Master's Thesis: MuZero

Deep Reinforcement Learning with MuZero: Theoretical Foundations, Variants, and Implementation for a Collaborative Game

Content

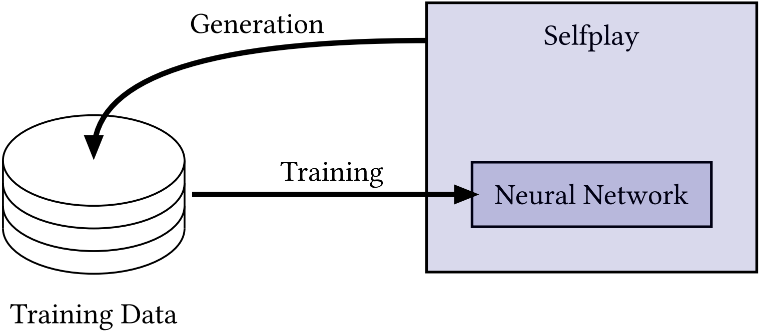
Overview

MuZero

- Model-based deep Reinforcement Learning algorithm
- Developed by Google DeepMind
- Games: Go, shogi, chess, Atari
- Evolution:
  - AlphaGo
  - AlphaGo Zero
  - AlphaZero
  - MuZero

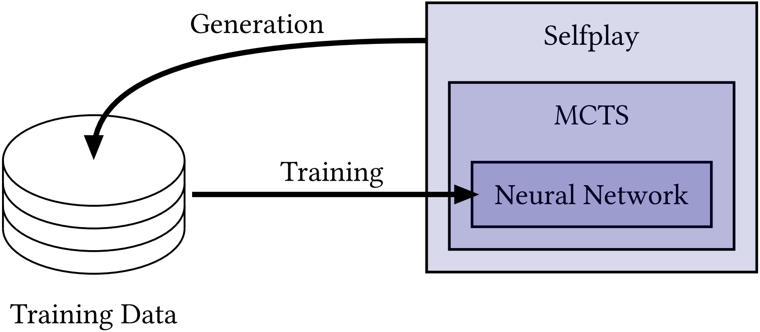
AlphaZero and MuZero

- Learn from scratch using selfplay
-



AlphaZero and MuZero

- Learn from scratch using selfplay
- Use Monte Carlo Tree Search (MCTS) to plan ahead



AlphaZero

Neural Network

- Input: 2D image of game board
- Residual CNN
- Two output heads:
  - **value**: Scalar
    How good is the current position?
  - **policy**: Distribution over actions
    What are promising moves to try in the search?

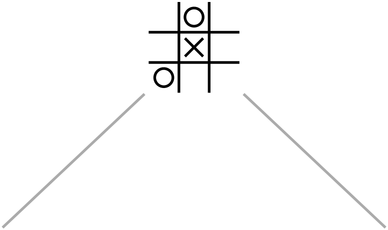
Monte Carlo Tree Search

- Builds (game) tree of possible future actions
- Stochastic algorithm: Random samples in the action space
- Tree grows iteratively:
  1. **Selection**: *Find the most urgent node*

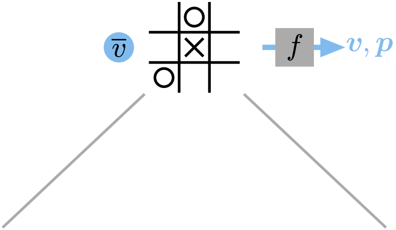
     Guided by:
     - network policy predictions
     - exploration
     - exploitation
  2. **Expansion**: *Add a new node, query network*
  3. **Backpropagation**: *Update statistics in the tree*
- New search tree at every game state to find a move
- Search results are used as policy training target

