CS171: Artificial Intelligence
Monte Carlo Tree Search and Alpha Go

Jia Chen

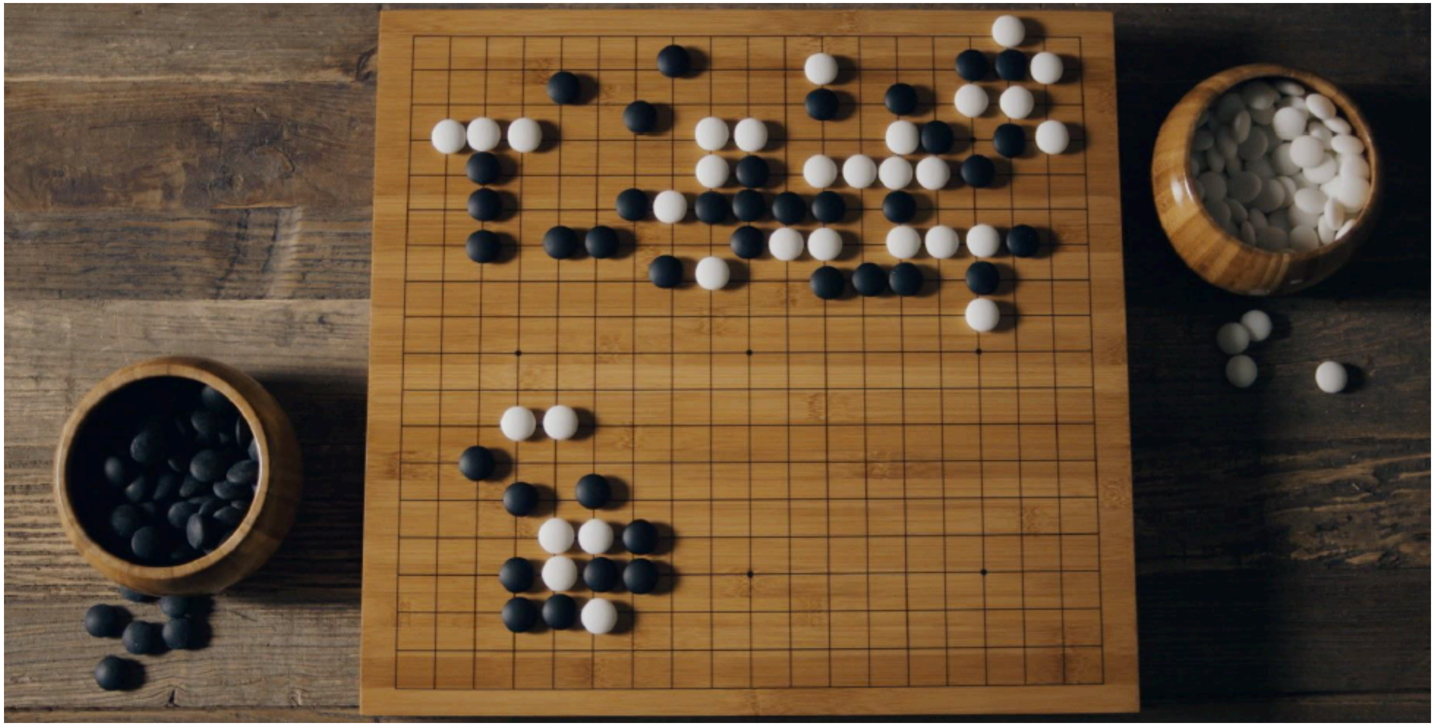
Dec 5, 2017

Schedule

- **Introduction**
- Monte-Carlo Tree Search
- Policy and Value Networks
- Results

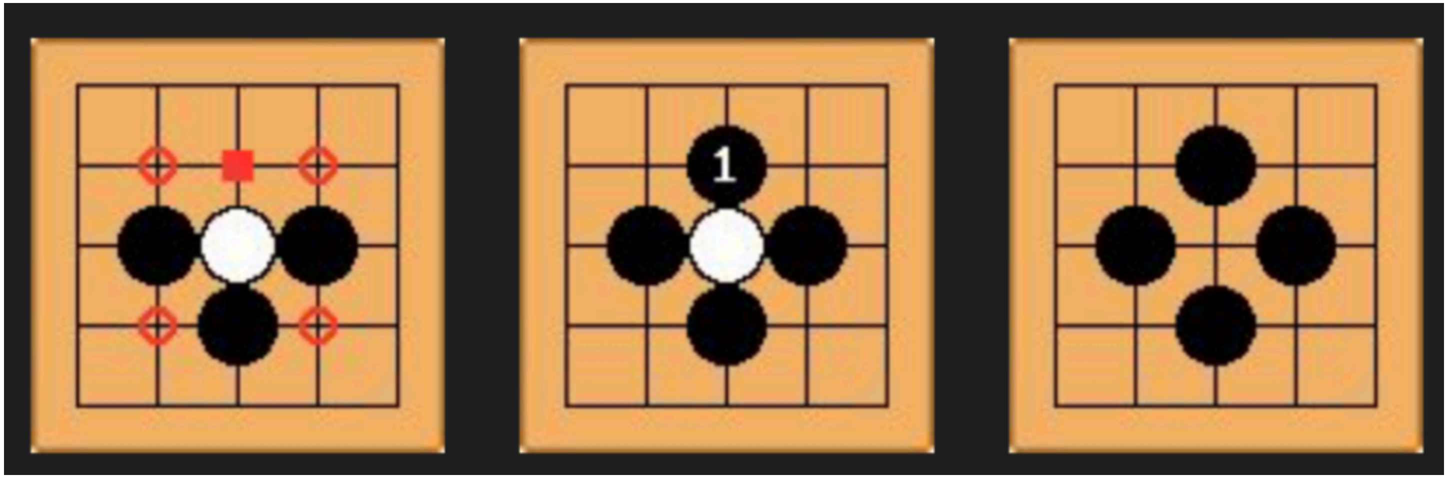
Introduction

- Go originated 2,500+ years ago
- Currently over 40 million players



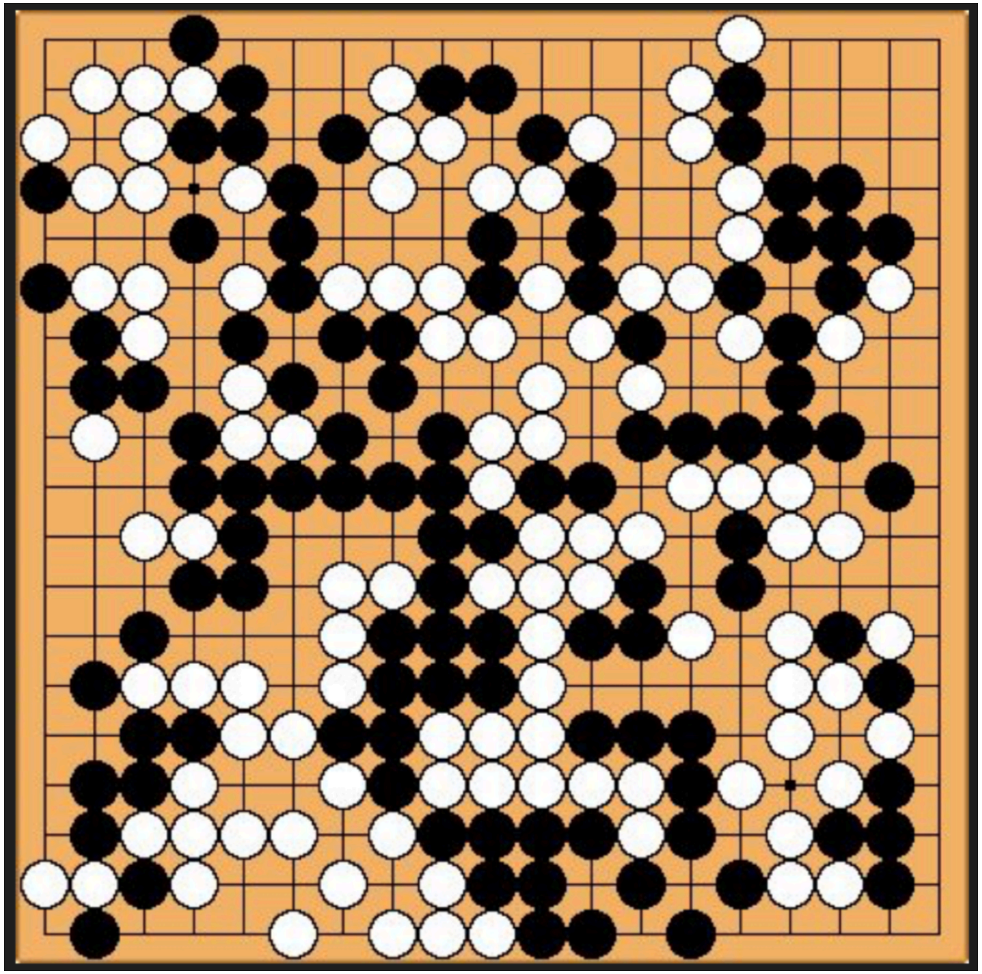
Rules of Go

- Played on a 19x19 board
- Two players, black and white, each place one stone per turn
- Capture the opponent's stones by surrounding them



Rules of Go

- Goal is to control as much territory as possible.



