Reinforcement learning agent playing Tricks card game

Motaz Alshaikh
1854122
BSc Artificial Intelligence & Computer Science
Supervisor: Dr. Peter Hancox
Word count: 9700

Table of Contents

Introduction

This project simulates the Tricks card game which consists of 5 subgames and implements the reinforcement learning technique Q learning with two reward function approaches to train an agent to play the game at humans' level or better and build the base for developing a more complex learning algorithm that will make the agent's level of play surpasses the level of the best human player of the game.

motivation

The recent breakthroughs of reinforcement learning in different domains: Atari Games, Go game, continuous control, to name but a few [1] was the motivation for making this project. Reinforcement learning consists of two branches: model-based and model-free. Due to the complexity of modelling the environment, most researchers focus on the model-free algorithms which are inspired by how biological systems where the decisions are learned through the interactions with environment. The main model-free method used is Q learning [[2],[3]] due to its simplicity.

project aim

despite the breakthroughs of reinforcement learning, it does not perform well when dealing with multiple agents, large decision space or sparse reward such as card games [1]. **In addition, in real-world complexity environment, the state space can be huge and infeasible to represent. Therefore, a compact representation that extract the main state features is required [4]. This project will focus on representing the environment and aims to introduce an algorithm that represent the Tricks game environment for the agent to learn and play at human level or better and lay the groundwork for a much better algorithm for representing the environment of the game that can be scaled to represent the environment for other card games. To evaluate how well is the representation of the environment, the agent learning how to play the game will be evaluated. Minimax algorithm will be used in the training of the agent and in the evaluation. Moreover, the agent will be evaluated when playing against human players.**

related work

As mentioned, Tricks is not a popular game, so there is not any paper that introduced a way of representing the game environment. This project will introduce a basic way of representing the game and build for a better representation as will be explained later in the report. However, many of the reward functions was engineered using human expert knowledge such as the chess engine Deep Blue. In addition, some papers suggested a modified versions of Nash equilibria [5]. In this project Minimax algorithm will be used as a reward function because it covers all the possible variations of the game. Although minimax is an exponential function and it might seem to be not the ideal choice since the possible actions in the game is very large, some optimizations will be made to improve the performance of minimax.

Background

Q learning

Reinforcement learning consists of an agent and an environment. The agent will observe the environment to recognize its state and will make an action based on its observation. The environment will be affected by the agent action and as a result the state of the environment will change. The agent will have a reward function that evaluate the quality of its action and it will assign a reward for it. After that the agent will observe the environment to recognize the new state and continuously repeat the same process. In Q learning algorithm the agent will have a table of all the states and the possible actions with initial reward of 0, and it will update the table with the rewards it receives from the reward function [6].

Minimax

Minimax is a decision rule function used to maximize the possibilities of winning and minimize the possibilities of losses. The function goes through all the possible variation of the game considering all the possible moves a player can make and all the moves its opponents can make as a reaction to its move [7].

Tricks

Tricks is a Middle Eastern four-player card game that consists of 5 subgames which are: tricks which is what the game is named after, diamonds, queens, king of hearts and jacks. The game is played in four sets each of which consists of five rounds. In tricks, sets are called “kingdoms”. Each set or kingdom has a “king” that decides which game is going to be played. Any game the king of the set chooses to play will not be possible to be played again in the set. Once the king has exhausted all his options, the kingdom will be passed on to the next player. The game ends by the end of the fourth kingdom and the player with the highest number of points will win.

Cards

Tricks is played using the standard 52 cards which ranked from high to low as follows A,K,Q,J,10,9,8,7,6,5,4,3,2 where A is the highest and 2 is the lowest.
Words 27

gameplay

The game play is basically the same for four of the five subgames namely tricks, diamonds, queens and king of hearts. Each game will start by the player next to the dealer playing any card in their hand. Then the next player must play a card from the same suit that the first player has played if possible or they have the liberty to play any card they choose. After all four players have played their cards the winner of the trick will be the player who have played the highest ranked card from the suit that the first player has played. Having decided

the winner of the trick, the winner will be the first player to play in the next trick. Once all the players have played all their cards the round will end. However, the Jacks subgame is played differently. The players will play their cards in certain sequence that start with a jack card and continues in the rank order upwards to the Ace and downwards to the two. If a player does not have any card that complete the sequence, then they do not have any legal card to play and they will pass the turn to the next player. The game will end when all the players have played their cards.

Tricks

The objective of the game is to avoid winning any trick. A player who wins a trick will receive a -15 points. In this subgame there is not any positive points, so the best score a player would aim for is 0 points. On the other hand, the worst score a player can get is -195 points.

Diamonds

The objective of the game as the name clearly suggests is to avoid winning any trick that contains a diamonds card. The players will receive a -10 points for each diamonds card they win. The players scores will be in the range from 0 to -130.

Queens

The objective in this subgame is to avoid winning any trick that contains a queen card. The players will receive -25 points for each queen card they win. The players scores will be in the range from 0 to -100.

King of hearts

The objective in this game is to avoid winning the trick that contains the king of the hearts card. The player who wins that trick is going to receive -75 points.

Jacks

Jacks is the only subgame that gives the players the opportunity to score positive points. As explained earlier the game is played in sequential order, hence the objective of the game is to play all your cards before the other players do. The first player who finishes his cards will receive a +200 points where the second receives +150 and the third +100 and the last player +50.

