Monte Carlo Tree Search in AlphaZero

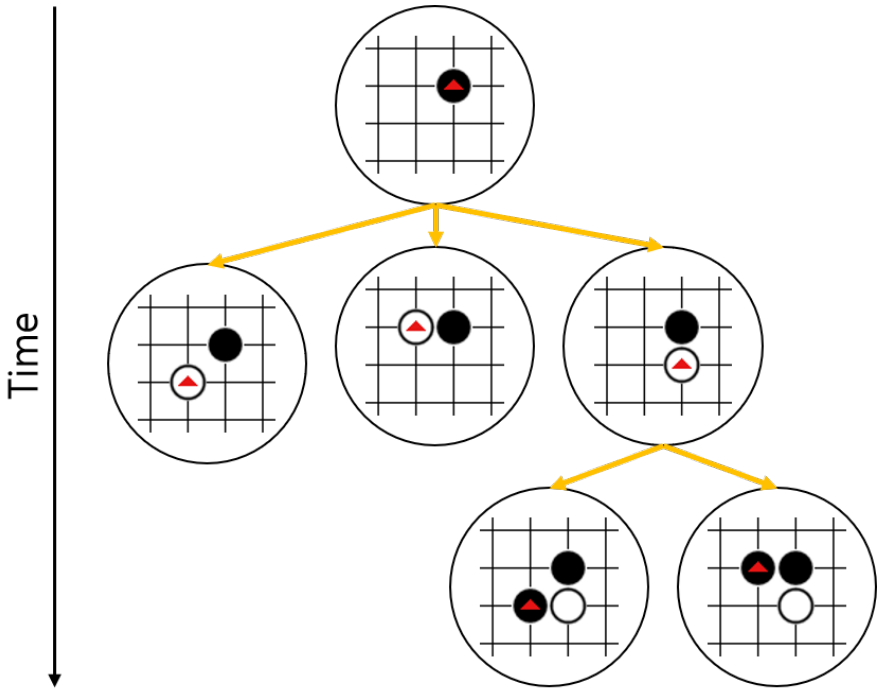
Junyeong Park

Hanyang University
Department of Computer Science

November 7, 2020

How to represent the game

- We can represent the game with **tree** structure.
- Root represents current state.
- Depth is chronological order.



How to represent the game

Each node stores a set of information and a set of statistics.

- s_t is state at timestep t.
- a_t is action at timestep t.
- $\mathrm{N}(s, a)$ denote that the total number of simulations.
- $\mathrm{W}(s, a)$ denote that the number of wins.
- and so on (ex. $\mathrm{Var}[V]$, $\sigma[V]$)

Tree Search

Our goal is to select the best move at state s_t.
$\longrightarrow$ We need to define the value of action.

How can represent the value of action?

- We have $\mathrm{W}(s, a)$ and $\mathrm{N}(s, a)$.
- So winning rate can be calculated by averaging the results of simulations.

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

- winning rate is also used for state-action value.

Tree Search Algorithm - Mini-max

Definition

Mini-max is an algorithm that find way to minmize maximum loss.

```
function MINIMAX(node, depth, maximizingPlayer)
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value ← −∞
        for all child of node do
            value ← max(value, Minimax(child, depth-1, false))
        return value
    else
        value ← +∞
        for all child of node do
            value ← min(value, Minimax(child, depth-1, true))
        return value
```

Tree Search Algorithm - Alpha-beta pruning

Definition

Alpha-beta pruning is a algorithm that seeks to decrease the number of nodes that are evaluated by the mini-max algorithm in its search tree.

- We don't need to search a whole tree.
- If one node is uninterested, do not search more.

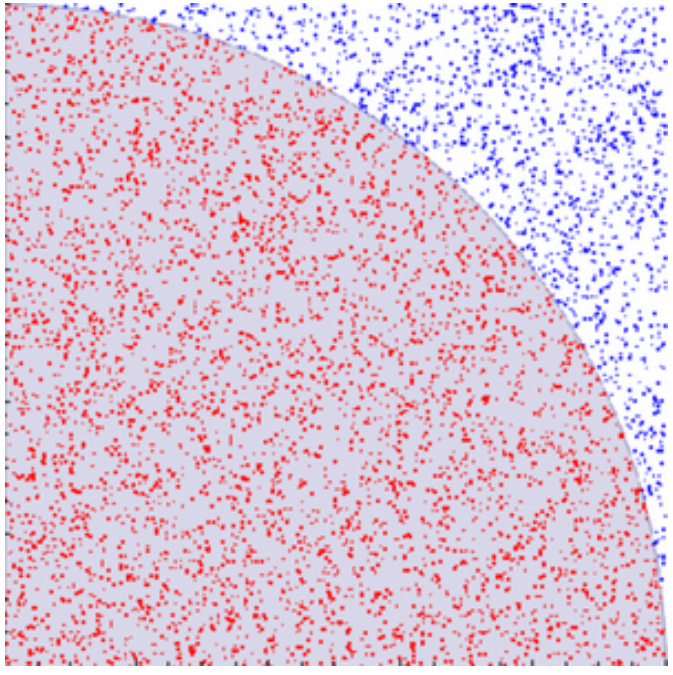
Limit of Mini-max and Alpha-beta pruning

