YARAL: Yet Another Reinforcement learning Approach to Ludo

Nathan Durocher

University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
nadur20@student.sdu.dk

Abstract. This paper details an approach for creating an intelligent Ludo player using the Q-learning method. The player was designed with an aggressive playing style through high rewards for high progress moves and for sending the opponent home while receiving low to no rewards for wasteful moves. When evaluated the player produced a 83% and 68% win rate against a random selection player and another Q-learning intelligent player respectively. The effects of action and state space design and other learning parameters were examined.

1 Introduction

Artificial intelligence, also referred to as machine learning, is the area of software research and development that allows computers to develop their own decision making process for a specified task. These tasks range from picking the next personalized ad to show you while browsing the internet, to assisting artists with painting or creating music. While these tasks seem completely unalike, they can both be achieved by providing a system enough relevant data to the machine learning algorithms and letting the software make hundreds of thousands or millions of classifications to produce a desired decision or product. As these algorithms and systems become more common and more sophisticated, there will be a growing need for people who understand how these systems are working to be able to create and manage them. Therefore, as an introduction to the topic, this project to create an intelligent player for the board game *Ludo* was assigned to better understand how these systems are created and how they work. This paper details a player created using the reinforcement learning method called Q-learning. Through the use of Q-learning the task was to achieve the high win rate against one to three random selection opponents. The player is also to be compared to another intelligent player created by a colleague.

2 Methods

2.1 Game Space

Given any situation possible within the game the player must to be able to make the best possible move to progress towards the goal. For a Q-learning method this is described as state-action pairs. To determine all possible pairs, a state space for the player's pieces must be created. The approach taken simplifies the game for each of the pieces into four possible states:

1. Home
2. Goal Zone
3. Safe
4. Danger

The home state refers to when the piece is not in play and can only be moved when a six is rolled. A piece is in the goal zone when it occupies one of the final six spaces before the goal. The safe state is when the piece is either on a globe tile or when it is not within six tiles of an opponents piece such that it could be sent home by the next roll. Finally, the danger state is defined when the pieces is within striking distance of an enemy piece or on the coloured globe tile at the opponents home. Completing the state-action pairs, the action space is described by ten possible moves:

1. Open
2. Normal
3. Goal
4. Star
5. Globe
6. Protect
7. Kill
8. Overshoot Goal
9. Goal Zone
10. Null

The open, normal, goal, star, globe, protect and kill moves are all standard playing moves defined by the rules of the game. However, the overshoot goal and goal zone moves are added classification moves to effect the priority of moves near the goal. In particular the overshoot goal move is used as an unwanted action that occurs when a piece is in the goal area but the dice roll is higher than the number of tiles needed to reach the goal. Without this action definition these moves would fall under the goal zone move since the piece being moved will land in that area. This can cause issues as generally a move into the goal area is a desirable action, while overshooting causes the piece to move farther away from the goal. Finally the null action is a placeholder for when the player has no active pieces to move. Using these state-action pairs a game space was defined for each of a players four pieces resulting in a 160 Q-values parameters that require training.

2.2 Rewards

The method used to train the player follows the standard Q-learning temporal difference rule as given in equation 1 to update the values.

$$\Delta Q(s,a) = \alpha(r_1 + \gamma \max_{a1} Q(s_1,a_1) - Q(s,a)) \tag{1}$$

Where α is the learning rate, r_1 is the reward for selecting a given action and γ is the discount for future rewards. Each of these values were selected based on initial numbers given in Poramate Manoonpong's lecture six notes [1] and adjusted during testing. For the learning rate α a value of 0.4 was chosen to reduce overshoot and osculations and ensure the convergence of the Q values while not requiring an large amount of training games.

