围棋人工智能AlphaGo系列算法的原理与方法

章胜[1],龙强[2*],孔轶男[3],王宇[2]

1. 中国空气动力研究与发展中心空天技术研究所,绵阳 621000
2. 西南科技大学数理学院,绵阳 621000
3. 中国空气动力研究与发展中心计算空气动力研究所,绵阳 621000

摘要　围棋人工智能AlphaGo系列算法是人工智能发展历史中的重要里程碑事件。它们不仅成功地求解了以围棋为代表的完全信息博弈问题,而且具有更加广泛的适用性。依算法的发展历程,从基本原理与技术特征方面对AlphaGo Fan到MuZero的一系列算法进行了梳理,说明了AlphaGo系列算法的落子原理,阐释与对比了其中采用的关键技术:蒙特卡洛树搜索和深度神经网络的建模及训练。AlphaGo系列算法对解决实践中的其他重要问题,从算法设计、神经网络建模到模型利用等方面都具有重要的参考意义,本文的总结有助于快速地掌握这些算法的基本原理,从而为相关算法的研究与拓展提供有益参考。

关键词　人工智能;AlphaGo系列算法;蒙特卡洛树搜索;深度神经网络;强化学习

“这种被自动化机器或是被一段看不见的算法挑战、超越甚至取代的感觉,正在成为我们社会的一个标准组成部分……每个职业都终将感受到这一压力,否则就意味着人类停止发展。”国际象棋世界冠军卡斯帕罗夫对人工智能(artificial intelligence)发展的这段评述深刻预见了人类社会可能的未来[1]。20世纪90年代,IBM的深蓝(Deep Blue)超级计算机就已经战胜了被誉为有史以来最伟大的国际象棋棋手卡斯帕罗夫,这是人工智能征服棋类游戏的重大事件[2]。相较于象棋,围棋具有几乎无穷的状态动作搜索空间,其棋盘位置及动作局面的评估难度更大,因此它一直被认为是人工智能经典游戏中最具挑战性的问题,被誉为“人类智慧的堡垒”[3]。近年随着深度学习的兴起,将深度学习的感知能力与强化学习的决策能力相结合,为复杂感知决策问题的求解开辟了新的思路,深度强化学习已成为通向人工智能的重要途径[4]。2015年以来,DeepMind公司发展了围棋人工智能AlphaGo及其各种演化版本,这些算法取得了一系列令人瞩目的成绩,它们将人们对人工智能、强化学习的研究推

收稿日期:2022-08-03;修回日期:2022-11-03

基金项目:国家自然科学基金项目(11902332)

作者简介:章胜,副研究员,研究方向为系统优化与人工智能,电子信箱:zszhangshengzs1@outlook.com;龙强(通信作者),副教授,研究方向为人工智能,电子信箱:Longqiang@163.com

引用格式:章胜, 龙强, 孔轶男, 等. 围棋人工智能AlphaGo系列算法的原理与方法[J]. 科技导报, 2023, 41(7): 79-97; doi: 10.3981/j.issn.1000-7857.2023.07.009

向了一个新的高潮,对人类的发展与进步有着深远影响。

按时间线,围棋人工智能AlphaGo系列算法的主要事件如表1所示。2015年AlphaGo击败樊辉是计算机围棋程序第一次在完整的比赛中毫无障碍地击败人类职业棋手,而此前人们认为这一壮举的实现至少还需要10年时间。该版本的AlphaGo称为AlphaGo Fan,它采用分布式训练架构,利用1202个CPU与176个GPU训练了大约1周[5]。2016年AlphaGo与围棋世界冠军李世石的世纪巅峰对决吸引了全球各界的目光,也自此在全球范围内将深度强化学习推向了研究热点。该版本程序称为AlphaGo Lee,它与AlphaGo Fan总体一致,但在神经网络的结构以及训练方面稍有不同。AlphaGo Lee也是一个分布式版本,它采用48个TPU进行了长达数月的训练[6]。DeepMind公司在2017年推出的AlphaGo Master具有更强的对弈能力,它在乌镇围棋峰会上以3:0击败柯洁后,AlphaGo研究团队宣布不再参加围棋比赛[7]。AlphaGo Master神经网络模型的输入与AlphaGo Lee相同,也考虑了人工处理的特征以及基于人类数据的监督学习,但它采用的神经网络架构与深度强化学习方法与AlphaGo Lee存在明显区别。AlphaGo Master没有采用分布式搜索,而是通过谷歌云中的1台配备有4个TPU的机器训练得到。

表1 AlphaGo系列算法里程碑事件

时间	版本	事件
2015年10月	AlphaGo Fan	5:0击败欧洲围棋锦标赛冠军樊辉[5]
2016年03月	AlphaGo Lee	4:1击败围棋世界冠军李世石[8]
2017年01月	AlphaGo Master	在网络围棋平台上以60:0击败中日韩顶级棋手[9]
2017年05月	AlphaGo Master	3:0击败中国围棋冠军柯洁[7]
2017年10月	AlphaGo Zero	击败AlphaGo Lee、AlphaGo Master[6]
2017年12月	AlphaZero	史上最强大的通用棋类人工智能,在围棋、国际象棋与日本将棋都达到顶尖水平[10]
2019年11月	MuZero	自学环境模型,性能可以匹敌具有精确模拟器的AlphaZero,并在Atari游戏上达到SOTA水平[11]

继AlphaGo Master之后,DeepMind公司在2017年又推出了AlphaGo Zero,它不需要人类示例或指导,通过强大的自学能力,在36 h后就超越了AlphaGo Lee,72 h后,在与首尔人-机比赛相同的条件下,AlphaGo Zero以100:0的绝对优势击败了AlphaGo Lee[6]。AlphaGo Zero采用与AlphaGo Master相似的神经网络架构与强化学习算法,利用谷歌云中的1台配备有4个TPU的机器训练了3 d时间。在AlphaGo Zero的另一个采用更大神经网络结构与更长训练时间(近40 d)的版本中,它以89:11击败了AlphaGo Master。

AlphaGo Zero的自学能力是否具有推广到其他棋类游戏的通用性是一个有趣的问题,2018年,DeepMind推出了通用棋类人工智能AlphaZero[10]。AlphaZero从随机对弈开始训练,在没有先验知识、只知道基本规则的情况下,成为史上最强大的棋类人工智能。在国际象棋中,AlphaZero训练4 h就超越了世界冠军程序Stockfish[12];在日本将棋中,AlphaZero训练2 h就超越了世界冠军程序Elmo[13];在围棋中,AlphaZero仅训练30 h就超越了AlphaGo Lee,大约150 h后其棋力就达到了AlphaGo Zero水平。AlphaZero凸显了深度强化学习在解决控制决策问题中的广泛适用性,正如《Science》杂志的评价称:“能够解决多个复杂问题的单一算法,是创建通用机器学习系统,解决实际问题的重要一步。”

AlphaGo Fan至AlphaZero等算法有一个共同特点,就是这些算法中有一个准确的环境模型,即模拟器(simulator),但是许多实际问题中,环境动力学通常是复杂且未知的。2020年,DeepMind在《Nature》发表了无需环境模拟器的MuZero算法,在

没有环境动力学知识的情况下,MuZero在围棋、国际象棋和日本将棋等精确规划任务中表现不俗,其性能可以匹敌具有精确模拟器的AlphaZero[11]。并且,MuZero还将AlphaZero扩展到包括单智能体域和中间时间步非零奖励的广泛场景,如Atari游戏,这表明AlphaGo算法不仅能解决完全信息博弈问题,它还具有广泛的适用性,Deepmind宣称MuZero向着通用算法又迈进了一大步,图1描述了AlphaGo系列算法性能与通用性的发展。

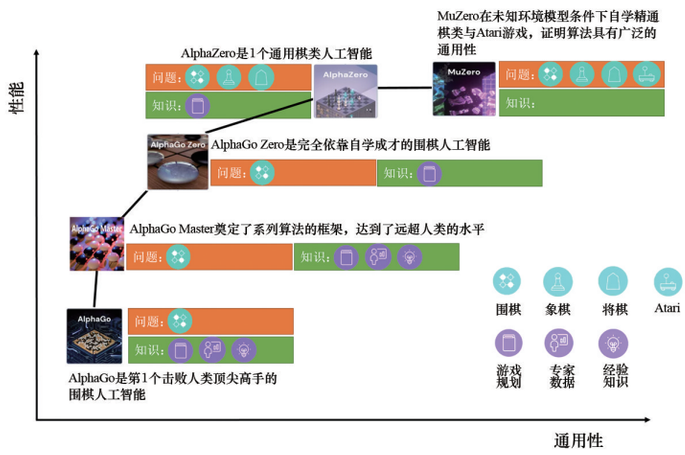

图1　AlphaGo系列算法的发展

AlphaGo系列算法是人工智能发展历史中的重要里程碑事件,它们对解决实践中的重要问题,从算法设计、神经网络建模到模型利用等方面都具有重要参考意义[14-20]。最近,Deepmind公司基于AlphaZero算法,发展了矩阵乘算法自动开发的人工智能AlphaTensor[21],在诸多维度矩阵上的计算结果达到或超过了人类目前最好算法的水平,实际上它具有进一步推广到其他数值算法任务开发的潜力,这一科学实践再次说明了AlphaGo系列算法的强大性能。虽然AlphaGo系列算法已经提出一段时间,不少文献[8, 22-28]与网络资料[29-30]对相关算法进行了很好的综述介绍,但是这些文献只涵盖了AlphaGo、AlphaGo Zero和AlphaZero算法,缺乏对MuZero的介绍。文献[31]、[32]对从AlphaGo到MuZero的系列算法进行了全面的介绍,详细说明了算法实现细节,但在神经网络训练原理的提炼方面略有不足,对不同算法的原理及建模对比有所欠缺。按AlphaGo系列算法发展的顺序,本研究从围棋程序落子运行与其深度强化学习原理的角度,分共性技术与个性技术对AlphaGo系列算法加以评述。本研究首先介绍通用技术蒙特卡洛树搜索(Monte Carlo tree search,MCTS),然后对比不同算法中采用的深度神经网络以及具体的强化学习算法。

