Computer Go :
Monte Carlo Search and Statistical Learning

Cheng Yang

NLP, Baidu

History

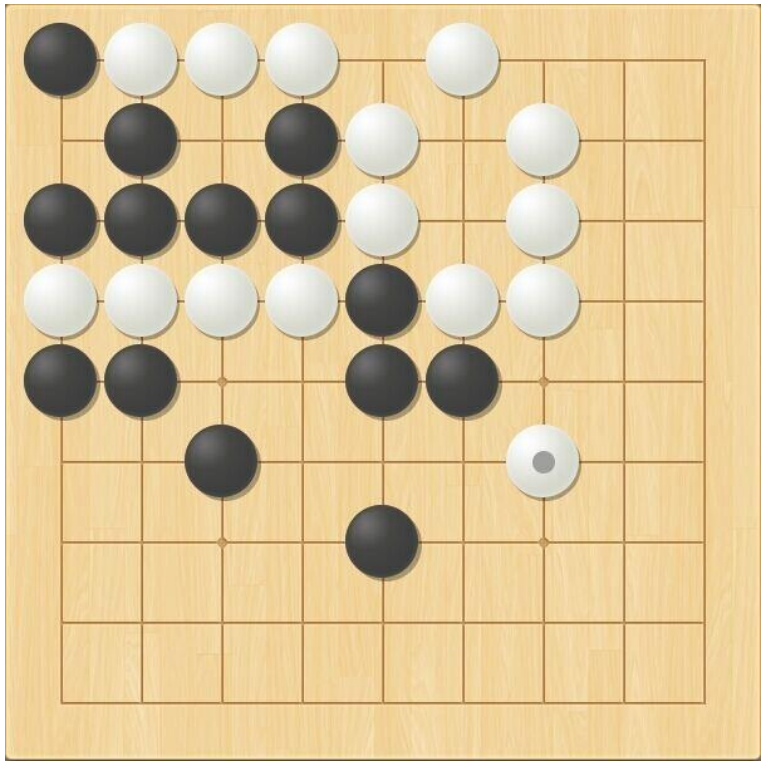
- **Event**
  - 1988 Othello Won Master (Kai-Fu Lee)
  - 1997 DeepBlue Won Kasparov (IBM)
  - 2007 Top 10 Breakthrough (Alberta)
  - 2011 Watson Jeopardy Challenge (IBM)
- **Progress**

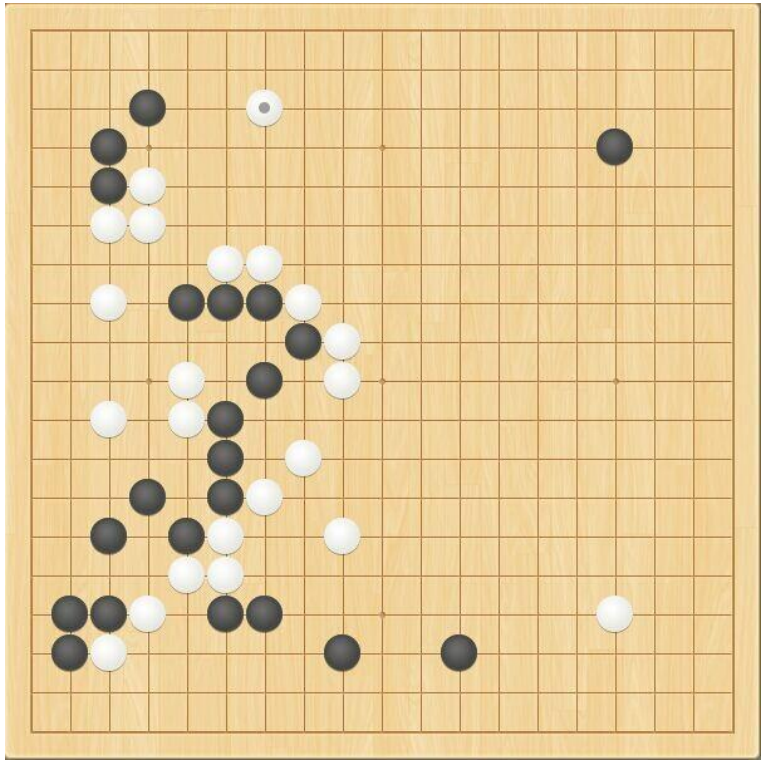
Computer Go

- Introduction
  - Go rule
    - 1) Who Has More Area Wins 2) Alive and Dead
  - One Of Most Difficult Challenges In AI
  - Drosophila Of AI ?
    - a) Fermat's Last Theorem b) Four Color Theorem
    - Biostatistics, Classical Planning, Active Learning
    - Page Search, Personalized Recommend, Natural Language
- Challenge
  - 1) Super Large Space 2) Unknown Evaluation
  - Alpha-Beta Search Totally Fails !

