的定义,假设函数 $h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})$ 表示为

$$h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})=\sum_{j=0}^{5} w_j x_j^{(i)}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}^{(i)}. \tag{13}$$

设当前经验池中样本数为 m,则全体样本组成的特征矩阵 $\boldsymbol{X}\in\mathbb{R}^{m\times 6}$,表示为

$$\boldsymbol{X}=\begin{pmatrix}(\boldsymbol{x}^{(1)})^{\mathrm{T}}\\(\boldsymbol{x}^{(2)})^{\mathrm{T}}\\\vdots\\(\boldsymbol{x}^{(m)})^{\mathrm{T}}\end{pmatrix}. \tag{14}$$

样本标签为优先级偏差 $\Delta\tilde{p}^{(i)}$,经验池中所有样本组成的标签向量表示为

$$\boldsymbol{y}^{\mathrm{T}}=(\Delta\tilde{p}^{(1)},\Delta\tilde{p}^{(2)},\cdots,\Delta\tilde{p}^{(m)}). \tag{15}$$

损失函数为平方误差损失,记为 $J(\boldsymbol{w})$. 经验池中所有样本的平均损失表示为

① https://gym.openai.com/envs/Pong-v0

② https://gym.openai.com/envs/Breakout-v0

$$J(\boldsymbol{w})=\frac{1}{2m}(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}). \tag{16}$$

参数向量 $\boldsymbol{w}$ 为使得损失 $J(\boldsymbol{w})$ 取最小值的解. 由于特征矩阵 $\boldsymbol{X}$ 的元素均大于 0, $\boldsymbol{w}$ 存在闭式解. 经推导可得, 最小化损失函数 $J(\boldsymbol{w})$ 的 $\boldsymbol{w}$ 的解为

$$\boldsymbol{w}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}. \tag{17}$$

偏差模型表征了算法在训练过程中的存储优先级、回放周期与优先级偏差之间的关系. 偏差模型建立后, 只需利用最新的模型参数就可预测当前经验池中每个样本的优先级偏差, 进而得到真实优先级的估计 $\hat{p}_{\mathrm{real}}^{(i)}$.

2.2.2 优先级校正

优先级校正是根据偏差模型参数 $\boldsymbol{w}$ 估计当前经验池样本真实优先级的过程. 输入当前时间步 t 经验池中样本组成的特征矩阵 $\boldsymbol{X}_t$, 预测每个样本的优先级偏差 $\Delta\bar{p}_t^{(i)}$, 进而得到真实优先级的估计. 主要过程分为样本特征提取和优先级预测.

对时间步 t 的经验池样本提取特征, 特征仍表示为存储优先级 $\bar{p}_t^{(i)}$ 和回放周期 $\bar{\tau}_t^{(i)}$, 并在此基础上进行 K 阶多项式特征提取, 如式(12)所示, 样本 $e^{(i)}$ 的特征向量记为 $\boldsymbol{x}_t^{(i)}$. 经验池中全部样本组成特征矩阵 $\boldsymbol{X}_t^{\mathrm{T}}=((\boldsymbol{x}_t^{(1)})^{\mathrm{T}},(\boldsymbol{x}_t^{(2)})^{\mathrm{T}},\cdots,(\boldsymbol{x}_t^{(i)})^{\mathrm{T}})$.

根据当前最新的偏差模型参数 $\boldsymbol{w}$ 和特征矩阵 $\boldsymbol{X}_t$ 可以预测时间步 t 所有样本的优先级偏差 $\Delta\hat{p}_t^{(i)}$, 如式(18)所示:

$$(\Delta\hat{p}_t^{(1)},\Delta\hat{p}_t^{(2)},\cdots,\Delta\hat{p}_t^{(m)})^{\mathrm{T}}=\boldsymbol{X}_t\cdot\boldsymbol{w}. \tag{18}$$

在此基础上, 将 $\Delta\hat{p}_t^{(i)}$ 与存储优先级求和即可得到样本真实优先级的估计 $\hat{p}_{\mathrm{real}}^{(i)}$, 如式(19)所示:

$$\hat{p}_{\mathrm{real}}^{(i)}=\Delta\hat{p}_t^{(i)}+\bar{p}_t^{(i)}. \tag{19}$$

图 9 显示了 Breakout 在训练过程中 $t=10^7$ 时间步的存储优先级、真实优先级和真实优先级的估计值 $\hat{p}_{\mathrm{real}}^{(i)}$ 的分布. 图 9 表明, 相比于存储优先级, 校正后的优先级 $\hat{p}_{\mathrm{real}}^{(i)}$ 更加逼近于真实优先级的分布.

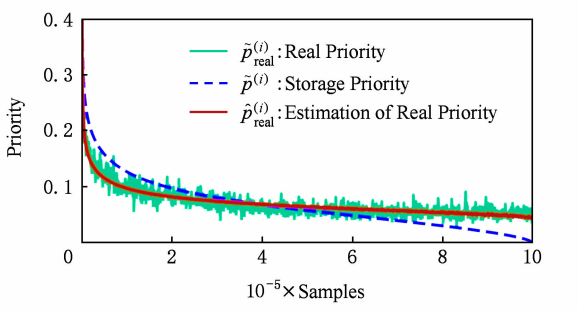


Fig. 9 Distribution of $\bar{p}^{(i)}$, $\tilde{p}_{\mathrm{real}}^{(i)}$ and $\hat{p}_{\mathrm{real}}^{(i)}$ at step $t=10^7$ in Breakout training

图 9 Breakout 训练中 $t=10^7$ 时 $\bar{p}^{(i)}$, $\tilde{p}_{\mathrm{real}}^{(i)}$ 和 $\hat{p}_{\mathrm{real}}^{(i)}$ 分布

2.3 算法描述和复杂度分析

2.3.1 算法描述

ATDC-PER 的算法流程如算法 1 所示.

算法 1. ATDC-PER 算法.

输入: 偏差模型特征阶数 K、偏差模型更新周期 D、奖励折扣因子 γ、样本批量大小 k、学习率 η、经验池容量 N、偏差模型参数 α、重要性权重 β、训练终止时间步 T、目标 Q 网络更新频率 L;

输出: Q 网络参数;

初始化: 经验池为空, 批量梯度 $\boldsymbol{\Delta}=\boldsymbol{0}$, 初始化 Q 网络和目标 Q 网络参数为 $\boldsymbol{\theta}$ 和 $\boldsymbol{\theta}^{-}$.