1　AlphaGo系列算法落子原理

围棋是典型的完全信息博弈问题,Minimax算法、Alpha-Beta剪枝算法与MCTS是求解完全信息博弈问题的3种主要算法[33]。策梅洛定理(Zermelo's theorem)指出:任何一种完全信息博弈的棋类游戏,在当前状态下都存在一个最优策略以及与之对应的最优状态价值函数[34]。理论上,该最优价值函数可以通过经典Minimax算法递推迭代求得,或利用无损的Alpha-Beta剪枝算法提高计算效率[35]。这种解决思路在象棋领域取得了巨大的成功,深蓝、Stockfish都是基于Alpha-Beta剪枝算法进行开发的。但是,基于Minimax原理的算法的计算量与棋类的深度及宽度有关,对于围棋,由于其巨大的状态空间(超过了宇宙中所有粒子的总数目),这个计算量是难以想象的[36]。因此,该类型算法在围棋上的应用效果并不理想[37]。采用蒙特卡洛方法进行随机模拟是人工智能中广泛采用的一种手段,蒙特卡洛模拟方法基本原理简单,简单的讲,它就是通过大量"暴力计算"去求解最佳的动作策略[38]。在蒙特卡洛方法的基础上,Coulom[39]在2006年提出了MCTS方法,它是一种采用确定规则驱动的启发式随机搜索算法,由于效果良好,自提出以来,许多

成功的棋类算法都采用了MCTS技术[40]。

AlphaGo系列算法也是基于MCTS方法进行落子,不过,AlphaGo系列算法中MCTS结合了深度卷积神经网络(deep convolutional neural network)[41]来提高模拟效率,减少模拟时的搜索深度与宽度,具体的,它使用价值网络(value network)评估棋盘局面的胜率(AlphaGo系列算法中采用的胜率与传统的[0,1]区间表示的胜率略有区别,其区间为[−1,1],1表示赢,−1表示输)来截断模拟深度,采用策略网络(policy network)选择走法以减少搜索宽度。图2说明了AlphaGo系列算法的落子原理,在t时刻局面(对应状态s_t),AlphaGo算法执行MCTS构建搜索树,返回包括动作访问次数等信息在内的统计结果。

根据MCTS结果,AlphaGo系列算法根据访问次数信息输出动作a_t(称为play)。其中,AlphaGo Fan与AlphaGo Lee选择访问次数最多的落子动作,即

$$a_t = \arg\max_b \left(N\left(s_t, b\right)\right) \tag{1}$$

式中,$N\left(s_t, b\right)$为搜索树中在状态s_t执行动作b的次数。

与AlphaGo Fan等略有不同,AlphaGo Master、AlphaGo Zero、AlphaZero、MuZero根据不同动作的访问次数,利用Softmax函数得到当前状态s_t下不同动作对应的概率π,并依据该概率选择落子,具体为

$$\pi\left(a\middle|s_t\right) = \frac{N\left(s_t, a\right)^{1/\tau}}{\sum_b N\left(s_t, b\right)^{1/\tau}} \tag{2}$$

式中,τ为温度参数,它可以调节具体概率分布,τ越小,$\pi\left(a\middle|s_t\right)$越趋近于选择最大访问次数动作的确定性策略(即式(1))。

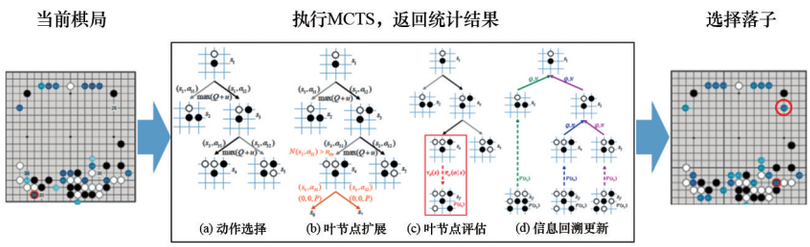

图2 AlphaGo系列算法落子原理

不同于许多强化学习决策机的使用,AlphaGo系列算法并不是直接基于训练得到的策略网络来进行落子,虽然它也可以这样做,但是这种做法的对弈能力并不够强(文献[5]中图4(b)与文献[6]中图6给出了测试结果),这个现象也说明了围棋问题的复杂性:即使经过大量训练得到的策略网络仍远不是最优网络。

在训练得到的策略与价值网络基础上,MCTS给出的动作策略比策略网络输出的动作策略更强。因此,MCTS可以视为一个有效的策略改善(policy improvement)算子,这也是MCTS自提出后就得到广泛应用的重要原因。AlphaGo系列算法的核心要素包括MCTS方法和深度神经网络的建模及训练,下面进行介绍。

2 MCTS方法

MCTS由确定规则驱动进行启发式随机搜索,其实质上也是一种强化学习方法[42],它通过与环境进行实际或模拟交互得到状态、动作和奖励的经验样本序列,利用这些经验数据支撑决策。MCTS方法从蒙特卡洛方法发展而来,但是两者存在一定区别,简单的讲,对于完全信息博弈问题,蒙特卡洛方法有偏的,而MCTS是无偏的。理论上随着模拟次数的增加,MCTS的结果可以收敛到最佳的动作策略与正确的动作价值[43-44]。

2.1 MCTS基本方法

MCTS即为建立一棵搜索树的过程[45]。树是一种数据结构,它由节点与边组成,搜索树中,每个节

点代表一个状态s,每条边代表一定的状态-动作对(s, a),如图3所示,树中最上方的节点称为根节点,一个节点的上一级节点称为父节点,下一级节点称为子节点,没有子节点的节点称为叶节点。搜索树中的每条边会记录一定的信息(由于$s \xrightarrow{a} s'$,也可以将信息记录在节点s'处,但此时需要记录相应动作以追溯其父节点[46]),包括总动作价值$W(s, a)$、访问次数$N(s, a)$与动作价值$Q(s, a)$,可记为$\{W/N\}$,动作价值计算为$Q(s,a)=\frac{W(s,a)}{N(s,a)}$,注意信息$\{W/N\}$一般均是在对弈双方中某一方的角度下进行记录的。

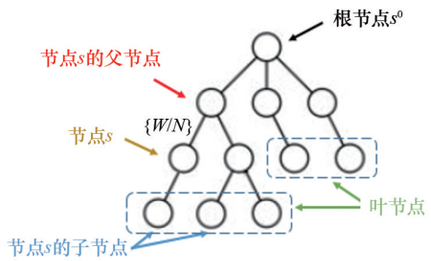

图3　MCTS树结构示意

MCTS基本方法的一次搜索一般可以分为4步[47],如图4所示,分别是动作选择(selection)、叶节点扩展(expansion)、叶节点评估(evaluation)和信息回溯更新(backup)。在一轮MCTS结束,玩家根据统计信息落子后,新的状态将成为新的根节点,其下子树(subtree)的数据可以保留,而之前搜索树的其他部分会被舍弃。

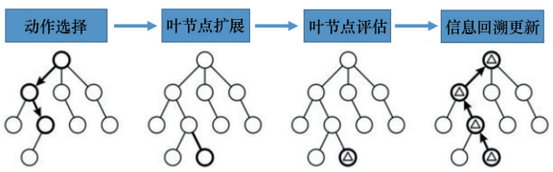

图4　MCTS的一次搜索过程

2.1.1　动作选择

MCTS中的动作选择策略称为树中策略(tree policy),一般分为两种情形,第1种情形指从根节点s^0出发,直到抵达某个尚未完全展开的节点s^{L-1}(经过$L-1$个虚拟时间步)。所谓尚未完全展开的节点,是指其子节点没有被全部访问过的节点。为平衡探索(exploration)与利用(exploitation),参考经典多臂赌博机问题中的UCB(upper confidence bound)最优算法[48],采用UCT(UCB in Tree)算法确定动作$a^k(k=0, 2, \cdots, L-2)$,为

$$a^k = \arg\max_a \left(Q(s^k, a) + c\sqrt{\frac{\sum_b N(s^k, b)}{N(s^k, a)}} \right) \tag{3}$$

式中,第1项代表利用,$Q(s, a)$的值越大,则相应动作越容易被选取;第2项代表探索,动作访问次数$N(s, a)$越小,则该动作越容易被选择到;c为调节探索与利用的正值权重参数。

第2种情形为抵达尚未完全展开的节点s^{L-1}时,对于s^{L-1}处没有访问过的动作进行随机选择来得到动作a^{L-1},通常这些未访问过动作的选择概率是相同的。

注意在博弈问题中,搜索树中信息通常是以某一方的角度进行记录的,当另一方利用式(3)进行动作选择的时候,需要对动作价值进行调整。比如围棋中动作价值函数1代表赢,-1代表输,如果黑方视角下的动作价值为0.8,那么白方的动作价值应为-0.8。

2.1.2　叶节点扩展

假设在状态s^{L-1}执行动作a^{L-1}得到新的状态节点s^L,可将最新得到的节点s^L扩展为搜索树上的一个叶节点。在新扩展的节点处,将它的每个可行动作的统计信息初始化为0,包括总动作价值$W(s^L, a)$、访问次数$N(s^L, a)$与动作价值$Q(s^L, a)$。注意并非搜索中遇到的新叶节点位置就须扩展到搜索树中,比如AlphaGo Fan中的MCTS是在某个新节点的访问次数超过一定数目后再进行扩展。

2.1.3　叶节点评估

叶节点评估步骤估计新叶节点位置s^L的状态价值$V(s^L)$,对于棋类游戏,状态价值也即胜率,其等于动作价值$Q(s^{L-1}, a^{L-1})$。状态价值$V(s^L)$通常采用模拟推演手段(称为playout或rollout)得到,从s^L开始,通过某种策略(如随机策略)进行模拟直到游戏结束,然后根据比赛的胜负情况,得到该点的状态价值估计。当然,若s^L即游戏结束状态,此时可

以直接根据游戏规则计算该状态的得分。模拟推演中采用的策略也称为默认策略(default policy)。

2.1.4 信息回溯更新

在完成对叶节点位置s^L胜率的估计后,利用此估值对从s^L(可能尚未扩展到搜索树中)到根节点s^0子树的所有边[即状态-动作对(s, a)]进行信息更新,包括动作价值与访问次数更新,1次推演模拟完成后的更新为

$$W(s,a) \leftarrow W(s,a) + V(s^L) \tag{4}$$

$$N(s,a) \leftarrow N(s,a) + 1 \tag{5}$$

$$Q(s,a) \leftarrow \frac{W(s,a)}{N(s,a)} \tag{6}$$

2.2 AlphaGo系列算法中的MCTS

AlphaGo系列算法中的MCTS结合了神经网络模型来构建搜索树,由于内嵌了围棋知识的深度神经网络可协助动作选择与价值估计,这也使其中的动作选择与叶节点评估相较于MCTS基本方法存在明显的改进:(1) 动作选择时利用策略网络来减少搜索宽度。在AlphaGo系列算法搜索树的每一条边中,除动作价值、访问次数信息外,还多存储了一个数据:先验动作概率$P(s, a)$,在新扩展节点初始化时候,通过策略网络对该节点处每一个动作的概率进行赋值,不同于基本MCTS中未执行动作的概率相同,利用策略网络给出动作先验概率可以增强对优势动作的利用。(2) 叶节点评估时候利用价值网络来减少模拟深度。对于新的叶节点位置,利用价值网络可以直接预测叶节点的状态价值。

