יונתן פרידמן 204169320

1

1.1

שחקן ב' / שחקן א'	אוכל	ארוחת	
אוכל	4,4	0,2	
ארוחת	2,0	2,2	

$$M = \begin{bmatrix} 4 & 0 \\ 2 & 2 \end{bmatrix}$$

$$N = \begin{bmatrix} 4 & 2 \\ 0 & 2 \end{bmatrix}$$

1.2

נסמן ב-p את הסיכוי ש-א יבחר אוכל וב-q את הסיכוי ש-ב יבחר אוכל.

בגלל שיווי משקל עליך למקסם

$$\frac{\partial V_1}{\partial p} = \frac{\partial V_2}{\partial q} = 0$$

נמצא שצב שקול ל:

$$q^* = \frac{M_{22} - M_{12}}{M_{11} - M_{12} - M_{21} + M_{22}}, \quad p^* = \frac{N_{22} - N_{21}}{N_{11} - N_{12} - N_{21} + N_{22}}$$

$$= q^* = \frac{2-0}{4-0-2+2} = \frac{2}{4} = \boxed{\frac{1}{2}}$$

$$p^* = \frac{2-0}{4-2-0+2} = \frac{2}{4} = \boxed{\frac{1}{2}}$$

ובאופן דומה סימטרי ולכן האסטרטגיה עבור בי שיווי משקל מעורבת הבאה.

Yuval_Friedmann_ex10_computation_and_cognition

January 20, 2020

[0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[0]:
```
import pandas as pd
import numpy as np
import os
os.chdir("/content/drive/My Drive/C&C/ex10_files/python")
import utils, Merchant_00000001, Merchant_00000002, Merchant_00000003
```

0.0.1 Q2

2.1 The average reward for Merchant_1 vs. Merchant_2:

[12]:
```
utils.run_game("Merchant_00000001", "Merchant_00000002")
```

[12]:
```
(6.602, 6.564)
```

2.2 Strategies
* Merchant_1:
In the first game the player chooses randomly his action (with probability 0.5 to each action). Afterwards, the player chooses the opposite action from his last game.
* Merchant_2:
The player chooses randomly his action (with probability 0.5 to each action) without considering his history and his opponent history.

2.3 My Algorithm
There is no "pure strategy", but there is a "best response": if the opponent chooses 'L' we better choose H, and if the opponent chooses 'H' we better choose 'L. Therfore, we will try to predict the next action of the opponent, and we will choose the opposite action.
The prediction of the opponent's next action is based on the previous sequence of our choices and the opponent's response to this sequence. We will check our previous i choices (i ranges from 1 to min{10, number of games played}. We won't check sequences larger then 10 for reducing complexity and avoiding 'overfitting' to the opponent older choiches).
Then, we will see what was the responses by the opponent to this sequence in the history, and will compute the mean of the responses (we will sign it by $res(i)$ = the mean response for last-sequence(i)).
We assumes that the opponent gives more importance to our recent choices and less importance

to the older choices (long sequences). Therfore, we will assign weights to each mean-response, depending on what sequence it came from: $w = (\frac{1}{2})^i$ (approx. summed up to 1). Then, each i will get a score equals to $w(i) * res(i)$. Note that this scoring method requires us to represent 'L' action by -1 instead of 0.
Finally, we will take the simple mean of the scores (which are ranged from -1 to 1) and compute its sign, for 1 we will return 0 and for -1 we will return 1.

2.4

Attached is a .py file