Deep Reinforcement Learning

[Mastering the Game of Go with Deep Reinforcement Learning and Tree Search, Nature 2016]

CS 486/686

University of Waterloo

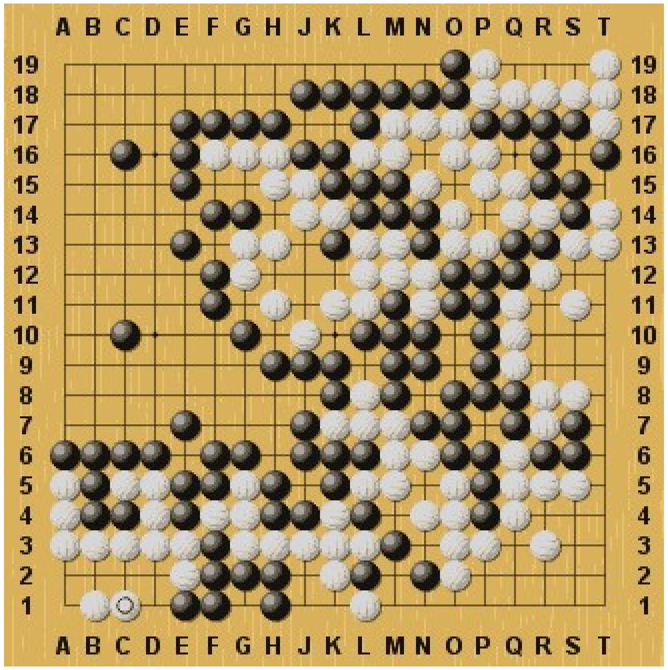
Lecture 21: July 12, 2017

Outline

- AlphaGo
  - Supervised Learning of Policy Networks
  - Reinforcement Learning of Policy Networks
  - Reinforcement Learning of Value Networks
  - Searching with Policy and Value Networks

