《物理与人工智能》

17. 蒙特卡洛树搜索

授课教师：马滟青

2025/10/27（第七周）

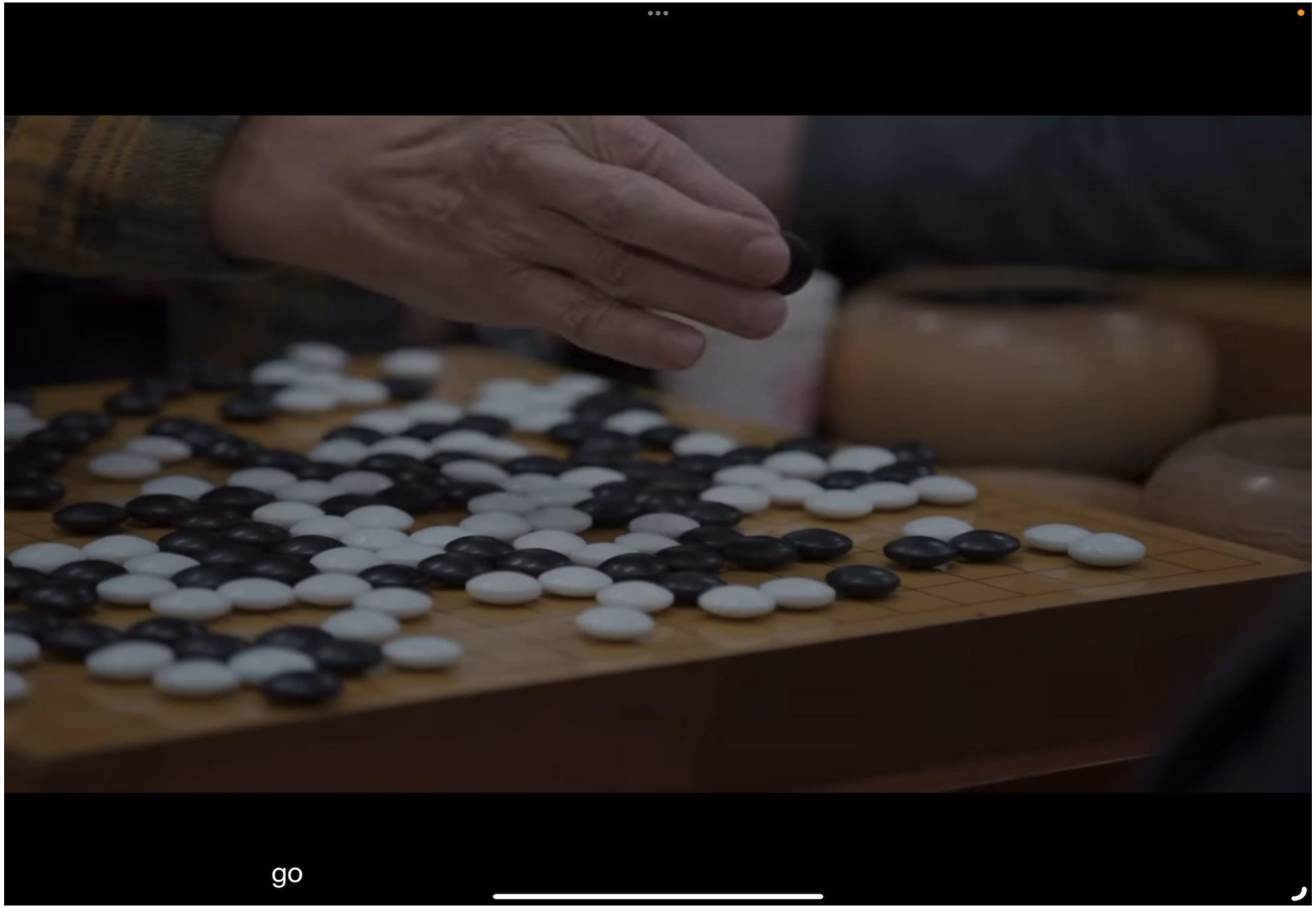
鸣谢：基于计算机学院《人工智能引论》课程组幻灯片

北京大學

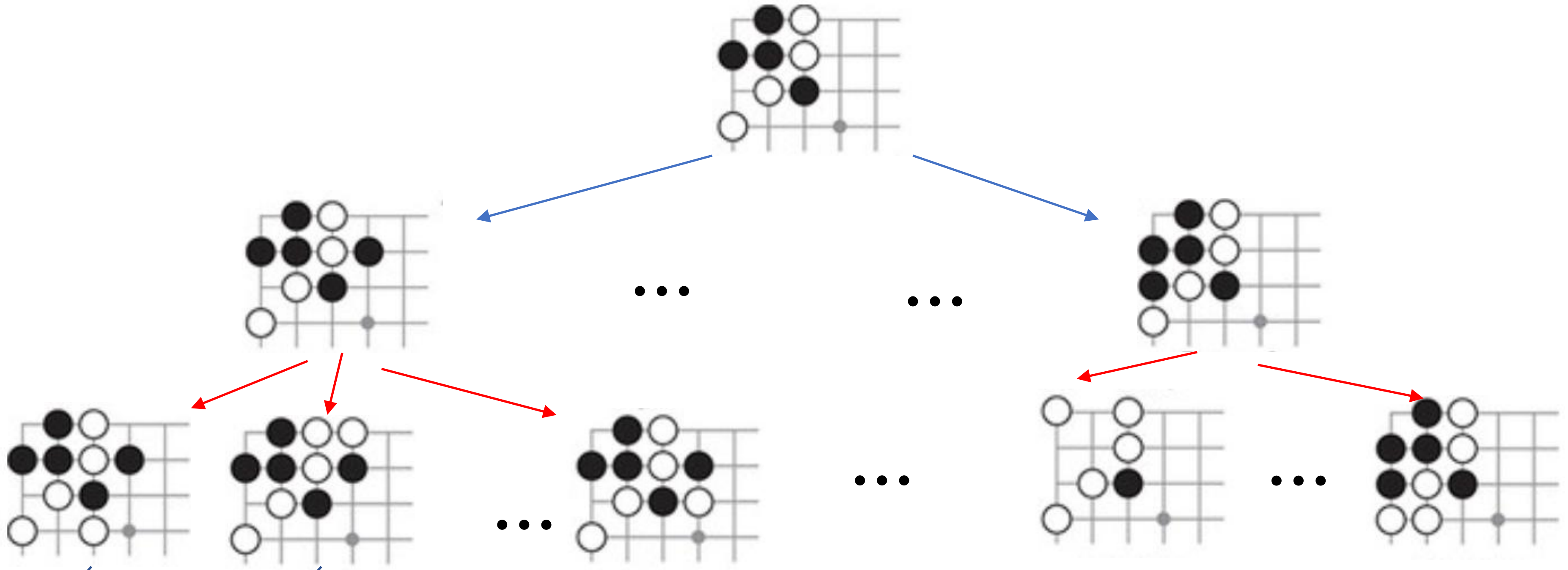
目录

- **如何估值?**
  - 蒙特卡洛方法
- **如何平衡不确定性?**
  - 上置信界（Upper Confidence Bound）
- **蒙特卡洛搜索**

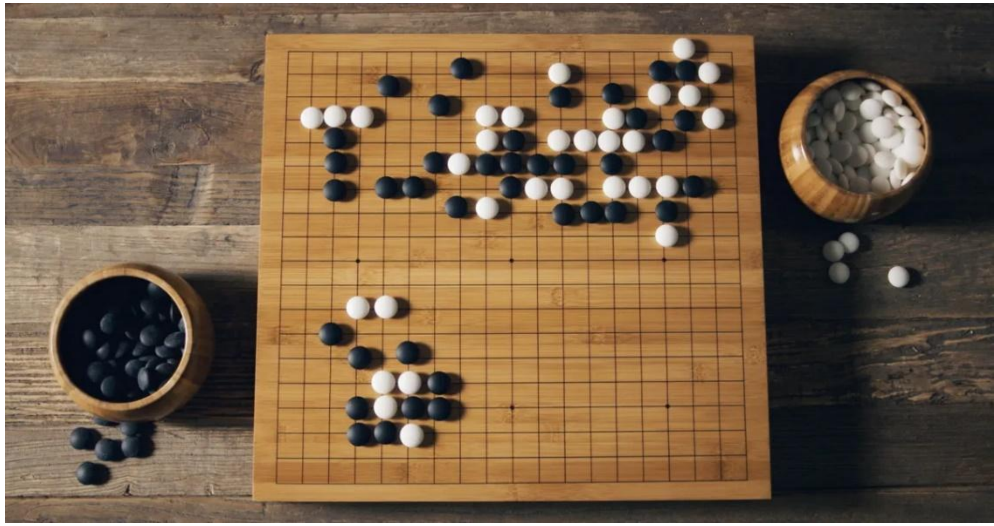
AlphaGo



Recap：围棋的博弈树

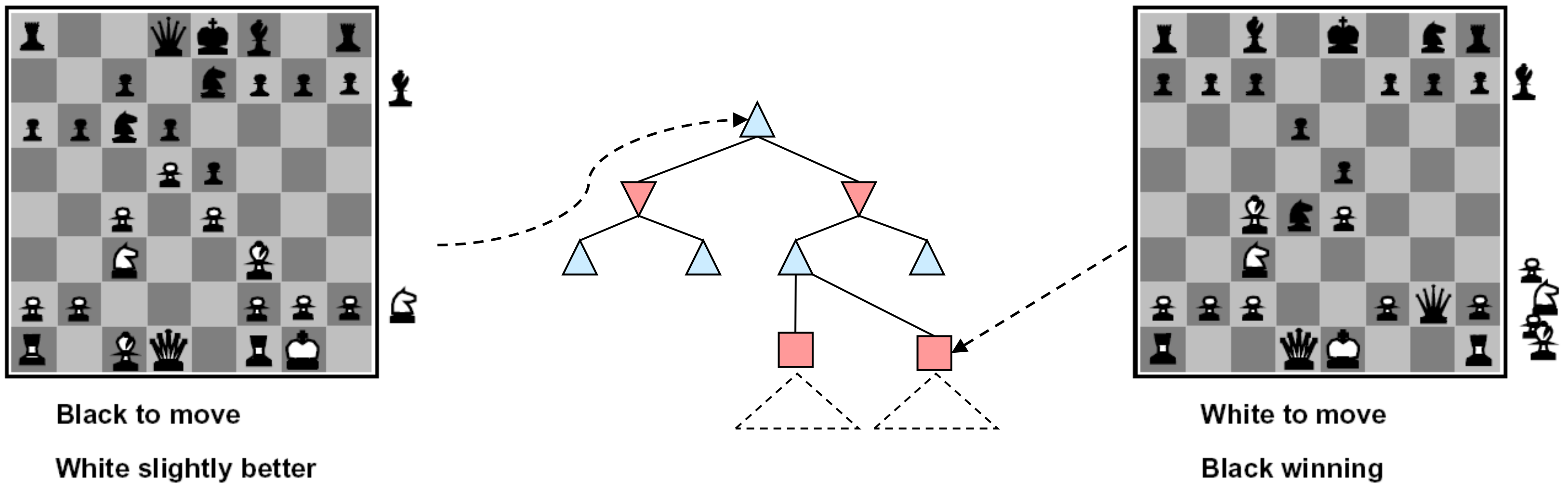


提前结束?



