Notebook

August 10, 2017

```
In [1]: import numpy as np
        import nash
```

1 Player strategies

1.1 Definition of mixed strategies

A mixed strategy for a player with strategy set S is denoted by $\sigma \in [0,1]_{\mathbb{R}}^{|S|}$ and corresponds to a probability distribution over the pure strategies of player i. So:

$$\sum_{i=1}^{|S|} \sigma_i = 1$$

The expected score of a player can then be calculated as a measure over the probability distributions.

1.2 Calculating utilities

Considering a game $(A, B) \in \mathbb{R}^{\times}$, if σ_r and σ_c are the mixed strategies for the row/column player (respectively). The utility to the row player is:

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

and the utility to the column player is:

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

This comes from:

- The probability of being in a given cell of A or B: $\sigma_{r i} \sigma_{c j}$
- The value of the particular cell: A_{ij} or B_{ij}

As an example consider the matching pennies game:

$$A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$$

with the following mixed strategies:

$$\sigma_r = (.2, .8) \qquad \sigma_c = (.6, .4)$$

We have:

$$u_r(\sigma_r, \sigma_c) = 0.2 \times 0.6 \times 1 + 0.2 \times 0.4 \times (-1) + 0.8 \times 0.6 \times (-1) + 0.8 \times 0.4 \times 1 = -0.12$$

$$u_c(\sigma_r, \sigma_c) = 0.2 \times 0.6 \times (-1) + 0.2 \times 0.4 \times 1 + 0.8 \times 0.6 \times 1 + 0.8 \times 0.4 \times (-1) = 0.12$$

1.3 Linear algebraic calculation

Note that we can rearrange the expressions for the utilities:

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

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

in turn this corresponds to the matrix vector product:

$$u_r(\sigma_r, \sigma_c) = \sigma_r A \sigma_c^T$$

$$u_c(\sigma_r, \sigma_c) = \sigma_r B \sigma_c^T$$

We can use numpy to verify this calculation:

```
In [2]: A = np.array([[1, -1], [-1, 1]])
        B = np.array([[-1, 1], [1, -1]])
        sigma_r = np.array([.2, .8])
        sigma_c = np.array([.6, .4])
        np.dot(sigma_r, np.dot(A, sigma_c)), np.dot(sigma_r, np.dot(B, sigma_c))

Out[2]: (-0.11999999999999998, 0.11999999999999998)
```

Finally we can also directly calculate this using a `nashpy` game:

```
In [4]: matching_pennies = nash.Game(A, B)
        matching_pennies[sigma_r, sigma_c]

Out[4]: array([-0.12,  0.12])
```