Exploring Mixed Strategies for Two-Player Repeated Games

Malav Dave
malavdave@lewisu.edu
CPSC-57200-001, Fall 2020
Artificial Intelligence 2
Lewis University

I. Introduction

The requirements of this week's assignment were to simulate a two-player repeated game using the *Nashpy* package in python. The goal was to try various mixed strategies for three games that were required to be tested. The experiments were focused such that the row player can maximize its expected utility.

The rest of the paper is organized as follows: section II discusses the methodology used to simulate the experiments, section III presents and discusses the results and, finally, section IV provides a conclusion and scope of further testing and improvements.

II. Methodology

A. Tools Used

The tools used to do the two-player game simulation was done entirely using *Python 3.8*. The coding environment used was *Jupyter Notebook*, an interactive web application software that functions as a python IDE. The bulk of the code had already been provided by Dr. Xu in the live lecture on December 1st, 2020. There were many dependencies for this program, including: *random*, *numpy*, *texttable*, and *nashpy*.

B. Summary:Simulating Two-Player Repeated Game

This section will describe the general flow of the code in detail. The code will be described generally, however, the same process was followed for all three games that were tested.

Firstly, a *Nashpy* game object was created using the payoff matrices for each user. All the Nash equilibria were then found using the *support_enumeration()* method that the *nashpy* package provides. The expected utilities were also found for each nash equilibrium.

The simulations of the mixed strategies (which includes pure strategies) were done using two slightly different versions of a custom function, *simulate_mixed_strategy* and *simulate_mixed_strategy_random*. For simplicity, only one of the functions, *simulate_mixed_strategy* will be explained in this paper as both functions have only minor differences. This function accepts 6 parameters, namely, *game*, *num_poss_moves_r*, *num_poss_moves_c*, *weights_sigma_r*, *weights_sigma_c*, *next_round_weights*. The parameter description is outlined in table 1.

TABLE I. Parameter Description for Simulate_mixed_Strategy Function

Parameter Name	Description
game	The Nashpy game object
num_poss_moves_r	Number of possible moves for the row player
num_poss_moves_c	Number of possible moves for the column player
weights_sigma_r	The probability with which the row player takes an action
weights_sigma_c	The probability with which the column player takes an action
next_round_weights	The probability weights of whether a next round will be played or not

In this function, firstly, the bit-vectors for which move will be taken were created for both the row and column players. The choices were then simulated using the weights for the column and row players (that were passed in as parameters). The choice simulation was done using *random.choices()* method. The expected utility was recorded and the simulation of the weighted coin toss to decide whether the next round would take place was performed. Again, this was done using the *random.choices()* method, using the *next_round_weights* parameter as the weights for the weighted coin flip.

Once the repetitions of the game had ended, the average utility for both the row and column players was calculated by dividing their respective cumulative utilities by the number of rounds played for the game.

III. Results And Discussion

The average expected utility was exactly the same for pure strategies across all 3 games. However, the average expected utilities for mixed strategies across all games were not the same as the expected utilities. Nonetheless, they were very close. This is expected because the mixed strategies rely on chance for picking any of the *n* actions for a player. There is a higher likelihood of the action with the highest probability being picked, however this does not necessarily mean the action with the maximum probability will be picked every time.

Since pure strategies never deviate from Nash equilibrium, the results for pure strategies will not be shown. The results presented in this section will include only mixed strategies. It should be noted that, since some games did not have mixed strategies at Nash equilibrium, some random mixed strategies were experimented with that may/may not be at Nash equilibrium. Even the games that did have 1 or more mixed strategies, random mixed strategies were tried for them as well, for the sake of completeness. A total of 2 different weights for the coin toss were also tried (99% and 70% chance of repetition).

A. *Game 1 Results*

Game 1 is defined by the payoff matrix shown in fig 1. The number of Nash equilibria a game has is dependent on the payoff matrix for the game.

	Column 0	Column 1
Row 0	1,0	2,1
Row 1	2,0	0,-1

Fig. 1 Payoff Matrix for Game 1

Support enumeration for this game revealed that there were 3 Nash equilibria, two of which were pure strategies, in other words, neither the row, nor the column player would ever deviate from the given strategy. The third strategy was a mixed strategy and the results of this simulation, along with other randomly generated mixed strategies, is shown in tables 2 and 3 for two different coin toss probabilities, respectively. It should be noted that the results in tables 2 and 3 are presented in increasing order of the probability of the row player playing move 1.

