An Extension of Counterfactual Regret Minimization for Multiplayer Card Games

Yu Cao[1,a)] Tomoyuki Kaneko[1,2,b)]

Abstract: Counterfactual Regret Minimization (CFR) [1] is one of the state-of-the-art methods for solving large imperfect-information games. It shows great performance in solving 1-to-1 poker games. But there is still little research about how to apply it to multi-player poker games. In this paper, we will apply CFR to an extension of poker which is played by 4 players (2-to-2), and compare its performance with random policy.

Keywords: Machine Learning, Counterfactual Regret Minimization, MCCFR

1. Introduction

In the real world, there are cooperation and competition in the teamwork. Members in the same team often get the same rewards just as team reward. It promotes cooperation among team members. However, in some situations, players will be rewarded based on their contribution to the team such as individual records of NBA basketball players. NBA basketball players have individual records in their basketball careers. The more points they get, the better treatment and higher salary they can get in the team. So there is also competition among players in the same team. It is a challenging problem of how to maximize personal benefits in teamwork in incomplete information games.

It is more difficult and complex for incomplete information games than complete information games. Players need to guess opponents' information according to the current state and their own experience. When they make a decision, they will worry about revealing their private information so that opponents can beat them through the information. To solve incomplete information games, a concept is proposed, which is called Nash Equilibrium. Players can't improve their rewards by only changing strategies by themselves while other players keep unchanged. This strategy profile and rewards consist of a Nash Equilibrium. However, in a long time, the algorithm can only solve lightweight problems. When it is needed to solve large information games, it costs too much time and memory. In 2007, a breakthrough occurred that Counterfactual Regret Minimization(CFR) was proposed by Martin Zinkevich and others [1].

CFR is one of the state-of-the-art methods for solving large imperfect-information games. It tracks past play to compute regret values when choosing an action and makes strategies proportional to regret values. It shows great performance in solving 1-to-1 poker games [2]. But there is still little research about how to apply it to multi-player poker games. In this paper, we will show how to maximize individual benefit while playing poker games in teams. An extension of poker game, which is played by 2 groups, 2 members for each, will be studied. This game's information sets are too large that CFR costs too much time and space to compute it. In this situation, Monte Carlo Counterfactual Regret Minimization (MCCFR) by Marc Lanctot, Kevin Waugh and others [3] is proposed to reduce time cost. It makes sampling rather than traverses the whole game tree on each iteration. We will start study from 8 cards by applying CFR to the poker game, and compare its performance with random policy. Then gradually increase the number of cards. While CFR can't solve the problem efficiently, in this case, Monte Carlo Counterfactual Regret Minimization can be adopted instead of CFR.

1 Interfaculty Initiative in Information Studies, the University of Tokyo
2 JST, PRESTO
a) souyu@g.ecc.u-tokyo.ac.jp
b) kaneko@acm.org

2. Background and Related Work

2.1 Extensive Games

An extensive-form game [3] is a kind of model of sequential decision-making multi-player imperfect information games. It is composed of components as the following part, in which we followed the standard notation in the study [3]:

- N: A finite set N of players.
- c: The chance factor of the game.
- $-i$: every player except player i.
- H: a sequence of actions that were played
- Z: a set of terminal histories, $Z \subset H$.
- $A(h)$: a set of available actions after history h if $h \notin Z$.
- $P(h)$: player function that assigns a player p to take an action after history h, $p \in N \bigcup \{c\}$.
- $\mathcal{I}_i$: information sets of player i, namely a partition of $\{h \in H : P(h) = i\}$. h and h' being in the same information set is equivalent to $A(h) = A(h')$.
- I_i: an information set for player i, $I_i \in \mathcal{I}_i$.
- $\mathrm{f}_c(a|I)$: probability of a occurring when given the information set I.
- u_z: a utility function that assigns a reward to each player when the terminal state z is reached.