- They need to investigate all nodes of the tree.
- However, a lots of environments' search space is huge.
  (ex) Go, Atari games, etc.
- Needs a lots of times and memories.

Monte Carlo Method

Definition

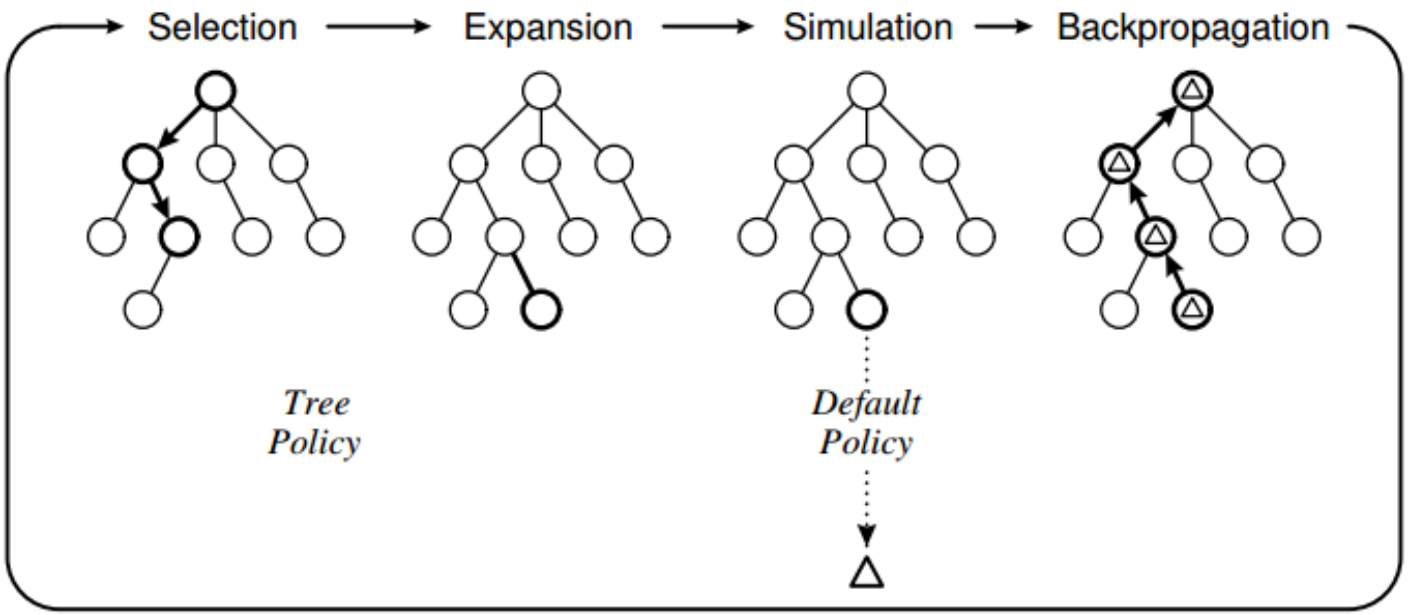
Monte carlo method is an algorithm that obtain numerical value with repeated random sampling.



Monte Carlo Tree Search

Definition

Monte carlo tree search is a tree search algorithm that adapts monte carlo method.