Computer Go

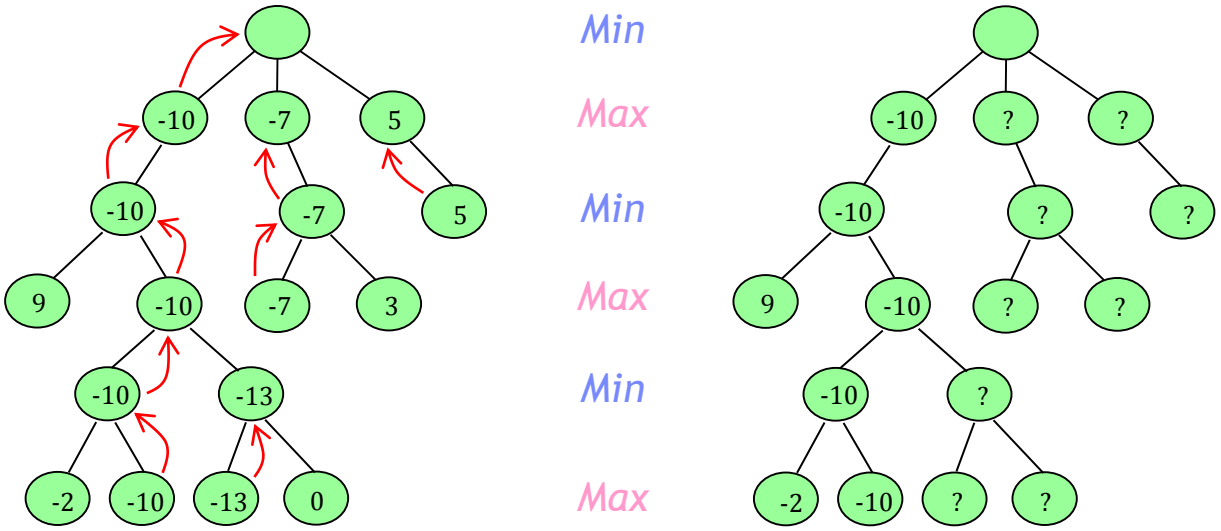
- BINGO!
  - 9x9: 6 Dan (Pro) 19x19: 1-2 Dan
  - Country: Top 1 World: Top Level





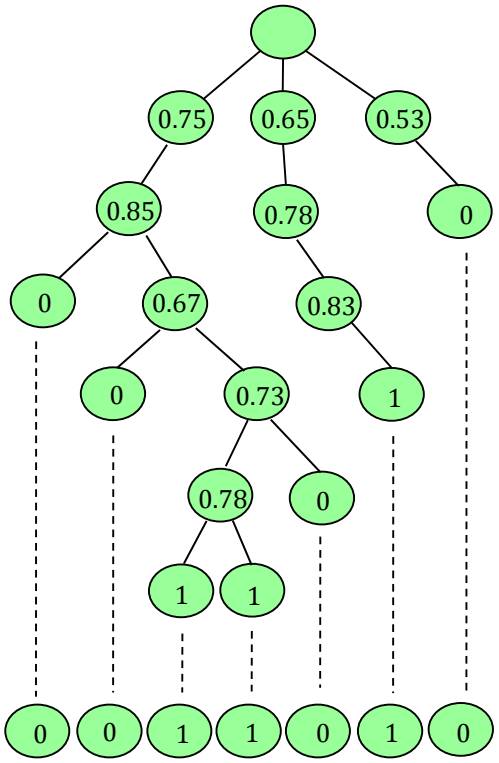
Mini-Max Search

- Principle
  - Solution To Zero-Sum Game
  - Max Player Maximize The Chances Of Winning
  - Min Player Minimize The Chances Max Player Winning
- Example



