即时差分学习在六子棋机器博弈中的应用†

徐心和 徐长明 马宗民 李新星
（东北大学信息科学与工程学院 辽宁省沈阳市 110004）
E-mail：xuxinhe@ise.neu.edu.cn

摘　要：鉴于六子棋的 TD 学习应用几乎处于空白,报告了所取得的阶段性成果.提出了采用即时差分(TD, Temporal Difference)学习算法学习六子棋估值函数权值,且实现了调整过程的自动化.避免了传统方法在调整估值函数权值上的重要缺陷，包括必须人工干预,其过程单调、漫长、易出错等.经过 10020 盘自学习训练,自学习程序 TDLConn6 的胜率提高了 8%左右,收到了良好的效果.

关键词：机器博弈 即时差分学习 六子棋

Using TD Learning in Computer Connect6

Xin-he XU, Chang-ming XU, Z.M. MA[1], Xin-xing LI
College of Information Science and Engineering, Northeastern University, Shenyang, 110004
E-mail: xuxinhe@ise.neu.edu.cn

Abstract: Since the application applied TD learning in Connect6 game is almost in blank, this paper reports the relevant staggered results achieved. A method using Temporal Difference (Abbr. TD) learning algorithm to adjust the weights of evaluation function in Connect6 is proposed, which also makes such adjustment process can be done automatically. In comparison, the traditional method to adjust weights of the evaluation function must be manual intervention, which is still a monotonous, endless, and error-prone procedure. After 10020 self-learning training, the program, TDLConn6, increased its winning rate with 8% around, which is a good result.

Key words: Computer Games; Temporal Difference Learning; Connect6

1. 引 言

即时差分(TD, Temporal Difference)学习是最重要的增强学习方法[1],已经成功地应用到许多实际问题中,包括机器人控制、天气预报等.TD 学习的重要目标是,在免于人工干预的情况下,通过自学习训练来调整估值函数的权值.它打破了监督学习的限制.

长期以来,要打造一个高水平的博弈程序,最耗时的工作莫过于估值函数的设计、实现以及人工调参[2],TD学习为改善这种困境提供了一种可能.TD学习算法诞生于Samuel早在1959年的西洋跳棋程序[3]中.此后,在机器博弈领域内就有了更多的应用,如 Tesauro 的西洋双陆棋程序 TD-Gammon[4]、Schaeffer 的西洋跳棋程序 Chinook[5,6]、Baxter 的国际象棋程序 KnightCap[7]等,其水平都接近或超越人类大师.但是,相关经验还不能推广到所有的棋类,时至今日,将 TD 学习方法应用到机器博弈领域仍然是一个开放性问题.

在六子棋程序 NEUConn6 的基础上,设计了支持 TD 自学习的程序 TDLConn6,本文报告

†国家自然科学基金项目(60873010);国家自然科学基金项目(60774097).

了相关的阶段性成果.

2 六子棋简介

connect(*m*, *n*, *k*, *p*, *q*)代表一族 *k* 子棋[8,9],也称连珠棋,博弈双方分别执黑和执白,黑先.*connect*(*m*, *n*, *k*, *p*, *q*)的涵义:第一,棋盘包含 *m*×*n* 个交叉点.第二,黑第一次下 *q* 枚棋子,此后,双方轮流下 *p* 枚棋子.第三,在（水平的、垂直的、对角线方向的）任何一条线上,能率先形成本方连续不间断的 *k* 子序列者取胜;若双方均无法取胜,判和.其中,六子棋可形式化地描述为 *connect*(*m*, *n*, 6, 2, 1).六子棋无禁手;一般采用 19×19 的棋盘.

设博弈双方表示为$side_1$和$side_2$,P_0~P_n是局面序列,初始的局面为p_0,则在着法的驱动下六子棋的棋局演化过程将如图1所示.

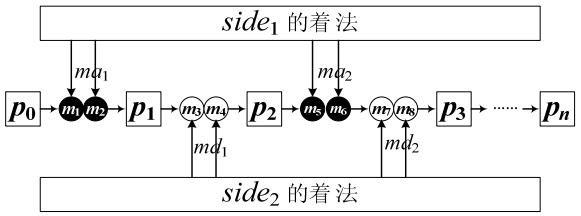

图 1□六子棋的棋局演化过程

Fig.1 Procedure illustrating the position evolution in Connect6 game

3 TD(λ)学习

TD(λ)可解决多步的预测问题,文献[1~4]等均对它作了深入的介绍和分析.本文以机器博弈为背景简要介绍基本TD(λ).令$x_1, x_2, \cdots, x_m, z$为一个状态（局面）序列,其中x_t是t时刻的状态(t=1, 2, …, m),每个状态x_t都对应着一个向量,而向量的每个分量对应着一个局面特征,z为走完最后一步棋的局面状态（即,可从z判断出胜、负、和）.对于上述局面状态所构成的观测序列,有一个相应的对z值的估计序列$P_1, P_2, \cdots, P_m$.理论上,每一个预测P_t应为x_t以及x_t之前所有状态的函数,但为了简化运算,只将P_t作为x_t的函数.令$P_t=P_t(w, x_t)$,其中,w是权值向量.于是,TD(λ)学习的过程便可归结为通过修正w来拟合估值函数的过程.