Monte Carlo Tree Search

- *Selection*
  Start from root R and select successive child nodes until a leaf node L is reached.
- *Expansion*
  In L, if the game has not ended yet, create one (or more) child nodes and choose node C from them.
- *Simulation*
  Complete one random playout from node C.
- *Backpropagation*
  Update statistics from C to R.

$$\mathrm{N}(s,a) \leftarrow \mathrm{N}(s,a) + 1$$
$$\mathrm{W}(s,a) \leftarrow \mathrm{W}(s,a) + z$$

AlphaZero

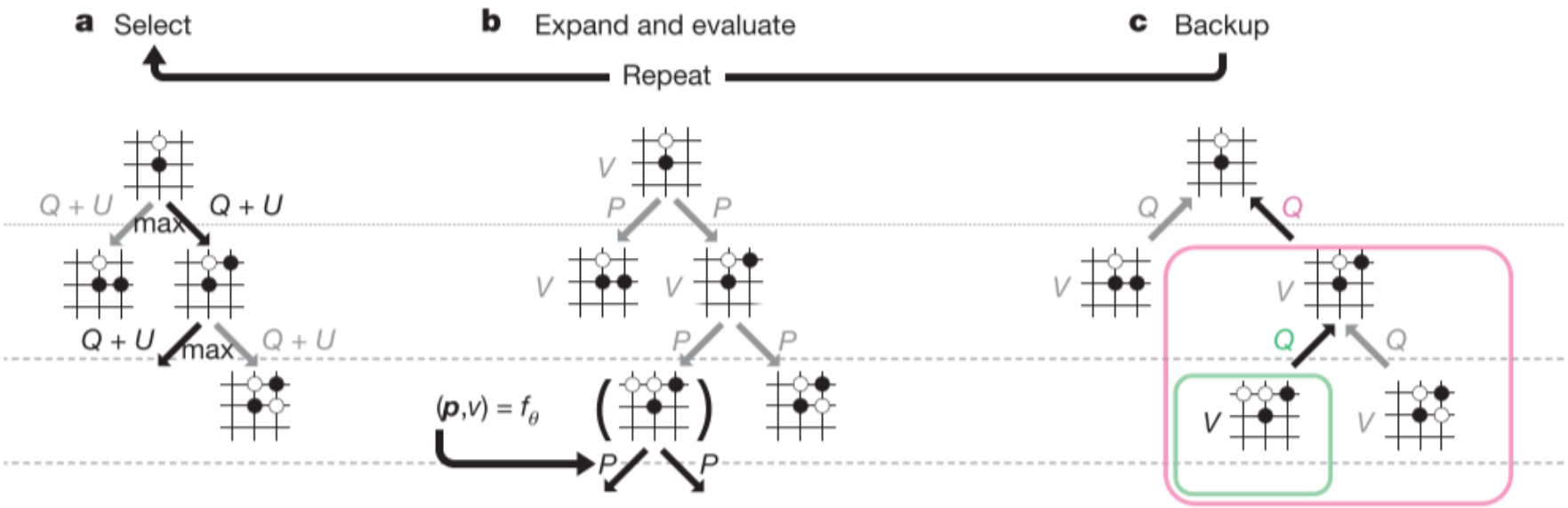
Definition

AlphaZero is reinforcement learning architecture to solve board games (Go, Chess, Shogi).

- It has neural network, $f_\theta(s) = (\mathbf{p}, v)$.
- Policy head produce move probabilities ($\mathbf{p}$).
  : reduce the breadth of tree.
- Value head produce winning probability (v).
  : reduce the depth of tree.

MCTS Simulation

AlphaZero use modified monte carlo tree search.



MCTS Simulation

Selection

- Use PUCT-algorithm.

