AlphaGo Zero

RL for Combinatorial Games

Aayush Borkar

Indian Institute of Technology Bombay

Summer of Science, 2024

Table of Contents

Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ)

Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ) where

- S is a finite set of states

Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ) where

- S is a finite set of states
- A is a finite set of actions

Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ) where

- S is a finite set of states
- A is a finite set of actions
- T is a transition function $T(s, a, s') = P_a(s'|s)$

Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ) where

- S is a finite set of states
- A is a finite set of actions
- T is a transition function $T(s, a, s') = P_a(s'|s)$
- R is a reward function $R(s, a, s')$

Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ) where

- S is a finite set of states
- A is a finite set of actions
- T is a transition function $T(s, a, s') = P_a(s'|s)$
- R is a reward function $R(s, a, s')$
- γ is a discount factor

ExpectiMax Tree

Any MDP can be represented as a tree where the nodes are states and edges are actions.

ExpectiMax Tree

Any MDP can be represented as a tree where the nodes are states and edges are actions. This tree is called the ExpectiMax Tree.

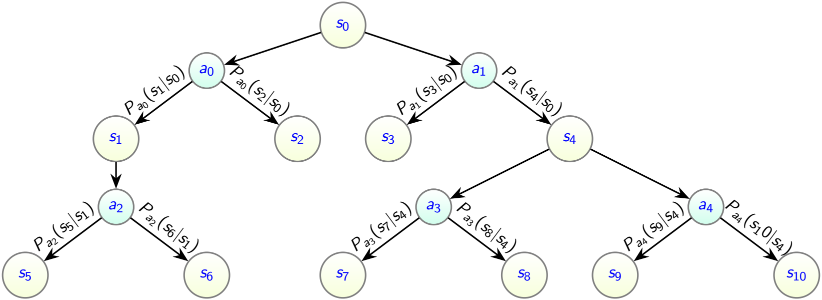


Table of Contents

Overview

- AlphaGo Zero uses a deep neural network f_θ (with parameters θ).

Overview

- AlphaGo Zero uses a deep neural network f_θ (with parameters θ).
- The neural network takes the state of the board s as input (along with history, depending on the nature of the game).

Overview

- AlphaGo Zero uses a deep neural network f_θ (with parameters θ).
- The neural network takes the state of the board s as input (along with history, depending on the nature of the game).
- It outputs two values, a vector $\boldsymbol{p}$ and v.

Overview

- AlphaGo Zero uses a deep neural network f_θ (with parameters θ).
- The neural network takes the state of the board s as input (along with history, depending on the nature of the game).
- It outputs two values, a vector $\boldsymbol{p}$ and v.
- $\boldsymbol{p}$ is a vector containing the probabilities of selecting every action.

Overview

- AlphaGo Zero uses a deep neural network f_θ (with parameters θ).
- The neural network takes the state of the board s as input (along with history, depending on the nature of the game).
- It outputs two values, a vector $\boldsymbol{p}$ and v.
- $\boldsymbol{p}$ is a vector containing the probabilities of selecting every action.
- v is a scalar. It is the probability of winning from the current state.

Self-play

The program plays against itself to generate training data. In each self-play episode, MCTS is initialized with parameters from the neural network. It then uses the MCTS to generate a sequence of states and actions.

Self-play

The program plays against itself to generate training data. In each self-play episode, MCTS is initialized with parameters from the neural network. It then uses the MCTS to generate a sequence of states and actions.

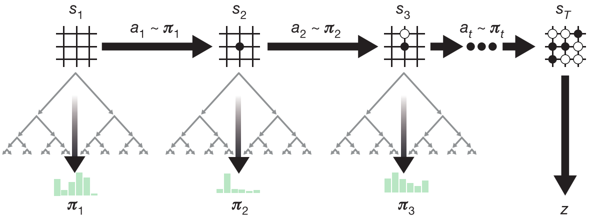


Figure: Self-play in AlphaGo Zero

Training

- MCTS outputs a policy vector, $\boldsymbol{\pi}_t$, and a game winner z.

Training

- MCTS outputs a policy vector, $\boldsymbol{\pi}_t$, and a game winner z.
- The neural network is given input the state of the board s_t, and outputs a policy vector p_t, and the predicted winner v_t.

Training

- MCTS outputs a policy vector, $\boldsymbol{\pi}_t$, and a game winner z.
- The neural network is given input the state of the board s_t, and outputs a policy vector p_t, and the predicted winner v_t.
- The neural network is trained to make $\boldsymbol{\pi}_t$ and p_t similar. And also to make v_t and z similar.

Training

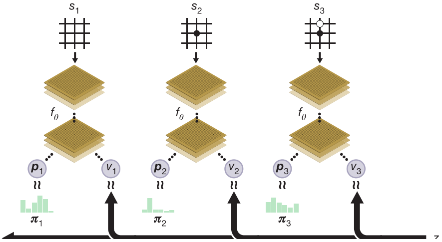
- MCTS outputs a policy vector, π_t, and a game winner z.
- The neural network is given input the state of the board s_t, and outputs a policy vector p_t, and the predicted winner v_t.
- The neural network is trained to make π_t and p_t similar. And also to make v_t and z similar.
- 

Figure: Neural Network Training in AlphaGo Zero

Table of Contents

MCTS

MCTS is an algorithm for making optimal decisions in RL for Combinatorial games. In AlphaGo Zero, MCTS is done in four steps.

