Mastering the game of Go without human knowledge

강동구

목차

배경지식

1. Policy iteration

- policy evaluation : 현재 policy로 value func 추정
- policy improvement : 현재 value func로 더 나은 policy 추출
- 알파고 제로는 MCTS로 policy iteration하는 거임
- 근데 더 쎔(Stronger)

Why AlphaGo Zero is amazing

1. Learn from self-play RL
2. Starting from random initial weight
3. No human supervision only raw board history as input
4. Single machine with 4TPU

And super-human performance

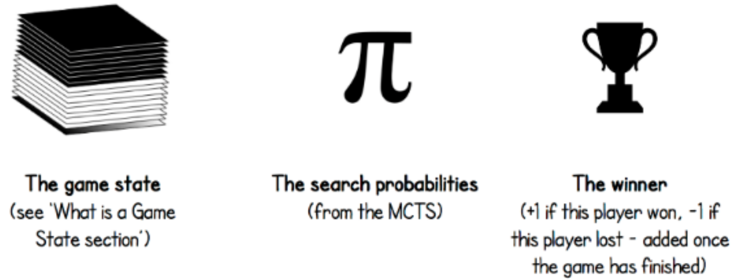
Difference

1. No rollout
2. Single neural network (PN & VN)
3. Leaf nodes are always expanded rather than dynamic expansion
4. Each search thread waits for the nn-evaluation rather than asynchronously evaluation & backup
5. No tree policy

How it works

1. self-play by MCTS

creating training set
best current player plays 25,000 games against itself
At each move, store (s,π,z)

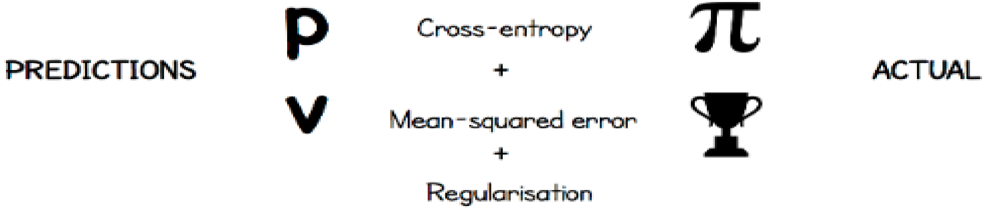


1. Neural network training $(\boldsymbol{p}, v) = f_\theta(s)$

optimize the nn weight

sample a mini-batch 2,048 positions form last 500,000 games

Loss :



every 1,000
training loop,
evaluate network

How it works

3. Evaluate Network

Test to see if the new network is stronger!

play 400 games between the latest nn and the current best nn

both players use MCTS to select their moves

latest player must win 55%, become current best nn

1, 2, 3 is executed in parallel

Details : What is game state?

19*19*17 image stack

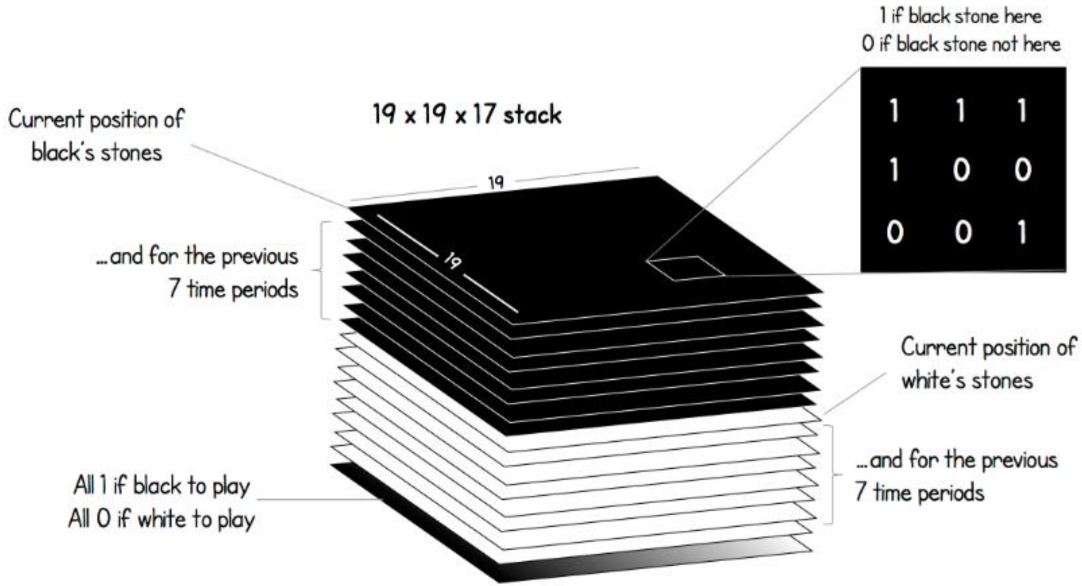
$$s_t = [X_t, Y_t, X_{t-1}, Y_{t-1}, \dots, X_{t-7}, Y_{t-7}, C]$$

