INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

An Explanable AI for poker bidding

Ye Yuan

Bachelorarbeit

Matrikelnummer 11795771
Betreuer Prof. Dr. François Bry
Mentoren Prof. Dr. François Bry,
Dr. Felix Weitkämper

Abgabe am 06.06.2021

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 06.06.2021 Ye Yuan

Abstract

Texas Hold'em is a particularly interesting target for artificial intelligence since it is a game with imperfect information - the cards of the opponent are not revealed during play, the three community cards are dealt faced up phase by phase and it generally requires one player to outwit the other by forcing them to make risky assumptions. In this paper we try to understand the bidding strategy of the top professional poker players in the first round of the Pre-flop phase using rule based algorithms to explore the factors that will influence the players' decision and make their moves explainable. We have used two different rule based algorithms, which are taken from propositional learning and probabilistic inductive logic programming (PILP), to conduct the experiment. The evaluation results of two preliminary experiments indicates that the probabilistic inductive logic programming algorithm outperforms propositional learning and we would draw a conclusion that the professional players would tend to make unusual moves based on deceptive strategies during the pre-flop phase of Texas Hold'em. As an extension we could say that PILP systems are moreover able to explain the black-box bidding approach used by professional players.

Zusammenfassung

Die deutsche Zusammenfassung ist eine eins-zu-eins-Übersetzung der englischen Zusammenfassung.

Acknowledgments

First i would like to thank Prof. Bry for providing me with all the necessary facilities for research and constructive advice on thesis proposal.

Then I would like to express my thank to my supervisor, Dr. Weitkämper, for the assistance when i have met obstacles and also for helping me polish the paper, without which the paper would never be accomplished.

Besides i would like to thank Dr. Rückschloss and Prof. Riguzzi for sharing expertise and valuable guidance. It is you that makes my project smoother.

I would also express my gratitude to my friends and parents who support and encourage me unceasingly when i met difficulties.

At last i also place on record, my sense of gratitude to one and all, who directly or indirectly, assisted me.

Contents

1 Introduction

Studying and imitating the decisions of professional players has played an important role in designing the artificial intelligence(AI) of a game, especially one with imperfect information like Texas Hold'em. Unlike the game with full information where every situation is predictable, a game with imperfect information contains so much possibilities which are difficult to calculate individually. Therefore, observing the actions of top professional players and the rationale behind such actions contributes in designing good AI.

Texas Hold'em is considered the most popular poker game in the world. It is trending primarily because of the level of exposure it has reached. Some of the poker tournaments like the World Series of Poker (WSOP) and World Championship of Online Poker (WCOOP) and they all have a prize pool of more than two million dollars. This enormous prize pool attracted a lot of attention, which then invited a lot of players.

Due to the popularity of Texas Hold'em, poker AI has developed rapidly. However, it has not shown dominance yet unlike IBM's AI called Deep Blue and DeepMind AI AlphaGo. These AI have beaten chess world champion Gary Kasparov in 1997 and go world champion Lee Sedol in 2016, respectively. [29]. Texas Hold'em AI claims to beat the aforementioned professional players, and this will be illustrated in the following chapter.

Texas Hold'em AI has not dominated because of four reasons. First, it is a game with imperfect information, which means that one would never know what the opponent's cards are. Thus, the situation is made more complicated. Second, it must involve good poker players who have good opponent modeling ability so that they could judge if the opponents are bluffing. Third, it must involve good poker players who can use some deceptive strategy to bluff the opponents and win the game even when given poor hand cards. Fourth, it must involve outstanding poker players who have the ability to take the opponents' deception into account. This makes the game more difficult as it is.

Instead of developing a complete poker AI, we want to uncover the hidden rules behind poker players' bidding decisions and to understand the reasons behind such movements to make their movements explainable. Therefore we will adopt rule based algorithms to poker.

This paper introduces two different rule based learning algorithms, which are implemented in the two-player no-limit Texas Hold'em's first round in the pre-flop phase. In the first round of the pre-flop phase, the only information the players have is the two hand cards, and they have to make a decision (either call, raise or fold) based on it. This is the only phase that the community cards are not shown but players bid just like other phases, which means there are less factors that would influence a player's decision. However, it could still exemplify the key feature of uncertainty without being too complex. This situation is perfect for out study.

This paper aims to discover the hidden rules of poker bidding by studying the behavior of the professional players using different rule based learning algorithms. It also aims to make the players' bidding explainable, and to find out the suitable rule based models for building explicit models that have been validated by expert players.

The paper is divided into following parts: chapter one explains the rules of Texas Hold'em and introduces the development of poker AI. It especially introduces the DeepStack AI developed by University of Alberta, which was used for background knowledge data set. The last section of chapter one gives an overview of rule based learning used, namely, propositional learnining and probabilistic inductive logic programming (PILP). Chapter two discusses the implementation of the algorithms and the problems encountered during the process. After that, chapter three illustrates the results of the experiments and discusses how our project can be improved. Finally, chapter four concludes with our experiment and some perspectives on explainable AI.

2 Related Work

Due to its explanability, the rule based system has been introduced to Texas Hold'em for a long time. Back to 1990s, Sklansky and Malmuth has established an expert system[30] to evaluate the strength of the poker hands. This expert system has been widely used by the later poker AI like Loki[1] or Poki[8]. Later, Robert Follek has proposed Soarbot,[12], a rule based system for playing poker and instead of the intuitive rules, Soarbot would set a hierarchy of the rules and from high to low. Besides, unlike the typical rule-based systems, Soarbot has the ability to learn to new rules. But still, it is unable to beat the professional players and one of the reasons is that Soarbot does not know how to bluff. In the other field, for example Bridge, which is also a game with imperfect information, Swann Legras has experimented state of the art propositional machine learning and ILP systems on it and both systems have got a nice result.[18]

Inspired by the work above, we would then make our poker project. We have introduced PILP system to handle the uncertainty of the poker bidding, namely bluffing or other strategies. And inspired by the work by Swann, we will apply a proporsitional rule learning model and a PILP model to our project and compare them. Poker players uses a black-box approach to make decision, and what we are doing to try to take a look inside the black-box and make their moves explainable.

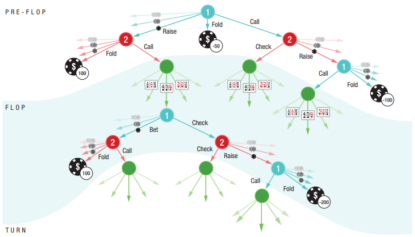
3 Texas Hold'em

3.1 Rule of Texas Hold'em

In this subsection we will introduce the rule of Texas Hold'em shortly so you may have a better understanding of it. Texas Hold'em is a popular poker game that can be played by 2 to 10 players. The ultimate goal of it is to either let all the opponents quit(fold cards) or beat the rest opponents in the final round by comparing the strength of the cards.

There will be maximal 5 phases in a game, which are pre-flop, flop , turn and river and players will make decisions in each phase. There are 3 options when making decisions: call, raise and fold. Call is for following, which means betting the same amount of chips as the previous players bid. Raise means betting more chips than the previous players and fold means throwing the cards and quit the game.