Recap：估值函数

- 估值函数为非终止状态打分：



- **理想的估值函数**：返回当前局面的实际极大极小值
- **实际的估值函数**：通常是特征的加权线性和

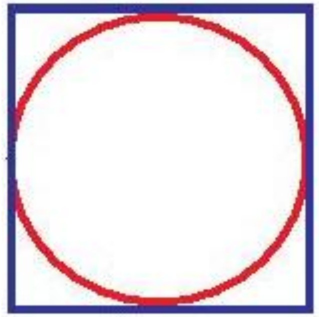
$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

- e.g. $f_1(s)$ = (num white queens – num black queens).
- 如何构建**更好的估值函数**：机器学习！

蒙特卡洛方法

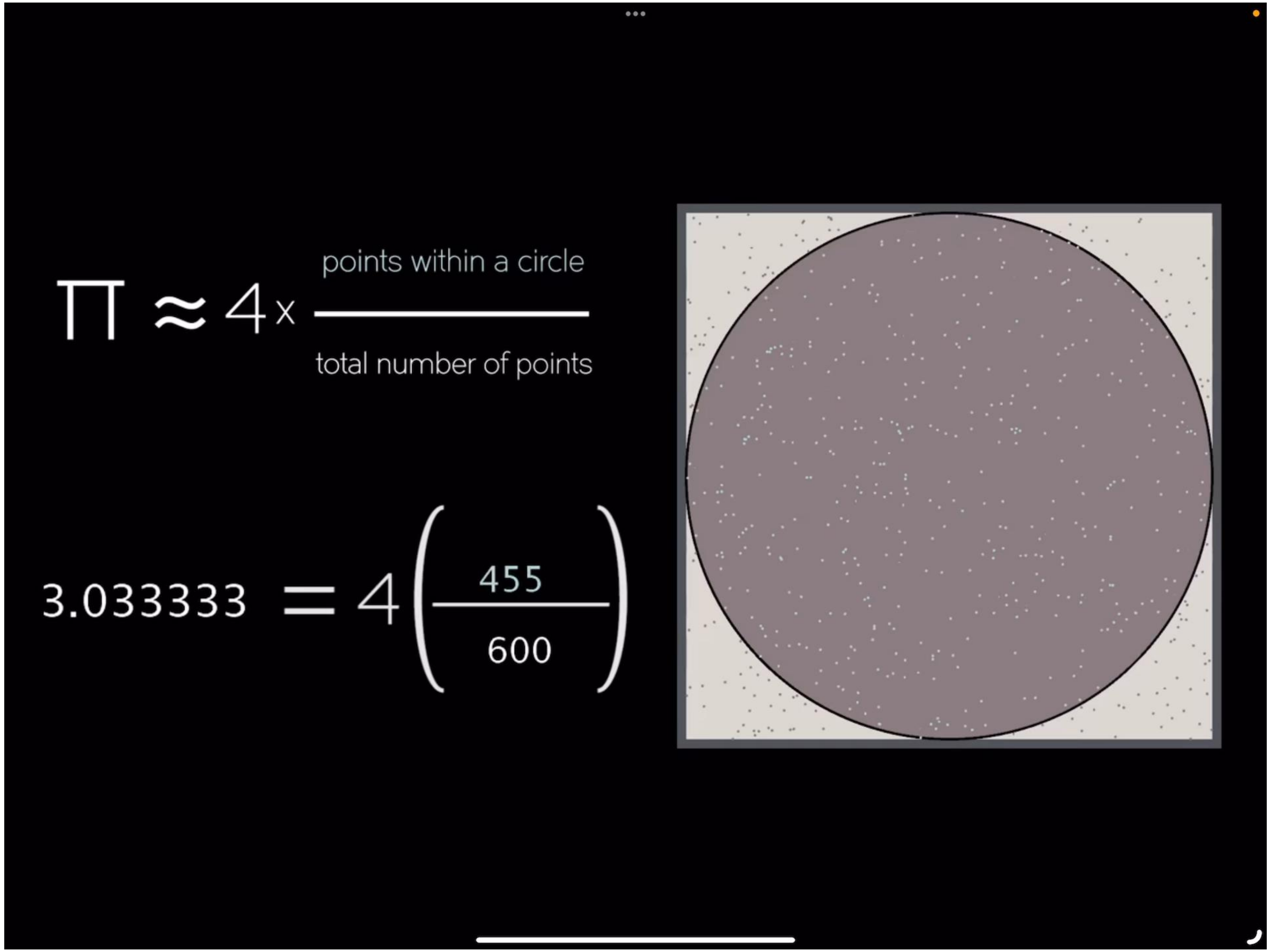
- 利用（大规模）随机抽样来近似问题的解
- 20世纪40年代，一群从事核弹制造的科学家以蒙特卡罗赌场命名
  - John von Neumann, Stanislaw Ulam and Nicholas Metropolis

蒙特卡洛方法：计算圆周率



- 圆的面积 / 正方形的面积 = π/4
- 如果随机往正方形里扔飞镖，有 π/4 概率落在圆内
- 反向思考，就随机往正方形里扔飞镖，统计有多少次落在圆内，多少次落在圆外

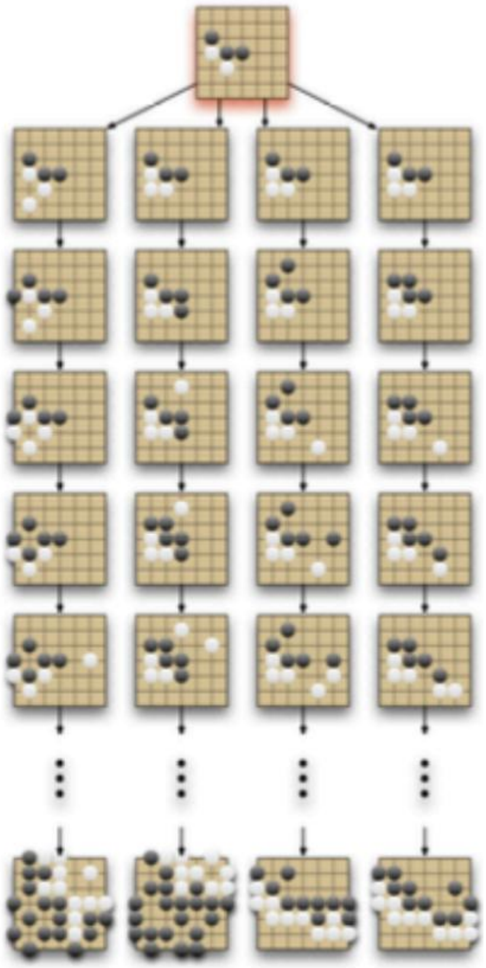
蒙特卡洛方法：计算圆周率



蒙特卡洛方法：性质

- 次数越多越准确（你会更相信上一页实验一开始的估计还是最后的估计？）
- 估计的准确与样本方差有关

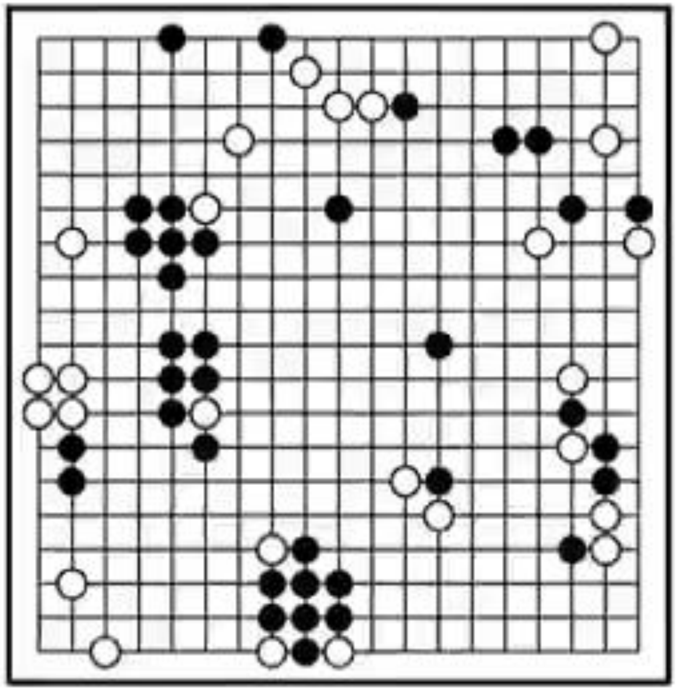
蒙特卡洛方法：围棋



1　1　0　0

- 在当前状态，我们的估值为
  2 / 4 = 0.5

真实的围棋



- 我们真的可以大规模抽样吗?
- 其实在每一个状态，只有很少的几个动作是高价值的

如何平衡抽样的不确定性：老虎机

$\mathbb{E}[a] = -\$0.5$

$\mathbb{E}[b] = -\$0.2$

$\mathbb{E}[c] = \$0.1$

$\mathbb{E}[d] = \$0.11$

如何平衡抽样的不确定性：老虎机

$\mathbb{E}[a] = ?$

$\mathbb{E}[b] = ?$

$\mathbb{E}[c] = ?$

$\mathbb{E}[d] = ?$

如何平衡抽样的不确定性：老虎机

北京大学 PEKING UNIVERSITY

-1, -1, 5

$\hat{\mathbb{E}}[a] = 1$

-0.2, -0.2

$\hat{\mathbb{E}}[b] = -0.2$

-0.5, -0.5, -0.5

$\hat{\mathbb{E}}[c] = -0.5$

-2, -2

$\hat{\mathbb{E}}[d] = -2$

如何平衡抽样的不确定性：老虎机

-1, -1, 5, -1, -1, -1, -1, -1, 2, 6, -1, -1, -1, -1, -1

$\mathbb{E}[a] = -\$0.5$

-0.2, -0.2

$\mathbb{E}[b] = -\$0.2$

-0.5, -0.5, -0.5

$\mathbb{E}[c] = \$0.1$

-2, -2

$\mathbb{E}[d] = \$0.11$

如何平衡抽样的不确定性：老虎机

北京大学 PEKING UNIVERSITY

- k：动作的数量（比如，有多少老虎机）
- $q^*(a)$：动作 a 的真实价值
- $N_t(a)$：在时间 t 之前，动作 a 被选中的次数
- $Q_t(a)$：在时间 t 的时候，对动作 a 的估值
- R_t：在时间 t 得到的价值
- A_t：在时间 t 采取的动作

