HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN
Fakultät für Elektrotechnik, Informatik und Mathematik
Heinz Nixdorf Institut und Institut für Informatik
Intelligente Systeme und maschinelles Lernen
Warburger Straße 100
33098 Paderborn

The Game: Face to Face

Seminar Report Summary

Submitted to Intelligent Systems and Machine Learning Group
in Partial Fulfillment of the Requirements for the
Seminar

"Machine Learning"
by
Avishek Mishra

Prof. Dr. Eyke Hüllermeier
Supervisor:
Alexander Tornede

Paderborn, March 5, 2020

1 Introduction

The present report discusses the rule-based AI player for The Game: Face to Face, which is a game played between two players, each player has either a 60 silver cards or 60 gold cards of the same shading with numbers 1 to 60 on them.(total 120 cards). Each player lay down the row cards(1-59 and 60-2) in front of them in table. Over the span of the game the number cards are laid to the right of these cards.

Thoroughly, both player shuffle there 58 cards and then each player picks up the first 6 cards from their own respective pile, which make up his hand cards. The game is played in turns and the rule is that you have to discard at least two or more cards from your hand(descending or ascending pile of your own or opponent's based on the game rules) and then pick up cards from your own draw pile to end your turn. In this way, the game is played until one player has no more cards in their hand. This player will be the winner.[1]

Determining a good heuristic for an AI player in The Game: Face to Face implies that the AI agent should know all the rules of the game and based upon that, it performs its decision making. If one or more rule is missing, the AI agent cannot make a better decision towards the goal.

Furthermore, the goal behind the course was not just to make an AI player which can win, yet in addition to show how it performs in various circumstances, e.g., If a rules is missing, then how accurate an AI player is playing against opponents random player. If AI player wins how many times it wins/ loses. Ideally, this analysis will reveal a path for making the AI better, i.e., improving its performance and rules. Finally, we built up a rule-based AI player which gets features about the game state and dependent on it, it performs its moves by keeping to the game guidelines.[1]

1.1 Motivation

The motivation behind this project was to produce a AI Player which can play "The Game: Face 2 Face". The AI player should have complete knowledge of the current game state and based on the best strategy / tactics it should play the game solely based on the game rules.

2 Detailed description of the approach

Most of General Game Playing involve finding a heuristics for evaluating the game state after making a possible moves. A popular method to find these heuristics is using machine learning. However for your approach we used rule-base AI player. The main reason to used rule-based AI player is as it is easy to implement, game is not too complex, opponent is a random player which should be easy to beat. We decide to make our rule-based AI Player based on two specific rules as follows:

- Rule 1: Help the Opponent player, but help as little as possible.
- Rule 2: Place cards on our own pile, hurt ourself as little as possible.

The reason behind to chose these two rule was to make an AI player that can play game more efficiently and follow the specify rules of the game manual.

Referring to Rule 1, here we are forming a rule for AI player to place at most one card on each round on opponent's player pile and the card that to be laid down on opponent's player pile must improve his pile (In this case AI player should follow only one rule , that is AI Player should place smaller card on Ascending pile and place a higher card on descending pile,). The benefit of this rule is that AI player can pick up as many cards from own draw pile until it has 6 cards in his hand again and it wouldn't be penalized with having to add 2 new cards to our hand cards.

The Algorithm for the Rule 1 is describe below, here AI player is placing "at most" one card in each round. For placement of cards on ascending pile of opponent's we see AI player checks whether its current hand cards is less than Top Card of Opponents Ascending Discard Pile and it should be greater than the minimum Helping Ascending Card (i.e initially minimum Helping Ascending Card is assigned with -1 value). Here, our goal is look for minimum or least helping card to place on ascending pile.

Similarly, we can see for placement of card on descending pile of opponent's we check reverse condition that is current hand card should be greater than Top Card of Opponents Descending Discard Pile and less than minimum helping descending card(i.e initially minimum Helping descending Card variable is assigned with 61 value) here our objective is looking for higher no card to be placed.

In this way by checking all game rules final placements of cards on opponent's ascending and descending pile is done.

