

大作业-强化学习部分

张辛农
2022-11-08

Outline

- Games Intro
 - Game-1: 四子棋
 - Game-2: 圣诞糖果
 - Game-3: 太空采矿
 - Game-4: 光明小镇
- Tutorial (take Gomoku as a example)

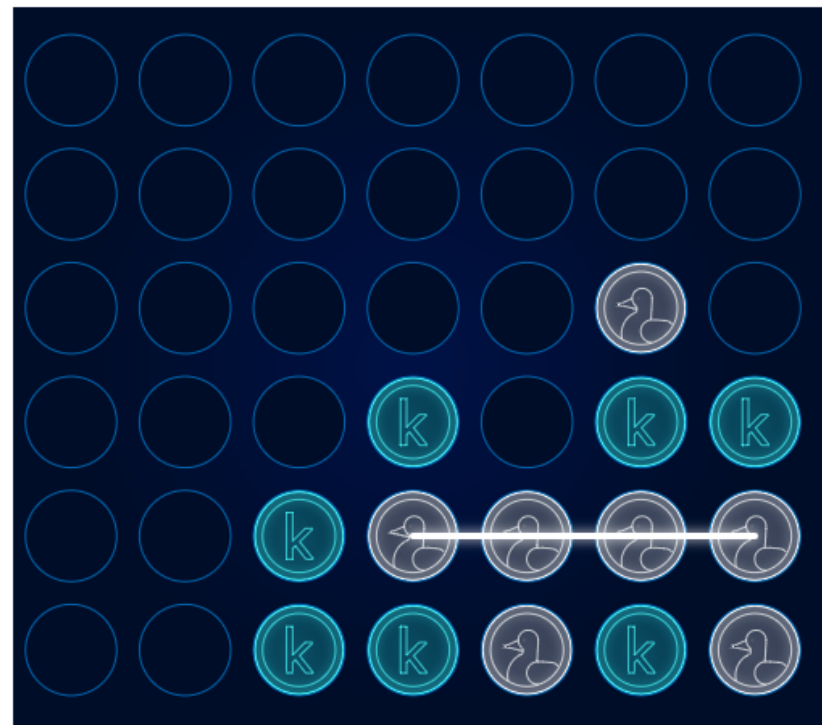
Games Introduction

- Game-1: 四子棋 (同第一阶段)
- 使用强化学习方法优化原先的MCTS策略!
- 比赛主页

<https://www.kaggle.com/competitions/sds-ai-connect-four>

- 参赛链接：

<https://www.kaggle.com/t/1f78e3ba557b47bc960dc15b4f8cf9bc>



Games Introduction



- Game-2: 圣诞糖果
- 基于多臂老虎机模型的博弈问题:
 - 100台自动售货机, 每台售货机基于自身特定的概率分布随机提供糖果(回报);
 - 每轮每个玩家选择一台售货机, 选择后下一轮该售货机奖励的概率降低3%;
 - 可以看到对手的选择, 但不知道对手是否获得了糖果;
 - 2个智能体, 各2000轮, 共4000次;
- 比赛主页 <https://www.kaggle.com/competitions/sds-ai-santa>
- 参赛链接 <https://www.kaggle.com/t/889af6e30458499a84add6c3d150b379>
- 参考链接 <https://www.kaggle.com/competitions/santa-2020/overview>

Games Introduction

- Game-3: 太空采矿
- 21✖21的太空中随机对称分布着2750个kore矿物;
 - 开局一艘飞船+500矿;
 - 船厂: 制造飞船、制定计划;
 - 舰队: 按照计划飞行、转化为船厂;
 - 多艘舰队在同一格时, 按规则吞并矿物、撞毁
 - 400轮后, 持有最多矿物的一方获胜

- 比赛主页

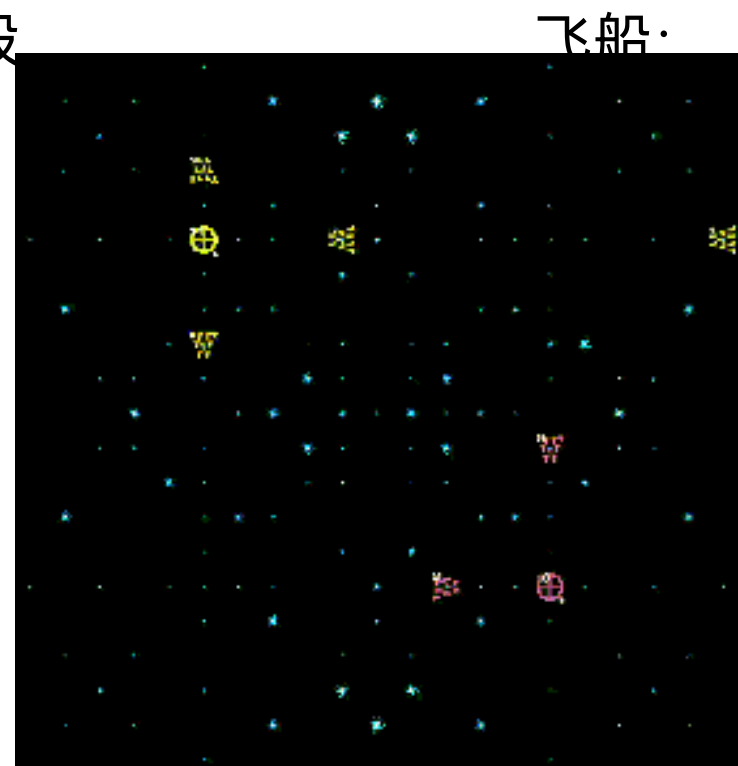
<https://www.kaggle.com/competitions/sds-ai>

