Solving LUDO using Q-learning with simple and complex state-space representations

Martin Androvich
marta16@student.sdu.dk

University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark

Abstract. In this paper, a Q-learning agent learns to play a game of LUDO against random players, comparing the use of localized, simple and complex state-space representations. A simple state space representation encodes only the basic position details of a token, whereas a complex representation attempts to encode human-like judgment, such as whether a state is safe. Against random players with a win-loss ratio of $p = 0.25 \pm 0.014$, the simple agents achieves a win-loss ratio of $p = 0.599 \pm 0.017$ and the complex agents achieves a win-loss ratio of $p = 0.622 \pm 0.015$. Comparing the two agents against each other, the complex agents comes ahead with a win-loss ratio of $p = 0.553 \pm 0.019$ against its counterpart.

1 Introduction

The LUDO board game, as seen in Fig. 1, is a stochastic environment in which two to four players compete against each other. Solving the problem using symbolic artificial intelligence techniques, such as uniform cost search, is impractical due to the stochastic nature of the game and the size of its state space; the games has a high branching factor with an estimated state-space of up to 10^{22} states [1].

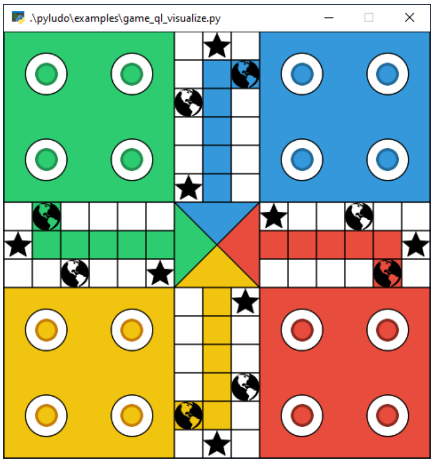

Fig. 1. Screenshot of the `pyludo` Python LUDO simulation software.

This paper evaluates the usage of vanilla Q-learning in Python with the purpose of learning an optimal policy for playing the LUDO board game; two state-space representations, a simple and complex, are evaluated, in which Q-learning agents compete against random players and each other.

The Python LUDO simulation software used in this paper is a fork of [2] and is available at `https://github.com/martinandrovich/pyludo`, which contains the game rules, experiment data and any other code relevant to this paper.

2 Methods

2.1 Q-learning

Q-learning is a value-based, off-policy, reinforcement learning method that uses a temporal difference update rule in order to iteratively estimate the quality values for state-action pairs, known as Q-values [3]. These values are stored in a Q-table, which acts as a lookup table for a given state-action pair, allowing to infer the most optimal action for a given state.

In order to apply Q-learning, it is necessary to design an applicable state-space representation, including the definition of states, possible actions, and rewards for transitions between states. The choices in state-space representation impact the outcome of the Q-learning training. Furthermore, several techniques can be used to optimize the learning process (e.g., greedy epsilon decay) and selecting the correct hyper-parameters also impacts the training success.

2.2 State-space representation

Human players have a global state-space representation when playing a game of LUDO, meaning they have the ability to account for the actions of opponents, such as hunting down an opponent that is close to its goal state. Implementing such capabilities for an AI would require a global representation of the state of the game, e.g., an image of the configuration of the current board. Due to the immense number of possible state permutations, a Q-table is infeasible for such a state-space representation. Instead, this paper focuses on evaluating somewhat local state-space representations, namely a simplified and complex local state-space representation.

A local state-space representation encodes the state of individual player tokens, where both the simple and complex representations have the same pool of actions, as shown in the Q-table in Fig. 2. The simple representation has a total of 5 states; whether the token is home, on the common path, on a globe, on the victory road or in the goal. The complex space representation has a total number of 14 states, and includes information on whether the token is with a buddy and permutations of whether it is in danger, has the opportunity to kill an opponent token or both. The complex state-space representation attempts to leverage information about the surrounding opponents and dynamically reward actions that move the token into a safer state.