如何平衡抽样的不确定性：老虎机

$k=4$

$t=11$

$R = \{-1, -1, 5\}$

$N_{11}(a) = 3$

$q_*(a) = -\$0.5$

$Q_{11}(a) = 1$

$R = \{-0.2, -0.2\}$

$N_{11}(b) = 2$

$q_*(b) = -\$0.2$

$Q_{11}(b) = -0.2$

$R = \{-0.5, -0.5, -0.5\}$

$N_{11}(c) = 3$

$q_*(c) = \$0.1$

$Q_{11}(c) = -0.5$

$R = \{-2, -2\}$

$N_{11}(d) = 2$

$q_*(d) = \$0.11$

$Q_{11}(d) = -2$

如何平衡抽样的不确定性：老虎机

- 两种动作之一：剥削，选取现在最优的

-1, -1, 5

$Q_{11}(a) = 1$

-0.2, -0.2

$Q_{11}(b) = -0.2$

-0.5, -0.5, -0.5

$Q_{11}(c) = -0.5$

-2, -2,

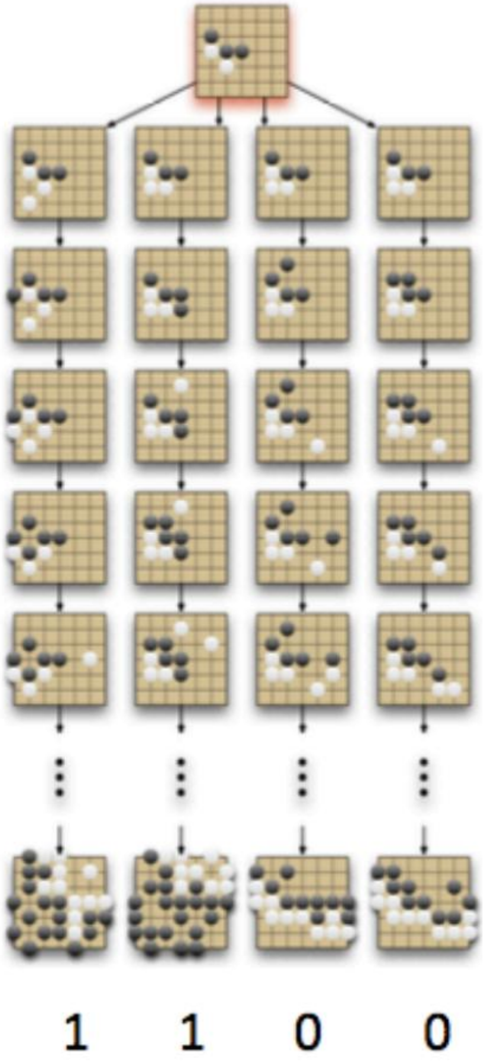
$Q_{11}(d) = -2$

如何平衡抽样的不确定性：老虎机

- 矛盾点：既想对最优的有一个精确的估计，又怕自己判断错

如何平衡不确定性：ε-greedy

- 我们现在的估值: $Q_{t+1}(a) = \begin{cases} \frac{\sum_{t:A_t=a} r_t}{N_t(a)}, & N_t(a) > 0 \\ 0 & , N_t(a) = 0 \end{cases}$



如何平衡不确定性：ε-greedy

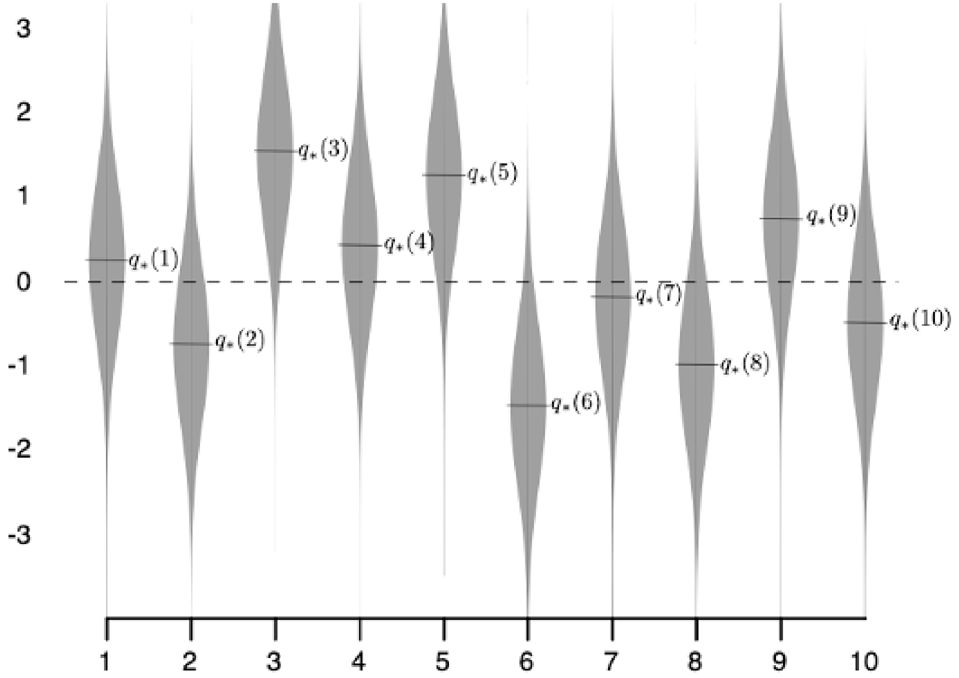
- 我们现在的估值: $Q_{t+1}(a) = \begin{cases} \frac{\sum_{t:A_t=a} r_t}{N_t(a)}, & N_t(a) > 0 \\ 0, & N_t(a) = 0 \end{cases}$

- 1- ε的概率：选当前最优的
- ε的概率：随机选取一个当前不是最优的

如何平衡不确定性：ε-greedy

- $k = 10$
- $A = \{1, \dots, 10\}$
- $\Pr\{r|a\} \sim \mathcal{N}(q_*(a), 1)$

- 最优应该是选取动作3
- 但我们一开始不知道

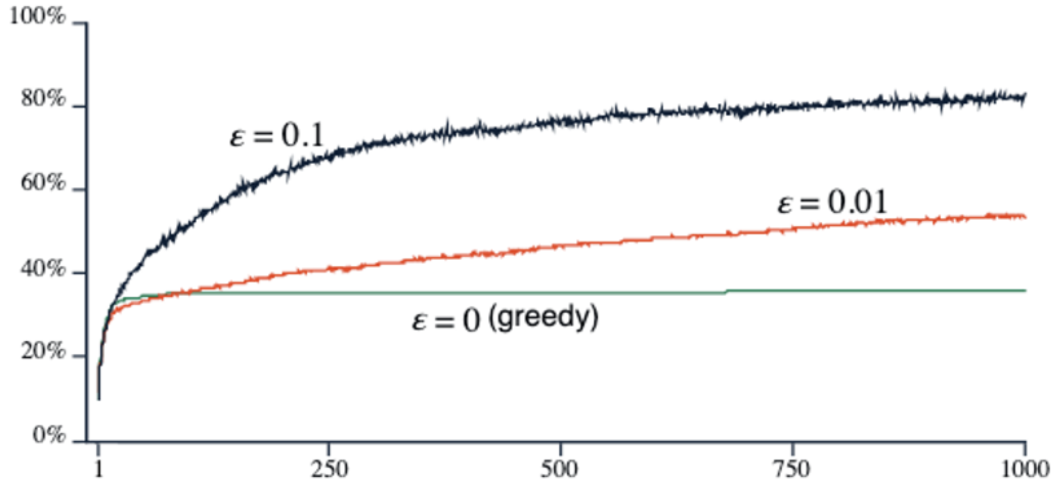


如何平衡不确定性：ε-greedy

- 三种不同的ε
  - $\varepsilon = 0$
  - $\varepsilon = 0.01$
  - $\varepsilon = 0.1$

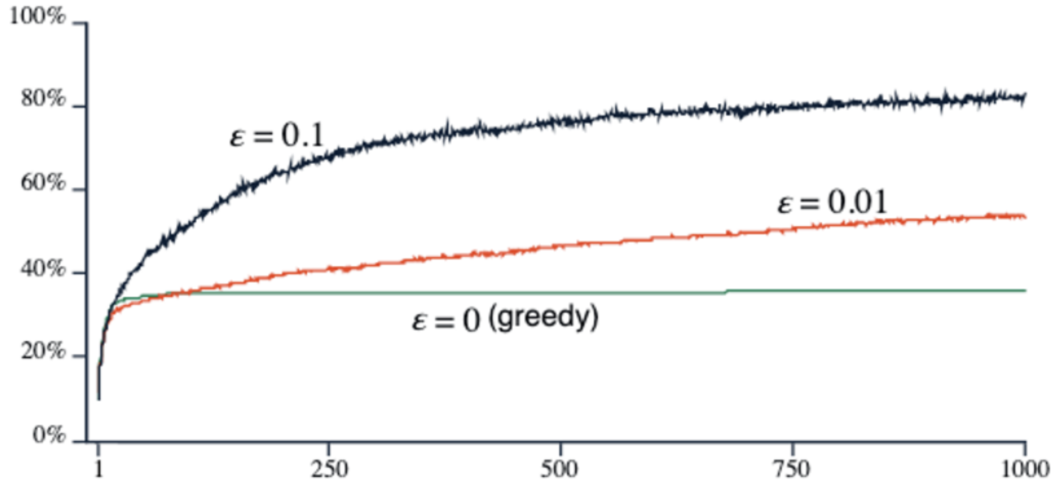
- 每次实验采取动作1000次，共跑2000次实验。

- 选到最优动作的概率?



如何平衡不确定性：ε-greedy

- 探索是永远需要的，因为我们的估计永远有不确定性
- 所谓的剥削只是针对当前看起来最优的，别的动作有可能更好
- ε-greedy强制不是看起来最优的也会被尝试到，但对所有别的动作都一视同仁
- 如果我们可以更多地去试那些更有希望成为更优的，那整个算法的效率更高。如何实现？



