六子棋上基于路的即时差分学习模型

范深[1]

（电子科技大学，计算机科学与工程学院 四川成都 611731）

摘　要：以六子棋机器博弈为背景，运用基于“路”的棋局评估模型，并且采用即时差分学习的方法对评估参数进行训练寻优，设计出的算法框架具有解释性强、效率高、可扩展的性质，经过 500 余场训练，新的参数集超越旧参数获得了更高的成绩，并成功开发出可玩性强的应用程序。

关键词：六子棋；路；　即时差分学习；博弈；

Temporal Difference Learning Algorithm Based on Road of Connect6 Game

Shen Fan

(School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Temporal Difference learning algorithm was used to adjust weights of evaluation function by using Connect6 game as tested in this paper, which makes the weights adjustment process can be done automatically. A new promoted evaluation scheme named Road was proposed, which is clear and of high interpretability. After 500 self-learning training, the new weight sets outperforms the old one, which is a good result.

Key words: Connect6; Temporal difference learning; Computer games

博弈程序设计，是指通过搜索的方法寻找目标解的一个合适操作序列，并满足棋局的各种约束。六子棋是近年新兴的一种对弈游戏，是台湾交通大学吴毅成教授[1,6]发明的一种新型计算机博弈平台。六子棋的发明主要是为了解决其前驱“五子棋”先手必胜的公平性问题[6]。理论可以证明六子棋是潜在公平的，并且在 ThinkNewIdea 公司支持的六子棋在线游戏平台的测试下亦可证实六子棋的公平性。

目前对六子棋博弈采取的常用策略有极大极小值搜索、Alpha-Beta 剪枝搜索等算法。其中吴毅成教授编写的搜索深度为 3 的 alpha-beta 搜索树的六子棋程序，在对战平台中，可以打败 70%的玩家，有着不俗的表现。

本文首先对六子棋规则以及博弈流程进行简单介绍，接着介绍基于“路”的评估方法，然后提出该方法的 TD-Learning 框架和算法，最后对该方法进行实证评估。

作者简介：范深（1991-），男，电子科技大学计算机科学与工程学院计算机科学与技术专业2010级本科生，主要兴趣领域：机器学习、社会计算。

1 六子棋模型

定义 1. Connect(m,n,k,p,q)表示在 m*n 的棋盘上，有两个玩家，表示为黑方和白方，黑方初始先下 p 子，从后白黑交替下 q 子，哪方先获得 k 子连续相连（包括水平、垂直、斜方）就判为获胜。

例如:五子棋可表示为 Connect(15,15,5,1,1)，即在 15*15 的棋盘上，黑白交替各下一子，直至连珠 5 子方获胜。但是简单分析可知，对黑方没有禁手的话，黑方有利于白方。黑方每下完一步，都比白方多一个棋子。

六子棋的定义可由 **Connect(19,19,6,2,1)**表示。即在 19*19 的棋盘上，黑棋先下一子，之后双方轮流着子，每手放**两个**棋子于棋盘上。棋局中先完成连六或连六以上者获胜，当棋盘交点全部下满而无人连六以上，或对局双方皆同意打平时即为和局。经过这样的规则设定，可以发现，在行棋过程中，双方交替领先对方一子，便解决了五子棋中黑子始终占优的不公平性。

2 博弈流程

按照人工智能的一般研究方法，我们可将六子棋博弈系统划分为开局(opening)、中局(middle-game)和收尾(end-game)3 个阶段和棋形判断、搜索引擎、评估函数、走法生成 4 个部分组成。

开局中，一般有比较经典的行棋套路。所以可以通过查询策略库、定式库来快速选择着法 [10]。一般开局是以 9~11 颗棋子为分界点，在六子棋中，大概是双方都下 3 手棋后，进入中局阶段。

中局包含了博弈的核心算法。一般程序是，对棋形信息进行搜集后寻找较优的着子点，寻优的过程就是搜索、评估的过程，最终选取评估系统认为最优的着法落子。中局大概在开局后的 20 步内，剩下的对弈即进入收尾阶段。

棋形判断是博弈程序设计中至关重要的一垒，对棋局判断是不足(under fit)、准确(just right)、还是过拟合(over fit)，是由判断棋形的特征选择及其评估函数决定的。若判断不敏锐，则易被对方偷袭，若过于警觉敏感，则容易凌乱阵脚。通过统计学习的方法对棋形判断内核进行训练，就可以有效的避免以上两种失误。

