Taylor Del Matto
AI ND

Alpha Go Paper Summary

Minimax/AlphaBeta doesn't really apply well to alpha go because the search space is larger than chess, and because the evaluation of the game is much more complex so it is not easy to develop a heuristic that reduces the necessary depth of the search to play against humans effectively. Monte carlo Tree search is more effective for playing go at a weak amateur level. Monte Carlo Search tree evaluates the game with no prior knowledge of its mechanics by simply playing the game over and over again and creating a win probability tree based on how likely a given move is to produce a win.

Deep convolutional neural networks have performed very well in learning visually based problems such as image recognition and atari games. CNNs are used to reduce the depth and breadth of the search tree. To build AlphaGo, first a policy network is trained with supervised learning initially against training examples of human experts, and then refined with reinforcement learning playing itself- this optimizes the policy network towards actually winning opposed to probability. A policy network a network of probabilities of a move being played similar to the monte carlo search tree.

More specifically, probability of playing a move given a certain game state p(a|s). Then a value network is trained that predicts the winner of games by playing the reinforcement learning network against itself. The policy and value networks are combined with a Monte Carlo Search tree. The supervised learning Convolutional Neural Network was trained using stochastic gradient ascent, then the same network was trained on Reinforcement learning using stochastic gradient ascent again. The reinforcement learning was trained on previous iterations of the policy network to prevent overfitting.

After RL the network was able to win 80% of the games it played against earlier iterations of itself before RL. The RL trained policy network won 85% of the games it played against Pachi, the world's greatest open source Go algorithm prior to this. The SL network won 11% of games against pachi. Then reinforcement learning was used again, this time to try to predict the actual outcomes of games based on the positions. The Reinforcement learning initially overfit the data while training against the original data set the test error was considerably larger than the training error. The overfitting was fixed by generating a new data set played by the RL policy network against itself. Seeing this helped me comprehend a real world practical example of how overfitting can be a problem. The new test set had comparable errors to the training set, indicating that there was no overfitting and the test error was also substantially reduced. Then this value network was combined with a monte carlo search tree which performs fast rollouts at every move.

The rollouts were performed on a fast rollout policy network p_{pi}. An equation was required to combine input from the monte carlo search tree and the policy networks

Taylor Del Matto
AI ND

i

The most successful combination equation used a lambda of .5. This means the most successful combination equation took equal parts input from the monte carlo search tree and the neural network.   This lambda won more than 95% of games against other variations of the equation, and possibly other factors. The algorithm demolished the professional Fan hui 5 - 0. The training using supervised and reinforcement learning was a new approach. AlphaGo's position evaluation mixes full rollouts with partially complete rollouts that default to a value function.