如何平衡不确定性：上置信界（Upper Confidence Bound）

- 已知：次数越多越准确，换言之我们对次数少的估计信任不足
- 那我们的算法应该平衡，估值的大小，和次数的多少

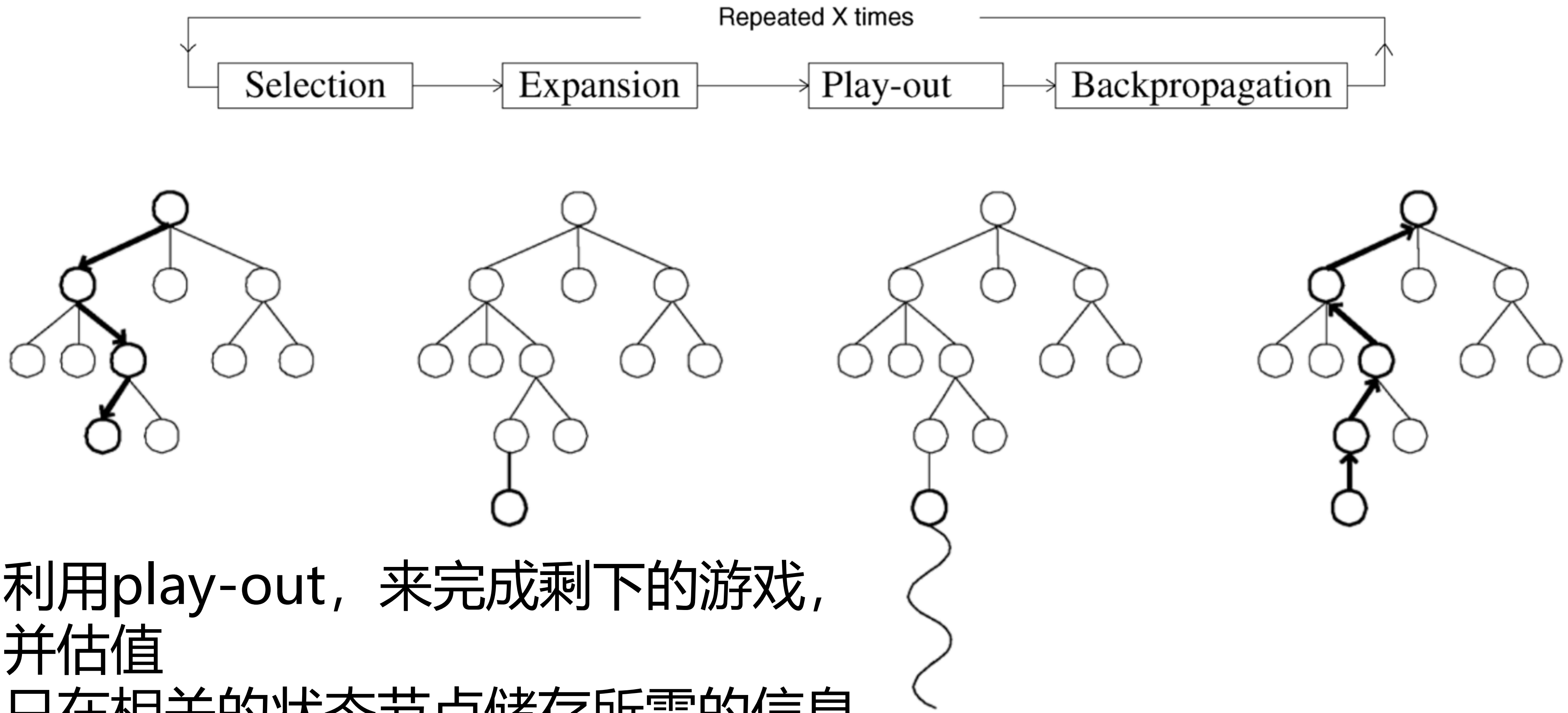
- $A_{t+1} = \text{argmax}_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$

  - ✓ $Q_t(a)$是选项截至到时间的平均收益。
  - ✓ c是一个常数，用于控制探索的程度，通常根据问题的具体情况进行调整。
  - ✓ t是当前的时间步数。
  - ✓ $N_t(a)$是选项在时间之前被选择的次数。
  - ✓ argmax_a表示使函数取最大值对应的参数a 。

从UCB到博弈（单次动作到序列动作）

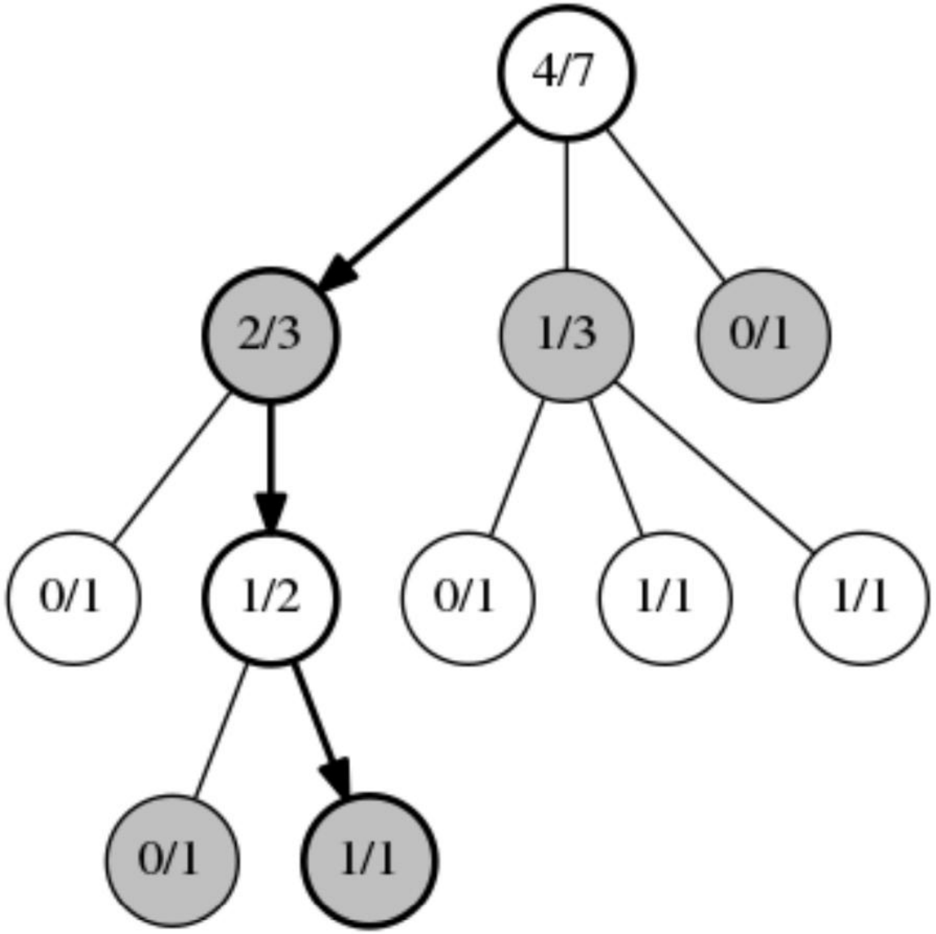
- 构建向前看的博弈树
- 通过随机模拟，来产生节点的估值函数
- 在博弈树的中间节点，根据UCB来选择动作

从UCB到博弈（单次动作到序列动作）



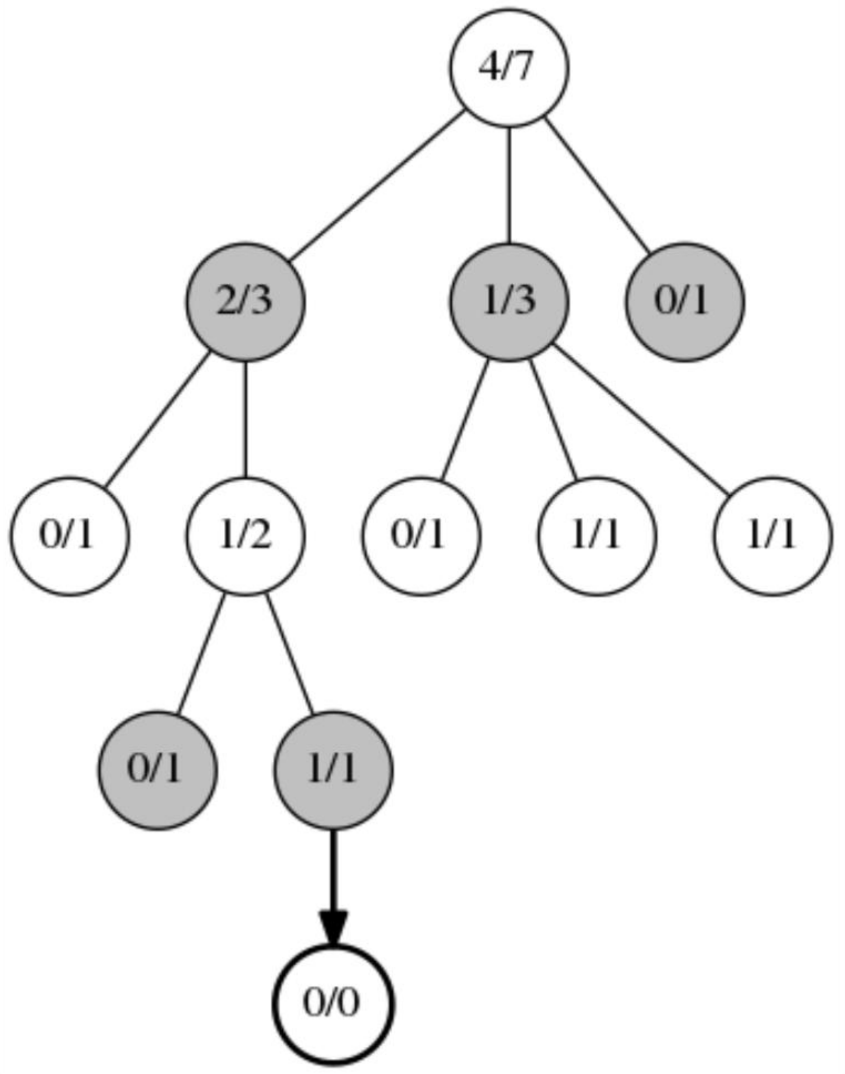
- 利用play-out，来完成剩下的游戏，并估值
- 只在相关的状态节点储存所需的信息

从UCB到博弈：选择（selection）



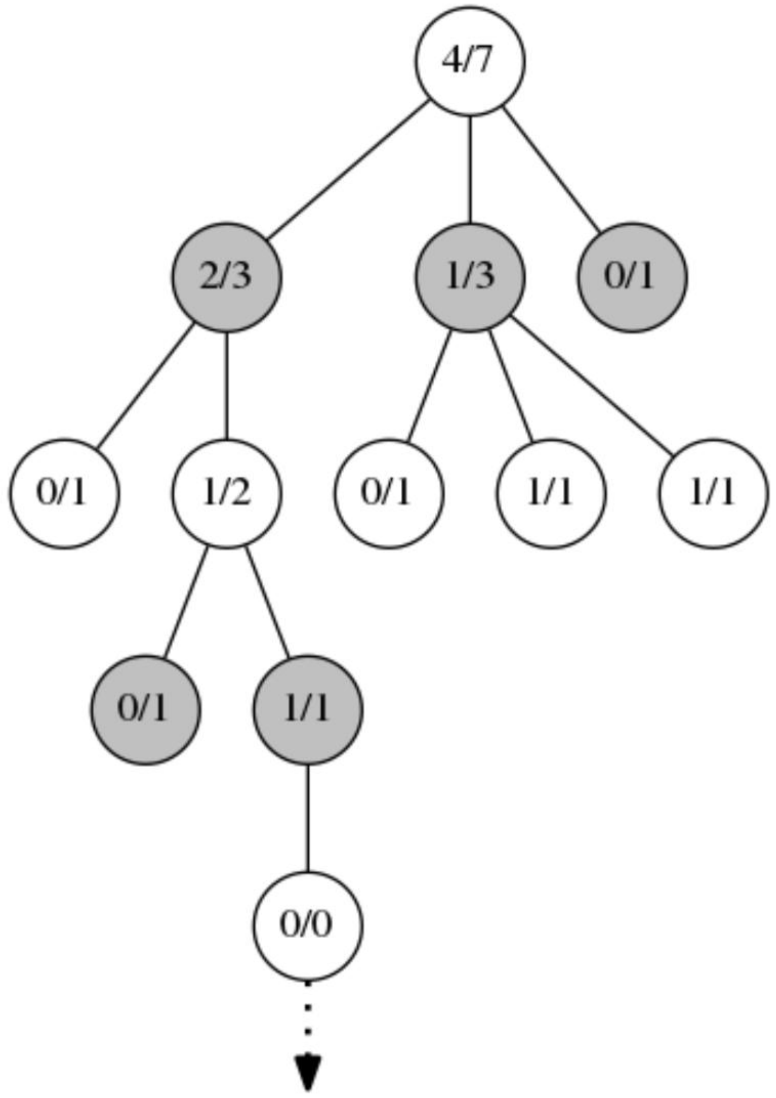
- 每个步骤中UCB算法选择的动作都用粗体标记。
- 已经运行了大量模拟来积累所显示的统计信息。
- 每个圆圈都包含获胜次数 / 播放次数。

