Model-Base Reinforcement Learning

-from AlphaGo to MuZero

Xuzhe Dang

Introduction

Section I - Backgrounds

An overview of Reinforcement Learning and MCTS

Section II - Algorithms

An Introduction of AlphaGo, AlphaGo Zero, Alpha Zero and MuZero

Section III - Summary

Summaries and QA

Backgrounds

Reinforcement Learning

- A machine learning algorithms to train model to make decisions
- Generate training data by interacting with environment (different from Supervised Learning)
- It’s goal-directed - maximize the rewards (different from Unsupervised Learning)
- It’s more like how animal and human learn

Backgrounds

Model-Free RL vs Model-Based RL

- Reinforcement Learning can be classified to various types according to different aspects !!!
- Whether a model of environment is used during the training or decision making
- The model of environment could be a real simulator or a learned model of simulator
- Model-Free RL
  - Train the policy from the real trajectories
  - Make decision directly from policy
- Model-Based RL
  - Use the virtual trajectories from interacting with a learned model
  - Make decision by planning on the model of environment

Backgrounds

MCTS

- It’s a heuristic search algorithms for decision making
- It’s sample based method
- There are a lot of successful applications of MCTS
  - Board Games (Go, Chess)
  - Poker
  - Strategy Video Games (Total War: Rome II)
  - Self-driving car (Tesla)

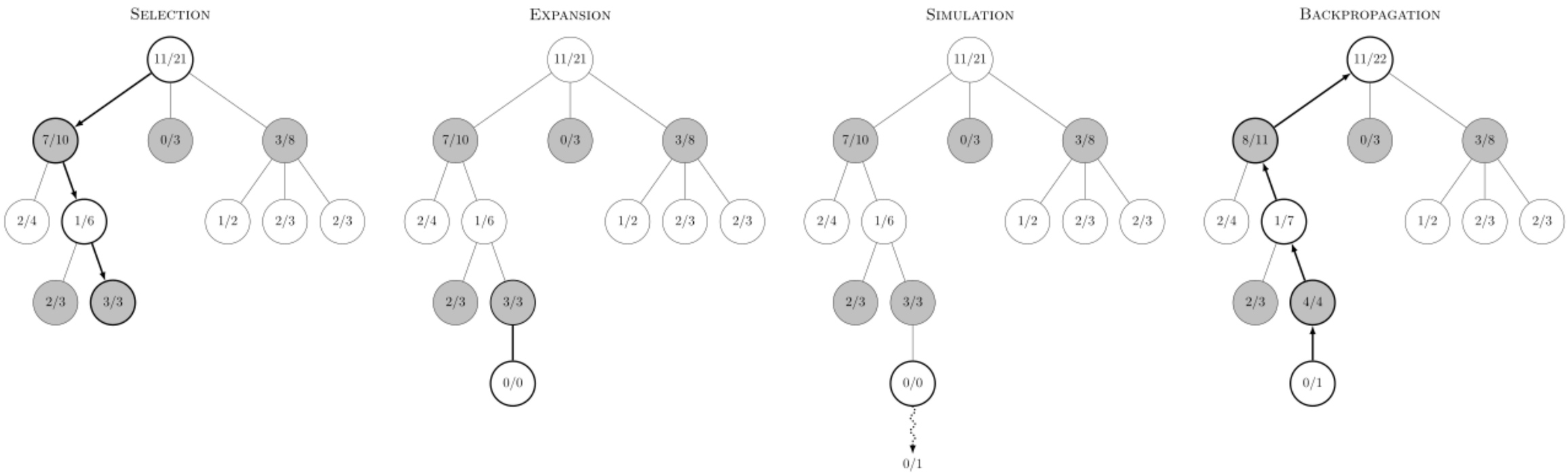
Backgrounds

MCTS

- Selection: select the node with max value recursively until reach the leaf node
- Expansion: randomly select a new node is all actions of leaf node are not sampled
- Simulation: randomly select actions to the end of game
- Back-propagation:update values and visited times of nodes recursively

Backgrounds

MCTS



Backgrounds

MCTS

- Exploitation and Exploration matters
  - Because it's sample-based, each node should be visited for several times to make estimation reliable.
- UCB1 + Minimax Tree Search = Upper Confidence Bounds for Trees
  - $UCB1 = \overline{v}_i + c\sqrt{\frac{lnN_i}{n_i}}$

Algorithms

AlphaGo

- One of the state of art Go AI developed by Deepmind
- UCT + Value Network + Policy Networks
- Supervised Learning + Reinforcement Learning