  → **Policy improvement**

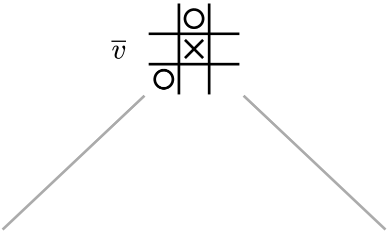
MCTS: Root Node



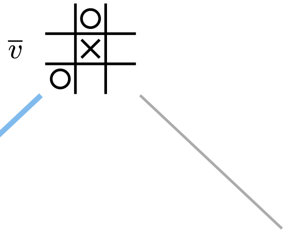
MCTS: Root Node: Network Inference



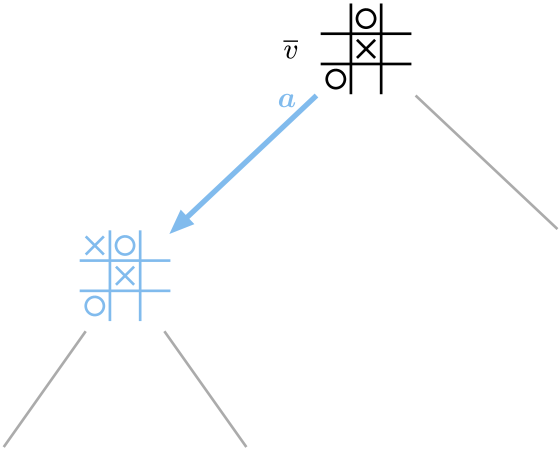
MCTS: Root Node



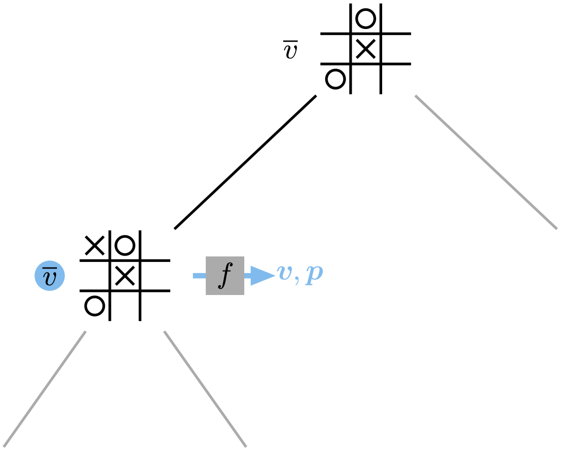
MCTS: Iteration 1: Selection



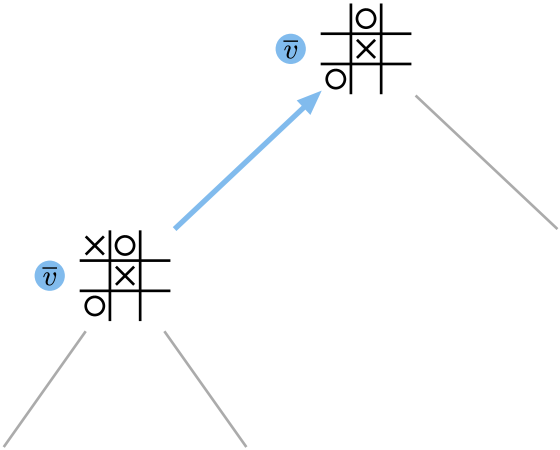
MCTS: Iteration 1: Expansion



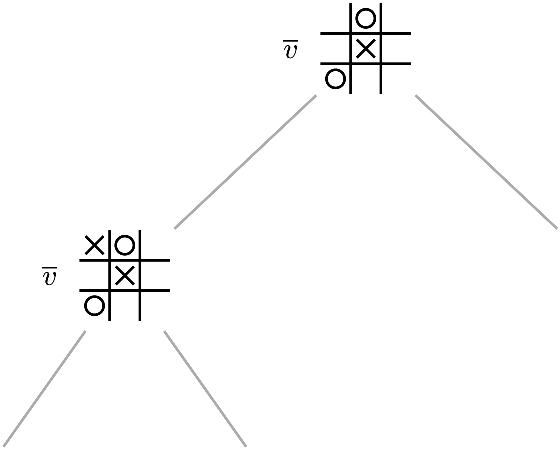
MCTS: Iteration 1: Expansion (Network inference)