TABLE II. Mixed Strategy PayOffs for Game 1 With 99% Chance of Repetition

Row Player Strategy	Column Player Strategy	Number of Rounds	Payoff (avg. expected util.)
[0.3, 0.7]	[0.667, 0.333]	5	[0.8, -0.2]
[0.4, 0.6]	[0.5, 0.5]	33	[1.273, 0.061]
[0.5, 0.5]	[0.667, 0.333]	19	[1.368, 0]
[0.9, 0.1]	[0.4, 0.6]	198	[1.485, 0.460]

TABLE III. Mixed Strategy PayOffs for Game 1 With 70% Chance of Repetition

Row Player Strategy	Column Player Strategy	Number of Rounds	Payoff (avg. expected util.)
[0.3, 0.7]	[0.667, 0.333]	1	[2, 0]
[0.4, 0.6]	[0.5, 0.5]	2	[2, 1]
[0.5, 0.5]	[0.667, 0.333]	1	[2, 0]
[0.9, 0.1]	[0.4, 0.6]	3	[1.667, 0.667]

The general observation from tables 2 and 3 is that the row player is always positioned to win, no matter which move he/she makes. It can be seen from table 1 that the overall payoff for the row player increases as the chance of playing action 1 increases. When the payoff values in table 2 is compared with table 3, it can be seen that they are vastly different. In fact, they are closer to the payoff values of the pure strategies. This can be attributed to the extremely low number of rounds that were performed in table 3.

B. *Game 2 Results*

Game 2 is defined by the payoff matrix shown in fig 2.

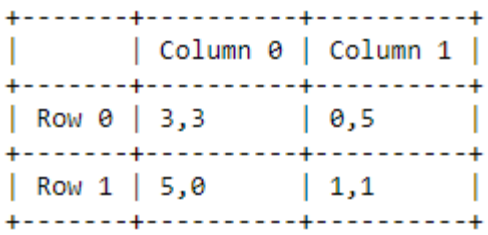

Fig. 2 Payoff Matrix for Game 2

Support enumeration for this game revealed that there was 1 Nash equilibrium, which was a pure strategy. Some randomly generated mixed strategies were tried for this game. The results are shown in tables 4 and 5 for two different coin toss probabilities, respectively. It should be noted that the results in tables 4 and 5 are presented in increasing order of the probability of the row player playing move 1.

TABLE IV. MIXED STRATEGY PAYOFFS FOR GAME 2 WITH 99% CHANCE OF REPETITION

Row Player Strategy	Column Player Strategy	Number of Rounds	Payoff (avg. expected util.)
[0.2, 0.8]	[0.3, 0.7]	20	[1.950, 1.450]
[0.4, 0.6]	[0.6, 0.4]	326	[2.709, 1.804]
[0.6, 0.4]	[0.8, 0.2]	168	[3.119, 2.197]
[0.75, 0.25]	[0.2, 0.8]	425	[0.819, 3.701]

TABLE V. MIXED STRATEGY PAYOFFS FOR GAME 2 WITH 70% CHANCE OF REPETITION

Row Player Strategy	Column Player Strategy	Number of Rounds	Payoff (avg. expected util.)
[0.2, 0.8]	[0.3, 0.7]	1	[1, 1]
[0.4, 0.6]	[0.6, 0.4]	4	[1.75, 1.75]
[0.6, 0.4]	[0.8, 0.2]	7	[2.857, 2.857]
[0.75, 0.25]	[0.2, 0.8]	2	[2, 2]

The payoff matrix in fig. 2 show us that even there is only 1 state where row player wins and only 1 state where column player wins. If a mixed strategy is simulated, however, even those restrictions become loose. It can be observed from table 4 that in cases where both column and row players' chances of playing action 2 are higher, then row player wins. This observation deviates from what the payoff table shows for the same case, where the expected payoff should be (1, 1). We can attribute this to the probabilistic approach of choosing actions and the number of rounds that are conducted. If the same entry is looked at in table 5, where only 1 round was conducted, the payoff is the same as the expected payoff. Similar patterns are found throughout other entries of tables 4 and 5.

C. Game 3 Results

Game 3 was different from games 1 and 2, in that the column player had 4 possible actions he/she could take, whereas the row player only had 2 possible actions. Game 3 was also unique in the aspect that it was a degenerate game, which mean that it had more than *k* pure best moves for *k* support [1]. This can be seen in the payoff matrix for game 3 in fig. 3, where the third column has two pure best moves (for support of 1). Degenrate games yield an even number of Nash equilibria states [1].

