Testing of several function approximations for a partially observable card game

Jana Cathrin Backhus[1,a)] Hidetoshi Nonaka[1,b)] Takeshi Yoshikawa[1,c)] Masanori Sugimoto[1,d)]

Abstract: Game playing is considered as an important research topic in the field of Artificial Intelligence, because its features resemble those of real world problems. Research aims contain creating strong computer players for the entertainment of humans and creating human-like learning algorithms. Wizard is a partially observable multi-player card game that is structured in rounds with different grades of information imperfectness. A computer player using reinforcement learning algorithm is considered for the game. Function approximation plays an important role in reinforcement learning. In our research, we tested different function approximation approaches for the game Wizard under the aspect of having different grades of information imperfectness in the game rounds. Experimental results show that the success of the function approximations is differing a lot for the game rounds as the game winning rate of the learning agent is differing between 0% and 26.74%.

1. Introduction

Games have been platforms of interest since the first days of computer research. One of the aims of game research is creating strong computer players for the entertainment of human players. Another aim is to create human-like learning algorithms that can be generalized and applied to a broad number of applications. This makes games an important research topic in the field of artificial intelligence. Earlier objects of interest were fully observable games like chess or checkers. It was expected that creating strong computer players for these games would make it possible to create computers that can act like humans. But it was possible to create strong computer players without imitating human-like thinking, so more complicated games like Shogi, Go or card games became the new objects of interest.

Card games are considered complicated, because they are only partially observable and most of them are played with multiple players, which provide new challenges to the researchers. The partial observability of the game state makes it necessary to handle unknown environment states and multiple players make it impossible to use effectual search algorithms, which are often used in applications for two-player games. So other methods have to be investigated.

One possible approach is the introduction of reinforcement learning (RL) to the card game problem. Reinforcement learning is a machine learning framework that can make an agent learn a policy for a known or unknown environment via trial-and-error interactions. Many of the traditional reinforcement learning algorithms have been designed for problems with small and finite state and action spaces that are computationally tractable. However, reinforcement learning is often used in applications with realistic decision-making situations, which have a large or continuous state and/or action spaces. Since for this kind of problems, learning of an exact policy is infeasible, approximation methods have to be used. One problem is the choice of the function approximation method, because it is not known yet which type of function approximator works best for which kind of problem [9].

There is a broad number of applications of reinforcement learning algorithms to games and for some of them the strongest computer players are using reinforcement learning approaches. Because calculating optimal decisions in games is intractable in most cases, all algorithms must make some assumptions and approximations. For most of the card games, information is incomplete and a way has to be found to deal with it. An approach to deal with partially observability of card games directly is formulating the problem as a partially observable Markov decision process [5].

In this paper, the focus is laid on state evaluation via function approximation for a partially observable multi-player card game. As a sample application, the partially observable card game Wizard is considered. Wizard will be described as a partially observable Markov decision process. The main focus is laid on state evaluation for the bidding phase of the game. In the here introduced approach for Wizard, a learning agent (LA) is able to learn a strategy for the card game and in one of the tested approaches the LA is able to outperform its rule-based opponent players after 5000 training games by winning 26.74% of the games.

This paper proceeds as follows: In section 2 information about partially observable multi-player card games are given. In section 3 the implemented method is introduced and in section 4 experiments are presented. Finally, a conclusion follows in section 5.

[1] Hokkaido University, Kita-ku, Sapporo 060–0814, Japan
[a)] jana@main.ist.hokudai.ac.jp
[b)] nonaka@main.ist.hokudai.ac.jp
[c)] yosikawa@main.ist.hokudai.ac.jp
[d)] sugi@ist.hokudai.ac.jp

2. Partially Observable Multi-Player Card Games

2.1 Characteristics of Card Games

Most of the card games have properties, like a multi-agent setting and unobservable information, which make them an interesting and challenging application domain for RL. A multi-agent setting consists of three or more players that interact with each other during the game. Normally there is unobservable information about the current environment state, because hand cards of the opponent players and cards in the card deck can not be seen. These properties are similar to the properties of real world domains, which makes them a well-defined test-bed for realistic decision making problems.