① 根据初始状态 S_0, 以 ε-贪心选择动作 $A_0\sim\pi_{\boldsymbol{\theta}}(S_0)$;

② for $t=1$ to T do

③ 观察到状态 S_t 并获得奖励 R_t;

④ 将经验序列 $(S_{t-1},A_{t-1},R_t,S_t)$ 存储到经验池中, 并赋予当前经验池中最大的优先级;

⑤ if $t\%D==0$

⑥ 样本特征提取, 构造如式(14)所示的特征矩阵 $\boldsymbol{X}$; /*2.2.1 节*/

⑦ 样本标签提取, 根据式(10)计算 $\Delta\bar{p}^{(i)}$, 得到如式(15)所示的样本标签向量 $\boldsymbol{y}$; /*2.2.1 节*/

⑧ 偏差模型求解, 将偏差模型参数更新为 $\boldsymbol{w}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$; /*2.2.1 节*/

⑨ end if

⑩ 构造样本特征 $\boldsymbol{X}_t$, 预测经验池样本优先级偏差; $(\Delta\hat{p}_t^{(1)},\Delta\hat{p}_t^{(2)},\cdots,\Delta\hat{p}_t^{(m)})^{\mathrm{T}}=\boldsymbol{X}_t\boldsymbol{w}$; /*2.2.2 节*/

⑪ 得到真实优先级的估计 $\hat{p}_{\mathrm{real}}^{(i)}=\Delta\hat{p}_t^{(i)}+\bar{p}_t^{(i)}$; /*2.2.2 节*/

⑫ 根据校正后的优先级提取 k 个样本 $j\sim P(j)=\hat{p}_{\mathrm{real}}^{(i)}/\sum_k\hat{p}_{\mathrm{real}}^{(k)}$;

⑬ for $j=1$ to k

⑭ 计算重要性权重 $\phi_j=(N\times P(j))^{-\beta}/\max_k\phi_k$;

⑮ 计算 TD-error $\delta_j=R_j+\gamma Q(S_j,\arg\max_a Q(s_{j-1},a;\boldsymbol{\theta});\boldsymbol{\theta}^{-})-Q(S_{j-1},A_{j-1})$;

⑯ 根据 TD-error 更新经验池中样本的优先级;

⑰ 累计批量的梯度 $\boldsymbol{\Delta}=\boldsymbol{\Delta}+\phi_j\times\delta_j\times\nabla_{\boldsymbol{\theta}}Q(S_{j-1},A_{j-1},\boldsymbol{\theta})$;

⑱ end for

⑲ 更新 Q 网络权重 $\boldsymbol{\theta}=\boldsymbol{\theta}+\eta\boldsymbol{\Delta}$,重置 $\boldsymbol{\Delta}=\mathbf{0}$;

⑳ 如果 $t\%L==0$,更新目标 Q 网络参数 $\boldsymbol{\theta}^{-}=\boldsymbol{\theta}$;

㉑ 以ε-贪心选择下一个动作 $A_t\sim\pi_{\boldsymbol{\theta}}(S_t)$;

㉒ end for

其中行⑤~⑨是偏差模型更新的过程,需在 t,$t+D$,$t+2D$,…等时间步计算经验池的样本特征和优先级偏差进行训练,求解模型参数 $\boldsymbol{w}$. 行⑩⑪是优先级校正的过程,通过偏差模型得到样本真实优先级的估计 $\hat{p}_{\text{real}}^{(i)}$. 行⑫在优先级的基础上加入随机性,样本被采样的概率与优先级成正比,同时所有样本都有机会被采样. 行⑬~⑱计算损失函数的梯度,计算梯度时需在 TD-error 的基础上乘以重要性权重(importance sampling weight)系数[42],原因是按照优先级 $\hat{p}_{\text{real}}^{(i)}$ 采样与均匀采样相比会给梯度的计算带来误差,因此使用采样概率的比值 ϕ 进行补偿[11],并使用全部样本的 ϕ 的最大值进行规约,如式(20)所示.

$$\phi_j=(N\times P(j))^{-\beta}/\max_k\phi_k, \tag{20}$$

其中,$k=1,2,\cdots,N$,N 为经验池容量,β 是一个常数,$P(j)$为样本 $e^{(j)}$ 的采样优先级在全体样本中所占的比例,如式(21)所示.

$$P(j)=(\hat{p}_{\text{real}}^{(j)})^{\alpha}\Big/\sum_k(\hat{p}_{\text{real}}^{(k)})^{\alpha}. \tag{21}$$

2.3.2 复杂度分析

ATDC-PER 算法包括偏差模型和优先级校正两部分. 根据分段更新策略,在执行 D 次优先级校正的同时会执行一次偏差模型更新. 由于偏差模型更新的时间复杂度较高,而优先级校正仅执行矩阵乘法和加法,时间复杂度低,所以分段更新策略可以使复杂度高的部分以极低的频率执行,从而减少时间开销.

1) 偏差模型的复杂度分析. 偏差模型更新包括样本特征提取、样本标签提取和模型求解 3 个阶段,分析如下:

① 样本特征提取. 原始的样本特征维度为 2,提取 K 阶多项式后的特征向量 $\boldsymbol{x}^{(i)}\in\mathbb{R}^{\frac{K^2+3K+2}{2}\times1}$,特征矩阵 $\boldsymbol{X}\in\mathbb{R}^{m\times\frac{K^2+3K+2}{2}}$. 其中,$m$ 的值不超过经验池总容量,且在经验池填充满之后不再变化,是一个常数. 因此特征提取的时间复杂度和空间复杂度均为 $O(K^2)$.

② 样本标签提取. 得到样本的优先级偏差需计算样本真实的 TD-error 值 $\delta_{\text{real}}^{(i)}$,需将经验池的全部样本送入 Q 网络和目标 Q 网络进行前向传播. 由于经验池大小为常数,因此样本组成的输入矩阵维度固定,计算的时间复杂度和空间复杂度都为 $O(1)$.

③ 偏差模型求解. 模型求解阶段的时间消耗集中在式(17)中对参数向量 $\boldsymbol{w}$ 的求解上. $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\in\mathbb{R}^{\frac{K^2+3K+2}{2}\times\frac{K^2+3K+2}{2}}$,在此基础上求逆的时间复杂度为 $O(K^6)$,空间复杂度为 $O(K^2)$.

2) 优先级校正的复杂度分析. 优先级校正阶段在提取样本特征 $\boldsymbol{X}_t$ 后与参数向量 $\boldsymbol{w}$ 执行矩阵乘法,时间复杂度为 $O(K^4)$,空间复杂度为 $O(K^2)$.

由于 ATDC-EPR 算法中间隔 D 个时间步偏差模型才更新一次,D 的取值较大,其余时间步都仅执行优先级校正的操作,因此总体时间复杂度为 $O(K^4)$,空间复杂度为 $O(K^2)$. 另外,一般 $K=2$ 时偏差模型就可以很好地估计优先级偏差,此时求解参数 $\boldsymbol{w}$ 所需的逆矩阵 $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\in\mathbb{R}^{6\times6}$,规模较小,可以快速求解.

经过复杂度分析,本文提出的 ATDC-PER 算法以极小的时间和空间代价就可以在 Q 网络每次迭代中校正经验池中样本的存储优先级,使用校正后的优先级选择样本.

3 实验结果与分析

本节首先介绍实验平台和环境,随后介绍 ATDC-PER 算法的参数选择,通过对比实验说明参数对性能的影响,最后从智能体的学习速度和最优策略的质量 2 方面对实验结果进行分析.

3.1 实验平台描述

实验使用基于 Atari 2600 游戏平台 Arcade Learning Environment(ALE)[43]的 OpenAI 集成环境[44]. ALE 平台包含数十个视频游戏,并对外提供接口. ALE 平台的挑战在于游戏种类和数量丰富,包含益智类、体育类、策略类、射击类、搏斗类等多种类型的游戏,需多个游戏中使用同一算法,在超参数固定的情况下,仅以原始屏幕图像作为输入进行学习和训练.

本文选择 ALE 平台中的 19 个游戏进行实验,游戏的简要介绍列于表 1,各游戏界面如图 10 所示. 可以看出,游戏类型多样,游戏的界面风格不同,在训练中 Q 网络的输入不同. 同时,每个游戏中智能体需完成的任务不同,即最终学习的策略也不同.