2.2 An Extension of Poker

2.2.1 Original Rule

There is one of the famous poker games in China called 50K. It is characterized by three cards of 5, 10 and K being presented at the same time as the largest. 50K is played by two teams consisting of four people, in the form of 2 vs 2. The game uses two sets of poker without jokers, which means there are 104 cards in the game, and each player will get 26 cards per round. Its playing rule, which is showed in **Table 1**, is just like Dou Di Zhu [4].

Compared to Dou Di Zhu, 50K adds scoreboards and compute total points of players who belong to the same team to determine which team wins.

In the game, 5, 10 and K are scoreboards which represent 5, 10 and 10 points respectively. These points can be gotten only by playing the biggest card when there is someone playing scoreboard per round, just like player A played 5, and player B played K, and no one has larger card in his hand to play, so player B gets point card 5 and K, which means B gets 15 points. And the team who gets more points wins.

2.2.2 Simplified Version

For our experiment, we simplify the game by reducing to 8 and 12 cards, which means we only use 1, 2, four for each when there are 8 cards and use 1, 2, 3, four for each when there are 12 cards. Scoreboards are not considered in our experiment. We just set three different reward rules to show cooperation and competition within the team. First, we set the basic rule. if all players of team A play out all cards in their hands when both players of team B doesn't finish the game, team A gets 2 points, and team B loses 2 for each member. When there is one player in team B finishes the game, team A gets 1 point with team B losing 1 for each member. Second, we add the bonus rule to it. We give bonus to the player in the winning team who first plays out his cards. He can get 2.5 rather than 2, and the other player in the winning team will get 1.5 in the first situation of the basic rule. And in the second situation, players in the winning team will get 1.5 and 0.5 respectively. The last one is that the punishment rule will be added to the losing team. When the game is finished, there is one player in the losing game who played out all his cards, he will be punished slightly than his partner. They will lose 0.5 and 1.5 respectively.

2.3 Pluribus

Pluribus [6] is an efficient algorithm shown stronger than top human professional players in multi-player card games. It makes abstraction by removing some actions from consideration and bucketing similar decision states into one state. And it trains AI offline by self-play with Monte Carlo Counterfactual Regret Minimization, in which AI starts by playing randomly, and improves itself by beating previous versions of itself. When playing with human players, real-time search is taken to adjust strategies, in which AI looks some moves ahead at a leaf node in a limited depth unless it reaches terminal states to estimate expected utility value at the node. It supposes that opponents take k different strategies according to their bias. A more balanced strategy can be found by this method because choosing an unbalanced strategy will be more likely to lose when the opponents choose a strategy that just dominates it. Pluribus' success shows that it is possible to produce superhuman strategies for large multi-player imperfect-information games with a well-designed algorithm.

3. Methods

3.1 CFR

3.1.1 Nash Equilibrium

In terms of game theory, Nash Equilibrium [1], a concept named after John Forbes Nash Jr., is proposed to solve a game which is no cooperation among players. In Nash Equilibrium, Each player is supposed to know what strategies of the other players take, and all players can't improve their own rewards by changing their own strategies.

For example, if A is making the decision that maximizes his benefit as he can under the consideration of B keeping unchanged, and B is making the decision that maximizes his benefit as he can under the consideration of A keeping unchanged, A and B are in Nash Equilibrium. Similarly, if all players make the decision that maximizes their own benefits under the consideration of other players keeping unchanged, they are in Nash Equilibrium.

As is shown by mathematician Nash, there must be a Nash Equilibrium for every finite game. Therefore, we can use Nash Equilibrium to compute the optimal strategies while solving finite games.

3.1.2 Regret Matching

When an action was chosen in the current state, a regret will occur that we didn't choose another action if we can. The more you regret, the more you should choose another action in the state. So we compute the regret value of choosing a specific action in actions that are available in the current state, and make strategies proportional to the positive regret values. The greater the regret is, the greater the possibility we should choose the action. Thus, by choosing actions in this way, regrets will be minimized. This method is called Regret Matching. Suppose that we are playing Rock-Paper-Scissors. The winner can get a dollar from the loser. If we play paper when the opponent plays scissors, the opponent player wins and gets one dollar from us. We can make the dollar losing become our reward, so our reward for this play is -1 under this situation. The reward for playing scissors and rock against the opponent's scissors will be 0 and 1, respectively. We regret not playing rock most because we will get a greater reward when we play rock. The difference between the reward of choosing the action we haven't chosen and the reward of the action we actually chose under the situation that other players don't change their strategies.