2.2 Card Game Terms

Below two card game terms that are used in the rest of the paper are explained.

- **Hand** The term *hand* is referring to the hand cards distributed to every player at the beginning of the game. When it is said that several hands are played, it means that several rounds are played, where the player receives a new hand every time.
- **Trick-Taking** A *trick* is a finite unit in play, where each player has to play one card into this unit. When the card game's play of a hand centers on a series of tricks, then this is called *trick-taking*.

2.3 Related Research

After expanding the game research interest to card games, many card games have become the objects of research. One of the first successful application of machine learning algorithms to card games was Ginsberg's Intelligent Bridgeplayer (GIB) [1]. Ginsberg's major innovation is the introduction of partition search for a perfect information version of the game. Monte Carlo simulation is used to make the partition search applicable in a realistic game situation, where unseen cards are sampled. This kind of search algorithm is also called Perfect Information Monte Carlo (PIMC) search. For the bidding, the program is referring to an enormous database of hand-crafted rules.

Other card games that became the objects of interest are Skat ([3]), Hearts ([8], [2]) and Spades ([7]). In Skat, the focus of research centers more on approaches for the bidding phase of the game since it is said to be the real weakness of computer card players. Hearts is an object of interest since it is a completely competitive game without any cooperations between players. Spades is also completely competitive in the 3 player version and also has a bidding phase while Hearts has none. In the related research on Spades, the importance of opponent modeling in multi-player games is emphasized.

2.4 Wizard

Wizard is a competitive card game that can be played with three to six players. The card deck consists of 60 cards, 52 basic cards and eight special cards (4 wizard, 4 jester).

The game is divided into rounds. Every game round has the same game flow, but a different number of cards is dealt to the players in every game round. The number of dealt cards per player is increased from round to round, starting with one card in the first round and finishing with all cards being dealt to the players in the last round. The number of rounds is therefore determined by the number of players. Except for the last round, a trump color is determined from the cards that were not dealt to the players by opening the top card of the remaining card deck. In the last round, there is no trump color.

The game flow of a game round can be divided into two phases and after those two phases an evaluation of the game round result will take place. The game flow is also presented in Figure 1.

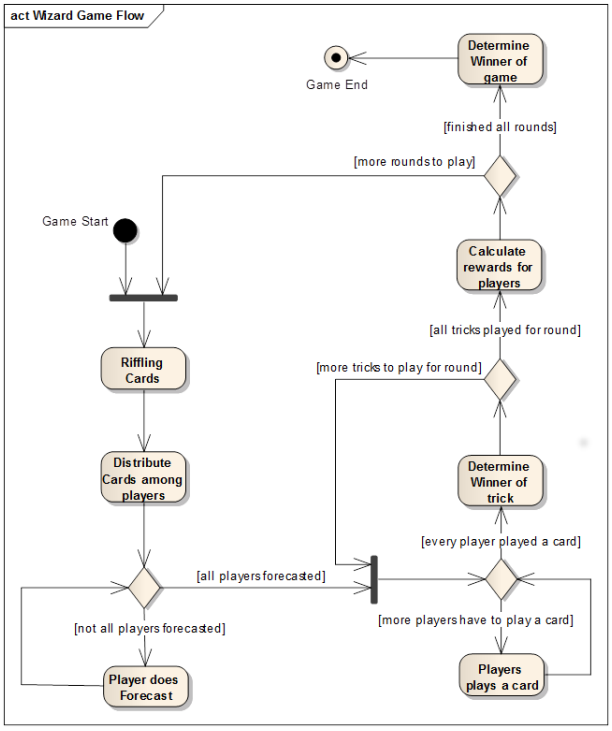

Fig. 1 Wizard game flow

2.4.1 The Bidding Phase

The task is to predict the number of tricks that will be winnable based on the own hand cards. The players bid one after another in the playing order for the round.

2.4.2 The Trick-Taking Phase

Trick-taking takes place as explained in Chapter 2.2 and the winner of every trick is decided.

2.4.3 Result Evaluation

Every player has to count the number of tricks, he is able to win and then compares it with his bid. If the number of won tricks equals his bid, reward is received, otherwise penalty is received.