Algorithms

AlphaGo

- Why Go is so complicated
  - The rule is simple
  - The board is large (19 * 19, the board of Chess is 8 * 8)
- Huge Search Space
  - Breadth: legal actions per position ≈ 250
  - Depth: game length ≈ 150
  - Search Space = Breadth ** Depth

Algorithms

AlphaGo

- Use MCTS(UCT) as the main part of algorithm
- Use Policy Net to limit the breadth
- Use Value Net to limit the depth
- Use a Small Policy Net for simulation

Algorithms

AlphaGo

- Policy Net
  - Supervised Learning
  - 30 millions positions from Human players
  - Policy Net accuracy 57.0 %(All features) and 55.7%(Raw Board Position)
  - Rollout Policy Net 24.2 %
  - $loss_{policy} = logP(a|s)$

Algorithms

AlphaGo

- RL version Policy Net
  - Policy Gradient
  - Current Policy Net vs Randomly Selected One from Pool (stabilize training by preventing overfitting)
  - Weights are initialized to SL version Policy Net
  - Rewards
    - 0 for non-terminal steps
    - +1 at terminal step if win, else -1
  - $loss_{policy} = logP(a_t \mid s_t)z_t$

Algorithms

AlphaGo

- Value Net
  - Reinforcement Learning
  - An independent network
  - Self-play
  - $loss_{value} = MSE(v_\theta(s) - z)$

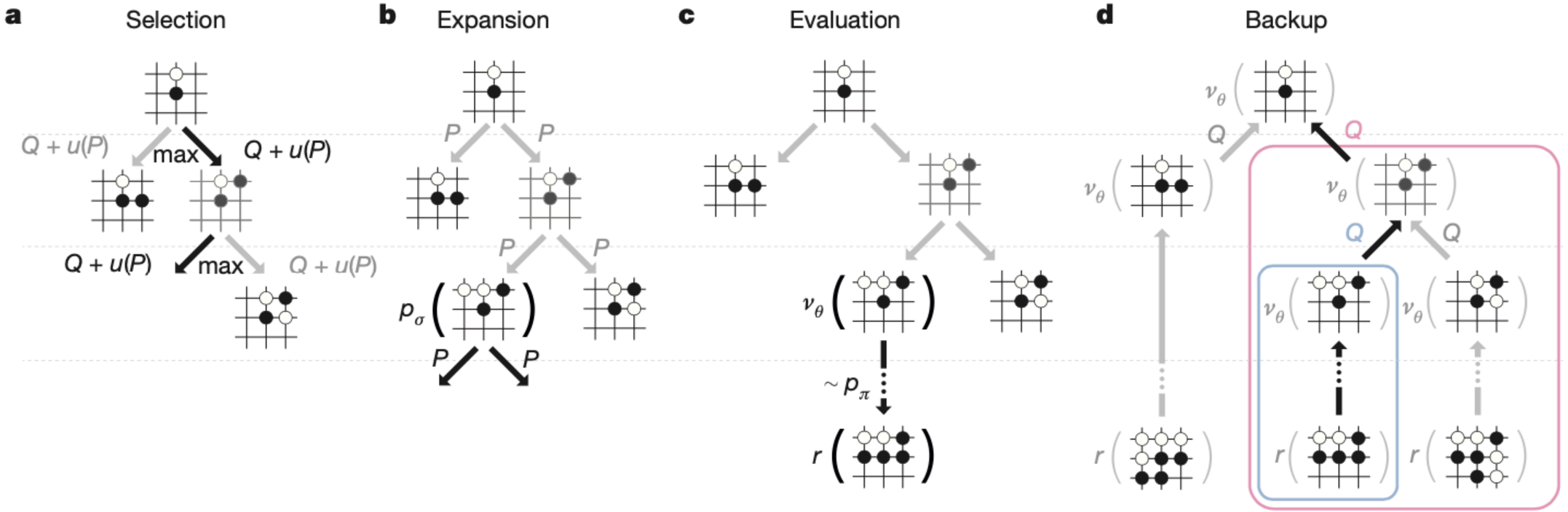
Algorithms

AlphaGo

- Which node to select
  - $a_t = argmax(Q(s_t, a) + u(s_t, a))$
  - $u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)}$
- What's the leaf value
  - $V_{(SL)} = (1 - \lambda)v_\theta(s_L) + \lambda z_l$
- $N(s, a) = \sum_{i=1}^{n} 1(s, a, i)$
- $Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{n} 1(s, a, i)V(s_L^i)$

Algorithms

AlphaGo



Algorithms

AlphaGo Zero

- A pure RL version of AlphaGo
- Improvements:
  - Policy Net and Value Net shares the same feature extractor
  - No more Supervised Learning
  - No more simulation

