Ludo Game, Q-learning player

Gian Paolo Currà

University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
gicur22@student.sdu.dk

Abstract. The primary objective of this study is to develop a Q-learning AI system for Ludo that can explore the environment, learn from rewarding and optimize gameplay strategies. By defining the state space, action space, rewards, and transition dynamics, the Q-learning agent learns how to make decisions to maximize the expected cumulative rewards.

1 Introduction

Q-learning is a type of Reinforcement Learning to develop an AI agent in a known environment (known states) with a set of known action. Reinforcement learning is based on the relation between the environment and the agent, rewarding it to indicate the quality of the taken action in a particular state. The task of the agent is to maximize its cumulative reward (the weighted sum of expected future rewards) by selecting the most beneficial actions across all states. By this way, it would be possible reaching its goal with the most optimal set of actions. Ludo, a well-known board game played by 2-4 players, serves as the testing ground for the Q-learning AI agent in this study. The objective of the game is for players to race their pieces around the board, moving them from start and bring all of them safely to the goal area [1]. However, one of the key characteristics of Ludo is its probabilistic nature, as the game's outcome depends on the decisions made by all players as well as the results of the dice roll. Additionally, Ludo presents an high complexity level. According to Alhajry, Alvi and Ahmed [1], the number of possible board configurations is more than 10^{22}. In the end, the results of this paper are compared with the Q-learning solution from a fellow student. The wanted result from this comparison is to see if the application of the Q-learning in a Ludo game could be an optimal performance choice, exploring three different strategies and test their effectivness.

2 Methods

In Q-learning, the agent can learn through experience and memorize the actions saved into the Q-table. It selects its actions based on new choices (exploration) and past experience (exploitation) [2], the ratio between these two type of selection is determined by ϵ, a parameter that goes from 1 to 0 and describe the quantity of randomness that our agent can have every time it takes an action.

$$Q\left(S_t, A_t\right) \leftarrow \underbrace{Q\left(s_t, a_t\right)}_{oldvalue} + \alpha \left[r_t + \gamma \max_a Q\left(S_{t+1}, a\right) - \underbrace{Q\left(s_t, a_t\right)}_{oldvalue} \right] \tag{1}$$

where
α = learning rate
r_t = reward
γ = discount factor
$\max_a Q\left(s_{t+1}, a\right)$ = estimate of optimal future value
$Q\left(s_t, a_t\right)$ = old value

The approach chosen in this paper is to display the differences between 3 strategies, extended from 1 to 3 opponents. In the program, ϵ is chosen to be a decaying variable, to simulate an increasing exploitation aspect during the training. After epsilon reaches the 0 value, the agent will count only on the experience and its memory, without explore the environment anymore.
The performance is being measured by the win-rate obtained by the agent during the gameplays. In the following sections it will be possible to see how this value changes depending on the strategy chosen and the parameters that are part of equation 2.

Strategies:

- Normal: No particularly rewarded actions. This mode serves as a baseline for comparison and provides insights into how the agent performs when the reward system has minimal impact;
- Go-to the Stars: the agent is highly rewarded if he lands on stars and utilizes the shortcuts;
- Aggressive: highly rewarded for landing on an opponent during the match. This mode aims to encourage the AI agent to make strategic decisions that maximize the chances of killing opponents' pieces.

α	γ	ϵ	$\Delta\epsilon$
0.2	0.5	0.9	0.09

Table 1. These parameters are fixed for the win-ratecomparison between strategies.

$\Delta\epsilon$ is the difference with ϵ value changes during the exploration phase.

2.1 Q-Table

States:

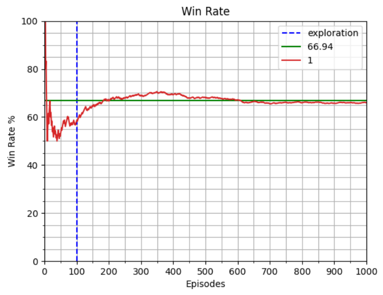
– Safe
– Unsafe
– Home

Actions:

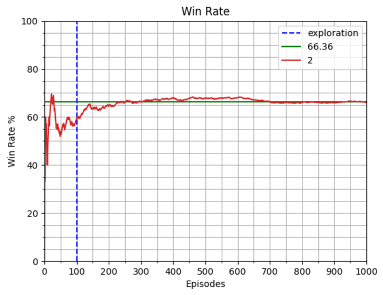
– MoveOut
– GoalZone
– MoveDice
– Goal
– Star
– Globe
– Protect
– Kill
– Die

2.2 Win-rate Analysis

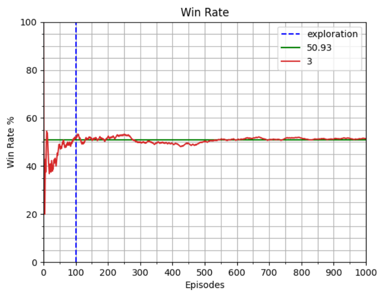
The subsequent graphs are used to emphasise the difference between the strategies and the number of opponents in win-rate and rewards terms. To compare the win rates of the AI agent across the three modes, simulations with increasing opponents (from 1 to 3) are conducted. Each test comprises 1000 plays where during the first 100 plays the epsilon is progressively decreased, from the 101st play, the agent no longer explores the environment but uses the information obtained from the first 100 plays with non-zero epsilon to fully exploit. Each strategy differs from the others because of the modified reward table. Each reward table is in fact characterised by an increased amount of reward depending on the strategy: Normal (no increased reward), Stars (stars choice reward increased) and Aggressive (kill choice reward increased). In this experiment, only 1 player is an agent and has a strategy, the opponents are moved randomly.