3. Implemented Method

The sample application Wizard is divided into two modules for the two phases of the game: bidding and trick-taking.

3.1 Bidding Phase

In the bidding phase of the game, the player needs to evaluate information from the environment to decide on a bid that he is

going to announce as his goal for the trick-taking phase of this game round. Therefore a game state evaluator is introduced for the problem. The bid is then made based on the game state evaluation. To take the ignorance of the player about his environment in account, evaluation is conducted based only on the available information. No further assumptions about non-visible parts of the game state are made.

For the game state evaluation a function approximator is trained in an on-line matter with reinforcement learning. The input data for the function approximator are information about the hand cards of the player. The output is a 1-dimensional vector that is used as the bid. As training labels for the output, the actual number of won tricks for a game round is used.

Several function approximators are considered as possible state evaluators for the bidding phase. From the actual input data a set of features is extracted and then used as input for the function approximator. Four function approximation methods are tested: Logistic Regression, Multi-Layer-Perceptron, Radial Basis Function Network and Normalized Gaussian network.

3.1.1 Logistic Regression

Logistic regression (LR) is a parametric model with a linear classificator which passes its linear function through a threshold function. The hard nature of the threshold can be an issue in linear classifiers. Therefore the LR uses a logistic function to soften the threshold. The logistic function is a common sigmoid function. The linear function contains weights as model parameters, which are adjusted during the training process. The weights are fitted to minimize the loss on the training data set [5].

3.1.2 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a kind of neural network, that uses the supervised backpropagation technique for training the network [4]. The network is trained to map input data onto an appropriate set of output data.

The network consists of an input layer, an output layer and one or more hidden layers in-between. Every layer has neurons, where the input layer and output layer have as much as needed to map the input and output vector dimensions. The hidden layers can have any appropriate number of neurons that fits the learning problem. Each layer is fully connected to the neighbor layers. In other words, every neuron of a layer is connected to all neurons in the neighbor layers. Except for the input layer, every neuron has a nonlinear activation function (e.g. sigmoid function), that maps the input to weighted outputs in each neuron. The nonlinear activation function ensures that a nonlinear function can be represented. The calculation of an output for an input is straightforward, while the training of the network is carried out through backpropagation starting in the output layer.

3.1.3 Radial Basis Function Networks

Radial Basis Function (RBF) network is a kind of neural network with a single hidden layer. The hidden layer has a nonlinear radial basis activation function, which is commonly presented by a Gaussian function. The mapping between the input and hidden layer is nonlinear, while the mapping from hidden to output layer is linear. Generally, the RBF network is trained in two phases. In the first phase, the parameters of the Gaussian kernels, mean and variance, are trained by an unsupervised clustering approach. Then in a second phase the relative weights of each Gaussian are trained to determine the influence of every Gaussian kernel. The second phase consists of a system of linear equations trained by supervised data [5].

3.1.4 Normalized Gaussian Network

The normalized Gaussian network (NGnet) is a function approximator that softly partitions the input space by normalized Gaussian functions and each local unit linearly approximates the output within the partition. Therefore the NGnet is a local model. Several implementations can be found for the NGnet, introducing off-line as well as on-line training approaches. In [6] an on-line trained approach is introduced that will be applied in this application. It is stated that the NGnet makes the learning process easier compared to global models like the MLP. Because of its local nature, it is possible to change parameters of several units in order to learn a single datum.

3.2 Trick-Taking Phase

Fujita et al. [2] introduced an RL algorithm for large-scale multi-agent environments with partial observability that describes the problem as a partially observable Markov decision process. The algorithm calculates an utility for every possible action in a current state and chooses the action with the best utility as the next action to take. Approximation methods are necessary, since due to the partial observability of the problems the state space is too large to be computed in the whole. The introduced algorithm uses particle filtering for sampling possible current and next time step's states. The state samples are then used to calculate the utility of an action at a time step. Figure 2 presents a pseudo code for the calculation of this utility.