- 参赛链接

<https://www.kaggle.com/t/cf9a2eec5f8c450da2d9ebcd1121d433>

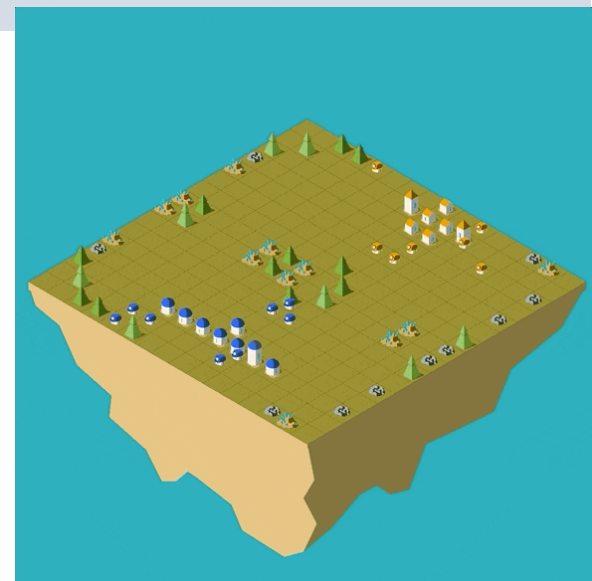
- 参考链接

<https://www.kaggle.com/competitions/kore-2022>



Games Introduction

- Game-4: 光明小镇
- 类似于Kore的竞赛规则，但是
 - 更丰富的场景设定：城镇、3种资源、2种智能体；
 - 更多的动作空间：工人移动、运输、建造；矿车移动、运输；
 - 更复杂的机制：冷却时间、日夜交替、公路等级；
 - 360轮后拥有最多城镇的一方获胜！
- 比赛主页 <https://www.kaggle.com/competitions/sds-ai-lux-ai/>
- 参赛链接 <https://www.kaggle.com/t/0ff3168e6a004b009e719df9b34665c5>
- 公开比赛参考 <https://www.kaggle.com/competitions/lux-ai-2021>
- 官方网站及详细规则 <https://www.lux-ai.org/>



温馨提示

- 每队选择一个Game进行参赛，**不同项目难度不同，最终的成绩依终期展示效果评定；
- 第二阶段必须使用基于值函数、策略函数评估的强化学习方法；
- 使用python3编程，提交100m以内的.py文件；
- 每队每天可以提交2次；
- 请勿抄袭：)

Outline

- Games Intro
- Tutorial (take Gomoku as a example)
 - ADP
 - MCTS + Heuristic
 - ADP w/ MCTS
 - AlphaGo Zero

ADP for Gomoku

- ADP 算法 (Adaptive Dynamic Programming)

利用神经网络等函数近似结构,来逼近动态规划中的性能指标函数和控制策略,并保证控制量有界;

- ADP 核心思想:

通过TD-learning, 使用非线性函数 (神经网络) 计算动作和价值函数;

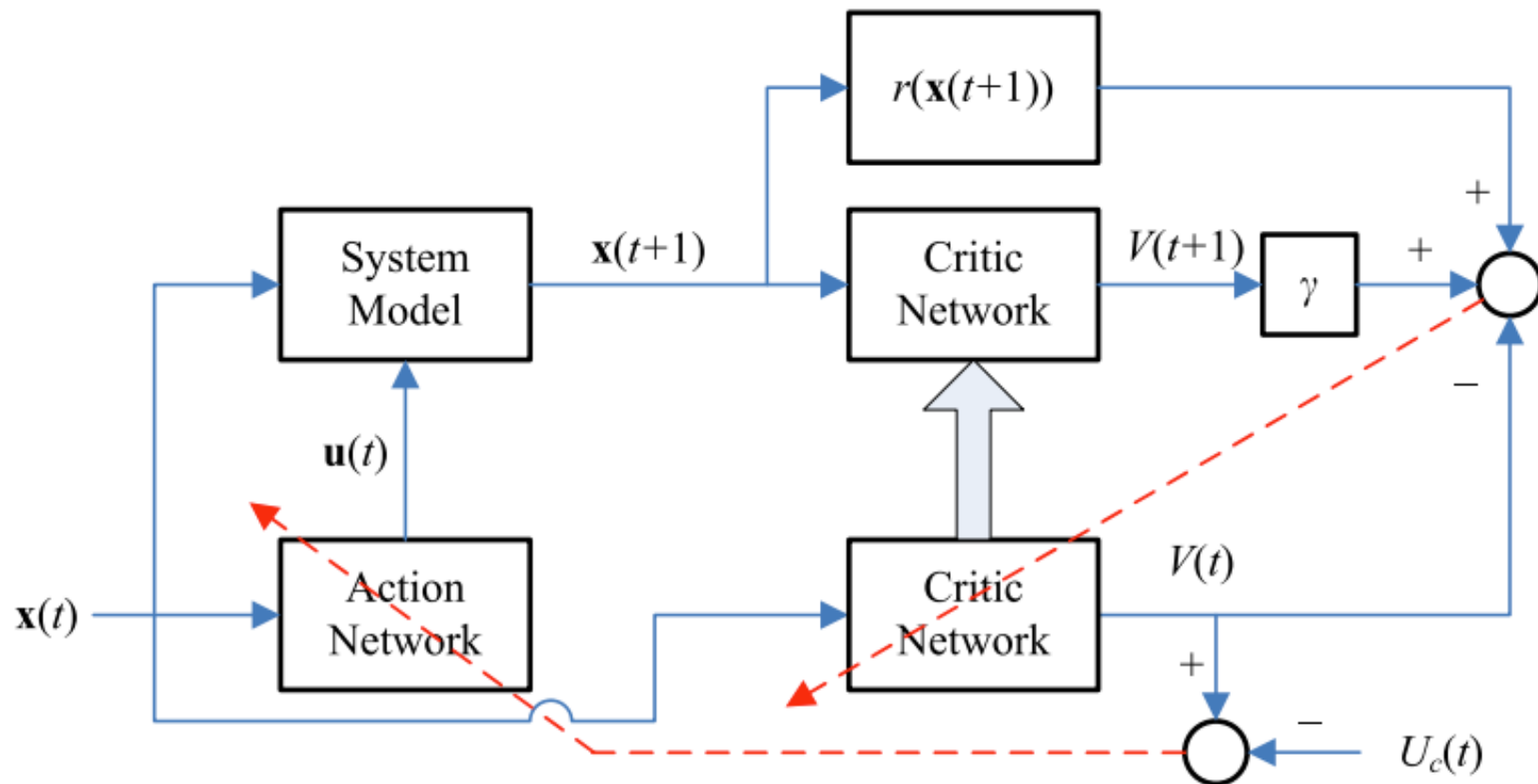
Dongbin Zhao, Zhen Zhang, and Yujie Dai.