3 算法设计

3.1 基于“路”的贪心算法

定义 2. 路就是指在棋盘上存在连续 6 个可能连成一线的点位集 [5,7]。也有研究者把它称为“6 窗口”。

“路”的总数很少，按照横、纵、左斜、右斜 4 个方向的特点，可以分别计算不同方向中“路”的数目：横向和纵向各有 19 ×（19-6+1）= 266 路，左斜和右斜各有 14 ×（19-6+1）= 196 路，所以，全盘“路”的总数有：942 路。

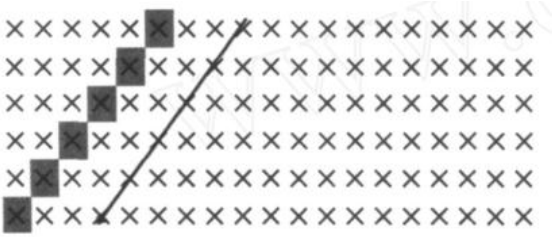

图 1 斜向“路”的形成示意图

通过统计 942 条路上的状况，就能对棋局进行相当准确的评估，定义评估函数如下：

$$V = \sum_{i=1}^{6} MyRoad[i] * \theta[i] + \sum_{i=1}^{6} EnemyRoad[i] * (\theta[i+6]) \quad (1)$$

其中 MyRoad[i]表示的是我方 i 路棋的个数，EnemyRoad[i]表示对方 i 路棋的个数。$\boldsymbol{\theta}$ 是一个权重系数向量，分别对双方各路棋给予相应的权重。参照文献[1]的结果，我们通过人工试验调整，最初设定权重的分配为：

$$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6 = \{12,65,201,771,1000,10000\};$$

$$\theta_7, \theta_8, \theta_9, \theta_{10}, \theta_{11}, \theta_{12} = \{-14,-66,-253,-3890,-5000,-10000\};$$

需要特别注意的是，在统计“路”的时候必须是有效的，单纯的路。即在该“路”上只能有一种颜色的棋子，否则该路将不可能形成连六，也不会成为我们考虑的情况了。

基于“路”的评估函数具有很好的可解释性，多子对局面的贡献必然高于少子，但高多少则有参数权重决定。比如，假设我方可选走法中，可新增 2 个 3 路棋或 1 个 4 路棋，那么当$\theta_3 = 141, \theta_4 = 788$时，必然选择新增 1 个 4 路棋为好，而当$\theta_4 = 788, \theta_5 = 1030$时，新增 2 个 4 路棋得到的回报明显优于新增 1 个 5 路棋。又如，我们采取了和我方路权重不对称的对方路权重参数，为了更好地监测险情，尤其是当对方存在连四、连五局面时，加大权重的绝对值，使得程序能对危险局面进行有效的防守。

在行棋中，对于每一步，穷举任意可能的测试点<x1,y1,x2,y2>，最多不超过 130321 种（19*19*19*19）组合。对于每一组测试点试放，并进行棋局评估（Evaluate-ChessStatus），将评估分值与当前最优着法分值进行比较，若更优则替换最优着法，收回本次试放棋子，进行下一次试放。最后选定着子后对局面评分提升最多的一种着法进行着子。

该算法伪代码见图 2。

Greedy Road Algorithm Applied to Connect6

```
Procedure Greedy(board)
    MyColor = board.WhotoPlay()
    a* = Pass; V* = 0
    for all a ∈ board.Legal() do
        board.Play(a,MyColor)
        V = Eval(board)
        if V ≥ V* then
            V* = V; a* = a
        end if
        board.Undo()
    end for
    return a*
end procedure
Procedure Eval(board)
    φ = board.GetFeatures()
    v = 0
    for all i ∈ φ do
        v +=φ[i]θ[i]
    end for
    return v
end procedure
```

```
Procedure GetFeatures()
    MyRoad.clear()
    EnemyRoad.clear()
    for all r ∈ Roads do
        Count_my,Count_Enemy = r.count()
        //统计路 r 上双方棋子数
        if Count_Enemy == 0 then
            MyRoad[Count_Enemy]++
        end if
        if Count_my == 0 then
            EnemyRoad[Count_Enemy]++
        end if
    end for
    for i=1:6 do
        φ_i = MyRoad[i]
        φ_{i+6} = EnemyRoad[i]
    end for
    return φ
end procedure
```