2.2.1 AlphaGo Fan、AlphaGo Lee中的MCTS

AlphaGo Fan、AlphaGo Lee中的MCTS完全相同,这里一并介绍。动作a^k的选择策略对式(3)进行了一定改进[49],为

$$a^k = \arg\max_a \left(Q(s^k,a) + c_{\text{puct}} P(s^k,a) \frac{\sqrt{\sum_b N(s^k,b)}}{1 + N(s^k,a)} \right) \tag{7}$$

式中,$P(s^k, a)$为先验概率;c_{puct}为控制探索的参数。

采用式(7),最初算法会倾向于选择具有高先验概率和低访问次数的动作,随着节点访问次数的增多,它会逐渐倾向于具有高动作价值的动作。同时,由于式(7)第2项中的分母$1+N(s^k, a)$一定大于0,因此搜索时不必再区分当前节点是否为完全展开的情形。

AlphaGo Fan中的MCTS不是每一次搜索都会扩展叶节点,它是经过多次"动作选择—叶节点评估—信息回溯更新"循环后,才进行一次叶节点扩展。每次搜索中,假设状态s^L为新抵达的叶节点位置,此时对其胜率(介于-1~+1)的评估一方面延续基本MCTS的做法,基于一个快速Rollout策略网络(见第3.1节)进行自我博弈得到最终胜负结果(记为z_L),另一方面通过强化学习训练得到的RL价值网络预测胜率$v(s^L)$。由于围棋棋局具有对称性,而对称局面的胜率相同,因此在预测$v(s^L)$时,采用隐式考虑对称性的做法,在s^L对应的镜像与旋转共8种对称局面中,随机选择一种计算胜率$v(s^L)$。虽然对称性利用中只考虑了一种情形,但通过大量的模拟,也能达到平均的效果。得到$v(s^L)$与z_L后,通过混合参数$\lambda \in [0,1]$加权形成对s^L的胜率估计$V(s^L)$

$$V(s^L) = (1-\lambda) v(s^L) + \lambda z_L \tag{8}$$

文献[5]考察了λ取不同数值时的效果,研究表明当λ=0时,算法的性能已超越了许多现有的围棋程序,说明价值网络为模拟评估提供了可行的替代方案;而当λ=0.5时,算法性能最好,对弈能力最强。得到胜率$V(s^L)$后,利用式(4)~式(6),回溯更新该次搜索中每一条边上的统计信息。

AlphaGo Fan中状态s^L被扩展为搜索树中叶节点的条件是它被访问的次数大于一定的阈值n_{thr}。为确保棋局添加到策略队列的速率与GPU计算策略网络的速率相匹配,参数n_{thr}是动态调整的。初始化先验动作概率时,采用基于人类标签的监督学习得到的SL策略网络计算先验概率$P(s^L, a)$,注意此处并非采用基于强化学习的具有更强对弈能力的RL策略网络,原因是人类动作中包含更加丰富的走法,具有更大的探索性,有利于得到更强的策略。计算先验概率时同样采用隐式考虑对称性的做法,随机选择s^L的一种对称状态进行计算。

需要指出的是,因为多次推演模拟的缘故,AlphaGo Fan实际应用中MCTS每一条边上采用的总

动作价值与访问次数统计量更为复杂,其总动作价值分为$W_v(s,a)$与$W_r(s,a)$两种,分别代表价值网络估计对应的总动作价值与推演模拟对应的总动作价值,相应的访问次数分别为$N_v(s,a)$与$N_r(s,a)$,而式(7)中采用的是$N_r(s,a)$。为了在分布式计算中鼓励探索新的树搜索动作,算法会将推演模拟的统计数据$W_r(s,a)$与$N_r(s,a)$进行一定处理[50]。这些细节不影响对MCTS原理的理解,感兴趣的读者可以进一步阅读文献[5]。

2.2.2　AlphaGo Master中的MCTS

与AlphaGo Fan相比,AlphaGo Master中的MCTS更为简单:(1)不同于AlphaGo Fan中搜索树的叶节点为动态扩展,这里搜索中每一个新的叶节点位置s^L都将扩展到搜索树中;(2)由于采用的神经网络为一体结构,可以同时给出胜率估计与先验动作概率,其中将"叶节点扩展"与"叶节点估计"2个步骤整合为一步"叶节点扩展与估计"。

对扩展的叶节点s^L进行初始化时,基于联合策略-价值神经网络f(见第3.3节),计算s^L处的先验动作概率$P(s^L,a)$与胜率预测$v(s^L)$,其中同样采用隐式考虑对称性的做法,选取8种对称情形中的任一种进行计算。在对s^L的胜率进行估计时,仍然基于式(8),综合考虑神经网络预测结果$v(s^L)$与Rollout策略网络模拟结果z_L。

2.2.3　AlphaGo Zero中的MCTS

AlphaGo Zero与AlphaGo Master中的MCTS步骤完全相同[6],同样包括"动作选择、叶节点扩展与估计、信息回溯更新"3个步骤。但是与AlphaGo Master不同的是,对于叶节点s^L胜率的估计,AlphaGo Zero不再采用推演模拟,仅采用神经网络的胜率预测结果,即式(8)中的混合参数λ为0。在AlphaGo Fan算法中已经研究过仅采用价值网络估计胜率的做法,当时的研究表明价值网络可以较好地预测比赛胜率,AlphaGo Zero完全采用了这种做法,从而进一步提高了MCTS的模拟效率。

2.2.4　AlphaZero中的MCTS

AlphaZero中的MCTS与AlphaGo Zero相同,但动作选择函数与式(7)略有区别,为

$$a^k = \arg\max_a \left(Q(s^k,a) + C(s^k)P(s^k,a)\frac{\sqrt{N(s^k)}}{1+N(s^k,a)} \right) \tag{9}$$

式中,$N(s)=\sum_b N(s,b)$为父节点的访问次数;$C(s)=\lg\frac{1+N(s)+c_{\text{base}}}{c_{\text{base}}}+c_{\text{init}}$为探索率,它会随着探索时间的增长缓慢增大;$c_{\text{base}}$、$c_{\text{init}}$为相应参数。

另一方面,因为对称特性对国际象棋与日本将棋不再成立,所以AlphaZero中不再利用棋局的对称特性,它仅从当前棋手视角下的棋局出发,对状态对应的动作概率与胜率进行估计。特别的,针对围棋、国际象棋和日本将棋,AlphaZero采用完全相同的MCTS以及超参数。

2.2.5　MuZero中的MCTS

MuZero的MCTS算法步骤与AlphaZero类似,但是由于不再具有环境模拟器,它利用神经网络模型构建搜索树,并生成每个节点(用一个内部隐藏状态s表示)的奖励、策略和状态价值的估计信息[51-52]。对于隐藏状态s,边(s,a)记录的统计信息为$\{Q(s,a),N(s,a),P(s,a),R(s,a),S(s,a)\}$,分别代表动作价值、访问次数、先验动作概率、奖励信息与状态转换信息,其中$R(s,a)$与$S(s,a)$记录神经网络环境模型展开中的动力学信息,用以方便搜索树的递推。此外,由于总动作价值$W(s,a)$与动作价值$Q(s,a)$、访问次数$N(s,a)$并非完全独立,此处不再设置$W(s,a)$。

如图5所示,在时刻点t,根据过去的m个观测o_{t-m+1}, o_{t-m+2}, …, o_t(对于围棋,m=8),MuZero通过一个表示网络h(见3.6节)得到根节点s^0,从s^0开始,搜索树中的每个节点都由一个神经网络模型提供的内部隐藏状态s^k表示(上标k指示搜索虚拟时间步),基于统计信息,通过最大化式(9)选择动作a^k,环境模型将过渡到新的状态s^{k+1},同时生成奖励r^{k+1},直到L步后到达一个新的叶节点s^L时结束本次搜索。不同于AlphaZero等算法在搜索时可以通过模拟器获得所有的可行动作,MuZero在搜索树内可能给出一个实际棋局中不允许的动作,但由于强化

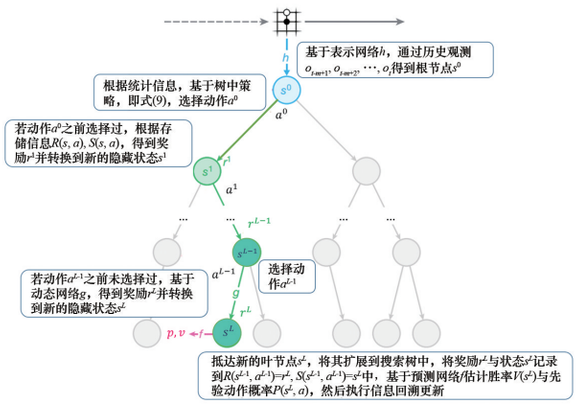

图5 MuZero中搜索树构建原理示意

学习中神经网络模型会迅速习得避免给出训练轨迹中从未执行过动作的能力,这种现象随着神经网络的训练会逐渐得以消除。除棋类游戏外,MuZero还可以应用于环境存在中间奖励、价值估计无界的问题(如Atari游戏),由于此时动作价值Q可能很大,MuZero在式(9)中采用的动作价值进行了归一化处理。

