The Project of Gomoku

Zhang Yanjian
School of Data Science
Fudan University
16300200020@fudan.edu.cn

Lin Geng
School of Data Science
Fudan University
16307100060@fudan.edu.cn

Abstract

The abstract paragraph should be indented 1/2 inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

The simplest Gomoku game is that if the two sides do not limit the conditions, whoever puts five pieces of the same color in one line, whoever wins. However, Gomoku has been proven to have a winning hand in the absence of restrictions, so Gomoku's game has also added a number of restrictions to reduce the advantage of the first mover. With the development of Gomoku, variants such as Renju have also begun to heat up.

In the area of chess, AI is a very hot topic, as is the case in Gomoku. Gomoku's rules are simple, but it still have a very large search space, so it's a good idea to try out various algorithms. A popular algorithm is Monte-Carlo tree search[1], which obtains the score of each candidate position through a large number of simulations by means of random movement. Alpha-Beta Pruning can significantly reduce the search space by removing unnecessary searches, which can effectively increase the search depth. In addition, Genetic algorithms[2], Reinforcement Learning[3], etc. have also been applied to Gomoku AI. Reinforcement Learning has been applied in many fields. With the development of machine learning, Reinforcement Learning has also become more prominent in Gomoku. This method calculates the score of the candidate position by encoding the board state and setting reward to train the relevant weights of the board state. Threat is an early method that focuses on the offensive and defensive patterns[4]. It doesn't take into account the long-term state, and you can get the next step with very little calculation. The algorithm combines some of Gomoku's expert experience, and strives to construct continuous killings in the short term and block the possible threats of the opponent, which has a good performance in the later game.

Our model uses Threat as the main algorithm. With the Reinforcement Learning and the strategy we call QI, we implement a Gomoku AI with less computation and speed, and do some research and experiments. In Section 2 we will introduce the Threat algorithm and implementation. In Section 3 we will explain how the QI strategy can help us choose the next step without threats, and in Section 4 we will introduce the implementation details and applications of Reinforcement Learning.

2 Threat Policy

We use the threat policy based on previous work[4]. We make a further step for their work, apart from the threat of our chess as they describe, we also find the threat of the opponent chess.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2.1 searching steps

We divide the threat into 3 level:

1. A must-win threat like 11111, we give priority 1 for this situation

2. A must-win threat like 011110, we give priority 2 for this situation

3. Two threat overlap, this is also a must-win policy in suitable time, we give them the priorities varying from 3.0 to 4.2 in different patterns which can be seen from our code.

4. Two rest for a new threat (combination), we can win when two chess rest together with appeared chess form a new threat

5. Three rest for a new threat (combination), we can win when three chess rest together with appeared chess form a new threat

If satisfy the above 5 conditions, we will return a certain action in both our side and the enemy's side. If not satisfied, we will construct a threat and return the list of points that may lead us to a new threat.

2.2 selecting steps

After searching, we have the threats which are surely lead us to win (our threat) and need swift prevention (opponent's threat). Generally, the highest priority one is chosen as our next step. We will make our choice considering the priority of our threats and the opponent's.

The threats which are searched out can be an action or a list of threat.When our threat list and opponent threat list have overlaps, we tend to choose the overlapping place for our next steps. In this way, we can construct our own offensive threats and destroying opponent's threats, improving our offensive efficiency. And when we construct our own offensive threats, we can also consider whether we can block the threat of the opponent. If there is no such overlap, and we have multiple choices of the same priority, then we enter each action into the network of Reinforcement Learning, get the score of each action through two MLP layers, and select the action with the highest score.

2.3 Usage of the priority

Now we have two kinds of threats given by our chess and the opponent's chess along with their priority. When our threats have a higher priority, we will adopt an offensive strategy, construct an offensive threat, and try to find a threat combination to win the game. When the opponent's threat has a higher priority, we will adopt a defensive strategy to destroy the opponent's threat. When the threats of the two sides have the same priority, we will still take the offensive approach, because we have the advantage of the first move when we search for it. In the process of dealing with threat priorities, we made some improvements, such as giving higher priority to the threat of three consecutive pieces, allowing AI to reduce vulnerabilities when making choices.

3 Policy of Qi

We use the Chinese character Qi (the breathing air) to describe the our searching policy when no threat appears. As we can see from Figure 1, the black chess have 3 same color chess in its 5×5 zone. Because of the block of the white chess, the dark chess can not form a Qi in the right direction. however, it can form the Qi in the other two direction. We calculate the number of Qi in our left choices, and use the highest Qi for our next step.

4 Reinforcement Learning

4.1 implementation

We implemented reinforcement learning through the ADP(Self-teaching adaptive dynamic programming) structure(Figure 2).

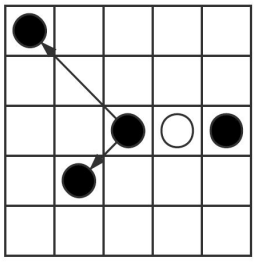

Figure 1: illustration of Qi

Through the structure of ADP, we can learn the relevant parameters through self-matching, and learn the parameters in the games with people or AI in the later stage.