Why is Go Challenging?

- Hundreds of legal moves from any position, many of which are plausible
- Games can last hundreds of moves
- Unlike chess, endgames are too complicated to solve exactly
- Heavily dependent on pattern recognition

Game Trees

- A game tree is a directed graph whose nodes are positions in a game and whose edges are moves
- Fully searching this tree allows for best move for simple games like Tic-Tac-Toe
- Complexity for tree $O(b^d)$, where b is the branching factor (number of legal moves per position), and d is its depth (the length of the game)

Game Trees

- Chess: $b \approx 35$, $d \approx 80$, $b^d \approx 10^{80}$
- Go: $b \approx 250$, $d \approx 150$, $b^d \approx 10^{170}$
- Size of search tree for Go is more than the number of atoms in the universe!
- Brute force intractable

A Brief History of Computer Go

- 1997: Super human chess w/ Alpha-Beta + fast computer
- 2005: Computer Go is impossible!
- 2006: Monte-Carlo Tree Search applied to 9x9 Go (bit of learning)
- 2007: Human master level achieved at 9x9 Go (more learning)
- 2008: Human grandmaster level achieved at 9x9 Go (even more learning)
- 2012: Zen program beats former international champion with only 4 stone handicap in 19x19
- 2015: DeepMind's AlphaGo beats European Champion 5:0
- 2016: AlphaGo beats World Champion 4:1
- 2017: AlphaGo Zero beats AlphaGo 100:0


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATUREASIA.COM
28 January 2016
Vol. 529, No. 7587

Techniques behind AlphaGo

- Deep learning + Monte Carlo Tree Search + High Performance Computing
- Learn from 30 million human expert moves and 128,000+ self play games

Image: AP/ Lee Jin-man

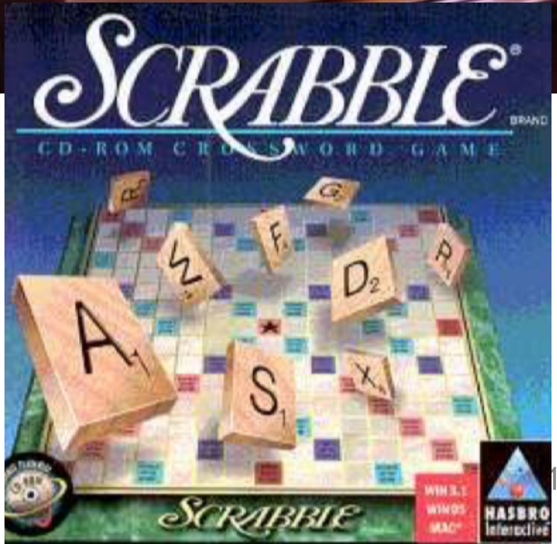
March 2016:
AlphaGo beats Lee Sedol 4-1

Schedule

- Introduction
- **Monte-Carlo Tree Search**
- Policy and Value Networks
- Results

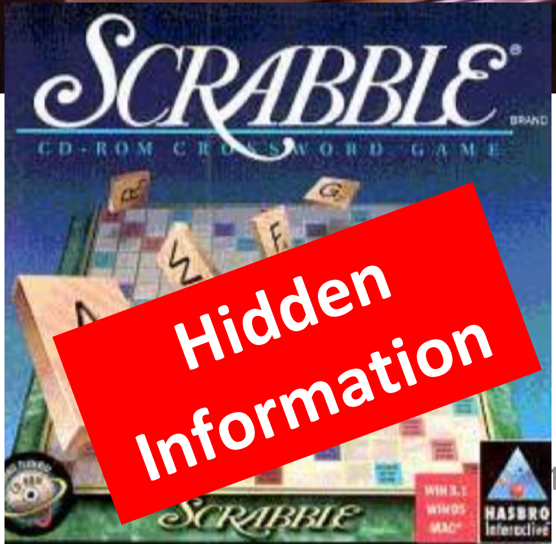
Game Tree Search

- Good for 2-player zero-sum infinite deterministic games of perfect information



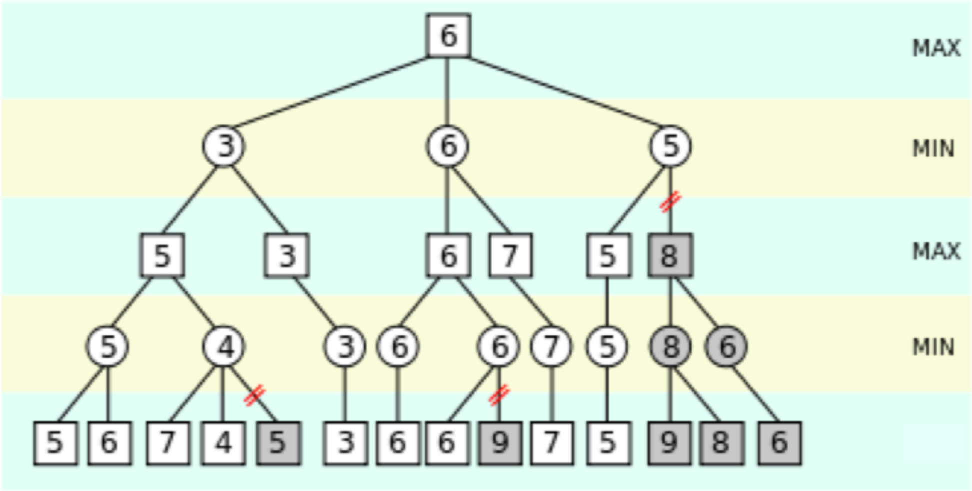
Game Tree Search

- Good for 2-player zero-sum finite deterministic games of perfect information



Conventional Game Tree Search

- Minimax algorithm with alpha-beta pruning



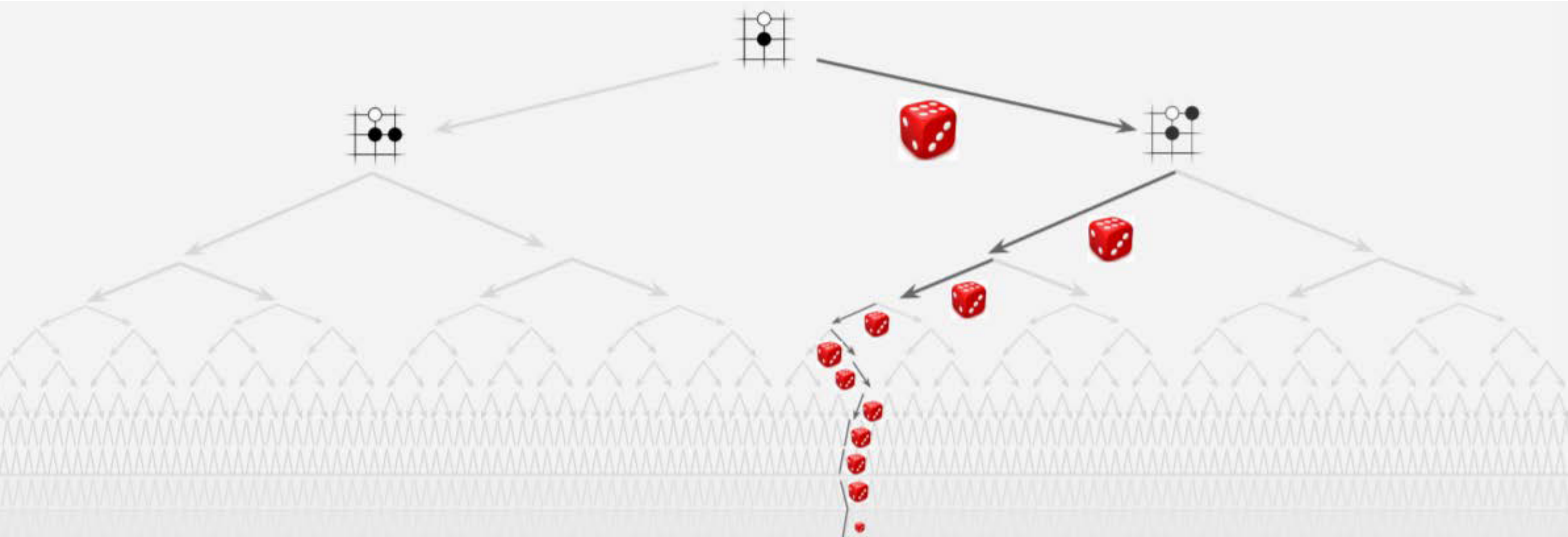
- Effective
  - When modest branching factor
  - When a good heuristic value function is known

Alpha-beta pruning for Go?