对于新扩展的叶节点s^L,首先将该状态s^L及奖励r^L记录到其父节点边的统计信息中,即$R(s^{L-1}, a^{L-1})=r^L$、$S(s^{L-1}, a^{L-1})=s^L$,然后将s^L每个边(s^L, a)的统计信息进行初始化,其中$Q(s^L, a)=0$,$N(s^L, a)=0$,先验动作概率$P(s^L, a)$与胜率估计$V(s^L)$通过神经网络模型计算同时得到。对于如AlphaZero等包含精确模拟器的算法,若叶节点为棋局终端状态,此时胜率估计可直接基于胜负情况给出。而因为MuZero采用的隐藏状态并不等价于实际状态,其无法判断终端状态,因此它始终采用神经网络估计胜率。由于无法判断比赛结束,搜索可能会经过相应于模拟器终端状态的隐藏状态并继续进行。对于这种情况,MuZero通过在训练时将终端状态视为吸收态,使得神经网络总是给出相同的胜率估计,从而有效地解决了这个问题。

搜索结束后,基于叶节点s^L的状态价值估计更新本次搜索轨迹的统计信息。对于棋类游戏,不考虑中间状态的奖励,仅利用叶节点的胜率估计$V(s^L)$进行回溯更新,动作价值更新为

$$Q(s,a) \leftarrow \frac{N(s,a)\cdot Q(s,a)+V(s^L)}{N(s,a)+1} \tag{10}$$

访问次数更新同式(5)。对于Atari游戏,从搜索树叶节点开始回溯,对于第$k=L, L-1, \cdots, 1$步,通过价值估计$V(s^L)$以及模型给出的奖励信息计算从叶节点到根节点之间每个节点的累积折扣回报

$$G^k=\sum_{\tau=0}^{L-k}\gamma^\tau r_{k+\tau}+\gamma^{L-k+1}V(s^L) \tag{11}$$

式中,γ为折扣率。

然后利用该回报估计对相应边上的动作价值进行更新

$$Q(s^{k-1},a^{k-1}) \leftarrow \frac{N(s^{k-1},a^{k-1})\cdot Q(s^{k-1},a^{k-1})+G^k}{N(s^{k-1},a^{k-1})+1} \tag{12}$$

访问次数更新同样为式(5)。

3 AlphaGo系列算法中深度神经网络的建模与训练

MCTS算法一经提出,便在许多围棋程序(如Zen[53]、Crazy Stone[54]、Pachi[55]、Fuego[56])上得到了应用,使程序的棋力有了巨大的提升。但是这些程序还主要依仗于搜索计算,并没有形成对棋局的感知能力,而正是通过与深度神经网络的结合,使AlphaGo完成了向具备棋局洞察力的飞跃。神经网络是模仿人类大脑的结构和功能而建立起来的一种信息处理系统,它具有并行处理、容错抗扰、可以逼近任意非线性函数的强大功能,是一种有效的建模工具。自从2006年Hinton提出深度置信网络以来[57],人们对神经网络的研究掀起了第3次高潮。相对于浅层神经网络,深度神经网络可在隐层之间逐层提炼特征,学习到更加复杂的特征,因而具有更好的性能[58]。实际上在AlphaGo算法之前,研究人员已经利用监督学习来训练深度神经网络进行围棋的落子预测与对弈[59-63],AlphaGo系列算法同样采用深度神经网络进行建模,但它们是在强化学习框架下进行训练的。

围棋是典型的零和博弈问题,其奖励具有典型的稀疏特性,对弈中间过程的奖励函数为零,比赛结束时候才能根据胜负给出奖励值,由于稀疏奖励的激励不足,这给围棋人工智能的强化学习提出了

不小的挑战。另一方面,围棋游戏涉及到对弈双方玩家,单步动作的奖励与对手的策略有关,因此其不能简单地通过单智能体强化学习方法来解决,理论上需要诉诸多智能体强化学习方案[64]。AlphaGo系列算法通过引入自我博弈(self-play)技术,使得单智能体强化学习也可以用于解决类似于围棋游戏这种双玩家博弈问题。与之前的其他研究工作相比,AlphaGo系列算法最大的技术特点就是采用自我博弈的深度强化学习来对网络模型进行训练。

3.1　AlphaGo Fan深度神经网络的建模与训练

3.1.1　神经网络建模

AlphaGo Fan中主要包含4个网络模型,包括3个深度卷积神经网络:1个监督学习策略网络(记为SL策略网络)$p(a|s;\sigma)$、1个强化学习策略网络(记为RL策略网络)$p(a|s;\rho)$、1个强化学习价值网络(记为RL价值网络)$v(s;\theta)$,此外还有1个基于模式特征的浅层推演(Rollout)策略网络$p(a|s;\pi)$,表2列出了这些神经网络模型的功能与结构,需要指出的是,虽然RL策略网络比SL策略网络更强(胜率超过80%)[5],但由于其在动作多样性方面不及SL策略网络,所以AlphaGo Fan在MCTS策略概率的估计中未采用RL策略网络。

3.1.2　神经网络训练

AlphaGo Fan中神经网络的训练包括SL策略网络与Rollout策略网络的监督学习、RL价值网络与RL策略网络的强化学习。

表2　AlphaGo Fan中神经网络模型的功能与结构

网络	功能	结构
SL策略网络	1)初始化RL策略网络; 2)给出MCTS叶节点的动作概率; 3)在对RL价值网络训练时,为增加自我博弈中数据的多样性,用于前若干步的动作决策	输入:48个19×19的棋局矩阵,包括我方落子位置、敌方落子位置以及“气”、“打吃”、“征子”等人工特征信息 输出:大小为19×19+1=362维的向量,代表动作概率,其中19×19指棋盘落子位置,1指不落子 主体结构:采用卷积神经网络,隐层数12,每层包含卷积层与Relu激活函数,卷积层包含192个滤波器,输出层为Softmax卷积层
Rollout策略网络	用于MCTS中叶节点至终止状态的快速推演模拟	输入:大小为109747的向量,描述“响应”模式、“非响应”模式以及其他少量人工特征信息 输出:同SL策略网络 主体结构:Softmax线性层
RL策略网络	自我博弈生成数据,用于训练RL价值网络	与SL策略网络相同
RL价值网络	用于MCTS中估计叶节点的胜率	输入:49个19×19的棋局矩阵,前48个同SL策略网络外,后1个包含当前玩家信息 输出:1个标量,代表对棋局胜率的评估 主体结构:采用卷积神经网络,隐层数14,其中2~11隐层与SL策略网络相同,12~13隐层是2个额外卷积层,14隐层是全连接Relu线性层,输出层为Tanh线性层

1)SL策略网络与Rollout策略网络的监督学习。

基于人类专家围棋对弈的标签数据,将人类输出动作的概率置为1,其他动作对应概率取为0,使SL策略网络$p(a|s;\sigma)$与Rollout策略网络$p(a|s;\pi)$的训练构造为一个多标签分类问题。利用KGS服务器上的约3000万盘棋局[65],并通过棋局的镜面与旋转对称特性对数据进行增强,通过最小化交叉熵损失函数(为统一,文中均为最小化损失函数),在随机采样的状态-动作对(s, a)上采用批梯度算法进行训练,SL策略网络$p(a|s;\sigma)$训练的交叉熵损失函数为

$$L(\sigma)=-\frac{1}{m}\sum_{k=1}^{m}\lg p\left(a_k|s_k;\sigma\right) \tag{13}$$

式中,m=16为小批量(mini-batch)大小。

SL策略网络与Rollout策略网络中,SL策略网络的预测精度较高(准确率接近60%),但计算较慢,每次大约需3 ms,Rollout策略网络的预测精度较低(准确率约25%),但计算速度快,单次计算仅需2 μs。

2)RL价值网络与RL策略网络的强化学习。

对于RL价值网络$v(s;\theta)$与RL策略网络$p(a|s;\rho)$,AlphaGo Fan采用REINFORCE策略梯度强化学习算法对它们进行训练[66],但在具体实现上与标准REINFORCE算法存在一定区别。图6给出了深度神经网络强化学习的流程框图,从图6中可以看到,RL策略网络的训练与RL价值网络的训练是2个相对独立的过程,但是其中又存在交互。

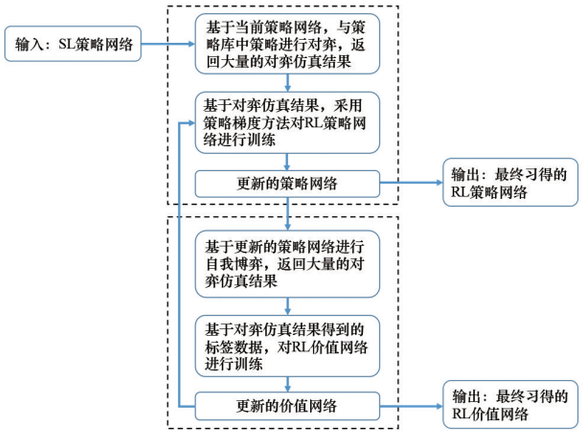

图6 AlphaGo Fan深度神经网络强化学习训练流程

强化学习中,RL策略网络$p(a|s;\rho)$参数首先通过SL策略网络$p(a|s;\sigma)$进行初始化。为了增强策略网络训练的稳定性,避免对策略网络的局部过拟合,采用RL策略网络与策略库中随机选择的策略进行对弈生成数据。策略库最初策略即SL策略,每500次训练后新增1个RL策略。围棋对弈中间过程状态s_t的奖励$r(s_t)$为0,比赛结束时刻状态s_T的奖励$r(s_T)$根据胜负结果确定(胜:+1,负:−1),依照玩家的胜负情况,确定中间状态s_t的回报为z_t=$r(s_T)$或z_t=−$r(s_T)$。基于REINFORCE策略梯度算法,构造对数似然损失函数为

$$L(\rho)=-\frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{T^i}\lg p\left(a_t^i|s_t^i;\rho\right)\left(z_t^i-v\left(s_t^i\right)\right) \quad (14)$$

式中,α为学习率,n=128为小批量大小,$v\left(s_t^i\right)$为价值网络输出,作为基准计算优势函数。注意在首次迭代时,基准并非采用价值网络输出,而是直接取为0。

对于RL价值网络$v(s;\theta)$,其训练目的是准确预测在状态s采用策略$p(a|s;\rho)$的胜率,即$\mathrm{E}\left[z_t|s_t=s,a_{t,t+1,\cdots,T}\sim p(a|s;\rho)\right]$。由于同一棋局前后的状态强相关,为了避免训练中非独立数据引起的局部过拟合问题,不同于传统REINFORCE方法利用与策略网络训练同源的交互数据对价值网络进行更新[66],AlphaGo Fan专门针对策略$p(a|s;\rho)$,通过自我博弈生成数据进行训练。为此,开展了3000万次对弈,每次对弈只抽取1个样本,从而打破数据之间的相关性。采用批训练方法,构造RL价值网络的损失函数为

$$L(\theta)=\frac{1}{m}\sum_{k=1}^{m}\left(z_k-v\left(s_k;\theta\right)\right)^2 \quad (15)$$

