Summary

This is a review of “Mastering the game of Go with deep neural networks and tree search”. Original article consists of three main sections:

- neural networks training pipeline description;
- tree search algorithm description, including usage of trained neural networks;
- results analysis.

Training pipeline

Described game tree search algorithms makes use of four different neural networks:

- “SL policy” – neural network trained by supervised learning to predict expert human move in given Go position. Takes board state as an input and returns vector of moves probabilities as an output. Was trained against human expert games records.
- “Rollout policy” – less accurate but faster version of SL policy.
- “RL policy” – neural network trained by reinforcement learning to return the best move in a given Go position. As well as SL policy, takes board state as an input and returns vector of moves probabilities as an output. Initialized with SL policy results and then trained by playing games between different versions of RL policy players. Outperforms SL policy playing Go against it.
- “Value network” – neural network trained to predict Go position outcome. Takes Go position as an input and returns a single number – game outcome estimate. Trained against the results of RL policy games.

Search algorithm

Described search algorithm combines Monte Carlo Tree Search (MCTS) with smart usage of trained neural networks. Key algorithm features:

- Multiple MCTS simulations are run from a given position.
- At each simulation step action (next move) is chosen to maximize a value function, which is calculated as a weighted sum of three items:
    - Value network output for the action;
    - Outcome of up-to-the-game-end simulation produced by rollout policy;
    - Action probability returned by SL policy, which is divided by number of position visits during all simulations to encourage game tree exploration.
- At the end of simulation action values of explored game tree part are updated to reflect average value across all conducted simulation.
- When all the simulations are done (i.e. time limit is reached), the action with a maximum value function is chosen as a next move.

From the implementation point of view, search is executed in parallel to utilize a number of CPUs and GPUs.

Results

For a long time, game of Go was considered as a hard problem for AI systems because of enormous game tree size with branching factor about 250 and depth about 150. AlphaGo, the implementation of described approach, has beaten previous state-of-the-art Go programs in 494 out of 495 games. And even more impressive result is the AlphaGo wins in matches against European and, later, World Go champions. That was the first time Go computer program defeated professional human players with no handicap and on the full-size Go board.