- Branching factor for Go is too large
  - 250 moves on average
  - Order of magnitude greater than the branching factor of 35 for chess
- Lack of good evaluation function
  - Too subtle to model: similar looking positions can have completely different outcomes

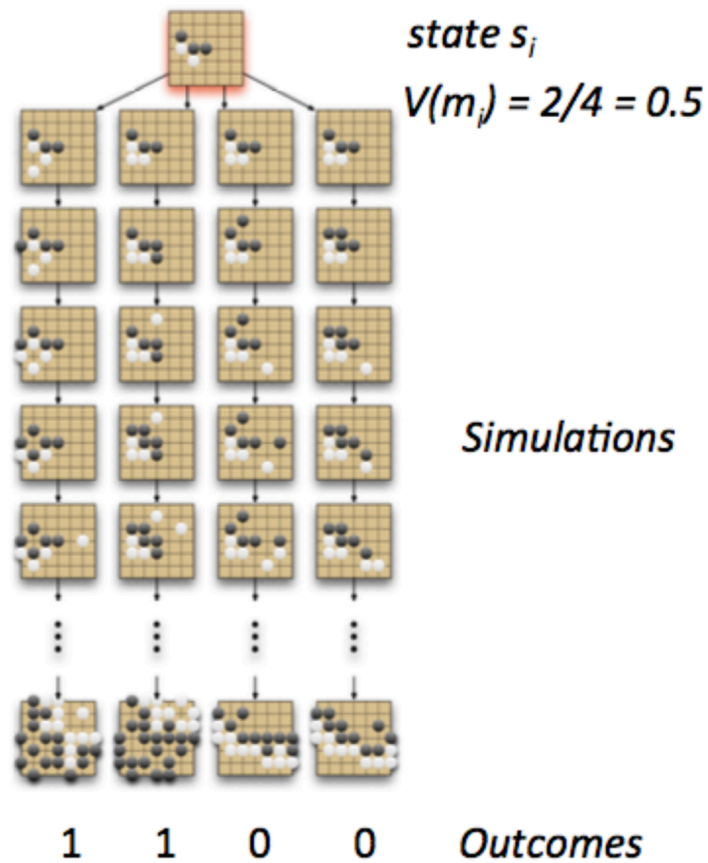
Monte-Carlo Tree Search

- Heuristic search algorithm for decision trees
- Application to deterministic game pretty recent (less than 10 years)



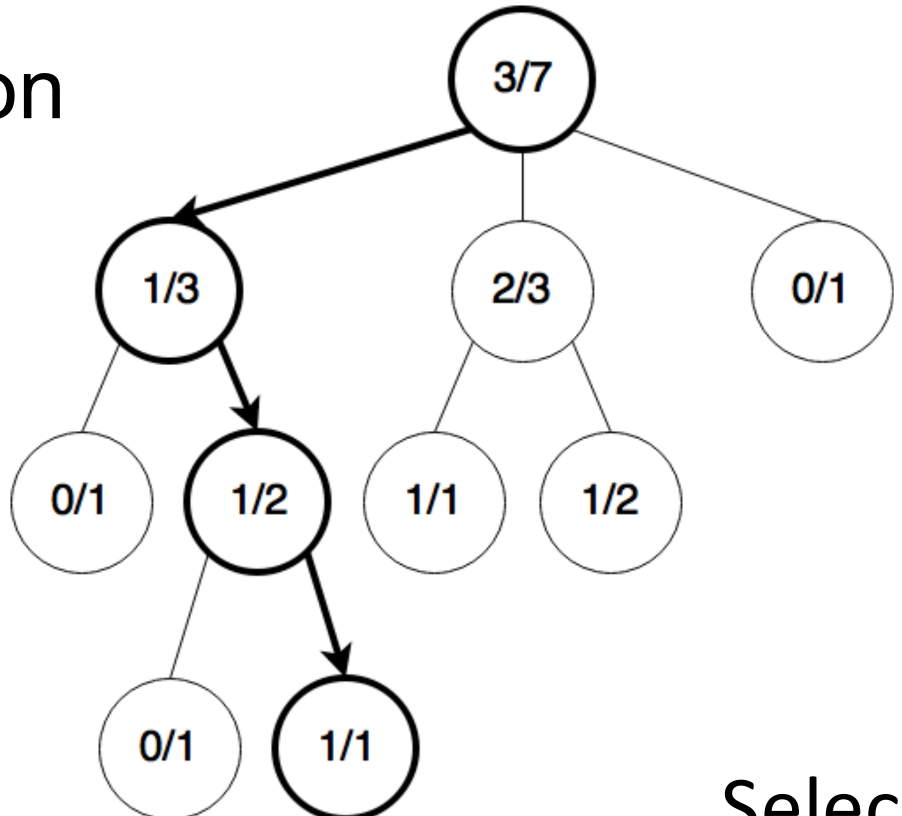
Basic Idea

- No evaluation function?
  - Simulate game using random moves
  - Score game at the end, keep winning statistics
  - Play move with best winning percentage
  - Repeat