Design and Implementation

During the development of this project over the course of two terms, critical decisions, that shaped the outcome of the project had to be made. First of all, deciding the suitable type of AI to implement for this card game was one of the most challenging tasks since the game is not a popular which result in the lack of resources that focus on it.

Reinforcement learning was the type of AI chosen to be implemented and Q learning algorithm is the intended algorithm to train the agent. Moreover, in order for the Q learning algorithm to work, an environment for the agent to interact with had to be created and an excellent reward function needs to be designed. The environment has been built from scratch and due to the reasons, that will be explained in this section, a modified version of the minimax algorithm has been chosen to be the reward function.

approach

In order to implement the Q learning algorithm a few design decisions had to be made. Firstly, there is in not any learning environment created for this game, so the environment had to be built from scratch. Representing all the possible states for card games can be infeasible, so a well approximated representation that extract the important features of the environment is required [4]. Several factors such as turn of play and evaluation of the cards to name but a few, were considered to approximate the state space and the action space. Moreover, the quality of an action depends on the reaction of the other players and the effect of that action will not appear immediately, so Minimax was chosen to be the reward function since it construct the game tree with all possible variations.

game environment

Creating an appropriate environment for the agent to explore and learn from was the first of the early stage's challenges faced in the project. Card games have a massive state space which cannot be represented up until now. The popular UNO game for example can have a state space that equals 10^{163} [1]**. The huge number of states is the result of several factors that the agent needs to consider. Firstly, and most importantly, the fact that other players' cards are hidden, which can increase the number of possibilities required to be taken into consideration by the agent to accurately navigate through future states when calculating the reward function as the report will demonstrate in the reward function sections. Secondly, the agent should not identify states based on its cards only and should include the other players cards into the process of determining which state of the game it is in. Obviously, this is not an easy task to accomplish since the other players' cards are hidden. As explained in the project aim section, the main purpose of the project is to successfully build a less complicated Q learning algorithm that focuses on developing an understanding of the basics of how the agent should explore and learn in such a complex environment and lay the groundwork for a much-sophisticated learning algorithm. Thus, the agent will not be concerned much about other players' cards at this stage of learning and it will try to learn how to formulate a plan to play the game, based on their cards. However, other players' cards cannot be ignored completely, and the agent should re-evaluate the situation in the game based on what the other players have played and adjust their plan accordingly. Therefore, an analysis algorithm was created to help the agent decide which state it is in more accurately. In addition to analysing what other player might play in the game, the agent will consider several factors to determine the state in the game as will be shown in the states space section.**

states space

In order for humans to perform a given task, they have to observe the environment using their senses and analyse the information gathered via these senses to accurately estimate which state they are in. Moreover, human usage of senses differs based on the nature of the task they want to achieve. For instance, if the task was to recognise other humans faces, the only sense they need to use is vision, but if the task was to watch the TV, then they have to analyse the information gathered using their eyes and ears. Similarly, in order for the agent to decide which state of the game it is in, it has to use its senses which are several functions written to help the agent analyse the environment and accurately estimate the state of the game it is in. the agent was designed to interact with the environment and learn how to play five subgames where each subgame has a different objective. Therefore, it will use different combination of function for each game. In addition, it will be interested in some information in one game and disregard it in the other as the report will explain in the next sections.

tricks' states space

As explained earlier the goal is for the agent to consider the most important factors at this stage of learning. Firstly, the agent's turn of play is one of the most important factors since the amount of knowledge the agent will have to use to analyse the situation and figure out the appropriate state and tries to learn the optimal action in that state highly depends on its turn of play. For instance, if the agent was the second player to play it will rely on the first card played only to analyse the trick where if it was the fourth player it will analyse the trick knowing all the cards played in the trick which will result in a better estimation of the state and presumably a better action. To give an example of possible scenario where the agent will benefit from considering the turn of play factor. In the scenario in figure 2 below the agent knew only about the first card and it learnt that the best action in this state which is "second player and have the same suit and does not have cards less than the highest card played", is the action "avoid winning the trick with the lowest card higher than the highest card played" As the figure shows the agent did win the trick with the worst card possible which will result in winning the next trick. the result of the agent action in this scenario is winning two tricks out of two where in the scenario in figure 1 the agent had all the necessary knowledge to recognise that it is in the state "fourth player and have the same suit and does not have cards less than the highest card played" and the best action is "win with the highest card" which helped it to win only one trick and avoid the other. Moreover, if the agent's turn of play was the first, it means that the agent will make the action instead of reacting to what other players have played which is clearly a different situation that require a different approach of play to be learned. Secondly, the stage of the game is another important factor that the agent should learn how to act based on it. The agent's approach of play in the early stage of the game should differ from the end or the middle stage of the game. For example, the agent should try to win the probably unavoidable tricks such as the trick won when playing an Ace card, at the early stage of the game to avoid winning it at the end stage of the game. If it did win that trick at the end stage, it possible that the agent will find itself in a situation where it must play the first card in the trick and it only has cards from a particular suit that all the other players have already get rid of their cards from that suit as the shown in figure 3 below. If such a scenario occurred, it will result in winning the rest of tricks which is a scenario that can be avoided if the correct approach of play was made in the early stage of the game.Thirdly, having a card from the same suit that the first player has played or not, is one of the factors that cannot be ignored. The idea that the agent is trying to learn will very much differ depending on this factor. If the agent does have a card from the

same suit, then it will try to learn the best action among "avoid winning the trick with highest card possible", "avoid winning the trick with the lowest card higher than the highest card played" or "win the trick with the highest card possible". In contrast, if it does not have a card from the same suit then it will try to learn the best action among "highest card of the suit with lowest number of cards", "highest card of the suit with middle number of cards" or "highest card of the suit with highest number of cards". Lastly, this factor is only considered in case of the agent having a card from the same suit played by the first player. The last factor is the evaluation of the agent to its card of the suit played in the trick. A function was written to help the agent with the evaluation which basically returns true indicating that the agent has a strong hand if the agent had more than one card that ranked lower than the highest ranked card played in the trick. Combining these factors resulted in the creation of 12 states for the agent to learn the optimal action in it.