Self-teaching adaptive dynamic programming for Gomoku.

Neurocomputing, 78(1):23–29, 2012.

ADP for Gomoku

- 最简单的ADP 结构: HDP

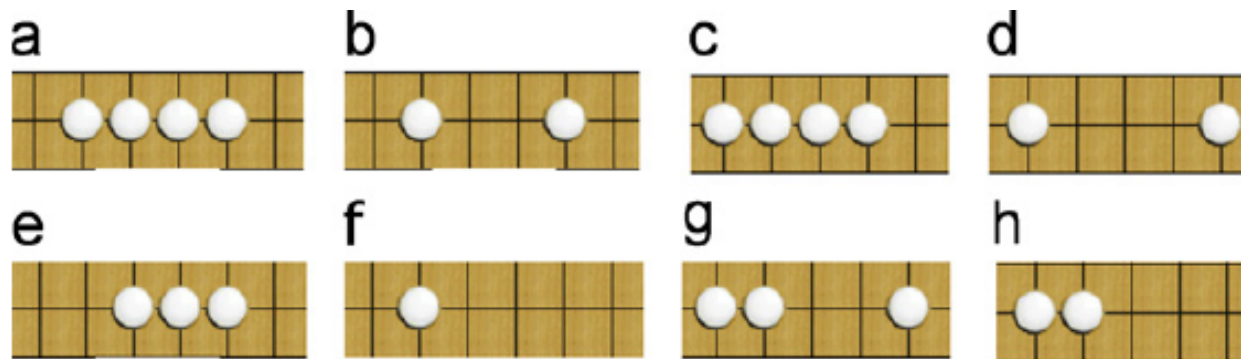


- $x(t)$: 当前状态;
- $x(t+1)$: 下一步状态;
- critic network 用于模拟价值函数 $V(t)$
- $u(t)$: 动作;
- $r(x(t+1))$: 奖励;

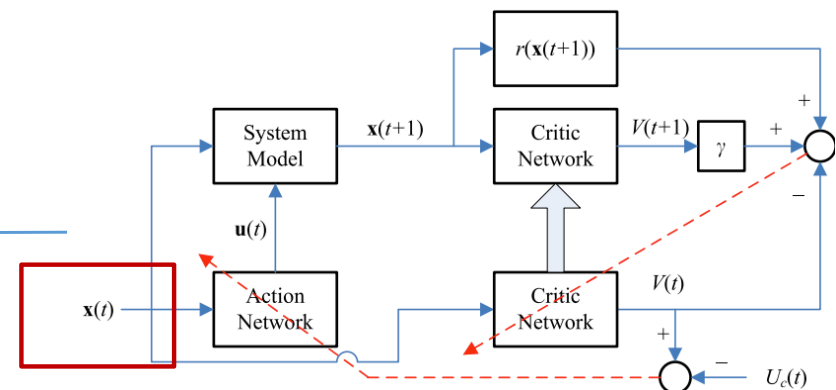
ADP for Gomoku

- 如何描述棋盘中的状态?

- 20 种模式 * 两位对局者, 一共 40 种模式



- 轮到谁行棋?
- 谁是先手?(谁走第一步)



ADP for Gomoku

- 如何描述棋盘中的状态？

- 通过五个节点描述其中一种模式出现的次数（五连除外）

(n 表示模式的数量)

Value of n	Input 1	Input 2	Input 3	Input 4	Input 5
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
> 4	1	1	1	1	$(n-4)/2$

- 五连使用一个节点表示。如果出现则为1, 否则为 0

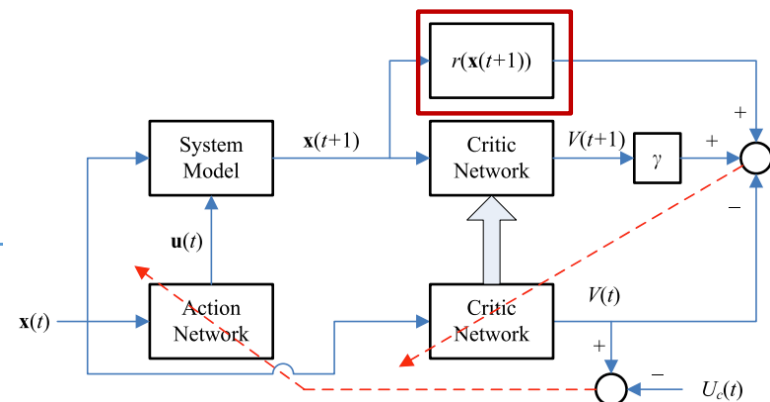
ADP for Gomoku

- 如何描述棋盘中的状态?
 - 对于每种模式我们使用两个节点来表示这个棋型属于黑棋/白棋
 - 使用两个节点来表示黑先/白先
 - 一共 $19*5*2+1*1*2+40*2+2 = 274$ 个节点

ADP for Gomoku

- 奖励函数

- 游戏进行的时候将奖励置为 0 .
- 游戏结束后, 如果玩家 1 胜出, 则奖励为 1, 如果他输了, 则奖励为 0, 平局为 0.5.



