Economics with Python 코딩대회

:게임이론을 이용한 <골든 볼>게임 내쉬 균형

경제학부

2020087147 김용환

1) 선정한 문제/주제

제가 선정한 문제는 게임이론이 실제 사례에 얼마나 적용이 되는지 보는 것입니다.

그 중 영국에서 방영된 <골든 볼>프로그램 사례를 볼 것입니다.

2) 선정 동기

평소 계량 경제나 통계 관련 문제를 이용한 파이썬을 이용해왔으나, 이번 코딩대회를 통해 제가 접하지 못하였던 게임이론을 알아보고 싶었으며, 게임이론 수업을 듣지 못하여 좀 더 쉽게 다가가기 위해 게임이론을 이용한 코딩을 준비하였습니다. 그러던 중 실제 사례로 적합할 <골든 볼> 프로그램을 알게 되었으며, 한번 적용해 보고 싶었습니다.

3) 채택한 모델링 방법론 및 패키지

연산과 수리적인 모형을 쉽게 도와주는 'numpy' 라이브러리와와 내쉬 균형을 확인해주는 'nash' 라이브러리를 이용하였습니다.

먼저 간단한 행렬을 만든 뒤, 내쉬 균형을 나타내 보았습니다.

```
In [1]: import numpy as np
        import nashpy as nash
        A = np.array([[2,0],[4,2]]) # A is the row player
        B = np.array([[4,2],[2,0]]) # B is the column player
        game1 = nash.Game(A,B)
        game1

Out[1]: Bi matrix game with payoff matrices:

        Row player:
        [[2 0]
         [4 2]]

        Column player:
        [[4 2]
         [2 0]]
```

```
In [2]: equilibria = game1.support_enumeration()
        for eq in equilibria:
            print(eq)

        (array([0., 1.]), array([1., 0.]))
```

(이미지1-1)

이미지1-1를 보면, 각 player의 행렬을 만들어 준뒤, 내쉬 균형을 나타내는 코드를 나타내면 2번 row와 1번 column (첫번 째 array는 row를, 두번 째 array는 column을 나타냄)이 내쉬 균형이며, 첫번째 예시의 내쉬 균형은 우월 전략이 될 것 입니다.

두 번째 예시로는 우월 전략이 아닌 내쉬 균형만 있을 경우입니다.

```
In [3]: A = np.array([[4,0],[0,2]]) # A is the row player
        B = np.array([[2,0],[0,4]]) # B is the column player
        game2 = nash.Game(A,B)
        game2

Out[3]: Bi matrix game with payoff matrices:

        Row player:
        [[4 0]
         [0 2]]

        Column player:
        [[2 0]
         [0 4]]
```

```
In [4]: equilibria = game2.support_enumeration()
        for eq in equilibria:
            print(eq)

        (array([1., 0.]), array([1., 0.]))
        (array([0., 1.]), array([0., 1.]))
        (array([0.66666667, 0.33333333]), array([0.33333333, 0.66666667]))
```

(이미지1-2)

이미지 1-2를 보게 되면, 첫 번째 내쉬 균형으로는 1행1열이 될 것이며, 두번째 균형으로는 2행 2열이 될 것입니다. 특히, 세번째 array를 보게 되면, 확률로 나타내어져 있습니다. 이는 확률적으로 복수의 전술이 채용되는 경우를 '혼합전략'이라고 하며, A플레이어는 각 행마다 66.67%, 33.33% 확률로 최적의 빈도를 의미합니다. B플레이어는 각 열에서 같은 의미로 해당이 됩니다.

이를 새로운 행렬로 만든 뒤 혼합 전략을 하게 되면 아래의 이미지1-3이 나옵니다.

```
In [5]: # Calculate Utilities
        sigma_r = np.array([.67,.33])
        sigma_c = np.array([.33,.67])
        pd = nash.Game(A, B)
        pd[sigma_r, sigma_c]

Out[5]: array([1.3266, 1.3266])
```

(이미지 1-3)

결과 값은 아래의 방법으로 추정이 가능하다.

$$u_r(\sigma_r, \sigma_c) = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}\sigma_{ri}\sigma_{cj}$$

(이미지2-1)

$$u_c(\sigma_r, \sigma_c) = \sum_{i=1}^{m}\sum_{j=1}^{n} B_{ij}\sigma_{ri}\sigma_{cj}$$

(이미지2-2)

A플레이어의 혼합 전략식은 이미지2-1와 같고, B플레이어의 혼합 전략식은 이미지2-2와 같다.