17 is binary feature

8 * X_t : 현재 플레이어의 돌이 놓여있으면 1, 비거나 상대면 0

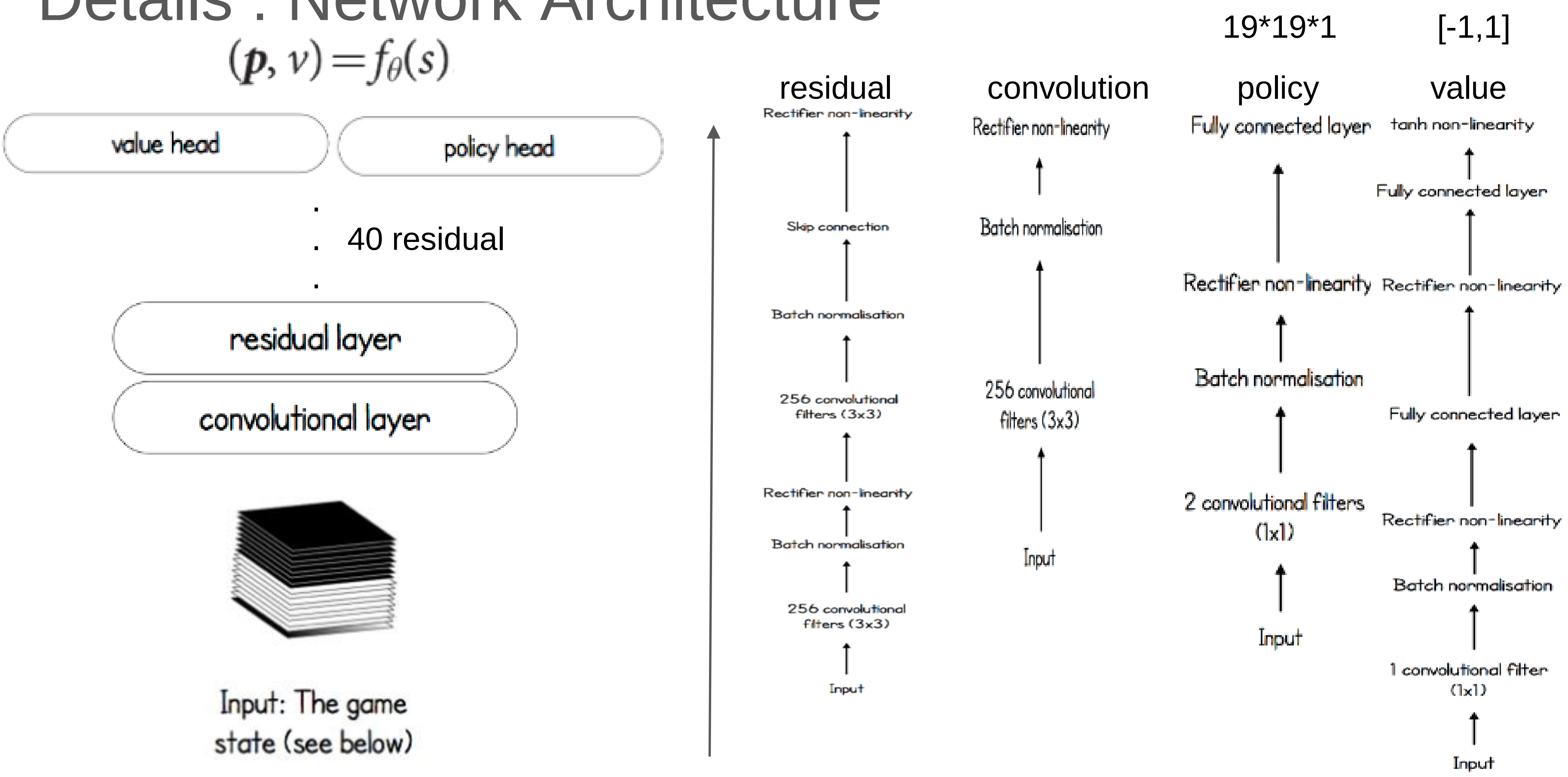
8 * Y_t : 상대 플레이어의 돌이 놓여있으면 1, 비거나 상대면 0