Table 1 A Brief Introduction to 19 Atari Games

表 1 19 个 Atari 游戏的简要介绍

Game	Type	Action Number	Brief Introduction
Alien	puzzle	18	Eat beans and extra bonus while avoiding enemies to chase
Amidar	strategy	10	The agent passes all tracks without touching the enemy in the maze
BankHeist	strategy	18	Reaches the prey to gain points, and the prey becomes an enemy chase
BeamRider	shooter	9	The agent launches artillery shells and hits each other
Boxing	wrestling	18	The agent avoid hitting the opponent while fist hitting the opponent
Breakout	strategy	4	The agent receives and ejects the ball, destroying the upper box in the shortest time
Centipede	strategy	18	The agent avoids the rapid emergence of bullets while hitting the top of the prey
ChopperCommand	shooter	18	Driving helicopters to shoot down opponent helicopters while avoiding ground
CrazyClimber	puzzle	9	Escape the enemy's blows, avoid obstacles and climb high-rise buildings
DoubleDunk	sports	19	2 vs 2 basketball game, one side with the ball and the other defensive
Enduro	driving	9	The agent need to avoid collisions while driving a car on the highway
NameThisGame	shooter	6	Shooting fast-moving sharks in the sea to avoid enemy artillery shells
Pong	sports	6	Agent prevents the opponent from receiving the ball in the table tennis game
PrivateEye	strategy	18	The agent jumps from the ground to meet the enemy's score in the changing scene
Riverraid	shooter	18	The agent drives aircraft to hit different targets and score different goals.
RoadRunner	strategy	18	The agent walks on fast moving roads, avoids obstacles and destroys enemies.
Robotank	shooter	18	Driving tanks move their own field of vision, and score quickly
TimePilot	shooter	10	In air combat, we need to drive planes, avoid obstacles and strike enemy planes
UpNDown	action	6	The agent hits the rear vehicle and reaches the destination as fast as possible

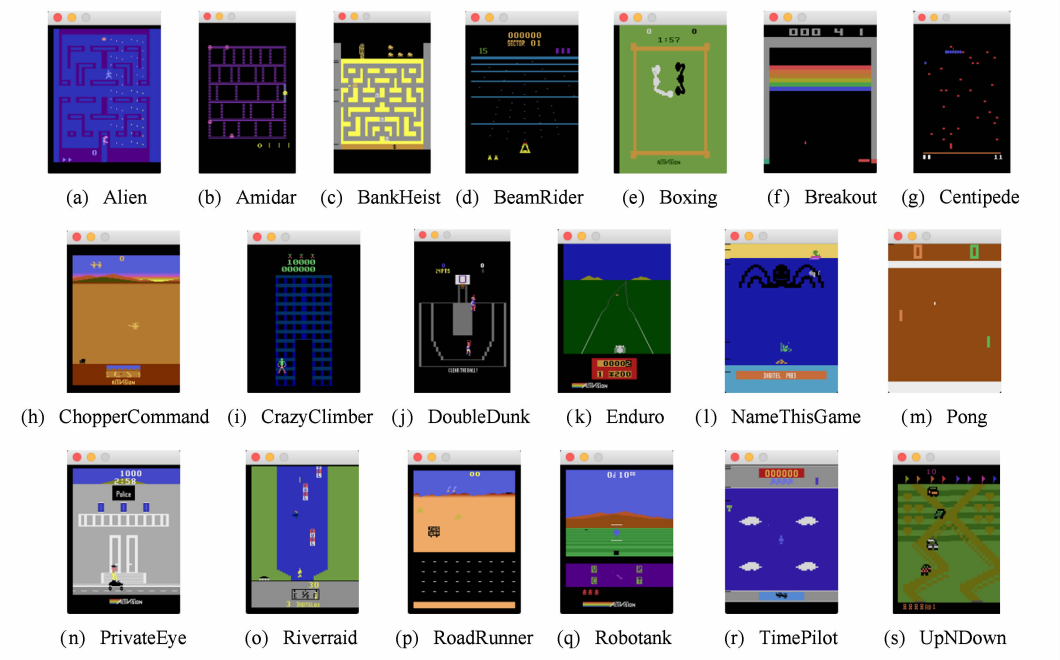


Fig. 10 State representations of 19 Atari games

图 10 19 个 Atari 游戏的状态表示

本文使用的计算平台为戴尔 PowerEdge T630 塔式工作站，工作站配置 2 颗 Intel Xeon E5-2609 CPU、4 块 NVIDIA GTX-1080Ti 显卡（GPU）和 64 GB内存，每个游戏使用单独的一块显卡进行训练. 卷积神经网络基于 Tensorflow[45] 开源库实现，本文核心代码、实验结果和测试视频均可下载①.

① http://pr-ai.hit.edu.cn/2018/0510/c1049a207703/page.htm

3.2 ATDC-PER 参数选择

ATDC-PER 算法在全部游戏中使用相同的超参数集合,以验证模型的通用性. ATDC-PER 算法在 PER 算法的基础上增加了 2 个超参数,分别是偏差模型更新周期 D 和偏差模型特征阶数 K,其余参数的设置与 PER 算法的设置相同. 状态表示为原始图像预处理并叠加 4 帧组成的 $84\times84\times4$ 的矩阵;Q 网络为 3 层卷积神经网络,输出每个动作的值函数,目标 Q 网络的更新频率设为 40 000;鉴于不同游戏的分值计算方式不同,使用符号函数对奖励进行规约;为了使训练更加稳定,对回传的梯度进行限制,使其不超过 10. 使用基于平方误差损失的 Adam 优化器进行训练,批量大小 $k=32$,学习率设为 10^{-4},折扣因子 $\gamma=0.99$;探索因子在前 400 万帧从 1.0 线性递减至 0.1,在随后的时间步内以之前速度的 1/10 线性递减至 0.01 后保持不变;经验池容量设为 10^6;优先级计算中使用的参数 $\alpha=0.6$;重要性权重计算中使用的参数 β 初始值设为 0.4,在训练中线性增加至 1.0.

3.2.1 偏差模型更新周期

通过在 Alien,Breakout,Pong 的训练过程中使用不同的偏差模型更新周期 D 来测试参数 D 对 Q 网络学习的影响. 在训练过程中的时刻 t 训练偏差模型得偏差模型参数 $\boldsymbol{w}_t$,使用不同的参数 D,测试模型 $\boldsymbol{w}_t$ 在位于时刻 $t+D$ 的经验池样本的平均损失,损失的计算如式(16)所示. 不同的偏差模型更新周期 D 下的平均损失对比列如表 2 所示:

Table 2 Average Loss of Bias Model Trained in t Steps and Tested in $t+D$ Steps Under Different D Values