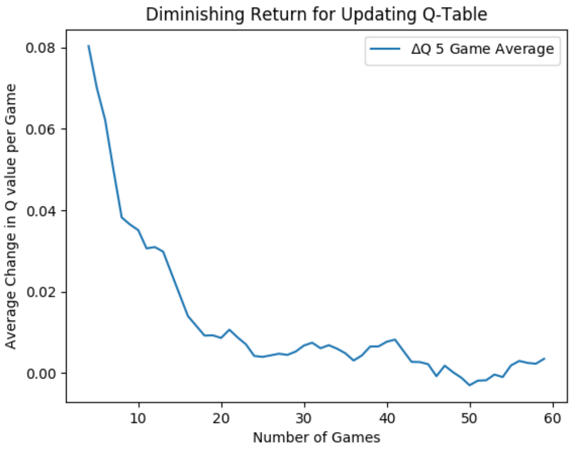

Fig. 1: Average Q value change per each game. Convergence to zero indicates training is complete.

Next, the discount rate was kept relatively low at 0.6 to increase emphasis on making a good move given the current roll. This reduces the effect of the randomness of the dice as the actual next possible move from the new state isn't know until a new dice roll has occurred. This was discovered through observing trained Q values where the "normal" move was consistently receiving the highest value. This is due to most of desired moves being tile based that are often not possible to be selected one after another without an intermediate move. This change was especially import for the "overshoot" action as the max Q will always be the goal move Q value, a high value by design, resulting in a greater $\Delta Q(s,a)$ and subsequently a larger Q value. Ensuring this value remained lower designated the move to only be selected as a last resort. The rewards are where the playing style is defined, for this player an aggressive style was adopted. The rewards for each action can be seen in table 1, where the "goal", "star" and "kill" actions received the largest reward. Using these values the player is encouraged to make move that have the greatest effect on progressing the game by moving many tiles using stars and getting pieces to the goal. The player is also encouraged to use the "kill" action to prevent the opponents from making progress. This style was chosen to reflect the designer's preferred play style. The other rewards place a relatively lower emphasis on moving a piece to safety through the "globe", "protect" and "goal zone" actions. The "normal" and "overshoot" moves have the lowest rewards as they are both considered undesired moves. Finally, the "open" action does receive a moderate reward, however, a design decision was made to always select the "open" action when a 6 has been rolled if any piece remain in the home. This decision is based on previous knowledge of the game and reduces the risk of have no active piece, causing the player to waste turns.

Table 1: Action rewards

Open	Normal	Goal	Star	Globe	Protect	Kill	Overshoot	Goal Zone
0.25	0.0001	0.9	0.5	0.4	0.2	0.5	0	0.4

2.3 Training and Testing

To facilitate the training the player was matched against three random opponents and completed between 50 to 1000 games. The large range of number of games where used to determine the convergence of the Q values as after a certain number of game it is expected the the average change Q value per game will become insignificant. Following some trials it was determined that less than 100 games were required for the values to converge. In terms of opponents, the player was matched against three during training to increase the chances of interactive states between pieces. In addition to the normal Q learning algorithm, at the end of each match played, the Q values for each state-action was averaged across the four pieces in the Q table. This extra step helps eliminates any added effects of more frequent use among the pieces, such as the observed slight bias to select the piece furthest along the board. Another added benefit of averaging across pieces is that no one piece is favoured following training, allowing all pieces to be equally as likely to be moved forward given any roll and a similar state. To test the player, it was played against one, two and three random movement opponents for 1000 games without updating the Q values further and the win rate was recorded. During that time a few games were recorded and inspected to verify actions were being selected as intended. Additional the player was tested against another Q learning semi smart player developed by Jan-Ruben Schmid. For this test, the player also played against one, two and three of Jan's players 1000 times for each number of opponents. To facilitate a fair comparison between the developed player and that of Jan-Ruben, his approach will be discussed in the next section.

2.4 Comparison of Methods

While also using a standard Q leaning approach, Jan-Ruben's player [2] uses a few key difference to develop a different behaviour. The first difference is in the parameters selected for Jan-Ruben's player, which can be seen in table 2.

