Neurocomputing 78 (2012) 23–29

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Self-teaching adaptive dynamic programming for Gomoku

Dongbin Zhao, Zhen Zhang*, Yujie Dai

State Key Laboratory of Intelligent Control and Management of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Available online 25 August 2011

Keywords:
Gomoku
Reinforcement learning
Adaptive dynamic programming
Temporal difference learning
Neural network

ABSTRACT

In this paper adaptive dynamic programming (ADP) is applied to learn to play Gomoku. The critic network is used to evaluate board situations. The basic idea is to penalize the last move taken by the loser and reward the last move selected by the winner at the end of a game. The results show that the presented program is able to improve its performance by playing against itself and has approached the candidate level of a commercial Gomoku program called 5-star Gomoku. We also examined the influence of two methods for generating games: self-teaching and learning through watching two experts playing against each other and presented the comparison results and reasons.

1. Introduction

Gomoku is a board game that originated from one of the various kinds of black and white chess games in ancient China. Nowadays, it has become a popular game played in many places of the world. Gomoku is played between two players on a 15×15 square mesh. A sufficient amount of black or white pieces are offered to each player. The players in turns place one piece on the board. The winner is the first who forms a line of at least five adjacent pieces of his color, in horizontal, vertical or diagonal directions on the board. Such winning line is called five-in-a-row which is also referred to the name of the game Gomoku. If the board is filled completely with pieces and no one gets a five-in-a-row, this game is a draw. Gomoku has simple rules but high complexity as well, which makes it a suitable test-bed for a wide class of artificial intelligent algorithms.

One popular algorithm of playing Gomoku is game tree searching, which is often combined with a board evaluation function of leaf board situations. This method searches the partial game tree with a root node of the current board situation. If there is a leaf board situation with five-in-a-row of its colors, the move sequence leading to this situation is returned. Otherwise a board evaluation function is used to select an optimal position among all leaf board situations. According to William [1], a complete search to a depth of n moves need evaluation of $p!/(p-n)!$ board situations, where p is the number of legal positions a piece can occupy. Thus a complete game analysis is impossible. The history heuristic and alpha–beta search can be used to speed up game tree searching [2]. Another algorithm was proposed by Allis and Herik [3]. The core idea of this algorithm is to search winning threat sequences. Besides, Freisleben presented a neural network that was able to learn to play Gomoku [4]. This neural network is penalized or rewarded according to a special reinforcement learning algorithm, called comparison training [5].

Reinforcement learning (RL) is to learn how to map an action to a state to get the maximal reward from interacting with the environment. The merit of this algorithm lays on the fact that no explicit teacher is needed during the learning process so we can expect to get a more adaptive system using reinforcement learning algorithm. The difficulty of it is that the learning system must solve a temporal credit assignment problem, i.e., assign credit to each of the states and actions that result in the final outcome of an episode. However, reinforcement learning has attracted considerable interest for many years. Apart from its application to many practical problems, it has also been used as a model for explaining the action-selection mechanism of the brain [6]. Temporal difference (TD) learning algorithm is one way to assign temporal credit. This learning method utilizes the difference between two temporally successive predictions. One of the most successful applications of TD learning algorithm is TD-Gammon, a program that plays the game called backgammon. The latest version of TD-Gammon approaches the level of the world's best human players [7–9]. TD learning has been applied to Gomoku by Mo [10], but the results are not as exciting as in TD-Gammon. TDLeaf(λ), a variation on the TD(λ) algorithm, has been applied to a chess program called KnightCap [11]. It is thought that KnightCap's success is partially contributed by the appropriate choice of initial weights [12]. To select initial weights to speed up training, a Mini-max initialization method is proposed [13]. Another TD learning algorithm, kNN-TD(λ) method has been proposed and proved a robust high performance implementation of RL algorithm [14]. In TD learning, the action decision or the value function can be also described in continuous form, approximated by nonlinear functions such as neural networks. This is the key idea of Adaptive Dynamic Programming (ADP), first proposed in [15–17]. ADP has

* Corresponding author.
E-mail addresses: dongbin.zhao@ia.ac.cn (D. Zhao), zhangzdlut@gmail.com, zhen.zhang@ia.ac.cn (Z. Zhang), daiyujie07@mails.gucas.ac.cn (Y. Dai).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.05.032

been applied to many fields such as missile guidance [18] and pendulum robots [19].

We present a program ST-Gomoku employing ADP to learn by playing against itself to be a good player for Gomoku. ST-Gomoku utilizes a neural network to evaluate board situations and decide which move is supposed to be taken next. By playing against itself, the neural network is trained to learn the winning probability of any possible board situation. During the training process, one player should select the move that leads to the board situation with his maximal winning probability. On the contrary, the other player should choose the move that leads to the board situation with the minimal winning probability of his opponent. It will be shown that ST-Gomoku approaches the candidate level of a commercial Gomoku program called 5-star Gomoku. Using ST-Gomoku as an expert, we also test and compare two different methods for generating games used for training: (1) self-teaching and (2) learning through watching two experts playing against each other. We did the comparison under the inspiration of Wiering's research on backgammon [20], with the objective to find out which method combined with temporal difference learning can gain the best performance and the reasons behind it.

