Tools of Artificial Intelligence

Shubham Kumar (shkum20)|490532

University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
shkum20@student.sdu.dk

Abstract. This papers is focused on developing a method AI player for Ludo Game.The player are trained using the method of Q-learning. For this method the game is divided into number of states and actions.It's compared with other training the Deep reinforcement learning agent in general resembles the normal Q-learning approach but has some differences.

1 Introduction

About Game : The game of Ludo originates in India as far back as 3300 BC. Easy to learn, this classic game is fun for children and adults alike. Ludo's popularity has gone on to spawn variations such as Parcheesi and Sorry. With centuries of tradition behind Ludo, you will surely become a fan as well.Ludo consists of a square board with four different colored bases in each corner. The first colored space outside of each base is the start position. A path leads clockwise around the board returning to a path with the same color as the base, then to the home column, which leads to the center home triangle.There are four different colored sets of playing pieces that start off in their matching bases.

Each set consists of 4 playing pieces. The included die is used to govern the movement of the pieces.When a player's piece has reached the home column of its own color, the piece continues its moves toward the center to its home triangle. When a player's die roll lands its piece on the home triangle, that piece has completed its journey. A piece can only be moved to the home triangle with an exact roll.The first player to have all four of his pieces finish their journeys wins. The remaining players continue the game to determine the runner-ups.

In this report of Tools in Artificial Intelligence the method implemented to train the player using reinforcement learning and more specifically Q-learning. Reinforcement Learning (RL) can be defined as the study of taking optimal decisions utilizing experiences. It is mainly intended to solve a specific kind of problem where the decision making is successive and the goal or objective is long-term, this includes robotics or game playing. The approach used in this paper are from the course of Tools of Artificial Intelligence. Analysis and approach would be define below.Apart from that some resarch part was taken with help of my collague who used DRL based method for training the player by playing game individual and then computing results

The Ludo game used during this project is a Python version of, which have been developed by SimonLBSoerensenand can found on the following GitHub repository:https://github.com/SimonLBSoerensen/LUDOpy

2 Method

Brief description of the method implemented known as Q-learning. Q-Learning is a basic form of Reinforcement Learning which uses Q-values to iterative improve the behavior of the learning agent.Q-values are defined for states and actions. Q(S, A) is an estimation of how good is it to take the action A at the state S. The agent during its course of learning experience various different situations in the environment it is in called as states. The agent while being in that state may choose from a set of allowable actions which may fetch different rewards(or penalties). The learning agent overtime learns to maximize these rewards so as to behave optimally at any given state it is in.

$$Q(S, A) = r(S, A) + \gamma max_a Q(S', A) \tag{1}$$

The Agent in RL can be defined as the entity which acts as a learner and decision-maker. It is empowered to interact continually, select its own actions, and respond to those actions. It is the abstract world through which the agent moves. The Environment takes the current state and action of the agent as input and returns its next state and appropriate reward as the output.The specific place at which an agent is present is called a state. This can either be the current situation of the agent in the environment or any of the future situations. Actions defines the set of all possible moves an agent can make within an environment.This is nothing but the feedback by means of which the success or failure of an agent's action within a given state can be measured. The Rewards are used for the effective evaluation of an agent's action.Policy is mainly used to map the states along with their actions. The agent is said to use a strategy to determine the next best action based on its current state.

