Vince's Game theory course Documentation

Vince Knight

Jan 25, 2023

Contents

The class notes are available here: vknight.org/gt/.

CHAPTER 1

Class meetings

Here are my notes for each class meeting.

1.1 00 Introduction to class

1.1.1 Corresponding chapters

- Introduction to the course
- Normal form games

1.1.2 Objectives

- Outline timetable and approach to class.
- Discuss feedback from previous years.
- Make students aware of learning resources available to them.
- Make students aware of assessment criteria
- Make students aware of coding aspect of course (both in assessment and in content)
- Introduce games.

1.1.3 Notes

Timetable and approach

Discuss timetable

I will not lecture

This course is taught in a flipped learning environment. All notes, homework, exercises are available to you.

In class we will work in parallel to the notes but not directly following them. You will be expected to play an active role in your learning:

- Activities that demonstrate concepts;
- Discussions

Feedback

Show feedback from previous year.

Learning and assessment

All resources can be found at http://vknight.org/gt/

Seeking assistance

Details on website.

Programming

The course notes will include code throughout, this is both to show you how to use code to study mathematics (an exemplar of modern mathematics) but also to illustrate the ideas.

Details about the programming involved can be found at http://vknight.org/gt/

Introduction to games

Using http://vknight.org/school-outreach/assets/playing-games/tex/form.pdf

1. Play two thirds of the average game
2. Rationalise
3. Play two thirds of the average game

Discuss notes on Normal form games.

1.2 01 Strategies and utilities

1.2.1 Corresponding chapters

- Calculating utilities of strategies
- Rationalisation

1.2.2 Objectives

- Define mixed strategies
- Understand utility calculation for mixed strategies
- Understand utility calculation as a linear algebraic construct
- Be able to identify dominated strategies
- Understand notion of Common knowledge of granularity

1.2.3 Notes

Utility calculations

Use `matching pennies form` have students play in pairs. Following each game:

- Ask how many people won?
- Ask why they won?

Mixed strategies

Look at definition for mixed strategies.

Consider:

$$A = \begin{pmatrix} 2 & -2 \\ -1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} -2 & 2 \\ 1 & -1 \end{pmatrix}$$

Let us assume we have $\sigma_r = (.3, .7)$ and $\sigma_c = (.1, .9)$:

$$u_r(\sigma_r, \sigma_c) = 0.3 \times 0.1 \times 2 + 0.3 \times 0.9 \times (-2) + 0.7 \times 0.1 \times (-1) + 0.7 \times 0.9 \times 1 = 0.08$$

because the game is zero sum we immediately know:

$$u_c(\sigma_r, \sigma_c) = -0.08$$

This corresponds to the linear algebraic multiplication:

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

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

(Go through this on the board, make sure students are comfortable.)

This can be done straightforwardly using `numpy`:

```
>>> import numpy as np
>>> A = np.array([[2, -2], [-1, 1]])
>>> B = np.array([[-2, 2], [1, -1]])
>>> sigma_r = np.array([.3, .7])
>>> sigma_c = np.array([.1, .9])
>>> np.dot(sigma_r, np.dot(A, sigma_c)), np.dot(sigma_r, np.dot(B, sigma_c))
(0.079..., -0.079...)
```

Strategy profiles as coordinates on a game

One way to thing of any game $(A, B) \in \mathbb{R}^{m \times n^2}$ is as a mapping from the set of strategies $[0, 1]^m_{\mathbb{R}} \times [0, 1]^m_{\mathbb{R}}$ to $\mathbb{R}^2$: the utility space.

Equivalently, if S_r, S_c are the strategy spaces of the row/column player:

$$(A, B) : S_r \times S_c \to \mathbb{R}^2$$

We can use games defined in `nashpy` in that way:

```
>>> import nash
>>> game = nash.Game(A, B)
>>> game[sigma_r, sigma_c]
array([ 0.08, -0.08])
```

Rationalisation of strategies

Identify two volunteers and play a sequence of zero sum games where they play as a team against me. The group is the row player.

$$A = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}$$

(No dominated strategy)

$$A = \begin{pmatrix} 1 & 2 \\ -1 & 2 \end{pmatrix}$$

(First column weakly dominates second column)

$$A = \begin{pmatrix} 1 & -1 \\ -1 & -3 \end{pmatrix}$$

(First row strictly dominates second row) (Second column strictly dominates first column)

$$A = \begin{pmatrix} 2 & 2 \\ -1 & 2 \end{pmatrix}$$

(First row weakly dominates second row) (First column weakly dominates second column)

$$A = \begin{pmatrix} -1 & 2 & 1 \\ -2 & -2 & 1 \\ 1 & 1 & -1 \end{pmatrix}$$

(First row dominates second row) (First column dominates second column)

Now pit the two players against each other, the utilities represent the share of the total amount of chocolates/sweets gathered so far:

$$A = \begin{pmatrix} 1 & 0 \\ 1.5 & .5 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1.5 \\ 0 & .5 \end{pmatrix}$$

Capture all of the above (on the white board) and discuss each action and why they were taken.

Iterated elimination of dominated strategies

As a class work through the following example.

$$A = \begin{pmatrix} 2 & 5 \\ 1 & 2 \\ 7 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 3 \\ 6 & 1 \\ 0 & 1 \end{pmatrix}$$

1. First row dominates second row

$$A = \begin{pmatrix} 2 & 5 \\ 7 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 3 \\ 0 & 1 \end{pmatrix}$$

2. Second column dominates first column

$$A = \begin{pmatrix} 2 \\ 7 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

3. Second (third) row dominates first row. Thus the rationalised behaviour is (r_3, c_1).

Now return to the last example played as a pair:

$$A = \begin{pmatrix} -1 & 2 & 1 \\ -2 & -2 & 1 \\ 1 & 1 & -1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & -2 & -1 \\ 2 & 2 & -1 \\ -1 & -1 & 1 \end{pmatrix}$$

1. The first row/column weakly dominate the second row/column:

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

There is nothing further that we can do here.