Algorithm 1 Rule 1: Help the Opponent player, but help as little as possible

```
1: for all card in HandCards() do                          ▷ Check Opponent Ascending Pile
2:     if card ≤ TopCardOppAscDiscardPile() && card ≥minHelpCardAscNum then
3:         minHelpCardAscNum = card.getNumber();
4:         minHelpCardAsc = card ;
5:     end if
6:     if card ≥ TopCardOppDescDiscardPile() && card ≤minHelpCardDescNum
   then
7:         minHelpCardDescNum = card.getNumber(); ▷ Check Opponent Descending
   Pile
8:         minHelpCardDesc = card ;
9:     end if
10: end for
11: if (TopCardOwnAscDiscardPile() - minHelpCardAscNum) ≤ (minHelpCardDesc-
   Num - TopCardOppDescDiscardPile()) && (minHelpCardAscNum !=null) then
        Placement of cards on opponent's ascending pile is done.
12: else if (minHelpCardDescNum != null) then  Placement of cards on opponent's
   descending pile is done.
```

Referring to Rule 2, here we are placing cards on our own pile. here we are forming a rule for AI player to place cards on its own pile and the card that to be laid down must be greater than the card that was laid before it. For example 10, 20, 30, 40 for ascending pile and when placing for descending pile the card number should be smaller then the one laid down before it. The benefit of this rule is that AI player can place cards in its own pile with minimum hurting itself(placement of cards as per game rules not randomly).

The Algorithm for the Rule 2 is describe below, here AI player is placing cards on ascending discard pile of its own. For placement of cards on ascending pile of its own's, AI player checks whether its current hand cards is greater than Top Card of Own Ascending Discard Pile and we are also checking for the exception rule of game for own ascending pile (placing a card that is exactly 10 less than the number showing on the pile in a Or condition) and current card should lesser than the minimum Hurting Ascending Card (initially minimum Hurting Ascending Card is assigned with value 61, because we are looking for smaller card). Here, our goal is to look for minimum or least hurting card to our self to place on ascending pile.

Similarly, we can see for placement of card on own descending pile .we check current hand card should be less than Top Card of Own Descending Discard Pile, Or condition we checked the difference between current cards and Top Card of Own Descending Discard Pile should have a difference of 10 and current card should be grater than minimum hurting descending card(i.e initially minimum hurting descending Card variable is assigned with -1 value) here our objective is looking for a card with less hurting to be placement in descending pile.

In this way by checking all game rules final placements of cards on own ascending and

descending pile is done.

Algorithm 2 Rule 2: Place on own pile, hurt ourself as little as possible:

```
1: for all card in HandCards() do                          ▷ check Own Ascending Pile
2:     if (card ≥ TopCardOwnAscDiscardPile()||TopCardOwnAscDiscardPile()- card )
   && card ≤minHurtCardAscNum then
3:         minHurtCardAscNum = card.getNumber();
4:         minHurtCardAsc = card ;
5:     end if
6:     if (card ≤ TopCardDescDiscardPile()||card -TopCardOwnDescDiscardPile())
   && card ≥minHurtCardDescNum then
7:         minHurtCardDescNum = card.getNumber();  ▷ Check Own Descending Pile
8:         minHurtCardDesc = card ;
9:     end if
10: end for
11: if (minHurtCardAscNum - TopCardOwnAscDiscardPile()) ≤ (TopCardOwnDe-
   scDiscardPile() - minHurtCardDescNum) && (minHurtCardAscNum !=null) then
   Placement of cards on own ascending pile is done.
12: else if (minHurtCardDescNum != null)
13: then Placement of cards on own descending pile is done.
```

2.1 Strengths

Firstly, rules-based system use 'if-then' rules to derive actions. so, the implementation is easy, assuming we already know a very good strategy.

Secondly, the advantages of rules-based system are that they are easy to understand and can be made to represent expert judgement on simple or complicated subjects. Despite the fact that the 'if-then' reasoning can become complex sometimes, a domain expert can check the rules and make alterations easily.

Finally, the rules-based system are cost efficient and accurate in term of its result and provides output in few seconds or based on the complexity of the rule base..

2.2 Weaknesses

Firstly, for a rules-based system to work properly, effective rules must be added, they don't learn as in case of machine learning.

Secondly, rules-based system are bad to handle unknown situation, incorrect or incomplete information. Sometimes rules-based system become more complicated for a problem it is trying to solve, which leads to conflicting and overlapping of rules.