Table 2: Jan-Ruben's action rewards

Open	Normal	Goal	Star	Globe	Protect	Kill	Die	Goal Zone
0.25	0.01	0.8	0.5	0.4	0.3	0.4	0	0.4

By comparison Jan has opted for a more conservative or all-round behaviour by selecting lower rewards for the "Goal" and "Kill" actions. Also, his reward for a "normal" move is significantly higher which likely reduces the emphasis on the player taking more dynamic actions. Another consideration is that the action "Die" was included, this occurs when a piece is moved to a tile with two opponent pieces causing that piece to be sent home. This is another undesirable action and thus has been given zero reward. In addition to different actions, Jan-Ruben implemented a different training technique by using the ϵ-greedy method with a 10% random selection chance. By including this techniques he allowed for exploration by selecting a random move instead of always selecting the max Q value available. Lastly, Jan-Ruben's player does not stop updating it's Q-table so even during the evaluation against the player described in this paper, the values are being changed following every move.

3 Results

The following table (3) measures the effect of discount rate on the win rate over a period of 100 games. All players were trained under the conditions specified earlier with the exception of the varying discount rate. This test was carried out against three Random opponents.

Table 3: Overall win rates over 100 games with varying discount rates.

Discount Rate	0.5	0.6	0.7	0.8	0.9
Win Rate %	82	89	75	30	10

To measure the effectiveness of hindering the selection of the overshoot action, the amount the action was chosen per game was recorded. For comparison the player was played against the Random players and Jan-Ruben's player over 100 games, figures 2 and 3 display those results.

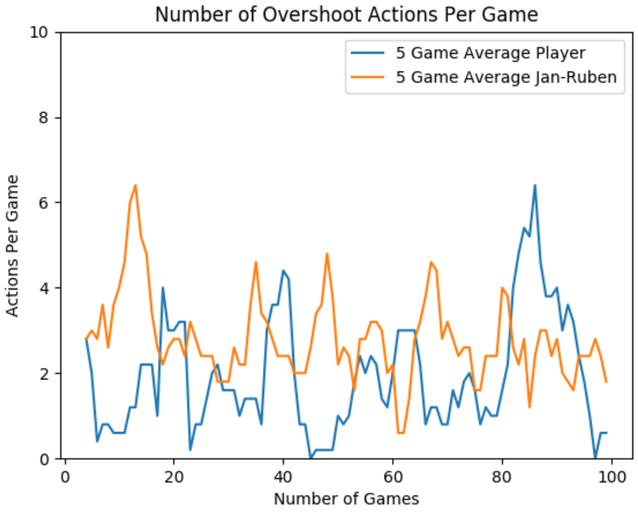

Fig. 2: Jan-Ruben's player's average number of overshoot or wasted moves per game against over 100 games.

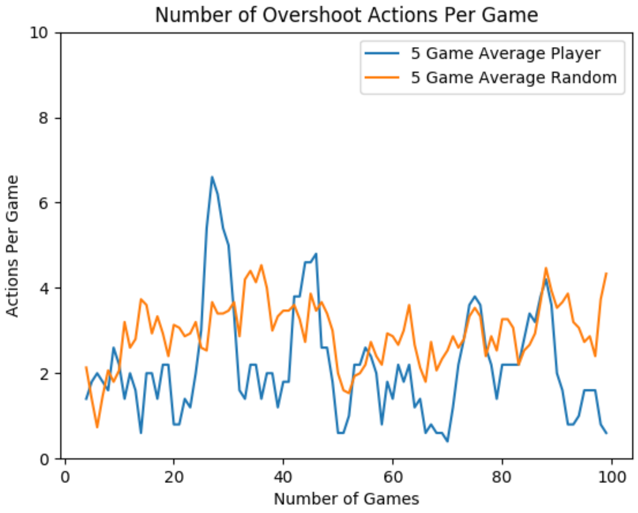

Fig. 3: A Random player's average number of overshoot or wasted moves per game against over 100 games.

The tables and figures below outline the success rates of the player against various opponents. The testing below was done in effort to display the capabilities of the developed player. Figures 4 and 5 display the running 5 game average success rate over the course of 1000 against one to three opponents for both the Random player and Jan-Ruben's player. The tables 4 and 5 contains the final success rate against the various opponents for both players. In addition sample Q-tables for both the developed player and Jan-Ruben's player are included in the appendix.