This paper is organized as follows. Section 2 gives a brief review of RL algorithm. In Section 3, we provide the topology of the evaluation neural network and how to train it. In Section 4, the experimental results indicating the performance of the obtained expert program and the comparison results between two different methods of generating games are presented and discussed. Section 5 concludes this paper and points out the direction for future research.

2. RL and ADP

RL uses cumulative reward $R(t)$ to evaluate the policy followed by an agent. It is defined as follows:

$$R(t)=r(t+1)+\gamma r(t+2)+\gamma^2 r(t+3)+\cdots=\sum_{(k=0)}^{T}\gamma^k r(t+k+1) \qquad (1)$$

where γ is the discount factor within [0, 1], T is the ending time of an episode and $r(t+1)$ is the reward received at time $t+1$. The cumulative reward is often called cost-to-go function, which is represented by $V(t)$ in ADP. The most basic structure of ADP is heuristic dynamic programming(HDP), whose structure is shown in Fig. 1.

The current system state $\mathbf{x}(t)$ is fed forward to the action network, which generates the control action $\mathbf{u}(t)$. The system model is used to predict the next step transition state $\mathbf{x}(t+1)$, which is fed forward to the utility function r to produce a reward $r(\mathbf{x}(t+1))$. The critic network is used to estimate the cost-to-go function V based on the system states only. Then the reward $r(\mathbf{x}(t+1))$, the estimation $V(t)$ and $V(t+1)$ are used to update the weights of the critic network to make the cost-to-go function V satisfy the Bellman equation. The difference between $V(t)$ and the desired objective $U_c(t)$ is used to train the action network to produce the optimal policy. Note that first, there can be no system model network in this structure and second, the critic network and the action network are not restricted to a particular structure.

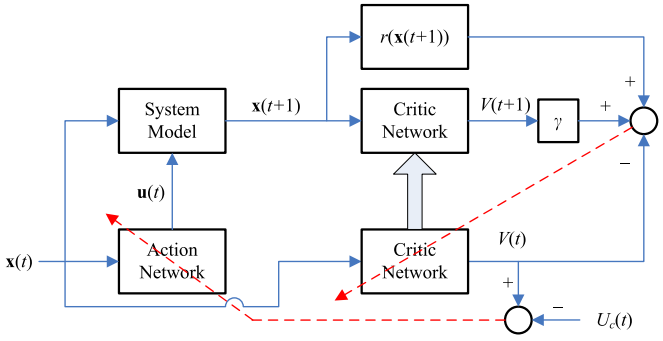

Fig. 1. The HDP structure.

3. Self-teaching ADP for Gomoku

In our self-teaching ADP for Gomoku, the critic network is a neural network which is used to evaluate board situations. The action network is not a neural network. It works together with the critic network to determine an action, which will be elucidated in Section 3.3. Hereby, the neural network means the critic network if there is no explicit explanation.

3.1. The state

To evaluate a board situation, the first important issue is how to describe it. Based on the research by Mo [10], we extend previous patterns to 20 patterns for each of the two players, that is, a total of 40 patterns to describe a board situation. Eight typical patterns are shown in Fig. 2. Each pattern is unique and consists of five or six adjacent positions in a line which are only occupied by the pieces of the same color. The number of each pattern in horizontal, vertical and diagonal lines can be identified to partially describe a board situation. Besides, whose turn to move is another important piece of information for describing a board situation. Finally, the factor of whether a player is in the offensive or in the defensive is added, considering the fact that in Gomoku the strategies are quite different between the offensive and defensive players. Therefore, patterns, turns and the offensive/defensive issue together constitute a state which is a generalized description for a board situation.

The patterns used in this paper are carefully selected. Though several patterns may not be accepted by a Gomoku master player, we still use them because on the one hand some standard patterns are complicated and recognition of them is time-consuming, on the other hand the patterns we use are simple and can be recognized with ease. We use patterns instead of raw information for three reasons. First, patterns reflect the essence and features of the game in a more readily comprehensible way than raw information does. Second, it is expected that raw information has a much broader state space than patterns have. Finally, we have attempted to use raw information to describe board situations. However, the results are not satisfying.

The turn, that is, whose move is it, is another important state variable for describing board situations. This issue is crucial for Gomoku in particular. For example, if both sides have already got a pattern e in Fig. 2, then clearly the winner is the one who is to move. We expect that the evaluation function for Gomoku is not as smooth as the one for backgammon and that two input nodes are not enough to reflect the importance of turn. Therefore, for each pattern we assign two input nodes to represent the turn. The offensive/defensive issue decides the opening of a game to a large extent. In some chess programs, an opening book is utilized to

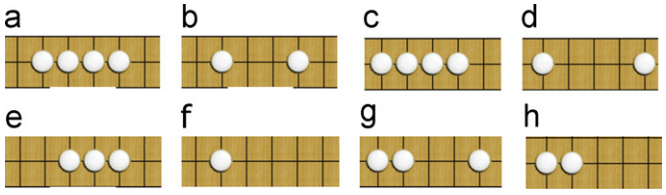

Fig. 2. Some patterns used in the presented program.