图 2 基于路的贪心算法

在图 2 的算法核心代码中，*Greedy(board)*用于选择最优的着子点<x1, y1, x2, y2>下子。*Eval(board)*的主要作用是返回局面的评估分值。*GetFeatures()*函数用于返回棋局的特征向量。

要打造一个高水平的博弈程序，人工调参是一个很冒险的做法，因为人工调参周期长、易出错。下面我们运用 TD 学习的方法，尝试对“路”的参数向量作优化，提升算法智能。

3.2 TD学习模型

TD 学习(Temporal difference Learning，简称 TD-Learning)，是一种强化学习的方法。这种方法结合了蒙特卡罗方法和动态规划的特性。TD-Learning 仿蒙特卡罗方法通过对局面取样进行学习，并且有很强的阶段性，如同动态规划一样根据前阶段的参数刻画局面并作局部参数优化。

TD 学习的过程通过自助统计（bootstrapping）来完成。一个经典的例子 [2] 是：如果你想预测本周六的天气情况，现在你手头上有若干个模型可以用来预测周六的天气。我们已经有了周一到周五的天气记录，在一般的方法中，我们会等到周六再来判断哪个模型预测最准确。而实际上我们在周五晚上就已经对周六的天气有了一个很好的感觉，那么通过 TD-Learning 的方法，我们可以回溯地修改在周一的观察（view）记录下对周六天气的预测。

作为强化学习的一种重要方法，TD-Learning 符合强化学习的系统结构。它由 Environment 和 Agent 两部分组成。Agent 和 Environment 不断进行试探交互，得到 Sensor data，Agent 在策略$\pi^*: S \to A$的指导下，选择最优的动作（其中 S 为状态集，A 为动作集），Agent 的任务就是找到一个最优控制策略$\pi^*: S \to A$，获得最大的累计奖赏 [9]，即求解π^*：

$$\pi^* = \mathrm{argmax}_\pi V^\pi(\mathrm{s}), \forall \mathrm{s} \in \mathrm{S} \tag{2}$$

于是，类似地，可以将 TD-Learning[3,4] 用于六子棋博弈算法设计中。

不妨设 S 表示棋局的所有可能状态。行棋按照有序的时间序列 t = 1,2,...。在时间 t 时，用$x_t \in S$来表示局面，并且备选落子点$\mathrm{a} \in \mathrm{A}_{x_t}$，那么从$x_t$采取动作 a 过渡到$x_{t+1}$的概率为$\mathrm{p}(x_t, x_{t+1}, a)$。

设函数$\widetilde{\mathrm{V}}: \mathrm{S} \times \mathrm{R}^{\mathrm{k}} \to \mathrm{R}$ 评估棋局局面。采用最简单的 TD(0)算法，它的迭代公式是：

$$V_{t+1}(\mathrm{x}_t) = V_t(\mathrm{x}_t) + \alpha(r_t + V_t(\mathrm{x}_{t+1}) - V_t(\mathrm{x}_t)) \tag{3}$$

其中α是学习率（learning rate），$V_t(\mathrm{x}_t)$为 Agent 在 t 时刻对状态x_t的值的函数估计，$V_t(\mathrm{x}_{t+1})$为 Agent 在 t 时刻对状态x_{t+1}的值函数估计，r_t为 Agent 从状态x_t转移到状态x_{t+1}时获得的即时回报值。

令$\delta_t = r_t + V_t(\mathrm{x}_{t+1}) - V_t(\mathrm{x}_t)$，在六子棋中，当游戏结束时，Agent 可以得到一个奖赏值r_N，用“1”表示赢，“0”表示平局，“-1”表示输，在游戏进行过程中，r_t为 0。并且游戏结束时，让$V_{\mathrm{N}}(\mathrm{x}_N) = \mathrm{r}_N$，则可以得到简化后的误差函数：

$$\delta_t = V_t(\mathrm{x}_{t+1}) - V_t(\mathrm{x}_t) \tag{4}$$

有了上面的推导，我们就可以来设计基于“路”的 TD-Learning 算法了。取我方“路”数统计值和对方路数统计值共 12 个变量来作为特征φ(s)，即：$\varphi(\mathrm{s}) = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}\}$