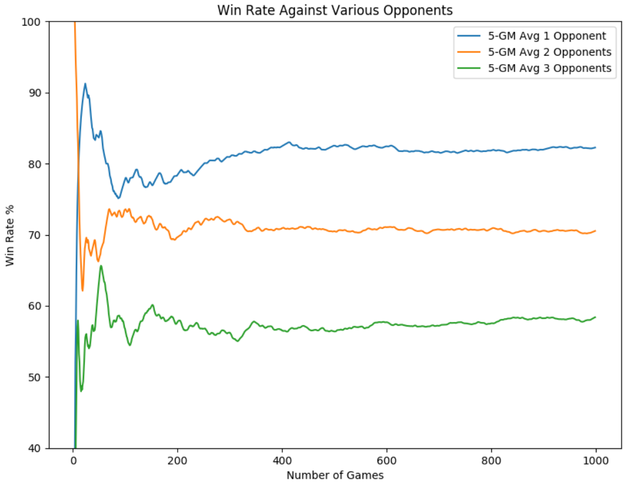

Fig. 4: Average success rate against one, two and three Random players over 1000 games.

Table 4: Overall win rates various situations against Random players.

Opponent(s)	1 Random	2 Random	3 Random
Win Rate %	83	70	59

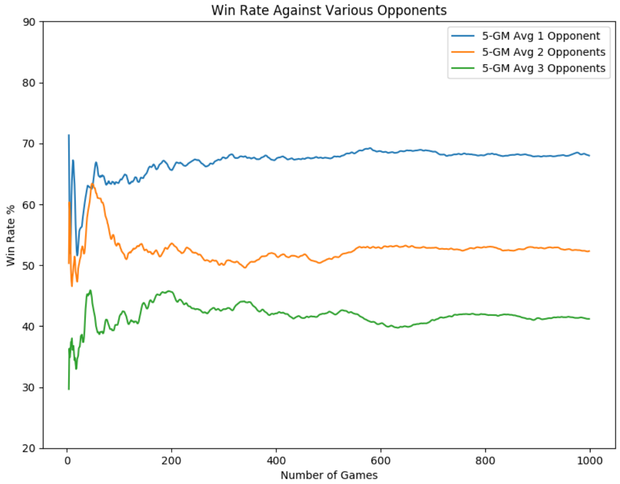

Fig. 5: Average success rate against one, two and three of Jan-Ruben's players over 1000 games.

Table 5: Overall win rates in various situations against Jan-Ruben's player.

Opponent(s)	1 Player	2 Players	3 Players
Win Rate %	68	52	41

4 Analysis and Discussion

The first test displays the effect of the discount rate on the win rate. It can be seen that a lower discount rate significantly improves the players performance. However this margin seems to go to zero if not negative when the value is lower than 0.6. The value could be explained by the average desired reward value, that being the average reward excluding the ”normal” and ”overshoot” actions. With that value being 0.45, the reward (in most cases) is the larger factor when updating the Q value ensuring more emphasis on the current action selection as opposed to future unknown actions.

The second test compares the amount of ”overshoot” moves made by the developed player, Jan-Ruben's player and a Random player. The results show that over the course of 100 games, the other two methods select the overshoot action on average 3 times per game while the player only selected it 2 times per game. This is a direct indication that the player has developed the desired outcome of limiting this selection. Further testing could prove whether or not the inclusion of this action has a significant effect on the win rate. Though through observations of games it seems that it is a plausible hypothesis.

The last set of tests shows that the player wins significantly more games than either the Random players or Jan-Ruben's player. When looking at the results when playing against Random opponents it can be seen that the player wins a majority of the games in all three situations, with a maximum of 83% against a single player. This win percentage is extremely high and likely can not be much higher due to the random nature of the game. Turning to the results against Jan-Ruben's player the win rates are lower than that of the Random player but the developed player is still superior. This indicates that Jan-Ruben's player is more challenging than the random player but suggests that some of the design decision proved more effective for the developed player.

Given both tests it can be concluded that the method outline in this paper produced a higher performing player than that of Jan-Ruben. The emphasis on aggressive playing style and the reduction in wasted moves are likely key factors that lead to this outcome.