Monte Carlo Tree Search

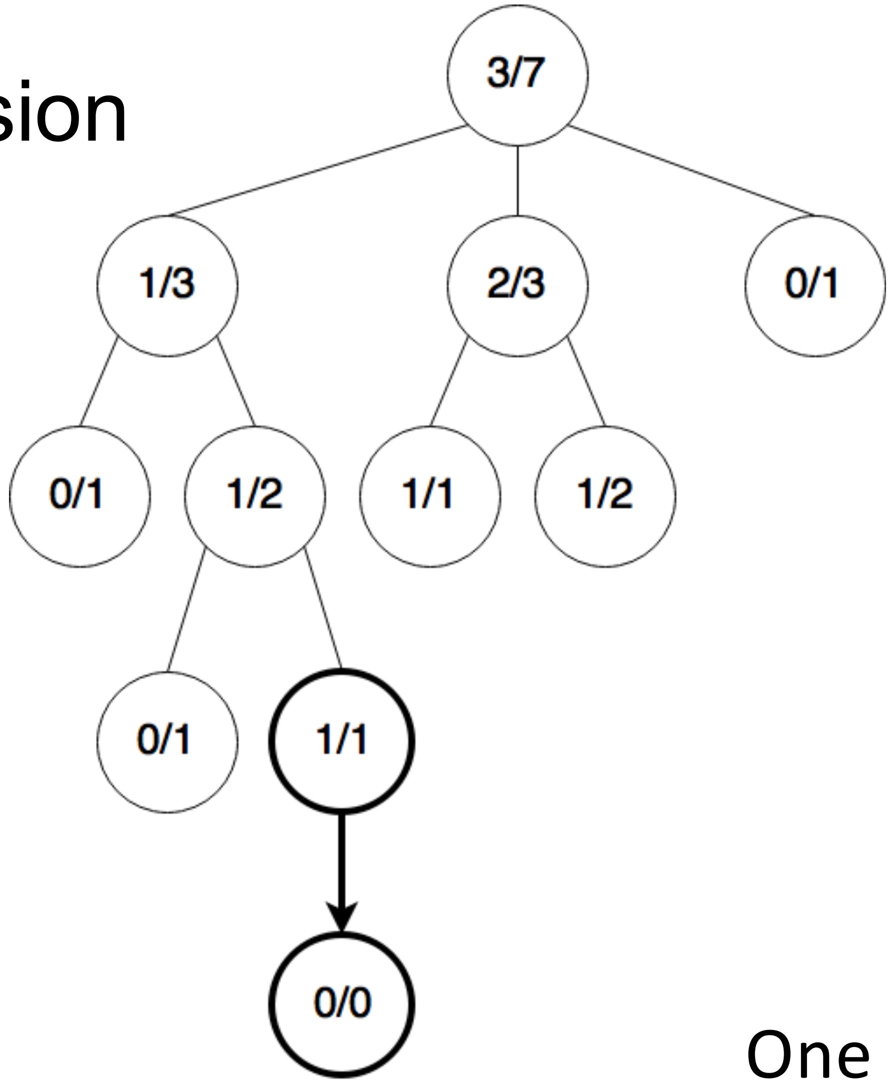
(1) Selection



Selection policy is applied recursively until a leaf node is reached

Monte Carlo Tree Search

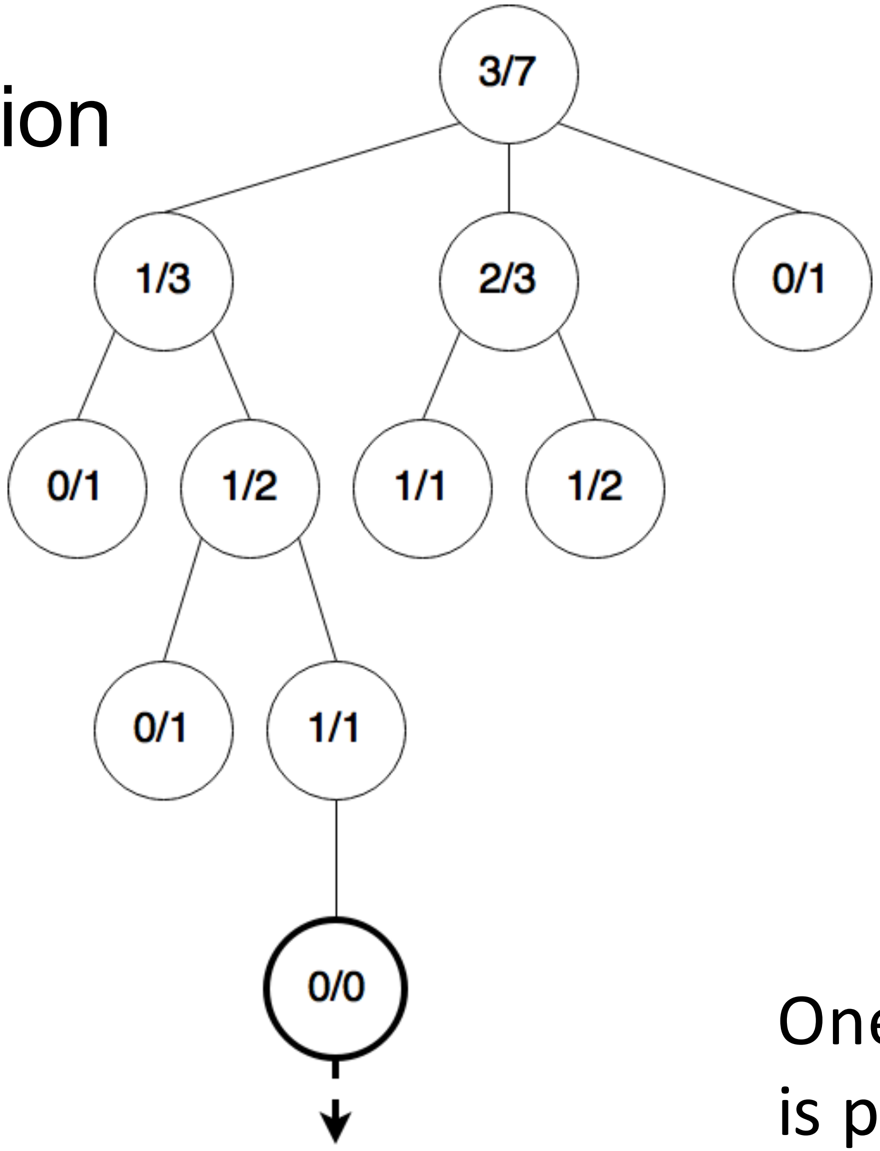
(2) Expansion



One or more nodes are created.

Monte Carlo Tree Search

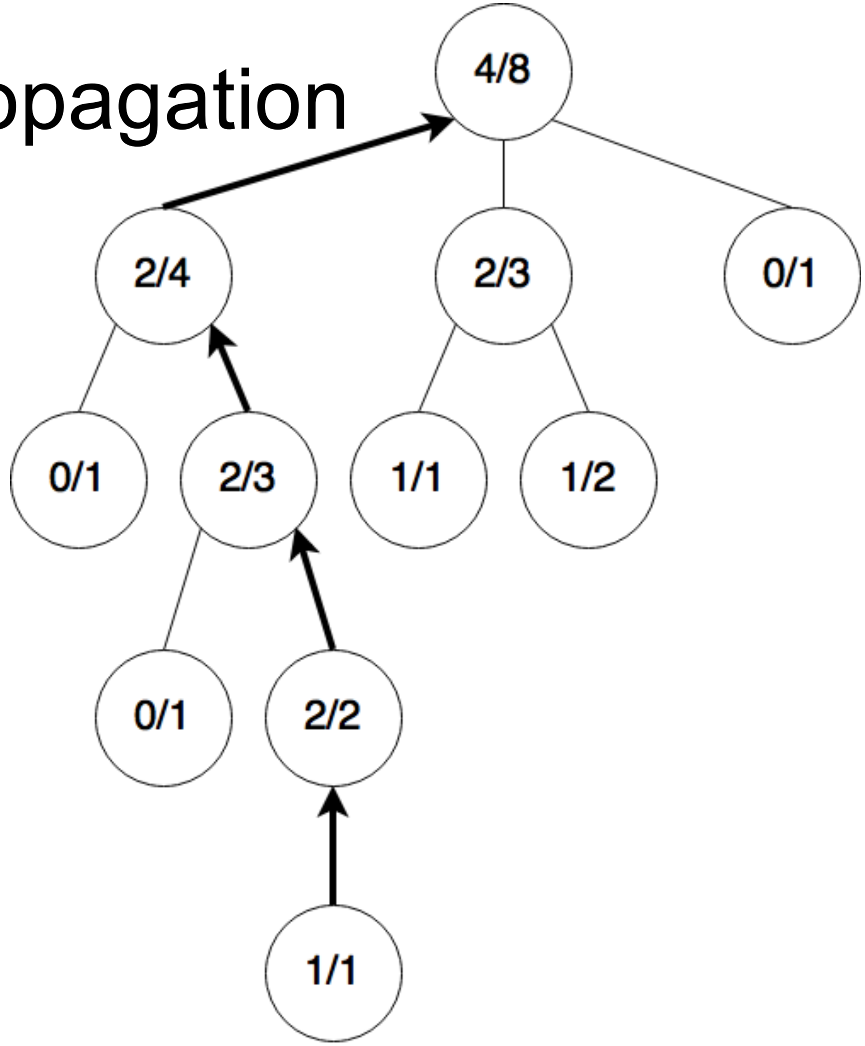
(3) Simulation



One simulated game is played.

Monte Carlo Tree Search

(4) Backpropagation



Naïve Monte Carlo Tree Search

- Use simulation directly as an evaluation function for alpha-beta pruning
- Problems for Go
  - Single simulation is very noisy, only 0/1signal
  - Running many simulations for one evaluation is very slow, e.g., typical speed for chess is 1 million eval/sec, for Go is only 25 eval/sec
- Result: MCTS is ignored for over 10 years in computer Go

Monte Carlo Tree Search

- Use results of simulation to guide the growth of the game tree
- What moves are interesting to us?
  - Promising moves (simulated and won most)
  - Moves where uncertainty about evaluation are high (less simulated)
- Seems two contradictory goals
  - Theory of bandits can help

Multi-Armed Bandit Problem

- Assumptions
  - Choice of several arms
  - Each arm pull is independent of other pulls
  - Each arm has fixed, unknown average payoff
- Which arm has the best average payoff?