式中,小批量大小m取为32。

对于训练得到的RL价值网络,计算表明其胜率估计结果与采用RL策略网络$p(a|s;\rho)$进行蒙特卡洛推演(Monte Carlo rollouts)的结果接近,但计算量仅为后者的1/15000,这极大地提高了决策效率。综上,AlphaGo Fan深度神经网络强化学习伪代码如算法1所示。

算法1 AlphaGo Fan中RL策略网络与RL价值网络学习算法

AlphaGo Fan中RL策略网络与RL价值网络学习算法

1. 输入:围棋模拟器,学习率α,经验池容量pool_size,小批量大小minibatch_size等参数。

2. 初始化:利用SL策略网络参数初始化RL策略网络参数,随机初始化RL价值网络参数,经验回放池D=∅。

3. 循环:

(1)基于带基线的REINFORCE策略梯度算法,利用批随机梯度方法训练RL策略网络;

(2)基于SL策略网络与新训的RL策略网络,通过自我博弈,生成独立的对弈数据置于经验回放池D;

(3)利用经验回放池D中数据,通过批随机梯度方法训练RL价值网络,返回第(1)步。

4. 输出:最终的RL策略网络与RL价值网络。

3.2　AlphaGo Lee深度神经网络的建模与训练

AlphaGo Lee中的神经网络模型与AlphaGo Fan基本相同,依然如表2所示,但与AlphaGo Fan相比,其中采用的卷积神经网络的结构更为庞大,使用了包含256个滤波器的12个卷积层。

AlphaGo Lee中SL策略网络$p(a|s;\sigma)$、Rollout策略网络$p(a|s;\pi)$与RL策略网络$p(a|s;\rho)$的训练与AlphaGo Fan相同,但是RL价值网络$v(s;\theta)$的训练存在区别,如图7所示,它是在算法1训练RL价值网络的基础上,进一步利用AlphaGo Lee自身自我博弈的数据来提升价值网络的性能,即在产生训练数据的模拟中结合了MCTS技术。

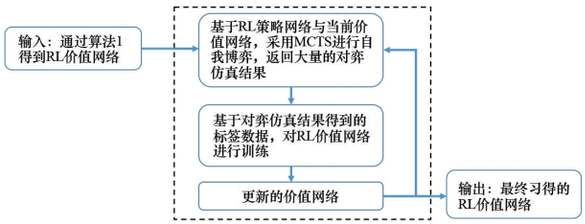

图7　AlphaGo Lee价值网络训练流程

AlphaGo Lee价值网络强化学习伪代码整理如算法2所示,注意其中不再涉及RL策略网络。

3.3　AlphaGo Master深度神经网络的建模与训练

与AlphaGo Fan和AlphaGo Lee不同,为了更好地共享状态特征,同时减小网络模型的计算量,AlphaGo Master中只包含1个神经网络f,称为联合策略-价值网络,该网络具有2个输出端,分别是策略输出端(policy head)与价值输出端(value head),策略输出端给出当前状态下的动作概率,价值输出端预测当前状态的胜率。AlphaGo Master中之所以将策略网络与价值网络共用同一神经网络主体,是因为研究表明相对于策略网络与价值网络独立表示的方式,一体结构的神经网络综合效果更好,具有更强的对弈能力[6]。网络结构方面,参考ImageNet冠军ResNet,AlphaGo Master采用了更为先进的残差卷积神经网络[67]。输入方面,同样考虑人工处理的特征。在策略输出端,模型不含Softmax函数,直接给出Logit值,但是后续计算也进行了Softmax处理,这样可以更方便地屏蔽不可行动作输出的影响(不可行动作的输出概率直接置0)。AlphaGo Master中神经网络模型的功能与结构详见表3。

算法2　AlphaGo Lee中RL价值网络学习算法

AlphaGo Lee中RL价值网络学习算法

1. 输入:围棋模拟器,学习率α,经验池容量pool_size,小批量大小minibatch_size等参数。

2. 初始化:利用算法1得到RL价值网络参数,经验回放池$D=\varnothing$。

3. 循环:

(1) 基于SL策略网络与当前的RL价值网络,结合MCTS进行自我博弈,生成独立的对弈数据存于经验回放池D;

(2) 利用经验回放池D中数据,通过批随机梯度方法训练RL价值网络,返回第(1)步。

4. 输出:最终的RL价值网络。

表3　AlphaGo Master中神经网络模型的功能与结构

网络	功能	结构	
		输入及网络主体结构	输出及输出端结构
联合策略-价值网络	策略输出端:用于MCTS中生成先验动作概率	输入:同表2中RL价值网络的输入 主体结构:采用残差卷积神经网络,隐层数20,包括1个卷积层与19个残差模块层。卷积层为整流批标准化卷积层,采用256个尺寸3×3、步长为1的滤波器。每个残差模块均由2个带有跳跃连接的整流批标准化卷积层构成,每个卷积层采用256个尺寸3×3、步长为1的滤波器	输出:大小为19×19+1=362维的向量,代表362个动作的Logit值,其中19×19指棋盘落子位置,1代表不落子 输出端结构:包括1个整流批标准化卷积隐层和1个全连接线性输出层
	价值输出端:用于MCTS中估计叶节点的胜率		输出:1个标量,代表对棋局胜率的评估 输出端结构:包含2个隐层和1个输出层,2个隐层分别为1个整流批标准化卷积层和1个全连接线性层,输出层为Tanh线性层

AlphaGo Master采用了一种与AlphaGo Fan明显不同的新颖强化学习技术,图8给出了神经网络训练的流程框图,首先基于人类数据采用监督学习对神经网络f的参数θ进行初始化,然后通过自我博弈强化学习算法进行训练。训练主要包括3步,分别是基于MCTS的自我博弈、利用标签数据的模仿学习(imitation learning)[68]与新训神经网络的考核环节。

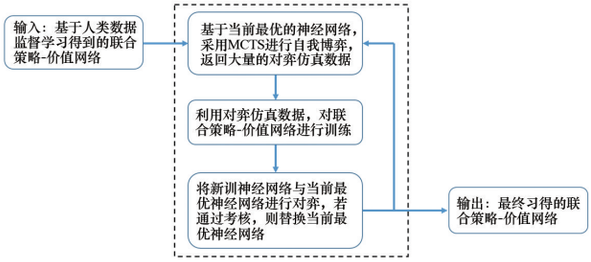

图8 AlphaGo Master深度神经网络训练流程

1)基于MCTS的自我博弈。

基于当前最优的联合策略-价值网络,采用MCTS方法进行自我博弈生成训练数据。在每个时间步t,从当前状态s_t出发开展MCTS,每轮进行1600次模拟(耗时约0.4 s),然后通过式(2)得到动作概率$\pi_t := \pi\left(a|s_t\right)$并选择落子,游戏结束后,根据结束时刻$T$的奖励$r_T \in \{-1, +1\}$回溯确定之前状态的回报$z_t=r_T$或$z_t=-r_T$,最终得到每个时间步$t$的数据元组$(s_t, \pi_t, z_t)$。自我博弈中,为了节省计算量,明显输掉的游戏将被放弃,判断准则是根节点胜率与子节点的最高胜率小于设定的投降阈值。为增加动作的多样性,一方面对式(2)中的温度参数τ进行调节,另一方面将Dirichlet噪声[69]添加到根节点的先验动作概率中实现额外的探索。

2)基于标签数据的模仿学习。

MCTS在每一轮探索后返回一个比策略网络输出更强的落子动作π,这是一个策略改进的过程,而在得到对弈结果后,对过程状态回报z进行确定,这又是一个策略估计的过程。通过MCTS的自我博弈产生的数据集(s_t, π_t, z_t),基于模仿学习开展深度神经网络的训练,使神经网络$f(s; \theta)$输出的动作概率p和胜率估计v趋近提升后的概率分布π和回报z,注意此时π、z相当于标签数据,分别代表期望的动作概率与胜率估计。在新近5×10^5次对弈生成的数据集(s_t, π_t, z_t)基础上,进一步利用镜面与旋转对称性对数据进行增强,然后通过小批量随机采样对神经网络进行训练。由于价值网络与策略网络共享同一个神经网络结构体,对二者统一进行训练,损失函数为

$$L(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left(z_i-v_i\right)^2-\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{\pi}_i^{\mathrm{T}}\lg\boldsymbol{p}_i+c\|\theta\|^2 \quad (16)$$

式中,$\left(z_i-v_i\right)^2$为胜率误差平方;$-\boldsymbol{\pi}_i^{\mathrm{T}}\lg\boldsymbol{p}_i$代表交叉熵;$c\|\theta\|^2$为消除神经网络过拟合的正则项;上标"T"表示转置符号;小批量大小取为m=2048;权重参数为$c=10^{-4}$。针对该损失函数,采用学习率退火的动量算法进行优化。

3)新训神经网络的考核。

一般训练更新后的网络将具有更强的性能,它们可以用于下一次自我博弈的迭代,以使MCTS的动作决策更加强大,从而进一步促进价值网络与策略网络性能的提升。但是由于强化学习并不能保证神经网络性能的单调改善,AlphaGo Master中引入考核机制,在对深度神经网络每训练1000次后,将新训的神经网络与当前最优的神经网络进行对弈,共400场比赛,通过考核(胜率大于55%)的神经网络才能替换当前最优神经网络,然后返回生成新的数据。

AlphaGo Master中深度神经网络训练的伪代码如算法3所示,注意如果价值网络与策略网络是独立建模,算法流程仍是可行的,不过此时应针对相应的损失函数对参数进行优化。在深度神经网络的训练中,AlphaGo Master充分利用了MCTS选择动作以搜集训练数据,而AlphaGo Fan在神经网络训练数据的生成环节中并没有用到MCTS。在价值函数的训练方面,AlphaGo Master与AlphaGo Lee有一定的相似性,但对策略网络的训练则明显不同:AlphaGo Master采用的是直接给定动作标签的模仿学习。

