Zhuo Li
AI Nano-degree Isolation Project
September 26, 2017

RESEARCH REVIEW

Mastering the game of Go with deep neural networks and tree search

The research paper with title "Mastering the game of Go with deep neural networks and tree search" has introduced a new approach to build a new game agent named AlphaGo for the game Go using "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. The paper also introduced a new search algorithm that combines Monte Carlo simulation with value and policy networks. Based on these new approaches, the new game agent AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0.

The research team has built AlphaGo using an asynchronous multi-threaded Monte Carlo tree search (MCTS) that executes simulations on CPUs and computes policy and value networks in parallel on GPUs. The first version has used 40 search threads, 48 CPUs and 8 GPUs. There has also been a distributed version of AlphaGo that uses 1,202 CPUs and 176 GPUs. By applying the above implementation, the research team was able to efficiently combine MCTS with deep neural networks.

The new approach to the game Go described in the paper can be roughly described in the following four stages.

The first stage is supervised learning. The research team trained a 13-layer policy network, which they called the SL policy network from 30 million positions from the KGS Go Server. The network takes input features from the board positions and output the probability map of the board for the next possible moves. 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. Small improvements in accuracy led to large improvements in playing strength; larger networks achieve better accuracy but are slower to evaluate during search. The research team also trained a faster but less accurate rollout policy using a linear softmax of small pattern features with weights; this achieved an accuracy of 24.2% using just 2 micro seconds rather than 3 milli seconds.

The second stage is reinforcement learning. By using policy gradient reinforcement learning, the research team trained an identical structure RL policy network. The training data was from games between the current policy network and a randomly selected previous iteration of the policy network. Randomizing from a poll of opponents in this way stabilizes training by preventing overfitting to the current policy. When played head-to-head, the RL policy network won more than 80% of games against the SL policy network. Also, the RL policy network won 85% of games against Pachi, a sophisticated MCTS program ranked at 2 amateur dan on KGS.

The third stage is reinforcement learning of value networks. This is the final stage of training pipeline focuses on position evaluation, estimating a value function that predicts the outcomes from position of games played by using policy for both players. This neural network has a similar architecture to the policy network but outputs a single prediction instead of a probability distribution. The research team trained the weights of the value network by regression on state-outcome pairs using stochastic gradient descent to minimize the mean squared error (MSE) between the predicted value and the outcome. To mitigate the overfitting problem, the research team has 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. Training on this data set led to MSEs of 0.226 and 0.234, which indicates minimal overfitting.

The fourth stage is searching with policy and value networks. This is the core of this new approach of AlphaGo. It combines the policy and the value networks in an MCTS algorithm that selects actions by lookahead search. It is worth noting that the SL policy network performed better in AlphaGo than the stronger RL policy network presumably because humans select a diverse beam of promising moves whereas RL optimizes for the single move. However, the value function derived from the stronger RL policy network performed better than what is derived from SL policy network.

In summary, the paper has demonstrated the incredible power of combining MCTS with deep neural networks.