C : 두어야 할 색깔 1: black, 0 : white



Details : Network Architecture

$$(\boldsymbol{p}, v) = f_\theta(s)$$



Details

1. Game terminate when both players pass
   or after 19 * 19 * 2 = 722 move
2. 바둑의 rule은 회전해도 변하지 않기 때문에 바둑판을 rotation, reflection한 데이터를 학습에 사용함
3. Tromp-Taylor scoring during MCTS simulation and self-play
   (왜냐하면, K,J,C의 human score is not well defined)
   using Chinese rule

Optimization f_{θ_i} 2. NN training

1. 64 GPU workers, 19 CPU parameter servers
2. batch-size : 32 per worker, so total 64*32=2,048 mini-batch size from 500,000 self-play game
3. SGD with momentum(0.9) and learning rate annealing(0.1~0.001) using the loss. (c = 10^-4)

$$l = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2 \qquad (1)$$

1. chekpoint every 1,000 training steps.
   checkpoint is evaluated by evaluator and used for generating
   next batch of self-play games

Evaluator   3. Evaluate Network

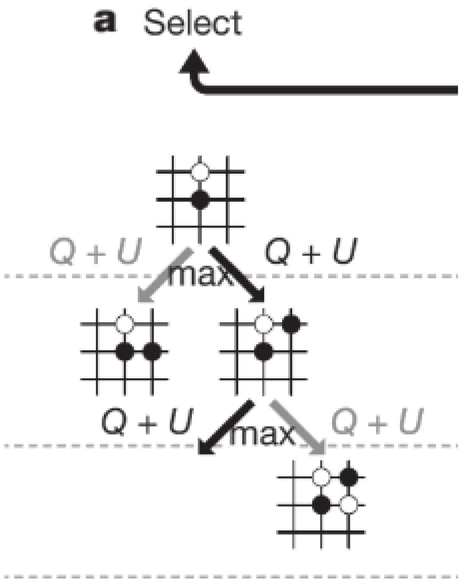
1. To generate best quality data, evaluate each nn check point f_{θ_i} against the current best network f_{θ_*}
1. Each evaluation consists of 400 games, using MCTS 1600 simulation to select each move
1. If new player wins by a margin of 55%, it becomes the best player α_{θ_*}
2. And used for self-play generation

Self-play 1. self-play by MCTS

1. α_{θ_*} (The best current player) is used to generate data
2. plays 25,000 games using 1,600 MCTS to select each move(0.4s)
3. first 30 move, temperature=1 after 30 temp ~> 0
4. Additional exploration : Dirichlet noise to prior probabilities in s0(root)

$$P(s, \mathrm{a}) = (1-\varepsilon)p_a + \varepsilon\eta_a, \text{ where } \eta \sim \mathrm{Dir}(0.03) \text{ and } \varepsilon = 0.25$$

Search algoritms



1. each node s in the search tree's edge (s,a) store

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

   N(s,a) : visit count
   W(s,a) : total action value
   Q(s,a) : mean action value
   P(s,a) : prior probability of selecting that edge

$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + U(s_t, a)) \qquad U(s,a) = c_{\text{puct}} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

처음에는 U(exploration)에 비중이 커서 prior prob가 높은 a를 선택하지만
점차 Q(less exploration)를 따라 action value가 높은 a를 선택

Search algoritms

2. leaf nose sL is added to queue for nn-work evalua $(d_i(\mathbf{p}), v) = f_\theta(d_i(s_L))$
   [di is dihedral reflection or rotation]
   i in [1...8]

