108071027 計財系 林羽霈

賽局理論(Game Theory)

一、摘要

本篇專題皆以兩人的情況進行分析與討論，以不同得失的四種遊戲情況討論納什平衡的可能發生情形，其中，遊戲四為典型例子「囚徒困境」。

二、簡介

賽局理論為決策者間互動的分析，考慮個體的預測行為及實際行為，並研究出最加畫策略。與傳統的決策者分析方式的最大不同之處，在於將每位決策者對其他決策者行為的知識與預期納入分析架構。當我們極大化自己的報酬時，對手也正努力極大化他自己的報酬。

三、應用

- Fig1：遊戲一

		Player B	
		Left	Right
Player A	Top	1,3	0,1
	Bottom	3,1	1,0[1]

從 Fig1 來看，如果 A 選手選擇 Bottom，B 選手選擇 Left，A 選手的得失為 3，而 B 選手的得失為 1。以這個遊戲案例來說，無論在任何情況下，Bottom 對 A 選手都是最佳選擇，因為(3,1)大於(1,0)，同樣道理，Left 對 B 選手是最佳選擇。依上述可得出，此遊戲的主導策略(dominant strategy)[2]為 A 選手選擇 Bottom，而 B 選手選擇 Left。

接著，我們用 python 找出遊戲一的納什均衡[3]，以下是程式碼：

```
import nashpy as nash
import numpy as np

# 建立遊戲一__ payoff matrix
A = np.array([[1,0],[3,1]])
B = np.array([[3,1],[1,0]])
game1 = nash.Game(A,B)
game1
# Find the Nash Equilibrium with Support Enumeration"
equilibria = game1.support_enumeration()
for eq in equilibria:
    print(eq)
```

```
In [38]: runfile('C:/Users/claire/清大/二上/Linear algebra/10807
wdir='C:/Users/claire/清大/二上/Linear algebra')
(array([0., 1.]), array([1., 0.]))
```

由上面的輸出結果來看，A 選手會選擇在矩陣中以”1”表示的第二個元素位置，也

[1] 得失矩陣 (Pay-Off Matrix)：面對未來情勢不確定的狀況之下，不管發生任何情況，都對自己比較有利的決策。

[2] 主導策略 (Dominant strategy)：每個玩家都有一個最佳策略，而與其他玩家採用何種策略無關。

[3] 納什均衡 (Nash equilibrium)：單獨一方改變策略時，無法提高報酬。

就是 Bottom，B 選手會選擇在矩陣中以”1”表示的第一個元素位置，也就是 Left，所以(Bottom,Left)就是遊戲一的納什均衡。

➢ <u>Fig2：遊戲二</u>

		Player B	
		Left	Right
Player A	Top	3,1	0,0
	Bottom	0,0	1,3

從 Fig2 來看，如果 A 選手選擇 Top，B 選手選擇 Left 為較佳決定，因為 2>0，而如果 A 選手選擇 Bottom，B 選手會選擇 Right，因為 4>0，所以(Top,Left)和(Bottom,Right)都是最佳選擇組合，而這兩種組合就稱為 Nash 均衡(Nash equilibrium)，因此，我們也可知 Nash 均衡可以有超過一種以上的組合。

接著，我們同樣用 python 找出遊戲二的納什均衡，以下是程式碼：

```
# 建立遊戲二__ payoff matrix
A = np.array([[3,0],[0,1]])
B = np.array([[1,0],[0,3]])
game2 = nash.Game(A,B)
game2
# Find the Nash Equilibrium with Support Enumeration
equilibria = game2.support_enumeration()
for eq in equilibria:
    print(eq)
```

```
In [39]: runfile('C:/Users/claire/淡大/二上/Linear algebra/108071027_game theory.py',
wdir='C:/Users/claire/淡大/二上/Linear algebra')
(array([1., 0.]), array([1., 0.]))
(array([0., 1.]), array([0., 1.]))
(array([0.75, 0.25]), array([0.25, 0.75]))
```

遊戲二的輸出結果和遊戲一的最大不同之處，在於有三行輸出。

第一行(array([1., 0.]), array([1., 0.]))：A 選手會選擇在矩陣中以” 1” 表示的第一個元素位置，也就是 Top， B 選手會選擇在矩陣中以” 1” 表示的第一個元素位置，也就是 Left，所以(Top, Left)是遊戲二的第一個納什均衡。

第二行(array([0., 1.]), array([0., 1.]))：A 選手會選擇在矩陣中以” 1” 表示的第二個元素位置，也就是 Bottom， B 選手會選擇在矩陣中以” 1” 表示的第二個元素位置，也就是 Right，所以(Bottom,Right)是遊戲二的第二個納什均衡。

第三行(array([0.75, 0.25]), array([0.25, 0.75]))：A 選手選擇 Top 的機率為 0.75，Bottom 的機率為 0.25;B 選手選擇 Left 的機率為 0.25，選擇 Right 的機率為 0.75，因此從積率來看，A 選手應該選擇 Top，而 B 選手應選 Right。