base for scenarios in figure 1 and 2

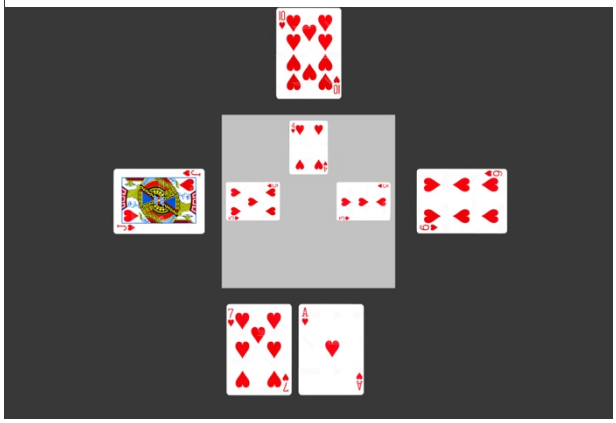


Figure 1

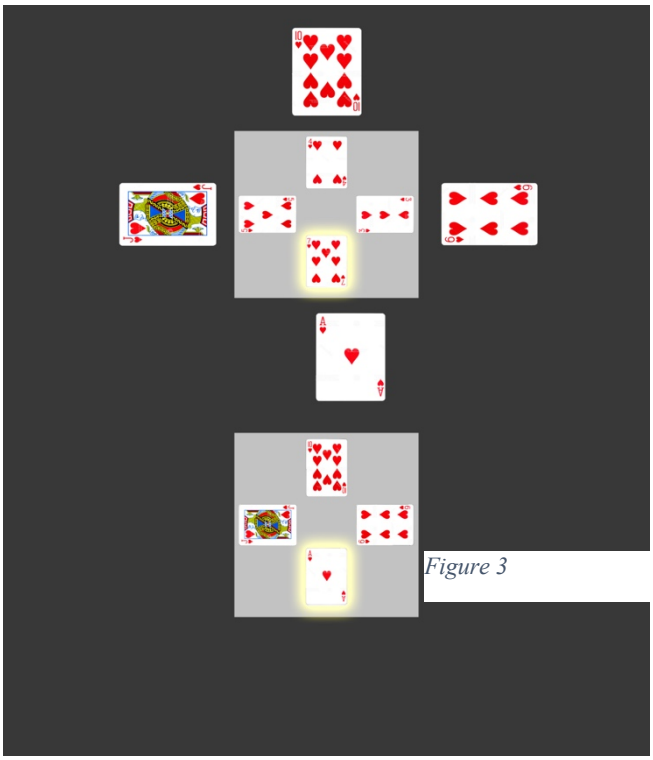
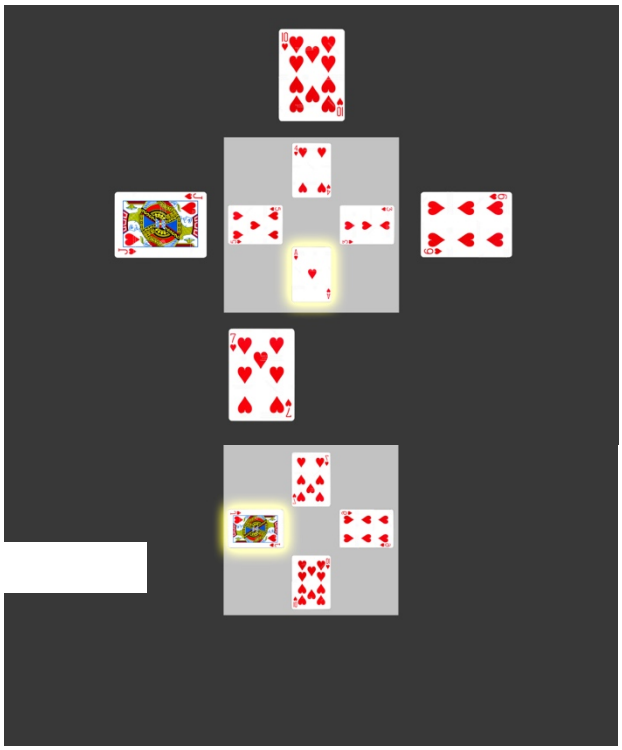
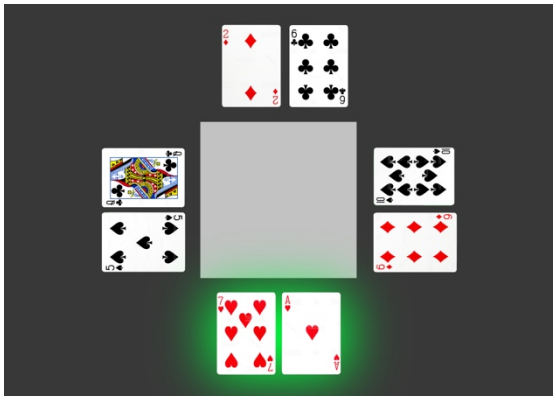