이를 풀어서 계산을 하게 되면,

0.67*0.33*4+0.33*0.67*0+0.33*0.67*0+0.33*0.67*2 = 1.3266 와

0.33*0.67*2+0.67*0.33*0+0.33*0.67*0+0.67*0.33*4 = 1.3266 값으로 이미지1-3의 값이 나온다는 것을 확인할 수 있습니다.

이번에는 실제 있을 법한 사례를 이용하여 적용을 해 보았습니다.

코카콜라와 펩시의 가격 결정 전략을 수립하는 방법을 보겠습니다.

	코카콜라		
		고가전략	저가전략
팹시	고가전략	1,000만, 1,000만	700만, 1,200만
	저가전략	1,200만, 700만	900만, 900만

(이미지3-1)

이미지3-1를 보게 되면, 두 회사의 고가와 저가의 전략을 나타내는 이미지입니다. 이를 내쉬 균형을 돌려 최적의 전략을 확인해 보았습니다.

```
In [8]: # Create the game with the payoff matrix
        pep = np.array([[1000,700],[1200,900]]) # A is the row player
        coca = np.array([[1000,1200],[700,900]]) # B is the column player
        game5 = nash.Game(pep,coca)
        game5

Out[8]: Bi matrix game with payoff matrices:

        Row player:
        [[1000  700]
         [1200  900]]

        Column player:
        [[1000 1200]
         [ 700  900]]

In [9]: # Find the Nash Equilibrium with Support Enumeration
        equilibria = game5.support_enumeration()
        for eq in equilibria:
            print(eq)

        (array([0., 1.]), array([0., 1.]))
```

(이미지3-2)

보시는 바와 같이 두 회사가 저가 전략을 선택하는 것이 내쉬 균형이라고 확인이 됩니다.

그럼, <골든 볼>프로그램의 사례를 적용해 보겠습니다. <골든 볼>프로그램에서 게임 중 <Split or Steal>게임을 적용해 보겠습니다. 여기서 <Split or Steal>게임이란, 두 명의 플레이어는 Split과 Steal 적힌 공을 가지게 되는데, 2명 모두 Split 공을 고르게 되는 경우, 상금은 50%씩 가지게 되며, 1명만 Steal공을 고르게 되면 그 1명이 상금을 모두 차지하게 됩니다. 2명이 Steal를 고르게 되면 아무도 상금을 가지지 못하게 됩니다.

이를 파이썬에 적용해 보겠습니다.

```
In [10]: # Create the game with the payoff matrix
         a = np.array([[0.5,0],[1,0]]) # A is the row player
         b = np.array([[0.5,1],[0,0]]) # B is the column player
         game6 = nash.Game(a,b)
         game6
```

```
Out[10]: Bi matrix game with payoff matrices:

         Row player:
         [[0.5 0. ]
          [1.  0. ]]

         Column player:
         [[0.5 1. ]
          [0.  0. ]]
```

(이미지4-1)

각 플레이어의 행렬을 만들었습니다. 바로 내쉬 균형을 적용해 보겠습니다.