2.1 Game Representation

The method require game is divided into possible states and action. The possible states defined in games are as follow in below tables :

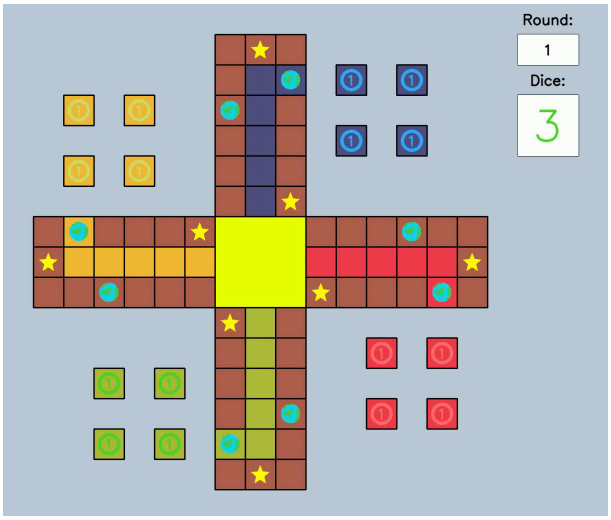

Fig. 1: Ludo Game Board Representation

States	Row in Q Table
Home	0
Goal Zone	1
Goal	2
Danger	3
Safe	4
Normal	5

Table 1: Different States represent in program of Q table row

For each available 6 states there are 11 action.These can be detected by the relative position of player.Therefore it can be determined what action would be performed by looking at the player current position.

The immediate reward is given based on the position of the piece that was played. This enforces the strategy that it is desired that the players move forward in the game, rather staying in the same position. When the goal state has been reached will a bonus reward is added additionally to the position based reward, to enforce the strategy that moving to the goal position is desired. A minor punishment is given if the player does not move forward, and chooses to not do any move, unless the player is at home position, and forced to stay in home.

The punishment consist of subtracting reward points from the computed reward. This punishment was partly added to learn the player that doing nothing is not the best choice, as the positional reward is higher by moving forward, but if no other possible actions is available, or if the movement of player lead it being killed and sent back home, will the punishment be even greater, and thus will the

Actions	Column in Q Table
Move Out	0
Normal	1
In Goal Zone	2
Use Star	3
Enter goal Zone	4
Goal Zone	5
Move to safe	6
Move away safe	7
kill enemy	8
Suicide	9
Nothing	10

Table 2: Different Actions represent in program of Q table Column

right choice sometimes be not to move the player which is farthest in the game. The minor punishment is intended to be a "friendly reminder" that staying in the same position is not good, but is allowed for a certain number4 of turns, in which the player should be in a state where moving will not lead to suicide. If a player is in the goal state, will it not be considered as movable, and ignored the rest of the game.

Event	Reward
Move Out	0.3
Kill enemy	0.2
In Goal Zone	0.05
Normal Zone	0.05
Enter Goal Zone	0.25
Use Star	0.15
Move to safe	0.1

Table 3: Reward for each Action

Event	Punishment
Suicide	-0.8
Move away from safe	-0.1
No action	-0.1
Returned Home	-0.25

Table 4: Punishment for each Action

2.2 Training the system

The system is trained using epsilon - greedy such that the rate of exploration and exploitation is controllable . The training is done in stages, with different ratios of exploration and exploitation. After each stage is a test performed to see how well the Q - table performs against the 3 random players, and based on the result will it be determined whether more exploitation or exploration is needed. While training will the discount factor be set to a low value, thus making it only consider immediate reward, and not the long term. This is was chosen as this system has many different valued rewards, and each of them provide the Q-table some form of knowledge where the player is in the game, and update it based on that. A higher discount factor would be ideal if the system had fewer rewards. The learning rate () will be changed according to result from the test. A higher learning rate would make the system learn based on new information, but might have a hard time converge to the ideal values, and vice versa would a lower learning rate make it converge slowly to the ideal values.

2.3 Second Method Comparison from colleague

The second method was compared with my fellow classmate Karol Szurkowski. It's really great to know about his findings and the results in the Ludo game. He followed and applied the DRL method for training of ludo agents.In method implemented in this paper, choice of the action should be understood as choice which piece of the current player's to move given the roll of the dice and knowing the state of the board. Knowing the current state st of the environment (board) agent chooses an action at, executes it and receives the immediate reward.