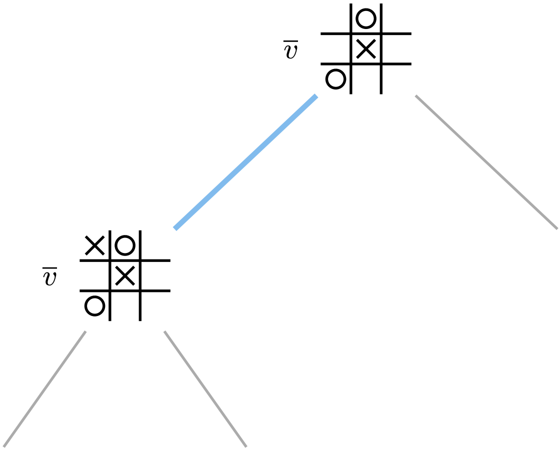
MCTS: Iteration 1: Backpropagation



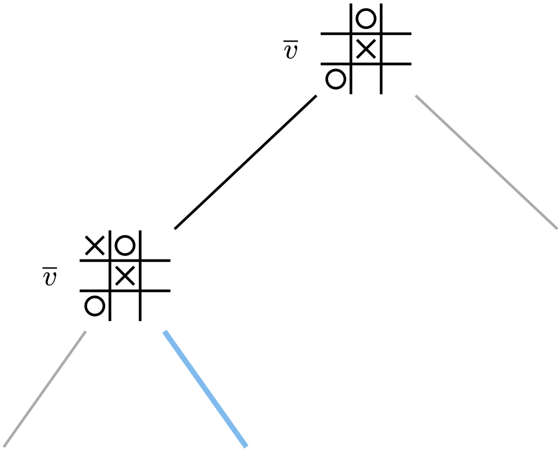
MCTS: Iteration 1



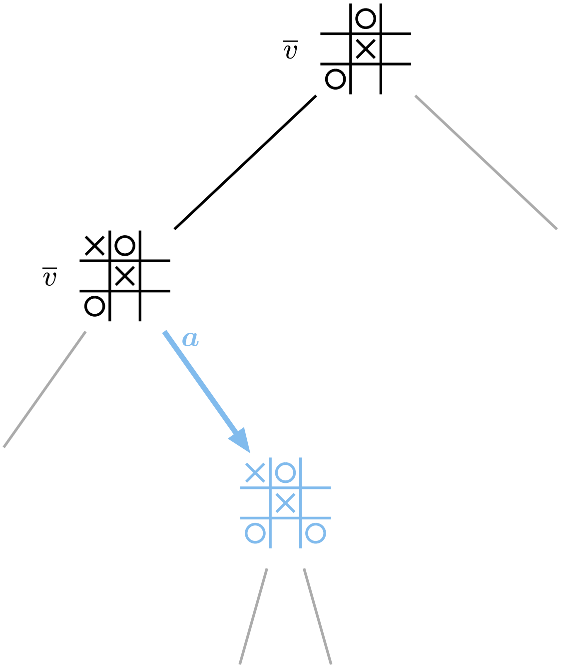
MCTS: Iteration 2: Selection



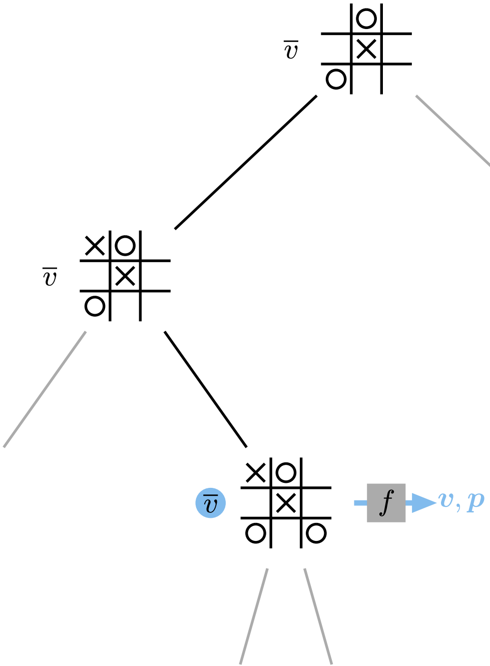
MCTS: Iteration 2: Selection



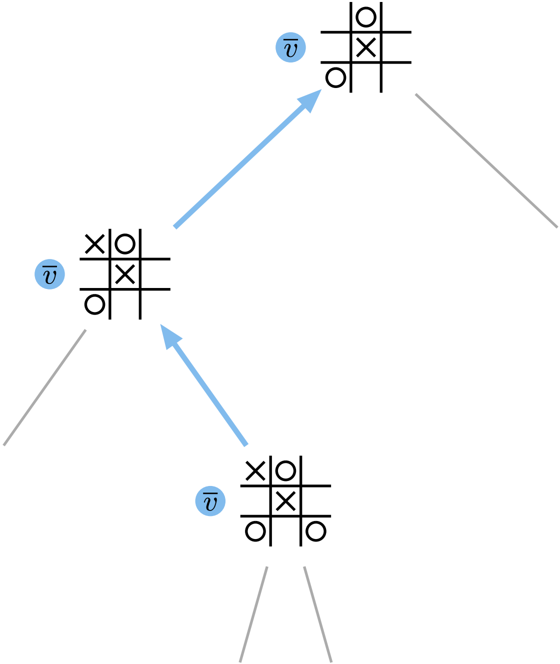
MCTS: Iteration 2: Expansion



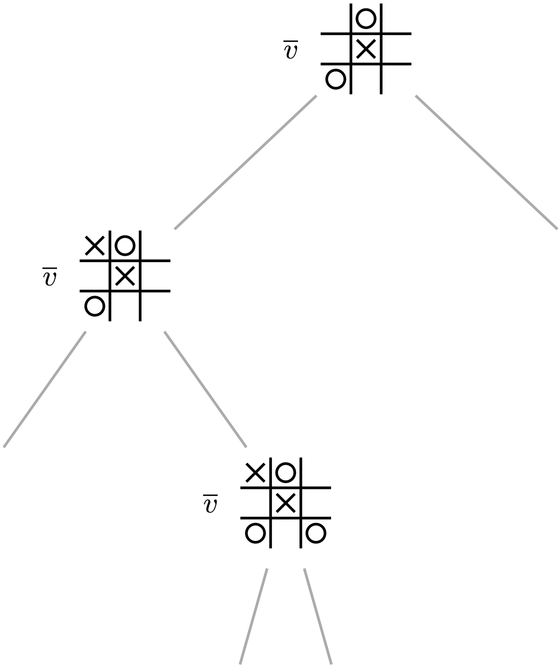
MCTS: Iteration 2: Expansion (Network inference)



