A Regret Minimization Approach to Competitive Rock-Paper-Scissors

Jackson Spell and **Ben Wiley**
{jaspell,bewiley}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

The classic and simple game of Rock-Paper-Scissors, also known by Roshambo, is a cyclic, zero-sum two player game with an established and easy to determine Nash equilibrium. However, players using the Nash equilibrium mixed strategy (random choices between each of the three moves) will, on average, always tie opponents. Thus, it is advantageous for players to employ an adaptive strategy which searches for, and exploits, non-random strategies in its opponents. We describe the creation of a regret-minimization algorithm to compete against opponent algorithms in Rock-Paper-Scissors.

1 Introduction

Rock-Paper-Scissors (RPS) is a simple game played between two opponents. Each round, players make simultaneous moves, each employing one of three hand positions, representing three different actions: rock, paper, or scissors. The goal for each player is to choose the action that will defeat the opponent's move subject to the rules of the game: rock defeats scissors, scissors defeat paper, and paper defeats rock. Figure 1 illustrates the cyclic rules for winning a round.

Succeeding in RPS is highly dependent on chance, as is obvious from a cyclic game with a small number of moves of equivalent value. To avoid indistinguishable wins, algorithms are compared by their competitive performance over a large number of rounds – for the purposes of this paper, 10000.

The Nash equilibrium is the balanced equilibrium of moves if both players play optimally, knowing the other player's strategy, and will not benefit from changing their strategy. For Rock-Paper-Scissors, the Nash equilibrium is a probabilistic balanced distribution of each of the three moves (Neller and Lanctot 2015). The Nash equilibrium will, on average, tie every strategy. Therefore, it is to our advantage to use a strategy that will optimize its play against non-random players.

To improve performance against non-Nash algorithms, we can consider choices of moves in terms of regret. Regret, in general, is the response to learning

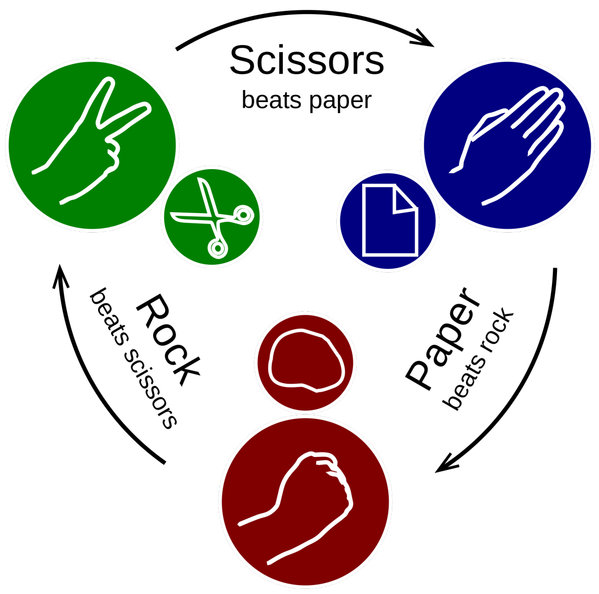

Figure 1: Rock-Paper-Scissors has a circular set of rules (Wikipedia user Enzoklop 2015).

that, for a past decision, another choice would have resulted in a better outcome. In application to games, regret can be applied in hindsight to evaluate the benefit of having made a different choice of action. By determining the regret associated with past moves, a player can anticipate the regret associated with a given upcoming move. In order to win, players naturally intend to minimize regret. Regret minimization algorithms use this approach to choose the optimal choice at each step in order to anticipate, and therefore beat, the opponent's move.

In this paper, we describe the creation of a regret-minimization algorithm to compete against computational opponents in Rock-Paper-Scissors.

2 Regret Minimization

In order to evaluate performance, we must first assign the gain associated with results of playing a given move, known as the move's utility U. We assign moves resulting in a win the utility of +1, a tie (both players chose the same action) 0, and loss −1.

Consider a single move, where the opponent plays Rock and we play Scissors, with resulting utility

$$U(\text{Scissors, Rock}) = -1$$

Then we evaluate our regret to be the difference in utility between other moves we could have played and the move we opted for. We evaluate our utility as

$$r = U(\text{Rock, Rock}) - U(\text{Scissors, Rock}) = 0 - (-1) = 1$$

$$r = U(\text{Paper, Rock}) - U(\text{Scissors, Rock}) = 1 - (-1) = 2$$

It is important to note that the regret for a move that we **did** make is 0 – choosing to play Scissors in that previous move would result in precisely the same outcome as it did when it was already played.

So, for the single round, we have regrets $r = (1, 0, 2)$ (in rock, paper, scissor order). To determine the next move, we probabilistically choose from each of the moves according to a mixed strategy derived from the normalized distribution of the regrets; that is, probabilistically from $(\frac{1}{3}, 0, \frac{2}{3})$. If there are no positive regrets in a round, indicating that we played optimally, we opt for a random choice.