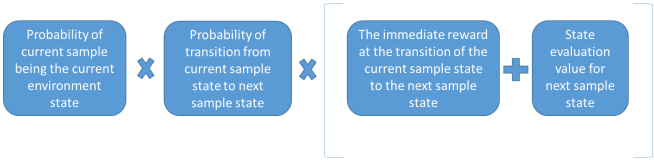

Fig. 2 Pseudo code for utility calculation of an agent action

The probabilities of the current sample state and the next sample state are calculated to consider the likelihoods of these sampled states under the available information. Further approximation is introduced by a state evaluator for the next time step sample state that is presented by a trained function approximator. The immediate reward is referring to the reward one would receive in case of transition from the current sample state to the next sample state.

The described method is adapted for the trick-taking phase of Wizard.

4. Experiments

4.1 Experimental Environment

Before the experiments can be carried out, some general decisions have to be made. Wizard can be played with three to six players, but for the experiments here, only the four player's variant is evaluated. The game environment consists out of one learning agent and three opponent players (see Chapter 4.1.1).

For every type of experiment several test runs were conducted. So the results displayed in section 4.2 are all averaged results of several test runs.

Every test run is divided into evaluation episodes and training episodes. For and after every training episode an evaluation episode was conducted. So here for example the number of training episodes is 10 and the number of evaluation episodes is 11. Every training episode consists of 500 training games, where the cards are randomly distributed for each game. The seating order is also randomly appointed for every game. In the evaluation games, 100 deals are played in four different seating orders. So there is a total of 400 games, where every player has once the first, second, third and last seat. For better comparison the card distributions stay the same for all evaluation episodes in the same test run.

The learning agent is trained on-line. Training was conducted in two ways. Since Wizard is separable into rounds which all have different grades of information incompleteness, training can be conducted separately for every round as well as for all rounds together. In the first case, each round has its own function approximator (hereafter also called SEP). In the second case, one function approximator is trained for all rounds together (hereafter also called TOG). All our function approximators are tested in both ways. For the RBF approach, variances and means of the radial basis function are set randomly at the beginning to enable on-line training. Additionally for comparison, a rule-based bidding approach is also tested.

4.1.1 Opponent Players

A rule-based player is constructed by creating rules for a bidding and a trick-taking module. The player is choosing its actions in the bidding and trick-taking phases according to these definite rules. Every opponent player is represented by one clone of the rule-based player in the following experiments.

4.2 Experiments

In the first two experiments, the different function approximator types are tested for the round seperated (Figure 3) and all together (Figure 4) training methods. As the results show, the learning agent is not able to beat the three opponent rule-based players since his game winning rate is much lower than that of his opponents. Since this is a four player game, a winning rate of 0.25 for all players would mean that all players win equally often. All function approximator approaches as well as the rule-based bidding show rather bad results, but some performance differences are visible between the approaches.

Figure 3 shows the results where function approximators were trained separately for each round. In the SEP approach LR has the best performance results with a winning rate around 20%, which is a good result compared to the performance of the other function approximators that all have winning rates under 10%.

Figure 4 shows the results where one function approximator was trained for all rounds together. In the TOG approach MLP performs best out of the function approximator types with a winning rate near to 20%. Surprising is the result for the LR approach. While it was performing best in the SEP trained approach, it is performing worst in the TOG approach with a winning rate that is almost 0%. The winning rate for RBF is similar in both approaches. NGNet shows big performance differences since it has 0% winning rate in the SEP approach while it is able to reach more than 15% winning rate in the TOG approach. Therefore it is showing a performance gap between the two approaches that is similar to the one of the LR function approximator.

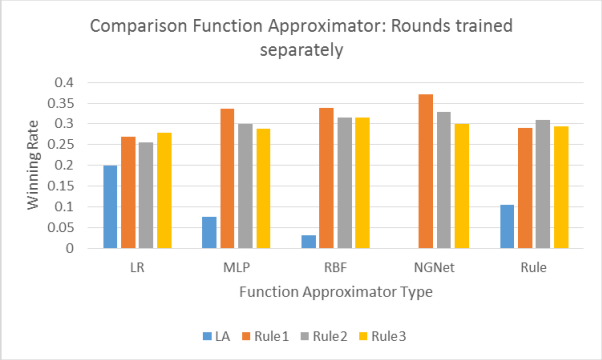

Fig. 3 Comparing different function approximators (SEP)