3.4 AlphaGo Zero深度神经网络的建模与训练

AlphaGo Zero中神经网络的结构与AlphaGo Master基本相同,但是输入更为简单,它仅利用黑

算法3　AlphaGo Master中联合策略-价值网络学习算法

AlphaGo Master中联合策略-价值网络学习算法

1. 输入:围棋模拟器,学习率α,经验池容量pool_size,小批量大小minibatch_size等参数。

2. 初始化:利用人类专家数据初始化联合策略-价值网络参数θ,初始化最优联合策略-价值网络参数$\theta'=\theta$,经验回放池$D=\varnothing$。

3. 循环:

(1) 基于最优联合策略-价值网络,通过自我博弈产生对弈数据,存入经验回放池D;

(2) 基于经验回放池D的采样数据,通过批训练梯度方法训练联合策略-价值网络;

(3) 对新训神经网络进行考核,确定是否更新最优神经网络参数为$\theta'=\theta$,返回第(1)步。

4. 输出:最终的最优联合策略-价值网络。

子与白子的位置作为输入,不再考虑"气"等人工特征。表4给出了AlphaGo Zero中神经网络模型的功能与结构,注意表中给出的是以100∶0击败AlphaGo Lee版本的结构,实际上AlphaGo Zero还有一个更强版本,其网络主体采用了40层隐层,包括1个卷积层与39个残差模块。

AlphaGo Zero采用的深度强化学习方法与AlphaGo Master完全一致,但是与AlphaGo Master不同的是,其神经网络参数的初值为随机生成,不再采用人类的知识进行初始化。图9给出了AlphaGo Zero中深度神经网络训练的流程框图,由于训练伪代码如前面算法3,仅在网络参数的初始化处有所不同,这里不再给出。

3.5　AlphaZero深度神经网络的建模与训练

与AlphaGo Zero相似,AlphaZero同样采用残差卷积结构的联合策略-价值网络,表5给出了AlphaZero应用到不同棋类时的神经网络结构,它们采用的神经网络主体结构相似,主要是输入输出存在区别。状态输入方面,不同棋类的状态描述以及对棋局形势的刻画存在区别,需要针对性地进行选取。动作输出方面,围棋的落子规则相对简单,而国际象棋和日本将棋的动作规则是不对称的,不同的棋子有不同的下法,例如国际象棋中兵棋通常只能前移一步,而后棋可以四面八方无限制地移动,同时棋子的移动规则还与位置相关,这些都需要在神经网络的输出中加以考虑。

表4　AlphaGo Zero中神经网络模型的功能与结构

网络	功能	结构	
		输入及网络主体结构	输出及输出端结构
联合策略-价值网络	策略输出端:用于MCTS中生成先验动作概率	输入:17个19×19的棋局矩阵,其中前8个矩阵描述当前及过去7个时刻点的我方落子位置棋局,后8个表示敌方落子位置棋局,最后1个矩阵表示当前玩家(全1为黑棋,全0为白棋)	输出及输出端结构:同表3
	价值输出端:用于MCTS中估计叶节点的胜率	主体结构:同表3中的主体结构	输出及输出端结构:同表3

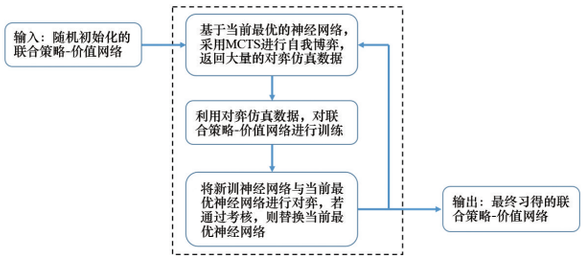

图9　AlphaGo Zero深度神经网络训练流程

AlphaZero中神经网络的训练原理与AlphaGo Zero基本一致,不同在于AlphaZero弃用了AlphaGo Zero中的考核环节,它直接利用新训模型进行自我博弈,实践表明该举措有助于提升训练效率,图10给出了AlphaZero神经网络训练的流程框图。

围棋的对弈结局只有输赢两种,而国际象棋和日本将棋还有平局这一结局,尤其对于国际象棋,平局被认为是最优结果[70]。因此,AlphaZero神经网络的训练中,需要考虑平局情形(用0表示),采

表5 AlphaZero应用到不同棋类游戏时的神经网络结构

神经网络结构	围棋	国际象棋	日本将棋
输入	19×19×17张平面,包含过去8个时刻点的棋局(大小为19×19),每副棋局分别采用我方落子位置与敌方落子位置共2张平面表示,最后一张平面表示当前玩家(全1为我方,全0为敌方)	8×8×119张平面,包含过去8个时刻点的棋局(大小为8×8),每副棋局分别采用敌我双方每种棋落子位置与位置重复情况共14张平面表示,此外还有7张平面用于表示当前玩家、移动数等信息	9×9×362张平面,包含过去8个时刻点的棋局(大小为9×9),每副棋局分别采用敌我双方每种棋落子位置、位置重复情况与俘虏数共45张平面表示,此外还有2张平面用于表示当前玩家和移动数
策略输出	大小为19×19+1=362维的向量,代表362个动作的Logit值,其中19×19指棋盘落子位置,1代表不落子	8×8×73张平面,代表4672个动作的Logit值,其中前56张图代表棋后的移动,后8张代表棋马的移动,最后9张代表"低升变"	9×9×139张平面,代表11259个动作的Logit值,其中前64张图代表棋后的移动,然后2张代表棋马的移动,然后64张代表升级棋后的移动,然后2张代表升级棋马的移动,最后7张代表"打入"
价值输出(相同)	1个标量,代表胜率		
主体结构(相同)	由1个整流批标准化卷积层和19个残差模块组成,每个残差模块都由2个带有跳跃连接的整流批标准化卷积层构成,每个卷积层采用256个尺寸为3×3、步长为1的滤波器		
策略输出端结构	包含1个整流批标准化卷积层,然后是包含362个输出端的线性层	包含1个整流批标准化卷积层,然后是包含73个滤波器的卷积层	包含1个整流批标准化卷积层,然后是包含139个滤波器的卷积层
价值输出端结构(相同)	包含1个整流批标准化卷积层,采用1个尺寸为1×1、步长为1的滤波器,然后是1个含256个单元的整流线性层,最后是1个大小为1的Tanh线性层		

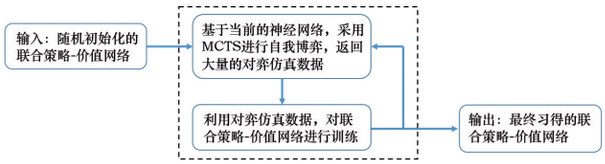

图10 AlphaZero深度神经网络训练流程

用包含相应标签的数据进行训练。由于一般棋类不再具有对称特性,AlphaZero神经网络训练中也不再利用对称性来增强训练数据。AlphaZero中神经网络训练算法如算法4所示,训练中,除学习率以及MCTS根节点先验概率中加入的探索噪声外,AlphaZero没有为不同棋类游戏做特别设置,它们采用完全相同的MCTS、神经网络主体结构以及超参数。

3.6 MuZero深度神经网络的建模与训练

基于模型的强化学习是一类重要的强化学习方法,但是这类方法的困难在于有的问题中环境复杂,其模型难以刻画与学习[71]。由于不再使用环境模拟器,MuZero中不仅将神经网络用于表示策略函数和价值函数,而且还用于动力学环境模型的学习。MuZero中动力学模型采用隐藏状态表示,作为内部变量,隐藏状态不需要与实际的环境状态相一致,它们的作用是预测与环境相关的策略、价值

算法4 AlphaZero中联合策略-价值网络学习算法

AlphaZero中联合策略-价值网络学习算法

1. 输入:棋类游戏模拟器,学习率α,经验池容量pool_size,小批量大小minibatch_size等参数。

2. 初始化:初始化联合策略-价值网络参数θ,经验回放池$D=\varnothing$。

3. 循环:

(1) 基于联合价值网络与策略网络,通过自我博弈产生对弈数据,存入经验回放池D;

(2) 基于经验回放池D的采样数据,通过批训练梯度方法训练联合价值网络与策略网络,返回第(1)步;

4. 输出:最终的联合策略-价值网络。

及奖励信息。MuZero中包括3个神经网络模型,分别是表示网络(representation network)h、动态网络(dynamics network)g与预测网络(prediction network)f,它们的功能与结构如表6所示,其中预测网络即联合策略-价值网络,它同时包含策略输出端与价值输出端。

表6　MuZero中神经网络模型的功能与结构

网络	功能	结构
表示网络	基于对环境的观测对初始状态进行初始化	输入(针对围棋):19×19×17的平面,前16个平面代表玩家双方最近的8个棋盘状态编码,最后1个平面表示当前玩家; 输出(针对围棋):19×19×256的平面,代表隐藏状态; 主体结构:采用残差卷积神经网络,其中具有16个残差模块,每个卷积层采用256个尺寸3×3的滤波器
动态网络	用作MCTS中的规划模型,得到更新的(隐藏)状态与奖励	输入(针对围棋):19×19×257的平面,前256个平面表示隐藏状态,后1个平面表示动作,动作平面中,落子的位置用1表示,其余为0,不落子则平面元素值均为0; 输出(针对围棋):19×19×256的平面和1个奖励标量; 主体结构:采用残差卷积神经网络,其中具有16个残差模块,每个卷积层采用256个尺寸3×3的滤波器
预测网络	1)预测网络的策略输出端用于MCTS中生成先验动作概率; 2)预测网络的价值输出端用于MCTS中估计叶节点的胜率。	输入(针对围棋):19×19×256的平面,代表隐藏状态; 输出(针对围棋):包括大小为19×19+1=362维的动作向量与1个胜率标量; 主体结构:与AlphaZero相同

图11给出了MuZero中神经网络训练的流程框图,从图中可以看到,MuZero中深度神经网络的训练与AlphaZero基本一致,其中没有考核环节,每训练1000步后最新得到的神经网络将用于基于MCTS的自我博弈以生成新的训练数据,但此处需训练的神经网络更多,包括表示网络、动态网络与预测网络3个网络。