ADP for Gomoku

- ADP的评价网络 (用于模拟价值函数)

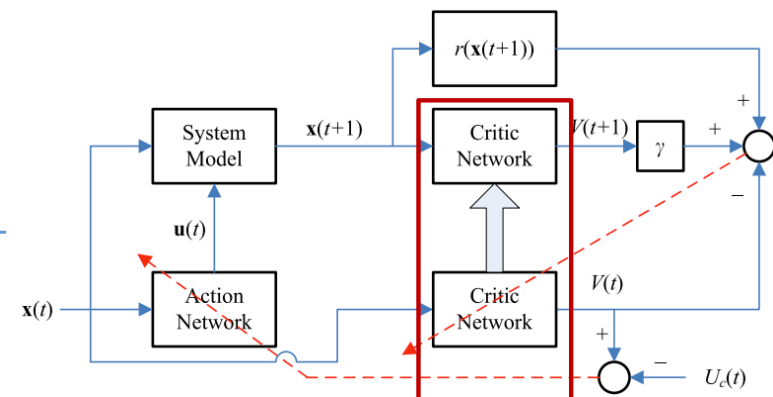
- 训练函数模拟器

- 定义预测误差

- $$e(t) = \alpha[r(t+1) + \gamma V(t+1) - V(t)]$$

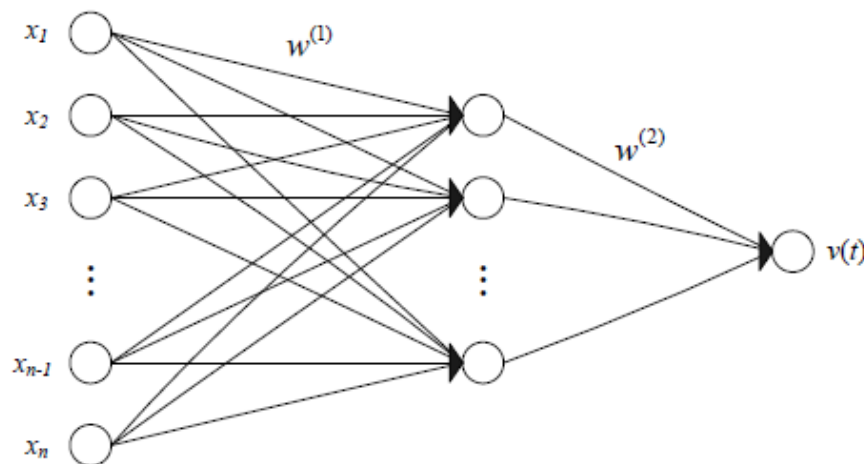
- 最小化均方差

$$E(t) = \frac{1}{2}e^2(t)$$



ADP for Gomoku

- ADP的评价网络 (用于模拟价值函数)
 - 用于评价棋盘状态(玩家1的胜率)
 - 用于输入三层全连接网络

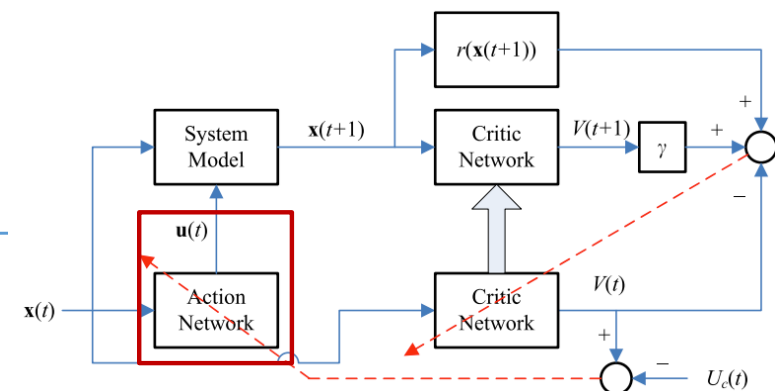


- 除了全连接网络之外也可以尝试其他函数模拟器

ADP for Gomoku

- 动作选择
- 策略

- Player 1 选择评价函数中使自己胜率最高的动作.
- Player 2 选择评价函数中使对方胜率最低的动作.



ADP for Gomoku

- 动作选择
 - 减少状态空间
 - 只考虑周围有棋子的可落子点
 - 当所有点具有相同的胜率时，选择那个最后发现的点

ADP for Gomoku

- 动作
 - 平衡最优性和可扩展性
 - 方法一：先手方第一步落子在棋盘中央，后手方第一步随机落子；
 - 方法二：使用 ϵ -greedy policy

$$a(t) = \begin{cases} \arg \max_a V(t+1) & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

ADP for Gomoku

- 自对弈
 - 自己和自己下棋
 - 使用 Psvorky 平台
 - player1 和 player2 使用相同的网络结构



ST-Gomoku

Case	Input (Turn)	Hidden	Training	Beginner	Diletante	Candidate
Case 1	274 (80)	100	60,000	30:0	22:8	13:17

ADP with MCTS for Gomoku

- 蒙特卡洛树搜索(MCTS)
 - MCTS 的基本流程
 - HMCTS
 - UCT
- ADP with MCTS

Zhentaο Tang, Dongbin Zhao, Kun Shao, and Le Lv.