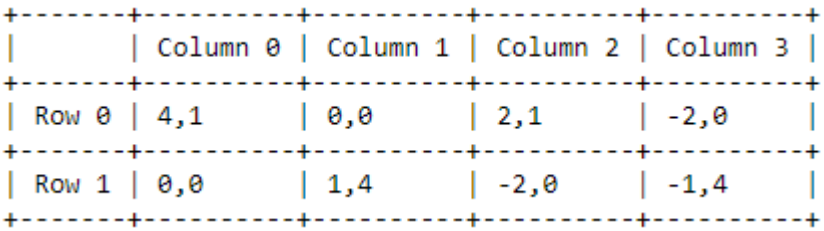

Fig. 3 Payoff Matrix for Game 3

Support enumeration for this game revealed that there were 6 Nash equilibria, four of which were pure strategies. Two of the six strategies were mixed strategies and the results of their simulation, along with other randomly generated mixed strategies is shown in tables 6 and 7 for two different coin toss probabilities, respectively. It should be noted that the results in tables 6 and 7 are presented in increasing order of the probability of the row player playing move 1.

TABLE VI. MIXED STRATEGY PAYOFFS FOR GAME 3 WITH 99% CHANCE OF REPETITION

Row Player Strategy	Column Player Strategy	Number of Rounds	Payoff (avg. expected util.)
[0.2, 0.8]	[0.2, 0.5, 0.1, 0.2]	56	[0.125, 2.161]
[0.4, 0.6]	[0.3, 0.1, 0.5, 0.1]	15	[-0.067, 1.6]
[0.6, 0.4]	[0.2, 0.2, 0.2, 0.2]	81	[0.543, 1.111]
[0.8, 0.2]	[0.2, 0.8, 0, 0]	7	[0.143, 0.571]
[0.8, 0.2]	[0, 0.8, 0.2, 0]	4	[0.25, 1]

TABLE VII. MIXED STRATEGY PAYOFFS FOR GAME 3 WITH 99% CHANCE OF REPETITION

Row Player Strategy	Column Player Strategy	Number of Rounds	Payoff (avg. expected util.)
[0.2, 0.8]	[0.2, 0.5, 0.1, 0.2]	9	[0, 0.889]
[0.4, 0.6]	[0.3, 0.1, 0.5, 0.1]	1	[-2, 0]
[0.6, 0.4]	[0.2, 0.2, 0.2, 0.2]	1	[2, 1]
[0.8, 0.2]	[0.2, 0.8, 0, 0]	4	[1, 0.25]
[0.8, 0.2]	[0, 0.8, 0.2, 0]	8	[0.75, 0.375]

From table 6, we can see that more strategies were in favor of the column player, rather than the row player (compared to what was seen in the previous two games). It is a very difficult task to derive a pattern that can show when the payoff will be in favor of the row player vs. the column player. Although, it can be observed that, like the previous two games, the number of rounds played affected the average payoff dramatically. This can be seen in the differences for the *Payoff* column for tables 6 and 7.

IV. CONCLUSION AND FURTHER TESTING

Overall, it was interesting to see what happens when randomized strategies are played over many repetitions. It can be said that these simulations would most likely represent real-world scenarios, where agents are not completely rational (in terms of maximizing expected utility). This can help us glean into what situations would be like with human agents where the utility function varies from person to person. The results showed that the number of rounds played affect the average payoff significantly and more rounds usually mean the average payoff is closer to the expected payoffs (for mixed strategies at Nash equilibrium only).

There is scope for further testing, in that, a more thorough analysis of different weighted coin flip values could be tried to truly see the effects of the number of rounds played on average payoff. The randomized strategies tested were in fact generated by hand. An improvement on this can also done, where an algorithm generates random mixed strategies based on the parameters of the game, namely the payoff matrix.

On the more difficult side, the code can be edited so that it shows the exact move the row/column players picked during a round. This can allow for the calculation of descriptive statistics such as variance, standard deviation, average deviation, etc. This can indeed reveal more information about mixed strategies that are not at Nash equilibrium.

REFERENCES

[1] Knight, V. (2017). Degenerate games — Nashpy 0.0.19 documentation. Nashpy. https://nashpy.readthedocs.io/en/latest/reference/degenerate-games.html#degenerate-games