1.3 02 Best responses and Nash equilibrium

1.3.1 Corresponding chapters

- Nash equilibria

1.3.2 Objectives

- Define best responses
- Identify best responses in pure strategies
- Identify best responses against mixed strategies
- Theorem: best response condition
- Definition of Nash equilibria

1.3.3 Notes

Best response against mixed strategies

Use `best responses against mixed strategies` have students play against a mixed strategy (sample using Python).

Discuss the definition of a best response. Identify best responses for the game considered:

$$A = \begin{pmatrix} \underline{2} & -2 \\ -1 & \underline{1} \end{pmatrix} \qquad B = \begin{pmatrix} -2 & \underline{2} \\ \underline{1} & -1 \end{pmatrix}$$

Consider the best responses against a mixed strategy:

- Assume $\sigma_r = (x, 1-x)$
- Assume $\sigma_c = (y, 1-y)$

We have:

$$A\sigma_c^T = \begin{pmatrix} 4y-2 \\ 1-2y \end{pmatrix} \qquad \sigma_r B = \begin{pmatrix} 1-3x & 3x-1 \end{pmatrix}$$

Here is the code to do this calculation with `sympy`:

```
>>> import sympy as sym
>>> import numpy as np
>>> x, y = sym.symbols('x, y')
>>> A = sym.Matrix([[2, -2], [-1, 1]])
>>> B = - A
>>> sigma_r = sym.Matrix([[x, 1-x]])
>>> sigma_c = sym.Matrix([y, 1-y])
>>> A * sigma_c, sigma_r * B
(Matrix([
[ 4*y - 2],
[-2*y + 1]]), Matrix([[-3*x + 1, 3*x - 1]]))
```

Plot these two things:

```
>>> import matplotlib.pyplot as plt
>>> ys = [0, 1]
>>> row_us = [[(A * sigma_c)[i].subs({y: val}) for val in ys] for i in range(2)]
>>> plt.plot(ys, row_us[0], label="$(A\sigma_c^T)_1$")
[<matplotlib...>]
>>> plt.plot(ys, row_us[1], label="$(A\sigma_c^T)_2$")
[<matplotlib...>]
>>> plt.xlabel("$\sigma_c=(y, 1-y)$")
<matplotlib...>
>>> plt.title("Utility to player 1")
<matplotlib...>
>>> plt.legend();
<matplotlib...>

>>> xs = [0, 1]
>>> row_us = [[(sigma_r * B)[j].subs({x: val}) for val in xs] for j in range(2)]
>>> plt.plot(ys, row_us[0], label="$(\sigma_rB)_1$")
[<matplotlib...>]
>>> plt.plot(ys, row_us[1], label="$(\sigma_rB)_2$")
[<matplotlib...>]
```

(continues on next page)

(continued from previous page)

```
>>> plt.xlabel("$\sigma_r=(x, 1-x)$")
<matplotlib...>
>>> plt.title("Utility to column player")
<matplotlib...>
>>> plt.legend();
<matplotlib...>
```

Conclude:

$$\sigma_r^* = \begin{cases} (1,0), & \text{if } y > 1/2 \\ (0,1), & \text{if } y < 1/2 \\ \text{indifferent}, & \text{if } y = 1/2 \end{cases} \qquad \sigma_c^* = \begin{cases} (0,1), & \text{if } x > 1/3 \\ (1,0), & \text{if } x < 1/3 \\ \text{indifferent}, & \text{if } x = 1/3 \end{cases}$$

Some examples:

- If $\sigma_r = (2/3, 1/3)$ then $\sigma_r^* = (0, 1)$.
- If $\sigma_r = (1/3, 2/3)$ then *any* strategy is a best response.

Discuss best response condition theorem and proof.

This gives a finite condition that needs to be checked. To find the best response against σ_c we **potentially** would need to check all infinite possibilities alternatives to σ_r^*. Now we simply need to check the values of the pure strategies against $sigma_c$:

- Either there will be a single **pure** best response;
- There will be multiple **pure** strategies for which the row player is indifferent.

Return to previous example:if $\sigma_r = (1/3, 2/3)$ then $(\sigma_r B) = (0, 0)$ thus $(\sigma_r B)_j = 0$ for all j.

$(\sigma_r, \sigma_c) = ((1/3, 1/2), (1/2, 1/2))$ is a pair of best responses.

Discuss definition of Nash equilibria.

Explain how the best response condition theorem can be used to find NE.

- All possible supports (strategies that are played with positive probabilities) can be checked.
- All pure strategies must have maximum and equal payoff.