5 Conclusion

This paper has detailed an approach for creating an intelligent Ludo player using the Q-learning method. The player was designed with an aggressive playing style through high rewards for high progress moves and killing the opponent and low to no rewards for wasteful moves. The agent was trained using unsupervised learning where it played against three random selection opponents. The player was then evaluated on its ability to learn to avoid wasteful moves, the effect the discount rate has on win rate and finally its performance against various intelligent and random opponents. It was found that player learned to limit wasteful moves, which lead to a 83% and 68% win rate against a random selection player and another Q-learning intelligent player respectively. The effect of the discount rate was unexpected and the realization of the emphasis it placed on current or future move selection was interesting.

Future work could be done on the optimization of the reward for each action, where genetic algorithms could be implemented as a means of selection. Also more observation of the game could be done to find more actions or states that result in wasted or unproductive turns such as tiles with high chances of being sent home. Finally, training against a more intelligent player could also prove to be beneficial in further increase the win rate.

6 Acknowledgements

The author would like thank Jan-Ruben Schmid for the use of his Q-learning player for test and comparison. Thanks is also given to Poramate Manoonpong, Associate professor at SDU, for the project proposal and his brilliant and enthusiastic lectures throughout the semester. Finally the author would like to

thank Simon Lyck Bjært Sørensen for his Ludo game python environment, saving many hours of development and allowing for the focus to be on the artificial intelligence aspect of the project.

References

[1] Poramate Manoonpong. *Lecture 6 AI2 Tools of Artificial Intelligence.* URL: `https://khkgears2.net/catalog3/SR2.5-100`.

[2] Jan Ruben Schmid. *LudoPy.* URL: `https://github.com/janschmid/LudoPy`.

7 Appendix

Table 6: Resulting Q table.

State\Action	Open	Norm	Goal	Star	Globe	Protect	Kill	O/S	G/Z	Null
Pc. 1 Home	0.9221	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Pc. 1 G/Z	0.0000	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000	0.5787	0.9620	0.0000
Pc. 1 Safe	0.0000	0.6811	0.0000	1.1492	1.0223	0.6369	0.9878	0.0000	0.9775	0.0000
Pc. 1 Danger	0.0000	0.6851	0.0000	1.1209	1.0711	0.3694	1.1327	0.0000	0.7625	0.0000
Pc. 2 Home	0.9221	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Pc. 2 G/Z	0.0000	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000	0.5787	0.9620	0.0000
Pc. 2 Safe	0.0000	0.6811	0.0000	1.1492	1.0223	0.6369	0.9878	0.0000	0.9775	0.0000
Pc. 2 Danger	0.0000	0.6851	0.0000	1.1209	1.0711	0.3694	1.1327	0.0000	0.7625	0.0000
Pc. 3 Home	0.9221	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Pc. 3 G/Z	0.0000	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000	0.5787	0.9620	0.0000
Pc. 3 Safe	0.0000	0.6811	0.0000	1.1492	1.0223	0.6369	0.9878	0.0000	0.9775	0.0000
Pc. 3 Danger	0.0000	0.6851	0.0000	1.1209	1.0711	0.3694	1.1327	0.0000	0.7625	0.0000
Pc. 4 Home	0.9221	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Pc. 4 G/Z	0.0000	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000	0.5787	0.9620	0.0000
Pc. 4 Safe	0.0000	0.6811	0.0000	1.1492	1.0223	0.6369	0.9878	0.0000	0.9775	0.0000
Pc. 4 Danger	0.0000	0.6851	0.0000	1.1209	1.0711	0.3694	1.1327	0.0000	0.7625	0.0000

Table 7: Resulting Q table from Jan-Ruben's player.

State\Action	Open	Norm	Goal	Star	Globe	Protect	Kill	Die	G/Z	Null
Home	6.3020	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Safe	0.0000	6.0870	6.7590	6.5580	6.4830	6.3480	6.4390	5.9210	6.4490	6.0350
Unsafe	0.0000	6.0670	6.7620	6.5560	6.4810	6.3340	6.3700	5.8740	6.4360	6.0360
Danger	0.0000	6.0840	6.7720	6.5620	6.4510	6.2230	6.4680	5.9470	6.4460	0.0000