For this example, we will have regret u(scissors,scissors) − u(paper,scissors) = $0 - (-1) = 1$, and we will have regret u(rock,scissors) − u(paper,scissors) = $1 - (-1) = 2$. According to regret matching, we will sum the regret values of all actions that are available, then compute every action's probability by dividing their own positive regret by the sum. The next action will be chosen proportionally to the positive regrets as choosing action scissors, paper and rock with the probabilities $\frac{1}{3}$, 0, and $\frac{2}{3}$ respectively. So in next play, we will choose rock with probability $\frac{2}{3}$ while the opponent player chooses paper. We will also update our regret by adding the regret values of this play to the previous

Table 1 Rules of 50*K*

Type	Form	Comments
Single Card	x	ranking as 3<4<...<10<J<Q<K<A<2
Pair	xx	two cards of the same number, the ranking just like a single card
Triplet	xxx	three cards of the same number
Triplet with an attached card	xxx+a	a triplet with any single card added such as 4445. These rank according to the rank of the triplet such as 6663 is bigger than 5554.
Triplet with an attached pair	xxx+aa	a triplet with a pair added, whose ranking is determined by the rank of the triplet such as 66633 is bigger than 55544.
Sequence	abcde+...	at least five cards of consecutive rank, from 3 up to A such as 8910JQ. 2 can't be added into it.
Sequences of pairs	aabb+...	at least two pairs of consecutive ranks, from 3 up to A. 2 can't be added into it. For example 8899.
Sequence of triplets	aaabbb+...	at least two triplets of consecutive ranks from three up to A. For example 555666.
Sequence of triplets with attached cards	aaabbb+x+y	two single cards are added to triplets respectively. For example 44455529. The extra cards must be different from other cards. 2 can't be used as triplets, but it can be used as extra cards
Sequence of triplets with attached pairs	aaabbb+xx+yy	two extra pairs are attached to triplets. Only the triplets have to be in sequence - for example 5556663344. The pairs must be different in rank from each other and from all the triplets. A singled card and a pair can't be attached to the triplets in the same time such as 555666344.
Bomb	xxxx+...	four or more cards of the same rank. A bomb can beat everything except 50K, and a higher ranked bomb which means more same cards can beat a lower ranked one.
Quadplex set	xxxx+a or xxxx+aa	it is just similar to triplets with attached cards and with attached pairs. What's different is that it needs 4 same cards rather than 3. And two single cards or pairs can be added into it rather than one such as 222234 and 22223344. Its ranking rule is as same as a triplet with an attached card or pair, such as 222234 is bigger than 555567. And it is smaller than bomb.
False 50k	-	only three, 10, K three cards, the patterns are not exactly the same
True 50k	-	5, 10, K three cards with the same pattern and the same color

play's. So we will get cumulative regret 3, 2, 1 respectively for choosing action scissors, rock, and paper. So the mixed strategy will be updated to $(\frac{3}{6}, \frac{2}{6}, \frac{1}{6})$.

3.1.3 Counterfactual Regret Minimization

The front part shows how to apply regret matching to single-decision game. It is unclear that How to extend regret matching algorithm to sequential games, where players should make a sequence of decisions to finish one play of the game. To solve sequential games, an extension of regret matching algorithm is proposed as known as Counterfactual Regret Minimization (CFR) [1]. Counterfactual Regret Minimization is one of the efficient methods for solving large imperfect-information games, which uses the regret-matching algorithm presented earlier. Here will give some definitions in CFR algorithm.