By aggregating the regrets and the mixed strategies over a large number of games in the recent history of the current opponent, we establish the opponent's current mixed strategy. Note that this does not indicated an understanding of the opponent's algorithm, merely their performance over the most recent moves. The result is a mixed strategy for our algorithm for the upcoming move.

3 Experiments

We compare our algorithm against several algorithms produced by classmates by competing algorithms in pairs for 10 rounds of 10000 games against each opponent. Algorithms are compared via their average performance, measured by the average margin of victory over the 10 rounds in order to avoid chance performance fluctuations.

Opponent algorithms can be grouped, based on general technique, into 6 categories. Six algorithms use meta-strategies, which evaluate a number of approaches and choose the best anticipated approach. 2 algorithms use historical prediction, which looks back into the opponents history to check what the opponent played after similar preceding moves. 2 algorithms use frequency analysis to predict what the opponent will most likely play next. 3 algorithms simply use static mixed strategies, one uses reinforcement learning, and one chooses from a preset list of 3-move play sequences known as gambits.

4 Results

The results of the competition can be seen in Table 1. Meta-strategy algorithms are labelled with M's, historical prediction with H's, frequency analysis with F's, static mixed strategies with S's, the reinforcement learning with an L, and the gambit technique with a G. Note that negative margins of victory indicate that our algorithm was, on average, less successful than the opponent algorithm, and that small average margins of victory (below 100) indicate that the algorithms primarily tie against each other, and would likely average out in the long term.

Overall, it's quite clear from the data that this algorithm is not particularly competitive in Rock-Paper-Scissors. It loses rather spectacularly to the meta-strategy algorithms and to the historical prediction algorithms. While the regret minimization algorithm does typically beat frequency analysis algorithms, the victory is by a relatively small margin.

We suspect that the historical prediction algorithms perform so strongly against the regret minimization algorithm because our algorithm is prone to sequences of repeated action choice when it identifies a sizable regret in the recent past. The historical prediction algorithms likely identify such sequences quickly and react accordingly, as a sequence of identical plays need not be particularly long before a historical prediction algorithm is simply attempting to match a those identical plays.

The performance of meta-strategy algorithms against our algorithm is unsurprising, as such algorithms are employed by the most successful algorithms in public competitions (Knoll 2015). Such meta-algorithms, which choose the anticipated best algorithm from amongst a set of candidates, are extremely versatile.

Of curious note is the regret minimization algorithm's performance against the static mixed-strategy algorithms. The original regret minimization technique from which our algorithm is derived was intended for use as an approach for beating static mixed strategies. The altered algorithm no longer outcompetes such algorithms, instead tying or losing. We suspect that the alterations we have made, needed to adapt to non-static strategies, have compromised the algorithm's performance against true static mixed strategies. In particular, in exploring a limited depth of recent history to build a current regret prediction, the algorithm may be susceptible to short-term trends in opponent performance which, while meaningful against adaptive and logical opponents, limits the algorithm's accuracy against static mixed strategy algorithms by encouraging more erratic behavior (Neller and Lanctot 2015).

5 Conclusions

Though a fitting choice against Rock-Paper-Scissors opponents employing a fixed mixed strategy, the regret minimization algorithm does not fare well in competition with more adaptable non-fixed algorithms. The regret minimization algorithm described in this paper is significantly outperformed by most other al-

Algorithm	Margin of Victory
M1	−777.8
M2	−893.7
M3	−1082.4
M4	−1521.0
M5	−1821.6
M6	−2702.4
H1	−2147.3
H2	−2859.9
F1	285.8
F2	271.8
S1	9.7
S2	0.9
S3	−251.1
L1	31.9
G1	35.7

Table 1: Regret-minimization algorithm margin of victory against other algorithms.

gorithms tested, which are designed for competitive use. At fundamental fault is this algorithm's need for a relatively static opponent strategy, which cannot adjust quickly nor accurately enough to avoid predictable play. The authors recommend historical prediction or meta-strategy algorithms when designing for head-to-head competitive Rock-Paper-Scissors, as well as for any game with quickly changing opponent strategies.

6 Acknowledgements

The authors would like to thank their classmates in Artificial Intelligence for providing their assistance and opponent algorithms for Rock-Paper-Scissors competition.

References

Knoll, B. 2015. Rock paper scissors programming competition. `http://www.rpscontest.com/leaderboard`.

Neller, T. W., and Lanctot, M. 2015. An introduction to counterfactual regret minimization. `http://modelai.gettysburg.edu/2013/cfr/cfr.pdf`. [Online; posted 09-July-2013].

Wikipedia user Enzoklop. 2015. A chart showing how the three game elements of ”rock-paper-scissors” interact. `http://en.wikipedia.org/wiki/Rock-paper-scissors#mediaviewer/File:Rock-paper-scissors.svg`.