Multi-Armed Bandit Problem

P(A wins)=45%

P(B wins)=47%

P(C wins)=30%

- But we don't know the probability, how do we choose a good one?
- With infinite time, we may try each one for infinite times to estimate the probability
- But in practice?

Exploration strategy

- Want to explore all arms
  - We don’t want to miss any potentially good arm
  - But, if we explore too much, may sacrifice the reward we could have gotten
- Want to exploit promising arms more often
  - Good arms worth further investigation
  - But, if we exploit too much, may get stuck with sub-optimal values

Upper Confidence Bound

- Policy
  – First, try each arm once
  – Then, at each time step
    - Choose the arm that maximizes formula:

$$v_i + C \times \sqrt{\frac{\ln(N)}{n_i}}$$

total number of trials

value estimate

tunable parameter

num trials for arm i

Prefers higher payoff arm

Prefers less played arm

Schedule

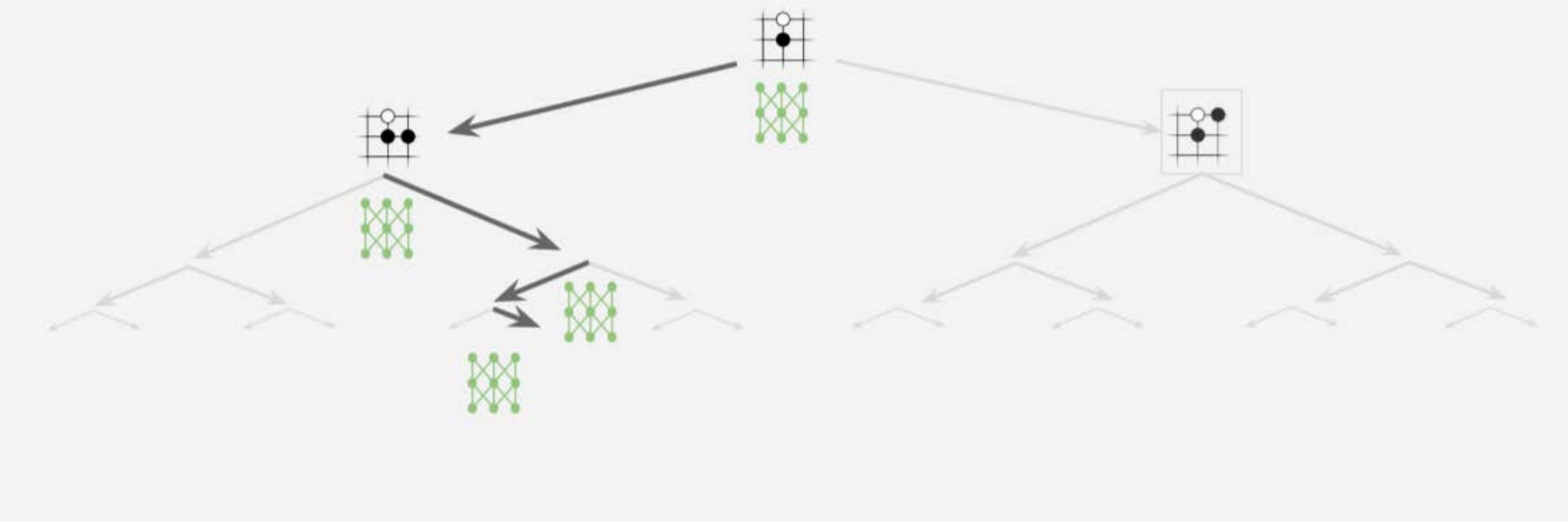
- Introduction
- Monte-Carlo Tree Search
- **Policy and Value Networks**
- Results

Policy and Value Networks

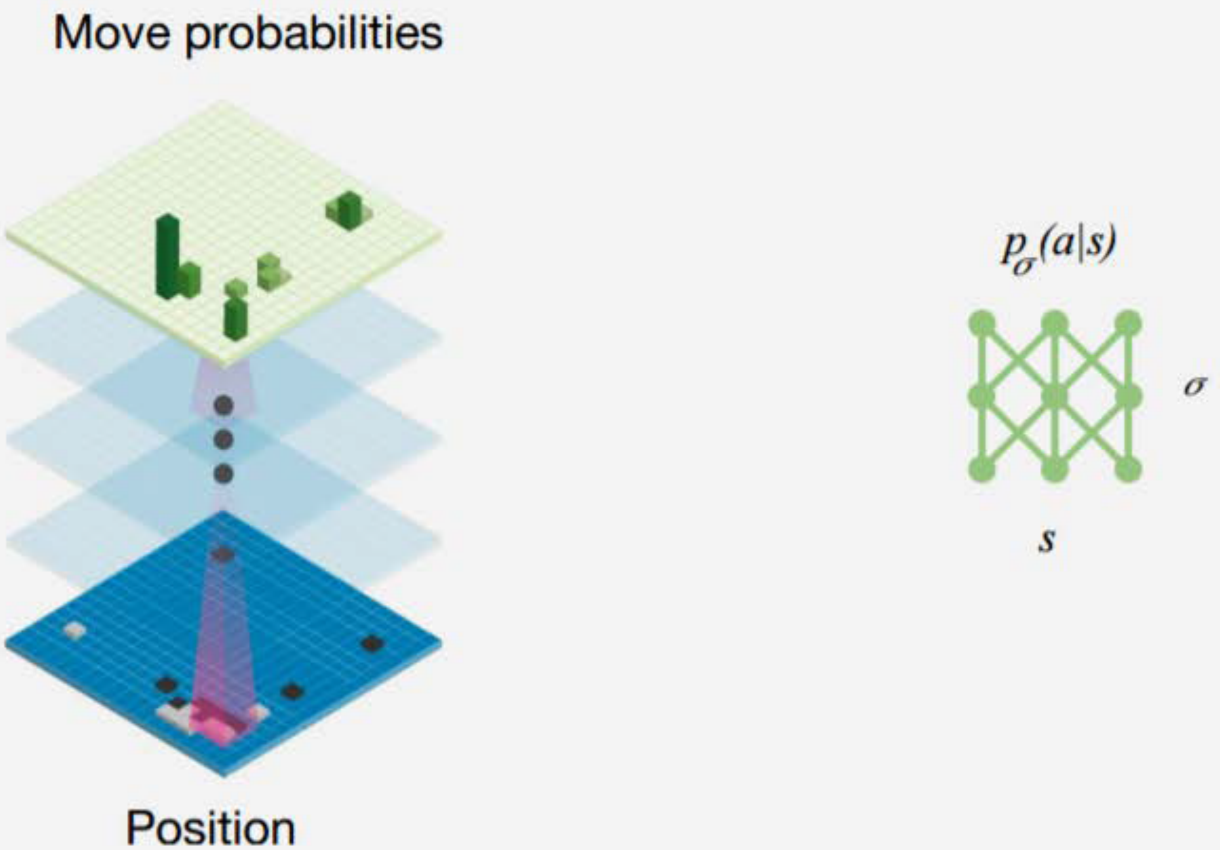
- Goal: Reduce both branching factor and depth of search tree
- How?
  - Use policy network to explore better (and fewer) moves
    - How?
  - Use value network to estimate lower branches of tree (rather than simulating to the end)
    - How?

