Mastering the game of Go without human knowledge

Soheil

1 Main Ideas

AlphaGo is trained solely by self-play reinforcement learning, starting from random play, without any supervision or use of human data. Second, it uses only the black and white stones from the board as input features. Third, it uses a single neural network, rather than separate policy and value networks. Finally, it uses a simple tree search that relies upon this single neural network to evaluate positions and sample moves, without performing any Monte Carlo rollouts.

In self-play reinforcement learning a value function was trained by regression or temporal difference learning from training data generated by self play.

1. *States* Based on our table size (9 or 13 or 19), we have n*n cells that can have 3 values, black, white, empty. So, we have n^2 state values, and, therefore, it has 3^{n^2} states.

2. *Actions* In each rollout, we have two actions for each states (only if it's not occupied), to fill that state or to not play.

3. *Rewards* We get a -1 if we lose the game at the end, and a +1 if we win.

4. *Transition Probabilities* If we consider each board history as a state vector, the probability of going from one state vector to another is the Transition Probabilities.

2 Questions

1. What kind of problems would Monte Carlo Tree Search (MCTS) excel?

When dealing with a sequential process that has large state and action space, taking care of all possible action/state combination(tree) in the look ahead process in bellman equation is tedious. In Monte Carlo, instead of the whole tree, we only take samples, and base our calculations on samples. So, they drastically reduce the complexity of the bellman updates.

2. Why not use value iteration to win go?

It takes forever! In Go, in best case scenario we have (13*13)!*2 states. This is a gigantic state space which is practically unsolvable.

3. How would you improve this work?
No idea.