Figure 3

Figure 2





diamonds' states space

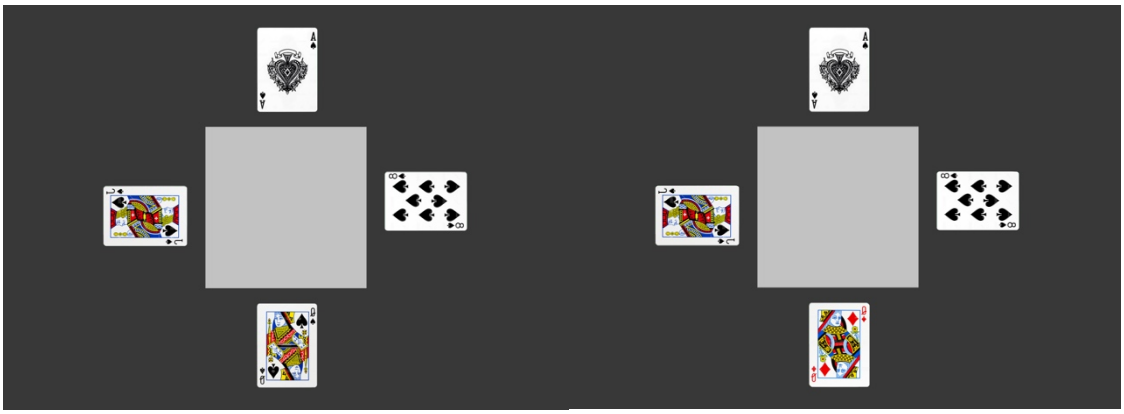
The states space for this game is created using two of the factors used in the Tricks subgame in addition to two other factors for this particular game. It shares the basic factors, turn of play and having a card from the same suit. Before we go into the details of the other two factors, the possible ways of winning a diamond card needs to be explained. A player wins a diamonds card if they won a trick that the first player in it has played a diamonds card. The other way is winning a trick that the first player has played a card from any suit other than diamonds, but one of the other players have played a diamonds card in that trick. The third factor is only considered when the agent has a card form the same suit played in the trick. the agent will consider the possibility of this trick having a diamonds card with the help of a function that returns 0 indicating that the trick indeed contains a diamond card, returns 1 if it is more likely for the trick to contain a diamond card and returns 2 if it is less likely to contain a diamond card. The details of the function will be explained in additional functionally section. The fourth factor is the evaluation of the agent's cards of the diamond suit which is done using an evaluation function that will be described in detail in interaction with the environment section. Clearly, this is an important factor since the game objective is to avoid winning any diamond card. For instance, if the agent was the first player of the trick and has a strong hand in the diamonds suit it might be better for it to play a diamonds card because it already knows that it less likely to win a diamonds card and it will decrease the chances of it winning a diamonds card in the second way mentioned earlier. The fifth factor is the evaluation of the agent's cards in the suits other than the diamonds. The fourth factor helps the agent learn the best action to avoid the first way of winning a diamonds card where this factor meant to address the second way. The agent will decide with the help of an evaluation function whether it is has a weak hand in a particular suit which means that it might be forced to win a trick in that suit knowing that it contains a diamonds card. Knowing its vulnerabilities, the agent should learn the best action to deal with it. Combining the factors discussed earlier resulted in the creation of 25 states for the agent to learn the optimal action in it.

queens' states space

Creating a states' space for this game was far more complex than the previous games because the goal is to teach the agent how to formulate a general approach of play where the objective of this game is more specific comparing to the previous games. The agent might learn an excellent plan for playing the game, but it could play one inaccurate move and it will lose the game regardless of how good the previous moves was. As in the diamonds game, the agent can win a queen card directly by playing an Ace or a King card of any suit which will result in winning the queen of that suit if it was played in the trick. or it can win a queen card indirectly by winning a trick in the hearts suit for example and one of the players did not have a hearts card and decided to play a queen of diamonds or any other suit as shown in figure 4 below. (show 2 ways of winning a queen).

similar to the previous games, turn of play and having a card from the same suit played in the trick are the main two factors. The third factor is considering the possibility of a trick having a queen card. the agent will use the same function used in the diamonds game where if the returned value is 0 would mean that the trick contains a queen card and 1 means that it most likely will contain a queen card and 2 means that it is less likely to contain a queen card. In the diamonds game the agent was more concerned about the diamonds suit, so it would make since for the agent to consider whether it has a queen card or not as the fourth factor. If the agent realised that it has a queen card, then it might focus on getting rid of this card and avoid the negative points resulted from winning a trick containing that queen card. If it does not have a queen card, then the objective is simple, just try to avoid winning other players queens. As explained, most of the subgames shares the same ideas, so the agent will use the same evaluation function used in the diamonds game to evaluate its cards in all the suits it has. Having evaluated its card the agent will consider the fifth factor which focus on whether the agent has a strong hand in at least one suit and the sixth factor which focus on whether the agent has a weak hand in at least one suit which could result in it been forced to win a trick containing a queen card. The last factor considered by the agent is whether it has a few cards of a particular suit. For instance, if the player has only two cards of a particular suit there is a high chance of it having the advantage of playing any card in their hand if that suits has been played three times. Combining these factors resulted in the creation of 73 states for the agent to learn the optimal action in it.