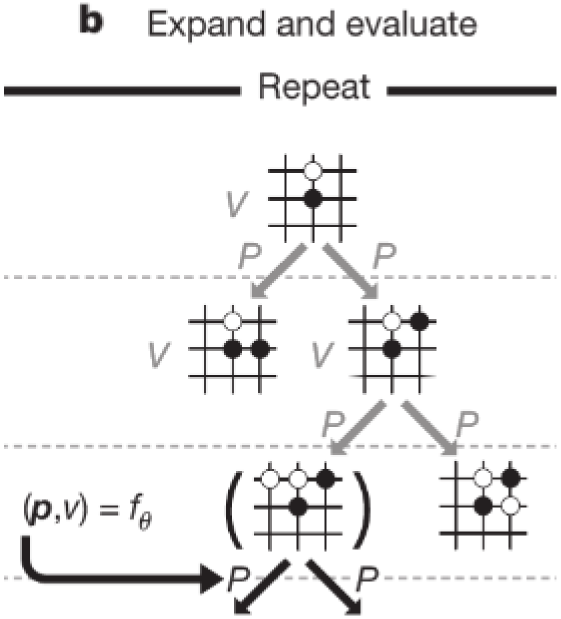
2-1. positions in the queue are evaluated by nn
   using mini-batch size of 8

2-2. leaf nodes edges (sL,a) is initialized

$$\{N(s_L,a)=0, W(s_L,a)=0, Q(s_L,a)=0, P(s_L,a)=p_a\}$$

2-3. value v is backed up

$$N \rightarrow N+1$$
$$W \rightarrow W+v$$
$$Q = W/N$$



Search algoritms

3. edge statistics(N,W,Q,P) are updated in backward pass through t<=L.

$$N(s_t, a_t) = N(s_t, a_t) + 1,$$

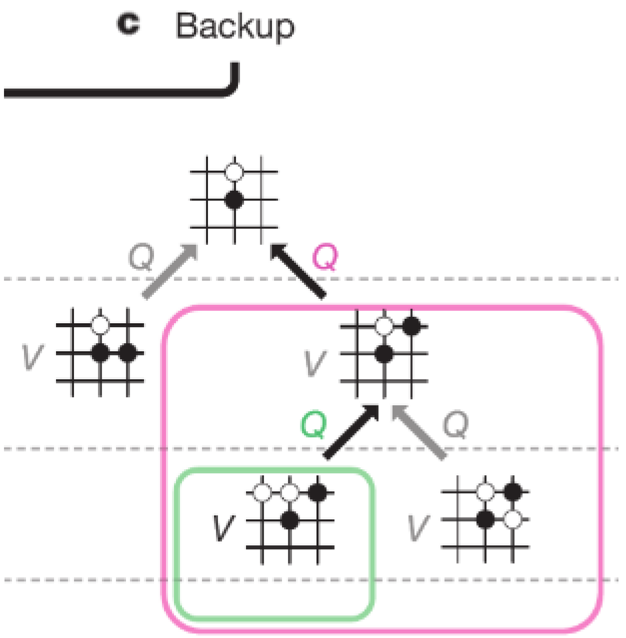
$$W(s_t, a_t) = W(s_t, a_t) + v$$

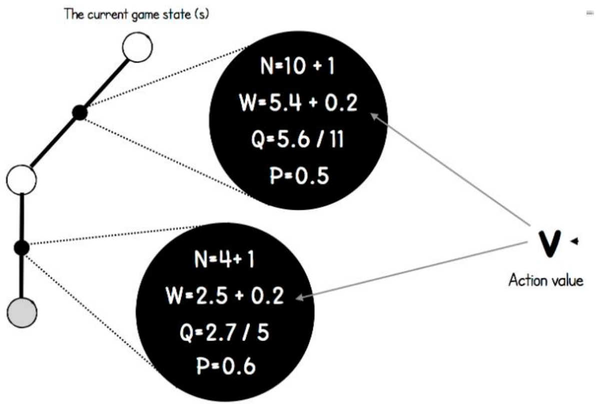
$$Q(s_t, a_t) = \frac{W(s_t, a_t)}{N(s_t, a_t)}$$