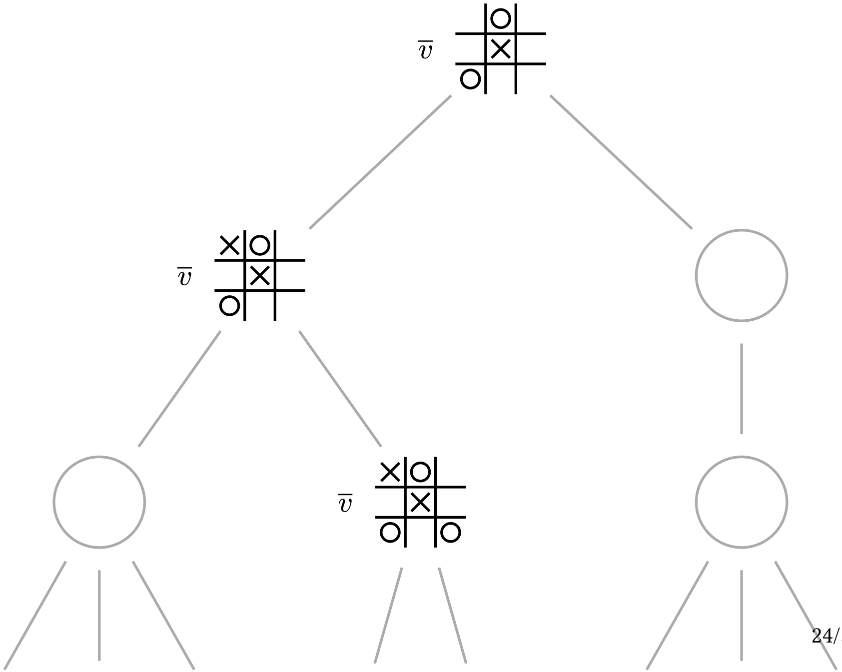
MCTS: Iteration 2: Backpropagation



MCTS: Iteration 2



MCTS: After many Iterations



Selfplay and Neural Network Training

- Play games using MCTS for both players
- Record training data (s, π, z) for each game state
  - $\boldsymbol{s}$: Game state Observation image
  - $\boldsymbol{\pi}$: MCTS policy Distribution over actions

    Which actions of the root node were visited by the search?
  - $\boldsymbol{z}$: Game Outcome Scalar

    Ended the game in a win, loss or draw?
- Supervised training on this data

MuZero

Summary

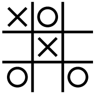
- Like AlphaZero, but no simulator in the tree search
- Instead: Learns a model of the environment dynamics
- Represents game states in a latent space

Neural Networks

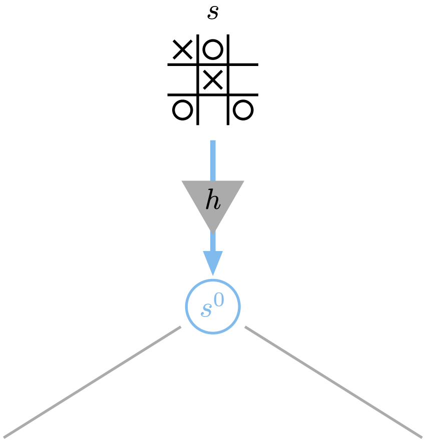
- 3 Networks:
  - Representation: Observation $\xrightarrow{h}$ Latent
  - Prediction: Latent $\xrightarrow{f}$ Policy, Value
  - Dynamics: Latent, Action $\xrightarrow{g}$ Latent, Reward