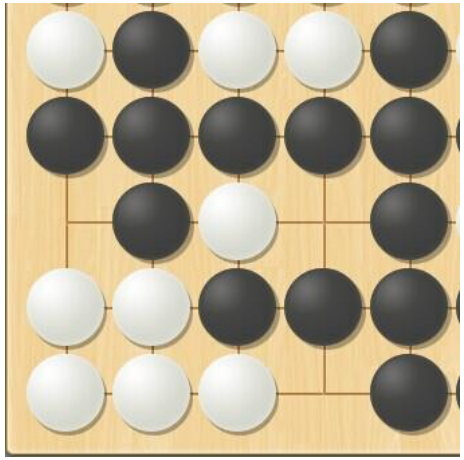
Monte Carlo Search

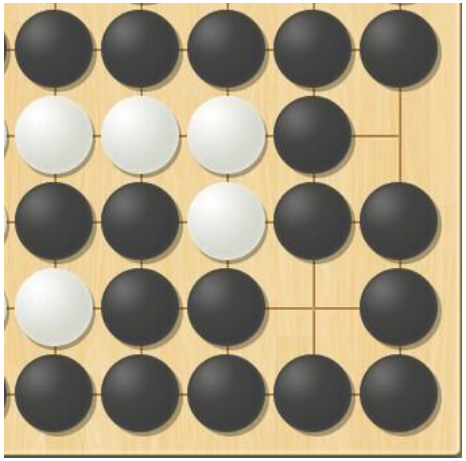
- Introduction
  - Brugmann Applied In Go (1993)
  - Seemingly Ridiculous Idea
  - Necessarily Bad or Slow ?
- In-Tree Part
  - Best First Search
  - Mean-Value Comparison
- Out-Tree Part
  - Randomly Playing Moves
  - Until Terminal State



Monte Carlo Search

- Descend Process
  - In-Tree Part And Out-Tree Part
  - Until When Each Block Has Only Simple Eyes
- Backup Process
  - Evaluate The Score Accurate And Fast
  - Backup The Score Along The Path Reversely





Multi-Armed Bandit

- Problem
  - Each Arm Provide Reward From Distribution
  - Independent Distribution Associated To Each Arm
  - Gambler Has No Initial Knowledge About Arms
  - Objective: Maximize Reward Sum By Iterative Plays
- Background
  - Extensively Studied Problem In Statistics
  - Fundamental In Reinforcement Learning
  - Exploration Vs. Exploitation Dilemma
  - What Is Good Policy ?

UCB Algorithm

- Introduction
  - UCB = Upper Confidence Bounds
  - Auer Proposed (2002)
- UCB1
  - Initialization: Play Each Arm Once
  - Loop: Play k^{th} Arm That Maximizes This Formula

$$R_k / t_k(n) + \text{sqrt}\,(2\log(n) / t_k(n))$$

Where n Is Overall Number Of Plays

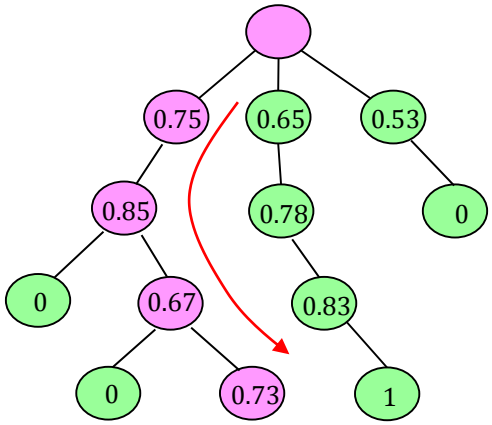
- UCB1-Tuned

$$R_k / t_k(n) + \text{sqrt}\,(\log(n) / t_k(n) \cdot \min\{1/4,\ T_k(t_k(n))\})$$

  - Auer Declares UCB1-Tuned Substantially Better Than UCB1

UCT Algorithm

- Principle
  - UCT = UCB Applied To Tree
  - Consider Each Node As A Bandit
  - Play Sequences Of Bandits
  - Begin From Root And End At Leaf



- Formula

$$w/v + \mathrm{sqrt}\,(2\log(n)/v) \qquad (1)$$

$$w/v + \mathrm{sqrt}(\log(n)/v \cdot \min\{1/4,\ T(v)\}) \qquad (2)$$

  - (2) Substantially Better Than (1) In Auer's Experiments
  - Correspond To Our Early Results

UCT Algorithm

- How Does Tree Grow
  - Expand Some Node After Each Simulation
  - Expand Nodes With Threshold T
  - Expand Nodes Gradually (Progressive Widening)
- How To Backup Values
  - Optimal Arm Is Played Exponentially More Often Than Any Arm

    When Rewards Are In [0, 1]
  - Not Return Real Score
  - Only Return Win =1 Or Lose=0