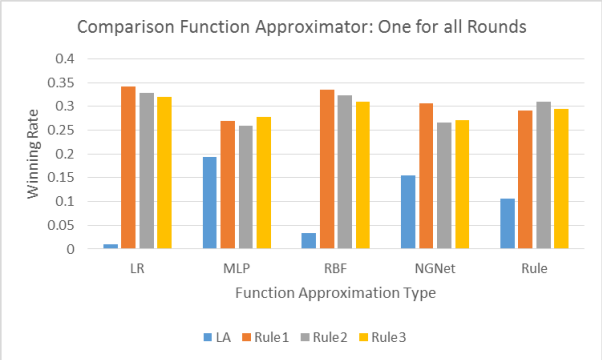

Fig. 4 Comparing different function approximators (TOG)

Since Wizard is a round-based game, where the rounds have different sized hands and therefore different grades of information imperfection, it is very likely that strategies have to change for each round. A look into the winning rate of each round for the two experiments above shows that the success is differing a lot for each round. Figure 5 shows the results for the round seperated training method while figure 6 shows the results for the all together training method. The figures' result presentation is reduced to the results of the learning agent, where the winning rate of the learning agent is displayed for the four function approximators as well as the rule-based approach for each of the 15 rounds in a four player game.

Figure 5 shows the results for the SEP approach. The performance results for the NGNet is 0% for the game as a whole, but splitting the winning rates for every round shows that the NGNet is not necessarily performing that bad in all rounds, but is equally successful as LR in the last third of the game. But since for the whole game, all round results are counted together, it is not able to make up for the bad performance in the former two thirds of the game. Therefore a solution is needed to improve the performance for the rounds in the first two thirds.

Figure 6 represents the results for the TOG approach. In the TOG approach. the LR approach performs very bad in total. A

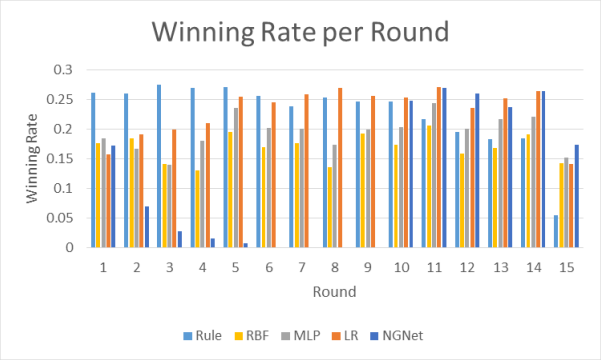

Fig. 5 Comparing different function approximators round-based (SEP)

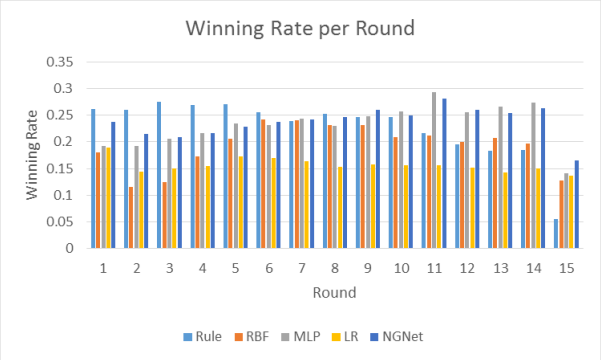

Fig. 6 Comparing different function approximators round-based (TOG)

look into the winning rate of the rounds separately shows that the LR approach has a winning rate of around 15% for almost all of the game rounds. But in total, this seems to be not enough to win the game, therefore the total game performance is low. Improvement possibilities need to be considered. Also it should be remarked that the reason for the bad performance differs from that of the SEP trained NGNet.

Comparing the results of the function approximator with the rule-based bidding approach shows that the rule-based bidding approach is more successful in the earlier stage of the game while the function approximators are more successful in the later stage of the game. This applies for both learning approaches. Therefore a mixed approach of rule-based bidding and function approximation may be likely to improve the performance of the learning agent and is tested in the following. The round-based results are used to decide, which of the rounds should use a rule-based bidding approach instead of a function approximator. Table 1 presents the round numbers until which the function approximator approaches will use rule-based bidding instead of the approximator. In the TOG trained approach, the function approximator is still trained over all of the rounds, even when the rounds use rule-based bidding.