1.4 03 Support enumeration

1.4.1 Corresponding chapters

- Support enumeration

1.4.2 Objectives

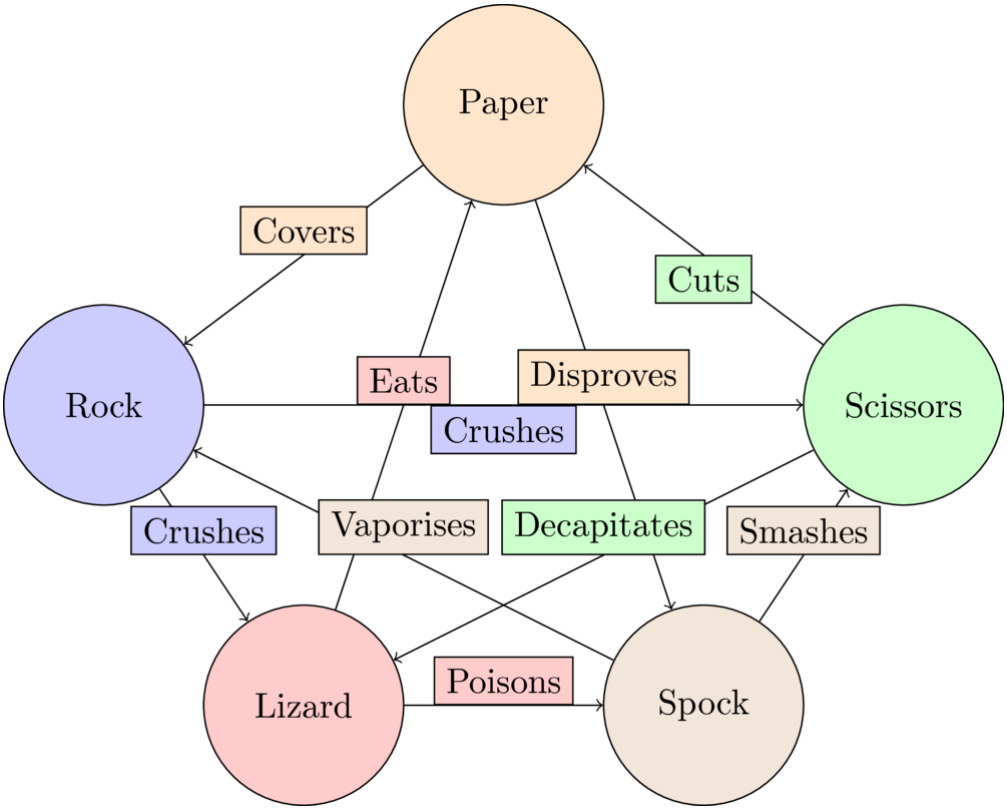
- Define of support of a strategy
- Define nondegenerate games
- Describe and use suport enumeration

1.4.3 Notes

Play a knockout RPSLS tournament

16 volunteers and play RPSLS as a dropout tournament.

Explain rules and show:



Following the tournament: have discussion about how winner won etc…

Explain that we will study this game using Game Theory:

$$A = \begin{pmatrix} 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & -1 & -1 & 1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & 1 & -1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 0 \end{pmatrix}$$

Then go over chapter 5:

- Define support of a strategy (relate to tournament previously played). For example the strategy $(0, 1/2, 0, 0, 1/2)$ has support $\{2, 5\}$:

```
>>> import numpy as np
>>> sigma = np.array([0, 1/2, 0, 0, 1/2])
>>> np.where(sigma > 0)
(array([1, 4]),)
```

- Discuss non degenerate games and show how RPSLS is in fact a degenerate game:

```
>>> A = np.array([[0, -1, 1, 1, -1],
...               [1, 0, -1, -1, 1],
...               [-1, 1, 0, 1, -1],
...               [-1, 1, -1, 0, 1],
...               [1, -1, 1, -1, 0]])
>>> sigma_c = np.array([1, 0, 0, 0, 0])
>>> np.dot(A, sigma_c)  # Two element give max
array([ 0,  1, -1, -1,  1])
```

- Discuss support enumeration algorithm. Theoretically could be used for RPSLS but would need to consider supports of unequal size. For the purpose of the illustration consider RPS:

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

Apply the algorithm.

1. We see there are no pairs of best response, so consider $k \in \{2, 3\}$.
2. We have the following set of utilities to consider:
    1. $I = \{1, 2\}$
        1. $J = \{1, 2\}$
        2. $J = \{1, 3\}$
        3. $J = \{2, 3\}$
    2. $I = \{1, 3\}$
        1. $J = \{1, 2\}$
        2. $J = \{1, 3\}$
        3. $J = \{2, 3\}$
    3. $I = \{2, 3\}$
        1. $J = \{1, 2\}$
        2. $J = \{1, 3\}$
        3. $J = \{2, 3\}$
    4. $I = J = \{1, 2, 3\}$
3. Now we consider (some not all as they are mainly the same) of the corresponding linear equations.
    1. $I = \{1, 2\}$
        1. $J = \{1, 2\}$

$$0\sigma_{r1} - \sigma_{r2} = \sigma_{r1} + 0\sigma_{r2} \tag{1.1}$$
$$-\sigma_{r2} = \sigma_{r1} \tag{1.2}$$

$$0\sigma_{c1} - \sigma_{c2} = \sigma_{c1} + 0\sigma_{c2} \tag{1.3}$$
$$\sigma_{c1} = -\sigma_{c2} \tag{1.4}$$

        2. $J = \{1, 3\}$

$$0\sigma_{r1} - \sigma_{r2} = -\sigma_{r1} + \sigma_{r2} \tag{1.5}$$
$$\sigma_{r1} = 2\sigma_{r2} \tag{1.6}$$

$$0\sigma_{c1} + \sigma_{c3} = \sigma_{c1} - \sigma_{c3} \tag{1.7}$$
$$\sigma_{c1} = 2\sigma_{c3} \tag{1.8}$$

