Learning to Play Texas Hold'em Using Reinforcement Learning

UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

Department of CSIS

Bachelor of Science in Computer Systems

Author: Jamie Mac Manus

ID: 15147312

Supervisor: J.J Collins

Abstract

In recent years the area of machine learning has gained a lot of ground in a broad range of areas. A particularly interesting problem pertaining to machine learning is how we can develop useful AIs in a more hands off fashion. This problem is addressed by a machine learning paradigm named reinforcement learning. Reinforcement learning allows us to set up an agent in an environment after which the agent can explore the environment and begin to learn the more which actions that it should take in the different scenarios it can find itself in. This avenue of machine learning is suited to a broad range of problems but one interesting area is that of imperfect information games such as texas holdem.

Contents

List of Figures

Chapter 1

Introduction (10)

In this section I will introduce the subject area of this Final year Project (FYP). I will then go on to give an overview of the report, establish some goals for the project along with some of the motivations for choosing this subject area.

1.1 Overview (3)

Since the inception of machine learning, games have been a key problem area that has seen a lot of focus from top academics. Furthermore, the development of some machine learning strategies that can be applied to games has also lead to these strategies being applied in many different domains, many of which being very beneficial in practice.

1.2 Objectives (2)

1.2.1 Reinforcement Learning for Leduc Hold'em

In the past, methods such as counterfactual regret minimization (CFR) have been used to develop agents that can play no-limit texas hold'em to a super-human level. This includes the 2018 champion of the Annual Computer Poker Competition, slumbot(Jackson 2013). There have also been attempts to solve the limit version of the game using reinforcement learning (RL)(Heinrich & Silver 2016).

Throughout the course of this project I will be using an iterative approach to solving the problem. As such the first agent that I will develop will seek to tackle a simplified version of hold'em. Specifically I will be attempting utilise the algorithm outlined in(Heinrich et al. 2015) to develop an agent that can play a simplified version of texas Hold'em called Leduc Hold'em.

1.2.2 Apply the Algorithm to a More Complex Poker Variant

When I have successfully applied the algorithm to Leduc Hold'em my goal is to then apply the same algorithm to a more complex game.

1.2.3 Web Interface to Facilitate Play Against the Agent

The focus of this report will largely be research. However it is also my goal to create a product that will be fun and useful for the general public. As such another objective will be to create a website that will allow users to play heads-up against the final product.

1.2.4 Understanding Reinforcement Learning

As this project is very specific and academic, one of the larger challenges will be to gain a strong knowledge of the domain. This means learning the history of RL, the types of problems that it has been used to solve and the specific details of different RL algorithms.

1.2.5 Understanding ML in Imperfect Information Games

A successful project will require a high degree of knowledge from the broader domain of RL. However, it is also the case that I must become closely familiar with the existing academic literature in the area of RL with respect to imperfect information games. This will allow me to avoid taking approaches that have previously shown to fail and also allow me to contribute to the existing literature without simply replicating what has already been done.

1.3 Contribution (1)

1.4 Methodology

1.5 Motivation (2)

Chapter 2

Background (18)

The aim of this chapter is to give the reader background information on the problem domain in order for them to understand the rest of the report. Then we will go into more detail on the areas of reinforcement learning, game theory and texas hold'em. However, this final year project is heavily based on machine learning so I think it is important for the reader to have some background on this area first.

Machine learning is an area of computer science that tackles how we construct computer programs that improve with experience(Mitchell et al. 1997). The term was coined by Arthur Samuel in his 1959 paper where he discussed machine learning methods using checkers. Since then there has been a great deal of advancement in the field. Some of the notable early contributions being the discovery of recurrent neural networks in 1982 and the advancement of reinforcement learning by the introduction of Q-Learning in 1989. Recently we have seen some of this early academic work culminate in more practical achievements such as Facebook's DeepFace system which, in 2014, was shown to be able to recognise faces at a rate of 97.35% accuracy, a rate that is comparable to that of humans. Another example of recent achievement is Google's AlphaGo program which, in 2016, became the first

program to beat a professional human player.

It should be becoming clear that machine learning can be a solution to a wide array of problems and as both hardware and software continue to improve it's reach will only continue to grow. We are starting to see machine learning systems become a key component of many companies business model. Since certain machine learning techniques are great at prediction, machine learning has been widely used for content discovery by companies such as Google and Pinterest. Other business applications include the use of chatbots as a part of customer service, self-driving cars and even in the field of medical diagnostics.

