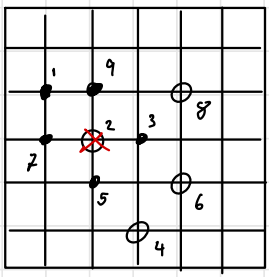
Lecture 20

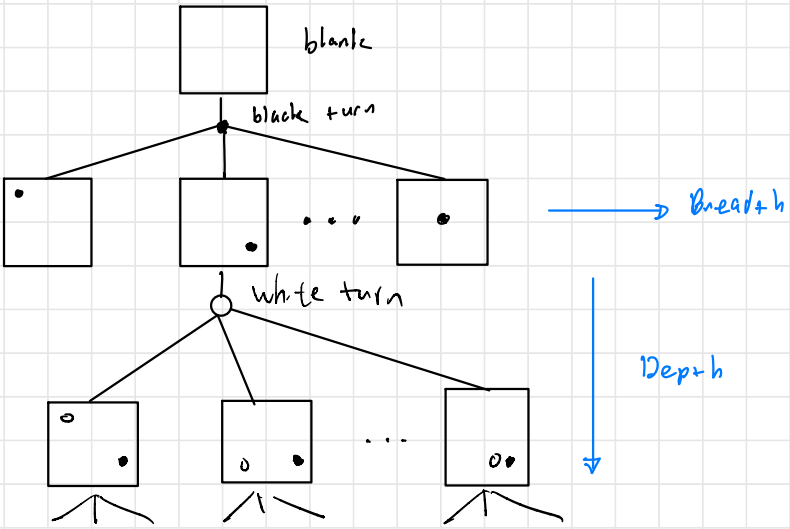
- Alpha Go :

  - Go: game



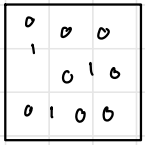
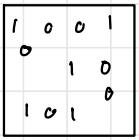
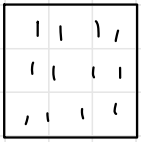
Player 1 : Black stone ●
Player 2 : White stone ○

- Game Tree :



- End Result : $Z = \begin{cases} +1 & \text{black wins} \\ -1 & \text{white wins} \\ 0 & \text{draw} \end{cases}$ ● ○

- The Game Tree is extremely big

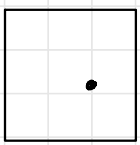
- State s = ( ,  , )

19 × 19

1 : position of black stone

1 : white stones

all 1's if black
all 0's if white

- action : a = 

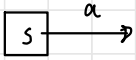
19 × 19 + 1 = 362

↓

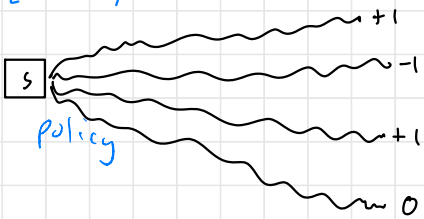
pass

- Policy : $p(a|s)$ classification
(actor)

category Image

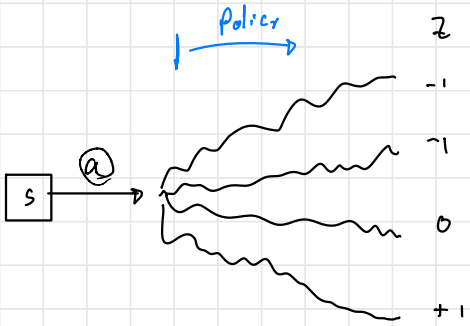


- Value : $v_{policy}(s) = \mathbb{E}_{policy}(z|s)$
(critic)



Action Value

- $Q_{policy}(s, a) = \mathbb{E}_{policy}[Z \mid s, a]$



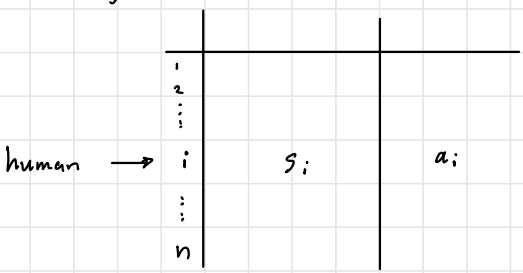
- Optimal Policy:

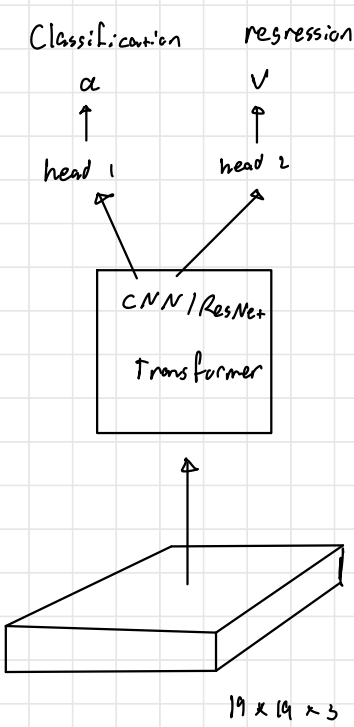
$$\max_{policy} V_{policy}(s) = V^*(s)$$

$$\max_{policy} Q_{policy}(s, a) = Q^*(s, a)$$

- Step 1 : Supervised Learning

Training Data





Supervised Policy Network $P_\sigma(a|s)$

$$\max_\sigma \frac{1}{n} \sum_{i=1}^{n} \log P_\sigma(a_i|s_i)$$

MLE

$$SGD: \quad \Delta\sigma \propto \frac{\partial}{\partial\sigma} \log P_\sigma(a|s)$$

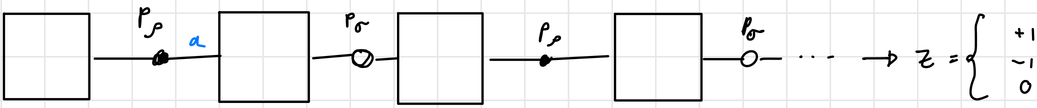
↑ generated by Teacher

Imitation Learning
Behavior Cloning

- Step 2 : Reinforcement Learning : $p_\rho(a|s)$

  ↓ reinforcement

- initialize $\rho = \sigma$
- improve ρ while fixing σ



RL:

self

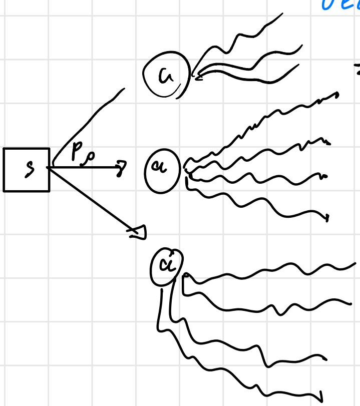
$$\Delta \rho \propto \frac{\partial}{\partial \rho} \log p_\rho(a|s)\, z$$

consequence

reinforcement

- If (a) leads to a win
  increase probability

- If (a) leads to a loss
  decrease probability



$$\max_\rho \mathbb{E}_\rho[z] \quad , \quad \mathbb{E}_\rho[z] = \sum_a z\, p_\rho(a|s)$$

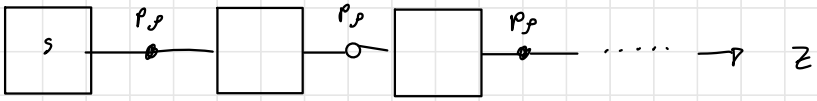
- $$\frac{\partial}{\partial \rho} \mathbb{E}_\rho(z) = \sum_a z \left[\frac{\partial}{\partial \rho} p_\rho(a|s) \right]$$

$$= \sum_a z \left[\frac{\partial}{\partial \rho} \log p_\rho(a|s) \right] \times p_\rho(a|s)$$

$$= \mathbb{E}_\rho \left[z \frac{\partial}{\partial \rho} \log p_\rho(a|s) \right]$$

Policy Gradient

- Step 3: $V_\theta(s)$



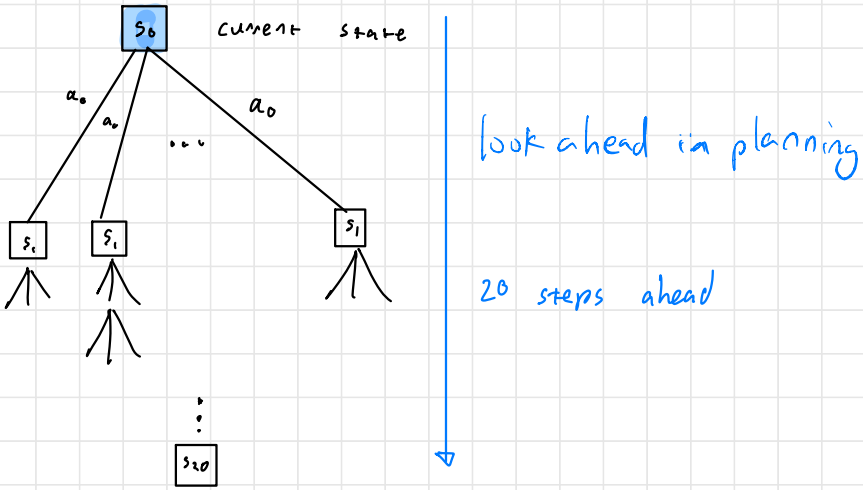
$$\Delta\theta \propto -\frac{\partial}{\partial\theta}\left(z - V_\theta(s)\right)^2$$