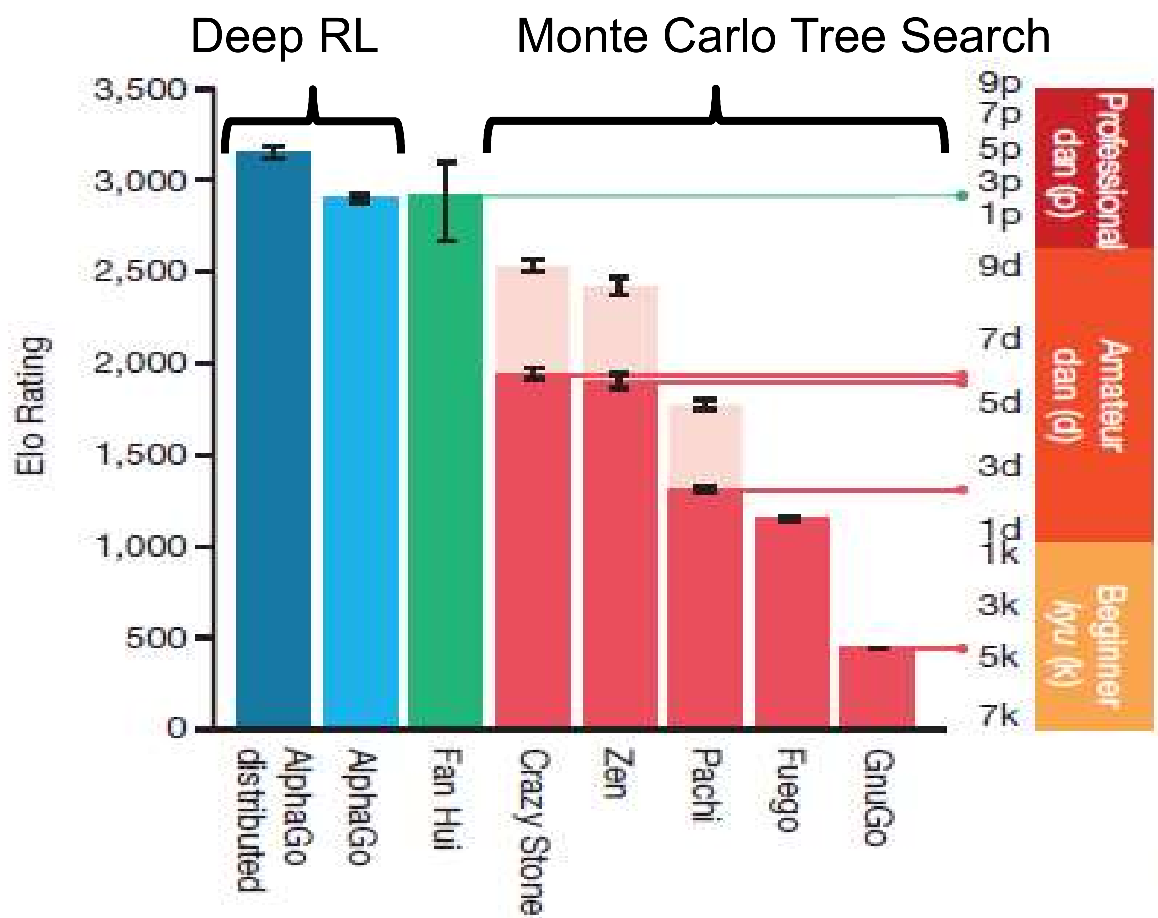
Game of Go

- (simplified) rules:
  - Two players (black and white)
  - Players alternate to place a stone of their color on a vacant intersection.
  - Connected stones without any liberty (i.e., no adjacent vacant intersection) are captured and removed from the board
  - Winner: player that controls the largest number of intersections at the end of the game



Computer Go

- Oct 2015:



- March 2016: AlphaGo defeats Lee Sedol (9-dan)

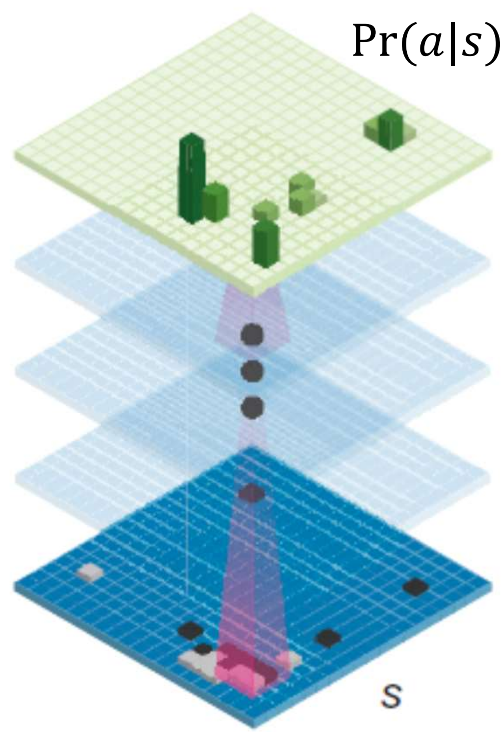
Winning Strategy

- Four steps:

1. Supervised Learning of Policy Networks
2. Reinforcement Learning of Policy Networks
3. Reinforcement Learning of Value Networks
4. Searching with Policy and Value Networks

Policy Network

- Train policy network to imitate Go experts based on a database of 30 million board configurations from the KGS Go Server.
- Policy network: $\Pr(a|s)$
  - Input: state s (board configuration)
  - Output: distribution over actions a (intersection on which the next stone will be placed)



Supervised Learning of the Policy Network

- Let $\boldsymbol{w}$ be the weights of the policy network

- Training:
  - Data: suppose a is optimal in s
  - Objective: maximize $\log \Pr_{\boldsymbol{w}}(\mathrm{a}|\mathrm{s})$
  - Gradient: $\nabla w = \frac{\partial \log \Pr_{\boldsymbol{w}}(a|s)}{\partial \boldsymbol{w}}$
  - Weight update: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \nabla \boldsymbol{w}$