MCTS: Observation

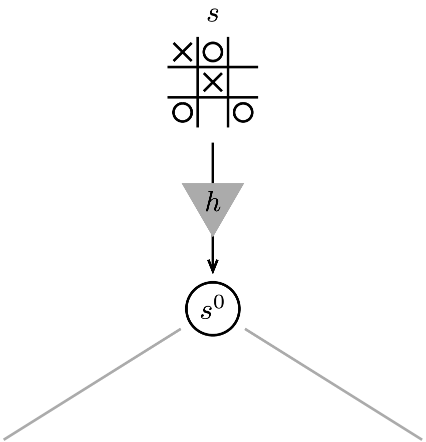
s



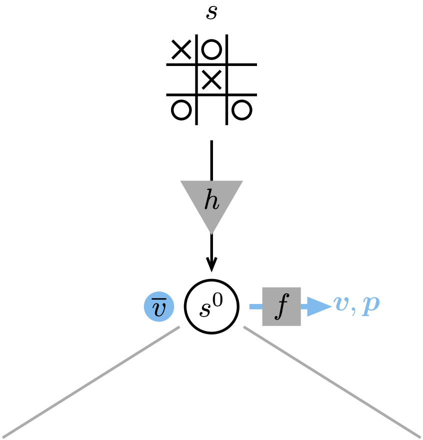
MCTS: Representation Network



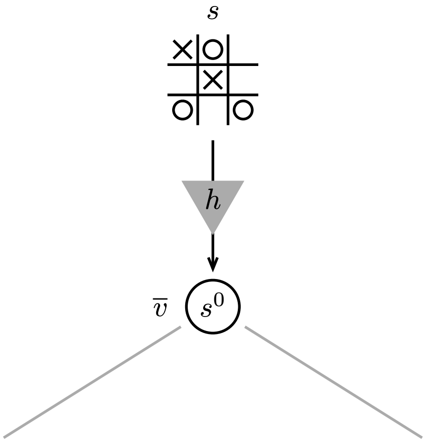
MCTS: Root Node



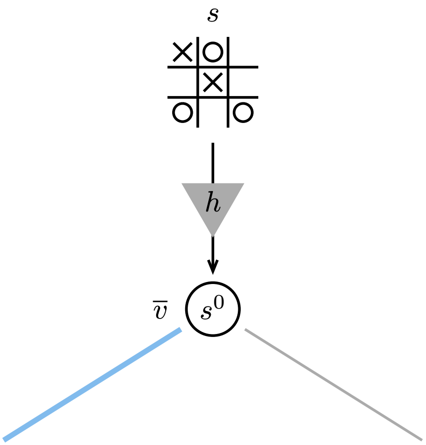
MCTS: Root Node: Network Inference



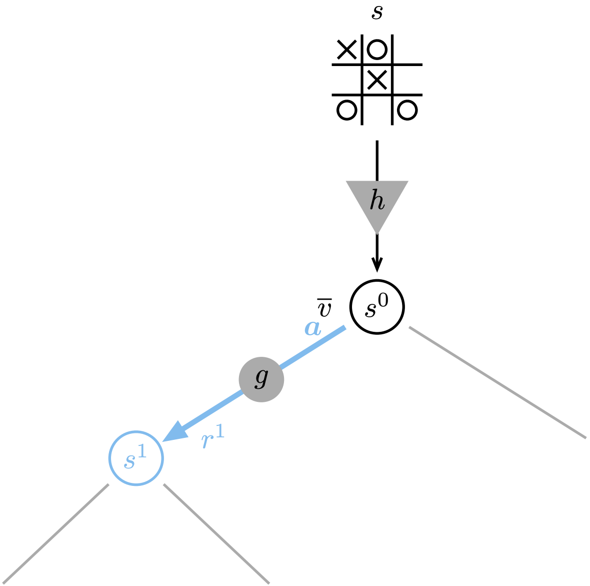
MCTS: Root Node



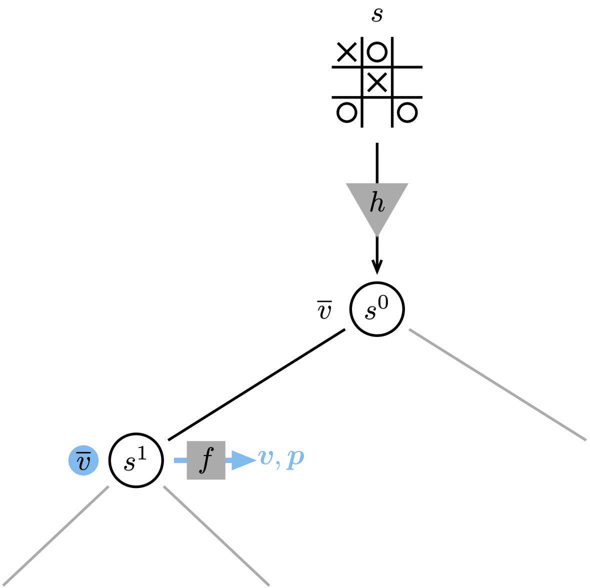
MCTS: Iteration 1: Selection



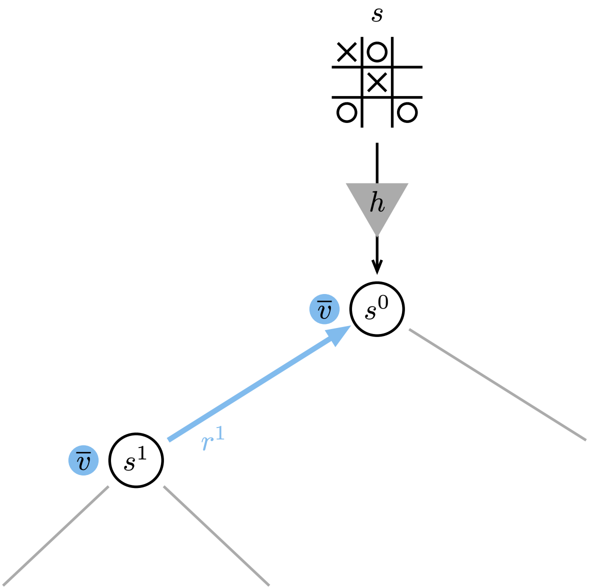
MCTS: Iteration 1: Expansion



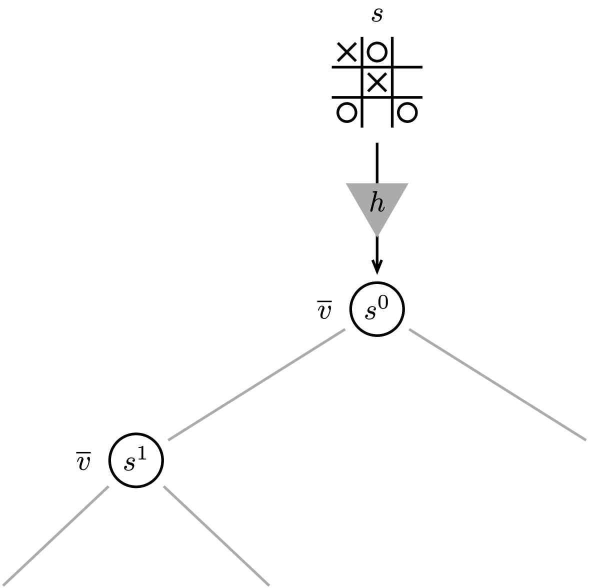
MCTS: Iteration 1: Expansion (Network inference)