UCT Algorithm

- Advantage
  - Highly Selective Best First Search
  - Only Require Black Box Simulator
  - Usually Works Without Any Knowledge
  - Converge To Optimal Policy With Appropriate Exploration
  - Anytime, Computationally Efficient And Highly Parallelizable

Rapid Action Value Estimation

- UCT's Weakness
  - Initially Every Action Samples Once
  - For Low Variance Every Action Samples Multi Times
  - Slow Learning In Super Large Space
- Rapid Action Value Estimation
  - Brugmann (1993) Gelly (2007)
  - Consider Every Subsequent Action As First One

Let $Rave(s,a)$ be rapid action value

For episode $s_1, a_1, s_2, a_2, \ldots, s_t$

$Rave(s_m,a_n)$ is updated when

$\forall s_m \in S, a_n \in A(s_m), m \leq n$ and $\forall r < n, a_r \neq a_n$

Rapid Action Value Estimation

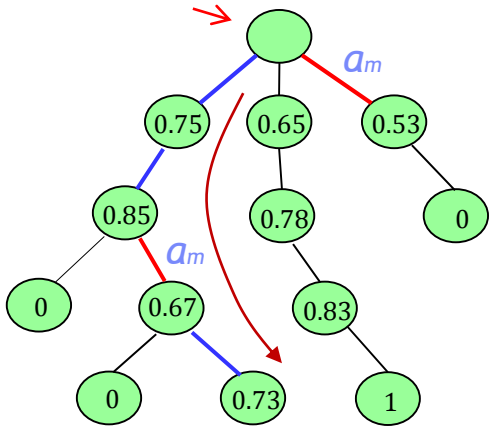
- *Rapid Action Value Estimation*
  - *Linear Combination To UCB Formula*

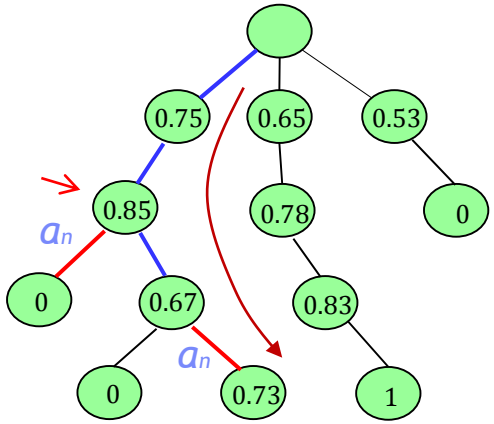
$$a \cdot (w/v) + b \cdot (rw/rv) \qquad (3)$$

Initially b≈1, RAVE Value ominates

Later b Decreases, a Increases

Eventually a≈1, b≈0, UCT Value Dominates





Rapid Action Value Estimation

- Advantage And Disadvantage
  - Learn Quickly With More Statistics (78%)
  - Extensible To Partially Transposed Sequences
  - Tricky, Empirical And Tuning

Supervised Learning

- Learn What
  - Is 5x5 Pattern OK ?
  - 3x3 Pattern And Other Rules
- How To Learn
  - Generalized Bradley-Terry Model