表 2 比较不同 D 值下时刻 t 训练的偏差模型在时刻 $t+D$ 测试的平均损失

$10^5\times t$	Alien, $10^{-5}\times D$					Breakout, $10^{-5}\times D$					Pong, $10^{-5}\times$D				
	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
15	0.0033	0.0031	0.0031	0.0028	0.0032	0.0012	0.0011	0.0016	0.0016	0.0033	0.0042	0.0042	0.0044	0.0050	0.0043
20	0.0028	0.0038	0.0055	0.0106	0.0048	0.0029	0.0037	0.0038	0.0033	0.0085	0.0042	0.0044	0.0052	0.0044	0.0048
25	0.0026	0.0021	0.0021	0.0026	0.0045	0.0009	0.0005	0.0011	0.0021	0.0017	0.0040	0.0042	0.0045	0.0041	0.0042
30	0.0015	0.0017	0.0016	0.0014	0.0039	0.0004	0.0004	0.0007	0.0011	0.0011	0.0030	0.0037	0.0032	0.0028	0.0036
35	0.0032	0.0182	0.1033	0.0249	0.0051	0.0007	0.0009	0.0009	0.0008	0.0009	0.0026	0.0046	0.0030	0.0032	0.0029
40	0.0039	0.0181	0.0121	0.0149	0.0241	0.0004	0.0012	0.0018	0.0025	0.0013	0.0015	0.0031	0.0027	0.0020	0.0029

由表 2 可知,随着偏差模型更新周期 D 的增加,损失总体升高. 但从数值上看,损失的升高并不明显,特别是在 10^5 个时间步内,因此实验中将 D 的值设置为 10^5. 表 2 数据表明本文提出的偏差模型具有较强的泛化能力,对偏差模型更新周期 D 的选择不敏感,偏差模型可以在训练后较长的时间步内适用,同时也验证了图 7 所示的分段更新策略的合理性.

3.2.2 偏差模型特征阶数

偏差模型的特征阶数 K 越大,模型复杂度越高,训练误差越小,但计算复杂度也随之升高,同时可能引起过拟合. 通过在游戏 Alien,Breakout,Pong 的训练过程中进行粗搜索来选择 K 的值. 实验使用不同的 K 值进行训练,记录多个位于偏差模型更新周期中心的经验池样本,比较经验池样本的平均损失,结果列于表 3. 表 3 中数据表明,随着 K 值的增加,损失总体下降,但在 $K>2$ 时损失的变化已经不显著,同时在 Alien 中 $K>3$ 时有过拟合的趋势. 因此,综合考虑损失和计算消耗,本文实验将多项式特征的阶数设为 $K=2$.

Table 3 Average Loss at the Center of Renewal Period of Correction Model Under Different K Values

表 3 比较不同 K 值下处于偏差模型更新周期中心的经验池平均损失

K	Alien					Breakout					Pong				
	$10^{-5}\times t$				AVG	$10^{-5}\times t$				AVG	$10^{-5}\times t$				AVG
	4.5	9.5	14.5	19.5		4.5	9.5	14.5	19.5		4.5	9.5	14.5	19.5	
1	0.0108	0.0088	0.0069	0.0042	**0.0077**	0.0015	0.0035	0.0032	0.0036	**0.0030**	0.0043	0.0059	0.0054	0.0034	**0.0048**
2	0.0105	0.0082	0.0035	0.0028	**0.0063**	0.0014	0.0033	0.0030	0.0032	**0.0027**	0.0038	0.0052	0.0049	0.0032	**0.0043**
3	0.0096	0.0071	0.0025	0.0022	**0.0054**	0.0013	0.0032	0.0028	0.0032	**0.0026**	0.0037	0.0051	0.0048	0.0031	**0.0042**
4	0.0095	0.0072	0.0028	0.0021	**0.0054**	0.0013	0.0032	0.0028	0.0032	**0.0026**	0.0037	0.0051	0.0048	0.0031	**0.0042**
5	0.0095	0.0070	0.0035	0.0022	**0.0056**	0.0013	0.0032	0.0027	0.0032	**0.0026**	0.0036	0.0051	0.0048	0.0031	**0.0042**

3.3 对比实验与分析

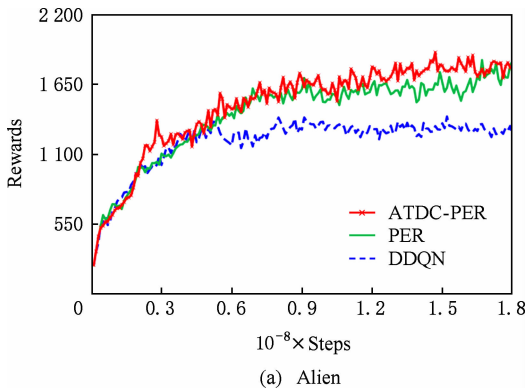
本文选择 2 个基线算法进行对比，分别为 DDQN[27] 和优先经验回放[11]（PER）. 其中，DDQN 算法中样本没有优先级，在训练时从经验池中随机取样；PER 算法在 DDQN 的基础上为每个样本设置存储优先级，存储优先级与样本上次参与训练时的 TD-error 绝对值成正比，在采样时根据存储优先级依概率选择样本. 本文通过偏差模型对存储优先级进行校正，用校正后的真实优先级的估计进行采样，提高了样本的利用效率.

算法的评价标准包括智能体的学习速度和智能体学习到的最优策略的质量.

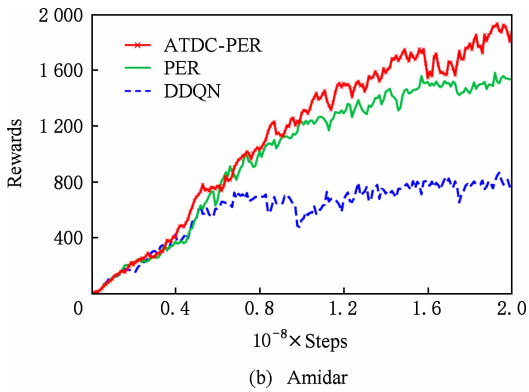
3.3.1 智能体的学习速度

图 11 显示了 19 个 Atari 游戏的训练曲线，横轴代表时间步，每个时间步智能体与 ALE 环境交互一次，ALE 环境会产生一帧新的图像. 纵轴代表智能体在周期内获得的累计奖励. 在训练开始后，随着迭代的进行，智能体的水平不断提高，奖励值不断增大，达到峰值后神经网络的训练可能趋于过拟合，因此训练后期奖励值可能会有波动.

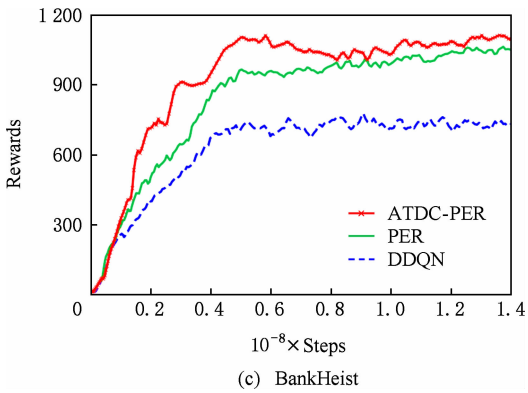
ATDC-PER 算法在 19 个游戏中的 15 个游戏收敛速度更快，能够以更少的交互次数达到最大奖励，如图 11 所示. 特别是在 Breakout，DoubleDunk，Enduro，Pong，Robotank，UpNDown 这 6 个游戏中