	MOVE_FROM_HOME	MOVE_FROM_HOME_AND_KILL	MOVE	MOVE_ONTO_STAR	MOVE_ONTO_STAR_AND_DIE	MOVE_ONTO_STAR_AND_KILL	MOVE_ONTO_GLOBE	MOVE_ONTO_GLOBE_AND_DIE	MOVE_ONTO_ANOTHER_DIE	MOVE_ONTO_ANOTHER_KILL	MOVE_ONTO_VICTORY_ROAD	MOVE_ONTO_GOAL	NONE
HOME	0	0	0	0	0	0	0	0	0	0	0	0	0
HOME_CAN_KILL	0	0	0	0	0	0	0	0	0	0	0	0	0
GLOBE	0	0	0	0	0	0	0	0	0	0	0	0	0
GLOBE_CAN_KILL	0	0	0	0	0	0	0	0	0	0	0	0	0
GLOBE_IN_DANGER	0	0	0	0	0	0	0	0	0	0	0	0	0
GLOBE_IN_DANGER_CAN_KILL	0	0	0	0	0	0	0	0	0	0	0	0	0
COMMON_PATH	0	0	0	0	0	0	0	0	0	0	0	0	0
COMMON_PATH_CAN_KILL	0	0	0	0	0	0	0	0	0	0	0	0	0
COMMON_PATH_IN_DANGER	0	0	0	0	0	0	0	0	0	0	0	0	0
COMMON_PATH_IN_DANGER_CAN_KILL	0	0	0	0	0	0	0	0	0	0	0	0	0
COMMON_PATH_WITH_BUDDY	0	0	0	0	0	0	0	0	0	0	0	0	0
COMMON_PATH_WITH_BUDDY_CAN_KILL	0	0	0	0	0	0	0	0	0	0	0	0	0
VICTORY_ROAD	0	0	0	0	0	0	0	0	0	0	0	0	0
GOAL	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2. Q-table representing the state-space for the LUDO game.

The rewards used for both of the state-space representations are similar and are shown in Table 1. The agent is rewarded/punished for moving from home, moving, dying, killing, getting onto the victory road, getting onto the goal, and winning the game. The rewards for moving and killing are proportional to the number of steps taken n, whereas the reward for moving from home is amplified if a kill occurs. The complex state-space representation furthermore rewards/punishes the agent with a value b based on whether it moves into a more or less safe state, which is computed using token vulnerability: an integer that estimates the number of moves required to send a token home.

Table 1. Rewards used in the simple and complex state-space representations for Q-learning. Here, n denotes the number of steps taken by a token, and b is a bonus value computed for the complex representation based on change in token vulnerability.

Representation	**Reward**	**Value**
Simple	Move	$\frac{n}{13} \cdot 5 + b$
	Move from home	$5 + 5\,if\,kill$
	Die	-50
	Kill	$5 + \frac{n}{13} \cdot 5 + b$
	Get onto victory road	$50 + b$
	Get in goal	$100 + b$
	Win	1000
Complex (b)	Move to safety	25
	Move to danger	-25

2.3 Learning process

Initial learning is achieved by initializing an empty Q-table, which is shared (and simultaneously updated) by two Q-learning agents that play against two random players for N episodes. Several values are logged during training: (a) the cumulative reward, (b) an incremental moving average with window size $k = 100$ of the win-loss ratio, (c) the sum of differences ΔQ between the previous and current Q-table, and (d) current epsilon value.

The number of episodes for training is chosen after evaluating a preliminary session with a conservative learning rate of $\alpha = 0.001$ and discount factor $\gamma = 0.9$, in which ΔQ converges at around $N = 20\,000$ episodes. The values of the learning rate and discount factor are not further varied, since these provide sufficient convergence.

Balancing between exploration and exploitation during training is achieved using a decayed-ε-greedy behavior policy [4], where epsilon decays in the range $\varepsilon \in [0.7; 0.01]$. This is more advantageous than the regular ε-greedy approach, since it favors exploration in the early stage of the training, but gradually decreases the exploration as the state-values become better approximated and exploitation is more favorable.

To further balance exploration, the argmax used in the Q-learning update rule has been replaced with a random argmax, such that is selects a uniformly random value when presented with multiple choices, instead of picking the first value.

2.4 Experiment design

Players are evaluated based on their average win-loss ratio, which is computed over 1000 episodes (games), repeated 100 times, with a total of 100 000 episodes per evaluation. The win-loss ratio of each state-space representation is computed by playing against: (a) three random players, and (b) three expert players. To determine the superior state-space representation, the simple and complex representations are compared to each other, playing a game with two players of each representation. Statistical analysis is then used to determine whether the complex representation provides any statistically significant result.