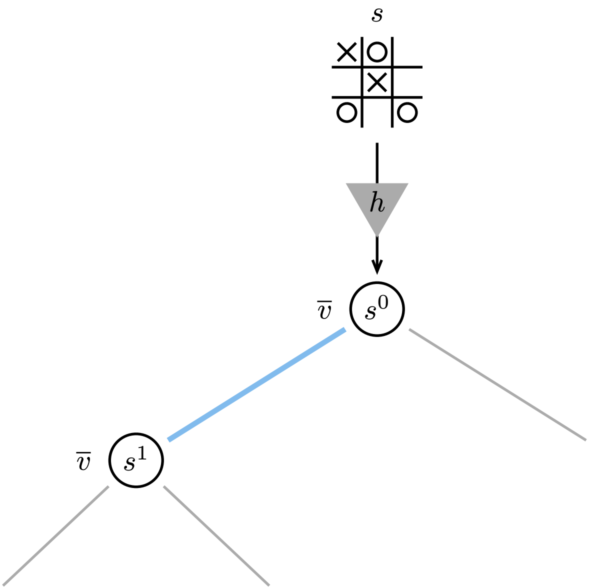
MCTS: Iteration 1: Backpropagation



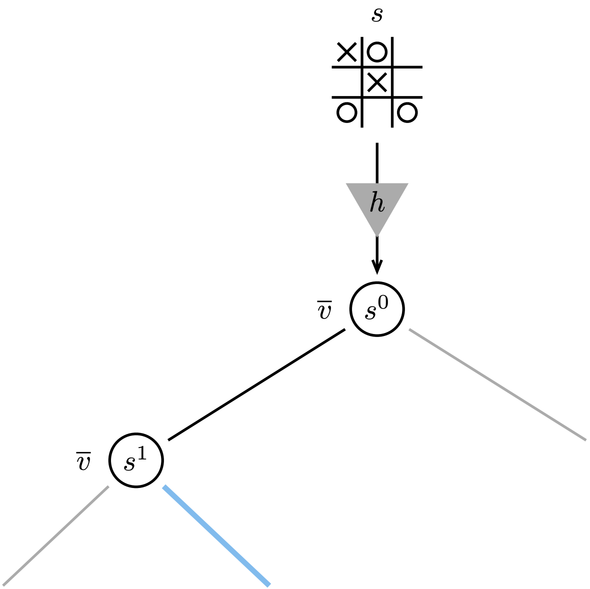
MCTS: Iteration 1



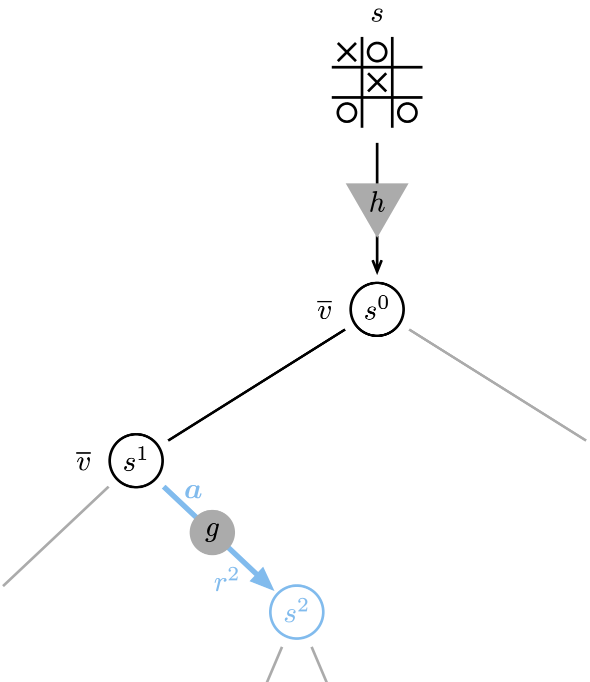
MCTS: Iteration 2: Selection



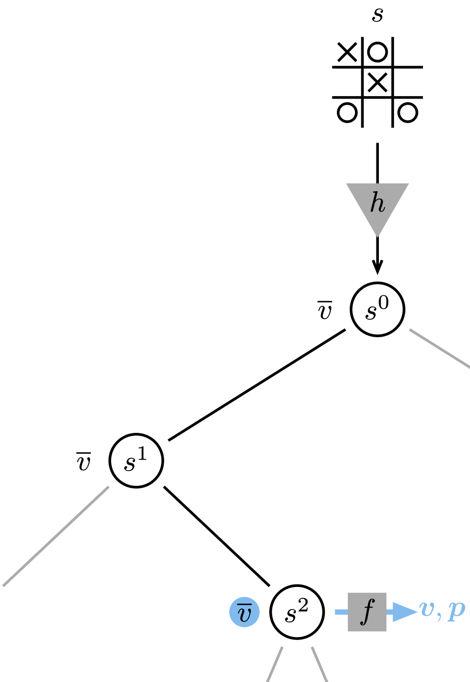
MCTS: Iteration 2: Selection



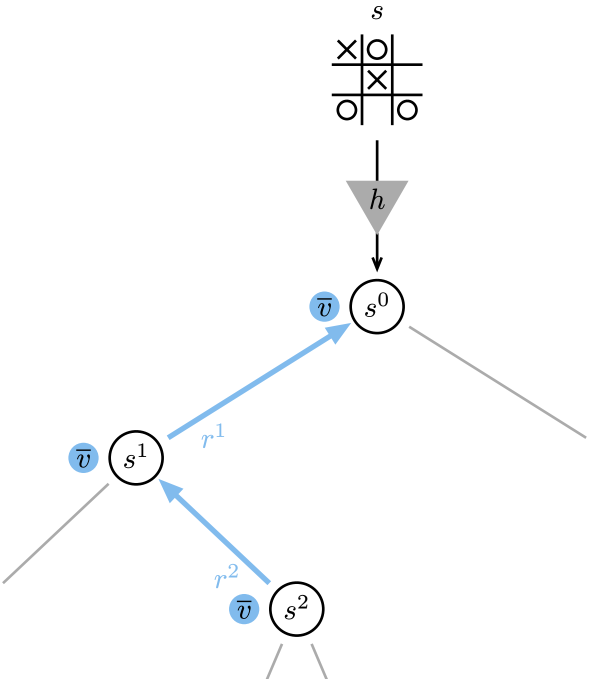
MCTS: Iteration 2: Expansion



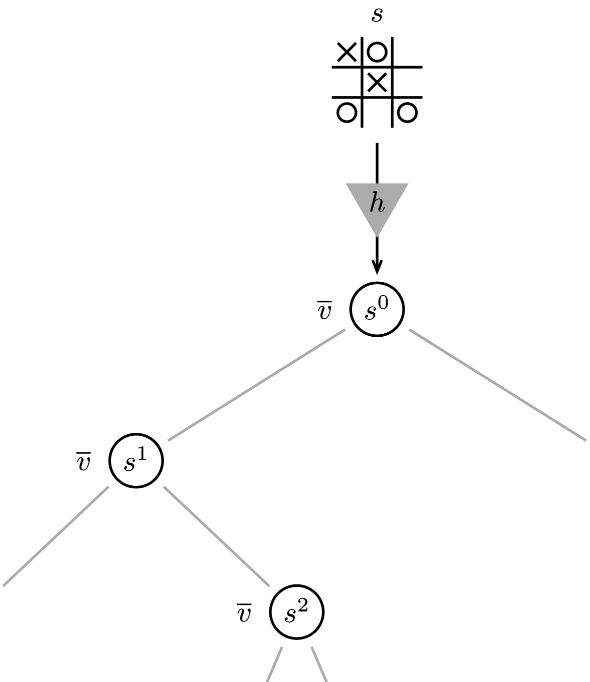
MCTS: Iteration 2: Expansion (Network inference)