The goal of the agent is to choose the actions which gives the highest discounted cumulative reward-Training the DRL agent in general resembles the normal Q-learning approach but has some differences. When agent is about to select an action in an e-greedy[egreedy] manner, instead of calculating the current Q(s, a) the estimation of the Q^(s, a) is obtained from the MLP.

In this moment, update of the values of the state-space can not be observed. Update of the network happens after the decision is taken and when there is enough training data in memory. Because of the fact that initial estimations of the Q might be far from optimal due to random decisions of actions, it is beneficial for the system to forget some of the estimates, and forget them from the memory. That is why the training of the MLP begins, when the memory has enough samples and the training can be done in batches with fixed size of the stored experiences.

3 Results

The first training set consist of full exploration, with a high learning rate. These were chosen such that the player would get to know how the world is, and also how the Q-table should be. The Q-table is initialized with all its entries being

zero. First training sample data of one run consist of alpha = 0.7, gamma = 0.1 and epsilon = 1, for 1000. The results are shown in as table below. The player was then tested against an 3 random players for 1000 plays, from which the result can be seen

0.35687	0.	0.	0.	0.	0.	0.	0.	0.	0	0.02675
0.	0.	0.12394	0.	0.27276	0.	0.08209	-0.0564	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.10857	0.	0.23861	0.	0.22909	0.07062	0.	0.12331	-0.9534	0.
0.	0.09104	0.	0.27159	0.	0.17405	-0.1254	-0.0717	0.2559	-0.9859	0.
0.	0.0796	0.	0.26263	0.	0.065	0.1187	0.	-0.07282	-0.9438	0.

The win rates of all methods are evaluated by letting players of play against each other for 1000 games, which is then repeated for 30 times, which means that in total for each evaluation.

The system would be tested based on how many players is in goal, when a winner has been declared. This does not explicitly state who would have become the runner up and so on, but provides an idea of the how far the runner up has been compared to the other players, such that one would be able to argue whether the loss was significant out performs or not.The result of the test shows that the AI player wins 38 % of the time, against random players, but still loses overall against them. It seems that the player has a hard time moving the last piece into the goal.

Second training sample data of one run consist of alpha = 0.5, gamma = 0.01 and epsilon = 0.4, for 4000.Other the player might perform poorly against these random players could be due to the player couldn't determine when it should be aggressive or defensive, as none of the state explicitly . Way of avoiding this, using the already defined states, would be training with a combination of exploration and exploitation, such that the player also would receive punishment when the most desirable action is chosen,and thus adjust the value based on how likely an defensive strategy or aggressive strategy should be used. The next training session will therefore consist of the being 70 % exploitation and 30 % random. The training and testing was done for same number of iterations iterations each.We get the below table which shows the results were bit improved performance boost in a win rate from 38 % to 45 %.

0.30605	0.	0.	0.	0.	0.	0.	0.	0.	0	-0.06902
0.	0.	0.07161	0.	0.21331	0.	0.09559	-0.08930	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.042568	0.	0.206823	0.	0.115162	-0.06527	0.	0.204583	-1.01919	0.
0.	0.061468	0.	0.21792	0.	0.162417	0.04810	-0.0929	0.210965	-1.04227	0.
0.	0.05644	0.	0.20694	0.	0.020461	0.09880	0.	0.21198	-1.02084	0.

Using the MLP brings more parameters: batch size = 1200 - which specifies the number of experiences of selecting an action given current state to calculatethe Q(s, a). It was chosen to store experience from only around the last 14 games. MLP update - networks update frequency.

It says how many times to train the estimating network before updating the decision-making network. = 0.005 is a learning-rate of the weights update. It is chosen to be small to try to prevent the learning from becoming too unstable.As mentioned before to reduce the biases coming from the initial weight association to the neural network, human expert data was used to pre train it. Games played by human were recorded, and the moves were used as the training data.

MLP was then trained for epochs = 200 with the bath size = 50.the losses were still present even after 200 epochs.