TD(λ)利用回报值进行学习,这样,利用现有的监督训练技术就能实现函数的拟合.一般的监督学习方法会构造一个监督学习序列$(x_1, z), (x_2, z), \cdots, (x_m, z)$,从而对权值向量$w$加以训练.而在即时差分学习中,用于训练的则是一个差分序列,即,$(x_1, P_1), (x_2, P_2), \cdots, (x_m, P_m)$.考虑到机器博弈的特点,采用“每个序列一更新”的权值更新方式,即,每结束了一盘棋之后,才一次性地将序列的所有的权值增量都叠加到权值向量w上

$$w \leftarrow w + \Delta w = w + \sum_{t=1}^{m} \Delta w_t \tag{1}$$

根据梯度法则,TD(λ)的Δw_t可由下式求出:

$$\Delta w_t = \alpha (P_{t+1} - P_t) \sum_{k=1}^{t} \left[\lambda^{t-k} \frac{\partial P_k}{\partial w} \right] \tag{2}$$

这里,梯度$\frac{\partial P_k}{\partial w} = \frac{\partial P_k(w, x_k)}{\partial w}$是$P_k$关于向量$w$的偏导数.$\alpha \in [0.0, 1.0]$是学习速率.α决定着参数调整的幅度.α越大,权值的调整幅度就越大;反之,幅度越小.$\lambda \in [0.0, 1.0]$是衰减因子.λ决定着学习过程是从短期的预测中学习,还是从长期的预测中学习.特别是,当λ=0时,仅从下一个状态中学习;当λ=1时,仅从最终结果中学习,TD学习退化成监督学习.

最后,权值向量更新结果由下式得出:

$$w \leftarrow w + \sum_{t=1}^{m} \alpha (P_{t+1} - P_t) \sum_{k=1}^{t} \left[\lambda^{t-k} \frac{\partial P_k}{\partial w} \right] \tag{3}$$

其中,$P_{m+1} = z$. (3)式中还包含着可增量执行的计算[1].若令

$$e_t = \sum_{k=1}^{t}\left[\lambda^{t-k}\frac{\partial P_k}{\partial w}\right] \tag{4}$$

则有

$$\begin{aligned} e_{t+1} &= \sum_{k=1}^{t+1}\left[\lambda^{t+1-k}\frac{\partial P_k}{\partial w}\right] \\ &= \frac{\partial P_{k+1}}{\partial w} + \sum_{k=1}^{t}\left[\lambda^{t+1-k}\frac{\partial P_k}{\partial w}\right] \\ &= \frac{\partial P_{k+1}}{\partial w} + \lambda e_t \end{aligned} \tag{5}$$

4 估值函数

估值函数的设计是博弈程序最重要的议题之一.由于博弈问题的复杂性，设计出一个好的估值函数是非常困难的事情.

4.1 手工调整参数

在早期的博弈程序中,手工调参方法占据着主流地位.在初步设计好程序的估值函数之后,主要通过观察大量对局中“犯严重策略性失误”的局面,来调整估值函数的各个参数.但是,手工调参必然要求人类专家花费大量的时间和精力,而且,该过程非常漫长,也容易出错.

能击败人类世界冠军的西洋跳棋程序 Chinook[5],其开发者耗费了近五年的时间手工调整估值函数的权值.可见,手工调参的工作量之大.

4.2 权值调整自动化——BP 神经元网络

作为 TD 学习的主要目标,估值函数的设计对学习的效果有深远的影响.与手工调整参数相比,神经元网络就有较强的优势.神经元网络具有极强的泛化能力,网络的演化也不需要人为干预.理想地,它能学习任何复杂的非线性的映射关系,或者拟合任何连续函数.网络的训练没有假设任何先验的训练数据分布或输入-输出映射函数.对于非完整的,有缺失的和有噪声的数据,网络具有很强的适应性和容错性.必须承认,神经元网络也有明显的缺陷.关于如何解释神经元网络的能力,缺乏足够的理论支撑,所以常被看作黑盒.神经元网络还容易陷入局部的最优解.

