How AlphaGo Beats Human?

Jiyanglin Li

School of Statistics and Management
Shanghai University of Finance and Economics

August 17, 2017

Go: An ancient and fascinating game

- China

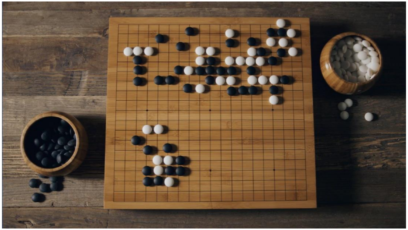
  Yao 2337-2258 BCE
  Tso Chuan 4th century BCE
  Tang Dynasty 618-907 CE

- Asia

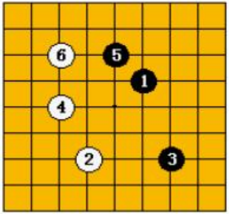
  KOR and JPN $5^{th} \sim 7^{th}$ century
  JPN Royal 8^{th} century
  JPN Public 13^{th} century

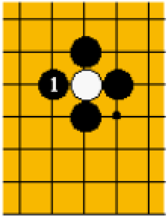
- Western countries

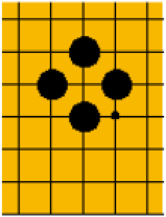
  Germany $19^{th} \sim 20^{th}$ century
  Rest countries After 1950s

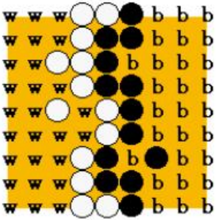


Rules of Go









Complexity: Big challenge for computers to play Go

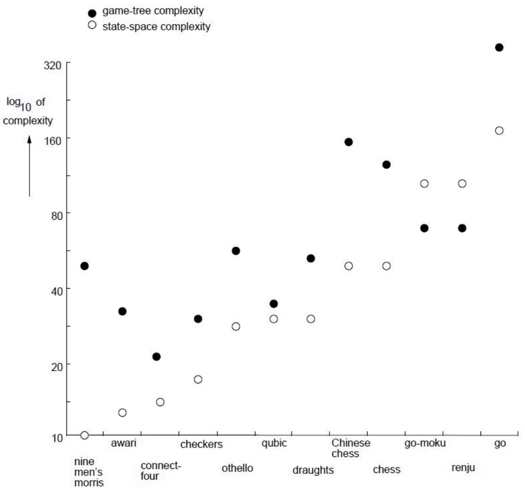
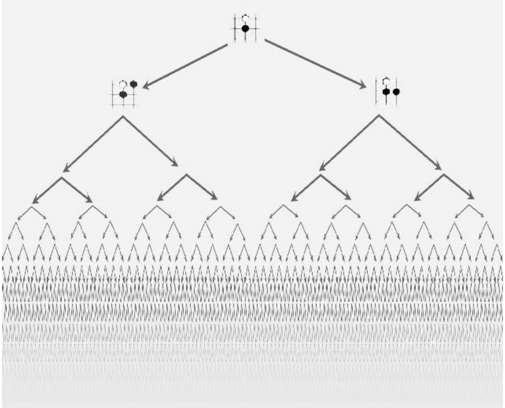


Figure: Complexity of different games (Allis, 1994)

Brute force search? Intractable!



- Huge search space: b^d possible sequences of moves
- Evaluating the game is difficult

Prerequisites: Agent and Environment

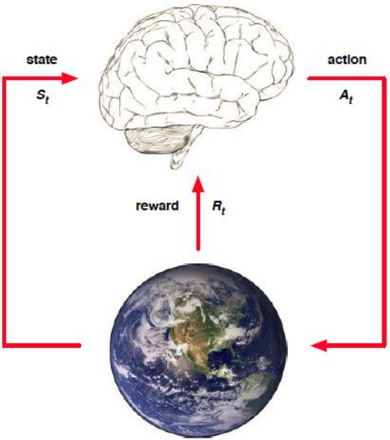


Figure: Picture Source: UCL Course, David Silver.

Prerequisites: Conceptions (Go in AI perspective)

- Agent: The subject who takes actions in an environment.
- State: s, board position.
- Move: a, possible action the agent can take.
- Policy: $p(a|s)$, probability distribution over possible moves a in position s.
- Reward: $r(s_t) = 0$ for $t < T, r(s_T) = \begin{cases} 1 & \text{for winning;} \\ -1 & \text{for losing.} \end{cases}$

  T is the terminal time step.
- Value function: $v^p(s) = E(z_t | s_t = s, a_{t \dots T} \sim p)$, where $z_t = r(s_T)$ is a outcome.