In CFR, σ^t means a strategy profile consisting of all players' strategies at time t. Let $\sigma_{I\to a}$ denote a strategy profile that action a is always chosen while other strategy profile is same as σ. And $\pi^\sigma(h)$ represents the reach probability of history h while players' strategy profile is σ. $\pi^\sigma_{-i}(h)$ means counterfactual reach probability of history h. So we can get counterfactual regret value equation as:

$$v_i(\sigma,h) = \sum_{z\in Z, h\sqsubset Z} \pi^\sigma_{-i}(h)\pi^\sigma(h,z)u_i(z). \qquad (1)$$

When action a is not taken at history h, the counterfactual regret will be:

$$r(h,a) = v_i(\sigma_{I\to a},h) - v_i(a,h). \qquad (2)$$

Likewise, if action a is not taken at information set I, the counterfactual regret will be:

$$r(h,a) = \sum_{h\in I} r(h,a). \qquad (3)$$

Then the cumulative counterfactual regret is denoted as:

$$R_i^T(I,a) = \sum_{t=1}^{T} r_i^t(I,a). \qquad (4)$$

Like regret-matching algorithm, CFR making its strategies proportional to cumulative regrets just as shown:

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a\in A(I)} R_i^{T,+}(I,a)} & \text{if } \sum_{a\in A(I)} R_i^{T,+}(I,a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \qquad (5)$$

In this equation, $R_i^{T,+}(I,a)$ denotes non-negative counterfactual regret value of player i until turn T for choosing action a after getting to the information set I. Player i's strategy in turn $T+1$ is proportional to positive cumulative regrets of past plays if the cumulative regret is positive. If it is not, a uniform random strategy is used.

3.2 MCCFR

It is unfeasible to solve large incomplete information games by CFR because of its large cost in traversing large information set.

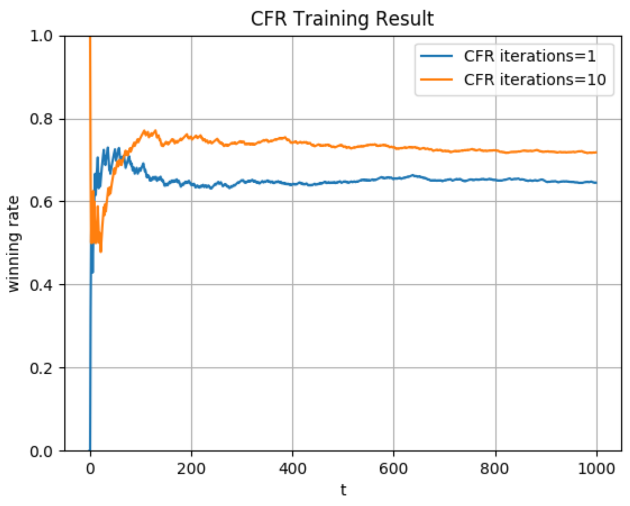

Fig. 1 {1,2}, each for 4 cards—basic rule

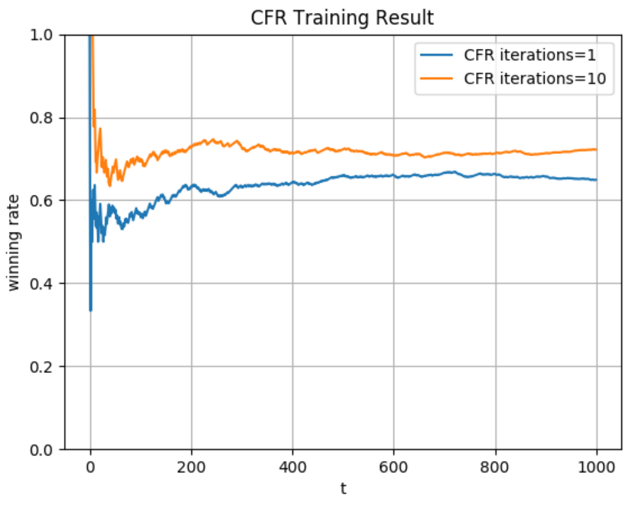

Fig. 2 {1,2}, each for 4 cards—bonus rule

An extension of CFR, Monte Carlo Counterfactual Regret Minimization [3], is proposed by Marc Lanctot and Kevin Waugh and others to reduce time cost of traversing the game tree on each iteration. On each iteration only some of the terminal histories will be considered to update the counterfactual value. There are many sampling methods, such as outcome-sampling, external sampling and average strategy sampling [5].