Figure 4



king of hearts' states space

After creating the states space for the queens game, it was an easy task to create a states' space for this game. All the 7 factors for the queens game have been reused with a few changes. The two main factors obviously are treated the same as in all previous games. Instead of using the function to check the possibility of having a queen in a trick, it has been used to check if the trick contains the king of hearts card. similarly, instead of checking if the agent has a queen card, the agent will check if it has the king of hearts card. Moreover, the last three factors are exactly the same. The agent will evaluate its card and learn about its weaknesses and strengths and will check if it has a few cards in a particular suit which will give it a sort of an advantage. Combining these factors resulted in the creation of 73 states for the agent to learn the optimal action in it.

Jacks' states space

As explained earlier this game is played differently, therefore it requires a different approach of representing the states' space. The objective of the game is playing all your cards before the others do. Therefore, the agent will evaluate its card and sort the suits based on the lowest ranked card on each one. The suit with the lowest ranked card will be considered the worst suit since it requires a higher number of cards to be played for the agent to play their lowest card. For instance, if the agent has a 2 card, then all the cards starting from the jack card until the 3 card of that suit needs to be played, so the agent can play its 2 card as shown in the figure below. The suit with the second lowest ranked card will considered as a bad suit, the third is considered as a good suit and the suit with the highest lowest ranked card will considered as the best suit. Moreover, since the sequence of play starts with the jack card of each suit, playing a card ranked higher than the jack is considered to be a good move because it does not give the other players a lot of possibilities to play comparing to playing a lower ranked card as shown in the figure below. Combining these two factors resulted in the creation of 28 states for the agent to learn the optimal action in it.

action space

When humans are asked to perform a given task. They analyse the situation, then perform one of the possible actions in that situation. The possible actions for human to choose from depends on the situation they are in and the objective they want to achieve. For instance, if the task was to greet another person, one of the factors that formulate the possible actions is the time of the day. The morning greetings phrases will differ from the night greeting phrase. Another factor might be the objective they want to achieve. An individual might use formal greeting phrases to greet a co-worker with the objective of maintaining a level of courtesy or might use informal greeting phrases to greet the co-worker with the objective of establishing friendship. Similarly, in order to represent the action space for the action, the state of the game and the objective desired to be achieved need to be considered. Different states will obviously have different action space and it is same case when the objective of the game differs.

tricks' action space

In this game, the action space is represented based on the state of the agent's turn of play. Firstly, if the agent was the first player in the trick, then the action space represented to the agent is relatively high comparing to other states. After evaluating its cards, the agent will have the option to choose a combination of three options. The first option is the number of cards they have in a particular suit. It can play from the suit with the minimum number of cards or the maximum or in-between. The second option is how strong is the suit. It can play from a suit that has been evaluated its card to be strong or play from a weak suit. The third option is the rank of the card. It can play from a low ranked, middle ranked or high ranked card. The agent should learn the best of the 18 actions possible in that state. Secondly, if the agent was not the first player and it does have a card from the same suit played in the trick. the agent will have to learn the best of the three following actions. The first action is to play a low card which means that the agent will play a card which guarantee that it will not win that trick. The second action is to play a middle card which means that the agent will play the lowest ranked card that is higher than the highest ranked played. The third action is to play a high card which means that the agent will play a the highest ranked it has and it will not be concerned about winning the trick. Thirdly, if the agent was not the first player and did not have a card from the same suit played in the trick. considering the objective of the game which is to avoid winning any trick the agent will play the highest card, but it needs to learn which suit should it play its highest card. There are three actions the agent needs to choose from. It can play the highest ranked card from the suit with the minimum number of cards or the maximum or in-between.

diamonds' action space

In diamonds game the action space represented to the agent is somewhat similar to the previous game. Firstly, after evaluating its cards, the agent will have the option to choose a combination of the two options. The first option is which suit should be played. It can choose a suit to play based on the evaluation function mentioned in the diamonds' state space section. The agent will learn whether to play from the diamonds suit, play form a suit knowing it has a strong hand in it, play form a suit knowing it has a weak hand in it or play from the suit with the lowest number of cards trying to take the advantage described in the states space section of the game. The second option is the objective of the agent. Does it want to ensure that it does not win the trick by playing the lowest card it has from a particular suit, trying to not win the trick while reserving their low cards to be played in future tricks by playing a low ranked card but not the lowest, or ensure that it gets rid of its highest card in a particular suit by playing the highest ranked card it has, to avoid being forced to play that card in a future trick. Secondly, if the agent was not the first player and it does have a card from the same suit played in the trick. Then it will choose one of three actions depending on its desire of winning the trick as explained in the later point. Thirdly, if the agent was not the first player and did not have a card from the same suit played in the trick. considering the objective of the game, which is to avoid winning any diamonds card, the agent will play the highest card, but it needs to learn which suit should it play its highest card. The

agent must decide which type of suit it wants to play, among the four types it considered when it was the first player of the trick.

Queens' action space

In the Queens game the action space represented to the agent is somewhat similar to the previous games. Firstly, the agent will evaluate its card and decide the suit it wants to play from considering four options. the agent might play from a suit knowing it has the queen card of that suit, play form a suit knowing it has a strong hand in it, play form a suit knowing it has a weak suit in it or play from the suit with the lowest number of cards trying to take the advantage described in the states space section of this game. Secondly, if the agent was not the first player and it does have a card from the same suit played in the trick. Then action space is the same as the previous games. The agent should learn when it should win a trick and it should not.
Thirdly, if the agent was not the first player and did not have a card from the same suit played in the trick. considering the objective of the game, which is to avoid winning any queen card, the agent will play a queen card or the highest card in any suit it has. The agents' options are playing a queen card, playing the highest card of the suit that it has a strong hand in it, playing the highest card of the suit that it has a weak hand in it. playing from the suit with the lowest number of cards.

kings' action space

since the states space for this game was almost the same as the queens' states space, the action spaces also share a lot of similarities. In the three main scenarios the action space is the same except that instead of considering a suit with a queen card, the agent will consider the hearts suit since the objective of the game focus only on the king of hearts game. Having the same action space does not mean that the agents evaluate its card in the same way for each different game as will be explained in later sections of the report.