Policy and Value Networks

- Reducing branching factor: Policy Network



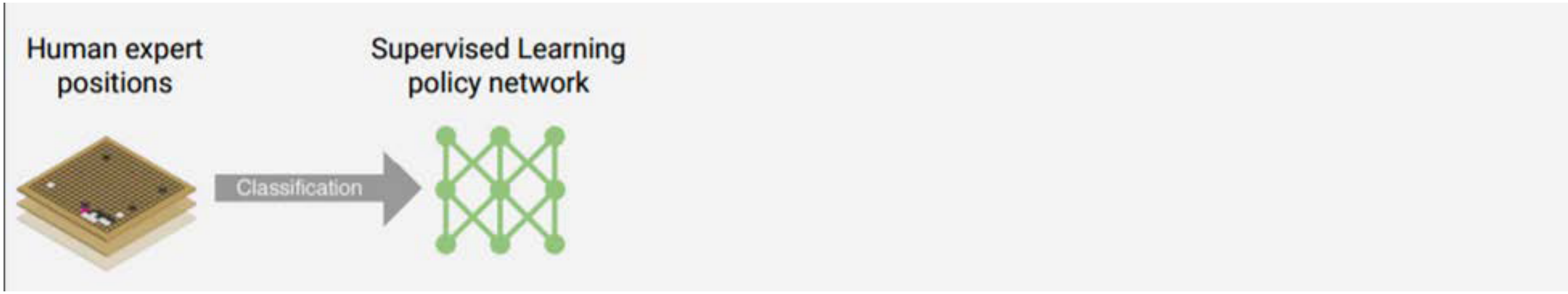
Policy and Value Networks



Predicts the probability of a move being best move

Policy and Value Networks

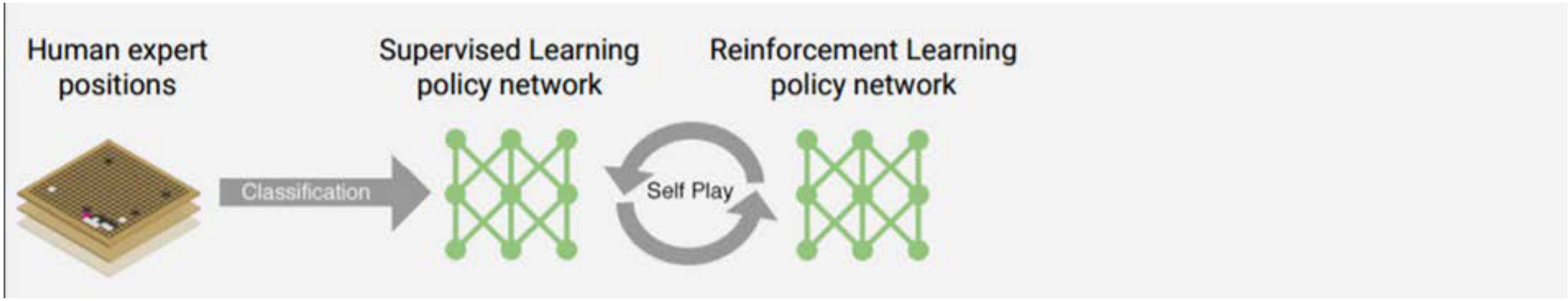
- Supervised learning



- Training data: 30 million positions from human expert games
- Likelihood of a human move selected at a state *s*
- Training time: 4 weeks
- Results: predicted human expert moves with 57% accuracy

Policy and Value Networks

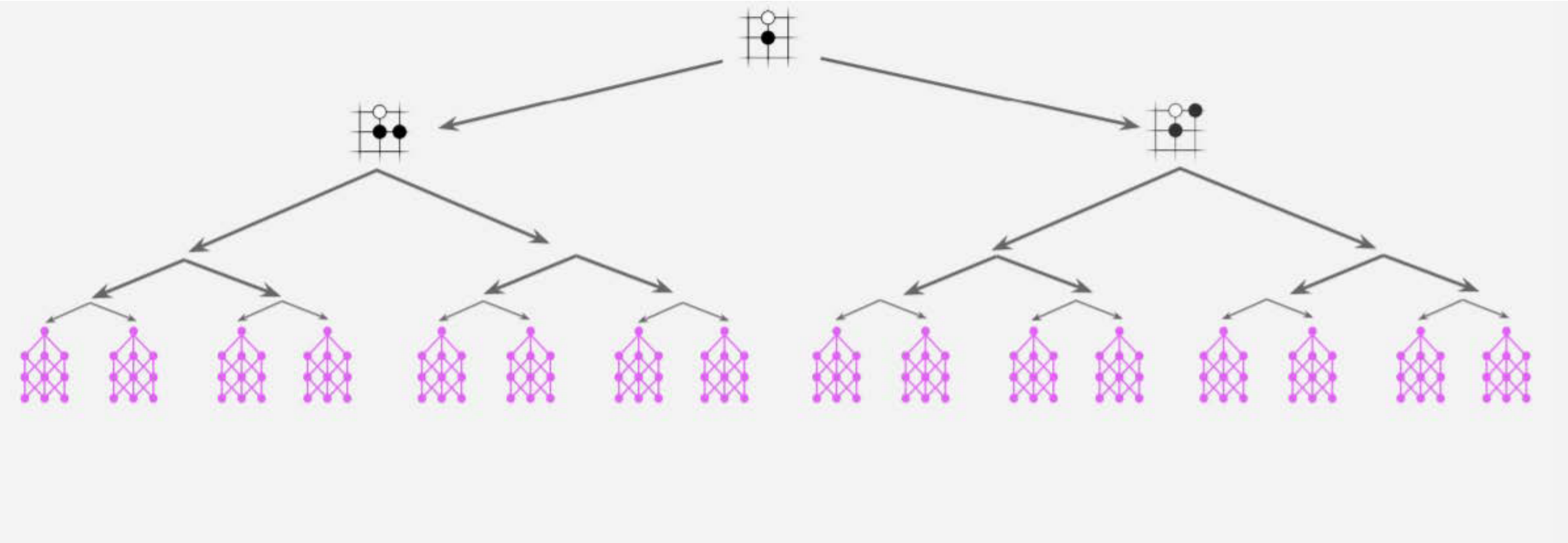
- Reinforcement learning



- Training data: 128,000+ games of self-play using policy network in 2 stages
- Training algorithm: maximize wins of the action $\Delta\sigma$
- Training time: 1 week
- Results: won more than 80% games vs. supervised learning

Policy and Value Networks

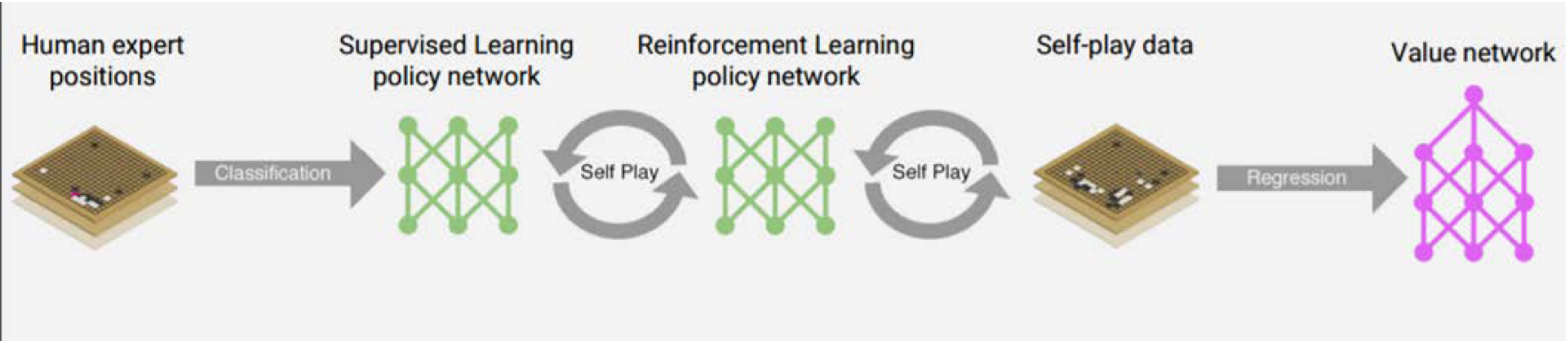
- Reducing depth: Value Network



- Given board states, estimate probability of victory
- No need to simulate to the end of the game

Policy and Value Network

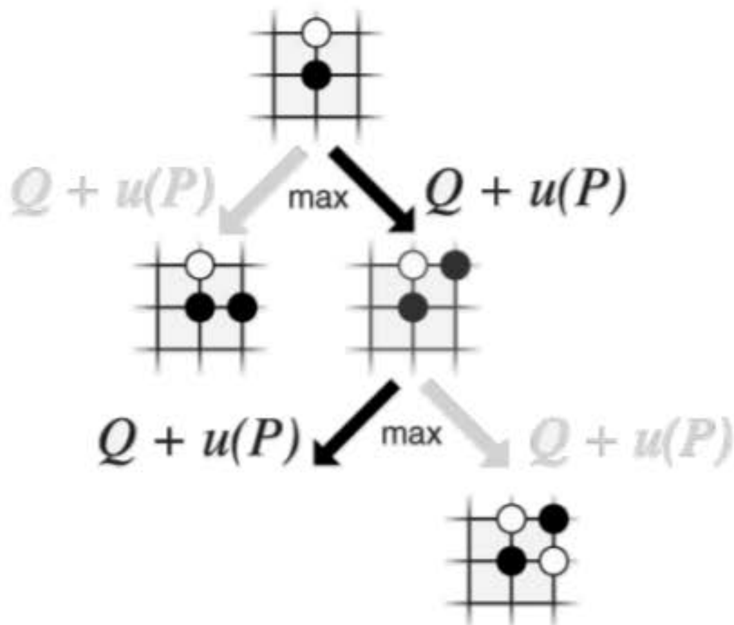
- Reinforced learning



- Training data: 30 million games of self-play
- Training algorithm: minimize mean-squared error by stochastic gradient descent
- Training time: 1 week
- Results: AlphaGo ready for playing against pros