2.1 Reinforcement Learning

When I began to research the possibility of creating a texas hold'em playing agent using machine learning techniques it quickly became apparent that reinforcement learning would be the most suitable approach. Thus I began to research the area in order to gain an in-depth understanding of the area. This research included a Udemy course as well as reading in part Reinforcement Learning: An Introduction a book published in 1998 by Andrew Barto and Richard S. Sutton. In the following section I will outline the findings rendered by this research.

Reinforcement learning is a way of programming agents by reward and punishment without needing to specify how the task is to be achieved(Kaelbling et al. 1996). As such the primary components of a reinforcement learning problem are an agent which exists in an environment. From a simplified perspective we can think of the environment as a set of states, actions and rewards. The objective for the agent is to maximise cumulative reward. This is done by developing a policy that will dictate which actions should be taken in each state.

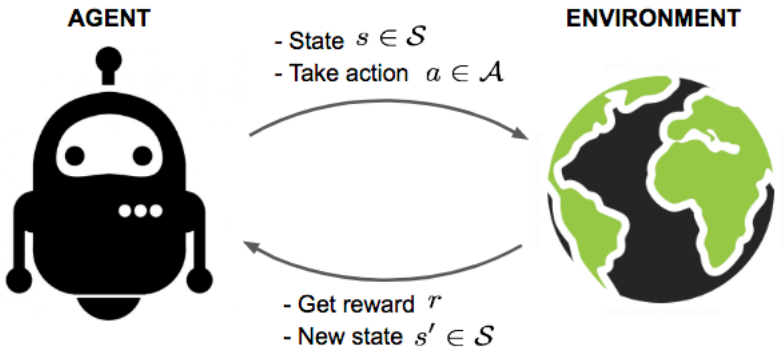

Figure 2.1: Reinforcement Learning

2.1.1 Explore-Exploit Dilemma

When it comes to reinforcement learning one of the first questions that we have to ask is how we explore the state space. An example that is often used to conceptualize this problem is the multi armed bandit problem. Let's say an agent is in a room with a number of gambling machines. Each of these machines has an arm that, when pulled will return a reward of 0 or 1 based on some underlying probability(Kaelbling et al. 1996). The agent has a limited number of total pulls. So the question becomes how do we distributed these pulls in order to maximise return? Well, first we have to ensure that we explore enough that we find the machine with the best reward probability and second, we must then exploit this machine to the best of our abilities.

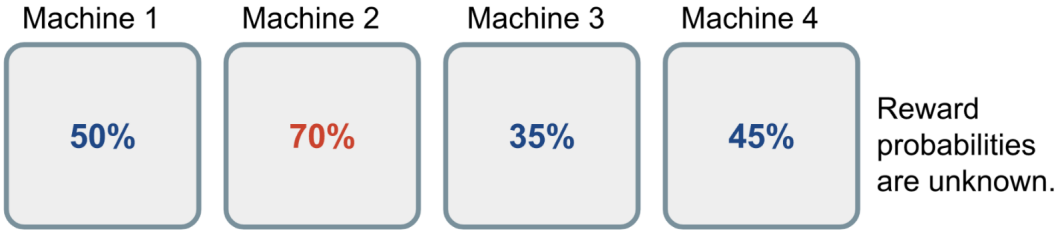

Figure 2.2: Multi Armed Bandit

There are a number of approaches that can be taken to solve this problem, we will now briefly discuss two of these methods.

ϵ-Greedy Solutions

The first approach that we will discuss is the ϵ-greedy strategy. This approach was first proposed in(Watkins 1989) and is a very simple and widely used method. The ϵ-greedy strategy involves choosing a random lever some proportion ϵ of the time, and choosing the lever that has been established to give the highest reward the rest of the time.

There are a number of variations of this method, the first being the ϵ-first strategy. With this strategy we take all of our random choices first, allowing us to establish the best bandit, after which we exploit this bandit. However, as stated in(Vermorel & Mohri 2005) this simple approach is sub-optimal because asymptotically, the constant factor prevents the strategy from getting arbitrarily close to the optimal lever.

This is where the ϵ-decreasing strategy becomes useful. Here, the proportion of random lever pulls decreases with time. Generally if our initial epsilon value is ϵ_0 then our epsilon value at time t will be $\epsilon_t = \frac{\epsilon_0}{t}$.

Interval Estimation Strategy

Another approach that can be used is called the interval estimation strategy. With this method we initially give an optimistic estimate of the reward to each bandit within a certain confidence interval. Then we simply take a greedy approach to our exploration. Less explored bandits will have a artificially higher reward estimate and thus they will be greedily chosen, thus allowing us to evaluate each of the bandits.