Reinforcement Learning of the Policy Network

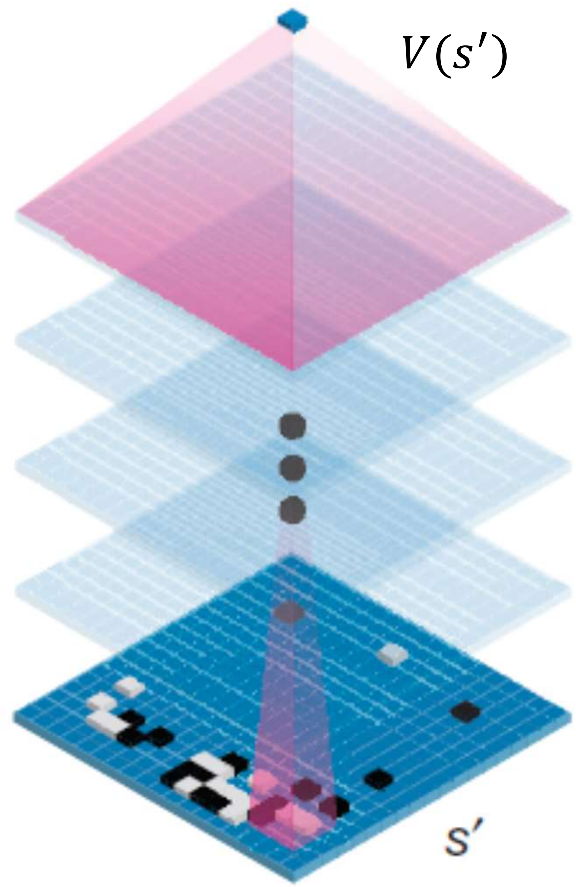
- How can we update a policy network based on reinforcements instead of the optimal action?
- Let $R = \sum_t \gamma^t r_t$ be the discounted sum of rewards in a trajectory that starts in s by executing a.
- Gradient: $\nabla \boldsymbol{w} = \frac{\partial \log Pr_{\boldsymbol{w}}(a|s)}{\partial \boldsymbol{w}} R$
  - Intuition rescale supervised learning gradient by R
  - Formally: see derivation in [Sutton and Barto, Reinforcement learning, Chapter 13]
- Weight update: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \nabla \boldsymbol{w}$

Reinforcement Learning of the Policy Network

- In computer Go, program repeatedly plays games against its former self.
- For each game $R = \begin{cases} 1 & win \\ -1 & lose \end{cases}$
- For each (s_t, a_t) of turn t of the game, compute
  - Gradient: $\nabla \boldsymbol{w} = \frac{\partial \log Pr_{\boldsymbol{w}}(a_t|s_t)}{\partial \boldsymbol{w}} R$
  - Weight update: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \nabla \boldsymbol{w}$

Value Network

- Predict $V(s')$ (i.e., who will win game) in each state s' with a value network
  - Input: state s (board configuration)
  - Output: expected discounted sum of rewards $V(s')$



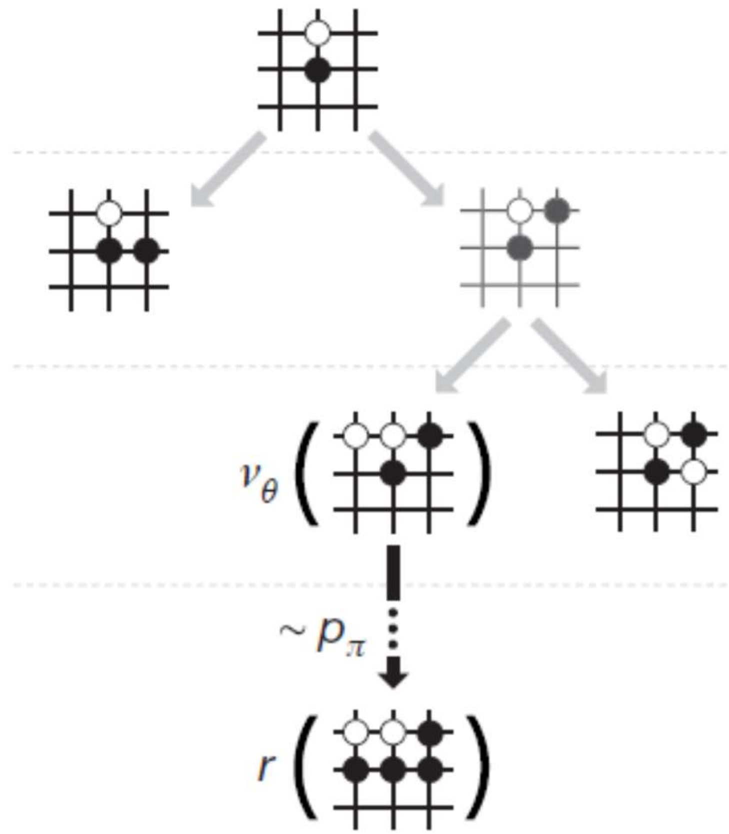
Reinforcement Learning of Value Networks