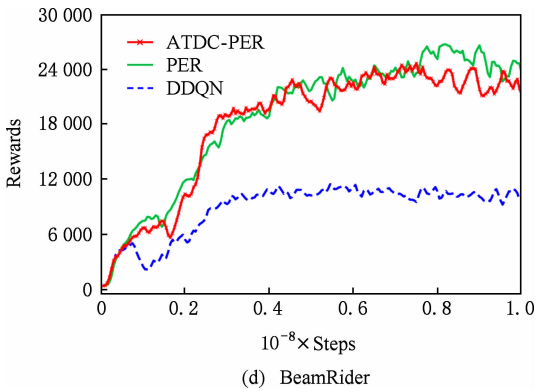
(a) Alien



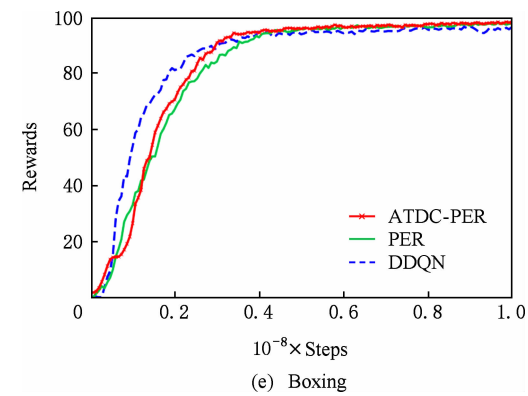
(b) Amidar



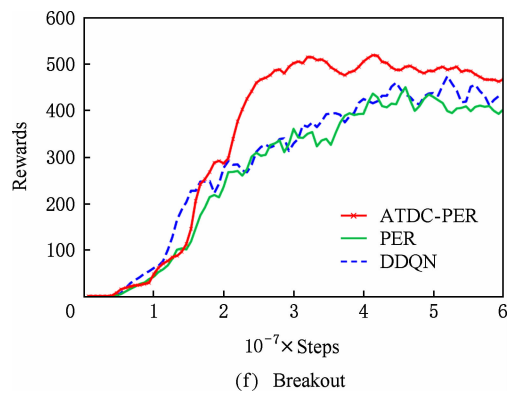
(c) BankHeist



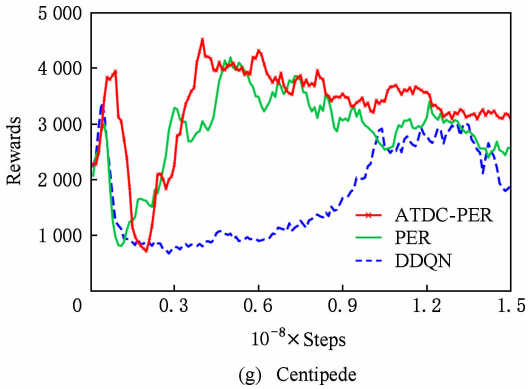
(d) BeamRider



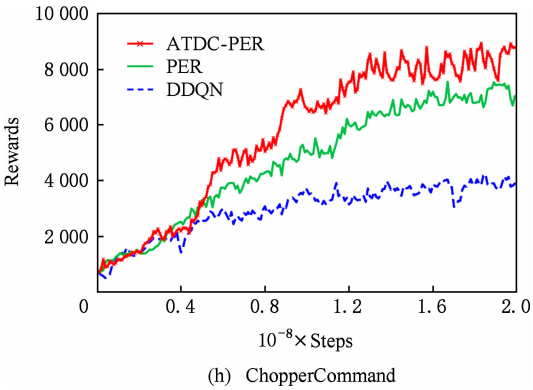
(e) Boxing



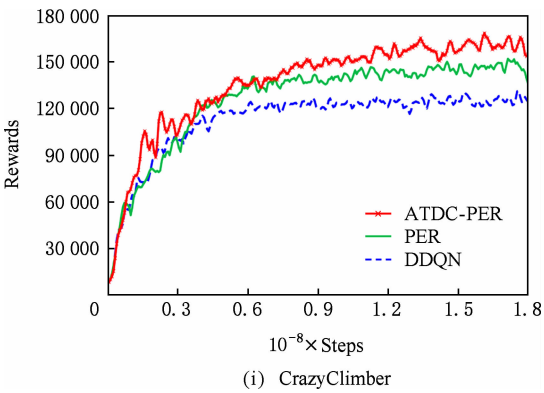
(f) Breakout



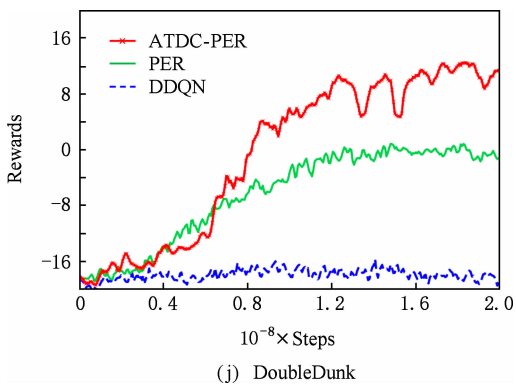
(g) Centipede



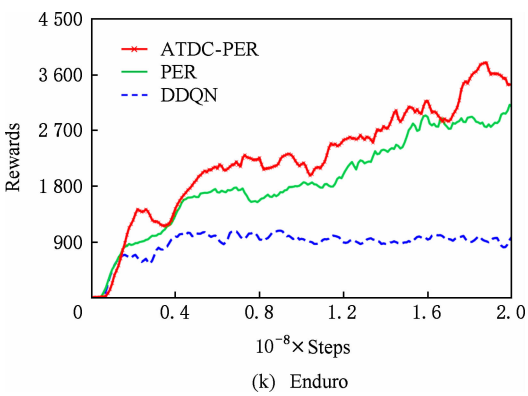
(h) ChopperCommand



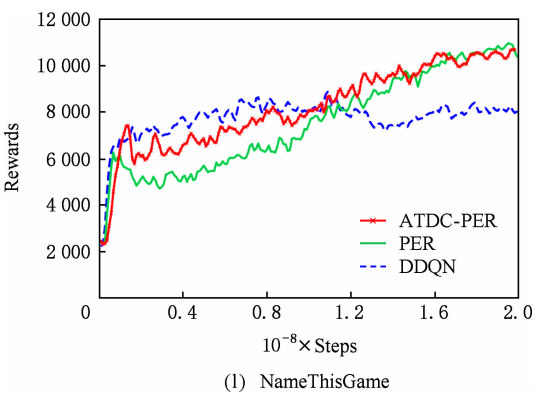
(i) CrazyClimber



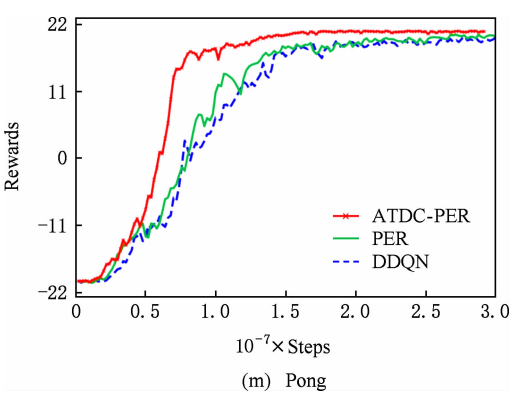
(j) DoubleDunk



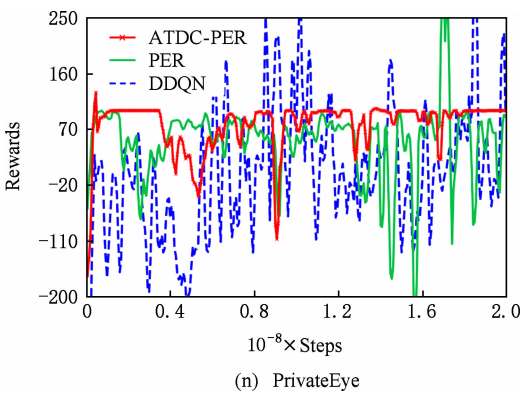
(k) Enduro



(l) NameThisGame



(m) Pong



(n) PrivateEye

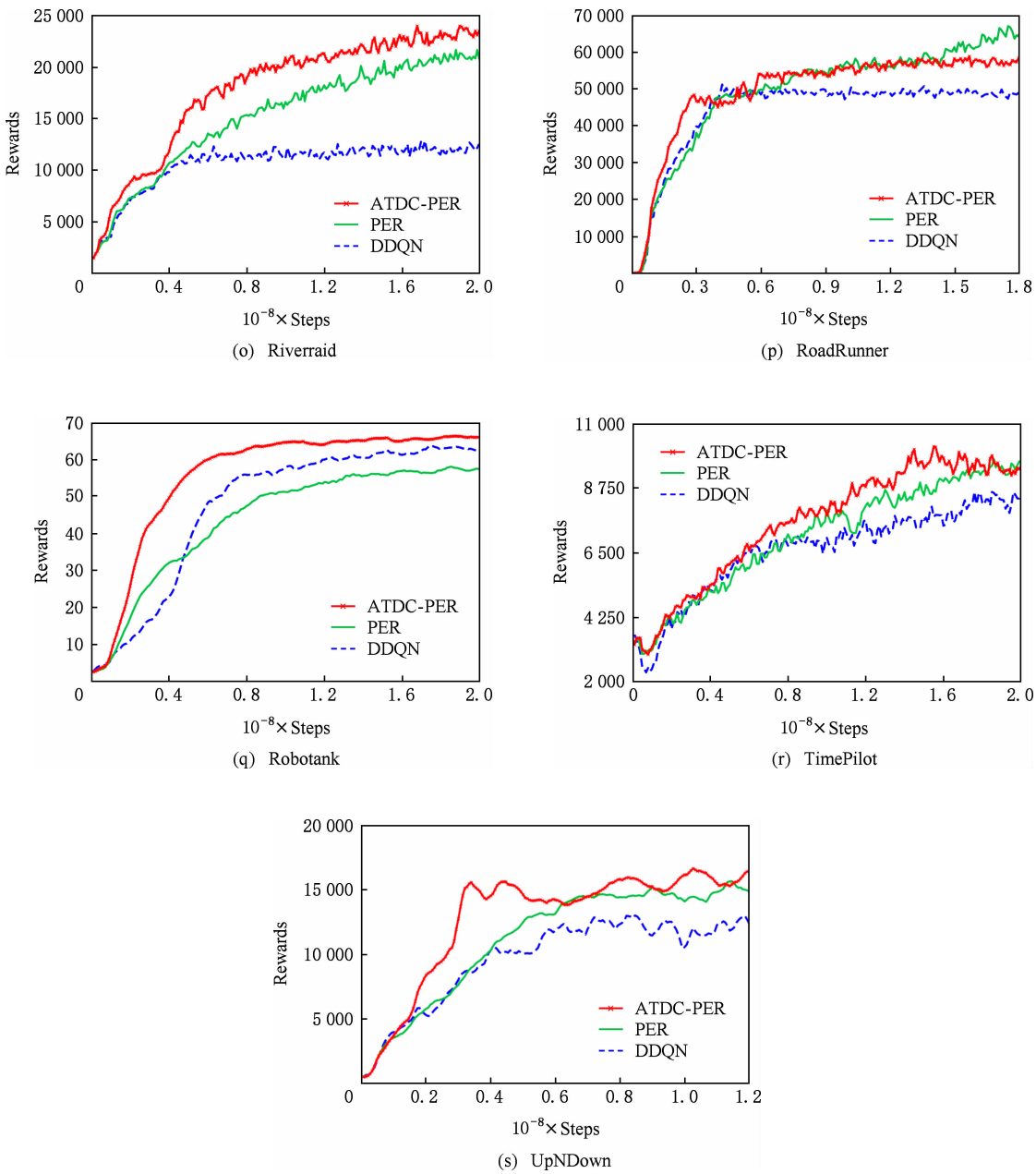


Fig. 11 Comparison of our method with PER[11] and DDQN[27] algorithms in training curve of 19 Atari games

图 11 本文方法在 19 个 Atari 游戏中与 PER[11] 和 DDQN[27] 算法的训练曲线对比