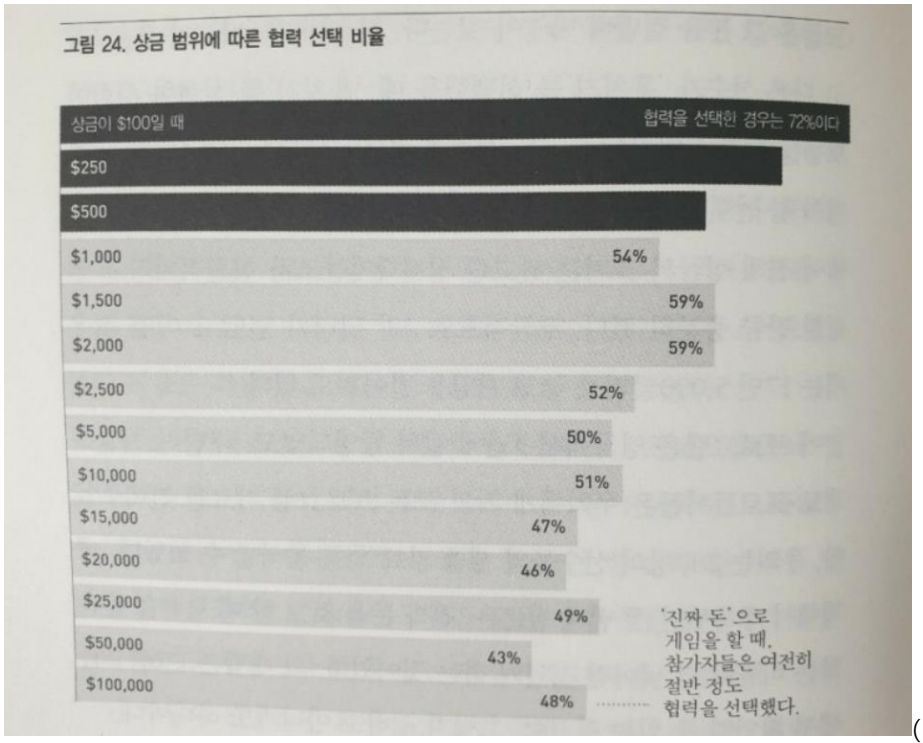
```
In [11]: equilibria = game6.support_enumeration()
         for eq in equilibria:
             print(eq)
```

```
(array([1., 0.]), array([0., 1.]))
(array([0., 1.]), array([1., 0.]))
(array([0., 1.]), array([0., 1.]))
```

(이미지4-2)

결과는 내쉬 균형이 3개가 나왔습니다. 두 플레이어가 Split를 고르는 경우만 제외하고 나머지가 내쉬 균형이라고 나타납니다.

그럼, 실제로 프로그램에서 많은 결과값이 나온 것은 위에 결과랑 같은 지 확인해 보겠습니다.



(이미지5)

(출처 책'똑똑한 사람들의 멍청한 선택')

이미지5를 보게 되면 상금은 상금이 적은 경우 70% 이상의 플레이어 들이 '협력'(둘 다 Split)을 선택하였으며, 상금이 늘어나도 절반 가까이 '협력'을 선택하는 결과가 나왔습니다.

저는 왜 어떻게 이러한 값이 나왔는지 의문이 생겼습니다. 상금과 상관없이 상당 수의 플레이들은 저의 파이썬 결과 값의 반대로 나타났습니다. 저는 당연하게도, 흔히 생각하는 '죄수의 딜레마'문제라고 생각을 하였습니다. 그러나, 생각을 해 본 결과 간과한 것이 있었습니다. '죄수의 딜레마'에서 중요한 조건이 빠져 있었습니다. 그것은 <Split or Steal>게임에서는 선택하기 전, 두 플레이어 간의 '토론'을 한다는 점이었습니다. '토론'으로 인해 내쉬 균형이라고 하지 않았던 'Split'(=협력)을 고르게 된 것입니다.

4) 시사점

게임 이론은 좋은 이론이며, 많은 일상과 보안(블록체인 등)에도 적용이 됩니다. 허나, 이와 같이 간단한 조건이 빠져 버리면, 내쉬 균형은 무너집니다. 이는 곧 게임 이론은 이론적으로 다가갈 수밖에 없으며, 많은 곳에 사용하기에는 어려움을 가졌습니다. 경제학을 배우는 사람이라면, 한 번은 관심을 가져야 하는 문제라고 생각이 들며, 실생활에 쉽게 적용이 되어진다면 게임이론 활용은 무궁무진 할 것이라고 생각이 듭니다. 저는 비록 게임이론을 이번 코딩대회 기회로 접하였으며, 이를 간단히 이해를 돕고자 코딩을 해 보았으나, 저에게는 아직 어려운 학문이었습니다. 하지만, 이번 계기로 좀 더 쉽게 다가갈 수 있었으며, 비록 간단한 코딩을 이용하였지만 좀 더 심화된 문제일지라도 파이썬을 이용하여 푼다면 좀 더 수월하게 접근할 수 있을 거라 생각합니다.