$$N \to N + 1$$

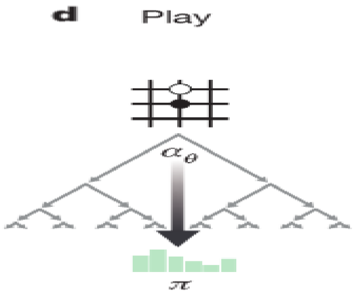
$$W \to W + v$$

$$Q = W / N$$





Search algoritms

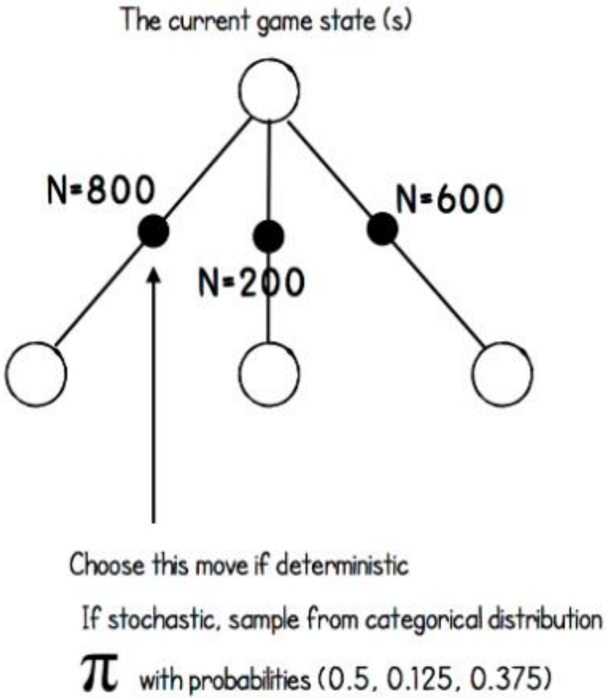


4. after 1600 MCTS sim

select move a to play in root position s0

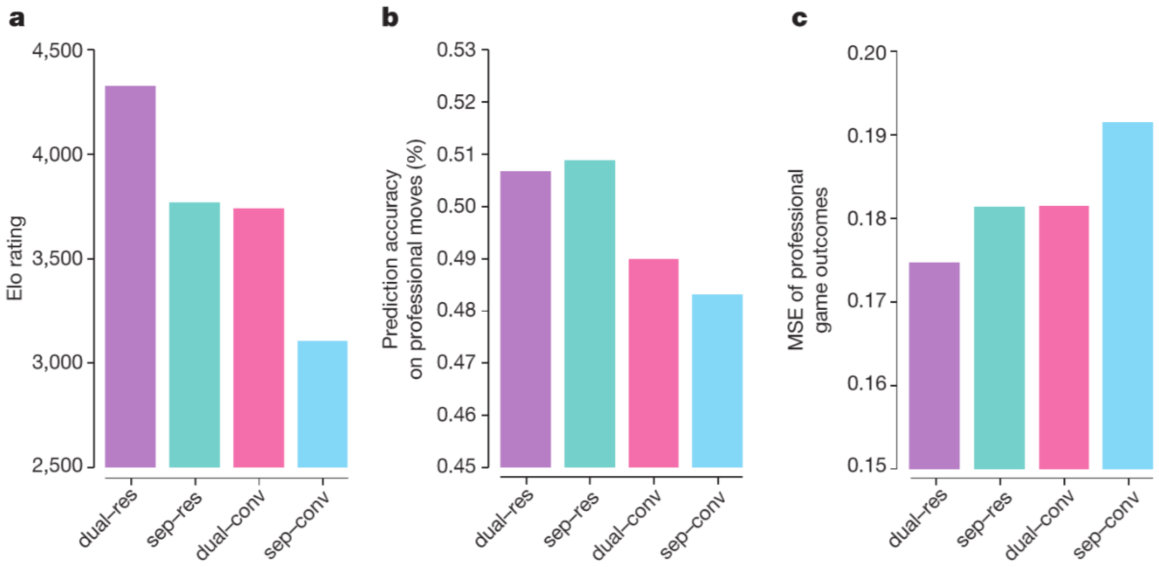
for(exploration play) $\pi(a|s_0) = N(s_0, a)^{1/\tau} / \sum_b N(s_0, b)^{1/\tau}$

for(competitive play) 결국 greatest N



temperature parameter 은 exploration의 수준 조절
(작을수록 exploration 안함)
선택된 Sub-tree는 재활용. 선택 안된 sub-tree는 버림
그러나 root value and best childe value < v_resign => resign

뭣이 중헌디



dual-res(알파고 제로)가 b에서 처럼 전문가를 따라할 확률은 쬐끔 낮아지지만 괜찮다.