2. $J = \{2, 3\}$

$$\sigma_{r1} + 0\sigma_{r2} = -\sigma_{r1} + \sigma_{r2} \tag{1.9}$$
$$2\sigma_{r1} = \sigma_{r2} \tag{1.10}$$

$$-\sigma_{c2} + \sigma_{c3} = 0\sigma_{c2} - \sigma_{c3} \tag{1.11}$$
$$\sigma_{c2} = 2\sigma_{c3} \tag{1.12}$$

2. $I = \{1, 3\}$ Similarly.
3. $I = \{2, 3\}$ Similarly.
4. $I = J = \{1, 2, 3\}$

   In this case we have:

$$-\sigma_{r2} + \sigma_{r3} = \sigma_{r1} - \sigma_{r3} \tag{1.13}$$
$$\sigma_{r1} - \sigma_{r3} = -\sigma_{r1} \tag{1.14}$$
$$\tag{1.15}$$

   which has solution:

$$\sigma_{r1} = \sigma_{r2} = \sigma_{r3}$$

   Similarly:

$$-\sigma_{c2} + \sigma_{c3} = \sigma_{c1} - \sigma_{c3} \tag{1.16}$$
$$\sigma_{c1} - \sigma_{c3} = -\sigma_{c1} \tag{1.17}$$
$$\tag{1.18}$$

   which has solution:

$$\sigma_{c1} = \sigma_{c2} = \sigma_{c3}$$

4. Now we consider which of those supports give valid mixed strategies:
   1. $I = \{1, 2\}$
      1. $J = \{1, 2\}$

$$\sigma_r = (k, -k, 0)$$

         which is not possible

      2. $J = \{1, 3\}$

$$\sigma_r = (2/3, 1/3, 0) \tag{1.19}$$
$$\sigma_c = (2/3, 0, 1/3) \tag{1.20}$$

      3. $J = \{2, 3\}$

$$\sigma_r = (1/3, 2/3, 0) \tag{1.21}$$
$$\sigma_c = (0, 2/3, 1/3) \tag{1.22}$$

   2. $I = \{1, 3\}$ Similarly.

3. $I = \{2, 3\}$ Similarly.
4. $I = J = \{1, 2, 3\}$

$$\sigma_r = (1/3, 1/3, 1/3) \tag{1.23}$$
$$\sigma_c = (1/3, 1/3, 1/3) \tag{1.24}$$

5. The final step is to check the best response condition:
   1. $I = \{1, 2\}$
      2. $J = \{1, 3\}$

$$A\sigma_c^T = \begin{pmatrix} 1/3 \\ 1/3 \\ -2/3 \end{pmatrix}$$

Thus σ_r is a best response to σ_c.

$$\sigma_r B = (-1/3, 2/3, -1/3)$$

Thus σ_c is **not** a best response to σ_r.

      3. $J = \{2, 3\}$

$$A\sigma_c^T = \begin{pmatrix} -1/3 \\ -1/3 \\ 2/3 \end{pmatrix}$$

Thus σ_r is **not** a best response to σ_c.

$$\sigma_r B = (-2/3, 1/3, 1/3)$$

Thus σ_c is a best response to σ_r.

   2. $I = \{1, 3\}$ Similarly.
   3. $I = \{2, 3\}$ Similarly.
   4. $I = J = \{1, 2, 3\}$

$$A\sigma_c^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Thus σ_r is a best response to σ_c.

$$\sigma_r B = (0, 0, 0)$$

Thus σ_c is a best response to σ_r.

We can confirm all of this using `nashpy`:

```
>>> import nash
>>> A = np.array([[0, -1, 1],
...               [1, 0, -1],
...               [-1, 1, 0]])
>>> rps = nash.Game(A)
>>> list(rps.support_enumeration())
[(array([ 0.333...,  0.333...,  0.333...]), array([ 0.333...,  0.333...,  0.333...]))]
```

Note that it can be computationally expensive to find **all** equilibria however `nashpy` can be used to find **a** Nash equilibrium by finding the first one:

```
>>> next(rps.support_enumeration())
(array([ 0.333...,  0.333...,  0.333...]), array([ 0.333...,  0.333...,  0.333...]))
```

Discuss Nash's theorem briefly. Highlight how that can seem contradictory for the output of `nashpy` (using support enumeration) for the degenerate game of the notes. However, that won't always be the case:

```
>>> A = np.array([[0, -1, 1, 1, -1],
...               [1, 0, -1, -1, 1],
...               [-1, 1, 0, 1, -1],
...               [-1, 1, -1, 0, 1],
...               [1, -1, 1, -1, 0]])
>>> rpsls = nash.Game(A)
>>> list(rpsls.support_enumeration())
[(array([ 0.2,  0.2,  0.2,  0.2,  0.2]), array([ 0.2,  0.2,  0.2,  0.2,  0.2]))]
```

Some details about the proof:

- Proved in a 19 page thesis! (2 pages of appendices)
- Noble prize for economics
- Watch a beautiful mind

1.5 04 Best response polytopes

1.5.1 Corresponding chapters

- Best response polytopes

1.5.2 Objectives

- Define the best response polytopes (using both definitions: convex hull and halfspaces).
- Labelling of vertices
- Equivalence between equilibrium and fully labeled pair
- Vertex enumeration