3 Results

In Fig. 3 and Fig. 4 the training data is shown for the Q-learning agents with the simple and complex state-space representations, respectively, obtained for each by playing random players over $N = 20\,000$ episodes. The data has been filtered with a Gaussian smoothing of window size $k = 100$.

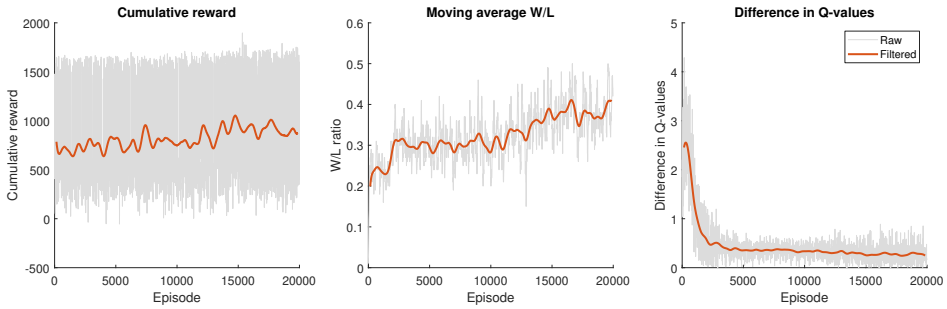

Fig. 3. Cumulative reward, moving average of win-loss ratio, and the difference in Q-values during training for the simple state-space representation.

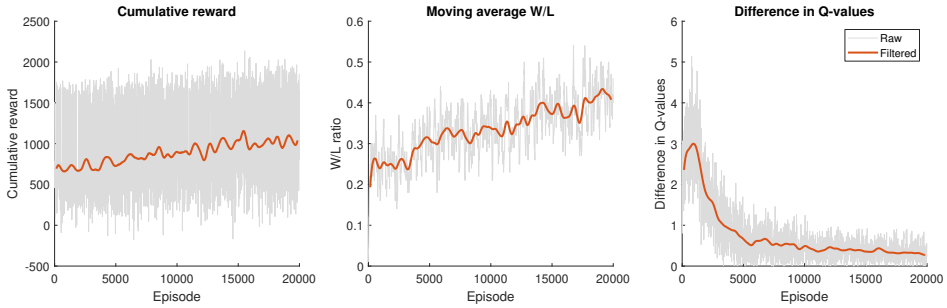

Fig. 4. Cumulative reward, moving average of win-loss ratio, and the difference in Q-values during training for the complex state-space representation.

A heat map of the resulting Q-tables for the simple and complex state-space representation is shown in Fig. 5 and Fig. 6, respectively. It should be noted that both representations use Q-tables of identical dimension, however, the simple representation is only using the cells valid for its assigned states.

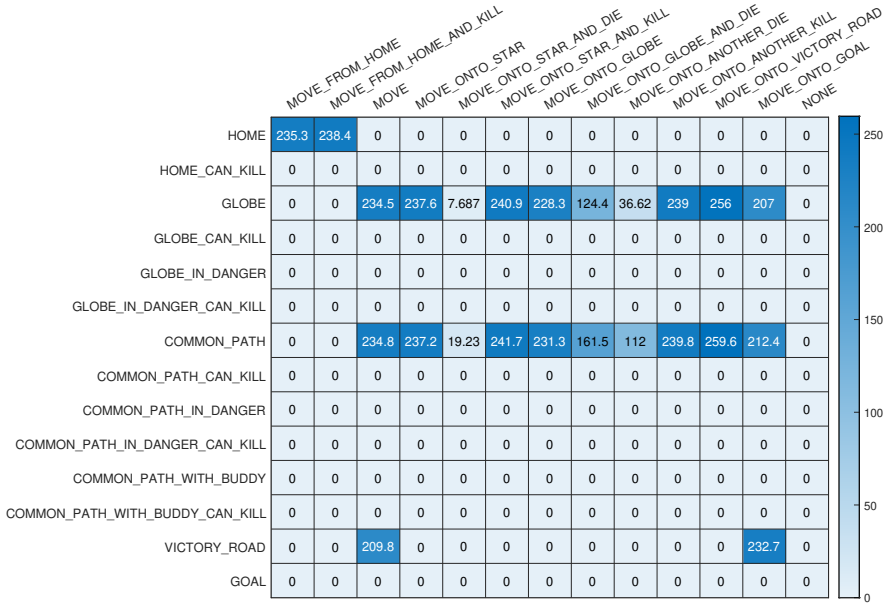

Fig. 5. Q-table for simple state-space representation after training for 20 000 episodes.