MCTS + Policy / Value Networks

- Selection



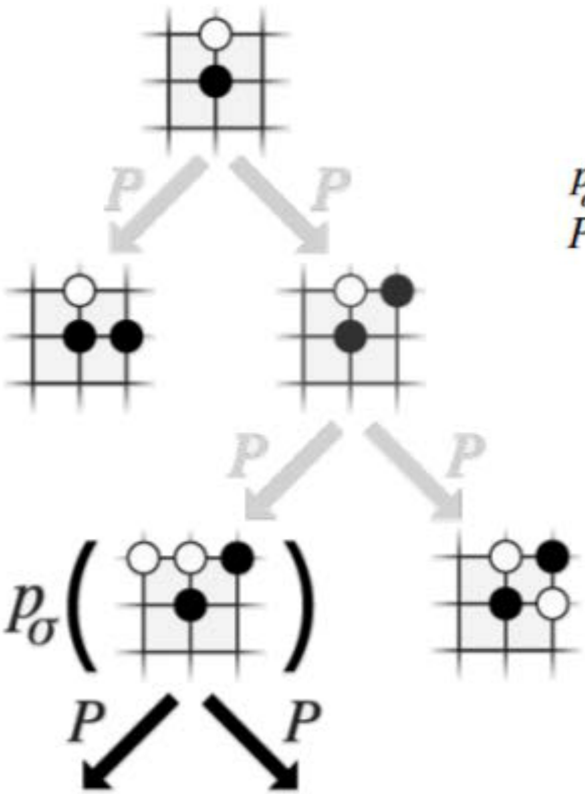
P prior probability
Q action value

Q+u(P)

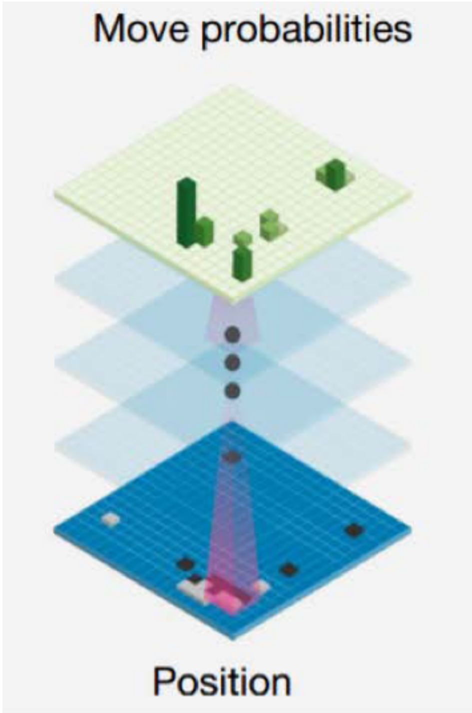
- Initially no simulation yet, so action value = 0, prefers high prior probability and low visits count
- Asymptotically, prefers actions with high action value.