This is how the whole process looks like:



In the pre-flop phase, each player will receive 2 hand cards and will have to make decisions in the order of their sitting positions. Only when every remaining player calls will we come to the next phase and if there is only one player left, then he win the game automatically and will take all the chips in the pool. Then in the flop phase, 3 community cards will be dealt face up and players will make decisions again based on the new situation. In the turn phase, another new community card will be dealt face up and players bet again. Finally in the river phase, the last community cards will be dealt face up and players will bet for the last round. After that, all the remaining players will show up their hands and compare the strength of their cards. The player with the strongest strength win the game.

The strength of the cards is show below:

Figure 1: Poker hand rankings

3.2 Overview of the Texas Hold'em AI

In this section, we will take an overview of the poker AI. Poker is a game with imperfect information, which makes it fascinated. The founder of the game theory, Von Neumann is also a big fan of poker, he once said "Real life is not like that. Real life consists of bluffing, of little tactics of deception, of asking yourself what is the other man going to think I mean to do. And that is what games are about in my theory."

Poker has intrigued people a lot and the study of artificial intelligence of poker has been starting for a long time.

3.2.1 Orac, probably the first poker AI ever

The first poker AI may be Orac, which was developed by Mike Caro, a professional poker player and poker book writer, in 1980's. However, no official documentations or events can be found on it.

3.2.2 Loki, the first real poker AI

Then after several years of development, the University of Alberta has released Loki, which is considered to be the first Proper AI, in 1998. It is the first AI that is designed to play a full table poker, which means nine players. It successfully uses plausible opponent modeling to improve its play. Opponent modeling is that in the game of imperfect information, where we could not get all the information we want, we describe our opponent's behavior by building proper mathematical models. It has got a very nice result in the beginning, but later once the human professional poker players have adapted to its way of playing, they will detect the patterns and weakness of Loki and then alter their strategy. Overall Loki performs slightly poorer than the human professional players [1]

Loki in pre-flop phase Loki has stored several bidding strategy for the pre-flop phase using expert information. It will choose a bidding strategy based on its current chip amount, position and other factors. Several thresholds will be set and once the thresholds are reached, it will choose the corresponding strategies.

3.2.3 Poki, a better version of Loki

Four years later, the University of Alberta has released a progressed version of Loki, which is also based on opponent modeling, called Poki. Poki uses MixiMax method to search imperfect information game trees directly. Due to the fact that poker is a game with imperfect information, the best moves of the opponent could not be determined since we do not know their hand cards. To solve this problem, they use their knowledge of the opponents to choose their mixed strategy. Mix is used instead of Min. Also neural networks and decision trees are used for the opponent modeling. The neural networks contains a set of eighteen inputs corresponding to the properties of the poker such as card information, chips, positions of the opponent and the output layer consists of three nodes, which stands for fold, call and raise. [8]

Poki in pre-flop phase The bidding strategy of pre-flop is that it uses a simple expert system built on top of a set of machine-learned tables containing an expected income rate for each hand. Poki would then make decisions based on the income rate it has calculated.[8]

3.2.4 Polaris, an poker AI stored with different strategies

Then in the year of 2009, the University of Alberta has introduced its new AI, called Polaris. Polaris mainly consists of two parts: a number of different poker strategies and a

component that will select which specific strategy to play with the opponent intelligently. The component could identify which strategy suits the best for this opponent. It also only requires little computation power so it could be run simply on a normal laptop.

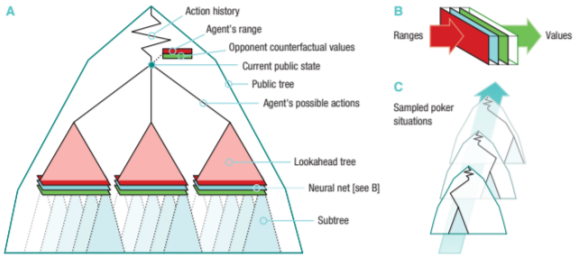
Polaris uses counterfactual regret minimization to solve abstract poker problems which may contain 10^12 states. [4]

3.2.5 DeepStack, the best poker AI and the one used for our research

DeepStack is a strong poker AI released by the University of Alberta in 2017. It is also considered as the best poker AI until now. The data we use for our study is also coming from the game sets DeepStack playing with other professional players. DeepStack has become the first poker AI ever to defeat with statistical significance professional poker players in heads-up no-limit Texas hold'em.

The DeepStack strategy consists of three parts: A sound local strategy to compute the current situation, A learned value function with limited depth, and a restricted set of looking ahead actions. Continual resolving has been used for the sound local strategy. It means rebuilding a target by re-solving every time when it needs to act; and never using the strategy beyond the next action. But continual resolving is still theoretical sound, not practical. To make it realizable, we will then have to limit the depth and breath of the re-solved subtree. The depth of the re-solved tree has been limited intuitively. DeepStack also uses the neural network to establish the value function for its depth-limited look ahead. It has used two networks, the first one would calculate the counterfactual value after the flop phase, the second one after the turn phase. Besides there will be an extra network for the pre-flop phase before the public cards are shown, which is used to accelerate the re-solving for early actions.

Figure 2: DeepStack Overview



DeepStack has achieved great accomplishment when playing with other professional players and has become the first poker AI ever to beat professional poker players. It has gotten this impressive result by solely little domain knowledge and without any expert human games, which makes it become a milestone of the poker artificial intelligence. [23]

3.2.6 Conclusion

Looking back the development of the poker artificial intelligence, from Orac to Loki, then to Poki, after that Polaris, and finally DeepStack; the playing style has changed from scanning physical cards to playing completely online; the algorithm has changed from the basic expert system to the pure math opponent modeling to the combination of newest neural network and Counterfactual Regret Minimization(CFR) algorithm; We have witnessed a huge progress of the poker AI. And i am confident that, the DeepStack AI is not an end

yet. There will be absolutely more and more exciting Poker AI or AI for imperfect games in the future, that can not only slightly beat the human professional players, but also surpass them greatly.

4 Rule based Algorithm

In this section we will have an overview of the rule based algorithm and will introduce the propositional rule induction, inductive rule induction and probabilistic inductive rule induction respectively.

Nowadays, when we talk about AI, we suddenly come up with some popular concept like deep learning, neural network. But in a lot of areas for example aerospace, military or finance, technology needs to be explainable, because of the security reasons, they need to know every detail of the model. And in the area of game with imperfect information, when we try to learn a game, we often take a look at the professional players' move. We want to know what will they act in this situation and want their move to be explainable. And that is exactly what rule based learning is about. Mathematical logic has a strong expressing ability. Most of the human knowledge could be described and expressed smoothly by it with a little adaption. An example is that "My father's father is my grandfather.". This knowledge is hard to be described by a mathematical function, but can be easily described by first order logic. Therefore, domain knowledge could be introduced naturally in the learning process. Another strength is that the abstract describing ability of the logic rules may show significant advantage when dealing with highly complicated mission. For example the QA System(Question Answering System). We may face the situation where there will be a lot of, or even countless answers and it would be of great convenience if we could describe it based on the logic rules.