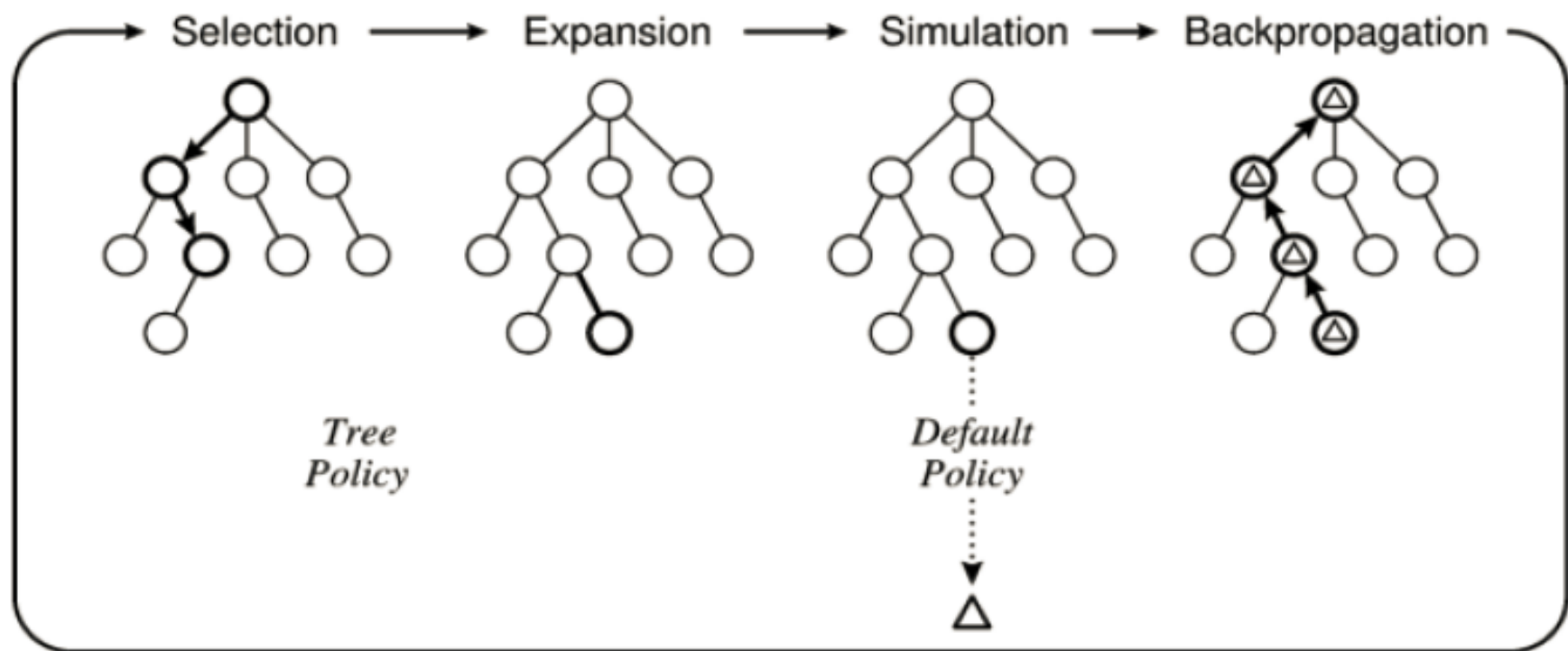
ADP with MCTS algorithm for Gomoku.

2016 IEEE Symp. Ser. Comput. Intell. SSCI 2016, (61273136), 2017.

MCTS

- 需要大量的模拟，并根据结果建立一个大的搜索树
- 随着模拟时间和访问节点的增加，估计值将更加准确。

The Basic process of MCTS



HMCTS

- 启发式蒙特卡罗树搜索
- 把启发式知识应用于策略模拟中

HMCTS

- 启发式知识
 - 如果我方出现了四连，则直接完成一个五连
 - 如果对手出现了四连，则封堵它可能出现五连的位置
 - 如果我方出现了三连，则直接完成一个四连
 - 如果对手出现了三连，则封堵它可能出现四连的位置

HMCTS

- 比随机抽样更节省时间，并且可以更早收敛；
- Total reward: Q-value 函数

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{N(s)} l_i(s, a) z_i$$

Algorithm 1: HMCTS for Gomoku

```
input original state  $s_0$ ;  
output action  $a$  corresponding to the highest value of MCTS;  
add Heuristic Knowledge;  
obtain possible action moves  $M$  from state  $s_0$ ;  
for each move  $m$  in moves  $M$  do  
    reward  $r_{total} \leftarrow 0$ ;  
    while simulation times < assigned times do  
        reward  $r \leftarrow \text{Simulation}(s(m))$ ;  
         $r_{total} \leftarrow r_{total} + r$ ;  
        simulation times add one;  
    end while  
    add  $(m, r_{total})$  into  $data$ ;  
end for each  
return action Best( $data$ )
```

```
Simulation(state  $s_t$ )  
    if ( $s_t$  is win and  $s_t$  is terminal) then return 1.0;  
    else return 0.0;  
end if  
    if ( $s_t$  satisfied with Heuristic Knowledge)  
        then obtain forced action  $a_f$ ;  
        new state  $s_{t+1} \leftarrow f(s_t, a_f)$ ;  
        else choose random action  $a_r \in$  untried actions;  
        new state  $s_{t+1} \leftarrow f(s_t, a_r)$ ;  
    end if  
    return Simulation( $s_{t+1}$ )
```

```
Best( $data$ )  
    return action  $a$  //the maximum  $r_{total}$  of  $m$  from  $data$ 
```