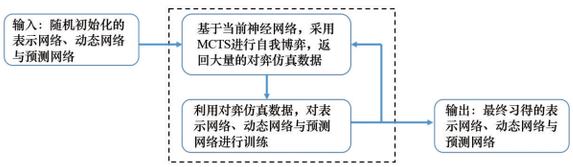

图11　MuZero深度神经网络训练流程

1)基于MCTS的自我博弈。

在每个时间步t,从观测信息o_{t-m+1}, o_{t-m+2}, …, o_t出发开展MCTS,每轮进行800次模拟,输出动作策略概率π_t(依式(2))和状态价值估计v_t(用于Atari游戏中回报的估计),基于返回的策略π_t选择动作$a_t \sim \pi_t$,执行动作a_t后从环境中得到一个真实奖励u_{t+1}(对于棋类游戏,中间状态的奖励值为0)和一个最新的观测o_{t+1}。依此方式,不断交换对弈玩家角色进行落子,直至比赛在时间步T结束。在一局对弈结束后,根据最终奖励$u_T \in \{+1, 0, -1\}$对本局游戏进行评分,进而得到每个时间步t的数据元组$(o_t, \pi_t, a_t, u_{t+1}, z_t)$,其中$z_t=u_T$或$z_t=-u_T$依据玩家最终的比赛结果确定。对于棋类游戏,为增强动作的多样性,MuZero采用与AlphaZero相同的探索方案产生动作。每局游戏结束后,产生的数据会被发送到内存的经验池用于训练,经验池中保留最新接收的100万次游戏数据。MuZero还可以应用于折扣回报问题(如Atari游戏),对于Atari游戏,由于其动作空间相对于棋类游戏少很多,每次MCTS只进行50次模拟,样本回报z_t通过取n=10步的奖励进行计算,为

$$z_t = u_{t+1} + \gamma u_{t+2} + \cdots + \gamma^{n-1} u_{t+n} + \gamma^n v_{t+n} \quad (17)$$

由于Atari游戏相对于棋类游戏长度更长(长达30分钟,有108000帧),因此游戏中每200步发送一次模拟数据到经验池,池中保留了最新的125000个

长度为200的数据序列用于训练。

2）基于标签数据的模仿学习。

由于MuZero中表示网络、动态网络与预测网络3个网络的关联性，对3个网络统一进行训练。不同于之前算法采用彼此独立的批数据进行训练（之前算法都要求尽可能地减少数据之间的相关性），MuZero是利用一段轨迹进行训练。训练时，从经验池中采样得到轨迹元组$(o_t, \boldsymbol{\pi}_t, a_t, u_{t+1}, z_t)$，利用包含$o_t$在内的历史观测信息与实际动作数据$a_{t+k}$生成对应的一段轨迹，如图12所示的模型展开，首先利用表示网络h接收所选轨迹m个历史观测$o_{t-m+1}, o_{t-m+2}, \cdots, o_t$作为输入得到初始状态$s_t^0$（下标代表数据真实时间，上标代表展开虚拟时间），模型随后将被循环展开K=5步，在每一步k=1, 2, …, K，动态网络g将前一步隐藏状态s_t^{k-1}和实际动作a_{t+k-1}作为输入，计算相应的奖励与转换状态，即$r_t^k, s_t^k = g\left(s_t^{k-1}, a_{t+k-1}\right)$，进一步再通过预测网络$f$得到不同状态对应的动作概率$p_t^k$与价值估计$v_t^k$。

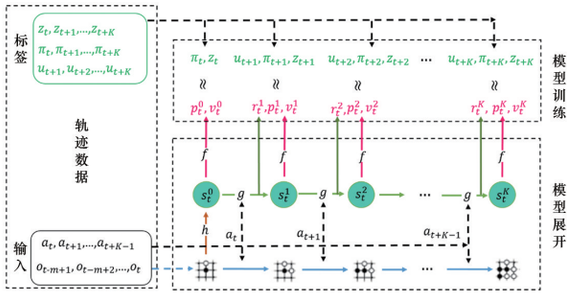

图12 MuZero神经网络模型展开及训练原理示意

如图12所示，模型训练的目的是使神经网络预测的3个量$\boldsymbol{p}_t^k$、v_t^k、r_t^k与自我博弈中得到的策略概率$\boldsymbol{\pi}_{t+k}$、回报z_{t+k}和奖励u_{t+k}相一致，定义损失函数为

$$l_t(\theta) = \sum_{k=0}^{K} l^p\left(\boldsymbol{\pi}_{t+k}, \boldsymbol{p}_t^k\right) + \sum_{k=0}^{K} l^v\left(z_{t+k}, v_t^k\right) + \sum_{k=1}^{K} l^r\left(u_{t+k}, r_t^k\right) + c\|\theta\|^2 \tag{18}$$

式中，l^p、l^v、l^r分别为策略损失函数、价值损失函数和奖励损失函数；c为正则化项的权重系数；θ包含表6中3个网络的参数。注意式（18）中策略与价值的训练考虑了k=0情形。对于棋类游戏，由于稀疏奖励的特点，取$l^r(u, r)$=0，l^p采用交叉熵函数，l^v采用平方误差函数，分别为

$$l^p\left(\boldsymbol{\pi}, \boldsymbol{p}\right) = -\boldsymbol{\pi}^{\mathrm{T}} \lg \boldsymbol{p} \tag{19}$$

$$l^v\left(z, v\right) = \left(z - v\right)^2 \tag{20}$$

对于Atari游戏，研究表明交叉熵损失函数比平方误差损失函数表现更稳定，故l^p、l^v、l^r均使用交叉熵损失函数。

MuZero中神经网络训练伪代码整理如算法5所示，类似于循环神经网络，表示网络、动态网络和预测网络参数需要沿时间反向传播进行端到端的联合训练。为了在训练中保持相似的梯度幅值，考虑梯度缩放，具体包括：每个输出端损失按1/K进行缩放，以确保梯度与展开步数K无关；将动态网络开始处的梯度缩放1/2，从而使得应用于动态网络的梯度保持恒定。

算法5 MuZero中表示网络、动态网络与预测网络学习算法

MuZero中表示网络、动态网络与预测网络学习算法

1. 输入：学习率α，经验池容量pool_size，小批量大小minibatch_size等参数。

2. 初始化：初始化表示网络参数、动态网络参数与预测网络参数（统一用θ表示），经验回放池D=∅。

3. 循环：

（1）基于当前的表示网络、动态网络与预测网络，通过自我博弈产生对弈数据，存入经验回放池D；

（2）基于经验回放池D的采样数据，通过批训练梯度方法对表示网络、动态网络与预测网络的参数θ进行联合优化，返回第（1）步。

4. 输出：最终的表示网络、动态网络与预测网络。

4 结论

人类从几千年来经历的数百万种围棋棋局中积累了大量的围棋知识，这些知识被提炼成模式、总结为谚语和书籍，其中蕴含了无穷的智慧。在这个号称“人类智慧堡垒”的领域，AlphaGo第1次实现了击败人类冠军的壮举，而AlphaGo Zero等算法更是在没有使用任何人类先验知识的条件下，从零开始，完全依靠强化学习进行训练，不仅重新发现了大部分围棋知识，而且习得了许多超出人类认知

的对弈技巧,达到了远超人类的水平。通过对它们对弈落子的研究来看,AlphaGo算法似乎具有了对棋局的洞察力,它们并不是简单大量计算后的推断,而是在历尽大量训练之后,其中的深度神经网络模型形成了对当前局面的理解与评估,这是AlphaGo算法的可怕之处,也是其令人着谜的地方。

AlphaGo系列算法的发展,是在实践中不断改进、提升和完善的过程,其中后期算法相对于前期算法总体上更为精简,有的之前采用的、认为有助于训练的手段后来又放弃了(比如考核机制),这凸显了实践对技术研究工作的重要指导作用。另一方面,AlphaGo系列算法并不是一蹴而就的,它们的提出也是植根于大量的基础研究工作,比如蒙特卡洛模拟、价值函数逼近等技术,这些方法早在之前就已有广泛的研究,AlphaGo系列算法结合深度神经网络取得了突破性的进展,生动说明了建立坚实研究基础,同时关注前沿技术发展对科技创新的重要性。

AlphaGo系列算法体现了强化学习的强大,尤其是"蒙特卡洛树搜索+深度神经网络"方法"完美"地求解了以围棋为代表的完全信息博弈问题,此外它在Atari游戏上的惊人表现[11]、在多智能体非完全信息博弈问题中的有益尝试[19]、在连续动作空间的积极拓展[20]、在算法自动开发中的成功探索[21]说明这套方法具有更加广泛的通用性,给其他类似问题的求解提供了良好参考。但是同时也要看到,现阶段深度强化学习对算力、数据要求巨大,得到一个好的神经网络模型通常代价不菲,目前开展的基于有限数据的训练研究(如EfficientZero[72])为智能体开发的改进提供了一个可行的方向。人工智能可以极大地解放生产力与发展生产力,而AlphaGo系列算法开启了通向人工智能的一道大门,但是前方还有许多挑战等待人类的攻克,正如AphaGo之父杰米斯·哈萨比斯所说:"我们才刚刚迈上了这个阶梯的第一步。"

参考文献(References)

[1] 加里·卡斯帕罗夫. 深度思考——人工智能的终点与人类创造力的起点[M]. 集智俱乐部, 译. 北京: 中国人民大学出版社, 2018.

[2] Campbell M, Hoane A J Jr, Hsu F H. Deep blue[J]. Artificial Intelligence, 2002, 134(1/2): 57-83.

[3] Müller M. Computer go[J]. Artificial Intelligence, 2002, 134(1/2): 145-179.

[4] Silver D. Tutorial: Deep reinforcement learning, Google DeepMind[EB/OL]. [2022-08-31]. https://www.davidsilver.uk/wp-content/uploads/2020/03/deep_rl_tutorial_small_compressed.pdf.

[5] 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(7587): 484-489.

[6] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.

[7] 尹立蒙. 上帝能否创造一块自己推不动的石头: AlphaGo击败柯洁[M]//社会热点解读(2017). 北京: 北京出版社, 2018: 335-353.

[8] 赵冬斌, 邵坤, 朱圆恒, 等. 深度强化学习综述: 兼论计算机围棋的发展[J]. 控制理论与应用, 2016, 33(6): 701-717.

[9] DeepMind. master-series-60-online-games[EB/OL]. [2022-08-31]. https://www.deepmind.com/research/ highlighted-research/alphago/master-series-60-online-games.

[10] Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that Masters chess, shogi, and Go through self-play[J]. Science, 2018, 362(6419): 1140-1144.

[11] Schrittwieser J, Antonoglou I, Hubert T, et al. Mastering Atari, Go, chess and Shogi by planning with a learned model[J]. Nature, 2020, 588(7839): 604-609.

[12] Stockfish: Strong open source chess engine[EB/OL]. [2022-08-31]. https://stockfishchess.org/.