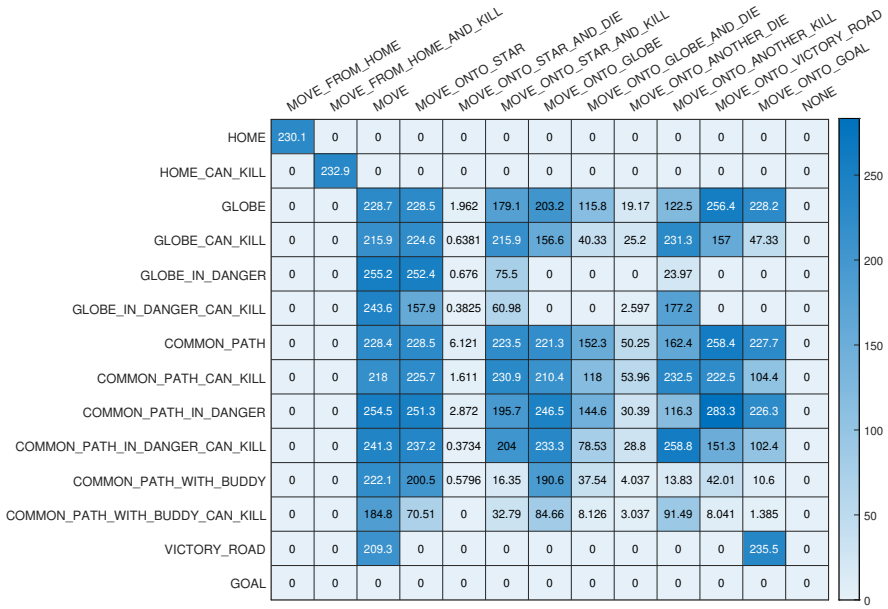

Fig. 6. Q-table for complex state-space representation after training for 20 000 episodes.

For evaluation of the players, the baseline win-loss ratio is established for the random players. After a total of 100 000 episodes, the probability of a random player winning is normally distributed as $p = 0.250 \pm 0.0136$ when playing against other random players, as shown in Fig. 7. The trained Q-learning players are first evaluated against the expert players, with an average win-loss ratio of $p = 0.512 \pm 0.0149$ and $p = 0.555 \pm 0.0148$ for the simple and complex state-space representations, respectively.

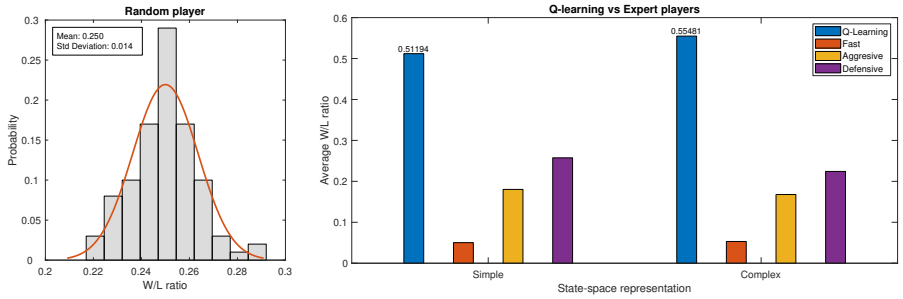

Fig. 7. From left: win-loss ratio of a random player (versus other random players), and win-loss ratios of the Q-learning agents versus expert players.

Continued evaluation of the trained Q-learning players is shown in Fig. 8. Comparing the players of the simple and complex state-space representations with the random players boasts with a win-loss ratio of $p = 0.599 \pm 0.017$ and $p = 0.622 \pm 0.015$, respectively. Comparing the two Q-learning players against each other shows a win-loss ratio of $p = 0.447 \pm 0.019$ for the simple state-space representation and $p = 0.553 \pm 0.019$ for the complex state-space representation.