UCT

- 搜索树的置信度上限

- 基于 Upper Confidence Bounds(UCB) 确定：

$$\frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v)}}$$

- $\frac{Q(v')}{N(v')}$ is the 是节点 v' 的平均价值, $N(v')$ 和 $N(v)$ 是 v' 和 v 的探索数量, v 是 v' 的父节点；
 - 第一项期望更高的回报选择（exploitation），第二项期望访问更少的新节点（exploration）→ 平衡探索与利用之间的冲突，提早发现模型的最终结果；
 - *能够解决多臂老虎机问题

UCT

Algorithm 2: UCT for Gomoku

input create root node v_0 with state s_0 ;
output action a corresponding to the highest value of UCT;
while within computational budget **do**
 $v_l \leftarrow \text{Tree Policy}(v_0)$;
 Policy \leftarrow Heuristic Knowledge;
 reward $r \leftarrow \text{Policy}(s(v_l))$;
 Back Update(v_l, r);
end while
return action $a(\text{Best Child}(v_0))$

Tree Policy(node v)

while v is not in terminal state **do**
 if v not fully expanded **then** **return** Expand(v);
 else $v \leftarrow \text{Best Child}(v, 1/\sqrt{2})$;
 end if
end while
return v //this is the best child node

Expand(node v)

choose random action $a \in$ untried actions from $A(s(v))$;
add a new child v' to v
 with $s(v') \leftarrow f(s(v), a)$ and $a(v') \leftarrow a$;
return v' //this is the expand node

Best Child(node v , parameter c)

return $\arg \max_{v' \in \text{child}} ((Q(v') / N(v')) + c\sqrt{2 \ln N(v) / N(v')})$

Policy(state s)

while s is not terminal **do**
 if s satisfied with heuristic knowledge **then**
 obtain forced action a ;
 else choose random action $a \in A(s)$ uniformly;
 end if
 $s \leftarrow f(s, a)$;
end while
return reward for state s

Back Update(node v , reward r)

while v is not null **do**
 $N(v) \leftarrow N(v) + 1$;
 $Q(v) \leftarrow Q(v) + r$;
 $v \leftarrow \text{parent of } v$;
end while

MCTS

- UCT 和 HMCTS 的对比
 - UCT 源于 HMCTS.
 - 可以更早的发现合适的子节点.
 - UCT 相较于 HMCTS 能节省更多时间.

ADP with MCTS

- 使用 ADP 训练评价网络, 得到 top-5 可行点和他们的 ADP 胜率 w_1 ;
- 把这些可行点作为 MCTS 的根节点进行模拟, 得到他们的 MCTS 胜率 w_2 ;
- 将两者胜率综合:
$$w_p = \lambda w_1 + (1 - \lambda) w_2$$

ADP with MCTS

- ADP: ST-Gomoku

Algorithm 3: ADP with MCTS

input original state s_0 ;
output action a correspond to ADP with MCTS;
 $M_{ADP}, W_{ADP} \leftarrow \text{ADP Stage}(s_0)$;
 $W_{MCTS} \leftarrow \text{MCTS Stage}(M_{ADP})$;
for each w_1, w_2 in pairs(W_{ADP}, W_{MCTS}) **do**
 $w_p \leftarrow \lambda w_1 + (1-\lambda)w_2$;
 add p into P ;
end for each
return action a correspond to max p in P

ADP Stage(state s)

 obtain top 5 winning probability W_{ADP} from ADP(s) ;
 obtain their moves M_{ADP} correspond to W_{ADP} ;
 return M_{ADP}, W_{ADP}

MCTS Stage(moves M_{ADP})

for each move m in M_{ADP} **do**
 create m as root node with correspond state s
 obtain w_2 from MCTS(m, s)
 add w_2 into W_{MCTS}
 end for each
 return W_{MCTS}

ADP with MCTS

- 相较于 ADP :
 - 减少了神经网络的“短视性”,确保了搜索的准确性
- 相较于 MCTS :
 - 在搜索合适的落子点上节省了很多时间

Other Heuristic Functions


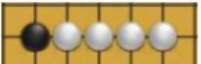
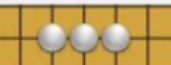
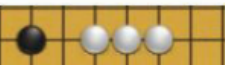

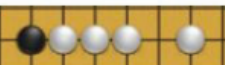
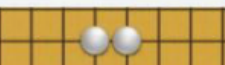
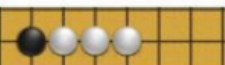


ID(Type)	Pattern	Value
1		10000
2		1000
3		1000
4		1000 * factor
5		1000 * factor
6		1000 * factor
7		100
8		100
9		100
10		100 * factor

Fig. 3. Example of the patterns and their heuristic value.

$$H_i = \sum \{10^{L_{open}} * factor^j + 10^{L_{hclose}-1} * factor^k\} \quad (3)$$

$$Factor^{j,k} = 0.9$$

$$UCB = v_i + k_1 * \sqrt{\frac{\ln(N)}{n_i}} + k_2 * \frac{H_i}{\max(H)}$$

Xu Cao and Yanghao Lin.

UCT-ADP Progressive Bias Algorithm for Solving Gomoku.