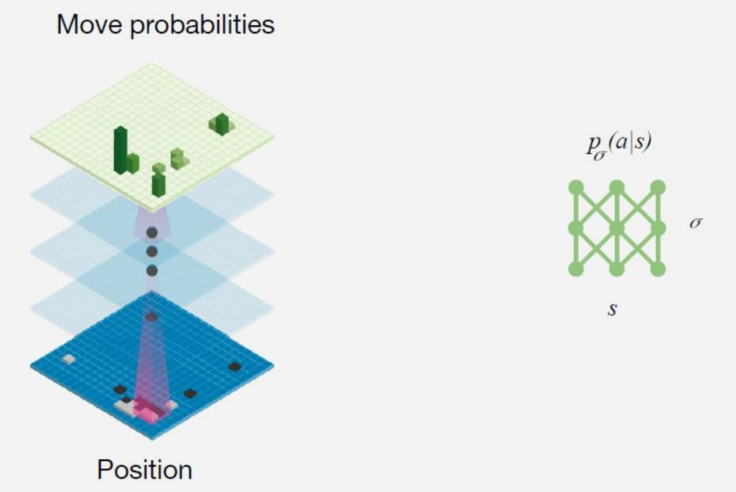
Prerequisites: Rollout

The essential of rollout is random simulation given a policy p_π:

- Uniform random rollout policy
- Fast rollout policy
- $\cdots$

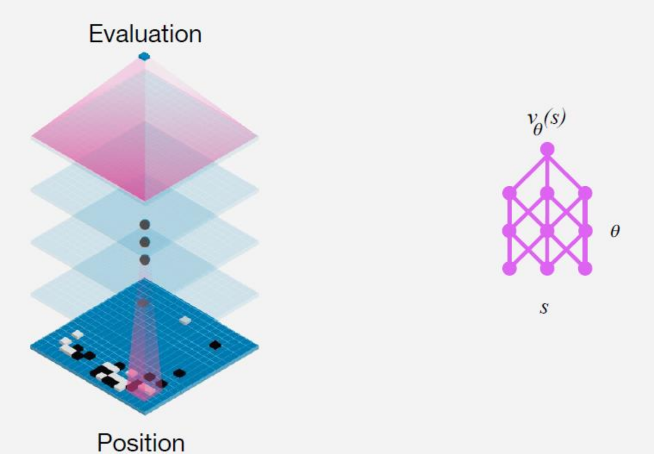
Prerequisites: Policy Network

Policy Network is to select moves.



Prerequisites: Value Network

Value Network is to evaluate board positions.



Key: $v_\theta(s) \approx v^p(s) \approx v^*(s)$!!!

Prerequisites: Monte Carlo Tree Search

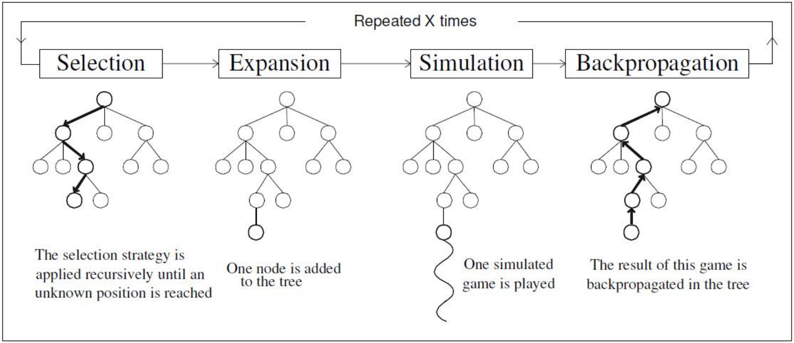
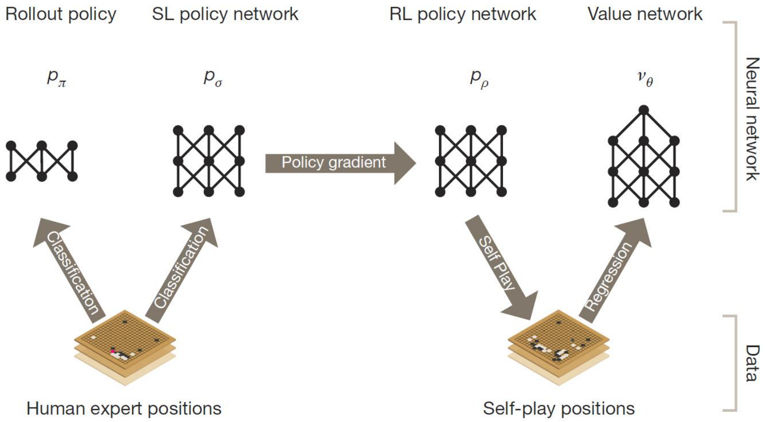


Figure: Outline of Monte-Carlo Tree Search

AlphaGo Training pipeline



Training details: Rollout Policy

- AlphaGo use a linear softmax policy.
- Training data: 8 million positions from human games on the Tygem server.
- accuracy: 24.2%.
- select an action: 6μs (3 ms for the policy network).