从UCB到博弈：扩展（expansion）



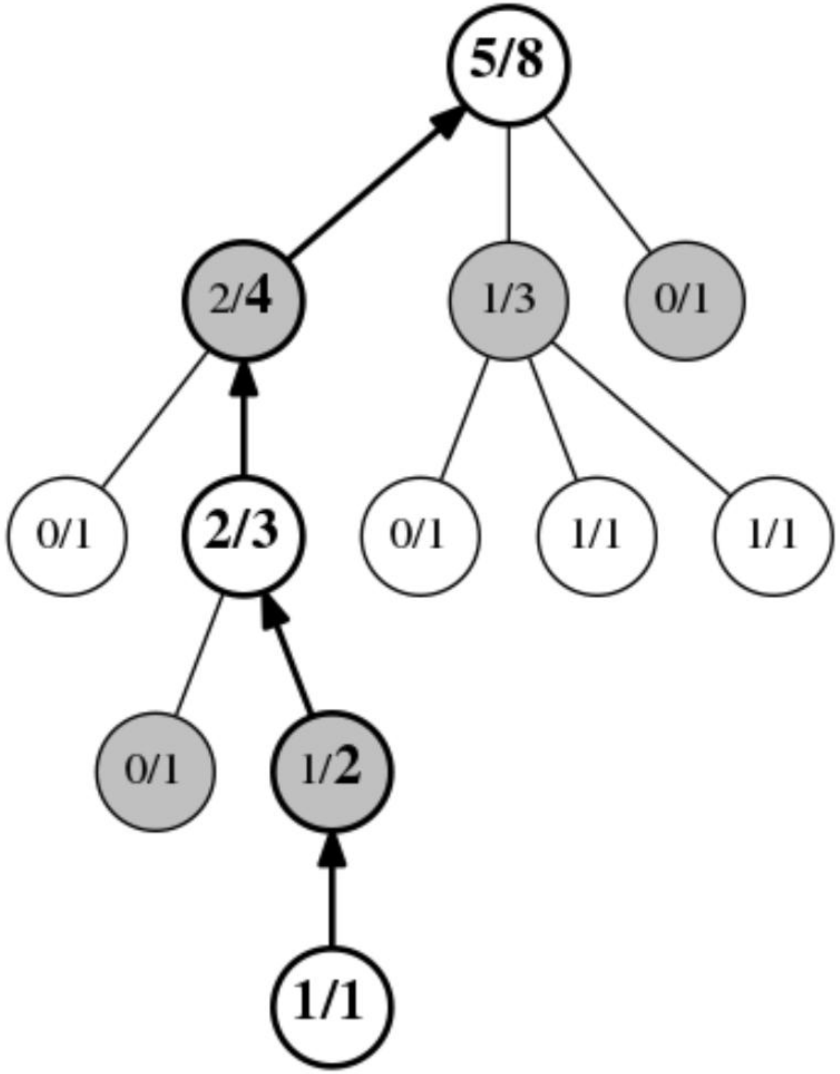
- 搜索的深度有一定限制，本图显示的深度为3。
- 树底部标记为1/1的位置下没有更多的统计记录，因此我们选择一次随机移动并添加一个新的记录（加粗），初始化为0/0。

从UCB到博弈：模拟（roll-out/simulation）



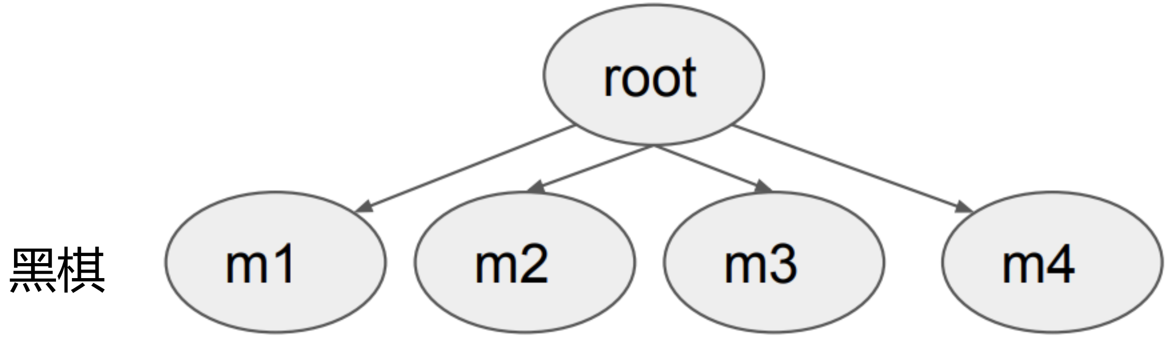
新记录添加后，蒙特卡洛模拟开始，如虚线箭头所示。模拟中的动作可能完全是随机的，也可能使用计算来加权随机性。

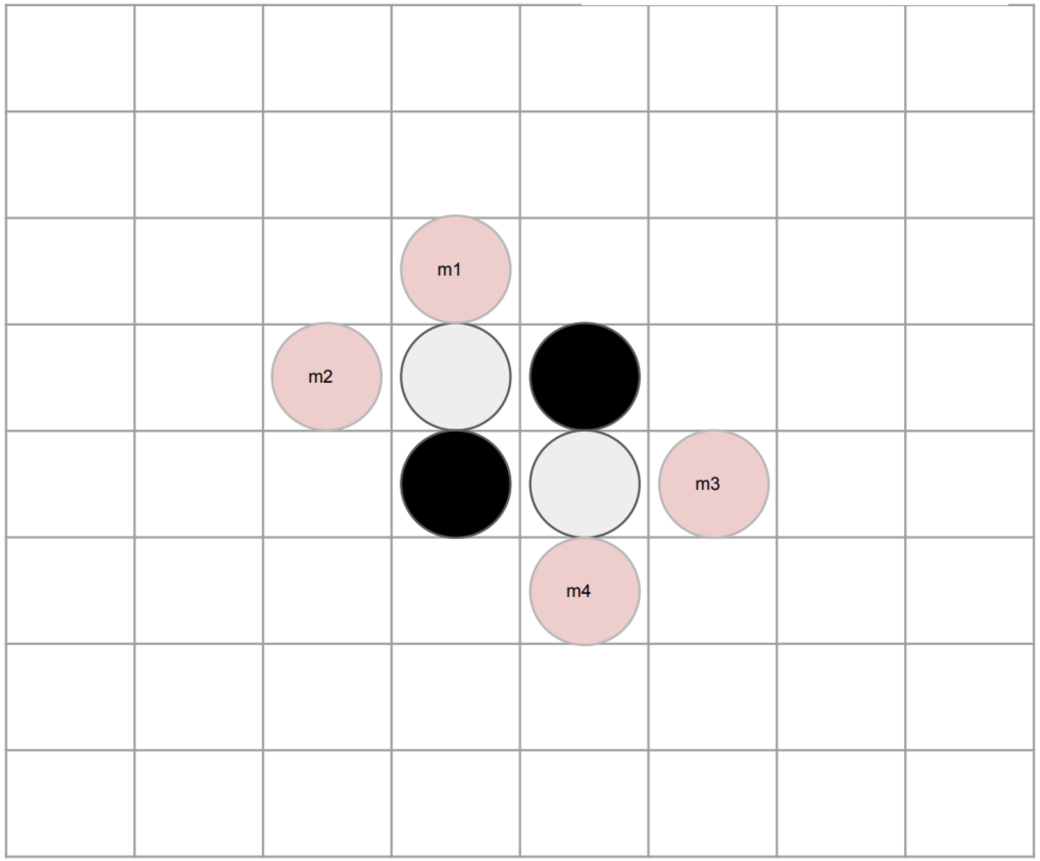
从UCB到博弈：回溯（backpropagation）



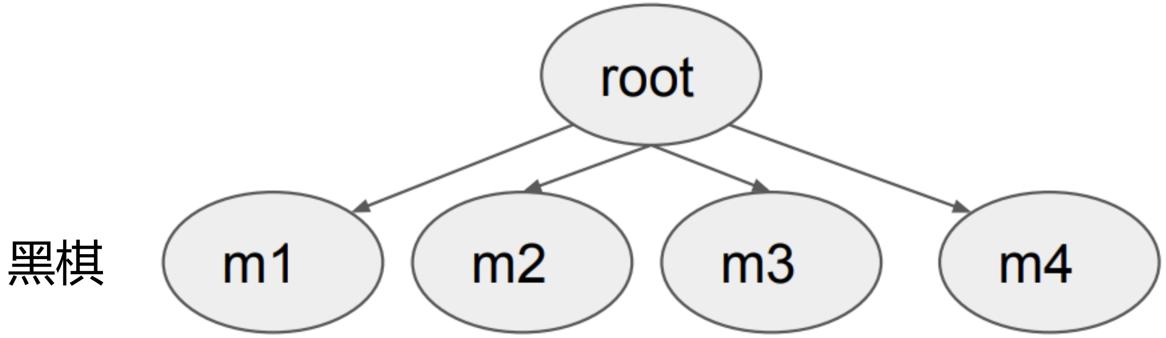
- 在模拟结束后，所采取路径上的所有记录都会被更新。
- 每个相关节点的次数都会加1，如果最后获胜了每个相关节点都会将胜利次数加1，这在加粗的数字中显示。