Jacks' action space

As described earlier this game is played in a sequence, so the action space for this game is relatively straight forward since the players do not have a lot of option to play. The maximum number of actions allowed for the player to play is 8, but the number of actions represented to the agent is 5.
The agent can consider completing the sequence in descending order which provide it with 4 options. However, if the agent wanted to complete the sequence in ascending order, it will not be concerned much about which suit it should play due to the reason explained in the states space of game.

reward function

card games environments may suffer from sparse rewards [1]. Any action might have a significant effect, but that effect might not occur immediately, so writing a reward function to measure the quality of an action can be extremely hard specially when dealing with imperfect-information games [2].
The reward function that was chosen to help the agent learn through its interaction with the environment is the minimax algorithm. Minimax seemed to be an excellent choice, because it constructs a game tree which covers all the possible variations. However, a major drawback for the algorithm is that it requires an extensive number of calculations that keep increasing exponentially whenever the number of the possible actions becomes higher. In order to overcome this major drawback, some optimizations are required to help the agent explore the game tree efficiently without excluding an excellent action that seems to be bad at first, but it is actually an excellent action when considering its effect on the game in a later stage.

game-based minimax

This approach is used to train the agent to play the games diamonds, queens, king of hearts and jacks. The reward function will have a prior knowledge about the cards of each player, and it will use this knowledge to construct the game tree considering the possible actions each player makes. The learning process using this approach is subject to the scenarios that occurs in the training stage. Therefore, one of the drawbacks of this approach is that the agent will not be able to make the best action when it experiences a new scenario that did not occur in the training. This drawback occurs in the learning process of humans. A human being might not choose the best action for a given situation if it did not have any experience dealing with such a situation.

generalised minimax

This approach is used to train the agent playing the tricks game. the aim is to overcome the fundamental problem of the learning process in general. Instead of learning from the scenarios occurred during the training, the agent will try to consider all the possible scenarios that could occur in general. In the game-based approach the reward function will build the game tree knowing the cards that each player has which limits the number of possible variations of the game. However, In the generalised minimax the game tree will be constructed assuming all the possible combination of cards each player might have. Having a higher number of possible actions that need to be considered, will require much more efficient optimizations which is one of the drawbacks of this approach. However, successfully designing efficient optimization for this approach will result in much more better result of learning in addition to the (difference in time and calculations:) comparing to the game-based approach.

Reward calculation

The function will evaluate the tricks considered in the future based on the game calculation points. In the tricks game for example, if the depth of the reward function was two then the function will assign a reward of -15 for each trick the agent has won in depth two. And then the same process will be done at the tricks in depth one with

addition to the cumulative reward of the possible variations start based on that trick. The reward for each trick in the future will be multiplied with discount factor of 1.05 that keep increasing by 0.0.5 whenever the function calculates the reward for a deeper variation.

Additional Functionality

3.4.1 creation of the game

play order

To decide the kingdoms order and which player should be the king of the first kingdom, the "deal_cards" function that shuffles the deck and deals the cards to the players will be called. In the first round of the game only, the function will set the order of the play starting from the player who has the 7 of hearts card.

gameplay

After dealing the cards, the king of the kingdom will be asked to choose a subgame to be played. Then, the game loop will repeatedly call the "play" function that will represent the players with the cards played in the board and ask them to play a card. After each player has played a card and if the subgame was not Jack, the winner of the trick will be decided, and the play order will be resets to start from the winner of the trick using the functions "trick_winner" and "play_order". If the subgame was Jacks, then it will repeatedly call the "play" function until all the players have played their cards. Once the players have played all their cards, the players will be represented with the scores of this round and the updated scores of the overall game. If the players have finished playing the four sets or kingdoms, then the overall scores will be shown to the them and they will be asked if they would like to play another game.

Interacting with environment

The agent interacts with the environment by using a helper functions that helps the agent observe the environment and recognise which state of the game it is in. then it will make an action in the form of playing a card expecting the evaluation of the quality of the action made from the reward function.

Tricks' agent

The agent identifies the states of the game based on the four factors mentioned earlier. Turn of play, stage of the game, having a card from the same suit played in the trick and whether it has more than one card that can avoid winning the trick.
Firstly, the turn of play is straightforward to calculate, the agent will check how many players have played a card in the game and based on that it will know its turn of play. If the board is empty, it will assume that it is the first player and if there is only one card its turn will be the second and so on. Secondly, since the players use 13 cards in each game, so the stage of the game is divided as follows. The first five tricks are considered

to be in the early stage of the game and the next four is in the middle and the last 4 is in the end stage.
Thirdly, the agent will check if it does have a card from the suit played by the first player of the trick and it will know whether it has a card from the same suit or not. Lastly, if the agent appeared to have some cards from the same suit played in the trick, it will check if it has at least two cards that considered to be lower ranked than the highest ranked card in the trick.