其中φ_1~φ_6统计我方路数，φ_7~φ_{12}统计对方路数。

则局面估计函数$\widetilde{\mathrm{V}}$可表示为：

$$\widetilde{\mathrm{V}}(\mathrm{x}) \coloneqq \varphi(\mathrm{x}) \cdot \theta \tag{5}$$

参数向量θ的更新公式如下：

$$\Delta\theta := \Delta V(x_t)\frac{\varphi(x_t)}{\|\varphi(x_t)\|^2} = \alpha\delta_t\frac{\varphi(x_t)}{\|\varphi(x_t)\|^2} \quad (6)$$

其中$\|\varphi(x_t)\|^2 = \sum_t \varphi(x_t)^2$

Algorithm TD(0) Learning Applied to Connect6

```
Procedure TD_Learning(n)
    i = 0
    while i < n do
        board.Initialize()
        SelfPlay(board)
        i++
    end while
end procedure
Procedure SelfPlay(board)
        t = 0
        φ0 = board.GetFeature()
        V0 = board.Eval()
        while not board.Terminal() do
            at = Greedy(board,ε)
            board.Play(at)
            t++
            φt = board.GetFeature()
            Vt = board.Eval()
            if t ≥ 2 then
                δ = Vt-Vt−2
                Norm = ‖φt−2[i]‖²
                for all i ∈ φt−2 do
                    θ[i] += αδφt−2[i]/Norm
                end for
            end if
        end while
end procedure
Procedure Greedy(board, ε)
    if Bernoulli(ε = 1) then
        return Random(board)
    end if
    MyColor = board.WhotoPlay()
    a* = Pass; V* = 0
    for all a ∈ board.Legal() do
        board.Play(a,MyColor)
        V = Eval(board)
        if V ≥ V* then
            V* = V; a* = a
        end if
        board.Undo()
    end for
    return a*
end procedure

Procedure Eval(board)
    φ = board.GetFeatures()
    v = 0
    for all i ∈ φ do
        v +=φ[i]θ[i]
    end for
    V = 1/(1 + e^−v)
    return V
end procedure
```

图 3 基于“路”的 TD(0) Learning 算法

3.3 评估

我们对 TD_Learning 设置$\alpha = 0.1$，$\varepsilon = 0.1$，并使用了标准化后的经验参数作为初始参数：

$$\theta = \{0.0012, 0.0065, 0.0201, 0.0771, 0.1000, 1.0, -0.0014, -0.0066, -0.0253, -0.3890, -0.5, -1.0\}$$

对于开局库的构造，我们采用如下方法：开局的前 t 步(t=6~10)，不对参数进行更新，仅使用当前参数。当行棋到第 t+1 步时，才开启参数更新模块和随机落子模块。实验表明，这样的设定是有效的。

调用 TD_Learning(500)对参数进行了 500 场（历时 7 小时 46 分）的训练后，我们得到参数如下：

$$\theta = \{0.00009, 0.0052, 0.0207, 0.0789, 0.1002, 1.0, -0.00003, -0.0048, -0.0267, -0.3887, -0.49, -1.0\}$$

将新参数 Agent 与原参数 Agent 进行 100 场对弈。我们记赢一场得 2 分，平局得一份，失败的 0 分。经过了 100 场比赛后，采用 TD_Learning 的算法获得了 138 分，原参数算法得到 62 分，成功超越！

4 总结与展望

在六子棋博弈算法设计中，权重参数的设定对于算法的博弈能力有至关重要的作用。我们通过 TD 学习的方法，在人工经验设定权重的基础上寻求更优的参数组合，取得了相当好的成绩。同时，基于“路”的棋局评估方法和 TD 学习的参数优化方法具有很好的可解释性和可移植性，可以应用在任意其他子连珠棋下。

为了更好的检验算法和应用算法，我们设计了成电小六（UESTC-Connect6）博弈平台，软件界面如图 4 所示。平台可选人机对战、双人对战、机机对战三种模式，在小范围用户测试中，得到了很高的评价。在实际软件设计中，还会遇到很多问题，诸如用户抱怨 Agent 的着子策略太单一容易被记录，我们针对此问题，设计了候选参数矩阵，每次从中随机选取一组参数应对用户，可提升 Agent 着法的多样性。

