Rocky

AlphaZero at home

Gergely Halácsy & Gergely Papp

Grand picture

0. Let teacher be a random agent
1. Teacher plays some games
2. Student learns from teacher's plays
3. Student challenges their teacher
4. Winner takes teacher's place

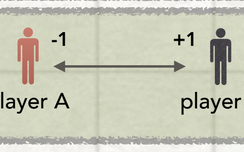
I. Self-play (data generation)

Play m self-plays that you believe are enough to have a good picture about the game in general. While playing, save the following information in each game:

Game state

π Policy — Probability distribution on the legal moves saying which move is likely to be the best

Value — The final winner (-1, 0, 1)



II. Learning (optimising neural network)

Make a copy of your best performing neural network, and learn from the games generated in the last k self-played games (k >= m).

$$\text{loss (state)} = \underset{\pi}{\text{cross\_entropy}} + \underset{\blacktriangledown}{\text{mean\_squared\_error}} + \text{regularisation}$$

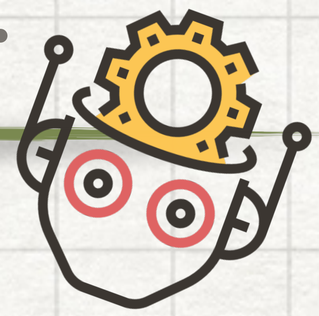
π Retrieved from MCTS by analysing how many times a move was played out of all

▼ A value based on which player won the game. It's either -1, 0 or 1.

III. Evolution

Make the fresh neural network play n games against the reigning one. Choose n to be a number which is enough to decide if a new prototype is better or not, and if it wins more than 55% of the games, replace the reigning network with the new one.

IV. Go back to step I.

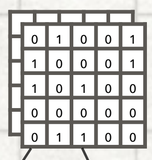


The neural network

Creating input image from state

1 layer for current player

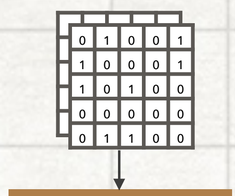
1 layer for opponent player



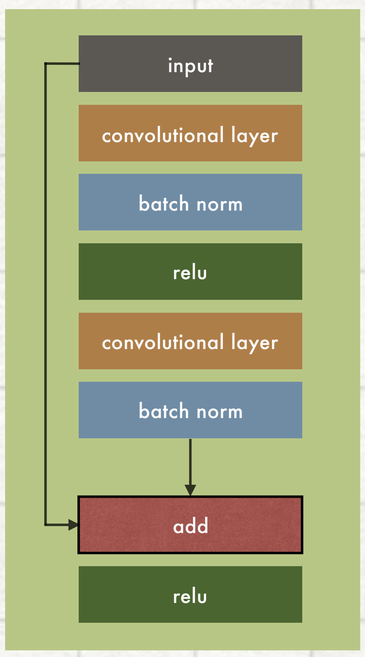
0: there is no piece here

1: there is a piece here

The neural network, everything together



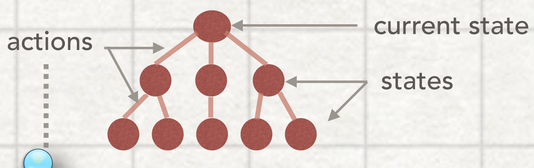
Residual layer



value head

policy head

Monte Carlo Tree Search (MCTS)



Action

- **N** The number of visits
- **W** Total value of next state
- **Q** Average value of next state
- **p** Prior probability

$$U = p + C * \sqrt{\frac{N}{\sum N_i}}$$

I. Selection

Choose action, that maximises **Q + U**

Q: The mean value of the state

U: Value controlling exploration

Repeat until a leaf node reached

II. Evaluation

Run the neural network on the leaf state and get predictions on **p, v**

p: Probabilities on next moves

v: The outcome of the game

III. Backpropagation

Update the traversed actions as:

N = N + 1 W = W + v Q = W / N

Run MCTS on current state for a couple of times (j). Then, choose your action as

$\frac{N^{1/\tau}}{\sum N_i^{1/\tau}}$ **stochastic way**	**deterministic way**
early game	late & competitive game
τ *temperature*	$argmax([N_1, N_2, ...])$

+ Some tiny dirichlet noise in order to achieve different plays

Achievements

Environment

Game: Connect4

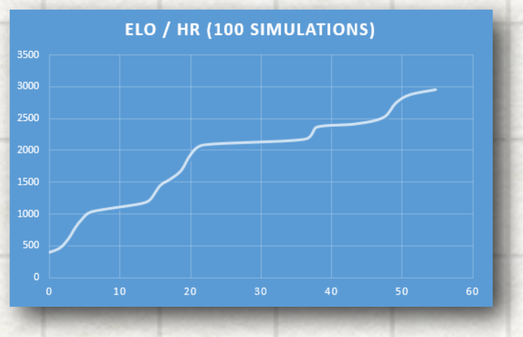
CPU: Intel Core™ 2 Quad, 2.40 GHz

GPU: -

Simulations: 100 (training), 1200 (testing)

Training time: 24 hours before testing, ~60 hours in total

Iterations: 16 before testing, ~35 in total



Results

The engine was tested (unofficially) against a small group of data scientists and physicists. Interestingly, after 24 hours of training Rocky managed to beat the human team 5 to 1.

Further work

We also tested what if we insert temporal difference methods into the algorithm, but it has not achieved anything outstanding. The idea is to reduce the difference between the predicted value and real outcome if a risky move was played in the next couple of states.