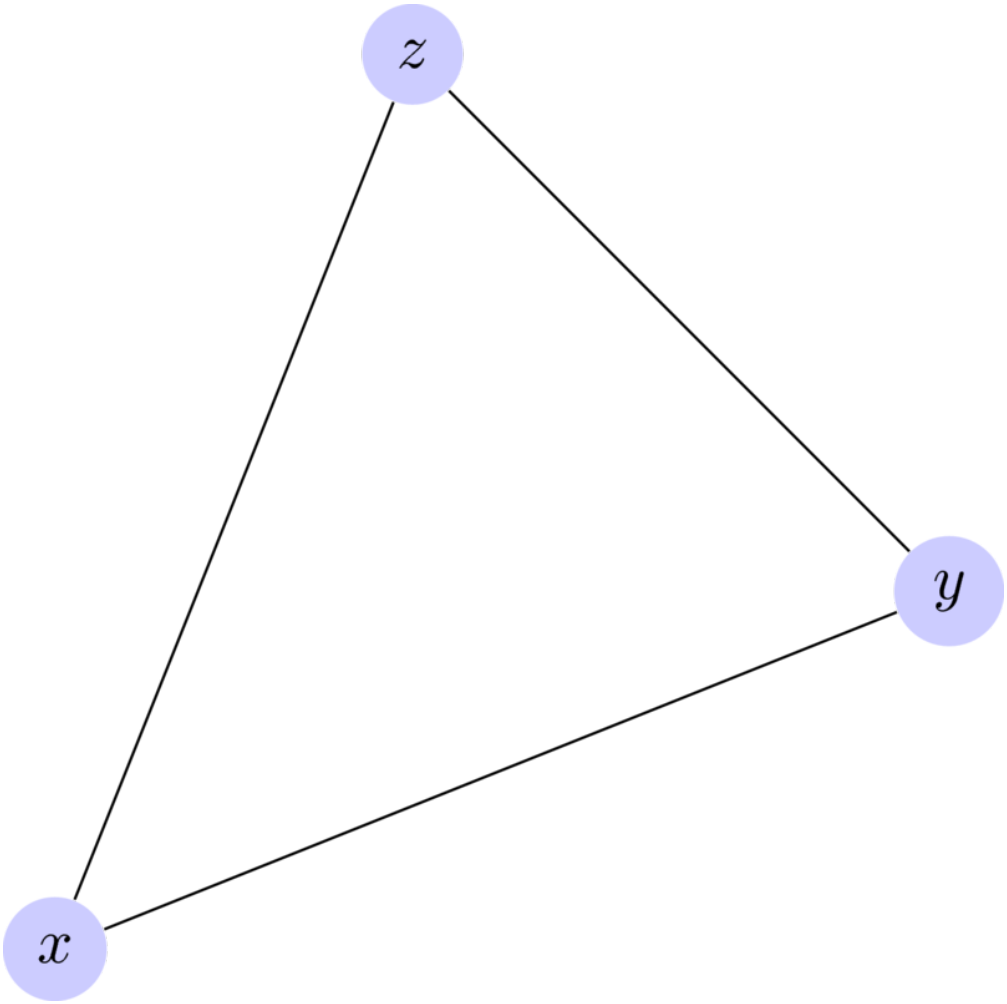
1.5.3 Notes

Defining best response polytopes

Discuss what an average is. Place this in the context of the line between two points. Weighted averages of two points are just a line. So an average is just a probability distribution. Expand this notion to the average over **two points**. How would you take the weighted average between three points x, y, z:

$$\lambda_1 x + \lambda_2 x + \lambda_3 x$$

This corresponds to something like:



Explain how another definition for that space would be draw inequalities on our variables. Give definition of best response polytopes for a game. Illustrate this with the battle of the sexes game (scaled!):

$$A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}$$

Go over the definition of the best response polytopes.

The *row* best response polytope is given by:

$$\mathcal{P} = \{x \in \mathbb{R}^m \mid x \geq 0; xB \leq 1\}$$

This corresponds to the following inequalities:

- $x_1 \geq 0$
- $x_2 \geq 0$
- $2x_1 \leq 1$
- $x_1 + 3x_2 \leq 1$

From this we can identify the vertices:

- $(x_1, x_2) = (0, 0)$
- $(x_1, x_2) = (1/2, 0)$

- $(x_1, x_2) = (0, 1/3)$
- $(x_1, x_2) = (1/2, 1/6)$

This can be confirmed using `nashpy`:

```
>>> import numpy as np
>>> import nash
>>> B = np.array([[2, 1], [0, 3]])
>>> halfspaces = nash.polytope.build_halfspaces(B.transpose())
>>> for v, l in nash.polytope.non_trivial_vertices(halfspaces):
...     print(v)
[ 0.5  0. ]
[ 0.          0.333...]
[ 0.5         0.166...]
```

The *column* best response polytope is given by:

$$\mathcal{Q} = \{y \in \mathbb{R}^m \mid Ay \leq 1; y \geq 0\}$$

This corresponds to the following inequalities:

- $3y_1 + y_2 \leq 1$
- $2y_2 \leq 1$
- $y_1 \geq 0$
- $y_2 \geq 0$

From this we can identify the vertices:

- $(y_1, y_2) = (0, 0)$
- $(y_1, y_2) = (1/3, 0)$
- $(y_1, y_2) = (0, 1/2)$
- $(y_1, y_2) = (1/6, 1/2)$

Confirmed:

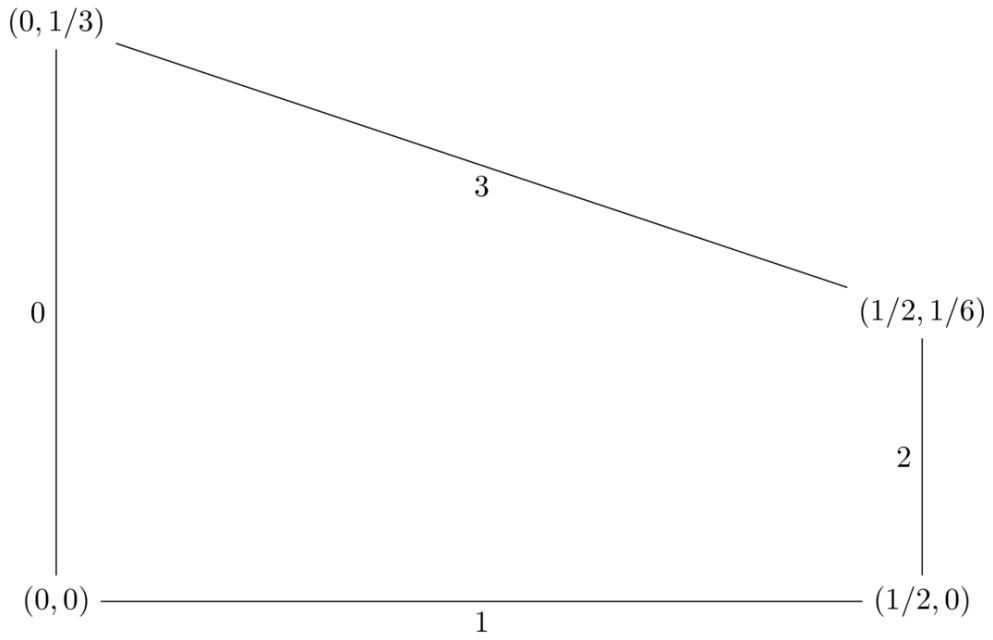
```
>>> import numpy as np
>>> import nash
>>> A = np.array([[3, 1], [0, 2]])
>>> halfspaces = nash.polytope.build_halfspaces(A)
>>> for v, l in nash.polytope.non_trivial_vertices(halfspaces):
...     print(v)
[ 0.333...  0.        ]
[ 0.   0.5]
[ 0.1666...  0.5       ]
```

Pair activity

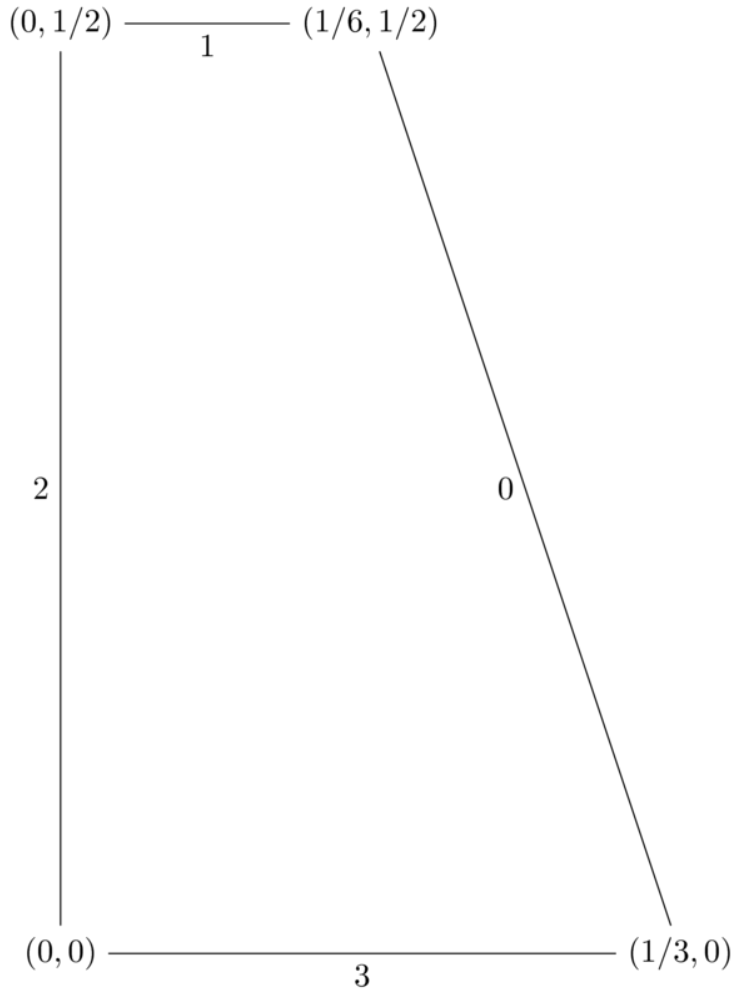
Ask everyone to draw these two polytopes.

Now describe how we label the vertices: **using the same ordering as the inequalities** (starting at 0), a vertex has the label corresponding to that inequality if it is a strict equality.

$\mathcal{P}$:



$\mathcal{Q}$:



Explain that what these polytopes represent is the scaled strategies when players maximum utilities are 1. So given, the action of an opponent, if the players' utility is 1 they are playing a best response.

Ask students to assign each other either $\mathcal{P}$ or $\mathcal{Q}$. Both players should choose a vertex and investigate the utilities.

For example, if:

- The row player ($\mathcal{P}$) picked: $(0, 1/3)$ with labels: $\{0, 3\}$.
- The column player ($\mathcal{Q}$) picked: $(1/6, 1/2)$ with labels: $\{0, 1\}$.

This implies:

- The row player is **not** playing their first strategy (label 0), so playing their second strategy. Also (label 3), the best response to this is that the column player plays their second strategy.
- The best response to what the column player is currently doing is to play both strategies.

At this point the column player has an incentive to move, they will move to the $(0, 1/2)$ vertex with labels: $\{1, 2\}$ which implies:

- The row player is as before.
- The column player is playing their first strategy (label 2). The best response to this is for the row player to play their second strategy (label 1).

So this is a Nash equilibria.