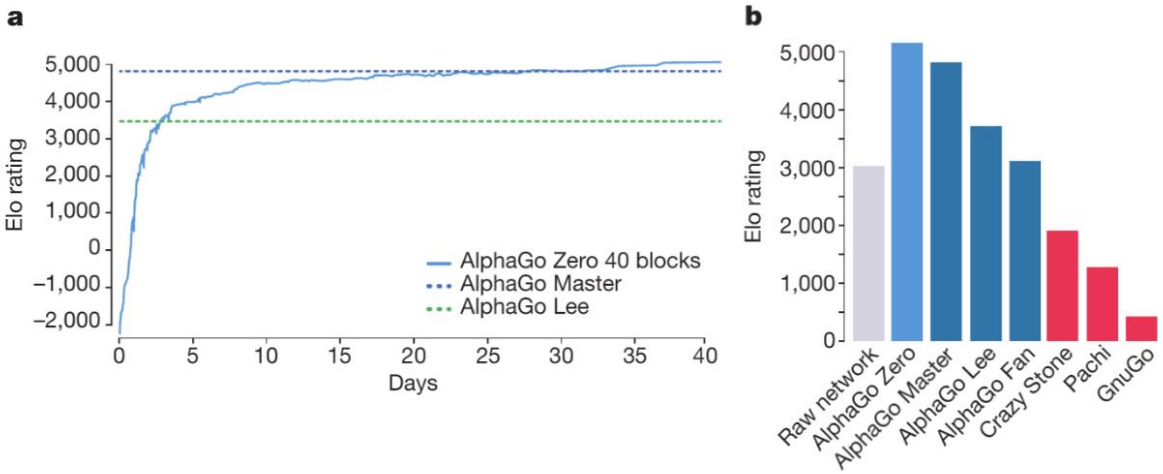
전문가보다 참신하게 두면서 전문가보다 잘하는

바둑을 art의 경지까지 끌어올림

Alphago Lee : sep + conv

Alhago Zero : dual + res

뭣이 중헌디



40-res block으로 40일동안 학습한 내용

알파고 zero와 알파고 master의

Elo rating 200점 차이는

알파고 제로의 승률이 75%를 말함

Raw network는 MCTS없이 신경망 f로만 두는 놈

0.4s의 생각할 시간도 필요없음.

Master는 Lee의 handcrafted feature와 Rollout사용 하고 human data로 SL해서 initialize함

결론(Conclusion)

현재 가장 도전적인 문제에서도 100% 강화학습 접근이 완전 가능하다는걸 보임

심지어 아무런 domain 지식(심지어 게임의 룰도)없이 super human의 경지로 끌어올림

심지어 하드웨어 소모도 크지않음

=> 인류가 몇 천년동안 쌓아왔던 지식이 무너졌다.

아아.. 배움이란 무엇이며, 지식이란 무엇이며, 인류란 무엇인가..

“ 교육의 목적은 비어있는 머리를 열려있는 머리로 바꾸는 것이다.”

- Malcolm Stevenson Forbes -

Reference

https://medium.com/oracledevs/lessons-from-alphazero-part-3-parameter-tweaking-4dceb78ed1e5
: 하이퍼파라미터 c_puct, dirichelt(α), temperature τ 에 대한 이해

https://adspassets.blob.core.windows.net/website/content/alpha_go_zero_cheat_sheet.png
: 알파고 zero cheat sheet. 이거 한장이면 끝.

https://blog.naver.com/ehdrndd
: 발표자 블로그.

https://github.com/utilForever/2020-OSS-Winter-AlphaZero
: 옥찬호님의 alphago zero알고리즘으로 오목풀기 자료