JAIST Repository
https://dspace.jaist.ac.jp/

Title	強化学習を用いたコンピュータ麻雀プレイヤ
Author(s)	山田，渉央
Citation	
Issue Date	2018-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/15213
Rights	
Description	Supervisor:池田　心，情報科学研究科，修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Build One Player Mahjong Player Based on Reinforcement Learning

Shouo Yamada (1510056)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 9, 2018

Keywords: Computer Mahjong, Machine Learning, Reinforcement Learning.

Extended Abstract

When creating computer game players that play with/against human players, the main and ultimate goal is usually to entertain the human players. While there are many conditions needed for entertaining human players, the most elementary one is that computer players have sufficient skills compared to human players. For many games, it was a difficult task to make such strong computer players, so research for strong computer players has been intensively done.

In the case of Chess, IBM Deep Blue won against the human world champion in 1997, and in the case of Shogi (Japanese Chess), many programs showed excellent performance in games against professional players. Finally, in the case of Go, AlphaGo from DeepMind won against the human world champion in 2017. These three games are two-player, perfect information and zero-sum games. In such games, we can conclude that the method of making very strong computer players has been achieved. Therefore, researchers interests have now shifted to other types of games, especially more complex games in a way. One of such interesting games is Mahjong.

Mahjong is a four-player, imperfect information and stochastic game. In addition, one game of Mahjong is composed of about 10 sub-games, and the total scores of the sub-games are compared. So, players should try to win single sub-games (by completing their hand), but it is often more important to change the strategy according to the current total score situation. For example, if a player has the highest score after some sub-games, it is more important to win earlier (even if small score gain) and to avoid being exploited, than to win with big score gain. In contrast, if a player has a lower score after some sub-games, it is more important to win with a big score gain, even if the probability is small, or slow. Of course, an intermediate strategy between these two strategies is sometimes optimal. Since various strategies for various situations are needed, and since many local rules are employed for actual Mahjong games, it is difficult to prepare good strategies by hand, or by using supervised learning.

Copyright © 2018 by Shouo Yamada

In this research, reinforcement learning is employed for preparing strong Mahjong computer players, with various strategies for various situations. We can obtain various strategies by changing the "reward function" and some parameters. In the actual experiments, we employed two different reward functions when a hand is completed: 1) fixed amount of reward, 2) score actually gained by the player. It is expected that a "strategy for early completion" is learned in the former case, and a "strategy for bigger score gain" in the latter case. In addition, it is expected that an intermediate strategy can be obtained by changing a parameter called discount factor gamma.

At first, we applied reinforcement learning to some simplified, small scaled and one-player Mahjong games, since experiments for such simplified games can be executed in shorter time, and theoretical values of state-action (Q) values can be strictly computed and compared to the result. Five simplified games are designed for this purpose, and several reinforcement learning methods are evaluated and compared. By using such games that have different scales of state-action spaces, we can understand the advantage and disadvantage of reinforcement learning methods.

The first method is called "table-type" Q-learning, where all state-action pairs are stored independently in a big array table. Each Q-value can be updated without changing other values, so that true Q-values can be represented and learned after sufficient learning episodes. On the other hand, if the state-action space is very big, such table cannot be stored in the PC memory, or a lot of episodes are needed for experiencing all the state-action pairs. This table-type Q-learning worked successfully in 3-tiles Mahjong and 5-tiles Mahjong, with about 100,000 state-action pairs, but could not work well in 8-tiles Mahjong.

The second method is called "feature-type" Q-learning, where each state-action pair is converted to a feature vector, and the inner product of the vector and a weight vector is used as the Q-value. A 76-dimension vector is used for 5-tiles Mahjong and a 112-dimension vector for 8-tiles Mahjong, respectively. The number of parameters for representing the policy is significantly smaller (76 and 112) than the first method, then there is no problem about PC memory. On the other hand, when a Q-value is updated, it means that other Q-values are also changed, since the common weight vector is used. By this limitation, it is difficult or usually impossible to represent theoretically optimal Q-values. In fact, comparing the two methods for 5-tiles Mahjong, the average moves for completion was 11.9 (table-type) and 12.4 (feature-type) respectively, which means that table-type was better in this small game.

We also employed 14-tiles Mahjong, which is the usual size of hand, but in a one-player game. Since table-type cannot be used, feature-type with 192 dimension vector was used. We employed two types of rewards, 1) fixed and 2) gained score, for 1) early completion and 2) big score completion. The average number of moves until completion was 30.3 and 34.8 respectively, meaning that 1) completed hands earlier, and average score at completion was about 3300 and 7700, meaning that 2) got higher scores, as expected. In addition, we modified the discount factor from 0.9 to 0.7 in order to guide the learning to earlier completion. Then the average number of moves became 32.6 and the average score became 4200. This means that an intermediate strategy was obtained, as expected.

Finally, the third method is called "neural network type", where feature vector is used but the Q-value is calculated by a neural network, instead of simple linear product. By this

method, representation ability, generalization ability and then performance are expected to be improved. At the same time, it cannot be helped that the calculation cost is significantly expensive compared to table-type or linear-feature-type. After applying this method to 8-tiles Mahjong, we could observe fast improvement of winning ratio. To achieve 90 % winning ratio, linear feature type required about 600,000 episodes (games), and neural network type only 30,000 episodes, while the actual learning time was not so different.