Research Review Paper: Mastering the game of Go with deep neural networks and tree search

Problem Statement:

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. These games may be solved by recursively computing the optimal value function in a search tree containing approximately b^d possible sequences of moves, where b is the game's breadth (number of legal moves per position) and d is its depth (game length) For Game Go where b ≈ 250 and d ≈ 150 exhaustive search is infeasible.

Brief Implementation:

In the paper author uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any look ahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. They also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, this program AlphaGo achieved a 99.8% winning rate against other Go programs.

They mainly used three stages of machine learning in their pipeline to train the model as mentioned below

Supervised learning of policy networks:

A 13-layer policy network, termed as SL policy network is trained on randomly sampled state-action pairs from 30 million positions from the KGS Go Server. The policy network takes input features from the board positions and uses stochastic gradient ascent to maximize the likelihood of the human move in that state . This outputs the probability of each move on the board being the actual human next move

The network predicted expert moves on a held out test set with an accuracy of 57.0% using all input features, and 55.7% using only raw board position and move history as inputs, compared to the state-of-the-art from other research groups of 44.4%

Reinforcement learning of policy networks

The second stage of the training pipeline aims at improving the policy network by policy gradient reinforcement learning (RL). Games are played between the current policy network and a randomly selected previous iteration of the policy network. Randomizing from a pool of opponents in this way stabilizes training by preventing overfitting to the current policy.

They have also used reward function which is 0 for non-terminal time steps , +1 for winning, -1 for loosing at step time t. Weights are then updated at each time step t by stochastic gradient ascent in the direction that maximizes expected outcome

When played head-to-head, the RL policy network won more than 80% of games against the SL policy network. Using no search at all, the RL policy network won 85% of games against Pachi (strongest open-source Go program)

Reinforcement learning of value networks

The final stage of the training pipeline focuses on position evaluation, estimating a value function that predicts the outcome from position of games played by using policy for both players. This neural network also had a similar architecture as policy network but it outputted a single prediction rather than probability distribution. To mitigate the problem, of overfitting they generated a new self-play data set consisting of 30 million distinct positions, each sampled from a separate game. Each game was played between the RL policy network and itself until the game terminated.

Searching with policy and value networks

AlphaGo combines the policy and value networks in an MCTS algorithm that selects actions by lookahead search.

The SL policy network performed better in AlphaGo than the stronger RL policy network. This was presumable because humans select a diverse beam of promising moves whereas RL optimizes for the single best moves. Also the value function derived from the stronger RL policy network performed than the value function derived from SL policy.