Training details: Policy Network

- Data: 29.4 million positions from KGS data set.
- Implemention: 50 GPUs, 3 weeks.
- Algorithm: stochastic gradient descent.
- accuracy: 57.0%.

Remarks: each position consists of s together with the move a selected by the human position

Training details: Value Network

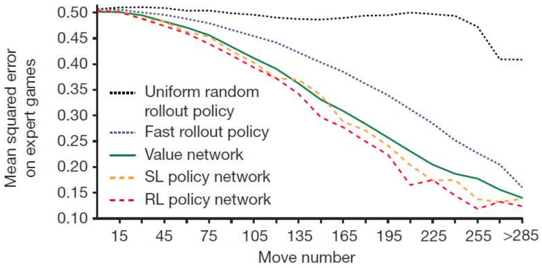
- Data: 30 million positions from self-play.
- Implemention: 50 GPUs, 1 week.
- Algorithm: stochastic gradient descent.

Remarks: each position is drawn from a unique game of self-play.

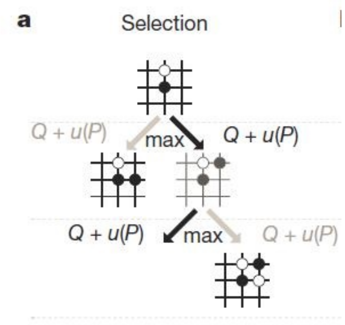
Training details: Value Network

- How strong the value network is?

  A single evaluation of $\nu_\theta(s)$ also approached the accuracy of Monte Carlo rollouts using the RL policy network p_ρ, but using 15,000 times less computation.



Monte Carlo tree search in AlphaGo



$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

$$P(s,a) = p_\sigma(a|s)$$

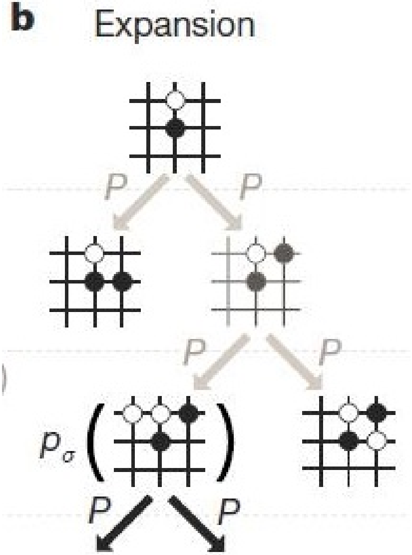
$$N(s,a) = \sum_{i=1}^{n} I(s,a,i)$$

n : the number of simulations until now

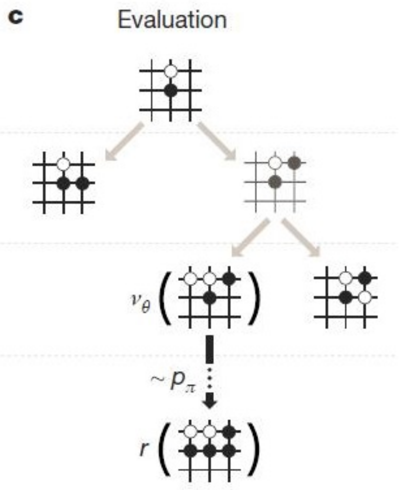
$I(s,a,i)$: whether an edge (s,a) was traversed during the ith search

$Q(s,a)$: action value, details in the later slides

Monte Carlo tree search in AlphaGo

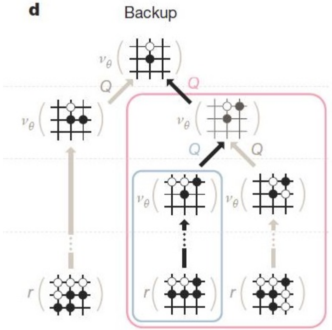


Monte Carlo tree search in AlphaGo



Monte Carlo tree search in AlphaGo

To be more easily understood, some notations are not the same with those in the paper.