Diamonds' agent

The agent identifies the states of the game based on the five factors mentioned earlier. Turn of play, having a card from the same suit played in the trick, possibility of the trick containing a diamond card and the evaluation of the agent's hand in the diamonds suit and in the other suits. Firstly, turn of play and having a card form the same suit played in the trick is calculated the same way as the previous game and it will be also the same for the queens and king of hearts games. Secondly, the agent will observe the cards played in the board and check if one of them is from the diamonds suit. If it does have a diamonds card then it will consider that fact when identifying the state of the game, and if not, it will use a helper function to analyse the trick. The function will return true if the trick is most likely will contain a diamonds card and return false it is less likely to contain a diamonds card. Thirdly, an evaluation function will be called to help the agent know whether it has a strong hand in the diamonds suit in particular and whether it has a strong hand in the other suits.

Queens' and agent

The agent will use the same functions used in the Diamonds game to analyse the observed data form the environment with slight changes and two additional factors to consider. Firstly, the agent will check if a trick contains a queen card instead of checking whether it has a diamonds card or not. Secondly, the evaluation function will be used in the same way as the diamonds game. Knowing whether it has a strong hand in one suit and weak hand in the other will help the agent decide which state of the game it is in. Thirdly, At the start of the game the agent will check whether it has a queen card in its hand or not and it will check whether it has a two or less cards from a particular suit. Knowing these information helps the agent to estimate the state of the game.

King of hearts' agent

The agent will use the same functions used in the Queens game to analyse the observed data form the environment with slight changes. Firstly, the agent will check if a trick contains the king of hearts card instead of checking whether it has a queen card or not. Secondly, the evaluation function will be used in the same way as the Queens game. Thirdly, At the start of the game the agent will check whether it has the king of hearts card in its hand or not and it will check whether it has a two or less cards from a particular suit.

Jacks' agent

As explained in the state space section, the suits will be sorted based on the lowest ranked card at the start of the game. The agent in this game will observe its legal moves and decide which state it is in base on its observation of its card at the start of the game. For instance, if the hearts suit was considered to be the "worst" suit and the agent has the option to play the card seven of hearts, then it will consider that it has the option to play from the worst suit. Moreover, If the agent is able to play any card ranked higher that the jack card it will consider that it has the option to play high ranked card regardless of the cards' suit. After analysing its legal moves the state of the game the agent should be able estimate which state of the game it is in.

Representing the action space

A crucial point that needs to be considered during the training of the agent is representing the agent with the valid actions. If the action space in a particular state consists of three actions and only one of them was a valid action, then the agent will be forced to pick that action and quality of that action should not be evaluated since it is forced action. For this reason, the agent was provided with two helper methods that check the validity of the actions. The first function is used when the agent has a card from the same suit that the first player played, and it consider three scenarios. Firstly, If the agent has only high cards that are higher than the highest card played in the trick, then the agent should not be presented with the action "play a card that guarantee avoid winning the trick". Secondly, If the agent has only low cards that are lower than the highest card played in the trick, then the agent should be presented with the action "play the highest card" only and the quality of the action should not be evaluated since it is forced. Regardless of what card the agent plays, it will not win the trick. Thirdly, if the agent has only one valid move, then it should be treated in the same way described in the second scenario.
The second function is used when the agent is the first player of the trick or when the agent does not have a card from the same suit played in the trick. The function will analyse the agent's evaluation of its card and it will remove any invalid action. For instance, if the agent has weak hands in all the suits, the function will remove any action that suggest that the agent will play from a suit that it has strong hand in it.

Analysing tricks

This function designed to help optimizing the exploration of the game variation in the reward function and help the agent when evaluating its cards. The function will be used in the tricks subgame only since it has a very high number of possible actions. Firstly, in the tricks game, if the highest cards played in the trick is an Ace, then all the players should play their highest cards. Therefore, if a player played in that trick the 8 card of that suit, the function would eliminate the possibility of that player having a card ranked higher than the 8 card. Another possibility would be if a player played a card from a different suit, then the function will eliminate the possibility of that player having a card ranked higher than the rank of the card it just played. For instance, if the player played the king card of a particular suit, then it should not have the Ace card of

that suit. The last scenario considered only if the winner of the trick was the last player. If the last player won the trick by playing the king card, then the possibility of it having the ace card should be eliminated because it observed the full boards and knew that it is going to win the trick, so the best action is to win the trick by playing your highest card possible.

Evaluating suits

The function used in diamonds, queens and king of hearts games. the function will evaluate the suits differently based on the objective of the game. The function will evaluate each suit based on whether the agent has the ability to avoid winning the trick if it wants. For instance, if the lowest ranked card the agent has is 5 then the function will evaluate the suit as weak suit since the agent might be in a situation where the other players play a lower card than its card and it will be forced to win the trick and that trick might contain a diamonds card, queen card or the king of hearts card. Another factor that the function takes into count is whether the agent has a high ranked card such as an Ace or king. If it does, then the evaluation will be that the agent is weak in that suit since it could be in the situation where it forced to play the Ace or king card when the trick contains a queen card. Similarly, in the king of hearts game the function will consider if the agent has the Ace of the hearts or not. The function should consider the number of cards the agent has when evaluating the suit based on the high cards because if the agent has a lot of cards from that suit, it is less likely to be forced to play that high card.