MCTS

MCTS is an algorithm for making optimal decisions in RL for Combinatorial games. In AlphaGo Zero, MCTS is done in four steps.

- Selection

MCTS

MCTS is an algorithm for making optimal decisions in RL for Combinatorial games. In AlphaGo Zero, MCTS is done in four steps.

- Selection
- Expansion

MCTS

MCTS is an algorithm for making optimal decisions in RL for Combinatorial games. In AlphaGo Zero, MCTS is done in four steps.

- Selection
- Expansion
- Backpropagation

MCTS

MCTS is an algorithm for making optimal decisions in RL for Combinatorial games. In AlphaGo Zero, MCTS is done in four steps.

- Selection
- Expansion
- Backpropagation
- Play

MCTS

Each node in the search tree stores the following information:

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

MCTS

Each node in the search tree stores the following information:

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

- $N(s,a)$ is the number of times action a has been taken from state s

MCTS

Each node in the search tree stores the following information:

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

- $N(s,a)$ is the number of times action a has been taken from state s
- $W(s,a)$ is the total action value obtained by taking action a from state s

MCTS

Each node in the search tree stores the following information:

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

- $N(s,a)$ is the number of times action a has been taken from state s
- $W(s,a)$ is the total action value obtained by taking action a from state s
- $Q(s,a)$ is the average action value obtained by taking action a from state s

MCTS

Each node in the search tree stores the following information:

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

- $N(s,a)$ is the number of times action a has been taken from state s
- $W(s,a)$ is the total action value obtained by taking action a from state s
- $Q(s,a)$ is the average action value obtained by taking action a from state s
- $P(s,a)$ is the prior probability of selecting action a from state s

Selection

- Start at the root node s_0, and recursively select a child node until we reach a leaf node, say L.

Selection

- Start at the root node s_0, and recursively select a child node until we reach a leaf node, say L.
- At each time steps, an action is selected according to the statistics in the search tree, $a_t = \text{argmax}_a \left(Q(s_t, a) + U(s_t, a) \right)$, where $U(s, a)$ comes from the Upper Confidence Bounds algorithm for mutli-armed bandits.

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

Higher values of c_{puct} encourage exploration.

Expansion

- Once a leaf node L is reached, we add it to a queued for neural network evaluation. It is done in mini-batches of size 8.

Expansion

- Once a leaf node L is reached, we add it to a queued for neural network evaluation. It is done in mini-batches of size 8.
- The leaf node is then expanded by adding a new child node for each possible action. Each new edge from L is initialized with

$$N(L,a) = W(L,a) = Q(L,a) = 0, P(L,a) = p_a$$

Backup

The edge statistics are updated in the back-up phase. $N(s,a)$ is incremented by 1, $W(s,a)$ is incremented by v, and $Q(s,a) = \frac{W(s,a)}{N(s,a)}$.

Play

- After searching, the algorithm selects an action a to play in the root position s_0.

Play

- After searching, the algorithm selects an action a to play in the root position s_0.
- The probabilities are proportional to their exponentiated visit count.

$$\pi(a|s_0) = \frac{N(s_0, a)^{\frac{1}{\tau}}}{\sum_b N(s_0, b)^{\frac{1}{\tau}}}$$

Play

- After searching, the algorithm selects an action a to play in the root position s_0.
- The probabilities are proportional to their exponentiated visit count.

$$\pi(a|s_0) = \frac{N(s_0, a)^{\frac{1}{\tau}}}{\sum_b N(s_0, b)^{\frac{1}{\tau}}}$$

- τ is a temperature parameter that controls the level of exploration.

Play

- After searching, the algorithm selects an action a to play in the root position s_0.
- The probabilities are proportional to their exponentiated visit count.

$$\pi(a|s_0) = \frac{N(s_0, a)^{\frac{1}{\tau}}}{\sum_b N(s_0, b)^{\frac{1}{\tau}}}$$

- τ is a temperature parameter that controls the level of exploration.
- For the first 30 moves of each game, $\tau = 1$.

Play

- After searching, the algorithm selects an action a to play in the root position s_0.
- The probabilities are proportional to their exponentiated visit count.

$$\pi(a|s_0) = \frac{N(s_0, a)^{\frac{1}{\tau}}}{\sum_b N(s_0, b)^{\frac{1}{\tau}}}$$

- τ is a temperature parameter that controls the level of exploration.
- For the first 30 moves of each game, $\tau = 1$.
- For the rest of the game, $\tau \to 0$.

Putting it all together

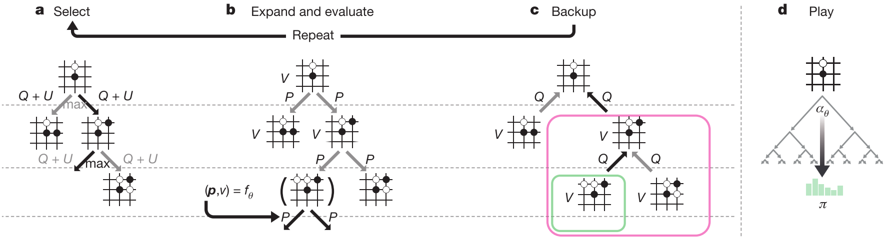


Figure: MCTS in AlphaGo Zero