Research Review

I have read the three papers and in this summary will provide a brief review of all of them, but with emphasis on AlphaGo as it presents the state of the art in gameplay AI.

Game Tree Searching by Min/Max Approximation

This paper uses a different search algorithm from the techniques presented during the course. In Minimax all the options are explored to find the best action, Alphabeta optimizes Minimax by ignoring nodes that will not impact the final decision. Instead this paper uses iterative search, which iteratively tries to select a single best course of action, explores it then backpropagate the result of exploration and see if that course of action is still the best or another path should be explored. This paper uses a Minimax approximation with generalized mean-valued operator to guide the selection of next leaf node to expand. Generalized mean values is an approximation to min/max values, but is derivable, which is more suitable to sensitivity analysis. It is used to calculate weights to determine which node, the value in the parent node depends the most on, and selects it for expansion. After trying this technique with 1000 games of Connect-Four the Min/Max approximation with iterative search proved to be superior to Alphabeta for the same number of calls to the basic move subroutine, but Alphabeta wins if the bound resource is time, because it is faster and could explore more nodes for the same allotted time.

Deep Blue

This paper describes Deep Blue system, which defeated the chess world champion Garry Kasparov in 1997. The system combines specialized hardware with software to massively search a huge number of combinations for each move. The hardware consists of 30 CPUs, with 1GB of RAM and 4GB of disk per processor, 480 single chip chess search engines with 16 chip per processor, a move generator implemented as 8x8 array of combinatorial logic, which is effectively a silicon chess board. Deep Blue search is based on alpha-beta with ideas like quiescence search, iterative deepening, and transposition tables. The search algorithm is implemented as a hybrid software search with compiled C code running on CPU with hardware search implemented in the chess chip. Software search offers flexibility for modifications, hardware search is parametrized, but suffers from limitations of modifying hardware. For example hardware search lacks access to transposition tables, which is implemented in the software search. The algorithm is massively parallel with a single CPU running as the master distributing work to other processors. Deep Blue evaluation function recognizes roughly 8000 features, but implemented as a combination of fast, and slow evaluation. The fast evaluation computes only the easily evaluated major terms with high values when an approximation is good enough. The evaluation function is implemented in hardware on the chess chip. Deep Blue uses an opening book database created by hand primarily by Grandmaster Joel Benjamin and others. The book consisted of about 4000 positions. An extended open book with 700,000 was used to influence decisions in absence of opening book information by summarizing the information for a move in the extended book to nudge Deep Blue in the consensus direction of chess opening theory. Endgame database with all chess positions with five or fewer pieces was used off-line to design the chess chips, and the software search used the database on-line. Multiple factors contributed to the success of Deep Blue in 1997 including the large parallel search capability, complex evaluation function, endgame database, and the extended opening book.

AlphaGo

Introduction

Go game is considered the most challenging perfect information zero-sum game. Deep Blue was able to defeat the chess world champion in 1997, but the search space of chess is orders of magnitude less than Go with 35 branching factor, and 80 depth. Go on the other hand has a branching factor of 250, and depth of 150. The state of the art AI Go programs prior to AlphaGo used Monte Carlo Tree Search and could only beat amateur level human players. It was believed that we are a decade away from beating world class Go players. AlphaGo combined MCTS with two deep convolutional neural networks for policy selection, and state evaluation trained by Supervised Learning, and Reinforcement Learning to beat the state of the art programs even without any lookahead search. The neural networks are used to effectively reduce the depth and breadth of the search trees with a policy network for sampling actions, and a value network for position evaluation.

Network Architecture

The policy and value networks, are both 13 layers convolutional networks. The input to the network is a 19x19x48 image stack of 48 feature planes. Networks with 128, 192, 256, and 384 filters has been tried, with the match version network composed of 192 filters.

Training Pipeline

The policy network was trained with 30 million positions from KGS Go Server. The network predicted expert moves with 57% accuracy using

all input features, beating the state of the art 44.4% at date of submission. A lighter weight network was also trained with accuracy of 24.2% but takes only 2 µs instead of 3 ms to select an action. The second stage of the pipeline is a Reinforcement Learning training to improve policy selection and adjust the goal from predicting expert moves to winning games. The current network competes against a random selection of previous networks to prevent overfitting with a reward function to guide the gradients. The final stage is training the value network. The network has the same architecture as the policy network, but with a single output instead of a probability distribution. Training the value network was challenging because of dependency between moves, and training on full games resulted in overfitting where the network memorized the game data and failed to generalize. To mitigate this, RL was used and a sample of 30 million positions each sampled from different game sampled from self-play games of the strongest policy network against itself.

Searching with policy and value networks

Without lookahead search the value and policy network provided performance comparable to the state of the art at publish time, but to beat human champions AlphaGo combined MCTS with the neural networks. Evaluating the policy and value networks requires order of magnitude more computation than traditional heuristics. AlphaGo used asynchronous multi-threaded search executing MC simulations on CPU, and networks calculation on GPU. The final version used 40 search threads on 48 CPUs, and 8 GPUs.

Evaluation and Results

AlphaGo was evaluated against the strongest open source, and commercial Go programs. The result was a staggering 99.8% rate winning 494 out of 495 games. Variants of AlphaGo that uses only value networks without Monte Carlo rollouts and the result was exceeding the performance of all other Go programs, demonstrating that value networks provide a viable alternative to to Monte Carlo evaluation in Go. But combining both provided the best results, suggesting the two methods are complementary. Finally AlphaGo played a match against European Go champion Fan Hui. AlphaGo won 5-0 running on 1202 CPUs, and 176 GPUs. It is worth noting that although Go has more branching factor than Chess, AlphaGo evaluated thousands of times fewer positions than Deep Blue; compensating by selecting those positions more intelligently.