4. Experiment

We start our experiment from studying situations where there are 8 cards, two for each player. Playing rule is also simplified, in which players can only play a single card or a pair. And we will do our experiment by using three kinds of reward rules to compare the performance of the algorithm under different reward mechanisms. We set the game iterations to be 1000 times. It means we will play the game 1000 times. And iteration times of the algorithm will be set to 1 and 10 times to compare the influence of different iteration times to performance of the algorithm. The winning rate is plotted to show the performance. The result is shown in following figures:

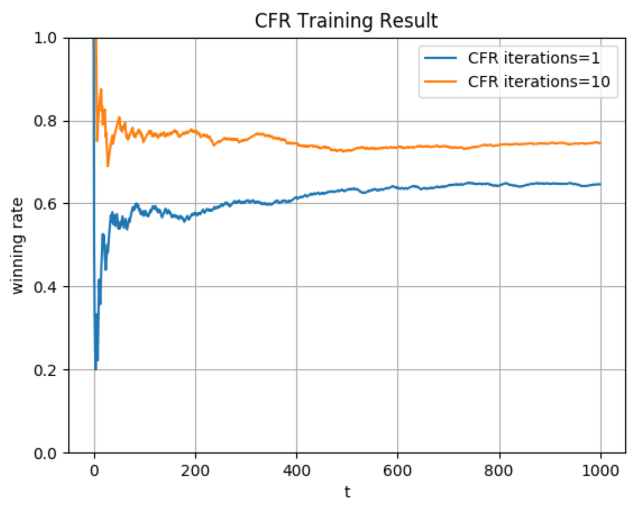

Fig. 3 {1,2}, each for 4 cards—punish rule

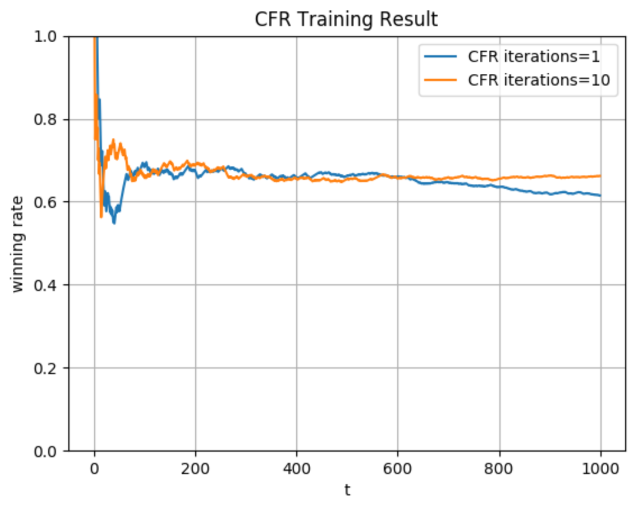

Fig. 4 {1,2,3}, each for 4 cards—punish rule

Table 2 Final winning rate

iteration times	2-basic	2-bonus	2-punish	3-punish
10	0.745	0.722	0.709	0.662

There are two teams, {S, N} and {E, W}. They sit in the corresponding place in the direction. At the start of the game, one player will be selected randomly to use AI, and other players will be random players, which means that the player will randomly select an action in the available action set. When it is the turn for AI to play, it gets the current state of the game, and simulates based on this state. AI will know its cards in its hand and the history of the current play, but it doesn't know other players' cards. Thus, AI simulates the card states of each player 1 or 10 times, and plays the game until a terminate state is reached. The counterfactual regret value of every action in the available action set under the initial state of simulating will be computed to get its strategy.