(a) Normal strategy vs 1 opponent



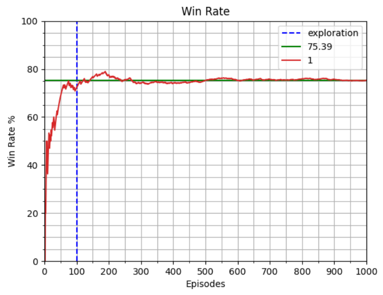
(b) Normal strategy vs 2 opponents



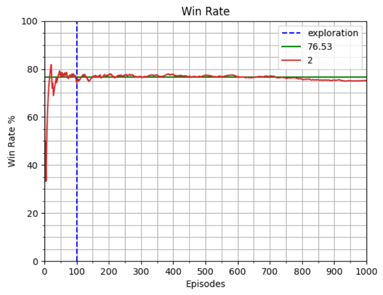
(c) Normal strategy vs 3 opponents

Out_base	Dice_mv	Goal_pos	Star_pos	Globe_pos	Protect_mv	Kill_mv	Die_mv	Goal_area
0.4	0.1	1.0	0.4	0.2	0.2	0.2	-1.0	0.8

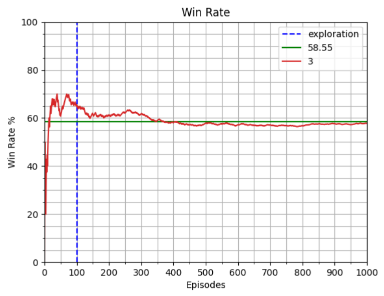
Normal strategy does not value a particular strategy (like ”landing on an enemy” or ”land on a safe space”) beside the necessary one to give the winning purpose to the agent and push it toward the target state (also called absorption goal[3]). The function of this mode is to be the starting point of how a reward change can influence the gameplay.



(d) Stars strategy vs 1 opponent



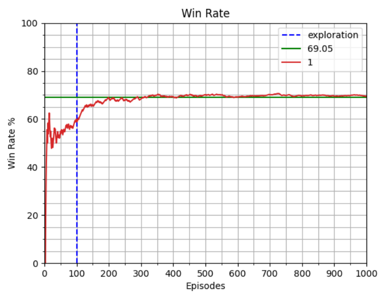
(e) Stars strategy vs 2 opponents



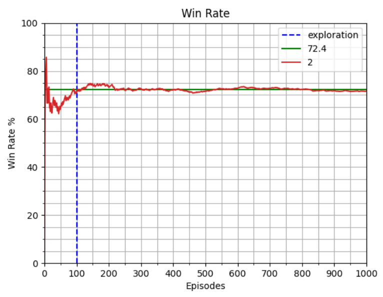
(f) Stars strategy vs 3 opponents

Out_base	Dice_mv	Goal_pos	Star_pos	Globe_pos	Protect_mv	Kill_mv	Die_mv	Goal_area
0.4	0.01	0.8	1.5	0.2	0.2	0.2	-1.0	0.8

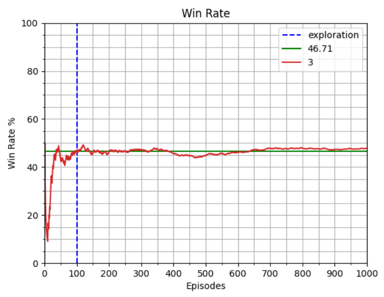
Go-to Stars strategy push the agent towards a shortcut taking approach. The counter effect of this method is its dependence from the randomness of the dice. The stars on the board are only 8 out of 28 positions.



(g) Aggressive strategy vs 1 opponent



(h) Aggressive strategy vs 2 opponents



(i) Aggressive strategy vs 3 opponents

Out_base	Dice_mv	Goal_pos	Star_pos	Globe_pos	Protect_mv	Kill_mv	Die_mv	Goal_area
0.4	0.01	0.8	0.4	0.2	0.2	1.5	-1.0	0.8

Aggressive strategy emphasise the landing of the agent's piece onto the same cell of the opponent piece, by this way the opponent's piece is forced to reset its position, returning to the home area and beginning the journey again. This strategy does not focus on the personal goal but focuses on obstructing the opponents, gaining the time needed to be able to make a full path and reach the goal.

3 Analysis and Discussion

3.1 Playstyle analysis

As it can be seen in the previous graphs, every strategy have advantages and disadvantages.

