Reliability Analysis on Temporal Difference Method for computer Go

Yuetong Liu

03/03/2020

Abstract

Monte-Carlo tree search is a recent algorithm that has been used to achieve play in computer Go. It uses the mean outcome of simulated episodes of experience to evaluate each state in a search tree. However, long-range spatiotemporal interactions in Go could make position evaluation extremely difficult. As one of the most broadly applied reinforcement learning methods, temporal-difference learning updates a value function from episodes of real experience, by bootstrapping from future value estimates, and using value function approximation to generalize between related states. In this research, we want to implement a temporal-difference search in computer Go. Besides updating the value function from simulated experience, it uses value function approximation and bootstrapping to efficiently generalize between related states. To evaluate the performance of the proposed method, we will define a set of metrics that quantitatively measure different aspects of reliability and design complementary statistical tests to enable rigorous comparisons on these metrics. We apply our metrics to both Monte-Carlo tree search and temporal-difference search in computer Go, compare them, and analyze the results.

1 Accomplishments (Summary)

2 Introduction

2.1 Go

Go is an abstract strategy board game for two players in which the aim is to surround more territory than the opponent. One player uses the white stones and the black. The players take turns placing the stones on the vacant points of a $19 * 19$ board. Once placed on the board, stones may not be moved, but stones are removed from the board if the stone (or group of stones) is surrounded by opposing stones on all orthogonally-adjacent points, in which case the stone is captured. The game proceeds until neither player wishes to make another move. This game has long been viewed as the most challenging game for artificial intelligence because of its enormous search space. Two common strategies for this game are supervised learning from human expert games and reinforcement learning from games of self-play.

2.2 Markov Decision Process

A Markov decision process (MDP) consists of a set of states S and a set of actions A. In Go, the state object is a $6 * 19 * 19$ numpy array. All values in the array are either 0 or 1. The first channel represents the black pieces. The second channel represents the white pieces. The third channel is a indicator layer for whose turn it is. The fourth channel shows invalid moves for the next action. the fifth channel is a indicator layer for whether the previous move was a pass. The sixth channel is a indicator layer for whether the game is over. The action object a list of 2 integers representing the row and column or "None" for passing. From any state s and for any action a, are determined by transition probabilities specifying the distribution over the next state s.

A reward function specifies the expected reward for a given state transition. If the game is ongoing, the reward is black area - white area. If black won, the reward is 19^2. If white won, the reward is -19^2. If tied, the reward is 0. A policy maps a state s to a probability distribution over actions. The value function is the expected return from state s when following policy. The action value function is the expected return after selecting action a in state s and then following policy .

TODO: Back up Diagram?

3 Mathematical Development

3.1 Monte-Carlo Tree Search

3.2 Temporal Difference Learning

3.3 Neural Network

4 Experimental Approach and Results

4.1 Training Procedure

4.2 Reliability Metrics

4.3 Statistical Test

4.4 Result

5 Next Steps

References

[1] Schraudolph N.N., Dayan P., Sejnowski T.J. (2001) Learning to Evaluate Go Positions via Temporal Difference Methods. In: Baba N., Jain L.C. (eds) Computational Intelligence in Games. Studies in Fuzziness and Soft Computing, vol 62. https://doi.org/10.1007/978-3-7908-1833-8_4

[2] Chan S.C.Y., Fishman S., Canny J., Korattikara A., Guadarrama S. (2020) Measuring the Reliability of Reinforcement Learning Algorithms. In: International Conference on Learning Representations. https://openreview.net/forum?id=SJlpYJBKvH

[3] Silver D., Sutton R.S. Müller M. (2012) Temporal-difference search in computer Go. In: Mach Learn 87, 183–219. https://doi.org/10.1007/s10994-012-5280-0