Summary of Thinking Fast and Slow with Deep Learning and Tree Search

We've all heard about AlphaGo Zero and what all it acoomplished - self-play, Monte-Carlo Tree Search, bells and whistles. I recently came across this research paper by another group that developed essentially the same thing and is considerably more approachable:
- Use tree search to select best move
- Train deep NN to make it predict same move as tree search does
- Next time tree search is applied it uses this NN to compute an approximation of Value function

There is self play and constant improvement of RL policy which results in strong player.

Goal

The paper aims to present Expert Iteration (EXIT), a novel reinforcement learning algorithm which decomposes the problem into separate planning and generalisation tasks. The paper also shows that compared to standard RL algorithms, which use neural network not only to generalise plans, but to discover them too; EXIT outperforms such algorithms for training a neural network to play the board game Hex(specifically REINFORCE) and that the ensuing tree search agent, trained tabula rasa, defeats MOHEX, the previous state-of-the-art Hex player.

Techniques

Exit attempts to mimic the process of human reasoning as described by the dual-process theory. According to this theory there are two systems that govern our reasoning process:

1. intuition/heuristic process - the fast, unconscious and automatic mode of thought resulting from expertise developed by experience
2. rule-based mode of reasoning - an evolutionarily recent process unique to humans, which is slow, conscious and explicit

It mimics this by separate planning and generalisation tasks. Planning new policies is performed by tree search, while a deep neural network generalises those plans. Subsequently, tree search is improved by using the neural network policy to guide search, increasing the strength of new plans. Tree search being the analogue to the rule-based mode of reasoning while the neural network performs the generalisation resulting in more intuitive play even for unexplored game states.

More specifically ExIt uses Monte Carlo Tree Search(MCTS) technique which uses repeated game simulations to estimate the value of states, and expands the tree further in more promising lines. When all simulations are complete, the most explored move is taken. When trying to do Imitation Learning(IL) from MCTS, the chosen learning target is called Tree Policy Targets(TPT) instead of the more traditional learning target used, viz simply the move chosen by MCTS, which the authors refer to this as chosen-action targets (CAT). TPT tries to match the

network output to the distribution over actions and is therefor cost-sensitive: when MCTS is less certain between two moves (because they are of similar strength), TPT penalises misclassifications less severely. This is a desirable property as it induces the IL agent to trade off accuracy on less important decisions for greater accuracy on critical decisions.

The online version of ExIt also uses data aggregation technique called DAGGER to prevent the loss of entire data sets and thus optimise run time since creating new data sets can add substantially to algorithm run time.

Results

The authors use the game of Hex to prove the game playing abilities of an agent trained using ExIt. All of their experiments were done on a 9x9 Hex game board.

Based on their initial dataset of 100,000 MCTS moves, CAT and TPT have similar performance in the task of predicting the move selected by MCTS, with average top-1 prediction errors of 47.0% and 47.7%, and top-3 prediction errors of 65.4% and 65.7%, respectively. However, despite the very similar prediction errors, the TPT network is 50 ± 13 Elo stronger than the CAT network, suggesting that the cost-awareness of TPT indeed gives a performance improvement. 4 We continued training the TPT network with the DAGGER algorithm, iteratively creating 3 more batches of 100,000 moves. This additional data resulted in an improvement of 120 Elo over the first TPT network. Our final DAGGER TPT network achieved similar performance to the MCTS it was trained to emulate, winning just over half of games played between them (87/162).

They also compared the time it takes for online ExIt to learn stronger policies as compared to batch mode and REINFORCE. They found that online ExIt substantially outperforms the batch mode, as expected. Compared to the ‘buffer’ version, the ‘exponential dataset’ version appears to be marginally stronger, suggesting that retaining a larger dataset is useful. They also found that ExIt learns stronger policies faster. EXIT also shows no sign of instability: the policy improves consistently each iteration and there is little variation in the performance between each training run. Separating the tree search from the generalisation has ensured that plans don’t overfit to a current opponent, because the tree search considers multiple possible responses to the moves it recommends.

Finally the researchers pitted an ExIt agent with 10,000 iterations against a 10,000 iteraion MoHex agent and won 75.3% of games. Against a 100,000 MoHex agent they won 59.3% but a 100,000 MoHex agent was also found to be 6 times slower than the ExIt agent.

Thus the paper shows that the ExIt algorithm significantly outperforms a variant of the REINFORCE algorithm in learning to play the board game Hex, and that the resultant tree search algorithm comfortably defeats the state-of-the-art in Hex play, despite being trained tabula rasa.