As shown in **Fig. 1**, **Fig. 2** and **Fig. 3**, we can see that with the winning rule becoming more complex, the final winning rate decreased slightly. As shown in **Table 2**, when there are 8 cards, {1, 2}, four for each card, the final winning rate of the basic rule is 0.745, and the bonus rule's winning rate is 0.722, and the punishment rule's winning rate is 0.709. We can infer that the cooperation and competition mechanisms may influence the performance of CFR algorithm. However, there are still many shortcomings in our experiments, such as limited experimental objects and too few trials. Whether this conclusion is correct or not remains to be discussed. And one more conclusion is that, the more times CFR simulates, the higher the winning rate it can get. What's the upper bound of iteration times is not studied yet. We will study it in the future.

This experiment is conducted with AMD Ryzen 7 1700 eight-core processor $\times$ 16, and develop software is pycharm with python 3.6.

When the situation that there were 8 cards were studied, the number of cards will be increased to 12. Compare the results in Fig. 3 and **Fig. 4**, we can see that with information sets increasing, the performance of the algorithm became worse, and iteration times of algorithm should be increased to improve the performance. With the number of cards increasing, a shortcoming is exposed that the computing time increased sharply, because it took more time to iterate one play. To solve this problem, Monte Carlo Counterfactual Regret Minimization algorithm will be adopted. However, due to limited time, we haven't implemented it. We will do this experiment in the future.

5. Conclusion

Counterfactual Regret Minimization is one of the state-of-the-art methods for solving large incomplete information games. We apply it to an extension of poker games which is played by two teams, 2 members for each team. It shows great performance in winning the game while playing with random players. We get three conclusions from the experiment. First of all, with the winning rule becoming more complex, the final winning rate decreased slightly. Second, the more times CFR simulates, the higher the winning rate it can get. At last, with information sets increasing, the performance of the algorithm became worse, and iteration times of algorithm should be increased to improve the performance. However, there are still many shortcomings in our experiments, such as limited experimental objects and too few trials. Whether this conclusion is correct or not remains to be discussed.

There are lots of work that can be done in the future. We only have studied the situation where there are 8 or 12 cards, but in the real game, there are 104 cards, and there are more complex playing rules. The number of information sets will be bigger than 10^{27}, so MCCFR may not possible to handle the problem efficiently. In this case, we can try to apply Pluribus algorithm or new methods to reduce the cost of traversing the game tree. And we found that the more times CFR simulates, the higher winning rate it can get. But optimal iteration times of CFR algorithm hasn't been found.

References

[1] Neller, T. W. and Lanctot, M.: An Introduction to Counterfactual Regret Minimization, *Proceedings of Model AI Assignments, The Fourth Symposium on Educational Advances in Artificial Intelligence (EAAI-2013)* (2013).

[2] Zinkevich, M., Johanson, M., Bowling, M. and Piccione, C.: Regret minimization in games with incomplete information, *Advances in neural information processing systems*, pp. 1729–1736 (2008).

[3] Lanctot, M., Waugh, K., Zinkevich, M. and Bowling, M.: Monte Carlo sampling for regret minimization in extensive games, *Advances in neural information processing systems*, pp. 1078–1086 (2009).

[4] McLeod, J.: Dou dizhu, `https://www.pagat.com/climbing/doudizhu.html`.

[5] Burch, N., Lanctot, M., Szafron, D. and Gibson, R. G.: Efficient Monte Carlo Counterfactual Regret Minimization in Games with Many Player Actions, *Advances in Neural Information Processing Systems 25* (Pereira, F., Burges, C. J. C., Bottou, L. and Weinberger, K. Q., eds.), Curran Associates, Inc., pp. 1880–1888 (online), available from ⟨http://papers.nips.cc/paper/4569-efficient-monte-carlo-counterfactual-regret-minimization-in-games-with-many-player-actions.pdf⟩ (2012).

[6] Brown, N. and Sandholm, T.: Superhuman AI for multiplayer poker, *Science*, p. eaay2400 (2019).