Table 1 Division of rounds for mixed approach

Function Approximator	SEP	TOG
MLP	10	8
LR	6	(14)
RBF	10	10
NGNet	9	6

Figure 7 shows the results for the mixed approach in the case

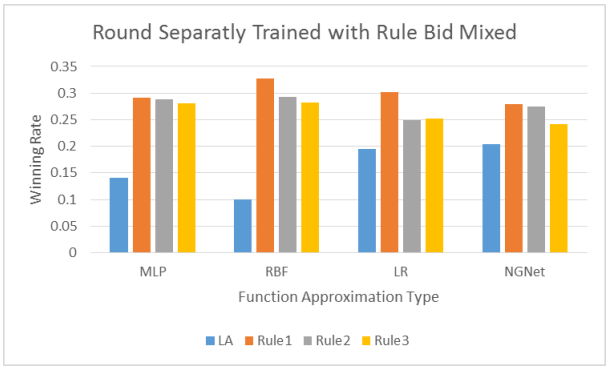

Fig. 7 Comparing different function approximators with partly rule bidding (SEP)

of SEP training. In three of four cases the function approximator were able to improve their performances. Only in case of the LR function approximator, the performance stayed almost the same. Since this function approximator only mixed in rule-based bidding until round 6, the performance may not have improved enough to have an influence on the total game result. For the MLP and RBF approach, performance improvements around 7% were possible. The biggest improvement was achieved for the NGNet approach. This may be related to the bad round performance that was eliminated by the introduction of rule-based bidding for those rounds. In the mixed approach NGNet is even performing slightly better than the LR approach and therefore becomes the best performing mixed approach for the SEP trained approaches. But in none of the function approximator approaches, the learning agent is able to outperform the opponent players.

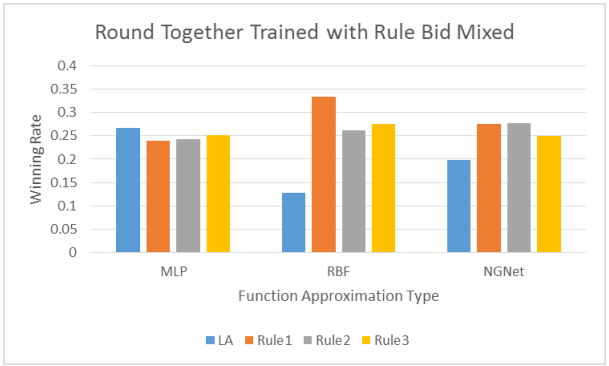

Fig. 8 Comparing different function approximators with partly rule bidding (TOG)

Figure 8 shows the mixed approach results for the TOG approach. Here the number of tested function approximators was reduced to three, because the LR approach is performing worse than the rule-based approach in 14 out of 15 rounds. Therefore a mixed approach would be almost equal to using the rule-based approach. The three tested function approximators were all able to improve their performance around least 5%. The performance ranking stays the same for the function approximators and the MLP performs best. The MLP approach is even able to outperform his opponents by achieving the highest winning rate.

An additional experiment was done for the all in one trained MLP bid approach mixed with rule bidding. As figure 6 shows, the difference between performance of rule based bidding and

MLP bidding is very small for round 9 and 10. In fact the MLP approach performs slightly better, therefore a combination of rule bidding until round 8 and MLP for all of the other rounds was chosen in the former mixed approach experiment (Figure 8). In Figure 9, three different combinations of the rule and MLP approach were tested, where rule bidding is used until round 8 (MLP_R08), round 9 (MLP_09) and round 10 (MLP_10).