Given a set of training samples, what the rule based learning do is to find classification rules to predict or to classify new samples. Normally it is shown in the form "IF...THEN... ", with several prerequisite in IF form and the prediction in THEN form. For example we want to know we will go skiing tomorrow. We then build a rule based model and if the model says "If weather is sunny, THEN go", it means that we would go skiing if tomorrow is sunny, which is very straight forward and intuitive. Unlike neural networks or support vector machine, who are considered as "black box", the rule based learning is more explainable and the user may have a deeper understanding on the way it makes decisions.

The advantage of rule based learning is obvious: explainable and rules have strong robustness regarding the correlations. In the meantime, straight-out "low performance" and "high complexity" were the main issues with probabilistic rule learning, while "uncertainty" is the big issue with classical rule learning.

Since the goal of the rule based learning is to find a set of rules that cover the samples as much as it could, the most intuitive way to do it is sequential covering: every time it learned a rule from the training set, the samples covered by that rule will be removed. And then repeat the process to the other training samples. Since it only deals with a part of the data, it is also called the "separate and conquer" strategy.

The logic rules can be classified into two parts: propositional rule and first order rule. In our study, we will use propositional rule induction and first order rule induction respectively and compare our result. We will also introduce these two logic in the following sections.

4.1 Propositional Rule Induction

Propositional logic To understand what propositional rule induction is we should know what propositional logic is.

The logic that deal with propositions is called *propositional logic*. It is the most ubiquitous and basic branch of mathematical logic. A proposition is a declarative sentence which is either true or false.

For instance there are three different sentences:
1) Angela Merkel is the chancellor of France.
2) Angela Merkel is the chancellor of Germany.
3) Who is the chancellor of Germany?

It is evident that 1) is false and 2) is true, and they all meet the requirement of a proposition which therefore makes them one. 3) is a question and it is not a proposition.

If a proposition is true, we would declare the truth value of this proposition is true and if a proposition is false, we would then say the truth value of this proposition is false.

Propositions which do not contain any of the logical operators or connectives (not, and, or,if-then, if-and-only-if) are called atomic propositions (or primitive/simple sentence). Statements which can be constructed by combining one or more atomic statements using connectives are called molecular or compound propositions (or compound sentences or simply sentences).[32]

A propositional rule used for classification is always represented as follows:

IF *CONDITIONS* THEN *C*

where C is one class of the target variable and Conditions is a conjunction of simple logic sentences. Therefore, a propositional rule is actually an implication *CONDITION* THEN *C* in propositional logic. [14]

Rule induction Rule induction is a method of machine learning using formal rules to represent the set of observations. The rules can either stand for a scientific model of the data or solely local patterns of the data. Major rule induction paradigms are:
•Propositional Rule Induction
•Association Rule Learning
•Decision Rule Algorithm(Decision Trees)
•Horn Clause Induction
•Inductive Logic Programming
•Probabilistic Inductive Logic Programming
In this paper, we will especially focus on propositional rule induction and probabilistic inductive logic programming, which we will illustrate later.

propositional rule induction It is obvious that propositional rule induction means using propositional formal rules to represent the set of observations. The most famous class learners in this area are the CN2 algorithm[6] ,RIPPER [5] and PRIM(learning rules from statistics literature)[13]. In this paper, we will use CN2 algorithm to fit our data and analyze the bidding strategy of DeepStack by looking at the propositional rules it learned, we also discuss the quality of the model, which we will explain later.

propositional rule induction and poker The reason why we use propositional rule learning to poker is that we would like to know the way DeepStack bid. So with propositional rule learning, we can get some intuitive rules, for example ”if two hand cards are in

the same suit then we raise", that would greatly help us understand the bidding strategy of the professional players.

4.2 ILP and PILP

4.2.1 Inductive Logic Programming(ILP)

Instead of using propositional logic, inductive logic programming, namely ILP, is using first order logic to represent the feature of the observations and hypothesis.[27]

First Order logic Comparing with declarative propositions, first order logic also deals with predicates and quantification. A predicate takes an entity or entities in the domain of discourse as input while outputs are either True or False. For example, we have two different sentences here:

1)Cristiano Ronaldo is a soccer player.

2)Lionel Messi is a soccer player.

In propositional logic, they are both true, but they are referring to two completely unrelated sentences. And we could see that sentence 1) and 2) both share a same predicate, which is "is a soccer player". And the variable is inistialized as "Cristiano Ronaldo" in the first sentence, and "Lionel Messi" in the second sentence. In first order logic, sentence 1) and 2) can be written as issoccerplayer(Cristiano Ronaldo) and issoccerplayer(Lionel Messi).

Comparing with propositional logic, first order logic additionally allows us to represent propositions as assertions of predicates about individuals or set of individuals. Another example has been shown here:

1) Tom comes from America.

2) every American speaks English.

If sentence 1) and 2) are both true, then as a causality we could infer that Tom speaks English. But in propositional logic, sentence 1) and 2) cannot lead to our conclusion because they are unrelated. In first order logic, propositions like "Tom comes from America" and "every American speaks English" may be represented by predicate argument representations such as $(\forall x, American(x) \rightarrow$ speaksEnglish(x)).

First order logic consists of syntax and semantics. A syntax in the FOL is a formal language representing concepts. The semantics of first order logic formula would explain how the truth value of any first order logic formula is determined.

First order logic is made up with an alphabet, a first order language, a set of axioms and a set of inference rules. The alphabet is made up of the following symbol classes:

•Constants: a, b, c, Tom, blue

•Variables: x, y, z

•Function symbols: American(Tom)

•Predicate symbols: bigger(4,3), speaklanguage(Tom, English).

•Connectives: AND, OR, IMPLIES

•Quantifiers - $\forall$, $\exists$

•Punctuation symbols: ')', '(' [7]

Logic programming Logic programming is a sister approach of functional programming based on formal logic. It consist of a set of logic sentences, expressions facts and rules in this specific area. The rules are displayed in following form:

H :- B1, . . . , Bn.

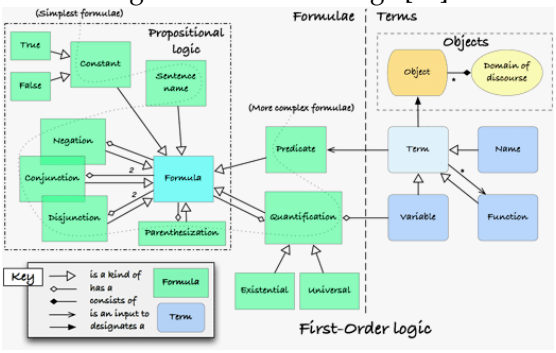
H represents the conclusion, namely the *THEN ....* part, while *B1...Bn* are set of prerequisites, namely the *If...* part. For example:

mother(X.Y):- parent(X,Y), female(X).

It means that X is Y's mother when X is Y's parent and X is female, which is quite understandable.

Logic programming can also be viewed as controlled deduction. This is because a vital concept in logic programming is to divide the programs into logic component and control component. In the context of the pure logic programming languages, the logic component

Figure 3: First order logic[19]



will decide the produced solution alone. The control component can determine how the program is executed. This feature is represented by:

Algorithm = Logic + Control,

in which "Logic" means a logic program and "Control" stands for various theorem-proving strategies. [17]

Inductive logic programming Based on logic programming, inductive logic programming is concerned with generalizing positive and negative examples. An ILP system will try to prove every positive examples and in the meantime to reject every negative examples. To be more concrete, ILP systems will output the description of the predicates from examples and background knowledge in a logic program style.