尽管神经元网络远不完美,作为拟合函数的工具应用到博弈程序中,还是具有良好前景的.TDLConn6引入了一个三层BP神经元网络,将它作为六子棋的评估函数的有机组成部分,并用TD(λ)算法训练它,其神经元网络的诸要素如下:

网络的输入层.参照吴的工作[8,9]以及我们早期的工作[11],TDLConn6 的局面特征主要选择为整个局面的重要棋型的类型及数目、重要空交叉点的类型和数目等.最终,TDLConn6 设计了 32 个综合性的局面特征作为网络的输入.其中,黑方和白方各 16 个特征.每个特征对应一个输入单元,共 32 个输入单元.

网络的隐藏层.隐藏层结点过少时,学习的容量有限,不足以存储训练样本中蕴含的所有规律;结点过多时,不仅会增加网络训练时间,而且会将样本中非规律性的内容如干扰和噪声存储进去,反而降低泛化的能力.TDLConn6 设计了 14 个隐藏层单元.

网络的输出层.TDLConn6 的网络输出层有 2 个单元.输出的值是实数类型.分别是对黑方和白方特征的综合评估值.局面最终的价值用“走棋方特征的综合评估值−另一方的综合评估值”这一差值来计算.

传输函数和初始权值.网络的相邻两层之间的传输函数对网络的拟合性能影响很大.TDLConn6 选用 Sigmoid 函数,即 g(*x*)=1/(1+exp(-*x*))作为传输函数.Sigmoid 函数的值域为(0.0, 1.0),对于走棋方的输出单元,用 1.0 表示取胜,0.5 表示和棋,0.0 表示输棋.权值的初始化决定了网络的训练从误差曲面的哪一点开始.在 Sigmoid 函数中,若每个结点的输入均在零点附

近,则输出均出在传输函数的中点,这个位置不仅远离转换函数的饱和区,而且是其变化最灵敏的区域,有助于加快网络的学习速度.因此,TDLConn6 将神经元网络的权值都初始化为零.当神经元网络的权值都为零时,网络各层神经元的加权和也全为零.为避免此情况,TDLConn6 在隐藏层和输出层都加入了介于-1.0 和 1.0 之间的随机实数作为偏置值.

5 实验及结果

5.1 训练集的生成

六子棋的开局、中局、残局策略几乎没有明显的不同,不妨选择常见的开局局面作为训练对象.我们挑选了 26 个均衡的开局局面,它们的第一颗黑子均在天元,都只有 3 枚棋子.取 S_T={与这 26 个局面按旋转对称、轴对称等价的 170 个局面}.S_T 仅有 170 个局面,因为太小而不宜直接用作训练集.在兼顾训练局面的质量基础上,提出一种带有随机性的自动扩展集合 S_T 的方法,使扩展后的局面集保证训练样本具有代表性.

已知 $S=S_T$,S'={自动生成的训练局面}.$||p||$表示局面 p 中棋子的数目.在 TDLConn6 中,取常数 c=8,令局面 $pos\in S$, $pos'\in S'$,且$||pos'||-||pos||\le c$,则随机生成满足上述条件的 pos'的算法见图 2.在图 2 中,当前局面为 $p\in S\cup S'$,p'是 p 的一个直接后继局面,$p'\in S'$.

```
1.  move_list := GenAllMove(p);
2.  k := Min(TOP-K, num_of_moves);
3.  best_k_list := SelectBestK(k, move_list);
4.  for ( i=1; i ≤ k; ++i) {
5.      probability[i]  = P'(best_k_list[i]|p);
6.  }
7.  x := SelectOneMoveAtRandom(probability, k);
8.  p' := MakeMove(p, best_k_list[x]);
```

图 2□训练集的随机性扩展算法

Fig2. The algorithm to extend the training set at random

TOP-K 决定了入选着法的范围大小.在图 2 的第 5 行中,根据(7)式计算着法 s 的概率 $P'(m_s|p)$.(7)式可使随机着法选择过程更加偏爱分值高的着法.

$$P(m_s \mid p)=\frac{V(Succ(p,side,m_s))}{\sum_{i=1}^{k}V(Succ(p,side,m_i))} \tag{6}$$

$$P'(m_s \mid p)=\frac{(k-s+1)^2 P(m_s \mid p)}{\sum_{i=1}^{k}[(k-i+1)^2 P(m_i \mid p)]} \tag{7}$$

(6)式中的 V(p)是局面的估值函数.由于着法选择带有随机因素,因此,即便从同一个初始局面出发,后续着法也往往不同,况且,每盘棋结束之后,神经元网络权值往往发生变化,候选着法列表和候选着法的估值也随之变化.所以,图 5 扩展 S_T 的方法是比较理想的,得到的也是一个相对公平的测试集.

在自学习训练过程中,自动生成训练局面的阶段($||pos'||-||pos||\le c$)采用带有随机性的着法选择策略;在剩余阶段($||pos'||-||pos||>c$),则利用估值函数,根据极大极小原理选择最佳着法.