In the context of reinforcement learning, state space exploration through the ϵ-greedy approach is generally sufficient.

2.1.2 Markov Decision Processes

Reinforcement learning problems are generally modelled according to what is called a markov decision process (MDP).

- S - a set of states.
- A - a set of actions.
- P - a set of state transition probabilities
- R - a set of rewards
- γ - a discount factor

2.1.3 Policy Evaluation and Policy Improvement

As mentioned above the primary focus of reinforcement learning is to find a policy (denoted by π) that allows the agent to take actions in states that lead to the maximum possible reward. There are two primary problems that we must solve in order to do so.

The first is called the prediction problem, also known as policy evaluation. This involves computing the values of states given some arbitrary policy(Sutton et al. 1998). For example a state would have a high value if the reward for reaching that state was high. A state would also have a high value if we were only one action away, according to the supplied policy, from a state that renders a high reward. However a state would have a low value if, according to the policy, there was no path to a state that would return a positive reward in the foreseeable future.

The second problem is known as the control problem, also known as policy improvement. This involves changing the policy in order to improve our cumulative reward. The policy improvement process can only occur when the we have performed policy evaluation. Let's say, after our evaluation step,

we know the value of some state s. Note that this value is calculated with the condition that we take some action a in state s. But, if we take some other action a' would this render a higher value for s? If the answer is yes then we update the policy.

These two operations can be seen as the core of reinforcement learning. In the next section we will discuss different reinforcement learning methodologies. Some of the main differences are in how method each approaches the prediction and control problems.

2.1.4 Dynamic Programming (3)

Dynamic programming (DP) refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision process(Sutton et al. 1998). Dynamic programming is not widely used in practical reinforcement learning applications due to its assumption of a perfect MDP and it's high computational requirements. Despite this it is very important from a theoretical standpoint as it serves as an introduction to a number of important reinforcement learning concepts. Furthermore, it provides a basis for many algorithms that are used in practical reinforcement learning applications.

Policy Evaluation in Dynamic Programming

When discussion policy evaluation we talk about a state-value function a value function. This is simply the mapping of states to their corresponding values and is denoted by v.

Since the environments's dynamics are completely known we can apply an iterative solution to finding the value function. If we consider a series of approximate value functions $v_0, v_1, v_2, ...$ The initial value function, v_0 is chosen arbitrarily and each successive generation is obtained by using the

Bellman equation for v_π as an update rule(Sutton et al. 1998):

$$v_{k+1}(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] \tag{2.1}$$

In order to produce each successive approximation of v_{k+1} from v_k we apply the operation outlined to each state s. As shown above our new value for s is based on a combination of the expected immediate rewards(R_{t+1}), and the expected values of each of the states that we can transition to(S_{t+1}) given policy π. This operation is applied to each s for each iteration(k) of the algorithm. It can be shown that as $k \to \infty$ v_k will converge to v_π, the correct value function for policy π. This algorithm is called *iterative policy evaluation*(Sutton et al. 1998).

Policy Improvement in Dynamic Programming

Since we have now determined how good it is to follow v_π we can use this information to determine how we should modify this policy in order to improve it's value. If we assume that π is a deterministic policy then $\pi(s)$ will return some action that we must take. Now the question becomes what if we take some other action $a \neq \pi(s)$? Well we must consider whether or not choosing this action, and then continuing to use the existing policy will improve the value of the policy. If it does then we will choose this new action.

The logical extension of this approach is to apply it to each state and each possible action. As such we will select what appears to be the best action at each state. We can thus denote our new greedy policy π' as:

$$\pi'(s) = argmax\, \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, A_t = a] \tag{2.2}$$

Essentially here we are determining the value of each available action in the current state, using the same operation as outlined in the policy evaluation phase and choosing. Then the argmax function will select the action with

the highest value. Finally we assign this action to be the one we will choose in state s, according to the new policy π'.

2.1.5 Monte Carlo (3)

In Monte Carlo, unlike dynamic programming, we do not assume complete knowledge of the environment. Monte Carlo methods require only experience. We sample sequences of states, actions, and rewards from interaction with the environment(Sutton et al. 1998). Monte carlo evaluation is an episodic process this means that we only update our action values after an episode has completed.

Monte Carlo Policy Evaluation

2.1.6 Temporal Difference Learning (3)

2.2 Game Theory

After taking a deep dive on reinforcement learning and the papers surrounding RL in texas hold'em it became clear that a pure reinforcement learning approach would not be feasible. The main reason for this was the fact that texas hold'em is an imperfect information game. As outlined by(Dahl 2001):

> Note that the concept of game state values, which is the key to solving perfect information games, does not apply to imperfect information games, because the players do not know from which game states they are choosing.