Popular ILP systems include Aleph[31], Claudien[9], FOIL[26], Progol[24]....

4.2.2 Probabilistic Inductive Logic Programming(PILP)

Definition According to Luc De Raedt, a PILP system can be defined as follows[10] :

Given:

a set of Example E,

A background knowledge B,

A probabilistic covers relation P(e| H, B)

A probabilistic logic programming representation language

Find:

The hypothesis H* = argmaxH P(E| H, B)

There will be two different sub tasks:

Parameter learning: The logic program structure has already been given, and the task is to estimate the parameter that would maximize the hypothesis.

Structure learning: The logic program is also not given, and will be learned by the data. In our project, we are dealing with a structure learning problem.

ILP and PILP Just as the name itself described, probabilistic inductive logic programming, namely PILP, is based on inductive logic programming, with extra probabilistic adding on it. Compared with inductive logic programming, there are two basic difference:

1.clauses are annotated with probabilistic values

2.Covers relation has also become probabilistic[10].

Besides PILP also belongs to Statistical Relational Learning.

Statistical Relational Learning is a sub-area of machine learning especially dealing with uncertainty and complex and relational structure.

The knowledge in a SRL system will be represented as follows:

Figure 4: Relationship between ILP and PILP

the relational properties will be presented by first order logics(FOL)
the uncertainties will be represented by probabilistic graphic models
Or probabilistic Inductive Logic Programming (PILP) So an ubiquitous SRL system may consist of
either 1. Probability + Logic.
Representatives are Probabilistic Relational Models(PRM)[16], Bayesian Logic Program(BLP)[15], Markov Logic Network(MLN)...
or 2. Logic + Probability, namely PILP.

The picture below shows how the examples and models are represented as well as the learning strategy of different SRL methods.

Figure 5: Learning strategy of different SRL methods

	Example representation	Learning model representation	Structure learning	Parameter learning
STATISTICAL LEARNING	table representations (propositional)	Probability	NO	YES
DEEP LEARNING	table representations in hierachical views	Probability	NO	YES
Markov Logic Network (MLN)	Symbolic	Symbolic Rules	NO* (limitations)	YES
Relational Depency Networks (RDN)	Symbolic & Integrates BK	Symbolic Rules	YES* (The type os the structure is fixed)	YES
PILP (KLog /KLogNLP)	Symbolic	Probability	NO	YES
nFOIL, QuickFOIL, ProbFOIL (De Raedt Group)	Symbolic	Symbolic Rules	FOIL: YES (just propositional symbolic rules. ProbFOIL is relational.	YES

PILP and poker The reason why we want to apply PILP methods to our poker project is that Texas Hold'em is not only a game with imperfect information, but also the top professional players would prone to do some bluffing frequently to help them win the chips even when they have weak hand cards. Therefore it would be great to find a way to represent the bluffing moves and the probability suits it perfectly. So we are confident that the PILP may have a better performance comparing with the other system. The result would be like for example "if both hand cards are having the same suit, then we have 70 percent possibility to raise, 20 percent possibility to fold and 10 percent possibility to call."

5 Implementation

In this chapter we will implement our project. We will first give an overview of the whole project and after that we will elaborate it step by step.

Overview

Our project is aimed to explore the bidding strategy of the professional Texas Hold'em player. So in order to do that, we would first find adequate data. After searching and comparing, we have then made the decision to use the data of DeepStack AI[23] as our research target. The data contains the information that DeepStack AI playing with other professional poker players. And then we would apply different rule based learning strategy to the data and to evaluate the result. We first implement propositional rule learning, in which we have used CN2 Rule Induction as our learning algorithm and Orange as our implementation platform. After that we would introduce PILP system to adopt to our data and the algorithm we use here is slipcover[3]. At last we will evaluate the result of two different types of rule based learning and have a summary.

5.1 data source

We have got the data from the official website of DeepStack Ai[23]. In the download area, there are a bunch of hand histories. The data we have used is ”DeepStack vs IFP Pros”, where IFP Pros stands for the professional players of International Federation of Poker. In this file, it contains hand information about DeepStack AI playing with more than 30 professional players and has been sorted into more 30 folder by different players.

The data set is presented as follows:

Figure 6: data set

id	AIVAT	Chips	All.Hands.Chips	Chance.Correction	Action.Correction	Your.Hand	Deepstack.Hand	Flop.Cards	Turn.Card	River.Card	Position	Pre.flop	Flop	Turn	River	Seconds.to.Act	Total.Seconds	Start.Time..UTC.
1	-6.059428	50	50	131.338834	-75.279406	QcKc	4d2h				bb	f				0	0.0336380004882812	2016/11/14 13:51:51
2	-2.120394	-100	-100	-42.463235	-55.416371	4d5s	7hTh				sb	cR300f				15.9376509189606	15.9685559272766	2016/11/14 13:52:22
3	-559.046552	750	750	201.977199	1107.069353	Ah9c	7d3d	8s4c8h			bb	r300R750c	R1550f			24.0305519104004	33.0627369880676	2016/11/14 13:52:43
4	4.545439	-50	-50	-54.545439	0	8c4d	2h3s				sb	f				5.88705396652222	5.88705396652222	2016/11/14 13:53:20
5	29.759668	500	500	496.454332	-26.214	8c5h	9s6c	8sKh9c	8d	4c	bb	cC	R200c	R500c	R1150f	41.295666217804	53.9480469226837	2016/11/14 13:53:29
6	9.604111	100	100	-30.583609	120.979498	6dKc	Tc4s				sb	r300F				6.51934003829956	6.550188064575 2	2016/11/14 13:54:28
7	-16.291435	-100	-100	-56.016774	-27.691791	2h5c	Th7h				bb	r300F				6.02929306030273	6.07038307189941	2016/11/14 13:54:41
8	-28.406391	-300	-300	-1.607224	-269.986385	Kc9h	8s8c				sb	r300R900f				25.5235748291016	25.5543529987335	2016/11/14 13:54:51
9	32.471278	300	300	189.483209	78.045513	QsTc	KdJs	6c7sQc	8d		bb	r300C	Cc	R750f		23.0435559749603	35.6094660758972	2016/11/14 13:55:22
10	-106.375013	250	250	15.415894	340.959119	6cKs	3s4s	TdJcTc	Jh		sb	r250C	Cc	Cr470F		18.3556797504425	34.1101596355438	2016/11/14 13:56:02
11	-9.882816	-300	-300	-155.337523	-134.77966	2hKs	9h8c	TdQh7c	5c		bb	cC	R300c	Cr600F		25.6110401153564	36.2100210189819	2016/11/14 13:56:38
12	-188.497257	500	446.504688	354.449332	280.552613	Qs5h	Kh7c	5s7s3d	Qd	9d	sb	cC	Cr200C	Cr500C	Cc	40.6516621112823	72.1798717975616	2016/11/14 13:57:20
13	-45.519158	-100	-100	-38.710949	-15.769893	4hKc	7sKd				bb	r200F				14.5605239868164	14.6023800373077	2016/11/14 13:58:37
14	-101.394052	200	200	-113.596217	414.990268	9c8h	Ks7h	3c2h4d	Td		sb	r200C	Cc	Cr450F		15.0396518707275	32.4336230754852	2016/11/14 13:58:56
15	-18.990386	-100	-100	-50.038767	-30.970847	5sTh	Ks9s				bb	r200F				6.37480092048645	6.41688585281372	2016/11/14 13:59:32

Here are the meaning of these attributes:

AIVAT: how many of DeepStack’s chips the participant was expected to take in this hand, adjusted for luck. It is used for evaluating each hand but irrelevant to our research.