速度提升明显,分别如图 11(f)(j)(k)(m)(q)(s)所示,在训练的相同时间步,ATDC-PER算法中智能体的得分水平总体优于 DDQN 和 PER 算法. 图 11 中的曲线表明,本文提出的 ATDC-PER 算法通过在训练中校正经验池样本的存储优先级,提高了经验池样本的利用效率,提升了智能体的学习速度,减少了智能体与环境的交互次数,使智能体以更少的迭代步收敛.

3.3.2 最优策略的质量

最优策略反映智能体最终的学习结果,训练结束后使用最优策略可以指导智能体在 ALE 环境中获得良好表现. ATDC-PER 算法中策略表示为 Q 网络参数,训练结束后智能体可以得到完整可复用的策略. 训练过程中保存每个时间步所在周期的奖励值,同时每隔 10^5 个时间步保存一次 Q 网络参数,训练结束后选择奖励曲线平滑后处于峰值点的 Q 网络参数作为最优策略,通过测试最优策略的质量对学习效果进行评价.

最优策略的评价方法与 DDQN[27] 中相同,首先,周期开始时智能体与环境随机进行 1～31 次无动作交互,为测试提供随机的初始状态. 随后,智能体按照最优策略的 ε-贪心策略选择动作,探索因子

为 0.05.每个周期智能体与环境最多进行 18 000 个时间步的交互,策略的最终得分为 100 个测试周期得分的平均值.鉴于每个游戏的分值计算方式不同,使用式(22)在实际得分的基础上计算规约得分.

$$score_{\text{normalized}}=\frac{score_{\text{agent}}-score_{\text{random}}}{|score_{\text{human}}-score_{\text{random}}|},\quad(22)$$

其中,$score_{\text{random}}$ 为随机智能体的得分,$score_{\text{human}}$ 为人类专家的得分,规约后的分值大于 100% 表明智能体的水平已经超过了人类专家水平.

19 个 Atari 游戏总体的最优策略评价结果比较列于表 4,单个游戏的评价结果列于表 5.表 5 中随机智能体(random)、人类专家(human)和 DDQN 算法的得分来源于文献[27],PER 算法的得分来源于文献[11].