- First, calculate values following

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

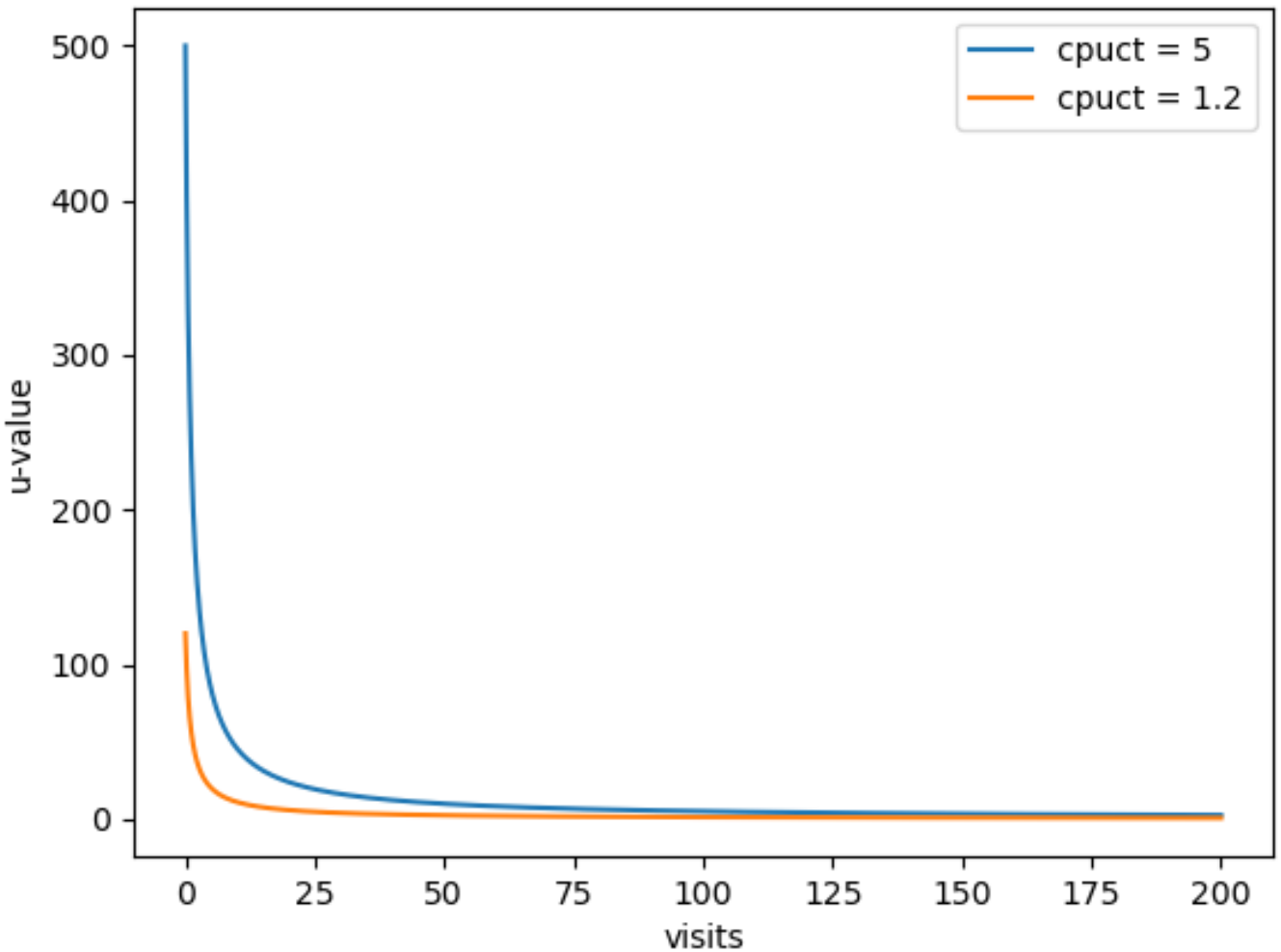
$$\mathrm{u}(s,a) = c_{\mathrm{puct}} \times \mathrm{P}(s,a) \times \frac{\sqrt{\sum_b \mathrm{N}(s,b)}}{\mathrm{N}(s,a)+1}$$

- Then, select action according to

$$a_t = \mathrm{argmax}_a(\mathrm{Q}(s,a) + \mathrm{u}(s,a))$$

until reach to leaf node s_L.

MCTS Simulation



MCTS Simulation

- The higher c_{puct}, the greater the breath of tree.
- The lower c_{puct}, the greater the depth of tree.

- AlphaZero apply heuristic for effective search.
  (except training)

$$\mathrm{PUCT}(s) = c_{\mathrm{init}} + \log\left(\frac{\sum_b \mathrm{N}(s,b) + c_{\mathrm{base}+1}}{c_{\mathrm{base}}}\right)$$

MCTS Simulation

Expand and Evaluate

- Add the leaf node s_L to neural network evaluation queue.
- Expand the leaf node and each edge (s_L, a) is initialized to

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

MCTS Simulation

Backup (backpropagation)

- Update edge statistics from edge to root.

- $\mathrm{N}(s,a) \leftarrow \mathrm{N}(s,a) + 1$
- $\mathrm{W}(s,a) \leftarrow \mathrm{W}(s,a) + v$

Play policy

After the simulation, we need to generate the best move.

- Select action that has highest value (winning rate).
- Select action that has highest visits.
- And so on.

Play policy

Select action that has highest value (winning rate)

- The value from neural network is often unreliable.
- So when low visit count, it's value is unreliable.

Play policy

Select action that has highest visits (winning rate)

- Visits increase only in the nodes that were best during the simulation.
- So it's generally reliable.

Play policy

Revisit: Select action that has highest visits (winning rate)

- We can calculate variance and deviation of the value from neural network.
- So we can compute confidence interval.

$$m - z \times \frac{\sigma}{\sqrt{\mathrm{N}}} \leq v \leq m + z \times \frac{\sigma}{\sqrt{\mathrm{N}}}$$

- We can select action that has highest lower confidence bound (LCB)