AlphaGO Paper Summary (528 words)

Goals

- Beat the best human player in the game of Go, which is more challenging compared to chess, since branching factor of a game tree is much larger (approx. 250, compared to approx. 35 for chess) and depth is larger (about 150, compared to 80 for chess).
- Improve on previous best Go programs which used Monte Carlo tree search, which have achieved strong amateur play. Use the power of deep convolutional neural networks to do so.

Training the Model

AlphaGo trains a model in three steps:

Step 1: Supervised Learning on Policy Networks

First, they trained a 13-layer neural network called "policy network", where the input is a game position and the output is the probabilities of each of the next moves. This is trained from 30 million datasets of expert's moves (an example of supervised learning). They also trained a faster-to-compute but less accurate neural network called "rollout policy network", which has the same input/output and is used when playing the game.

Step 2: Reinforcement Learning of Policy Networks

Next, it improves the policy network using reinforcement learning. The game is played by two computer players - one using the current iteration of a policy network, and one using a previous iteration of a policy network. Use each policy network to play the game until the end, and update weights on the policy network to maximize the expected outcome. The final policy network generated at the end of this step was already beating the strongest open-source Go program 85% of the time.

Step 3: Reinforcement Learning of Value Networks

Finally, it generates a value network, where the input is a game position and the output is the estimated outcome - essentially an evaluation function. This is trained using regression on state-outcome pairs.

Playing the Game

To choose a position when playing the game, it computes two values - one directly from the value network generated from step 3, and the other using Monte Carlo Tree Search using rollout policy generated from step 1. Then mixes those two values to predict the final answer.

Results

To evaluate AlphaGo, they ran an internal tournament among variants of AlphaGo and several other Go programs, most of which use Monte Carlo Tree Search, with 5 seconds allowed per move. AlphaGo achieved 99.8% win rate.

As mentioned before, AlphaGo uses a mix of both value network and Monte Carlo Tree Search using rollout policy - and mixing each evaluation by 50% (`0.5 * value network + 0.5 * monte calro`) achieved the best rate, compared to just value network or just monte carlo.

AlphaGo defeated the human European Go champion by 5 games to 0, and the Elo rating of non-distributed AlphaGo was ~2890, which is as good as that of the best human player.

Compared to the match between Deep Blue and Kasparov in chess, AlphaGo evaluated thousands of times fewer positions when playing against Fan Hui, which is perhaps closer to how humans play. Furthermore, AlphaGo's evaluation function is generated from gameplay dataset, whereas DeepBlue's evaluation function was handcrafted - so no Go expertise was required to train a strong model.

Source

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore, and Hassabis, Demis. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, Jan 2016. Article.