Moreover, to later check the performance of true, online training, the network was evaluated in action. With one DRL player trained on the expert data alone competing with 3 random agents, the DRL within 200 games managed to win 45 times, thus giving 22.5 % win-rate.

4 Analysis and Discussion

In this section analysis of collected data will be performed to reason about choices of selected parameters. Furthermore, discussion of center topics will be analyse or discuss :

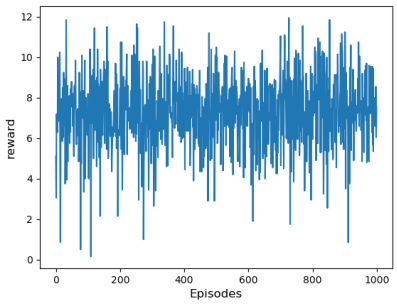

Fig. 2: Iteration of 1000

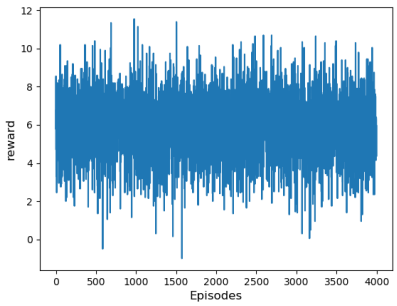

Fig. 3: Iteration of 4000

Fig. 4: Sum of reward versus episodes during Q-learning

The amount of rewards for the two Q-learning met were trained for are drastically different. The amount of training episodes are chosen based on when the accumulated reward stagnates during the learning process. From inspection of the graphs in it can be seen that the accumulated reward versus amount of episodes for the Q learning stagnates early in the training process.The parameters used for Q- learning have not been chosen in any sophisticated manner, to improve the choice of parameters, multiple strategies exist, one of them is simply to test combinations of different values for the parameters.According to the results shown in the win rate tables it can be seen that during q learning one is significantly better than other even though it has a less complicated state-action representation.
In section 3 were the training performed with a fixed discount factor on purpose, as the reward given here is based on the current position of the player, is provided as a reward. Keeping the discount factor low, result the update phase in the Q-learning being short sighted, which here is beneficial for the type of reward chosen. It would have been ideal testing different values to see if the assumption was true. It was also thought whether it would have been beneficial to train the player using or other types of exploration vs. exploitation algorithm and see whether it would make it better. Problem with greedy is that prioritizes all actions equivalently besides the highest ranking action. So if two actions has both were to be rather promising, would it only explore the most promising rather the next promising, as it would only be choose it as often as the worst possible action. One way of remedy this would work and select an action based on a probability of a Q[s,a], to give it greater probability of being chosen.

5 Conclusion

An AI player was trained using reinforcement learning. The AI player was trained using a method called Q-learning. To train the system using Q-learning was a finite set of states and actions determining where the AI player was in the game, and what it could do. The AI player was trained using an greedy method, such that the exploration rate and exploitation rate was controllable while training the player. As the player began to show promising result.

6 Acknowledgements

Thanks to Simon for developing a Python version of the Ludo game. Thanks to Poramate Manoonpong for providing lectures which formed the foundational knowledge for this project. Thanks to Karol Szurkowski for letting me compare my methods with his DRL based method.

7 Refrences

Sutton, R. S. and Barto, A. G. (2018). Reinforcement Learning: An Introduction, second edition. Adaptive Computation and Machine Learning series. The MIT Press, Cambridge, Massachusettts.

About Ludo game : https://www.ymimports.com/pages/how-to-play-ludo

Q-learning From Wikipedia, the free encyclopedia : https://en.wikipedia.org/wiki/Q-learning

Method compared from Github repo : https://github.com/Larook/DRLLUDO

Roberta Raileanu and Emily DentonArthur Szlam (2018) Multi-Agent Reinforcement Learning

N. Cesa-Bianchi and G. Lugosi. Prediction, Learning, and Games. Cambridge University Press, New York, NY, USA, 2006.