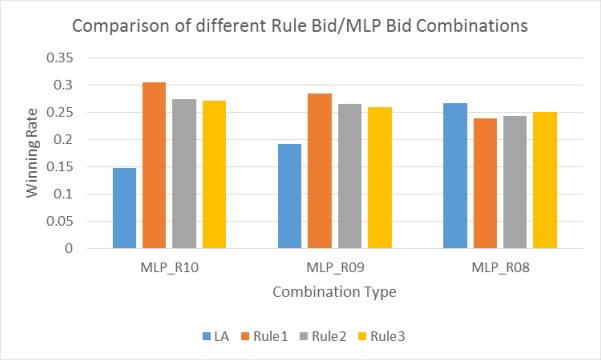

Fig. 9 Comparing different MLP/Rule Bid Combinations

The results show that the performance of the learning agent is increasing with reduced number of rule based bidding rounds. There are big performance differences although the performance of the rule based bidding was likely to perform equally well. It is expected that the increasing success of the play due to rule bidding in the earlier stage of the game has an influence on the all over all performance since even in the mixed approach the MLP is trained for all rounds. This is likely to be related to the trade off between bidding and play, because the play phase is depending directly on the bidding phase. In the play phase the play is adjusted to the actual bid to try the best to fulfill the made bid. In the TOG approach, learning success seems not to be related only to the success in the rounds separately, but successful playing in one round can have an impact on the learned function approximator and therefore on the success in other rounds.

5. Conclusion

In this paper, a reinforcement learning approach was introduced to a partially observable multi-player card game. Since the calculation of an exact solution for realistic problems in reinforcement learning is infeasible, approximation methods have to be used. Function approximation methods can be used to evaluate game situations and are introduced here for a bidding evaluation module for the partially observable card game Wizard. Many function approximation methods are existing, but there is not enough knowledge about which function approximation method works best for a certain problem. Therefore several function approximators are tested for the card game Wizard. The card game Wizard is a round-based game, where the grade of information imperfectness is changing due to a changing number of hand cards per round. The function approximators are therefore tested under these aspect for two different learning approaches.

Great performance differences can be seen in the function approximators on the game level as well as the game round level. None of the function approximator approaches was able to beat the rule-based opponents in the game independently. But it was possible to improve the performance of the bidding module by the partly introduction of rule-based bidding. Almost all function approximators were then able to improve their performance and in one case the learning agent was even able to beat his opponents by having a slightly higher game winning rate than all his opponents with 26.74%.

This time, rule-based bidding was introduced to improve the performance of the learning agent. But for future work, it would be more interesting to consider a function approximator that is able to perform well on its own and that can adjust to the different game round situation by itself.

References

[1] Ginsberg, M. L. (2001) *GIB: Imperfect Information in a Computationally Challenging Game*, Journal of Artificial Intelligence Research, Vol. 14, pp. 303–358.
[2] Fujita, H. and Ishii, S. (2007) *Model-Based Reinforcement Learning for Partially Observable Games with Sampling-Based State Estimation*, Neural Computation, Vol. 19, pp. 3051–3087.
[3] Keller, T. and Kupferschmid, S. (2008) *Automatic Bidding for the Game of Skat*, Lecture Notes in Computer Science, Vol. 5243, Springer Berlin Heidelberg, pp. 95–102.
[4] LISA lab (2014) *Multilayer Perceptron*, Deep Learning [online], `http://deeplearning.net/tutorial/mlp.html` [accessed 05 Feb 2014].
[5] Russell, S. J. and Norvig, B. (2010) *Artificial Intelligence - A Modern Approach*, Third Edition, Pearson Education, Inc., Upper Saddle River, New Jersey.
[6] Sato, M. and Ishii, S. (2000) *On-line EM Algorithm for the Normalized Gaussian Network*, Neural Computation, Vol. 12, Issue 2, pp. 407–432.
[7] Sturtevant, N. R., Zinkevich, M., and Bowling, M. (2006) *Prob-Max^n: Playing N-Player Games with Opponent Models*, Proceedings of the National Conference on Artificial Intelligence (AAAI).
[8] Sturtevant, N. R. and White, A. M. (2007) *Feature Construction for Reinforcement Learning in Hearts*, Lecture Notes in Computer Science, Vol. 4630, Springer Berlin Heidelberg, pp. 122–134.
[9] Wiering, M. and van Otterlo, M. (2012) *Conclusions, Future Directions and Outlook*, Wiering, M. and van Otterlo, M. (Eds.): Reinforcement Learning, ALO 12, Springer Berlin Heidelberg, pp. 613–630.