5.2 “零知识”自学习训练

在 TDLConn6 中,采用“零知识”的估值函数,即权值都为 0 的神经元网络作为估值函数,并且不采用 TSS[10]策略,经过 10000 盘自学习训练(约 4 小时),其着法仍然散乱无章.这说明,它还不能很好地学到占据中间位置的重要性,更不用说 TSS 这种高级知识了.在与入门级选手的 3 局比赛中,3 盘皆负.在“零知识”条件下,经 10000 盘训练的与未经训练的两个 TDLConn6 相比,除去和棋之外,其胜负比率几乎接近 1:1.显然,“零知识”条件不可取.

5.3 引入先验知识的自学习训练

在 TDLConn6 中,适度地加入了先验知识,主要包括位置分值,单个棋型分值、多个棋型交叉的分值等.此外,TDLConn6 还将 TSS 策略视为六子棋的高级知识.特别地,两个对弈的引擎思考时,都检验可否通过 TSS 直接取胜,若一方发现自己能够取胜,则立刻通知对手终止本局对弈,并开始新一局训练.

对 S_T 中的每个局面都进行 60 次自学习训练,共 10020 次.考虑到训练次数不多(与 TD-Gammon 训练 150 万次相比),令α=0.8,使参数调整的步幅较大;此外,取λ=0.7.整个训练耗时约 157 小时（约 6.6 天）.为了评价学习的效果,每训练 1000 盘还把神经元网络的权值保存下来,用于分析学习的效果.

对于 S_T 中的每个局面(共 170 个),都让 TDLConn6 与 NEUConn6 对弈一局(分先后手),同时关闭 TDLConn6 中的随机性扩展选项,在整个对弈过程中,每个引擎都选择各自的最佳着法.TDLConn6 执黑时,在 S_T 中的胜率提高了 3~5%左右;TDLConn6 执白时,胜率提高了 8%左右.程序水平变化的整体趋势是,随着训练的次数增多,水平逐渐提高.可见,应用 TD 学习有助于提高六子棋程序的博弈水平.

6 结 语

本文将 TD(λ)算法应用到六子棋机器博弈中,从零知识开始进行自学习训练的方案能够在西洋双陆棋中取得良好效果.但本文的实验结果表明,零知识训练不适合六子棋博弈.经过 10020 盘（耗时约 157 小时）自学习训练,程序的水平明显提高.可以断言,若像 TD-Gammon 那样进行上百万盘的自学习训练（大约将近 3 年时间）,TDConn6 的博弈水平必然会更好.

在本文工作的基础上,未来还将在以下方面进行探索: 1)把αβ深层搜索与 TD 学习密切结合起来;2)通过网络对弈平台,向人类高手学习等.

参考文献

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

[2] J Baxter, A Tridgell, L Weaver. TDLeaf(λ): Combining Temporal Difference Learning with Game-Tree Search [C]. *Proceedings of the 9th Australian Conference on Neural Networks*, 1998: 168-172.

[3] A L Samuel. Some Studies in Machine Learning Using the Game of Checkers [J]. *IBM Journal of Research and Devolopment*, 1959, 3: 210-229.

[4] G Tesauro. Temporal difference learning and TD-Gammon [J]. *Communications of the ACM*, 1995, 38(3), 58-68.

[5] J Schaeffer, N Burch, Y Bjornsson, A Kishimoto, M Muller, R Lake, P Lu, S Sutphen. Checkers Is Solved [J]. *Science*, 2007, 317(5844): 1518-1522.

[6] J Schaeffer, M Hlynka, V Jussila. Temporal Difference Learning Applied to a High-Performance Game-Playing Program [C]. *IJCAI*, 2001: 529-534.

[7] J Baxter, A Tridgell, L Weaver. KnightCap: A Chess Program that Learns by Combining TD(λ) with Game-Tree Search [C]. *In Proceedings of the 15th International Conference on Machine Learning*, Madison, 1998: 28-36.

[8] Wu I-Chen, Huang Dei-Yen. A New Family of *k*-in-a-row Games [C]. *In Proceedings of The 11th Advances in Computer Games Conference*, 2005: 88-100.

[9] Wu I-Chen, Huang Dei-Yen, Connect6 [J]. *ICGA Journal*, 2005, 28(4): 234-242.

[10] L V Allis, H J van den Herik, M P H Huntjens, Go-Moku and Threat-Space Search [J]. *Computational Intelligence*, 1995, 12(1): 7-24.

[11] Chang-ming Xu, Z.M. Ma, Xin-he Xu. A Method to Construct Knowledge Table-Base in *k*-in-a-row Games [C]. *ACM Symposium on Applied Computing*, 2009: 929-933.