[13] The 27th World Computer Shogi Championship[EB/OL]. [2022-08-31]. http://www2. computer-shogi. org/wcsc27/index_e.html.

[14] 苏剑波, 陈叶飞, 马哲, 等. 从AlphaGo到BetaGo——基于任务可完成性分析的定性人工智能的定量实现[J]. 控制理论与应用, 2016, 33(12): 1572-1583.

[15] 陶九阳, 吴琳, 胡晓峰. AlphaGo技术原理分析及人工智能军事应用展望[J]. 指挥与控制学报, 2016, 2(2): 114-120.

[16] Wang F Y, Zhang J J, Zheng X H, et al. Where does AlphaGo go: From church-Turing thesis to AlphaGo thesis and beyond[J]. IEEE/CAA Journal of Automatica Sinica, 2016, 3(2): 113-120.

[17] 杨小康. 未来人工智能: 从AlphaGo到BetaGo[J]. 科学, 2017, 69(3): 6-8.

[18] 胡晓峰, 贺筱媛, 陶九阳. AlphaGo的突破与兵棋推演的挑战[J]. 科技导报, 2017, 35(21): 49-60.

[19] Jiang Q, Li K, Du B, et al. Deltadou: Expert-level Doudizhu AI through self-play[C]//Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. 2019: 1265-1271.
[20] Yang X, Duvaud W, Wei P. Continuous control for searching and planning with a learned model[DB/OL]. arXiv preprint: 2006.07430, 2020.
[21] Fawzi A, Balog M, Huang A, et al. Discovering faster matrix multiplication algorithms with reinforcement learning[J]. Nature, 2022, 610(7930): 47-53.
[22] 田渊栋. 阿法狗围棋系统的简要分析[J]. 自动化学报, 2016, 42(5): 671-675.
[23] 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(7587): 484-489.
[24] 唐振韬, 邵坤, 赵冬斌, 等. 深度强化学习进展: 从AlphaGo到AlphaGo Zero[J]. 控制理论与应用, 2017, 34 (12): 1529-1546.
[25] 刘知青, 吴修竹. 解读AlphaGo背后的人工智能技术[J]. 控制理论与应用, 2016, 33(12): 1685-1687.
[26] 薛永红, 王洪鹏. 机器下棋的历史与启示: 从"深蓝"到AlphaZero[J]. 科技导报, 2019, 37(19): 87-96.
[27] 陈铭禹. AlphaGo与AlphaZero原理和未来应用研究[J]. 通讯世界, 2019, 26(12): 22-23.
[28] 唐川, 陶业荣, 麻曰亮. AlphaZero原理与启示[J]. 航空兵器, 2020, 27(3): 27-36.
[29] 李理. 深度学习理论与实战: 提高篇[EB/OL]. (2019-03-14)[2022-08-31]. http://fancyerii.github.io/ 2019/03/14/dl-book/.
[30] 浅述: 从Minimax到AlphaZero, 完全信息博弈之路[EB/OL]. [2022-08-31]. https://zhuanlan.zhihu.com/p/31809930.
[31] Redflashing. 与AI博弈: 从AlphaGo到MuZero [EB/OL]. [2022-08-31]. https://zhuanlan.zhihu.com/p/458714600.
[32] 从α到μ: DeepMind棋盘游戏AI进化史[EB/OL]. [2022-08-31]. https://baijiahao.baidu.com/s?id=1657590770469176607&wfr=spider&for=pc.
[33] CSDN. AI中的搜索(二)——对抗搜索(最小最大搜索Minimax、Alpha-Beta剪枝搜索、蒙特卡洛树搜索MCTS)[EB/OL]. [2022-08-31]. https://blog.csdn.net/hxxjxw/article/details/105848821.
[34] Amir R, Evstigneev V I. On Zermelo's theorem[J]. Journal of Dynamics and Games, 2017, 4(3): 191-194.
[35] Knuth D E, Moore R W. An analysis of alpha-beta pruning[J]. Artificial Intelligence, 1975, 6(4): 293-326.
[36] Herik H J D, Uiterwijk J W H M, Rijswijck J V. Games solved: Now and in the future[J]. Artificial Intelligence, 2002, 134: 277-311.
[37] Müller M. Computer Go[J]. Artificial Intelligence, 2002, 134(1/2): 145-179.
[38] Tesauro G, Galperin G. On-line policy improvement using Monte-Carlo search[C]//Proceedings of the 9th International Conference on Neural Information Processing Systems. New York: ACM, 1996: 1068-1074.
[39] Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search[C]//Proceedings of the 5th international conference on Computers and games. New York: ACM, 2006: 72-83.
[40] 王少锋. 棋类运动在人工智能背景下的发展问题研究: AI围棋对弈软件优化技术研究[C]//中国围棋论丛(第6辑). 杭州: 浙江古籍出版社, 2021: 310-339.
[41] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// Proceeding in Advances in Neural Information Processing Systems. 2012.
[42] Silver D. Reinforcement learning and simulation-based search in computer Go[D]. Edmonton: University of Alberta, 2009.
[43] Kocsis L, Szepesvári C. Bandit based Monte-Carlo planning[M]//Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006: 282-293.
[44] 28天自制你的AlphaGo(6): 蒙特卡洛树搜索(MCTS)基础[EB/OL]. [2022-08-31]. https://zhuanlan.zhihu.com/p/25345778.
[45] Monte Carlo tree search—Beginners guide[EB/OL]. [2022-08-31]. https://int8. io/monte-carlo-tree-search-beginners-guide/.
[46] 董豪, 丁子涵, 仉尚航, 等. 深度强化学习基础、研究与应用[M]. 北京: 电子工业出版社, 2021.
[47] 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.
[48] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem[J]. Machine Learning, 2002, 47(2/3): 235-256.
[49] Rosin C D. Multi-armed bandits with episode context[J]. Annals of Mathematics and Artificial Intelligence, 2011, 61(3): 203-230.
[50] Enzenberger M, Müller M. A lock-free multithreaded Monte-Carlo tree search algorithm[M]//Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010: 14-20.
[51] Silver D, Hasselt H V, Hessel M, et al. The predictron: End-to-end learning and planning[C]// Proceedings of the 34th International Conference on Machine Learning -Volume 70. New York: ACM, 2017: 3191-3199.
[52] Oh J, Singh S, Lee H. Value prediction network[C]//Proceding in the Conference and Workshop on Neural Information Processing Systems. La Jolla: Neural Information

Processing Systems (NIPS), 2017.
[53] 围棋人工智能程序AI之Zen系列[EB/OL]. [2022-08-31]. www.bilibili.com/read/cv13112575/.
[54] Coulom R. Computing Elo ratings of move patterns in the game of Go[J]. ICGA Journal, 2007, 30(4): 198-208.
[55] Baudiš P, Gailly J L. PACHI: State of the art open source go program[M]//Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012: 24-38.
[56] Müller M, Enzenberger M, Arneson B, et al. Fuego—An open-source framework for board games and Go engine based on Monte-Carlo tree search[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2010, 2(4): 259-270.
[57] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation, 2006, 18(7): 1527-1554.
[58] Lecun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.
[59] Clark C, Storkey A. Teaching deep convolutional neural networks to play go[DB/OL]. arXiv preprint: 1412.3409, 2014.
[60] Tian Y D, Zhu Y. Better computer go player with neural network and long-term prediction[DB/OL]. arXiv preprint: 1511.06410, 2015.
[61] Werf E, Herik H, Uiterwijk J. Learning to score final positions in the game of Go[J]. Theoretical Computer Science, 2005, 349(2): 168-183.
[62] Sutskever I, Nair V. Mimicking go experts with convolutional neural networks[M]//Artificial Neural Networks-ICANN 2008. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008: 101-110.
[63] Maddison C J, Huang A, Sutskever I, et al. Move evaluation in Go using deep convolutional neural networks [DB/OL]. arXiv preprint: 1412.6564, 2014.
[64] Yang Y, Wang J. An overview of multi-agent reinforcement learning from game theoretical perspective[DB/OL]. arXiv preprint: 2011.00583, 2020.
[65] KGS围棋服务器[EB/OL]. [2022-08-31]. https://www.gokgs.com/index.jsp?locale=zh_CN.
[66] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. 2nd edition. Cambridge: MIT Press, 2018.
[67] He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Press, 2016: 770-778.
[68] Hussein A, Gaber M M, Elyan E, et al. Imitation learning: A survey of learning methods[J]. ACM Computing Surveys, 2018, 50(2): 1-35.
[69] Khalid E A. Dirichlet processes, a gentle tutorial[EB/OL]. [2022-08-31]. Carnegie Mellon University, 2008, https://www.cs.cmu.edu/~kbe/dp_tutorial.pdf.
[70] Steinitz W. The modern chess instructor[M]. Russell Enterprises, 2017.
[71] Wang T, Bao X, Clavera I, et al. Benchmarking model-based reinforcement learning[DB/OL]. arXiv preprint: 1907.02057, 2019.
[72] Ye W, Liu S, Kurutach T, et al. Mastering atari games with limited data[DB/OL]. arXiv preprint: 2111.00210, 2021.

Principle and methodology of AlphaGo family algorithms

ZHANG Sheng[1], LONG Qiang[2*], KONG Yinan[3], WANG Yu[2]

1. Aerospace Technology Institute, China Aerodynamic Research and Development Center, Mianyang 621000, China
2. School of Mathematics and Physics, Southwest University of Science and Technology, Mianyang 621000, China
3. Computational Aerodynamics Institute, China Aerodynamic Research and Development Center, Mianyang 621000, China

Abstract AlphaGo family algorithms are important milestones in the history of artificial intelligence. These algorithms not only solve the typical complete information game problem such as Go but also are applicable to a wider range of problems. According to their development, this paper summarizes the fundamental principle and technical characteristics for the series of algorithms from AlphaGo Fan to MuZero, elaborating how the AlphaGo family algorithms work. The key technologies employed, including the Monte Carlo tree search, modeling and training of deep neural networks, are surveyed and compared. The AlphaGo family algorithms are of significant instructive value for addressing various problems in practice, from algorithm design, neural network modeling to model utilization. This paper helps to quickly understand the principle of these algorithms and is expected to provide useful reference for the further research and development of algorithms.

Keywords artificial intelligence; AlphaGo family algorithms; Monte Carlo tree search; deep neural network; reinforcement learning

（责任编辑　刘志远）