Finally, rules-based system has less self learning capacity and some time more complexity.

3 Description of the evaluation approach

To check the performance of our rule-based AI player, certain evaluation's has been performed to understand it behavior and game strategy against opponent's (random player).

1. How could we determine that our AI works at all ?.

   By comparison with a random player in N games trial.
   Hypothesis : Our AI player will win more than half of the N games.

2. What statistical test did we use in order to figure out how certain we can be that our hypothesis is correct?.

   The number of times our AI wins is binomial distributed. The statistical test that we use to determine how certain we are correct to our hypothesis, we use binomial proportion confidence interval, it is a confidence interval for the probability of success calculated from the series of success-failures experiments[2].

   Confidence intervals[BPW$^+$12] are useful for plotting a range (i.e., error bars) in which we are confident the true ratio of #wins/#total games actually lies.

   However, for a statistical test we talked about a simpler solution. Confidence intervals give you a lower bound for your number of wins, but also an upper bound, which we are not interested in (we only care about that our AI doesn't do worse than we think).

   Since we assume that #wins is binomially distributed, given the number of wins and the number of games we played, you can calculate exactly how likely it is that we got a false positive for thinking that our AI is better than the random player:

   $$\Pr(\text{false positive}) = 1 - F_{\text{bin}}(k; n, p), [2]$$

   Where F_{bin} is the binomial cumulative distribution function, k is the number of wins, n is the number of games, and p is the success probability in our zero hypothesis H_0 that we aren't better than the random player, i.e., $p = 50\%$.

   To get how confident we are with our AI player success for outcome of a series of success-failure experiments. We are calculating Normal approximation interval and Clopper–Pearson interval[2].

3. How could you determine which rules work and which don't, and how would you measure the importance of each rule?
   By deactivating one rule after another rule in N game trial.

4 Evaluation results and interpretation of results

For evaluation of results of the mention approaches, we will start with presenting the results and interpretation of all approaches.

4.1 Approach 1

For a game of 1000, between AI player and random player with seed value. The result after the game simulation was AI player has won 715 times against a random player(won 285 times). Similar N games evaluation were performed and results are shown below:

Table 1: Approach 1: Comparison of AI Player Game with Random Player

Result				
No of Games	AI-Player Wins	Random Player Wins	Elapsed Time(millisecs)	Winner (%)
1000	715 times	285 times	5904	AI (71.5)
10,000	7115 times	2885 times	17945	AI (71.15)
50,000	35254 times	14746 times	76851	AI (70.50)
100,000	70214 times	29786 times	177313	AI (70.21)
200,000	1,40,724 times	59,276 times	380759	AI (70.36)
400,000	2,81,769 times	1,18,231 times	638853	AI (70.44)
1000,000	705211 times	294789 times	1016256(16.94 minutes)	AI (70.52)

4.2 Approach 2

The binomial proportion confidence interval: using
1) **Normal approximation interval**: - It is a interval which is calculated by approximating the distribution of error about a binomial distributed observation $\hat{p}$, with a normal distribution
The success probability p is given as:[2]

$$\hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}; (1)$$

Here , $\hat{p}$= $\frac{n_S}{n}$ proportion of success

2) **Clopper–Pearson interval**: - It is often called an exact method because it is based on exactly the correct distribution rather than an approximation[2].

$$S_{\leq} := \left\{ \theta \;\middle|\; P\left[\mathrm{Bin}\left(n;\theta\right) \leq x\right] > \frac{\alpha}{2} \right\} \text{ and } S_{\geq} := \left\{ \theta \;\middle|\; P\left[\mathrm{Bin}\left(n;\theta\right) \geq x\right] > \frac{\alpha}{2} \right\}, \quad (2)$$

where $0 \leq x \leq n$ is the number of successes observed;
and Bin(n; θ) is a binomial random variable with n trials and probability of success θ [2]

Given data for our card game, confidence interval is calculated below:
N = Total no of game = 10000 ;
n_S = AI player wins = 7115;
n_F = Random player wins = 2885;
Confidence interval = $\frac{7115}{10000} \pm 1.96\sqrt{((\frac{1-7115}{10000})/10000)}$

Table 2: Approach 2: Binomial confidence interval [3][BPW+12]