MCTS：黑白棋/反转棋

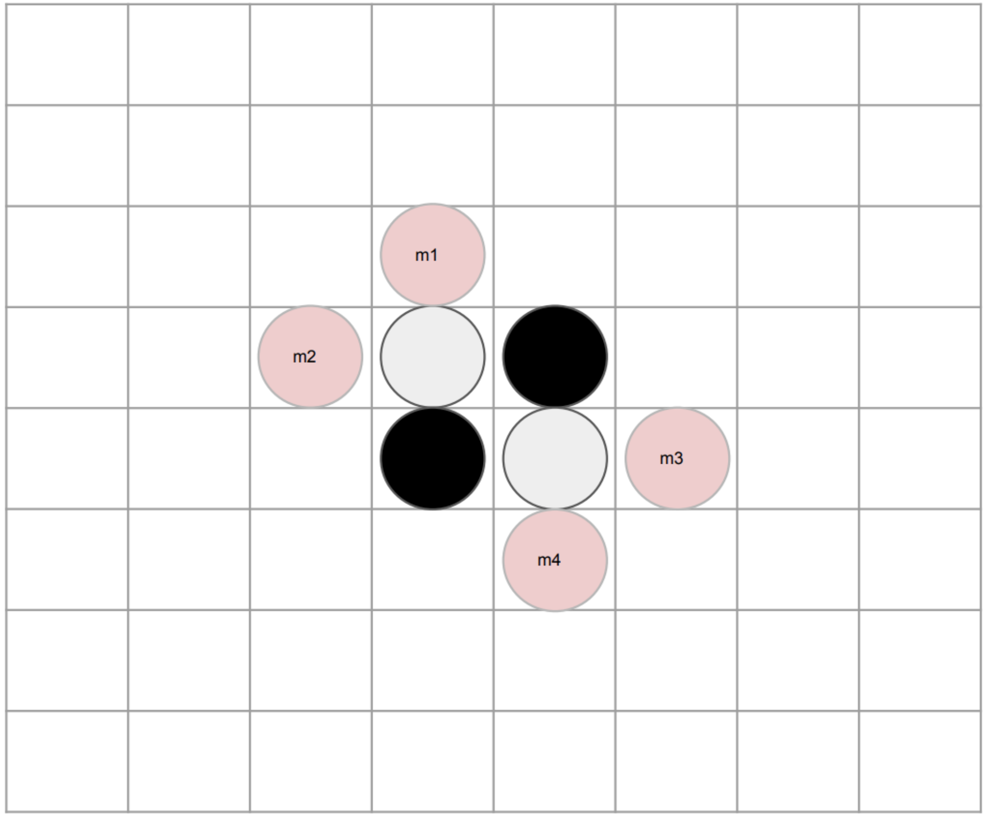




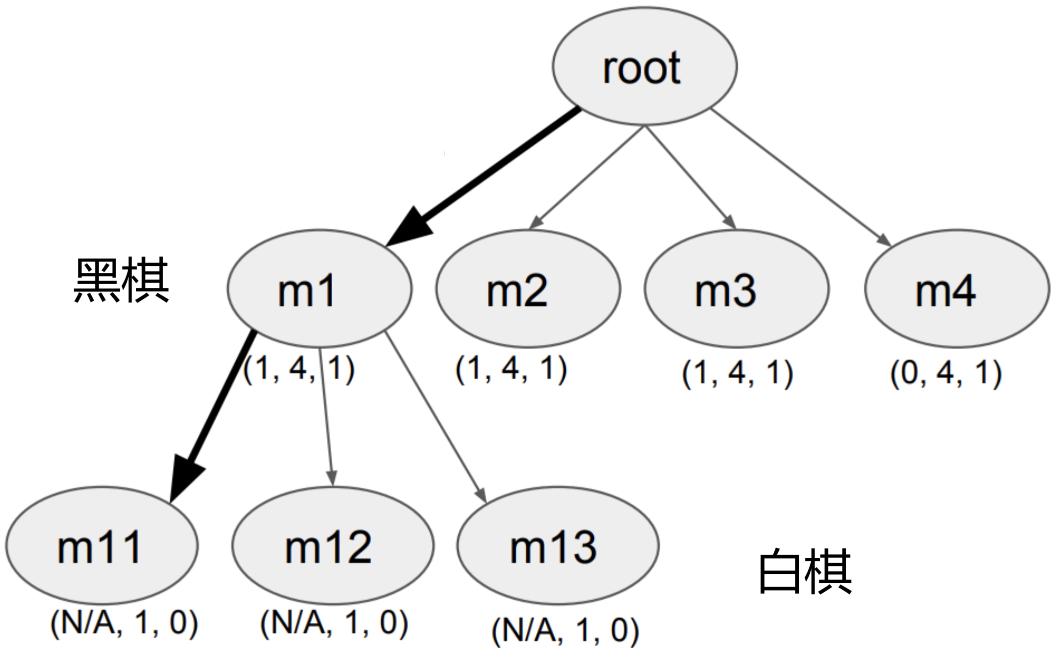
MCTS：黑白棋/反转棋

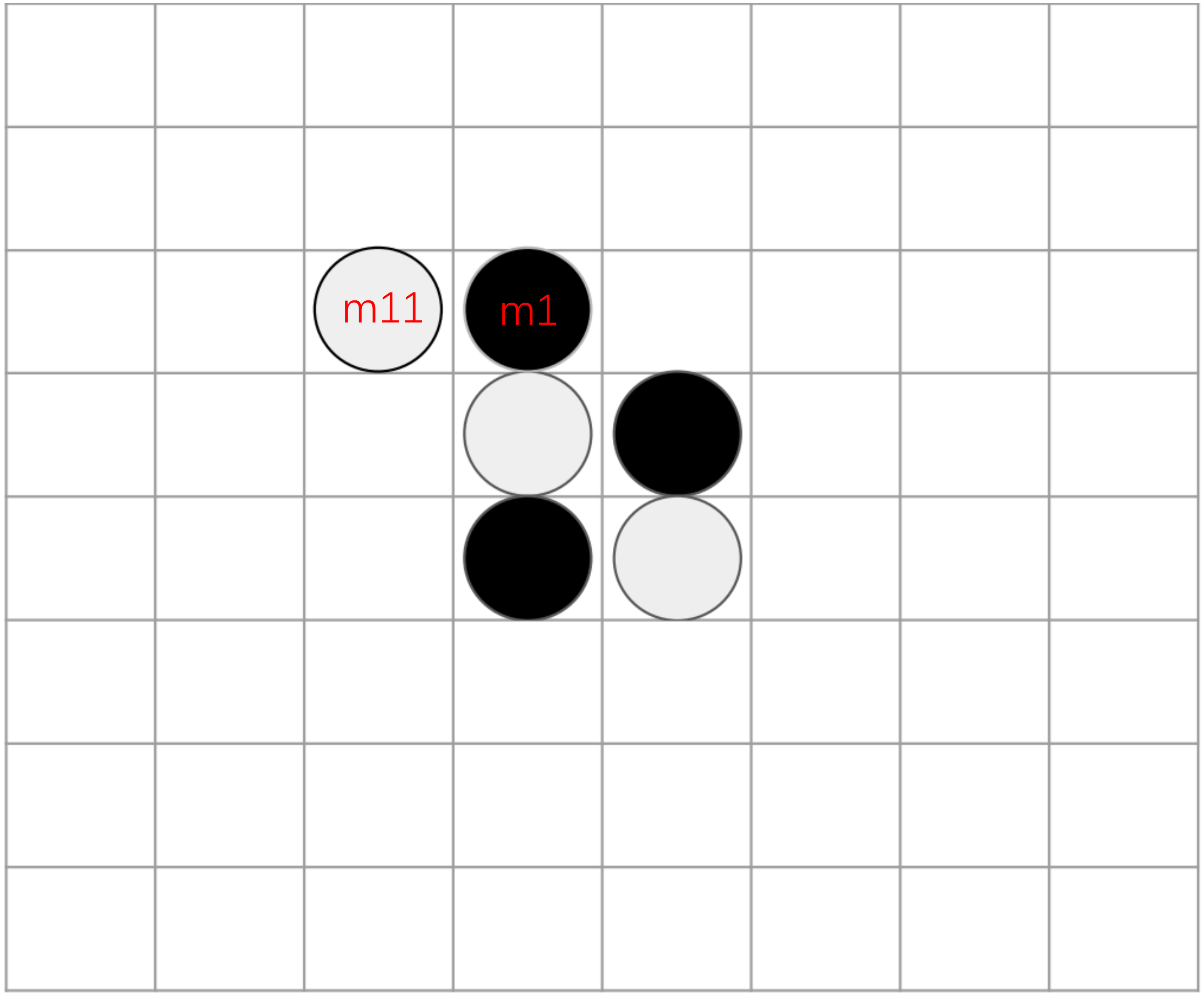


	估值	父母节点次数	次数
m1	1	4	1
m2	1	4	1
m3	1	4	1
m4	0	4	1

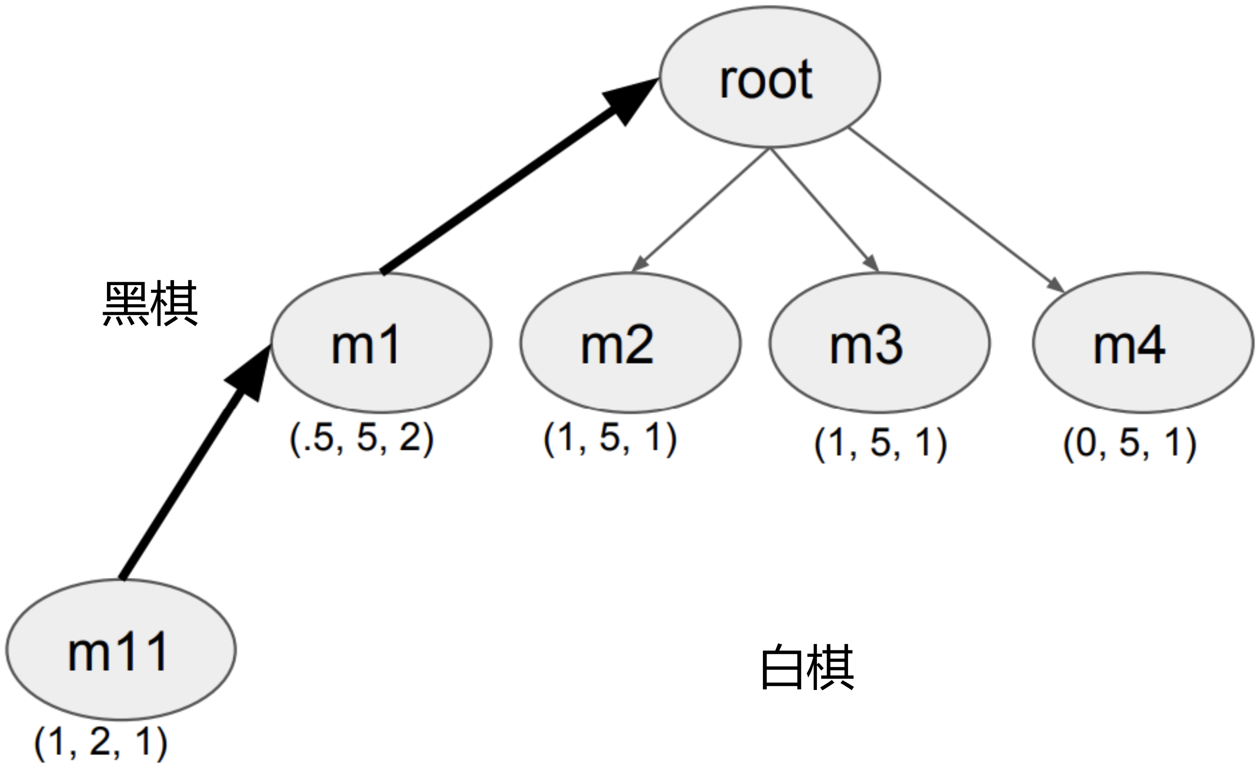


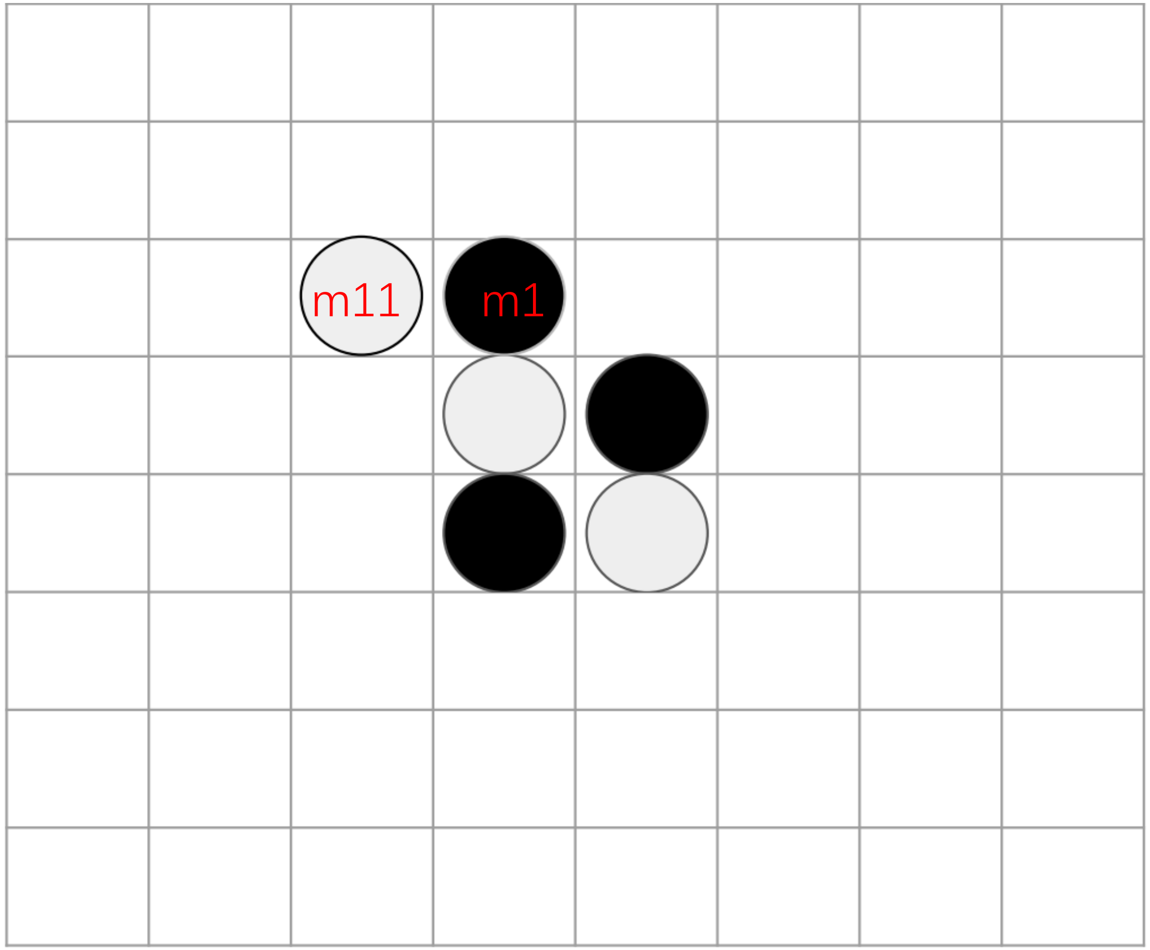
MCTS：黑白棋/反转棋


root
黑棋
m1
(1, 4, 1)
m2
(1, 4, 1)
m3
(1, 4, 1)
m4
(0, 4, 1)
m11
(N/A, 1, 0)
m12
(N/A, 1, 0)
m13
(N/A, 1, 0)
白棋