Algorithms

AlphaGo Zero

- Policy Net
  - Weights are initialized randomly
  - Cross-entropy loss
  - $loss_{policy} = \pi^T logp$

Algorithms

AlphaGo Zero

- Value Net
  - Weights are initialized randomly
  - MSE
  - $loss_{value} = (z - v)^2$

Algorithms

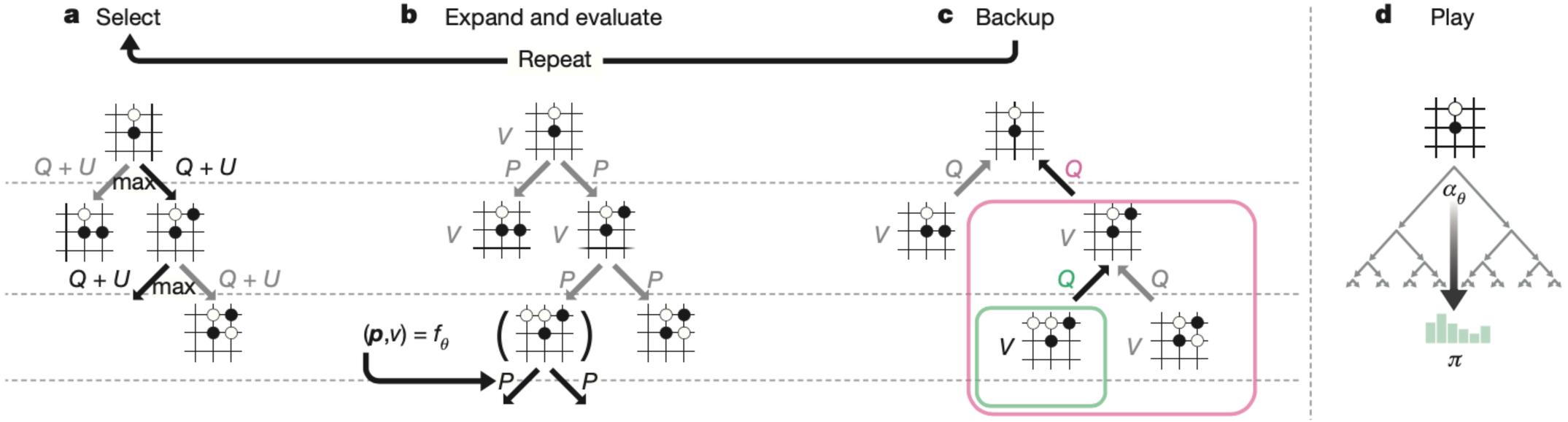
AlphaGo Zero

- Which node to select - PUCT algorithm
  - $a_t = argmax(Q(s_t, a) + u(s_t, a))$
  - $u(s, a) = c_{puct} P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}$

- $N(s, a) = \sum_{i=1}^{n} 1(s, a, i)$
- $Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{n} 1(s, a, i) V(s^i)$
- Which action to move:
  - $\pi(a \mid s_o) = N(s_0, a)^{1/\tau} / \sum_b N(s_0, b)^{1/\tau}$
    - τ is temperature parameter which controls the level of exploration

Algorithms

AlphaGo Zero



Algorithms

Alpha Zero

- A generalized AlphaGo Zero
- Improvements:
  - Play other board games
  - Remove rules for Go during the training

Algorithms

MuZero

- A Tree search algorithms combine with a learned model
- Matched the super human performance of Alpha Zero on board game
- Matched the state of art performance of reinforcement learning on Atari games
- A variant of MuZero improves the sample efficiency

Algorithms

MuZero

- Representation Function
- Prediction Function
- Dynamics Function

Algorithms

MuZero

- Representation Function
  - A neural network output a feature
  - Encode observation into a latent state
  - $s^0 = h_\theta(o_1, o_2, \ldots, o_n)$

Algorithms

MuZero

- Prediction Function
  - A neural network output probabilities of actions and value
  - Predict the probabilities of actions and value
  - $p^k, v^k = f_\theta(s^k)$

.

Algorithms

MuZero

- Dynamics Function
  - A neural network predict the latent state and reward if an action is taken
  - Simulate the dynamic of environment in latent state
  - $s^k, r^k = g_\theta(s^{k-1}, a^k)$

.