As such it became apparent that some other techniques would have to be incorporated in order to create a competent texas hold'em agent. This is when I discovered(Heinrich et al. 2015) which utilises a game theoretic model, called fictitious self-play, as an inspiration for a machine learning based approach to

imperfect information games. In the following section I will discuss elements of game theory that are relevant to this approach.

2.2.1 Extensive-form Games

2.2.2 Nash Equilibria

2.2.3 Fictitious Play

2.3 Supervised Learning

Supervised learning involves an agent which observes some example input-output pairs and learns a function that maps from input to output.(Russell & Norvig 2016). This learned function can then be used on new input data, that wasn't used to train the agent and the agent should be able to give an accurate output. As such this learning task is a generalization problem. The agent must be able to identify general features of the input data and how they map to the output. Common examples of

2.4 Texas Hold'em

Texas hold'em is one variant of the family of games called poker. Poker is a group of card games that combine gambling, strategy and skill. All poker variants have three core similarities. There is betting involved, there is imperfect information (ie cards remain hidden until the end of a hand) and the winner is determined by combinations of cards. We will now discuss texas hold'em poker in more detail.

2.4.1 Game Structure

Texas hold'em consists of four betting rounds. Initially each player is dealt two private cards. These remain face down and only the person who received these cards may view them. In the next three rounds five public cards are dealt face up on the table. The second round of dealing is called the flop, where three cards are dealt. The third round is called the turn where one additional public card is dealt. Finally in the fourth round another card is dealt which is called the river.

At each round, after the cards are dealt, the players are given the opportunity to take a number of betting related actions. We will discuss the permitted actions in the next section.

In order for players to be incentivized to continue playing in a wider array of situations, blinds are required. Blinds are a mandatory bet that must be posted by two of the players present at the game. These two bets are called the big blind and the small blind, the big blind being twice that of the small blind. As hands are played the big and small blinds are posted by different players in order to distribute the cost fairly.

The big and small blind are the first two bets that contribute to what's called the pot. The pot is the collection of all of the current chips bet by the players. When a player wins a hand then what they receive in return is the pot.

The final structural component of the game is player stacks. Each player will start the game with a certain amount of chips. If a player wins a pot then all of the chips in the pot are transferred to the winners stack.

2.4.2 Actions

2.4.3 Hand Values

Chapter 3

Application Development (10)

3.1 Requirements

3.2 Design

3.3 Backend API

3.4 Frontend Website

3.5 Testing

3.6 Issues

Chapter 4

Empirical Studies (21)

4.1 Experiment 1 (3)

4.2 Experiment 2 (3)

4.3 Experiment 3 (3)

4.4 Experiment 4 (3)

4.5 Experiment 5 (3)

4.6 Experiment 6 (3)

4.7 Experiment 7 (3)

Chapter 5

Conclusions (6)

5.1 Summary (2) w25

5.2 Reflections (2) w25

5.3 Future Work (2) w25

Bibliography

Dahl, F. A. (2001), A reinforcement learning algorithm applied to simplified two-player texas holdem poker, *in* 'European Conference on Machine Learning', Springer, pp. 85–96.

Heinrich, J., Lanctot, M. & Silver, D. (2015), Fictitious self-play in extensive-form games, *in* 'International Conference on Machine Learning', pp. 805–813.

Heinrich, J. & Silver, D. (2016), 'Deep reinforcement learning from self-play in imperfect-information games', *arXiv preprint arXiv:1603.01121* .

Jackson, E. (2013), Slumbot nl: Solving large games with counterfactual regret minimization using sampling and distributed processing, *in* 'AAAI Workshop on Computer Poker and Incomplete Information'.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement learning: A survey', *Journal of artificial intelligence research* **4**, 237–285.

Mitchell, T. M. et al. (1997), 'Machine learning. 1997', *Burr Ridge, IL: McGraw Hill* **45**(37), 870–877.

Russell, S. J. & Norvig, P. (2016), *Artificial intelligence: a modern approach*, Malaysia; Pearson Education Limited,.

Sutton, R. S., Barto, A. G., Bach, F. et al. (1998), *Reinforcement learning: An introduction*, MIT press.

Vermorel, J. & Mohri, M. (2005), Multi-armed bandit algorithms and empirical evaluation, *in* 'European conference on machine learning', Springer, pp. 437–448.

Watkins, C. J. C. H. (1989), Learning from delayed rewards, PhD thesis, King's College, Cambridge.