2019 IEEE Symposium Series on Computational Intelligence (SSCI).

ADP with MCTS

- Experimental Results

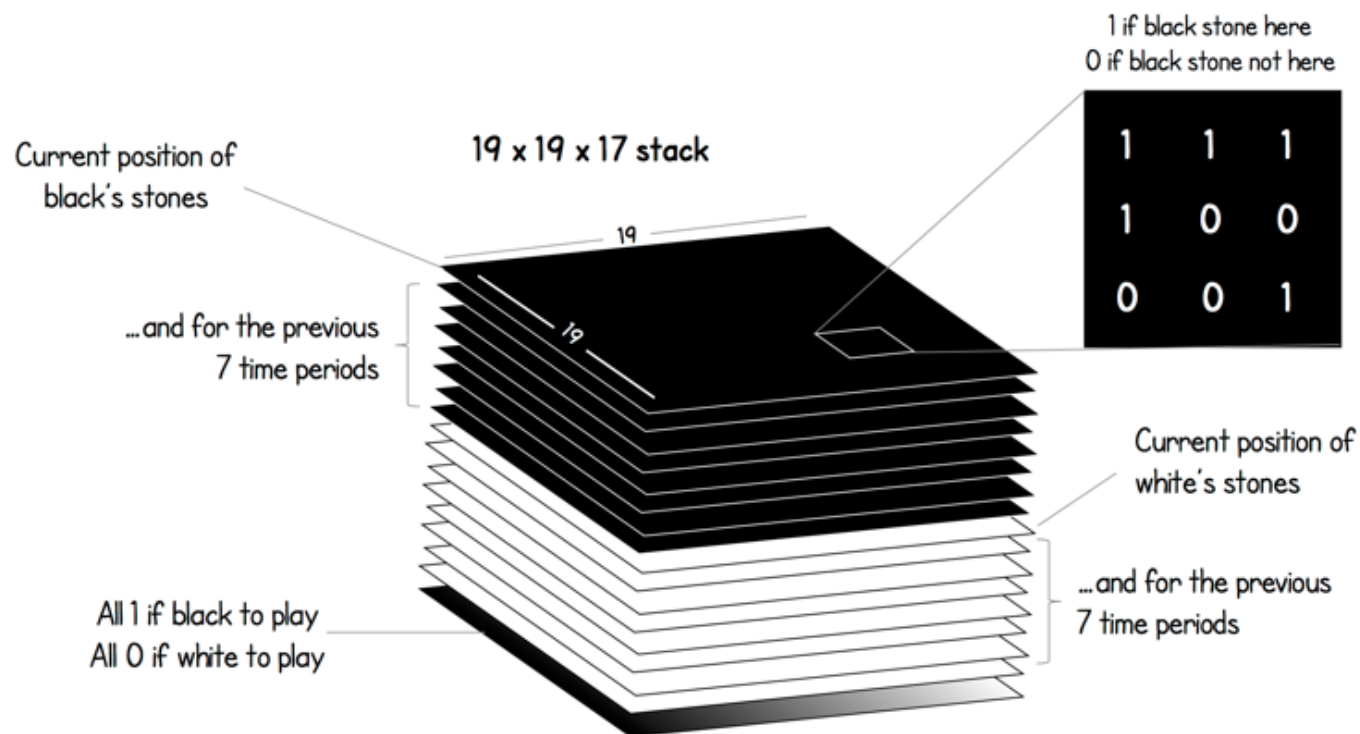
TABLE IV. COMPARISON AGAINST 5-STAR GOMOKU

Algorithm	Gomoku Level		
	Beginner	Dilettante	Candidate
ADP	100:0	73:27	43:57
HMCTS	46:54	13:87	0:100
ADP-HMCTS	100:0	89:11	71:29
ADP-UCT	100:0	82:18	64:36