Algorithms

MuZero

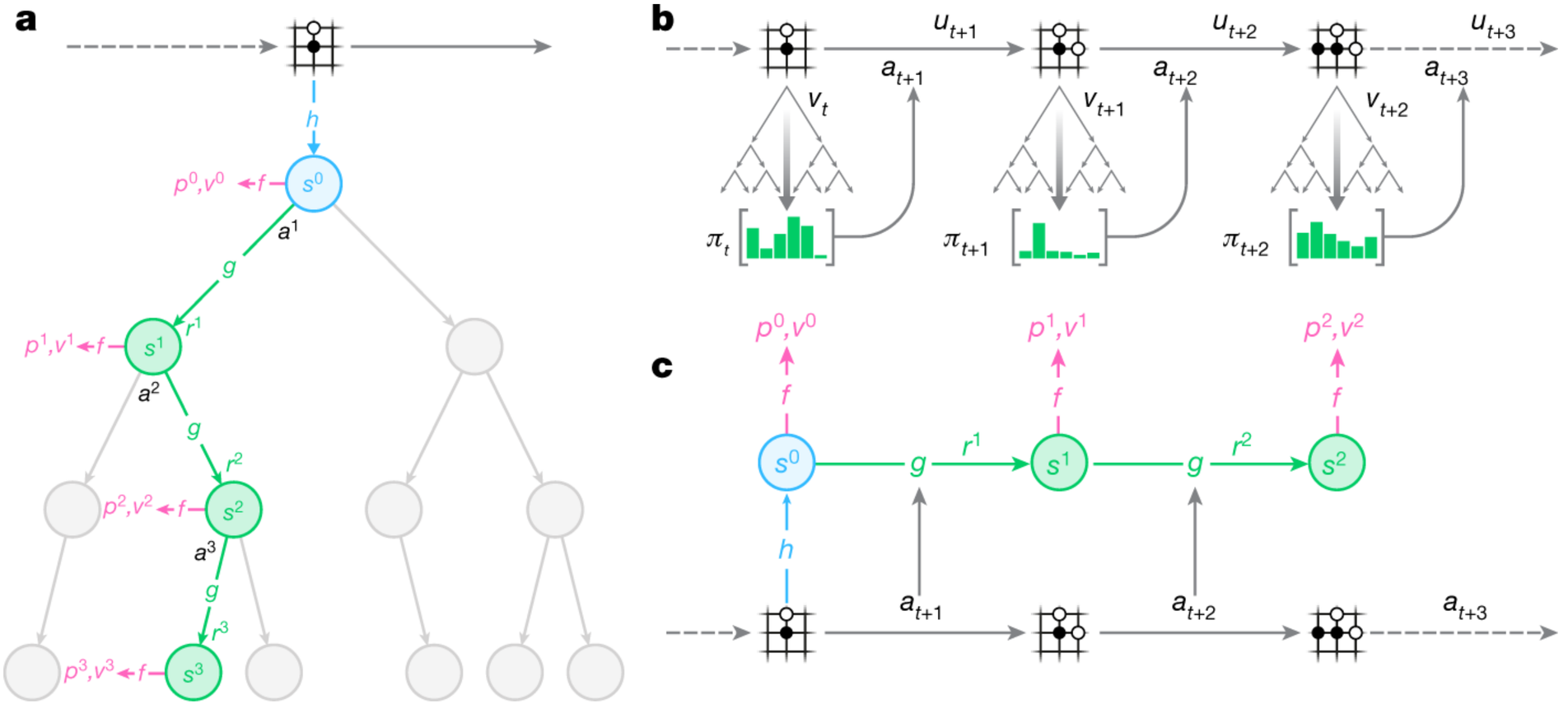
- Instead of roll out to the end of game, MuZero roll out for a certain steps
- Representation Function, Prediction Function and Dynamics Function are trained jointly
- $l_t(\theta) = \sum_{k=0}^{K} l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, p_t^k) + c\|\theta\|^2$

Algorithms

MuZero

Algorithms

MuZero



Algorithms

MuZero

- PUCT algorithm
  - $a_t = argmax(Q(s_t, a) + u(s_t, a))$
  - $u(s, a) = P(s, a)\frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}[c_1 + log(\frac{\sum_b N(s, b) + c_2 + 1}{c_2})]$
- $G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l$
- $Q(s^{k-1}, a^k) := \frac{N(s^{k-1}, a^k) \times Q(s^{k-1}, a^k) + G^k}{N(s^{k-1}, a^k) + 1}, N(s^{k-1}, a^k) := N(s^{k-1}, a^k) + 1$

Summary

Advantages

- It provide a general search algorithms, which works on many different environments
- It allows MCTS to get rid of simulator
- It's more rational than pure RL
  - Pure RL is strongly related to the rewards, but most time, there are no dense rewards

Summary

Future Works

- From perfect information environment to imperfect information environment
- Simultaneously Multi Agent Environment
- Introduce stochastic model

QA