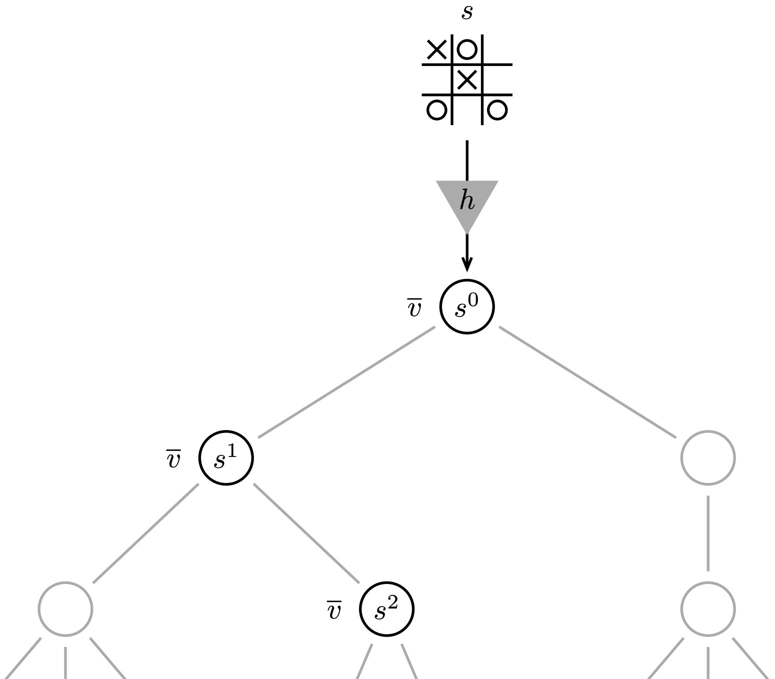
MCTS: Iteration 2: Backpropagation



MCTS: Iteration 2



MCTS: After many Iterations

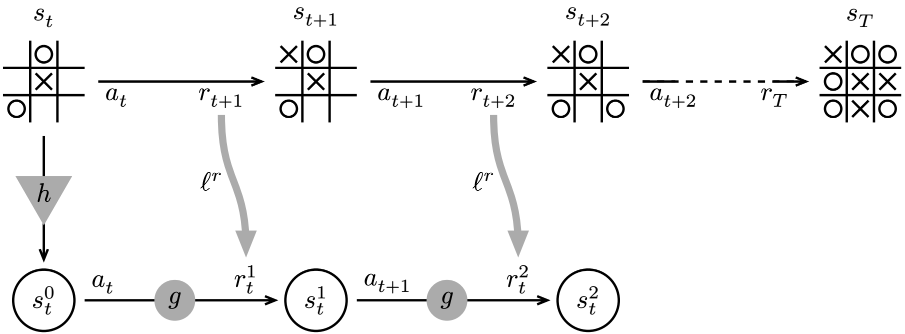


Selfplay

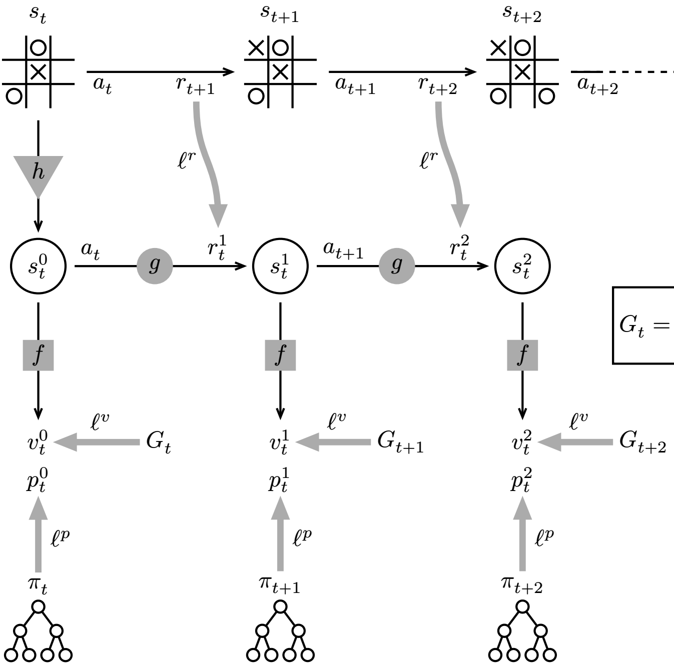
- Play games using MCTS for both players
- Record training data (s, a, r, π, G) for each game state
  - $\boldsymbol{s}$: Game state — Observation image
  - $\boldsymbol{a}$: Action taken — Onehot tensor
  - $\boldsymbol{r}$: Reward experienced — Scalar
  - $\boldsymbol{\pi}$: MCTS policy — Distribution over actions
  - $\boldsymbol{G}$: n-step return — Scalar

Neural Network Training

- Sample K consecutive training steps from buffer
- Start with observation
- Then unroll dynamics network for $K-1$ steps using actions
- Backpropagation-through-time
- End-to-end learning of policies, values and rewards K steps ahead



Neural Network Training (ii)



$$\boxed{G_t = \sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k}}$$

Multiplayer Modifications

Goals / Improvements over MuZero

- Multiplayer support
- Arbitrary turn order
- General-sum games
- Chance events / Stochasticity

Multiplayer MCTS

- At each node: Maximize current player's profit