像是遊戲二有多種納什平衡的投資組合，即稱為 mixed strategy Nash equilibrium[4]。那麼，在 mixed strategy Nash equilibrium 中，我們也可以用 python 計算出 A 選手及 B 選手的利益：

[4] 混合策略納什平衡 (mixed strategy Nash equilibrium)：每個玩家選擇的策略都是相對於其他玩家的策略的最佳策略。

```
# Calculate Utilities
sigma_r = np.array([.75,.25])
sigma_c = np.array([.25,.75])
pd = nash.Game(A, B)
pd[sigma_r, sigma_c]

u_r = 0.75*0.25*3 + 0.75*0.25*0 + 0.75*0.25*0 + 0.75*0.25*1
u_c = 0.25*0.75*0 + 0.75*0.25*0 + 0.75*0.25*1 + 0.75*0.25*3
print(u_r,u_c)  #r:row,c:column
```

```
In [45]: runfile('C:/Users/claire/淡大/二上/Linear algebra/108071027_game theory.py',
wdir='C:/Users/claire/淡大/二上/Linear algebra')
(array([1., 0.]), array([1., 0.]))
(array([0., 1.]), array([0., 1.]))
(array([0.75, 0.25]), array([0.25, 0.75]))
0.75 0.75
```

從輸出結果可看到，以一般統計算法：

u_r = 0.75*0.25*3 + 0.75*0.25*0 + 0.75*0.25*0 + 0.75*0.25*1

u_c = 0.25*0.75*0 + 0.75*0.25*0 + 0.75*0.25*1 + 0.75*0.25*3

算出的利益與用 python 算出的結果相同。

➤ Fig3：遊戲三

		Player B	
		Left	Right
Player A	Top	0,0	0,-2
	Bottom	2,0	-2,3

如果 A 選手選擇 Top， B 選手會選擇 Left (0 > -2)，如果 B 選擇 Left， A 會選 Bottom (2 > 0) ，如果 A 選擇 Bottom，B 會選 Right (3 > 0)，如果 B 選擇 Right，A 會選 Top (0 > -2)。以上分析結果顯示遊戲三中無納什平衡點。

小結：由遊戲一至三的分析中，可發現納什平衡點不一定只有一個，也可能有兩個以上（例：遊戲二），或是無平衡點（例：遊戲三）。

賽局理論中最典型的非零和例子為「囚徒困境」，因此我想特別提出此範例實作於 python，假設甲，乙兩人因案被捕，如果只有一個人認罪，那認罪的人只要被關一年，但否認犯罪的人要被關五年，如果兩人都認罪，都要被關三年，如果兩人都不認罪，由於證據薄弱，兩人只要被關兩年。明顯地，如果兩人都不認罪，對彼此都是最好的，這種情況稱為柏拉圖效率，但是此情況中的納什均衡點是否會等於最適點？以下用 python 實測：

➤ Fig4：遊戲四

		乙	
		認罪	否認
甲	認罪	-3,-3	-1,-5
	否認	-5,1	-2,-2

```python
# Create the payoff matrix
甲 = np.array([[-3,1],[-5,-2]])
乙 = np.array([[-3,-1],[-5,-2]])
game4 = nash.Game(甲,乙)
game4
# Find the Nash Equilibrium with Support Enumeration
equilibria = game4.support_enumeration()
for eq in equilibria:
    print(eq)
```

```
In [47]: runfile('C:/Users/claire/滄大/二上/Linear algebra/108071027_game theory.py',
wdir='C:/Users/claire/滄大/二上/Linear algebra')
(array([1., 0.]), array([0., 1.]))
```

從輸出結果可看出，甲會選擇在矩陣中以”1”表示的第一個元素位置，也就是認罪， 乙也會選擇在矩陣中以”1”表示的第一個元素位置，同樣是認罪，所以「兩人都認罪」是囚徒困境的納什均衡。

小結：由囚徒困境可發現納什平衡不一定能達到柏拉圖效率。

四、結論

上述四種情況加以分析之後，可歸納出幾項納什平衡的發生情況及特性：

1. 只有一個納什平衡點
2. 多個納什平衡點
3. 無納什平衡點
4. 納什平衡點的發生不一定伴隨柏拉圖效率

五、參考文獻

維基百科。
https://zh.wikipedia.org/wiki/%E5%8D%9A%E5%BC%88%E8%AE%BA
https://zh.wikipedia.org/wiki/%E5%B8%95%E7%B4%AF%E6%89%98%E6%9C%80%E4%BC%98

Mythili Krishnan。< Game Theory concepts with application in Python using Nashpy>。
https://towardsdatascience.com/game-theory-in-python-with-nashpy-cb5dceab262c

台大經濟系。<Nash 均衡>。
http://www.econ.ntu.edu.tw/uploads/archive_file_multiple/file/58f4cd7c48b8a101de002549/nash.html