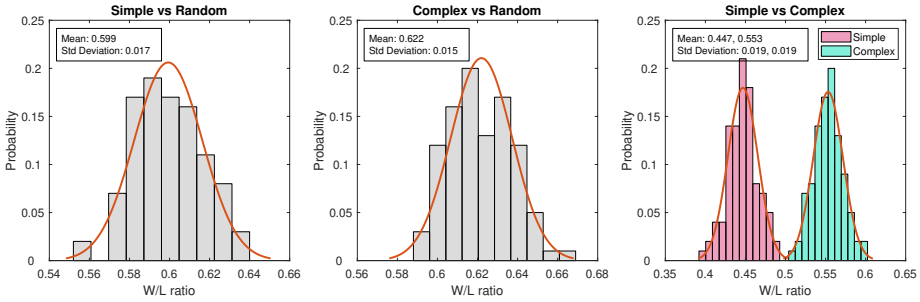

Fig. 8. From left: win-loss ratios of: simple Q-learning player versus random, complex Q-learning player versus random, and the two Q-learning players versus each other.

4 Analysis and Discussion

Looking at Fig. 3 and Fig. 4 of the logged values during training, the increasing cumulative reward, increasing average win-loss ratio, as well as the decreasing convergence measure, are all a clear indication of that the agents of both state-representations are successfully learning and improving their performance during training.

When analyzing the resulting Q-tables, it is clear that both agents learn to favor moving larger distances, such as taking the star shortcut when on the common path, instead of simply moving. Both agents also favor a move in which they can send an opponent token home (kill). Another interesting observation is that both agents prefer to move tokens that are currently not on the victory road - which is a method that most humans also play by, since the token on victory rode is considered "safe".

In terms of the win-loss ratio, when compared to the ratio of a random player, the trained agents boast a 139.748 % improvement for the simple state-space representation, and a 148.788 % improvement for the complex state-space representation. These results indicate that the complex representation is more favorable than the simple representation, having an increased 3.77 % win-ratio when compared to its counterpart. Furthermore, the complex representation has a 8.374 % higher win-loss ratio versus the expert players, when compared to the agent of the simple representation.

Letting the two representations compete against each other, the complex representation comes ahead with a win-loss ratio of $p = 0.553 \pm 0.019$. Using a two-sample t-test, the null hypothesis that the mean win-loss ratio of the simple representation is equal to the mean win-loss ratio of the complex representation, is rejected at a 95 % confidence interval with $p = 1.7822 \times 10^{-97}$, meaning that the complex representation has a statistically significant improvement of the win-loss ratio, when compared to the simple representation. This is likely due to the complex agent being able to leverage more information and make moves that prioritize keeping token safe, as well as move tokens out of danger. This notion is supported by the Q-table, seeing that, for example, when choosing between moving a token on the common path, or moving a token that is in danger on the common path, the agent prefers to move the token in danger.

5 Conclusion

For the LUDO board game, Q-learning with both simple and complex state-space representations can be used to obtain a better win-loss ratio than that of random players or simple expert players. A complex representation is more favorable due to its statistically significant increase of the win-loss ratio; it achieves a win-loss ratio of $\sim$62 % against random players, being $\sim$4 % more than the simple representation. When playing the representation against each other, the complex representation wins $\sim$55 % of the time.

6 Future work

The Q-learning algorithm used in this paper is a somewhat vanilla approach and can be improved in numerous way. For instance, hyper-parameter tuning can be utilized to improve training efficiency and potentially the performance (win-loss ratio) of the Q-learning agents.

LUDO as a game lends itself to various artificial intelligence solutions. A limitation of the current Q-learning method is the local state-space representation, not utilizing the complete information of the other players. It would be interesting to see how a Deep Q-learning network with global state-space representation, such as an image of the current board configuration, would fare against the local state-space representations of the Q-learning players in this paper.

7 Acknowledgments

I would like to thank Poramate Manoonpong for his lectures related to this course. I would also like to thank Rasmus Haugaard and Haukur Kristinsson for their work on the Python LUDO game.

References

[1] Alvi, Faisal and Ahmed, Moataz. "Complexity analysis and playing strategies for Ludo and its variant race games". In: *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. 2011, pp. 134–141. DOI: 10.1109/CIG.2011.6031999.

[2] Haugaard, Rasmus and Kristinsson, Haukur. *pyludo*. URL: https://github.com/RasmusHaugaard/pyludo.

[3] Watkins, Christopher JCH and Dayan, Peter. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[4] Koppula, Rajendra. *Exploration vs. Exploitation in Reinforcement Learning*. URL: https://www.manifold.ai/exploration-vs-exploitation-in-reinforcement-learning.