ALPHAGO (NOTES)

March 14, 2016

Bunch of people from DeepMind

Introduction

AI system for playing Go

- Combines Deep Reinforcement Learning and Monte Carlo Tree Search

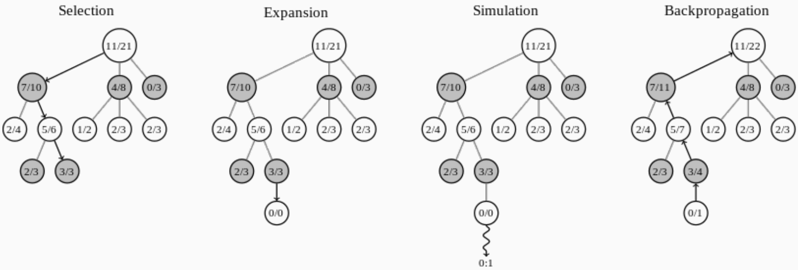
Current State of Markov Games

- Markov property, current state summarizes history

$$P(s_{t+1}, r_{t+1}|\Sigma_t, \ldots, \Sigma_0) = P(s_t, r_t|\Sigma_t) \tag{1}$$

- Given perfect information and players acting optimally, value function can be solved recursively by minimax tree search (approx b^d possible sequences of moves)
- Not possible for games such as Chess ($b = 35$, $d = 80$) and Go ($b = 250$, $d = 150$), need approximate solutions

Monte Carlo Tree Search



Each node contains count of successes and attempts

1. Simulate new game (start at root node), choose nodes, until reach new node (expansion)
2. Unless game ends, choose new action(s)
3. Upon termination, update successes and attempts

Simplification Strategies

1. Truncate tree and replace subtree with estimate of value at that node (reduce d)
2. Reduce breadth of search at each node by sampling from some prob. dist over actions (reduce b)

Strongest Go programs are based on MCTS

Adding more ML

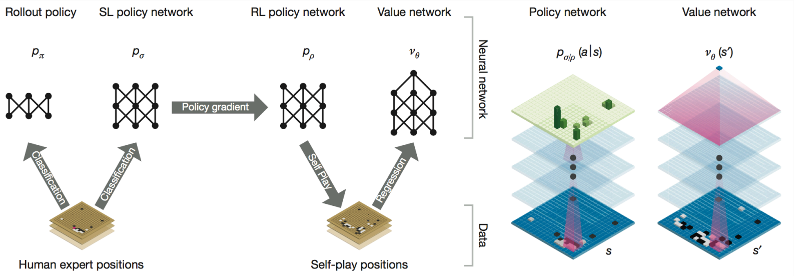
Standard explore-exploit tradeoff

- Previous attempts at ML using shallow + linear features,

$$v_\theta(s) = \varphi(s)'\theta \tag{2}$$

Contribution: use deep convolutional networks both to choose actions/reduce breadth (policy network) and estimate value at node/truncate tree (value network)

AlphaGo Architecture



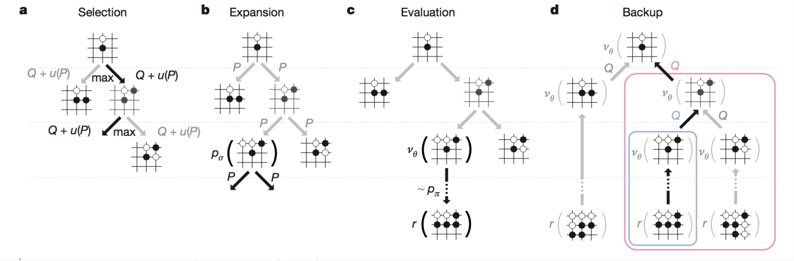
State is 19×19 image of board

- p_π: Fast sampling of actions from policy network
- p_σ: Policy network (SL) from supervised learning on predicting moves
    - accuracy of 57 percent on dataset of 30 million mves
- p_ρ: Policy network (RL) from Reinforcement Learning/self-play
- v_θ: Value network from self-play

Neural Networks are 13 layer Convolutional nets, with the same architecture between networks

Combining Deep Learning with MCTS

Final algorithm is MCTS combined with previous networks.



Each edge contains

- $Q(s, a)$, action value
- $N(s, a)$, visit count
- $P(s, a)$, prior probability

Running MCTS

Simulate a game, choosing actions the following way

$$a_t = \text{argmax}_a Q_(s_t, a) + u(s_t, a) \tag{3}$$

$$u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)} \tag{4}$$

Choose best action plus a term that is proportional to the prior but decays with visits (encourages early exploration).

Upon reaching non terminal leaf node, expand the leaf node using the SL

- store the SL probabilities as the prior probabilities
- determine the value of the leaf node by mixing the Value network and the outcome of a simulated rollout from fast rollout (z_L)

$$V(s_L) = \lambda v_\theta(s_L) + (1 - \lambda) z_L \tag{5}$$

At the end of the simulation, update the edges

$$N(s, a) = \sum_i 1(s, a, i) \tag{6}$$

$$Q(s, a) = \frac{1}{N(s, a)} \sum_i 1(s, a, i) V(s_L^i) \tag{7}$$

Note: Boatload of Neural Network evaluation, needs to occur asynchronously