Alpha-Zero Solution

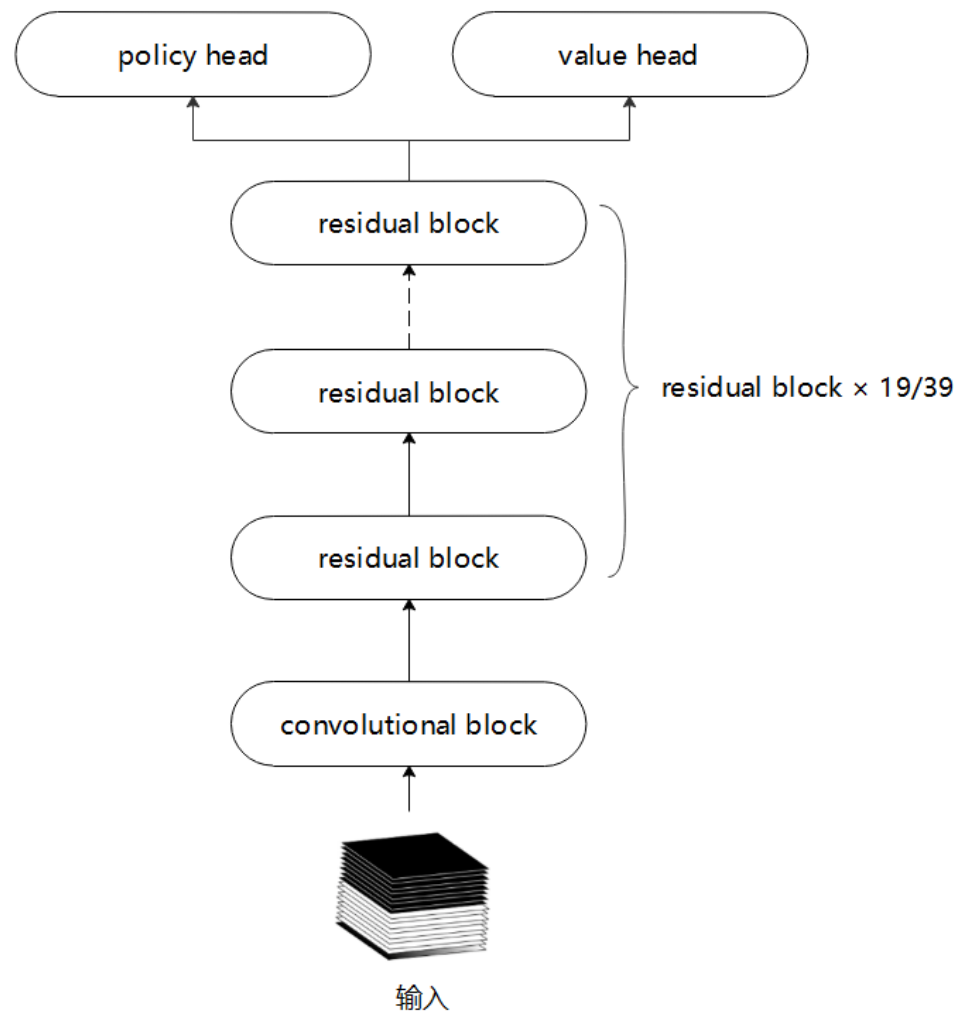
Alpha-Zero Solution

- 方法
- 特征提取



Alpha-Zero Solution

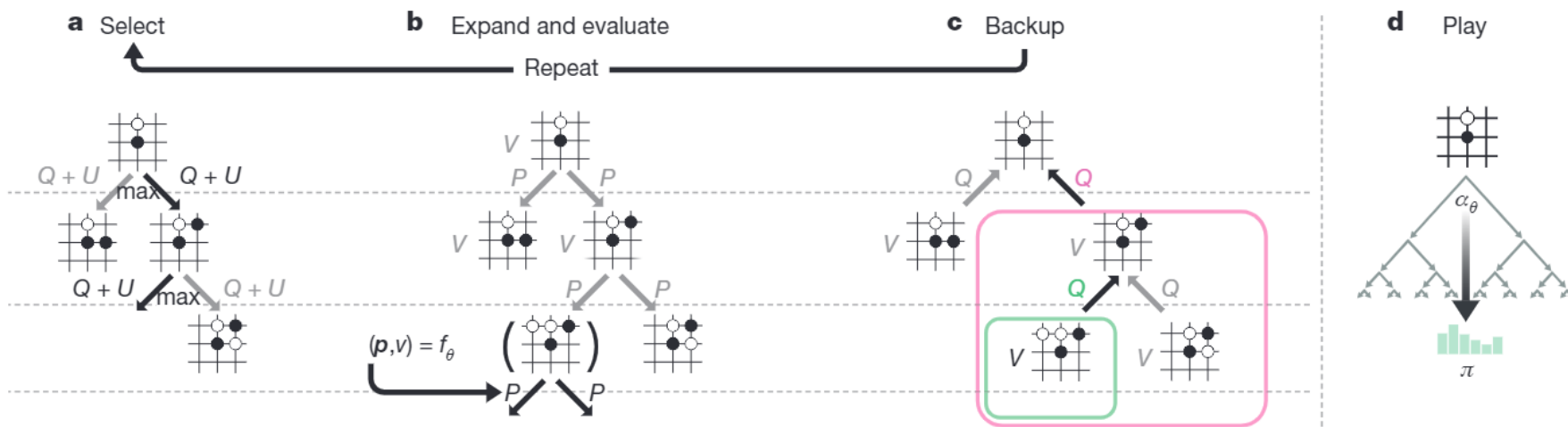
- 方法
 - 特征提取
 - Policy & Value Head



Alpha-Zero Solution

- 训练

- MCTS 生成

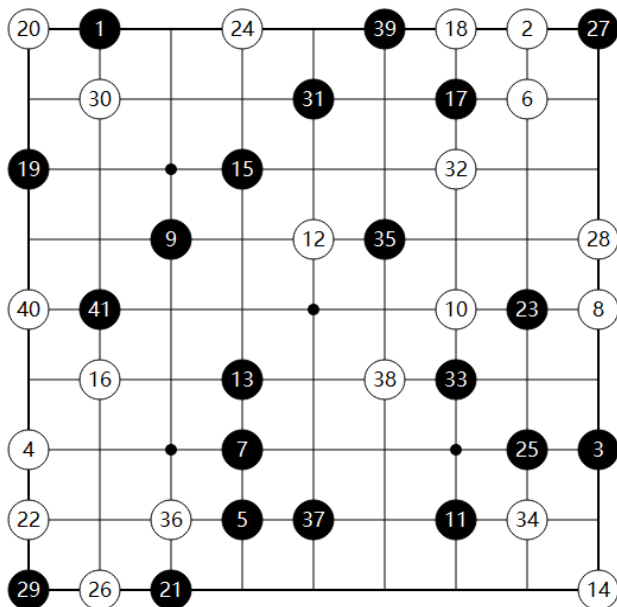


- 深度学习训练

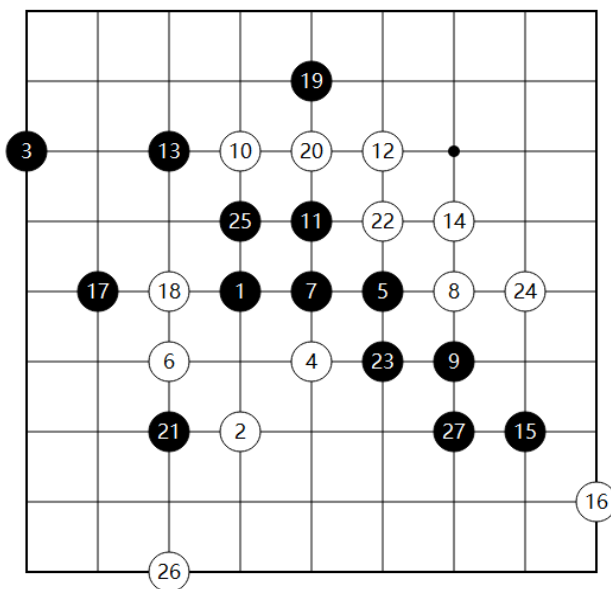
Alpha-Zero Solution

- 结果

Iteration 0



Iteration 800



Iteration 4400