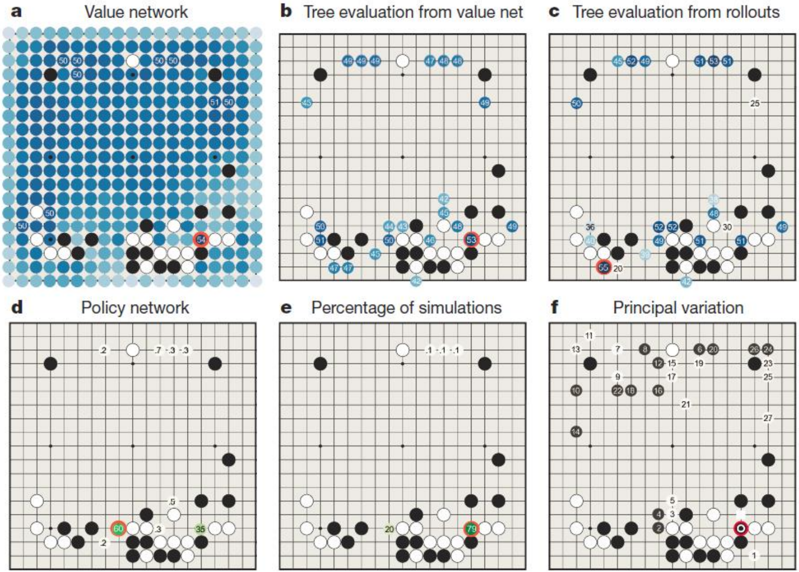
- $Q(s,a) = (1-\lambda)\bar{v}_\theta(s_L) + \lambda z_L$
- $\bar{v}_\theta(s_L) = \frac{1}{n}\sum_{i=1}^{n} I(s,a,i)v_\theta(s_L^i)$
- $\bar{z}_L = \frac{1}{n}\sum_{i=1}^{n} I(s,a,i)z_L^i$

s_L : the leaf node at the ith traversal.

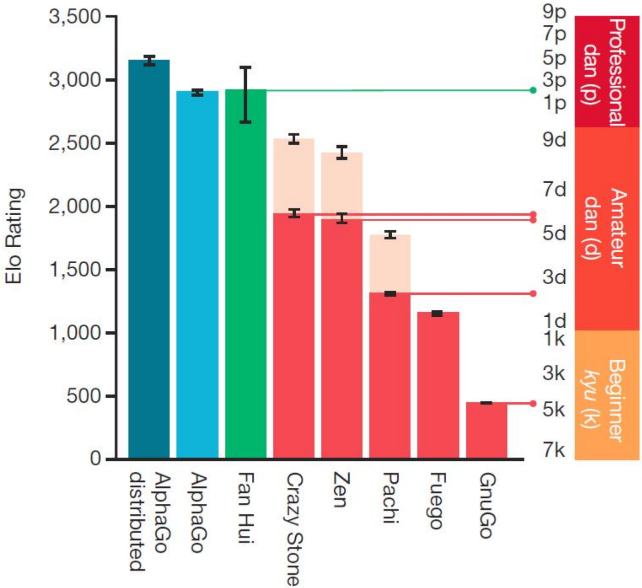
z_L : the outcome of the ith random rollout.

At the end of search AlphaGo selects the action with maximum visit count!

How AlphaGo (black, to play) selected its move



Tournament evaluation of AlphaGo



Tournament evaluation of AlphaGo

Winning rate

- AlphaGo vs other Go programs: 99.8%.
- Distributed AlphaGo vs other Go programs: 100%.
- Distributed AlphaGo vs AlphaGo: 77%.

Short name	Computer Player	Version	Time settings	CPUs	GPUs
α_{rvp}^{d}	Distributed AlphaGo	See Methods	5 seconds	1202	176
α_{rvp}	AlphaGo	See Methods	5 seconds	48	8

Tournament between different variants of AlphaGo

Short name	Policy network	Value network	Rollouts	Mixing constant	Policy GPUs	Value GPUs	Elo rating
α_{rvp}	p_σ	v_θ	p_π	$\lambda = 0.5$	2	6	2890
α_{vp}	p_σ	v_θ	–	$\lambda = 0$	2	6	2177
α_{rp}	p_σ	–	p_π	$\lambda = 1$	8	0	2416
α_{rv}	$[p_\tau]$	v_θ	p_π	$\lambda = 0.5$	0	8	2077
α_{v}	$[p_\tau]$	v_θ	–	$\lambda = 0$	0	8	1655
α_{r}	$[p_\tau]$	–	p_π	$\lambda = 1$	0	0	1457
α_{p}	p_σ	–	–	–	0	0	1517

$\alpha_{rv}, \alpha_{v}, \alpha_{r}$: no policy network (instead using tree policy p_τ).

Thank you!