4 Results

Due to the large number of possible actions, the agent could not explore the game tree to the enough depth despite the optimization functions used in the process. Therefore, the rewards in the Q table learned by the agent were not assigned accurately which resulted in poor performance. The agent was trying to maximize the reward in the current states and could not look further in the future and understand the later effect of its actions. Therefore, the agent was provided with a new Q table with rewards assigned by human to help the agent play and test the representation of the game environment. In order to evaluate the representation of the environment, the agent was tested playing 15 games against 2 professional players, 6 intermediates and 3 beginners with a total number of 11 players. The players were asked to complete a brief survey to evaluate the agent.

Jacks' agent

The agent won most of the games against the beginner players with average of 10 – 5 and lost by a small margin against intermediates with average of 8 – 7. However, the agent won by a small margin against professional players with the average of 8 – 7. We can infer from the results of the games and the survey the players completed that the representation of the environment enabled the agent to formulate complex plans that

beginner could not formulate. The fact that the agent lost to intermediates and won against professionals shows that the probability of winning this subgame depends on how good your hand is more than your decisions. Although the game environment was represented almost perfectly to the agent, still it lost against the intermediates because of the nature of the game not the agents' decisions.

Tricks' agent

The agent was able to win most of the games against the beginner and intermediate players with average of 13 – 2 and 9 – 6. However, it lost managed to win against a professional player 8-7 but lost against the other 9 - 6. We can infer from the results of the games and the survey the players completed that the representation of the environment enabled the agent to formulate complex plans that beginner and intermediates did not know how to react to it. The environment was represented to the agent accurate enough to enable it to play at a professional human players' level.

Diamonds' agent

The agent was able to win most of the games against the beginner players with average of 12 – 3 and lost most of the games against intermediate and professional players. The average score against the intermediates was 8 – 7 and the professionals was 10 – 5.
We can infer from the results of the games and the survey the players completed that the representation of the environment enabled the agent to formulate general plans that beginner most likely will not know how to react to it. The agent managed to win almost half of the games against intermediate and won a few games against professionals which suggests that it does not make very bad moves when it is in an easy situation to play, and it does not play the most accurate move. Even though, the states space for this subgame was bigger than the tricks and jacks subgames, the agent could not learn how to play in complex scenarios because the objective of the game is more specific than the two subgames.

Queens' agent

The agent was able to win most of the games against the beginner players with average of 10 – 5 and lost most of the games against intermediate and professional players. The average score against the intermediates was 9 – 6 and the professionals was 11 – 4.
We can infer from the results of the games and the survey the players completed that the representation of the environment enabled the agent to understand the general principles of the game but similarly to the diamonds subgame the agent does lack an understanding of complicated scenarios. The objective of this game is more specific than the diamonds game hence the results are worse.

king of hearts' agent

The agent was able to win most of the games against the beginner players with average of 8 – 7 and lost most of the games against intermediate and professional players. The average score against the intermediates was 10 – 5 and the professionals was 12 – 3. We can infer from the results of the games and the survey the players completed that the representation of the environment enabled the agent to understand the game at beginners' level. it did win against the beginners by a small margin but did not do well against intermediates and professionals. The objective of this game is more specific than all the games hence the results were the worst.

conclusion

The project showed that the state space for card games can be well approximated to enable the agent to play at intermediate or professional level depending on the complexity of the game. Although the state space was represented considering the agent's cards only and disregarded other important factors, the agent managed to perform well in two of the five subgames and play at professional players' level.

Discussion and Future Improvements

In conclusion the project had primary goal and secondary goal. The first goal is to create an environment for the agent to play and learn which was achieved. The second goal is to design a reward function that evaluates the actions accurately even though the effect of these action might occur at a later stage of the game. However, due to large number of action space considered the intended reward function did not work properly. Moreover, the project aimed to lay the groundwork for much better representation of the environment as will be covered in this section. Firstly, one of the obvious limitations of current representation of the environment is that the number of states is fixed. Thus, the agent will reach a state where it can learn any more regardless of how many games it plays. There are two possible solutions to this problem. The first solution is to train the agent to play and discover new states. The agent will use the best Q table to play the game and when the reward function assigns a negative reward to its action it will observe all the factors in the environment and create a new state. Once the training is finished and the agent learned new states, then it will start a process of going over the states and eliminate the irrelevant factors in the new states learned. Secondly, the agent needs to consider other players cards. Therefore, better analysis functions where the agent try to predict other players cards by putting itself in the other players shoes and assume that other players think and play the same way it does.

Bibliography

[1] Zha, D., Lai, K.H., Cao, Y., Huang, S., Wei, R., Guo, J. and Hu, X., 2019. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*.
[2] Lin, C.J., Jhang, J.Y., Lin, H.Y., Lee, C.L. and Young, K.Y., 2019. Using a reinforcement q-learning-based deep neural network for playing video games. *Electronics*, *8*(10), p.1128.
[3] Boussakssou, M., Hssina, B. and Erittali, M., 2020. Towards an Adaptive E-learning System Based on Q-Learning Algorithm. *Procedia Computer Science*, *170*, pp.1198-1203.
[4] Ribeiro, C.H.C., 1999, July. A tutorial on reinforcement learning techniques. In *Supervised Learning Track Tutorials of the 1999 International Joint Conference on Neuronal Networks*.
[5] Heinrich, J. and Silver, D., 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.
[6] Lin, C.J., Jhang, J.Y., Lin, H.Y., Lee, C.L. and Young, K.Y., 2019. Using a reinforcement q-learning-based deep neural network for playing video games. *Electronics*, *8*(10), p.1128.
[7] Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P. and Zhang, D., 2017, August. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval* (pp. 515-524).