AlphaGo Abstract

AlphaGo uses‘value networks’to evaluate board positions and ‘policy networks’to select moves. The‘value networks’and ‘policy networks are all deep neural networks which are trained by a combination of supervised learning (SL) from human expert games, and reinforcement learning (RL) from games of self-play. AlphaGo use a new search algorithm that combines Monte Carlo simulation with value and policy networks.

Firstly, AlphaGo trained a SL policy network p_σ directly from expert human moves. AlphaGo also trained a faster but less accurate rollout policy p_π, using a linear softmax of small pattern features with weights π; this achieved an accuracy of 24.2%, using just 2μs to select an action, rather than 3ms for the p_σ.

Next, AlphaGo trained a RL policy network p_ρ that improves the SL policy network p_σ by optimizing the final outcome of games of self-play.

Finally, AlphaGo trained a RL value network v_θ that predicts the winner of games played by the p_ρ against itself.

The policy network is trained on randomly sampled state-action pairs (*s, a*), using stochastic gradient ascent to maximize the likelihood of the human move *a* selected in state *s*.

The RL policy network p_ρ is identical in structure to the SL policy network, and its weights ρ are initialized to the same values, ρ=σ, it play games between the current policy network p_ρ 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. It use a reward function r(s) that is zero for all non-terminal time steps $t < T$. The outcome $z_t = \pm r(s_T)$ is the terminal reward at the end of the game from the perspective of the current player at time step t: +1 for winning and −1 for losing. 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. For the strongest open-source Go program, Pachi14, a sophisticated Monte Carlo search program, ranked at 2 amateur dan on KGS, that executes 100,000 simulations per move. Using no search at all, the RL policy network won 85% of games against Pachi. In comparison, the previous state-of-the-art, based only on supervised learning of convolutional networks, won 11% of games against Pachi.

The RL value network v_θ has a similar architecture to the policy network, but outputs a single prediction instead of a probability distribution. It trains the weights of the value network by regression on state-outcome pairs *(s, z)*, using stochastic gradient descent to minimize the mean squared error (MSE) between the predicted value v_θ, and the corresponding outcome *z*.

$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial \theta}(z - v_\theta(s))$$

AlphaGo generated a new self-play data set consisting of 30 million distinct states to avoid overfitting, 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 on the training and test set respectively.

AlphaGo combines the policy and value networks in an MCTS algorithm that selects actions by lookahead search. Each edge (s, a) of the search tree stores an action value Q(s, a), visit count N(s, a), and prior probability P(s, a).

To efficiently combine MCTS with deep neural networks, AlphaGo uses an asynchronous multi-threaded search that executes simulations on CPUs, and computes policy and value networks in parallel on GPUs. The final version of AlphaGo used 40 search threads, 48 CPUs, and 8 GPUs. It also implemented a distributed version of AlphaGo that exploited multiple machines, 40 search threads, 1,202 CPUs and 176 GPUs.

Eventually, AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0.