- Let $\boldsymbol{v}$ be the weights of the value network
- Training:
  - Data: (s, R) where $R = \begin{cases} 1 & win \\ -1 & lose \end{cases}$
  - Objective: minimize $\frac{1}{2}(V_{\boldsymbol{v}}(s) - R)^2$
  - Gradient: $\nabla \boldsymbol{v} = \frac{\partial V_{\boldsymbol{v}}(s)}{\partial \boldsymbol{v}}(V_{\boldsymbol{v}}(s) - R)$
  - Weight update: $\boldsymbol{v} \leftarrow \boldsymbol{v} - \alpha \nabla \boldsymbol{v}$

Searching with Policy and Value Networks

- AlphaGo combines policy and value networks into a Monte Carlo Tree Search algorithm

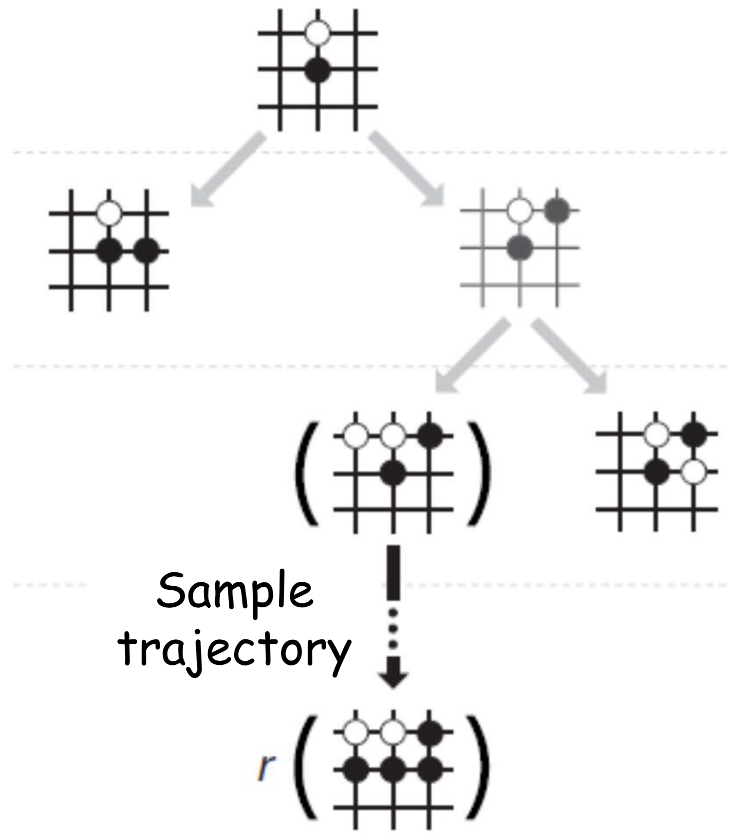
- Idea: construct a search tree
  - Node: s
  - Edge: a



Search Tree

- At each edge store
  $Q(s,a)$, $\Pr_{\boldsymbol{w}}(a|s)$, $N(s,a)$

- Where $N(s,a)$ is the visit count of (s,a)



Simulation

- At each node, select edge a^* that maximizes

$$a^* = argmax_a\, Q(s,a) + u(s,a)$$

- where $u(s,a) \propto \frac{P(S|a)}{1+N(s,a)}$ is an exploration bonus

$$Q(s,a) = \frac{1}{N(s,a)} \sum_i 1_i(s,a)\, [\lambda V_v(s) + (1-\lambda) R_i]$$

$$1_i(s,a) = \begin{cases} 1 & if\ (s,a)\ was\ visited\ at\ iteration\ i \\ 0 & otherwise \end{cases}$$

Competition

Extended Data Table 1 | Details of match between AlphaGo and Fan Hui

Date	Black	White	Category	Result
5/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by 2.5 points
5/10/15	Fan Hui	*AlphaGo*	Informal	Fan Hui wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Informal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
9/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
9/10/15	*AlphaGo*	Fan Hui	Informal	Fan Hui wins by resignation

The match consisted of five formal games with longer time controls, and five informal games with shorter time controls. Time controls and playing conditions were chosen by Fan Hui in advance of the match.