m11
m1

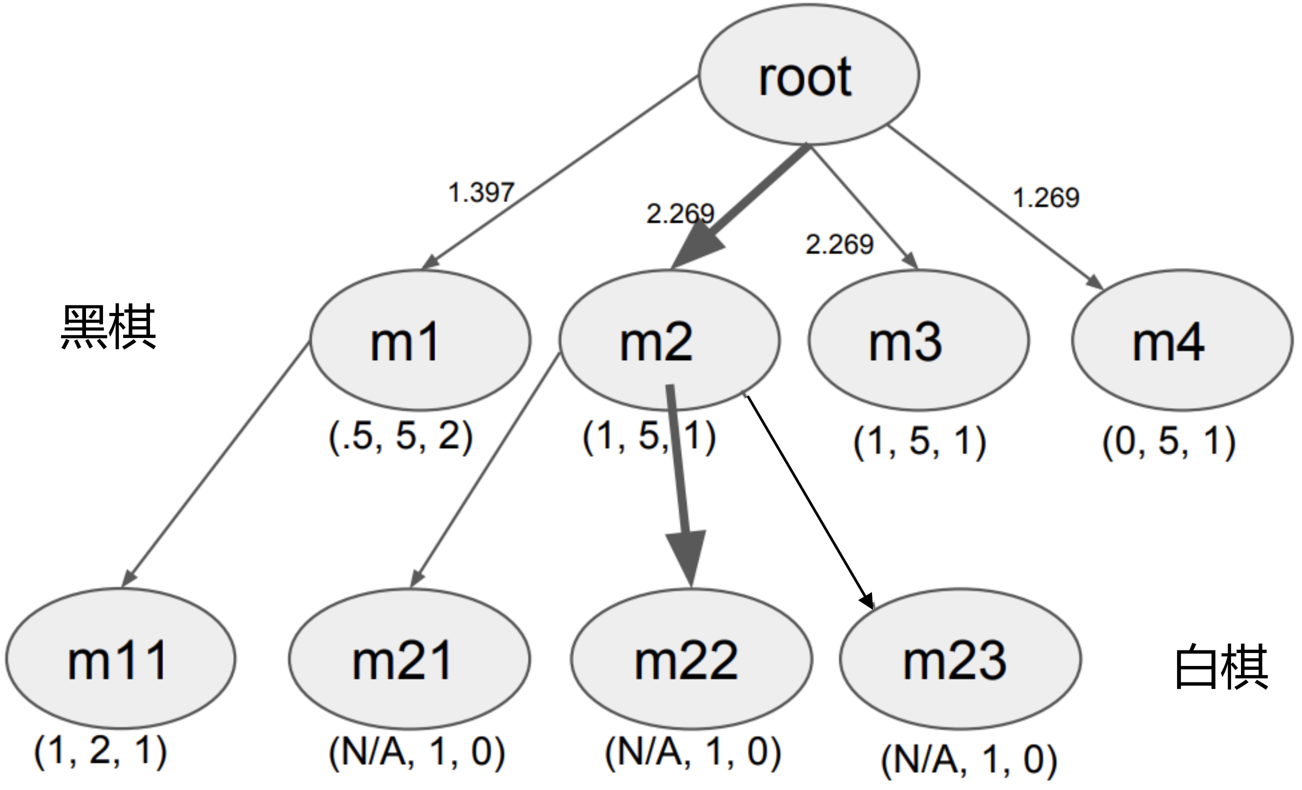
MCTS：黑白棋/反转棋



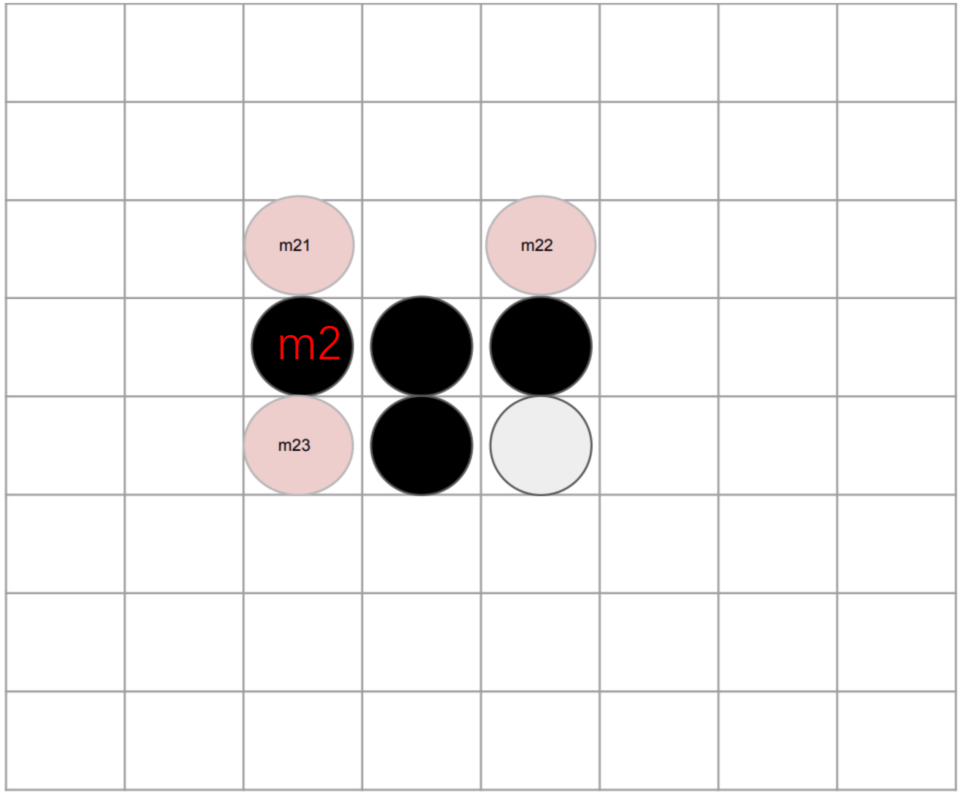


白棋获胜，回溯

MCTS：黑白棋/反转棋

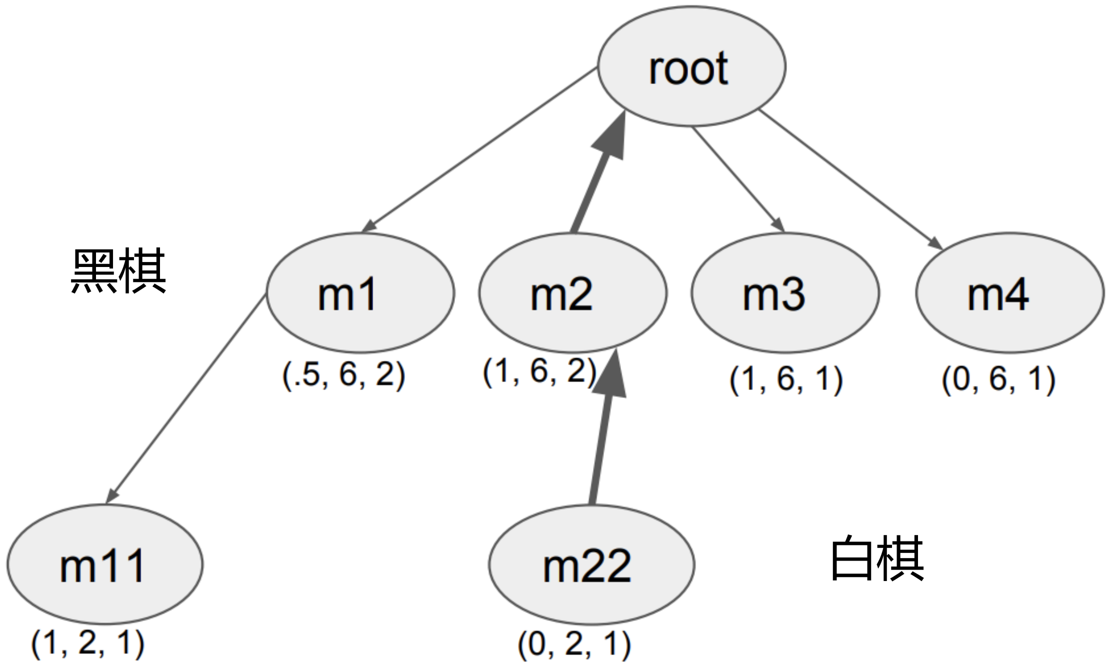


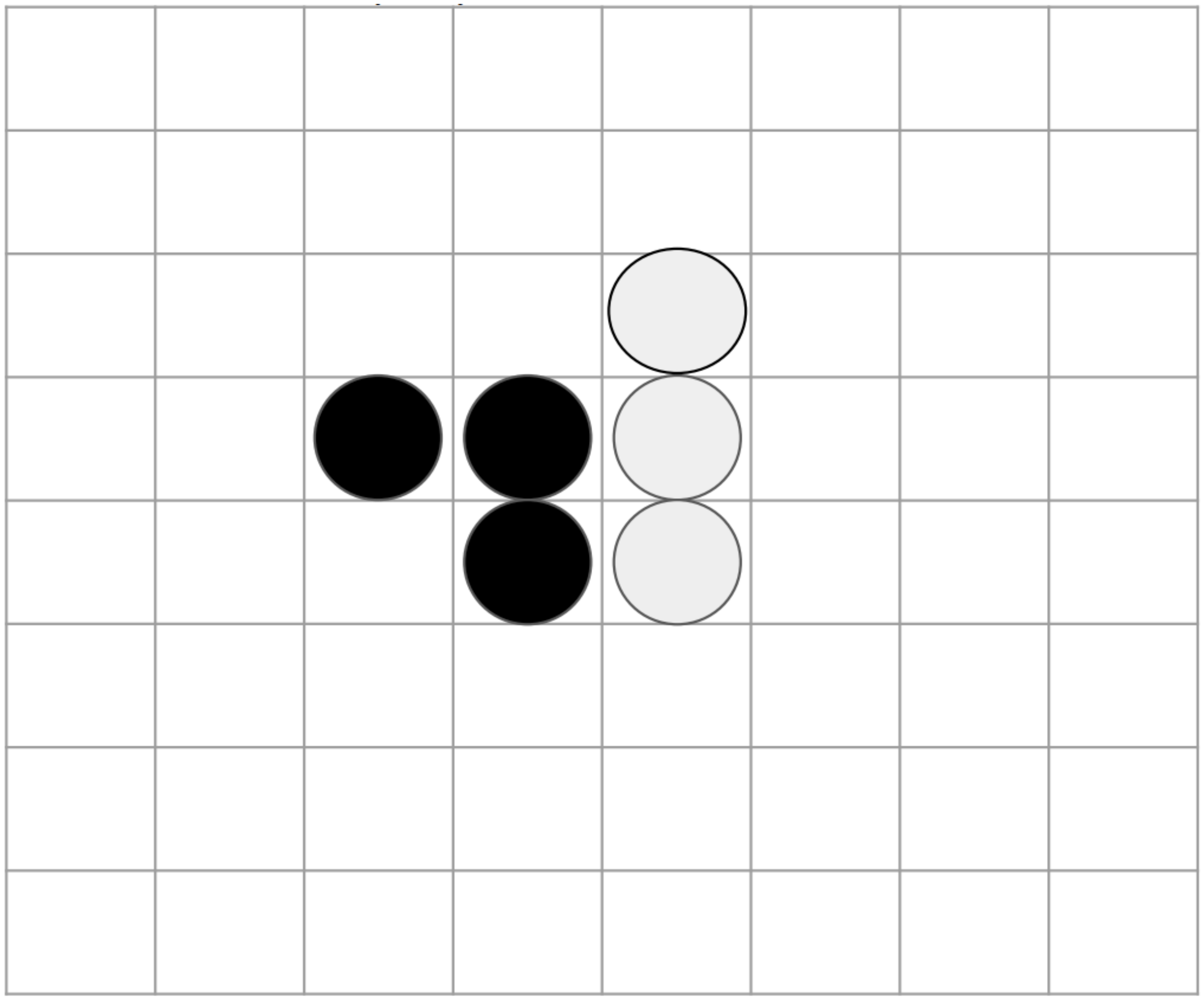
root
1.397
2.269
2.269
1.269
黑棋
m1
(.5, 5, 2)
m2
(1, 5, 1)
m3
(1, 5, 1)
m4
(0, 5, 1)
m11
(1, 2, 1)
m21
(N/A, 1, 0)
m22
(N/A, 1, 0)
m23
(N/A, 1, 0)
白棋



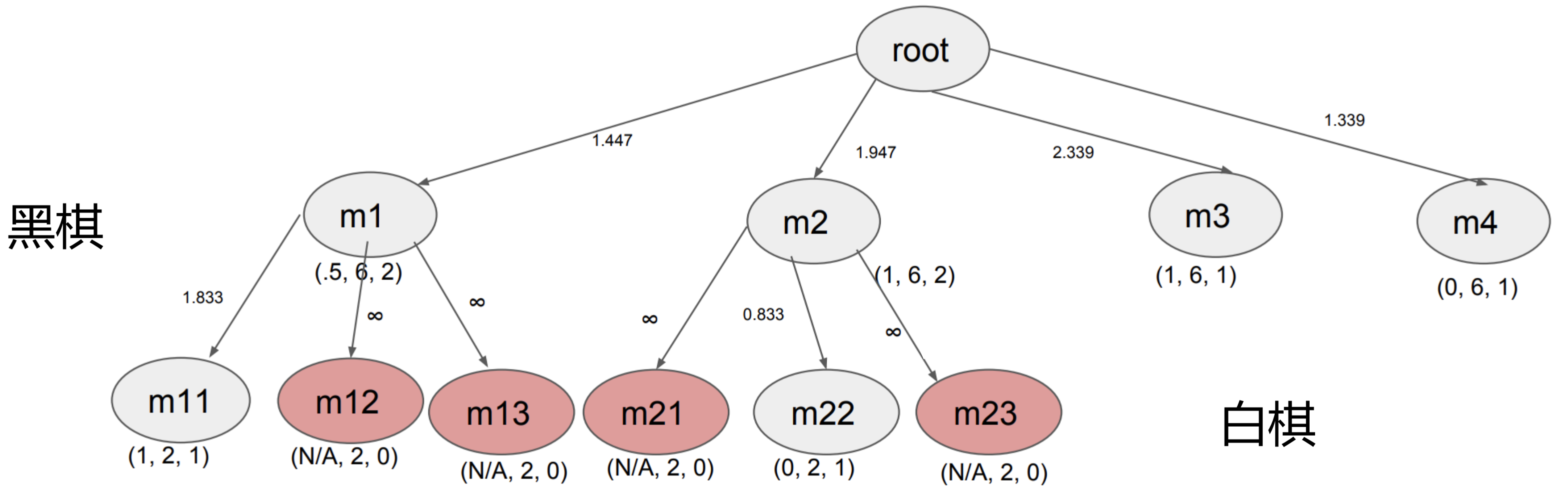
m21
m22
m2
m23

MCTS：黑白棋/反转棋





MCTS：黑白棋/反转棋



root
1.447
1.947
2.339
1.339
黑棋
m1
(.5, 6, 2)
m2
(1, 6, 2)
m3
(1, 6, 1)
m4
(0, 6, 1)
1.833
∞
∞
∞
0.833
∞
m11
(1, 2, 1)
m12
(N/A, 2, 0)
m13
(N/A, 2, 0)
m21
(N/A, 2, 0)
m22
(0, 2, 1)
m23
(N/A, 2, 0)
白棋

MCTS：伪代码

Algorithm 1 Monte Carlo Tree Search

```
procedure MonteCarloTreeSearch(x_0)
    τ_0 ← MakeTree(x_0)                         ▷ Initialize decision tree with root state x_0
    repeat                                       ▷ Main, iterative loop of the algorithm
        τ_i, r ← TreePolicy(τ_0)                 ▷ Exploration, returning new leaf and current reward
        x_i ← τ_i.state
        r ← DefaultPolicy(x_i)                   ▷ Simulation, completing episode and returning total reward
        Backpropagate(τ_i, r)                    ▷ Update node statistics along exploration path
    until Timeout()
    return BestAction(τ_0)                       ▷ Pick the approximately optimal root-level action
end procedure

procedure TreePolicy(τ)                          ▷ Decision policy for exploration
    r ← 0
    x ← τ.state
    while NonTerminal(x) do