Chips: how many of DeepStack’s chips the participant took in this hand.

All Hand Chips: how many of DeepStack’s chips the participant took, playing against DeepStack’s range. This value looks at all possible hands DeepStack could have held, and how likely they were.

Chance Correction: an estimate of how lucky the participant’s cards and the board cards were, in chips.

Action Correction: an estimate of how lucky DeepStack’s actions were for the participant, in chips.

Your.Hand and DeepStack.Hand: The rank and suit information of the two hand cards of the player and DeepStack AI. for example ’4d2h’ means that the first hand card has a rank of 4 and a suit of diamond and the second hand card has a rank of 2 and a suit of heart.

Position The position of the two players, bb for big blind and sb for small blind. In the pre-flop phase, sb will make decision first.

Pre.flop This contains the acting information of two players. for example, ’cR300f’ means that the small blind player called, the big blind player raised 300 chips and then the small blind player folded.

Other attributes are auxiliary and are irrelevant to our research.

Since we are focusing on the first action of DeepStack AI in the pre-flop phase, we are especially interested in the following attributes:

Position: the DeepStack AI will move first, so position of DeepStack AI should be sb(small blind).

DeepStack.Hand: In the preflop phase, two hand cards are already known,and this would be the key for DeepStack AI to make decisions.

Pre.flop: We are especially focusing on the first action of DeepStack AI in the pre-flop phase, we would like to know under what situation it will call, raise or fold. So the first action in attribute Pre.Flop would be our target variable.

Chip: The amount of current chips is supposed to be a factor that would influence the judgement of DeepStack, but the attribute ”Chips” and ”All Hand Chips” here do not stand

for the amount of current chips so the chip information is missing here. Therefore chips will not be taken into account in this study.

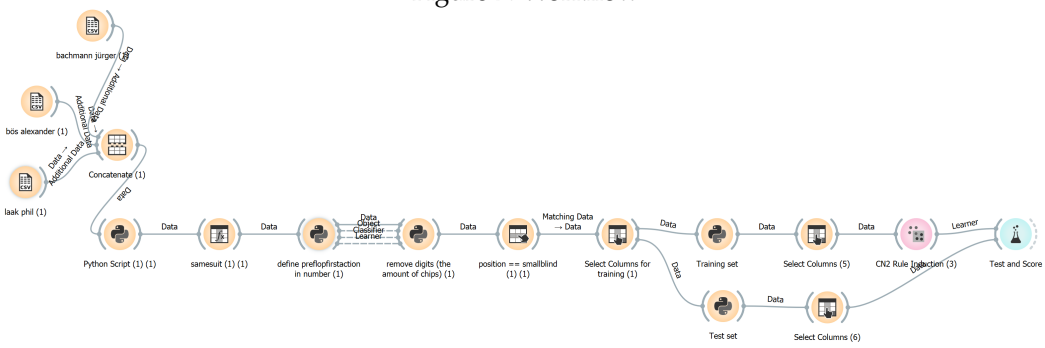
5.2 Implementation of the propositional rule learning

Having found the data set, we would then apply the data set to propositional rule learning. All of the following moves are implemented via Orange[11].

Orange is a popular platform for data mining and data visualization. It is very convenient because instead of programming, orange will allow you to concentrate on the exploratory data analysis with its friendly graphic user interface and very straightforward work flow. Its essential data processing unit is called widget, and each widget will implement one job such as data cleaning, data visualization, analysis. Widgets can be connected together and data can be sent and received through the linked widgets. Besides programming knowledge is not compulsive, everything could be done by mouse-clicking. However, if you have special need, you can still write some python script to realize your ideas.

Here it is what our project workflow looks like:

Figure 7: Workflow



Laak Phil, Bachmann jürger and Bös Alexander are all professional poker players. The three csv files here contain the hand information when they play with DeepStack AI. The reason why we have chosen three professional players instead of one is that: 1) Our data set will be larger 2) Observing the data DeepStack AI playing with various players may reflect the bidding strategy better. Both may increase our accuracy significantly.

The whole data set has been divided into two parts: training data and test data.

Before using Orange, we have already created two new attributes ”handcard1” and ”handcard2”, which contain the rank information of the two hand cards.

The ”Concatenate” widget will help concatenate the three csv files into one file.

The ”Python Script” widget helps create a new attribute: preflopfirstaction, in Which it has documented how will the players act in the first round of preflop phase and another 4 new attribute bigger1T, smaller1T, bigger2T, smaller2T, which means if the first and the second hand card are bigger or smaller than 10, since the rank information is critical for players to make decisions.

The ”samesuit” widget creates another new attribute samesuit, which may judge whether the two hand cards are in the same suit. When two hand cards are in the same suit, it increases the possibility that the player has a flush, which means increasing the strength of the cards. And because every other situation is unknown(other 5 community cards). Every suit in that time has the same strength. Therefore it is unnecessary to tell specifically which suit the hand card exactly has and so we have used the attribute ”samesuit” as the representation of the suit information.

The ”define preflopfirstaction in number” widget will classify the elements in preflopfirstaction attribute in number, where ”r(raise)” stands for 2, ”c(call)” stands for 1 and ”f(fold)” stands for 0.

The ”remove digits” widget will remove the amount of chips in the attribute Pre.Flop. This would make the bidding information of two players in the pre-flop phase more clear to observe.

The ”position == smallblind” widget will set the position of DeepStack AI to small blind, which means the DeepStack AI will take action first in the pre-flop phase and this is exactly what we are studying.

The ”select columns for training” widget will select the suitable column for our training process. The selected columns are ”handcard1”, ”handcard2” and ”samesuit”. And the target variable is ”preflopfirstaction in number”

The ”Training set” widget and the ”Test set” widget have divided the whole data set into two parts with 1:1 ratio, which means 50% data for training, 50% data for testing.

The ”CN2 Rule Induction” widget may apply CN2 rule induction algorithm[6] to our data set.

Another CSV file is our test data.

The ”Test and Score” widget may evaluate our trained model.

The ”CN2 Rule Viewer” will inform us the rules CN2 rule induction algorithm has learnt and gave us the rules intuitively.

CN2 Rule Induction is based on the ideas of the AQ[35] and ID3[25] algorithms. It uses entropy as its search heuristic. The CN2 algorithm can be divided into two parts: A searching algorithm using beam search and a control algorithm to implement the search repeatedly. The figure below shows the pseudo code of the CN2 algorithm.