- Requires:
  - Current player at turn
    - Predicted by the dynamics network
    - Trained on ground-truth labels from the game simulator
  - Per-player values and rewards
    - $v, r \in \mathbb{R}^n$ for n players
    - Trained on ground-truth labels from the game simulator

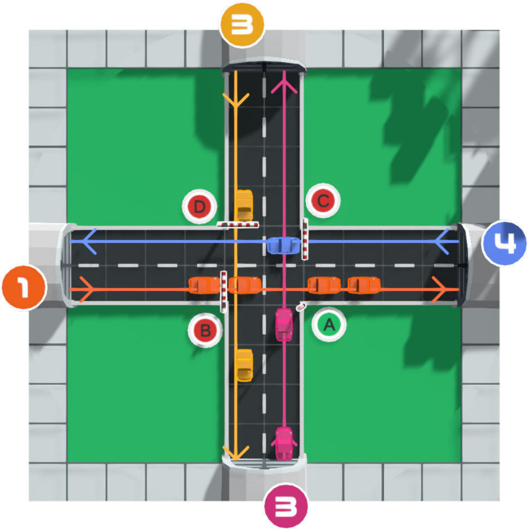
Stochasticity

- Add special chance player to the set of players
- Is at turn when chance events occur
- Policy targets are the chance outcome probabilities (ground-truth from simulator)
- MCTS selects actions according to predicted chance policy when the chance player is at turn

Evaluation

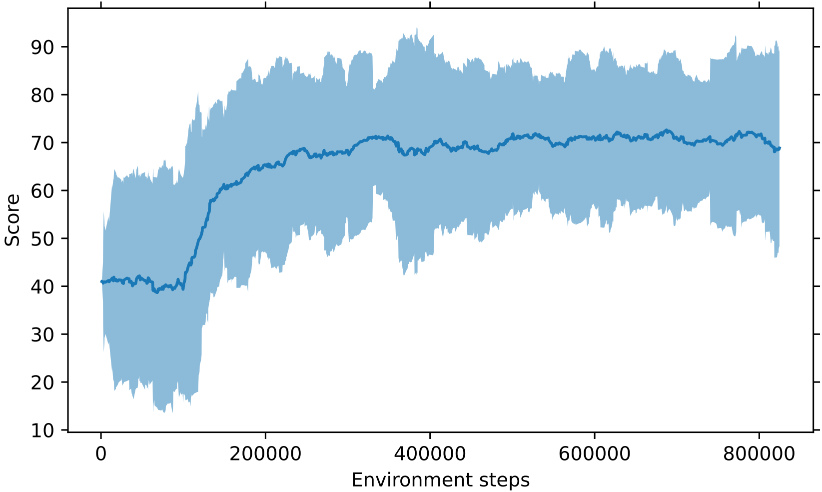
Carchess

- Cars spawn and drive along roads
- Cars may crash at intersections
- Players control traffic with barriers
- Goals:
  - Bring cars to their destination
  - Avoid crashes
- 10 Rounds of:
  - Each player toggles one barrier
  - Traffic advances for 5 steps
  - Spawn counts are updated randomly
- Collaboration: All players receive same reward



Carchess: Results

- Single training run
- 100.000 environment steps of random play at the beginning
- Mean score + 95% CI over last 30 plotted data points



Thanks

Questions