MCTS + Policy / Value Networks

- Expansion

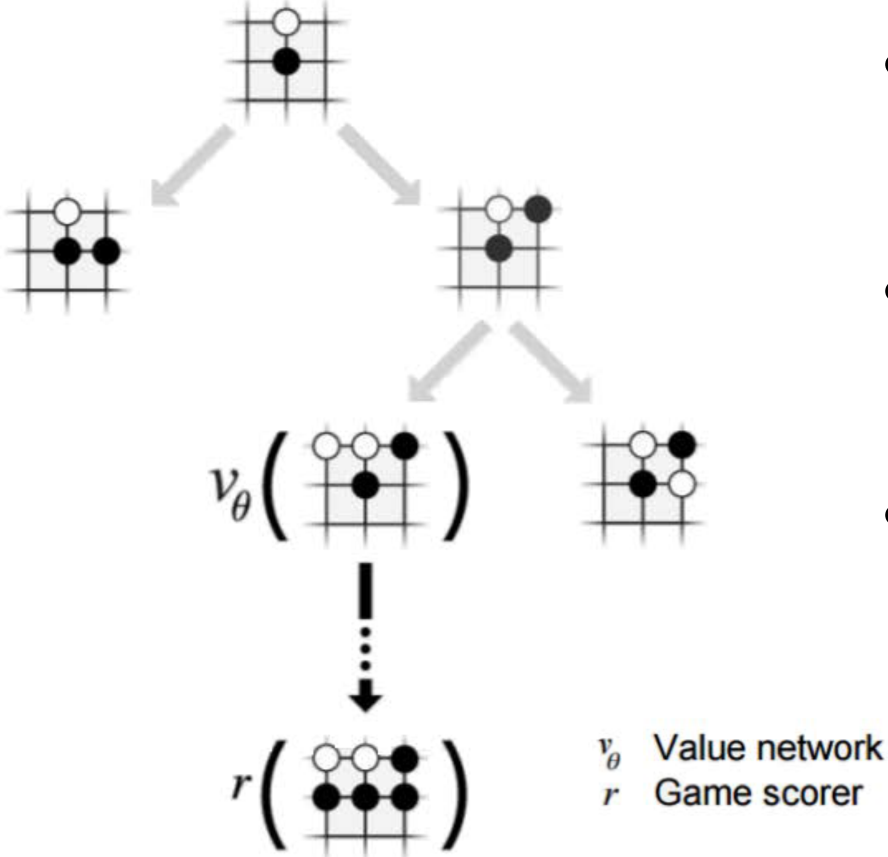


p_σ Policy network
P prior probability



MCTS + Policy / Value Networks

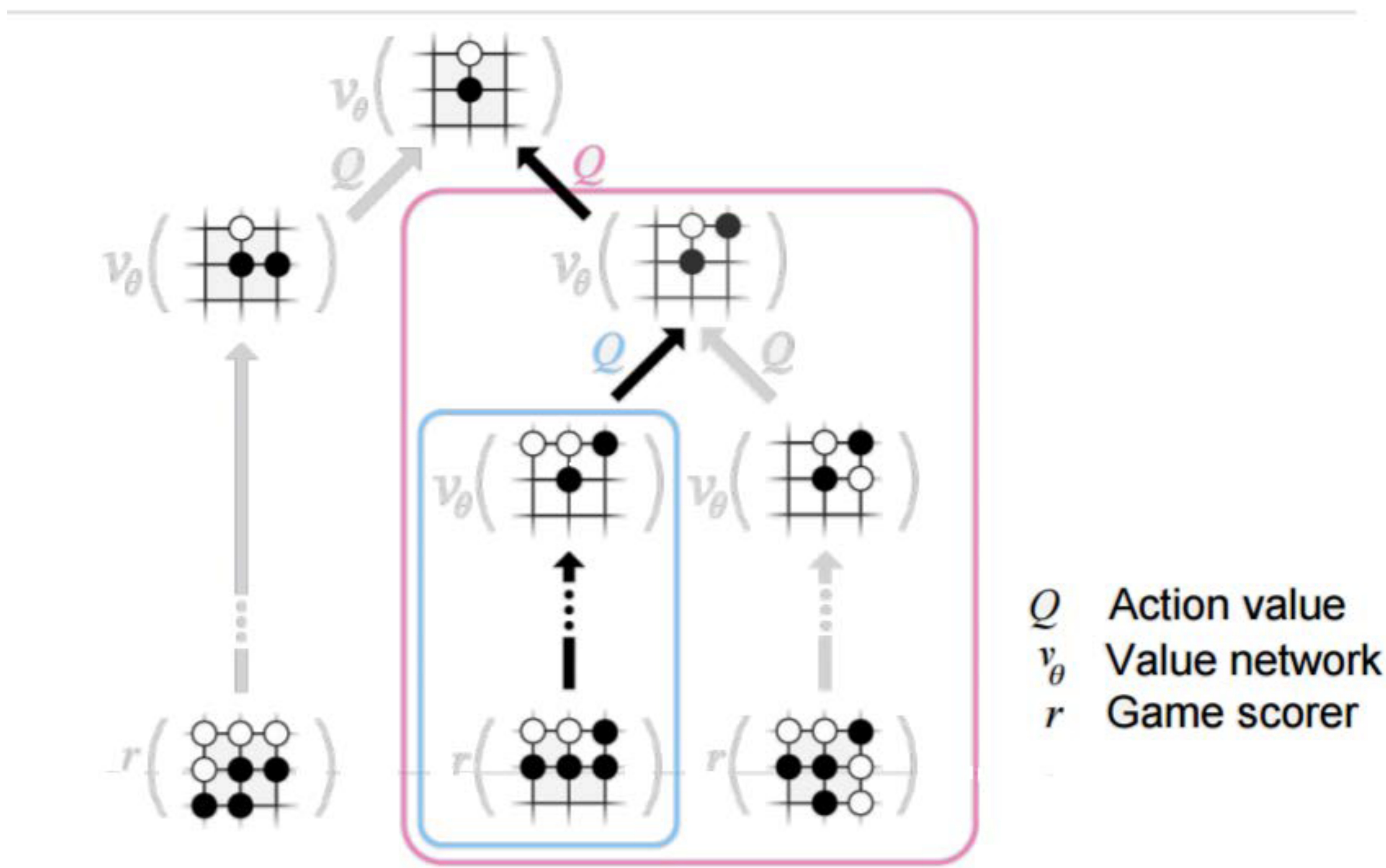
- Simulation



- Run multiple simulations in parallel
- Some with value network
- Some with rollout to the end of the game

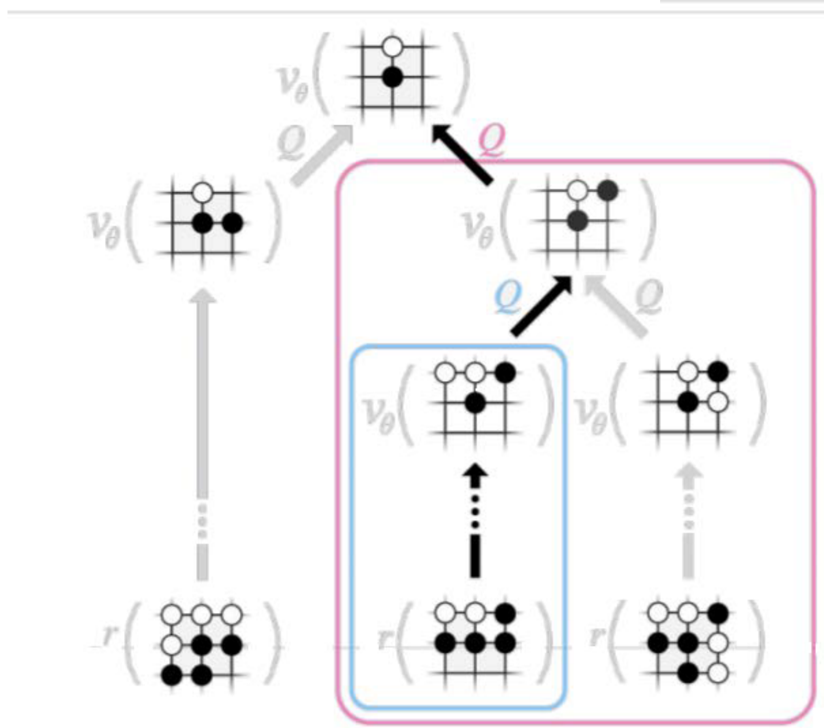
MCTS + Policy / Value Networks

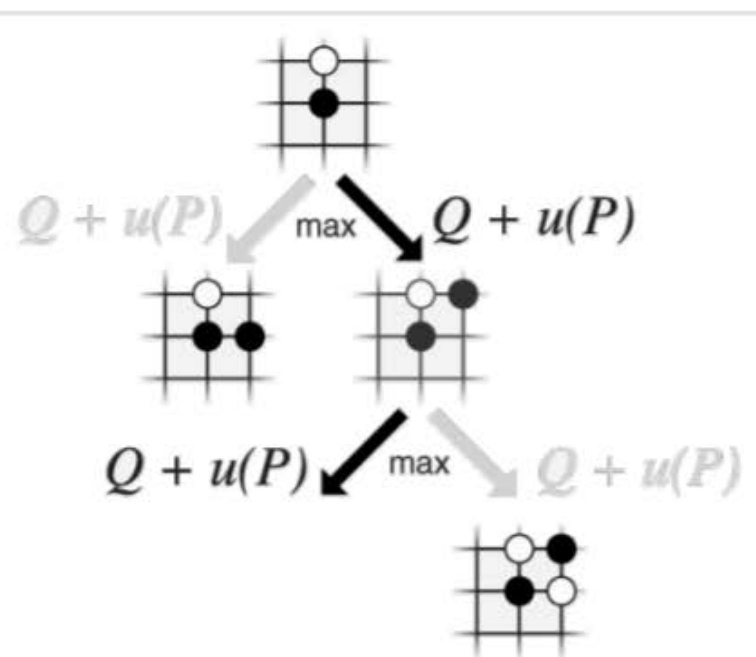
- Propagate values back to root



MCTS + Policy / Value Networks

- Repeat





Selection

AlphaGo Zero

- AlphaGo
  - Supervised learning from human expert moves
  - Reinforcement learning from self-play
- AlphaGo Zero
  - Solely reinforcement learning from self-play

AlphaGo Zero

- Beats AlphaGo by 100:0

What's next for AI?

Go is still in the "easy" category of AI problems.

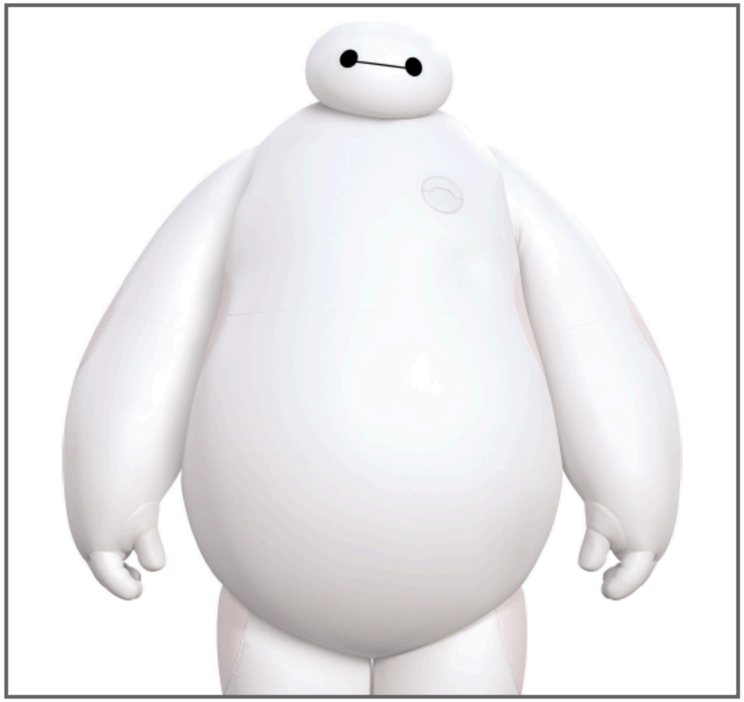
- **Fully observable** vs. partially observable
- Single agent vs. **multiagent**
- **Deterministic** vs. stochastic
- Episodic vs. **sequential**
- **Static** vs. dynamic
- **Discrete** vs. continuous
- **Known** vs. unknown

What's next for AI?

DeepMind's AI is Struggling to Beat Starcraft II - Bloomberg
https://www.bloomberg.com/.../deepmind-master-of-go-struggles-to-crack-its-next-mi...

What's next for AI?

The idea of combining search with learning is very general and is widely applicable.



References

- Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.
- Silver, David, et al. "Mastering the game of go without human knowledge." *Nature* 550.7676 (2017): 354-359.
- Introduction to Monte Carlo Tree Search, by Jeff Bradberry https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/