Figure 8: CN2 algorithm[6]

```
Let: E be a set of training examples;

Procedure CN2(E) returning RULE_LIST:

let RULE_LIST be the empty list;
repeat
    let BEST_CPX be Find_Best_Complex(E);
    if BEST_CPX is not nil then
        Let E' be the examples covered by BEST_CPX;
        Remove from E the examples E' covered by BEST_CPX;
        Let C be the most common class of examples in E';
        Add the rule 'if BEST_CPX then class=C' to the end of RULE_LIST,
until BEST_CPX is nil or E is empty.
return RULE_LIST.

Procedure Find_Best_Complex(E) returning BEST_CPX:
let the set STAR contain only the empty complex;
let BEST_CPX be nil;
let SELECTORS be the set of all possible selectors;
while STAR is not empty,
    specialize all complexes in STAR as follows:
    let NEWSTAR be the set {x ∧ y|x ∈ STAR, y ∈ SELECTORS};
    Remove all complexes in NEWSTAR that are either in STAR (i.e., the
        unspecialized ones) or are null (e.g. big = y ∧ big = n)
    for every complex Ci in NEWSTAR:
        if Ci is statistically significant when tested on E and better than
            BEST_CPX according to user-defined criteria when tested on E,
        then replace the current value of BEST_CPX by Ci;
    repeat remove worst complexes from NEWSTAR
        until size of NEWSTAR is ≤ user-defined maximum;
    let STAR be NEWSTAR;
return BEST_CPX.
```

Result We will first take a look at the result here. The picture below shows the rules CN2 algorithm has learnt

Figure 9: Top six rules CN2 has learned

0	handcard2=A AND handcard1=5	→	preflopfirstactioninnumber=2	[0, 0, 6]	11 : 11 : 78	-0.00	2
1	handcard2=A AND handcard1=6	→	preflopfirstactioninnumber=2	[0.0, 0.0, 6.0]	11 : 11 : 78	-0.00	2
2	handcard2=A AND handcard1=7	→	preflopfirstactioninnumber=2	[0.0, 0.0, 5.0]	12 : 12 : 75	-0.00	2
3	handcard2=A AND handcard1=8	→	preflopfirstactioninnumber=2	[0.0, 0.0, 3.0]	17 : 17 : 67	-0.00	2
4	handcard2=A AND handcard1=A	→	preflopfirstactioninnumber=2	[0.0, 0.0, 4.0]	14 : 14 : 71	-0.00	2
5	handcard2=A AND handcard1=T	→	preflopfirstactioninnumber=2	[0.0, 0.0, 5.0]	12 : 12 : 75	-0.00	2

These are the top six rules that CN2 has learnt. For example, the first rule is "IF the rank of the first hand card is A and the rank of the second rank card is 5, THEN the preflop-firstactioninnumber is 2, which means raise. From the rules we could see that if DeepStack AI has a high rank hand card, it is likely to raise, which is quite reasonable, because when the rank of our hand card is big, then we are likely to have a temporary leading position and raising would be a good choice.

Evaluation The figure below here shows the evaluation of our model.

Figure 10: Evaluation

Model	AUC	CA	F1	Precision	Recall
CN2 rule inducer	0.723	0.710	0.662	0.686	0.710

We could see that the AUC value of our model is 0.723. The AUC value means Area Under the Curve(here ROC curve) and indicates how much our model is capable of distinguishing between classes. Here an AUC value of 0.723 means that it means there is a 72.3% chance that the model will be able to distinguish between positive class and negative class, which is not a bad result. The CA value, which means classification accuracy, also lie in 0.71. It represents the (TP + TN)/(TP + TN + FP + FN), where TP stands for the number of true positive examples, TN the number of true negative examples, FP the number of false positive examples and FN number of false negative examples, and refers to the ratio of correctly predicted observation to the total observations. The recall value also lies above 0.70. It means TP/(TP+FN) and indicates the ratio of correctly predicted positive observations to the all observations in actual class. Since our test set comprises half of the total data set, it is large enough and also different from our training set. Therefore we believe that the metrics and meaningful and we would later make a comparison with PILP system.

5.3 PILP

Having applied propositional rule learning to our poker data set, we then apply probabilistic inductive logic programming to our data set again.

Structure learning and parameter learning In the statistical machine learning literature, scientists have made a difference between "parameter learning": which stands for learning the prior and conditional probability distributions, and "structure learning", which stands for learning the nodes and edges of a graph or the whole structure [22]. In our case, we want to know both how the rules look like and how the probability of the rules are distributed, so we are going to pick a structure learning algorithm.

5.3.1 Introduction of the prior knowledge

The algorithm we are using here is Slipcover[3], the programming language we are using is Prolog[21] and the platform we are using are Swish[33] and Swi-prolog[34].

Prolog is a logic programming language that uses first order logic as its base and it is considered as a declarative language, since the programming logic is formulated on terms of relations which stand for background knowledge facts and the rules. To compute the program, we run a query over these relations[20] The rule will be represented like this in Prolog:

Head :- Body

It means that if Body is true, then Head is True. The Body is composed of calls to predicates, which are called the rule's goal. The logical operator like conjunctions or disjunctions can also be in the Body part when necessary. But they are not allowed to be in the Head part. The fact will be represented as clauses with empty bodies. An example would be:

animal(tiger).

Which means tiger is an animal, and it is also equivalent to

animal(tiger) : true

Prolog and our poker project: When implementing PILP programs, the poker information will be constituted in prolog style. For example: handcard1(id,rank) to represent the rank information of the first hand card in a specific game and samesuit(id) will represent if the two hand cards are in a same suit.

Swi-prolog, developed by Jan Wielemakerwith and with its name deriving from Sociaal-Wetenschappelijke Informatica ("Social Science Informatics"), is an implementation of the logic programming language Prolog.

Here are some of its key features:

• Fast compilation.

• Robust and free memory leaks

• Small but scales well for large applications.

Besides, Swi-prolog is also widely used for educational purposes due to the fact that it is free and portable, and also because it is compatible with the textbooks and its easy-to-use environment.

Swi-prolog and our poker project: We will use swi-prolog to compile our data set in the prolog style using its package "cplint" and get a decent result.

Slipcover is the PILP algorithm we are going to use. It is the abbreviation of "Structure Learning of Probabilistic logic programs by searching OVER the clause space". In the field of probabilistic clauses, Just like CN2 algorithm, Slipcover also adopts the idea of beam search, and in the space of theories, slipcover has performed greedy search. Unlike CN2 algorithm which uses entropy as its search heuristic, slipcover here uses log-likelihood. In order to calculate the estimated log-likelihood, Slipcover uses Expectation Maximization with EMBLEM.[2].