    For Players, $P(i \text{ win}) = y_i / (y_1 + y_2 + \ldots + y_n)$

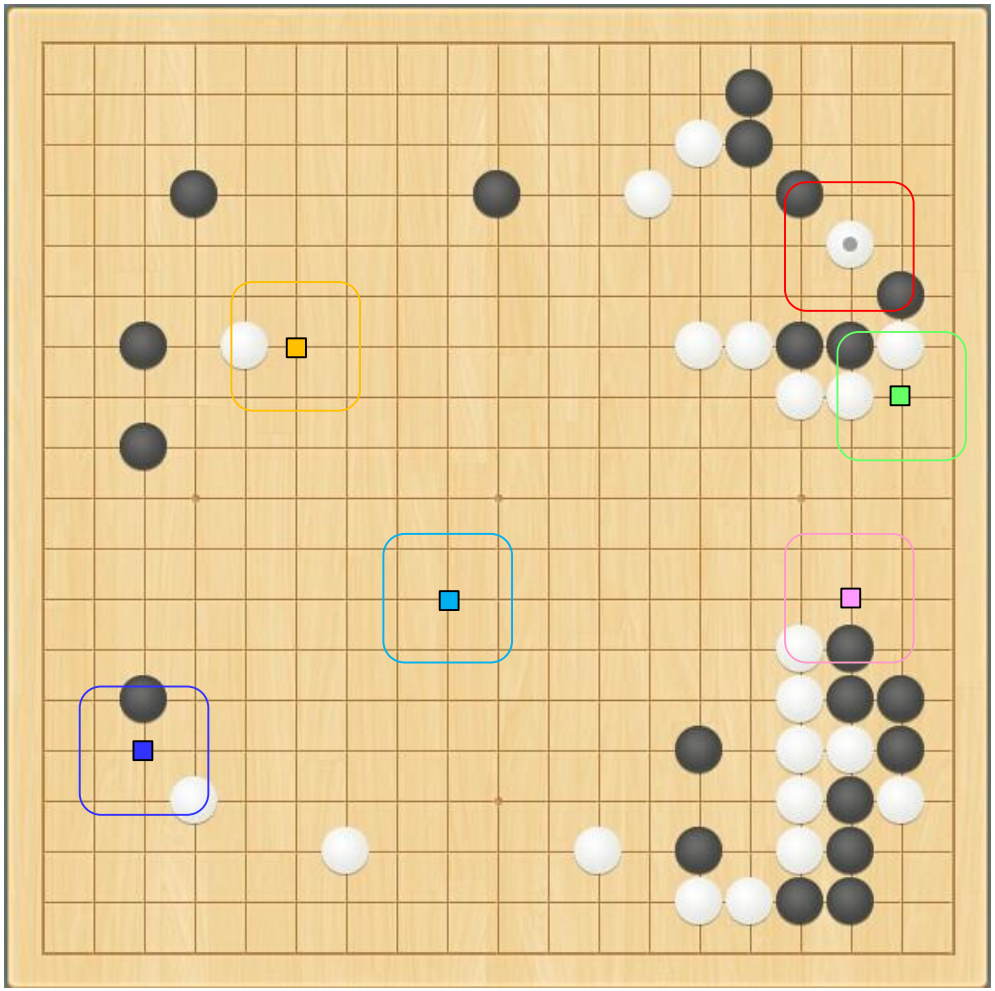
    For Teams, $P(1\text{-}2\text{-}3 \text{ win}) = y_1y_2y_3 / (y_1y_2y_3 + y_2y_4 + \ldots + y_1y_4y_5y_7)$

    Is Linearity Hypothesis Right ?
  - How to Use Pro Manual

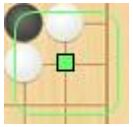
    Consider $(y_1, y_2, \ldots, y_n)$ $(R_1, R_2, \ldots, R_N)$

    $L = P(R_1)\cdot P(R_2)\cdots P(R_N)$, $P(R_i) = y_{i1}y_{i2}\ldots y_{in} / (y_{i1}y_{i2}\ldots y_{in} + \ldots)$

Supervised Learning

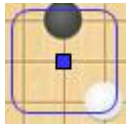


γ = 4.296



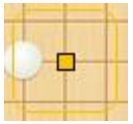
γ = 2.389

γ = 1.048



γ = 0.981

γ = 0.396



γ = 0.057

Biasing Search With γ

- Combination To (1)
  - Easily And Straightforward
  - γ's Impact Decreases Rapidly

$$w/v + \mathrm{sqrt}\,(2\log(n)/v) + H(p)/v$$

- Combination To (3)
  - Not Straightforward Combination
  - Worse Results Initially But Much Better Eventually

$$a\cdot(w/v) + b\cdot(rw/rv) + c\cdot H(p)$$

Initially $c\approx 1$, $H(p)$ Value Dominates

Later c Decreases, b Increases, RAVE Value Dominates

Eventually $a\approx 1$, $b\approx 0$, $c\approx 0$, UCT Value Dominates

Back to tree-growth

Thank You !

百度技术沙龙

关注我们：t.baidu-tech.com

资料下载和详细介绍：infoq.com/cn/zones/baidu-salon

“畅想•交流•争鸣•聚会”是百度技术沙龙的宗旨。 百度技术沙龙是由百度与InfoQ中文站定期组织的线下技术交流活动。目的是让中高端技术人员有一个相对自由的思想交流和交友沟通的的平台。主要分讲师分享和OpenSpace两个关键环节，每期只关注一个焦点话题。

讲师分享和现场Q&A让大家了解百度和其他知名网站技术支持的先进实践经验，OpenSpace环节是百度技术沙龙主题的升华和展开，提供一个自由交流的平台。针对当期主题，参与者人人都可以发起话题，展开讨论。

InfoQ 策划·组织·实施

关注我们：weibo.com/infoqchina