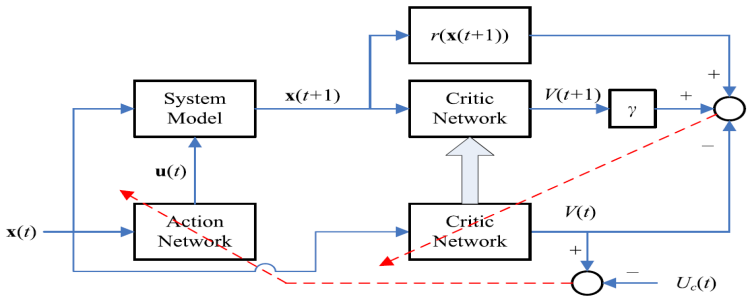

Figure 2: ADP structure

4.1.1 Action Network

Our model includes a part of reinforcement learning to make the final choice in some cases with a variety of similar chess methods. This step is critical.

When no threat appears on the board, our model will attempt to create an offensive pattern based on the aforementioned logic rules, and there are often multiple choices. We select the candidate results through reinforcement learning to get the one with the highest score.

When the threat test returns three rests close to each other and does not overlap with any of the rest of the opponent, we will select a more suitable rest by the parameters of the reinforcement learning.

4.1.2 System Model

When the action network give a action to us, we can capture the state of the next step, and generate our embedding We first define the sequences of the input of neural network. Based on Mo's research, we searched 16 patterns and generated an embedding of 218 dimensions, used to represent the state of the board. For example, the patterns include [10111], [11111], [01110] and opponent's [20222], [22222], etc.

4.1.3 Critic Network

The 218 dimensions embedding for the state is the input of the critic network, which is an neutral network contain an input layer of 218 dimension, a hidden layer of 60 dimension and a output layer of 1 dimension.

During the evaluation time, we use it to evaluate the value of each next steps. During the training time, we also use the current state to generate the current value of the whole chess. We use both of the value to make a fitting for Bellman Equation:

$$V(t) = r(t+1) + \gamma V(t+1)$$

The error of the final layer can be illustrated as follows:

$$e(t) = \alpha[r(t+1) + \gamma V(t+1) - V(t)]$$

Our reward is 1 in victory and 0 in failure. The discount rate γ is 1 for our implementation and The learning rate α is

We use Numpy for training version and get the weight matrix for neutral network for our fast version which just implement the evalution.

5 Experiment

We competed with some of the more famous AIs, and fixed the following three openings to reduce the advantage of the first mover.

The AI we used for the experiment included Eulring16, PISQ04, PureRocky, Valkyrie13, Mushroom11, Fiverow08, and the following results were obtained(Our AI is temply called LZ).

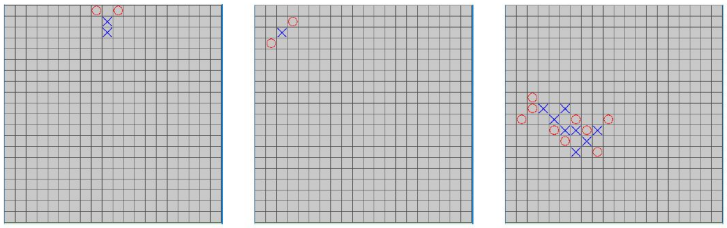

Figure 3: Three kinds of opening

Table 1: Symbols description

AI	Our model (WIN/LOSS)
$Fiverow08$	93 : 7
$Mushroom11$	92 : 8
$Valkyrie13$	74 : 26
$PureRocky$	77 : 28
$PISQ04$	61 : 39
$Eulring16$	5 : 95

From the results, the AI of Threat with Reinforcement Learning has achieved relatively good results, but there is still a big gap compared with the current AI. This illustrates the importance of deeper search for improving AI performance. The Threat algorithm has certain limitations. Interestingly, when Threat plays a game with a simple human player or a similar algorithm, it often takes a lot of steps to win. This shows that Threat's model is closer to the general human mind and more prominent on the defensive.

6 Conclusion

We use the Threat algorithm, combined with reinforcement learning and QI strategies, to achieve a fast and effective Gomoku AI. The Threat algorithm can try to find continuous threats and blocking strategies, which are very sensitive to offensive and defensive modes. Reinforcement Learning yields the better of several similar choices, and the QI strategy helps AI make choices without threats. However, there is still much room for improvement in this algorithm. Since the lack of calculation of long-range steps, this algorithm combination is relatively short-sighted. It is easy to ignore some winning modes, or it does not perform well at the beginning. When

confronted with some stronger algorithms, we observed the struggle of this algorithm at the beginning. If we choose a more ingenious algorithm at the beginning of the game, or set up some rewards in Reinforcement Learning for the opening, and later add a search for the winning mode, the AI can use some calculation time to improve the performance.

7 Reference

[1] Effective Monte-Carlo tree search strategies for Gomoku AI(2016), J H Kang and H J Kim.

[2] Evolving Gomoku Solver by Genetic Algorithm(2014), Junru Wang and Lan Huang.

[3] Self-teaching adaptive dynamic programming for Gomoku(2012), Dong-bin Zhao, Zhen Zhang, and Yujie Dai.

[4] Allis, L.V., Herik, H.J. and Huntjens, M.P.H., 1993. Go-moku and threat-space search. University of Limburg, Department of Computer Science.