At first, Slipcover will search all space of clauses and classify them into two parts: one is the clauses for target predicates, the other one is the clauses for the background knowledge with a discriminative approach. Then the search will begin with the "bottom clauses", just like Progol did[24], and using log-likelihood as its selecting threshold. Then greedy search is used to apply to the space of theories, by adding each clause for a target predicate to the temporary theory. In the end it applies paremeter learning with EMBLEM[2] to the best target theory plus the clauses for the background predicates.[3]

Figure 11: Pseudo code of slipcover[3]

Algorithm 4 Function SLIPCOVER

```
1: function SLIPCOVER(NInt, NS, NA, NI, NV, NB, NTC, NBC, D, NEM, ε, δ)
2:     IBs =INITIALBEAMS(NInt, NS, NA)                          ▷ Clause search
3:     TC ← []
4:     BC ← []
5:     for all (PredSpec, Beam) ∈ IBs do
6:         Steps ← 1
7:         NewBeam ← []
8:         repeat
9:             while Beam is not empty do
10:                Remove the first couple ((Cl, Literals), LL) from Beam   ▷ Remove the first clause
11:                Refs ←CLAUSEREFINEMENTS((Cl, Literals), NV)   ▷ Find all refinements Refs of
   (Cl, Literals) with at most NV variables
12:                for all (Cl′, Literals′) ∈ Refs do
13:                    (LL″, {Cl″}) ←EMBLEM({Cl′}, D, NEM, ε, δ)
14:                    NewBeam ←INSERT((Cl″, Literals′), LL″, NewBeam, NB)
15:                    if Cl″ is range restricted then
16:                        if Cl″ has a target predicate in the head then
17:                            TC ←INSERT((Cl″, Literals′), LL″, TC, NTC)
18:                        else
19:                            BC ←INSERT((Cl″, Literals′), LL″, BC, NBC)
20:                        end if
21:                    end if
22:                end for
23:            end while
24:            Beam ← NewBeam
25:            Steps ← Steps + 1
26:        until Steps > NI
27:    end for
28:    Th ← ∅, ThLL ← −∞                                        ▷ Theory search
29:    repeat
30:        Remove the first couple (Cl, LL) from TC
31:        (LL′, Th′) ←EMBLEM(Th ∪ {Cl}, D, NEM, ε, δ)
32:        if LL′ > ThLL then
33:            Th ← Th′, ThLL ← LL′
34:        end if
35:    until TC is empty
36:    Th ← Th ⋃_{(Cl,LL)∈BC} {Cl}
37:    (LL, Th) ←EMBLEM(Th, D, NEM, ε, δ)
38:    return Th
39: end function
```

5.3.2 implementation

To implement our idea, we will first convert our data into Prolog format.

Attributes we have now are *handcard1, handcard2, samesuit, bigger1T, smaller1T, bigger2T, smaller2T* and we will convert them one by one.

handcard1 and handcard2 The attributes handcard1 and handcard2 represents the rank information of the first and the second hand card. In every set of game, there will be different hand cards. So the handcard1 has been converted as *handcard1(id,rank), handcard2(id, rank)*, where "id" means the game set id and "rank" means the rank of the represented card.

samesuit, bigger1T, smaller1T, bigger2T, smaller2T These attributes can be represented as

samesuit(id), bigger1T(id)..., where id stands for the game set id and samesuit(1) means that the two hand cards are in the same suit when in the game id 1.

preflopfirstaction This is our target predicate and can be represented as bid(id,r), bid(id,f), bid(id, c), where id stands for the game set id and r,f,c are all constants representing raise, fold and call.

Settings We have set the verbosity to 1, number of iterations of beam search to 10, number of mega-examples on which to build the bottom clauses to 15, maximum number of theory search iterations to 50.

Background knowledge We could add the rule
bigger1T(id) :- \+ smaller1T(id).
bigger2T(id) :- \+ smaller2T(id).
to the background knowledge. Thus would facilitate our work and if we only put the attribute *smaller1T* into the Prolog file, Slipcover would automatically define the predicate *bigger1T* based on the rules we have provided.

Mode declaration Mode declaration will specify a legal form for calling a predicate. There are three simple mode types.(a) +*T* specifying that when a literal with predicate symbol p appears in a hypothesised clause, the corresponding argument should be an "input" variable of type T; (b) -*T* specifying that the argument is an "output" variable of type T; or (c) *T* specifying that it should be a constant of type T.

What is worth being mentioned is how we define the mode of the rank information, which are *handcard1* and *handcard2*. The mode of these two attributes has been defined as:
modeb(*, rank1(+id, -rank)).
modeb(*, rank1(+id, -♯rank)).
We have used two mode declaration here for one predicate, this is because 1) we want to know how certain rank of the card may influence the result. For example, if player has a A, the biggest hand card, he may take some special moves. That is why we use "-♯". 2) on the other hand, we don't want to let the rank information of the hand card to be the key factor that may influence player's action and try to be as objective as possible, so we add a "-".

Representation of the negative examples Unlike typical tasks learning a binary predicate, since we have three bidding options(raise, call or fold), we are learning a ternary predicate. The negative examples can be expressed as neg(bid(id,A)) and neg(bid(id,B)), where A and B stands for the other two bidding strategies except the one in the positive examples. For instance, if *bid(1,r)* is in the positive example lists, then *bid(1, f)* and *bid(1,c)* will be shown in the negative example lists automatically in order to let Slipcover know that the three options are mutually exclusive.

Traning set and test set Again we have divided the whole data set into two parts: 50% for training data and 50% for test data in order to exclude the possibility that the algorithm just learned a case-by-case classification.

5.3.3 Result and Evaluation

Learned Rules A lot of rules have been learnt by Slipcover. Below are some selected rendered ones:

bid(A,r) :0.0522 :-
rank1(A, B), samesuit(A),rank2(A,C).

bid(A,f) :0.012 :-
smaller2T(A).

bid(A,c) :0.065 :-
rank1(A,j), smaller2T(A)

For example, the last selected rules said that if the rank of the first hand card is j and the second hand card is smaller than ten, then there will be around 6 percent possibility that player will call. The single possibility is a little bit low, but the final decision is made, according to the work by Fabrizio Riguzzi, by performing inference, which means computing all explanations, then building a Binary Decision Diagrams(BDD) and then applying dynamic programming[28]

Evaluation In our Slipcover model, due to its uncertainty, the classification accuracy could not be measured.

We have got an AUCROC value of 0.785, which means that there is a 78.5% chance that the model will be able to distinguish between positive and negative classes and if we compared it with the AUCROC value of our CN2 Rule Induction model (0.723), we have achieved an improvement of 9%, which is a great improvement. Based on the fact that the test set is coming from half of the data set, which means it is large enough and also different from the training set, we believe this result is meaningful.

6 Discussion

If we look at the rules the propositional rule learning model has learned, we could see that the DeepStack AI would tend to raise when the rank of the hand cards are high (high card), when the two hand cards are in a same suit (potential flush) and also when the two hand cards have same rank(pair). It is very reasonable and match the strength of the hand cards that Sklansky has proposed in 1990[30]. The propositional rule learning model has showed the rules out and has made the DeepStack AI's move more expainable.

Since the AUCROC result of PILP model has a 9 percent improvement comparing with propositional rule model, we could conclude that our PILP model performs better than propositional rule model in the poker project, which also implies that the rules with probabilities are more persuasive than the determinant rules in this case.

From that we could indicate that the DeepStack AI prones to make moves with uncertainty, instead of following the "if...then..." rule, sometimes it tends to take unexpected action, which is also called bluffing in Texas Hold'em.

For example, if the DeepStack AI has a hand card of 2 in spade and another hand card of 7 in heart, both two hand cards have a very low rank and they are neither two consecutive cards nor in same suit. Normally we, as normal player, would call or fold since we have two pretty "bad" hand cards. But the DeepStack AI and other professional players may make bluffing moves to let their opponents think that they have strong hands to "terrify" their opponents to help them win the prize pool. Under our PILP model, the DeepStack AI may raise with a certain probability, which perfectly matches this bluffing phenomenon. Therefore the betterness of PILP model proves that the DeepStack AI and the professional players would make bluffing moves when bidding and the rules it returned has made the move of DeepStack more explainable.