Table 4 Summary of Normalized Score on All Games

表 4 全部游戏的规约得分评价表 %

Indicators	DDQN[27]	PER[11]	ATDC-PER
Median	139.0	392.5	**404.5**
Average	335.6	505.6	**588.9**

Table 5 Raw Scores and Normalized Scores on All Games

表 5 全部游戏的实际得分和规约得分表

Game	random	human	Score			Normalized Score/%		
			DDQN[27]	PER[11]	ATDC-PER	DDQN[27]	PER[11]	ATDC-PER
Alien	227.8	6 875.4	2 907.3	3 583.3	**3 856.1**	40.3	50.5	**54.6**
Amidar	5.8	1 675.8	702.1	2 051.8	**2 361.3**	41.7	122.5	**141.0**
BankHeist	14.2	734.4	728.3	1 126.8	**1 196**	99.2	154.5	**164.1**
BeamRider	363.9	5 774.7	7 654	**22 430.7**	22 250.6	134.7	**407.8**	404.5
Boxing	0.1	4.3	81.7	98.8	**99.5**	1 942.9	2 350.0	**2 366.7**
Breakout	1.7	31.8	375.0	381.5	**560.6**	1 240.2	1 261.8	**1 856.8**
Centipede	2 090.9	11 963.2	4 139	5 175.4	**5 738.7**	20.7	31.2	**36.9**
ChopperCommand	811	9 881.8	4 653.0	5 135.0	**6 581.0**	42.4	47.7	**63.6**
CrazyClimber	10 780.5	35 410.5	101 874	183 137	**191 354.9**	369.8	699.8	**733.1**
DoubleDunk	−18.6	−15.5	−6.3	4.8	**12.7**	396.8	754.8	**1 009.7**
Enduro	0	309.6	319.5	2 155.0	**3 827.0**	103.2	696.1	**1 236.1**
NameThisGame	2 292.3	4 076.2	6 997.1	**13 439.4**	12 713.9	263.7	**624.9**	584.2
Pong	−20.7	9.3	21	20.7	**21**	139.0	138.0	**139.0**
PrivateEye	24.9	69 571.3	**670.0**	200.0	361.5	**0.9**	0.3	0.5
Riverraid	1 338.5	13 513.3	12 015.3	20 494	**24 355.3**	87.7	157.3	**189.1**
RoadRunner	11.5	7 845.0	48 377.0	**62 785**	60 344.3	617.4	**801.3**	770.2
Robotank	2.2	11.9	46.7	58.6	**66.5**	458.8	581.4	**662.9**
TimePilot	3 568.0	5 925.0	7 964.0	11 448.0	**11 654.4**	186.5	334.3	**343.1**
UpNDown	533.4	9 082.0	16 769.9	34 082.7	**37 550**	189.9	392.5	**433.0**

3.3.3 综合评价

表 6 显示了 19 个 Atari 游戏中智能体学习速度和最优策略质量的综合评价.结果表明,本文使用偏差模型校正后的真实优先级的估计来选择样本能够在训练中提升智能体的学习速度,同时提高最优策略的质量.本文算法在 2 种评价指标下都取得最好成绩的游戏达 14/19 个,其中在 15/19 个游戏中提升了智能体的学习速度,减少了智能体与环境的交互次数;在 15/19 个游戏中改善了智能体的学习效果,提升了最优策略的质量.Boxing 游戏比较简单,3 种算法均能收敛到接近最大奖励值,本文算法和 PER 算法的收敛速度均略低于 DDQN 算法,但本文方法的最优策略得分优于其余 2 种算法,如图 11(e)所示.BeamRider,NameThisGame 这 2 个游戏不同方法的训练曲线交错上升,最后各算法达到的最优策略质量相似,均明显超过了人类专家水平.RoadRunner 中本文方法的收敛速度优于 DDQN 和 PER 算法,最优策略得分略低于 PER 算法.PrivateEye 游戏较难,3 种算法的最优策略得分均不足人类专家得分的 1%,均没有得到较好的结果.

Table 6 Comprehensive Evaluation on All Games

表 6 全部游戏的综合评价表

Game	Learning Speed			Quality of Optimal Policy		
	DDQN[27]	PER[11]	ATDC-PER	DDQN[27]	PER[11]	ATDC-PER
Alien			■			■
Amidar			■			■
BankHeist			■			■
BeamRider		■			■	
Boxing	■					■
Breakout			■			■
Centipede			■			■
ChopperCommand			■			■
CrazyClimber			■			■
DoubleDunk			■			■
Enduro			■			■
NameThisGame	■				■	
Pong			■			■
PrivateEye	■			■		
Riverraid			■			■
RoadRunner			■		■	
Robotank			■			■
TimePilot			■			■
UpNDown			■			■

Note: ■ shows this method is the best among all of the comparison methods.

由表 4 和表 5 数据可知,ATDC-PER 算法能够在全部 19 个游戏中的 15 个游戏获得更好的最优策略.最优策略规约得分的中位数相对于 PER 算法[11]提高 12%,平均数提高 83.3%.实验结果表明,本文方法通过提高经验池中样本的采样效率,可以改善智能体的最终学习效果,使智能体收敛到更好的最优策略.

4 结 论

深度 Q 学习中,基于优先经验回放的方法在训练中样本的存储优先级不能跟随 Q 网络迭代而更新,存储优先级不能准确反映经验池中样本 TD-error 分布的真实情况,从而降低了大量样本的利用效率.使用样本的真实优先级对经验池样本进行采样能够明显提高经验池样本的利用效率和学习的效果.

本文提出的基于 TD-error 自适应校正的主动采样方法,利用样本的回放周期和 Q 网络状态能够校正样本的优先级偏差,得到样本真实优先级的估计值,以极小的代价逼近真实优先级的分布.偏差模型在训练过程中分段更新,且对更新周期不敏感,偏差模型特征阶数较小时就可以很好地估计优先级偏差,分段更新策略显著降低了本文方法的计算复杂度.在 Q 网络迭代的每个时间步使用校正后的优先级进行采样,能够提升 Q 网络训练中经验池样本的利用效率.在 Atari 2600 中的实验结果表明,使用本文方法进行主动采样提升了智能体的学习速度,减少了智能体与环境的交互次数,同时改善了智能体的学习效果,提升了最优策略的质量.

参 考 文 献

[1] Sutton R S, Barto A G. Reinforcement Learning: An Introduction [M]. Cambridge, MA: MIT Press, 1998: 100-107

[2] Bertsekas D P, Tsitsiklis J N. Neuro-dynamic programming: An overview [C] //Proc of the 34th IEEE Conf on Decision and Control. Piscataway, NJ: IEEE, 1995: 560-564

[3] Tesauro G. Td-gammon: A self-teaching backgammon program [M] //Applications of Neural Networks. New York: Springer, 1995: 267-285

[4] Tesauro G. Programming backgammon using self-teaching neural nets [J]. Artificial Intelligence, 2002, 134(1/2): 181-199

[5] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [C] // Proc of the 26th Int Conf on neural information processing systems(NIPS). Cambridge, MA: MIT Press, 2012: 1097-1105

[6] Liu Quan, Zhai Jianwei, Zhang Zongchang, et al. A survey on deep reinforcement learning [J]. 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[J]. 计算机学报, 2018, 41(1): 1-27)

[7] Li Yuxi. Deep reinforcement learning: An overview [EB/OL]. New York: Cornell University, 2017 [2018-03-04]. https://arxiv.org/abs/1701.07274

[8] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning [EB/OL]. New York: Cornell University, 2013 [2017-10-04]. https://arxiv.org/abs/1312.5602

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

[10] Maei H R, Szepesvari C, Bhatnagar S, et al. Convergent temporal-difference learning with arbitrary smooth function approximation [C] //Proc of the 23rd Int Conf on Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 2009: 1204-1212

[11] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay [C] //Proc of the 4th Int Conf on Learning Representations(ICLR). New York: Springer, 2016: 256-265

[12] Nair A, Srinivasan P, Blackwell S, et al. Massively parallel methods for deep reinforcement learning [EB/OL]. New York: Cornell University, 2015 [2018-03-02]. https://arxiv.org/abs/1507.04296