        a ← SelectAction(x)                      ▷ Heuristically select an action from A
        x' ← Transition_P(x, a)                  ▷ Sample transition from generative model
        r ← r + R(x, a, x')
        if τ.children[a][x'] = null then         ▷ Is this the first observation of x --a--> x'?
            τ.children[a][x'] ← MakeTree(x')     ▷ Initialize leaf node for state x'
            return τ.children[a][x'], r          ▷ Move on to simulation phase
        end if
        τ ← τ.children[a][x']
        x ← x'
    end while
    return τ, r
end procedure

procedure DefaultPolicy(x, r)                    ▷ Decision policy for simulation
    while NonTerminal(x) do
        a ← RandomAction(x)                      ▷ Randomly select an action from A
        x' ← Transition_P(x, a)                  ▷ Sample transition from generative model of P
        r ← r + R(x, a, x')
    end while
    return r
end procedure

procedure Backpropagate(τ, r)                    ▷ Update statistics along path to this tree node
    repeat
        τ.reward ← τ.reward + r
        τ.count ← τ.count + 1
        τ ← τ.parent
    until τ = null
    return
end procedure

procedure BestAction(τ)                          ▷ Pick the approximately best action at this tree node
    return argmax_{a∈A} (Σ_{τ'∈τ.children[a][·]} τ'.reward) / (Σ_{τ'∈τ.children[a][·]} τ'.count)
end procedure
```

总结

北京大學 PEKING UNIVERSITY

- 如何评估博弈树具有随机性/难以遍历的状态：蒙特卡洛树搜索
- 如何平衡已有信息和不确定性：上置信界
- 搜索深度以外的部分：估值函数/蒙特卡洛模拟

谢
谢

北京大學
PEKING UNIVERSITY