On the other hand, because of the fact that bluffing can be used in every phase of Texas Hold'em, we could also learn that PILP model is more suitable than propositional rule learning model when dealing with Texas Hold'em because it could learn the rules with probabilities and probabilistic rules could represent the bluffing move well.

7 Future work

In this section we will discuss how our project can be improved and be implemented further in the future.

In our project we did not put chips into our input variables as the amount of chips would influence the players judgement, because our data source does not provide us the information of the current amount of chips. We believe our model would be improved if adding the chip information.

This paper suggests that the professional players may take bluffing strategy even in the first round of the pre-flop phase. This assumption may be also addressed in the future studies, we could continue using PILP model to explore the bidding strategy of the professional players in other phases for example to explore in which phases they would be more likely to do bluffing, in which phases they would tend to play conservatively.

Except for that, we could also try to apply the propositional machine learning systems, ILP or PILP system to other game with imperfect information like Bridge or Omaha, which also needs the player's move to be explainable. Swann has applied the ILP system to Bridge and has got a very impressive result.[18] We believe that rule based learning may have a good performance when applied to a game that needs explainability.

8 Conclusion

The aim of this paper is to build an explainable AI using rule based learning to try to understand and explain the bidding strategy of the top professional Texas Hold'em players in the first round of the pre-flop phase. To realize that we have built two models, one is a propositional rule learning model using the CN2 Rule Induction algorithm, and the other one is a probabilistic inductive logic programming model using the Slipcover algorithm. We have applied our model to the DeepStack AI since DeepStack AI has beaten many professional players. Both models have achieved good result and have properly explained the bidding strategy of the poker players. Since the PILP model performs better we can also conclude that poker players may tend to make bluffing strategy even in the first round of the pre-flop phase. We believe that the rule based learning has explained the bidding strategy of the poker players well and we are confident that rule based learning may play an important role where explainability is needed.

Bibliography

[1] *Aaai '98/iaai '98: Proceedings of the fifteenth national/tenth conference on artificial intelligence/innovative applications of artificial intelligence*, USA, American Association for Artificial Intelligence, 1998.

[2] Elena Bellodi and Fabrizio Riguzzi, *Em over binary decision diagrams for probabilistic logic programs*, Intell. Data Anal. **17** (2013).

[3] ELENA BELLODI and FABRIZIO RIGUZZI, *Structure learning of probabilistic logic programs by searching the clause space*, Theory and Practice of Logic Programming **15** (2015), no. 2, 169–212.

[4] Michael Bowling, Nicholas Risk, Nolan Bard, Darse Billings, Neil Burch, Joshua Davidson, John Hawkin, Robert Holte, Michael Johanson, Morgan Kan, Bryce Paradis, Jonathan Schaeffer, David Schnizlein, Duane Szafron, Kevin Waugh, and Martin Zinkevich, *A demonstration of the polaris poker system.*, 01 2009, pp. 1391–1392.

[5] Jaime Carbonell, *Machine learning research*, ACM SIGART Bulletin (1981), 29–29.

[6] Peter Clark and Robin Boswell, *Rule induction with cn2: Some recent improvements*, Proc. European Working Session on Learning (1998).

[7] Imperial College, *Logic programming*, 2017.

[8] Aaron Davidson, Duane Szafron, Robert Holte, and Witold Pedrycz, *Opponent modeling in poker: Learning and acting in a hostile and uncertain environment*, (2002).

[9] Luc De Raedt and Luc Dehaspe, *Clausal discovery*, Machine Learning **26** (1997), 99–146.

[10] Luc De Raedt and Kristian Kersting, *Probabilistic inductive logic programming*, Lecture Notes in Computer Science (2004).

[11] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan, *Orange: Data mining toolbox in python*, Journal of Machine Learning Research **14** (2013), 2349–2353.

[12] R.I. Follek, *Soarbot: A rule-based system for playing poker*, Pace University, 2003.

[13] Jerome Friedman and Nicholas Fisher, *Bump hunting in high-dimensional data*, Statistics and Computing **9** (1999).

[14] Johannes Frnkranz, Dragan Gamberger, and Nada Lavrac, *Foundations of rule learning*, Springer Publishing Company, Incorporated, 2012.

[15] Kristian Kersting and Luc De Raedt, *Bayesian logic programs*, Tech. report, 2001.

[16] Daphne Koller, *Probabilistic relational models*, Proceedings of the 9th International Workshop on Inductive Logic Programming (Berlin, Heidelberg), ILP '99, Springer-Verlag, 1999, p. 3–13.

[17] Robert Kowalski, *Algorithm = logic + control*, Commun. ACM **22** (1979), no. 7, 424–436.

[18] Swann Legras, Céline Rouveirol, and Véronique Ventos, *The game of bridge: A challenge for ilp*, Inductive Logic Programming (Cham) (Fabrizio Riguzzi, Elena Bellodi, and Riccardo Zese, eds.), Springer International Publishing, 2018, pp. 72–87.

[19] J. W. Lloyd, *Foundations of logic programming*, Springer-Verlag, Berlin, Heidelberg, 1984.

[20] ———, *Foundations of logic programming*, Springer-Verlag, Berlin, Heidelberg, 1984.

[21] John Malpas, *Prolog: A relational language and its applications*, Prentice-Hall, Inc., USA, 1987.

[22] Melchi M. Michel and Robert A. Jacobs, *Parameter learning but not structure learning: A Bayesian network model of constraints on early perceptual learning*, Journal of Vision **7** (2007), no. 1, 4–4.

[23] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling, *Deepstack: Expert-level artificial intelligence in no-limit poker*, Science **356** (2017).

[24] Stephen Muggleton, *Inverse entailment and progol*, New Generation Computing, Special issue on Inductive Logic Programming **13** (1995), 245–286.

[25] J. R. Quinlan, *Induction of decision trees*, Mach. Learn. **1** (1986), no. 1, 81–106.

[26] ———, *Learning logical definitions from relations*, Mach. Learn. **5** (1990), no. 3, 239–266.

[27] Luc De Raedt, *Inductive logic programming*, pp. 529–537, Springer US, Boston, MA, 2010.

[28] Fabrizio Riguzzi and Terrance Swift, *Well-definedness and efficient inference for probabilistic logic programming under the distribution semantics*, 2011.

[29] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, *Mastering the game of go with deep neural networks and tree search*, Nature **529** (2016), 484–489.

[30] David Sklansky and Mason Malmuth, *Hold'em poker for advanced players*, 1999.

[31] A. Srinivasan, *The Aleph Manual*, 2001, http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/.

[32] Bhowmik Subrata, *Propositional logic*, 09 2017.

[33] Jan Wielemaker, Torbjörn Lager, and Fabrizio Riguzzi, *SWISH: swi-prolog for sharing*, CoRR **abs/1511.00915** (2015).

[34] Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager, *SWI-Prolog*, Theory and Practice of Logic Programming **12** (2012), no. 1-2, 67–96.

[35] Janusz Wojtusiak, *Aq learning*, pp. 298–300, Springer US, Boston, MA, 2012.