[13] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning [C] //Proc of the 33rd Int Conf on Machine Learning (ICML). New York: JMLR, 2016: 1928-1937

[14] Andre D, Friedman N, Parr R. Generalized prioritized sweeping [C] //Proc of the 12th Int Conf on Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 1998: 1001-1007

[15] Bellman R. A Markovian decision process [J]. Journal of Mathematics and Mechanics, 1957, 6(4): 679-684

[16] Bertsekas D P. Dynamic programming and suboptimal control: A survey from ADP to MPC [J]. European Journal of Control, 2005, 11(4/5): 310-334

[17] Browne C B, Powley E, Whitehouse D, et al. A survey of Monte Carlo tree search methods [J]. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(1): 1-43

[18] Silver D, Huang Shijie, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484-489

[19] Degris T, Pilarski P M, Sutton R S. Model-Free reinforcement learning with continuous action in practice [C] //Proc of the 52nd American Control Conf. Piscataway, NJ: IEEE, 2012: 2177-2182

[20] Zhu Fei, Zhu Haijun, Fu Yuchen, et al. A kernel based true online Sarsa(λ) for continuous space control problems [J]. Computer Science & Information Systems, 2017, 14(3): 789-804

[21] Zhu Fei, Liu Quan, Fu Qiming, et al. A least square actor-critic approach for continuous action space [J]. Journal of Computer Research and Development, 2014, 51(3): 548-558 (in Chinese)
(朱斐, 刘全, 傅启明, 等. 一种用于连续动作空间的最小二乘行动者-评论家方法[J]. 计算机研究与发展, 2014, 51(3): 548-558)

[22] Liu Quan, Zhang Peng, Zhong Shan, et al. An improved actor-critic algorithm in continuous spaces with action weighting [J]. Chinese Journal of Computers, 2017, 40(6): 1252-1264 (in Chinese)
(刘全, 章鹏, 钟珊, 等. 连续空间中的一种动作加权行动者评论家算法[J]. 计算机学报, 2017, 40(6): 1252-1264)

[23] Barto A, Duff M. Monte Carlo matrix inversion and reinforcement learning [C] //Proc of the 8th Int Conf on Neural Information Processing Systems(NIPS). Cambridge, MA: MIT Press, 1994: 687-694

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

[25] Liu Quan, Zhou Xin, Zhu Fei, et al. Experience replay for least-squares policy iteration [J]. IEEE/CAA Journal of Automatica Sinica, 2015, 1(3): 274-281

[26] Hasselt H V. Double Q-learning [C] //Proc of the 24th Int Conf on Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 2010: 2613-2621

[27] Hasselt H V, Guez A, Silver D. Deep reinforcement learning with double Q-learning [C] //Proc of the 30th AAAI Conf on Artificial Intelligence. Menlo Park, CA: AAAI, 2016: 2094-2100

[28] Osband I, Blundell C, Pritzel A, et al. Deep exploration via bootstrapped DQN [C] //Proc of the 30th Int Conf on Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 2016: 4026-4034

[29] Tang Haoran, Houthooft R, Foote D, et al. Exploration: A study of count-based exploration for deep reinforcement learning [C] //Proc of the 31st Int Conf on Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 2017: 2750-2759

[30] Bellemare M, Srinivasan S, Ostrovski G, et al. Unifying count-based exploration and intrinsic motivation [C] //Proc of the 30th Int Conf on Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 2016: 1471-1479

[31] Stadie B C, Levine S, Abbeel P. Incentivizing exploration in reinforcement learning with deep predictive models [EB/OL]. New York: Cornell University, 2015 [2018-03-01]. https://arxiv.org/abs/1507.00814

[32] Wang Ziyu, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning [C] //Proc of the 33rd Int Conf on Machine Learning(ICML). New York: JMLR, 2016: 1995-2003

[33] Anschel O, Baram N, Shimkin N. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning [C] //Proc of the 34th Int Conf on Machine Learning (ICML). New York: JMLR, 2017: 176-185

[34] Rusu A, Colmenarejo S G, Gulcehre C, et al. Policy distillation [C] //Proc of the 4th Int Conf on Learning Representations(ICLR). New York: Springer, 2016: 343-352

[35] Yin Haiyan, Pan Jialin. Knowledge transfer for deep reinforcement learning with hierarchical experience replay [C] //Proc of the 31st AAAI Conf on Artificial Intelligence. Menlo Park, CA: AAAI, 2017: 1640-1646

[36] Liu Quan, Zhai Jianwei, Zhong Shan, et al. A Deep recurrent Q-network based on visual attention mechanism [J]. Chinese Journal of Computers, 2017, 40(6): 1353-1366 (in Chinese)
(刘全, 翟建伟, 钟珊, 等. 一种基于视觉注意力机制的深度循环Q网络模型[J]. 计算机学报, 2017, 40(6): 1353-1366)

[37] Hou Yuenan, Liu Lifeng, Wei Qing, et al. A novel DDPG method with prioritized experience replay [C] //Proc of the 13th IEEE Int Conf on Systems, Man, and Cybernetics (SMC). Piscataway, NJ: IEEE, 2017: 316-321

[38] Bruin T, Kober J, Tuyls K, et al. The importance of experience replay database composition in deep reinforcement learning [EB/OL]. 2015 [2017-11-04]. http://www.jenskober.de/deBruinNIPS_WS2015.pdf

[39] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms [C] //Proc of the 31st Int Conf on Machine Learning(ICML). New York: JMLR, 2014: 387-395

[40] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning [C] //Proc of the 4th Int Conf on Learning Representations (ICLR). New York: Springer, 2016: 232-241

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

[42] Mahmood A R, Hasselt H V, Sutton R S. Weighted importance sampling for off-policy learning with linear function approximation [C] //Proc of the 24th Int Conf on Neural Information Processing Systems(NIPS). Cambridge, MA: MIT Press, 2014: 3014-3022

[43] Bellmare M G, Naddaf Y, Veness J, et al. The arcade learning environment: An evaluation platform for general agents [J]. Journal of Artificial Intelligence Research, 2013, 47(1): 253-279

[44] Brockman G, Cheung V, Pettersson L, et al. OpenAI gym [EB/OL]. New York: Cornell University, 2016 [2017-10-04]. https://arxiv.org/abs/1606.01540

[45] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems [EB/OL]. New York: Cornell University, 2016 [2017-10-04]. https://arxiv.org/abs/1603.04467

Bai Chenjia, born in 1993. PhD candidate of Pattern Recognition and Intelligent System Research Center, Harbin Institute of Technology. Student member of CCF. His main research interests include reinforcement learning and neural network.

Liu Peng, born in 1975. Received his PhD degree of microelectronics and solid-state electronics from Harbin Institute of Technology in 2007. Associate professor at the School of Computer Science and Technology, Harbin Institute of Technology. His main research interest covers image processing, video analysis, pattern recognition, and design of very large scale integrated (VLSI) circuit.

Zhao Wei, born in 1972. Associate professor of School of the Computer Science and Technology, Harbin Institute of Technology. Her research fields include pattern recognition, machine learning and computer vision.

Tang Xianglong, born in 1960. Received his PhD degree of computer application technology from Harbin Institute of Technology in 1995. Professor at the School of Computer Science and Technology, Harbin Institute of Technology. His main research interest covers pattern recognition, image processing, and machine learning.