AlphaGo (2014-2017) DeepMind

Step 1: supervised learning for convolutional deep neural network

13 layers

input: $19 \times 19 \times 48$ (locations, features)

black
white
empty
opp captured
own captured

hand-coded features:
liberties
ladder capture
ladder escape

3 weeks

use database from games of expert players

– matched 55% of time

\+ smaller (faster) 25% of time

output: a move $(19 \times 19 + 1)$

Step 2: reinforcement learning for convolutional deep neural network

1 day

beat SL network 80% of time

Step 3: reinforcement learning for value network

+1 black win
⋮
−1 white win

using data from step 2 network
plays itself 30M times
samples 1 pos per game

Step 4: MCTS

default: use fast network from step 1

initialize new nodes' values using value net from step 3

tree policy: $\underbrace{q(s,a)}_{\text{exploit observed}} + c\,\underbrace{P(s,a)}_{\text{from larger step 1 network}}\sqrt{\dfrac{\#\text{ times parent visited}}{1+\#\text{ times child visited}}}$

Elo 3144 (2015, Fan Hui) → 3739 (2016, Lee Sedol) → 5185 (2017, retired)

Δ Elo 400 → higher rated player has 90+% chance of winning

AlphaGo Zero

input : 19 x 19 x 17 — current pos + last 7 positions + turn (all 1 = black, 0 = white)

output : move (19x19+1) and value [-1, +1]

play against self → (state, action probabilities, winner)

from MCTS

DB of last 500K games

batch of 2048

retrain

every 1000 iterations

play against best so far

win 55% to become new best, else reset

Deep Q Learning

$\pi^*(s) = \underset{a}{\operatorname{argmax}}\ q^*(s,a)$

two networks: learning, target

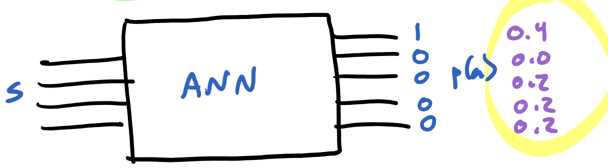
play using learning observe $(\underline{s}, \underline{a}, \underline{s'}, \underline{r})$ → add to replay database

compute error $\underbrace{\hat{Q}_t(s,a) - (r + \gamma \max_{a'} \hat{Q}_t(s',a))}$ ↳ (s,a,s',r) sampled from replay database

adjust learning accordingly

periodically copy learning to target

REINFORCE: learn policy directly



objective: maximize $E\left[\sum_{t=1}^{T} r_t\right]$

vector of weights in ANN

trajectory: $(s_1,a_1),(s_2,a_2),(s_3,a_3)\ldots$

objective to maximize ← $J(\theta) = \int \underbrace{\pi_\theta(\tau)}_{\text{prob ANN w/ weights } \theta \text{ yields } \tau} \cdot \underbrace{r(\tau)}_{\text{total reward over trajectory } \tau}\, d\tau$

gradient ascent on weights $\quad \nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) \cdot r(\tau)\, d\tau$

how to adjust weights θ to increase $J(\theta)$ most?

$$= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau)\, r(\tau)\, d\tau$$

$$\pi_\theta(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau) = \cancel{\pi_\theta(\tau)} \cdot \frac{\nabla_\theta \pi_\theta(\tau)}{\cancel{\pi_\theta(\tau)}}$$

$$= E\left[\nabla_\theta \log \pi_\theta(\tau) \cdot r(\tau)\right]$$

$$= E\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right) \cdot \left(\sum_{t=1}^{T} r(s_t,a_t)\right)\right]$$

$\cancel{\pi_\theta(\tau)}$

$$= E\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\cdot\left(\sum_{t=1}^{T} r(s_t,a_t)\right)\right]$$

$\pi_\theta(\tau)$ = prob of trajectory τ

$= \pi_\theta((s_1,a_1)(s_2,a_2)\cdots)$

$= p(s_1)\cdot\pi_\theta(a_1|s_1)\cdot p(s_2|s_1,a_1)\cdot\pi_\theta(a_2|s_2)\cdots$

$= \underbrace{p(s_1)\cdot\prod_{i=1}^{T}\pi_\theta(a_t|s_t)\cdot p(s_{t+1}|s_t,a_t)}$

$\log\pi_\theta(\tau) = \log$

$= \log p(s_1) + \sum_{i=1}^{T}(\log\pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t,a_t))$

$\nabla_\theta \log\pi_\theta(\tau) = \nabla_\theta \log p(s_1) + \sum_{i=1}^{T}(\log\pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t,a_t))$

$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta \log\pi_\theta(a_{i,t}|s_{i,t})\right)\cdot\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right)}$$

REINFORCE

get sample trajectory (or N of them) by running policy

compute estimate of $\nabla_\theta J(\theta)$

update θ : $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$