No of Games: 10,000				
CI Method	AI-Player Wins	Proportion	Lower Bound 95% CL	Upper Bound 95% CL
Normal Approx.	7115 times	0.7115	0.7026	0.7204
Clopper–Pearson interval	7115 times	0.7115	0.7025	0.7204
No of Games: 200,000				
CI Method	AI-Player Wins	Proportion	Lower Bound 95% CL	Upper Bound 95%
Normal Approx.	140724 times	0.7036	0.7016	0.7056
Clopper–Pearson interval	140724 times	0.7036	0.7016	0.7056
No of Games: 400,000				
CI Method	AI-Player Wins	Proportion	Lower Bound 95% CL	Upper Bound 95%
Normal Approx.	281769 times	0.7044	0.7030	0.7058
Clopper–Pearson interval	281769 times	0.7044	0.7030	0.7058

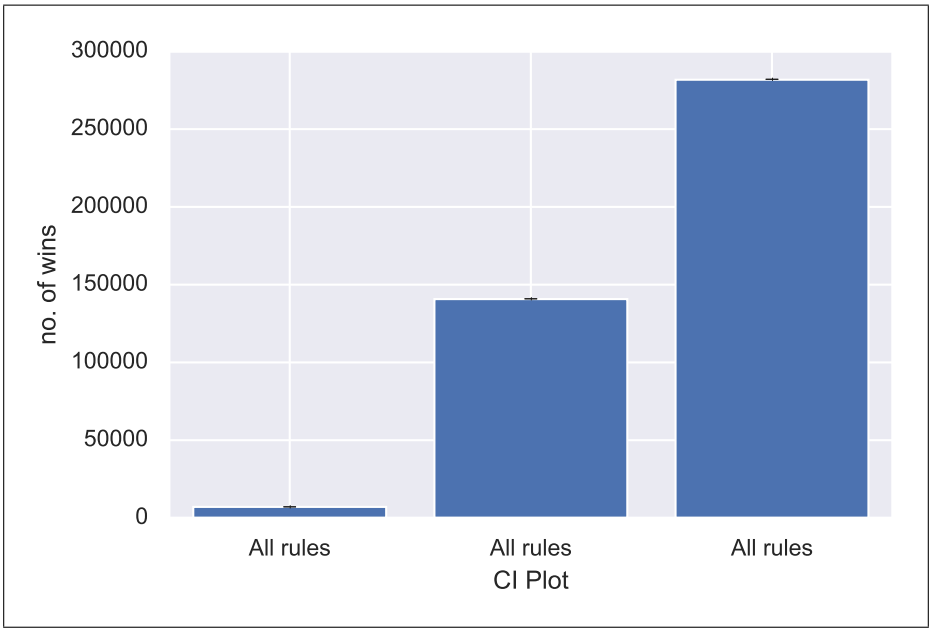

Figure 1: Confidence Interval.

From Figure 1, we can see the plotting of confidence interval for all rules in x-axis against no of wins on y-axis. The plotted bar graph shows the confidence level of our AI player to win for no of game [10,000, 200,000, 400,000] played against random player, for detail overview refer table 2.
For our rule-based AI player we find a success rate of approximately 70.6% (95% confidence interval [70%, 71%]). Since this success rate is significantly higher than 50% and therefore significantly better than a random player, we conclude that our strategy does indeed work.

4.3 Approach 3

To determine which rule in our AI player is performing better and which rule is giving worst result. we did a small experiment by playing a game of 1000 times between AI player and random player with switching off / deactivating Rule -2 (Placing card on own pile, hurt our-self as little as possible;). The result after game simulation was AI player wins zero times out of 1000 times. This interpret that Rule 2 is most important rule in our game and it play major role.
Similarly we repeated this experiment by playing a game of 8000 times between AI player and random player by deactivating rule 1(Help opponent player, but as little as possible;). The result obtained after game simulation was our AI player wins by 53%(AI wins 4232 times, random player wins 3768 times out of 8000 game, Elapsed time for game was 11720 milliseconds)

References

[1] The game face to face literature. pdf: the-game-ftf-gb.pdf.

[2] Binomial confidence interval. online at: https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval.

[3] Epitools. online at: https://epitools.ausvet.com.au/ciproportion.

[BPW+12] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.