- Learned $V_\theta(s)$ more accurate for s close to the end

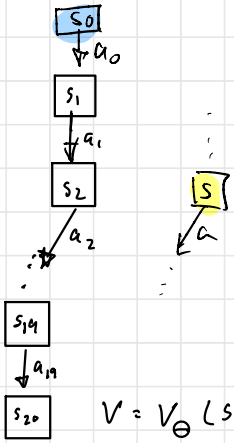
- Go to South Korea:

- Step 4: Monte Carlo Tree Search (MCTS)

  - Real game v.s. Top Player



- Keep Sampling Branches:



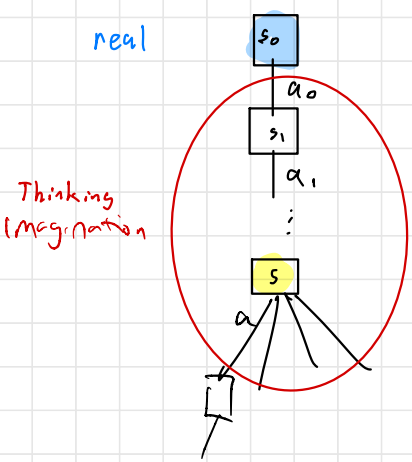
$V = V_\theta(s_{20})$ more accurate than $V_\theta(s_0)$

- For each (s, a) an branch

  - visit #: $N(s,a) \leftarrow N(s,a) + 1$
  - total score: $W(s,a) \leftarrow W(s,a) + V_\theta(s_{20})$

$$Q(s,a) = \frac{W(s,a)}{N(s,a)}$$

- How to sample branch ?



policy: reduce search breadth
value: reduce search depth

exploitation exploration

$$\max_a \left[Q(s,a) + C \cdot \frac{P_\sigma(a|s)}{N(s,a)+1} \right]$$

- After sampling many branches,
  choose a_0 : $\max_{a_0} Q(s_0, a_0)$

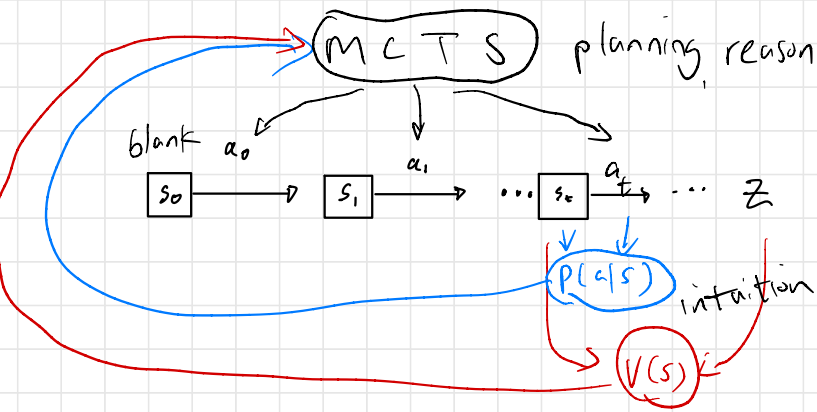
  or

  $\pi(a_0 | s_0) \propto N(s_0, a_0)^{\frac{1}{\tau}}$ ← Temperature

  throw away tree.

- Alpha Go Zero
  ↓
  No human

- In England: (Self-Play)



Markov decision Process { dynamics $p(s_{t+1} | s_t, a_t)$; reward $p(r_t | s_t, a_t, s_{t+1})$

$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \rightarrow s_t \xrightarrow{a_t} s_{t+1} \xrightarrow{a_{t+1}} \cdots$ model-based / model-free

$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$

$\max E_{policy}(G_t)$

recursive $G_t = r_t + \gamma G_{t+1}$

$= r_t + \gamma r_{t+1} + \gamma^2 G_{t+2}$

temporal difference