The first approach, the Normal Strategy, is the one in which it is possible to see that the average win-rate does not vary sharply as the number of players increases. Moreover, it remains mostly close to 66% until the case with the two opponents. Only after adding the third opponent there is a drop in the graph of 16%.

The second approach, the Go-To-Stars Strategy, is one that favors the speed on the completion of the game, with which the agent's tokens are brought to the end zone. This is because each star can increase the amount of cells traveled. For this reason, the win-rate value against 1 opponent is higher than in Normal Strategy and even Aggressive Strategy. This happens because the randomness of one opponent's tokens cannot compete with the high reward on taking the star. By adding more opponents and therefore more tokens, the situation changes. Now the population of pieces on the board is higher than before and the stars, even if hit randomly, are easier to end up on for the opponents and harder for the agent. Even in this approach, when 4 players are present, the drop in win-rate is almost 20%.

The last one on schedule, the Aggressive Strategy. It does not have a win-rate as high as the previous one, but still higher than the Normal Strategy. the hypothesis of the first strategy as a baseline for the next two is therefore confirmed. In this case, a win-rate drop of 30% between the most succesfull case and the worst is recorded.

3.2 Comparison with fellow student

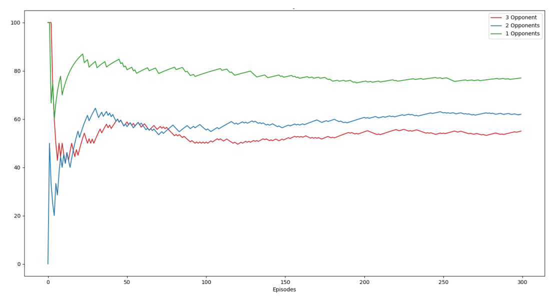


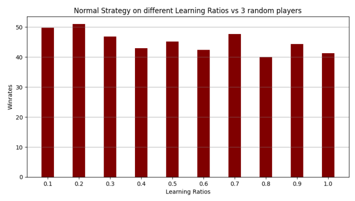
Fig. 1. win-rate for 1,2 and 3 opponents from fellow student Q-Learning, Aggressive playstyle

win-rates:

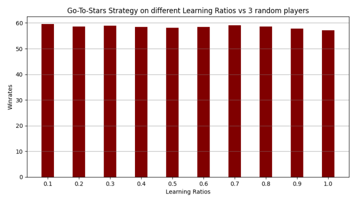
- 3 Opponents: 54.48
- 2 Opponents: 62.82
- 1 Opponent: 77.55

The win-rates are against random agents. He used Q-learning with rewards trying to promote an aggressive playstyle.

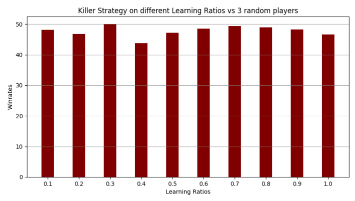
As it can be seen in both cases of Aggressive playstyle it is possible noticing that an approach that focus only on a single aspect of the entire game, like aggressive or star racing, is not effective when the opponent quantity rise.



(a) Normal Mode



(b) Go-To-Stars Mode



(c) Aggressive Mode

Fig. 2. Comparison average win-rates after the first 100 exploration episodes while increasing the learning ratio

3.3 Learning ratio Analysis

This section explains how the learning ratio influences the agent's win-rate against 3 opponents. The average win rate (after the first 100 environment exploration plays) is taken and compared to the same approach while increasing the learning ratio by a step of 0.1. The learning ratio (or α-value) measure how fast the Q-values can change and how much the Q value is influenced by past actions. If the alpha is near zero, the agent would not be able to learn anything from new actions. On the other side, if alpha-value is set to 1, the agent can ignore prior knowledge and only make evaluations on the most recent information. It can be noticed that the random aspect of the Ludo Game has a massive influence on the data. This is the reason why the win-rate percentage does not change much by increasing the learning rate (considering that the mean win-rate is calculated after the exploration phase).

4 Conclusions

Overall, the three approaches chosen drew a lot of results regarding Q-learning, this is because, out of three, two took different styles of play to the extreme and one served as a baseline.

Through the comparison with Anders' agent, it was concluded that the random component of dice in the game is very present and heavily influences the win rate. This means that it is not possible to concentrate the rewards in only one direction as was done for aggressive strategy and go-to-stars strategy but to distribute them more evenly in order to achieve a smaller win-rate decrease as the number of players increases

5 Acknowledgements

I would like to express my sincere gratitude to Professor Poramate Manoonpong for the guidance and support, and expertise throughout this semester and during the research project. I would like to thank also Anders Johannsen for his Q-learning method that made possible a comparison with my approach.

References

1. Alhajry, Majed and Alvi, Faisal: TD(λ) and Q-learning based Ludo players 2012 IEEE Conference on Computational Intelligence and Games,2012
2. Poramate Manoonpong: Tools of Artificial Intelligence lecture slides 2023 SDU
3. Watkins, C.J.C.H., Dayan, P. Q-learning. Mach Learn 8, 279–292 (1992)