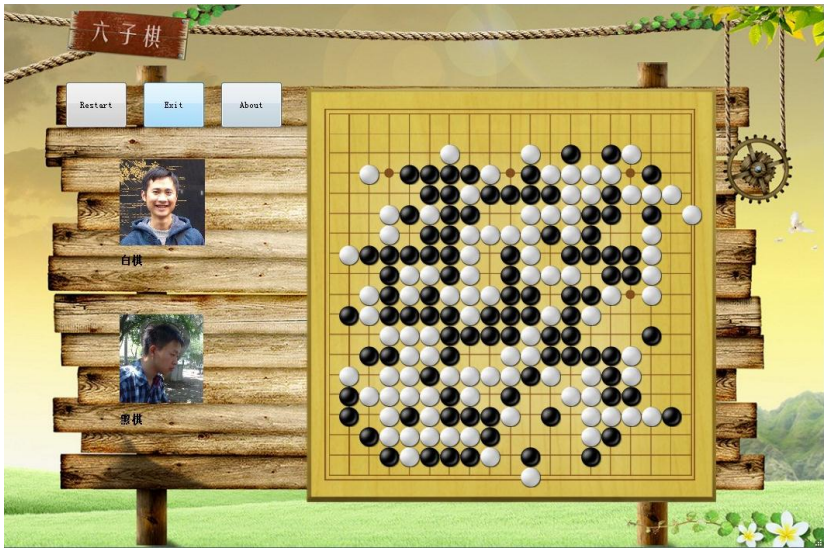

图 4 UESTC-Connect6 软件界面

本文以六子棋博弈为背景，在 TD 学习的基础上实现了有效的参数寻优。在算法设计中，依然还有很多可挖掘的地方，比如，六子棋博弈可以划分阶段(Stage)，不同的阶段采取的策略应有些许不同：开局以布位为主，中局进攻，进入尾声后，应在防守中寻求突破。如果能对不同阶段训练出对应的参数，程序的智能或许能更上一层；另外，在参数训练过程中，可以将机器自博弈与人工参与博弈结合起来，让“学习”参与到实战中，训练出更新颖、更敏锐的反应。

致谢 感谢高辉、刘震老师为我们提供了大量文献材料和理论指导。软件实现中，李勇军、蒋磊完成了用户交互程序，何菲菲为 UESTC-Connect6 程序设计了背景模板。感谢电子科技大学大学生创新训练项目、电子科技大学计算机学院“珠峰计划”、中国科学院“龙星计划”的资助和支持。

References:

[1] I-Chen Wu and Dei-Yen Huang, A New Family of k-in-a-row Games，Advances in Computer Games, 2006

[2] Richard Sutton (1988). "Learning to predict by the methods of temporal differences". Machine Learning 3 (1): 9–44. doi:10.1007/BF00115009

[3] I-Chen Wu, Hsin-Ti Tsai, Hung-Hsuan Lin, Yi-Shan Lin, Chieh-Min Chang, Ping-Hung Lin, "Temporal Difference Learning for Connect6", The 13th Advances in Computer Games Conference (ACG 13), Tilburg, The Netherlands, 20-22 November 2011.

[4] Baxter, J., Tridgell, A., Weaver, L.: Learning to Play Chess Using Temporal Differences.Machine Learning 40(3), 243–263 (2000)

附中文参考文献:

[5] 张小川,陈光年,张世强,孙可均,李祖枢. 六子棋博弈的评估函数. 《重庆理工大学学报》, 2010,24(2):64.

[6] 吴毅成 等. 六子棋. ICGA Journal, 2005

[7] 黄继平,张栋,苗华 六子棋智能博弈系统的研究与实现. 《电脑知识与技术》,Vol.5,No.25,September 2009, pp.7198-7200

[8] 李果 基于遗传算法的六子棋博弈——评估函数参数优化.《西南大学学报(自然科学版)》Vol.29 No.11.

[9] 刘忠,李海红,刘全 强化学习算法研究. 《计算机工程与设计》2008 年 第 22 期 5 页 5805-5809 页

[10] 闵文杰 六子棋计算机博弈关键技术研究[D];重庆交通大学;2010 年