Another example:

- The row player ($\mathcal{P}$) picked: $(1/2, 1/6)$ with labels: $\{2, 3\}$.
- The column player ($\mathcal{Q}$) picked: $(1/6, 1/2)$ with labels: $\{0, 1\}$.

This implies:

- The best response to what the row player is currently doing is to play both strategies.
- The best response to what the column player is currently doing is to play both strategies.

Neither player has a reason to move.

Another example:

- The row player ($\mathcal{P}$) picked: $(0, 1/3)$ with labels: $\{0, 3\}$.
- The column player ($\mathcal{Q}$) picked: $(1/3, 0)$ with labels: $\{0, 3\}$.

This implies:

- The row player is **not** playing their first strategy (label 0), so playing their second strategy. Also (label 3), the best response to this is that the column player plays their second strategy.
- The column player is **not** playing their second strategy (label 3), so playing their first strategy. Also (label 0), the best response to this is that the row player plays their first strategy.

Neither player is happy here. Once one moves (not allowing the origin because that's not playing) we arrive at a similar situation to before.

After students have walked through this themselves have a discussion about what property seems to indicate Nash equilibrium. Move to discussing the notes.

In particular highlight that we've scaled the utility so we need to (at the end) scale the vertices too!

Finally show how this is implemented in `nashpy`:

```
>>> A = np.array([[1, -1], [-1, 1]])
>>> matching_pennies = nash.Game(A)
>>> for eq in matching_pennies.vertex_enumeration():
...     print(eq)
(array([ 0.5,  0.5]), array([ 0.5,  0.5]))
```

1.6 05 Lemke Howson Algorithm

1.6.1 Corresponding chapters

- The Lemke Howson algorithm

1.6.2 Objectives

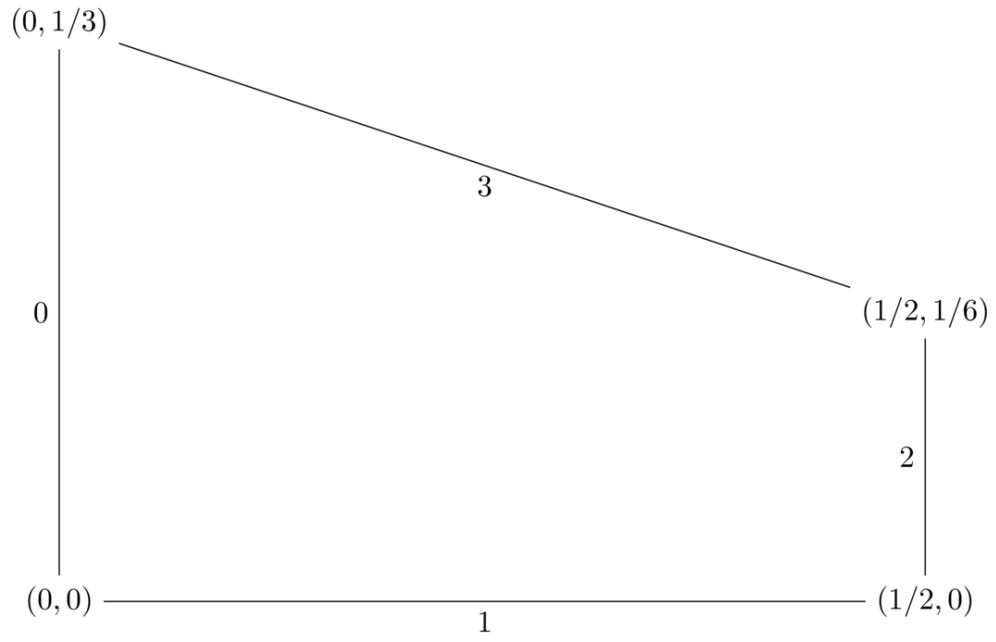
- Describe the Lemke Howson algorithm graphically
- Describe integer pivoting

1.6.3 Notes

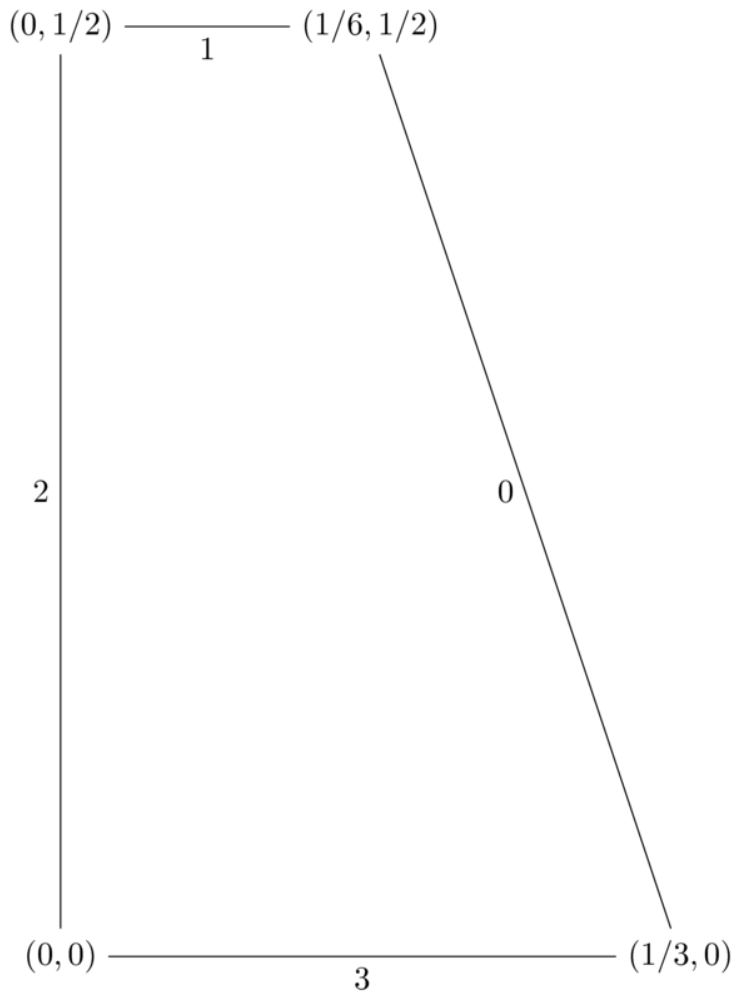
Describe the Lemke Howson algorithm

Using the polytopes for the matching pennies game:

$\mathcal{P}$:



$\mathcal{Q}$:



illustrate the Lemke Howson algorithm:

- $((0,0),(0,0))$ have labels $\{0,1\},\{2,3\}$. Drop 0 in $\mathcal{P}$.
- $\rightarrow ((1/2,0),(0,0))$ have labels $\{1,2\},\{2,3\}$. Drop 2 in $\mathcal{Q}$.
- $\rightarrow ((1/2,0),(1/3,0))$ have labels $\{1,2\},\{0,3\}$. Have an equilibria.

Try varying other initial labels. Is it possible to arrive at the mixed nash equilibrium?

Work through integer pivoting

Consider Rock Paper Scissors:

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}$$

We see that drawing this would be a pain. Let us build the tableaux, but let us first scale our payoffs so that all variables are positive (let us add 2 to all the utilities):

$$A = \begin{pmatrix} 2 & 1 & 3 \\ 3 & 2 & 1 \\ 1 & 3 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

We first need to scale the tableaux so that all variables are positive:

$$T_r = \begin{pmatrix} 2 & 1 & 3 & 1 & 0 & 0 & 1 \\ 3 & 2 & 1 & 0 & 1 & 0 & 1 \\ 1 & 3 & 2 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Following along in `numpy`:

```
>>> import numpy as np
>>> row_tableau = np.array([[2, 1, 3, 1, 0, 0, 1],
...                         [3, 2, 1, 0, 1, 0, 1],
...                         [1, 3, 2, 0, 0, 1, 1]])
```

and:

$$T_c = \begin{pmatrix} 1 & 0 & 0 & 2 & 1 & 3 & 1 \\ 0 & 1 & 0 & 3 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 3 & 2 & 1 \end{pmatrix}$$

Following along in `numpy`:

```
>>> col_tableau = np.array([[1, 0, 0, 2, 1, 3, 1],
...                         [0, 1, 0, 3, 2, 1, 1],
...                         [0, 0, 1, 1, 3, 2, 1]])
```

The labels of T_r are $\{0, 1, 2\}$ and the labels of T_c are $\{3, 4, 5\}$.

Let us drop label 0, so we pivot the first column of T_r. The minimum ratio test gives: $1/3 < 1/2 < 1/1$ so we pivot on the second row giving:

$$T_r = \begin{pmatrix} 0 & -1 & 7 & 3 & -2 & 0 & 1 \\ 3 & 2 & 1 & 0 & 1 & 0 & 1 \\ 0 & 7 & 5 & 0 & -1 & 3 & 2 \end{pmatrix}$$

Code:

```
>>> import nash
>>> dropped_label = nash.integer_pivoting.pivot_tableau(row_tableau,
...                                                     column_index=0)
>>> row_tableau
array([[ 0, -1,  7,  3, -2,  0,  1],
       [ 3,  2,  1,  0,  1,  0,  1],
       [ 0,  7,  5,  0, -1,  3,  2]])
```

This has labels $\{1, 2, 4\}$ so we need to drop label 4 by pivoting the fifth column of T_c. The minimum ratio test implies that we pivot on the third row.

$$T_c = \begin{pmatrix} 3 & 0 & -1 & 5 & 0 & 7 & 2 \\ 0 & 3 & -2 & 7 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 & 3 & 2 & 1 \end{pmatrix}$$

Code:

```
>>> dropped_label = nash.integer_pivoting.pivot_tableau(col_tableau,
...                                                     column_index=4)
>>> col_tableau
array([[ 3,  0, -1,  5,  0,  7,  2],
       [ 0,  3, -2,  7,  0, -1,  1],
       [ 0,  0,  1,  1,  3,  2,  1]])
```

This has labels $\{2, 3, 5\}$ so we need to drop 2 by pivoting the third row of T_r. The minimum ratio test implies that we pivot on the first row:

$$T_r = \begin{pmatrix} 0 & -1 & 7 & 3 & -2 & 0 & 1 \\ 21 & 15 & 0 & -3 & 9 & 0 & 6 \\ 0 & 54 & 0 & -15 & 3 & 21 & 9 \end{pmatrix}$$

Code:

```
>>> dropped_label = nash.integer_pivoting.pivot_tableau(row_tableau,
...                                                     column_index=2)
>>> row_tableau
array([[  0,  -1,   7,   3,  -2,   0,   1],
       [ 21,  15,   0,  -3,   9,   0,   6],
       [  0,  54,   0, -15,   3,  21,   9]])
```

This has labels $\{1, 3, 4\}$ so we need to drop 3 by pivoting the fourth column of T_c. The minimum ratio test implies that we pivot on the second row:

$$T_c = \begin{pmatrix} 21 & -15 & 3 & 0 & 0 & 54 & 9 \\ 0 & 3 & -2 & 7 & 0 & -1 & 1 \\ 0 & -3 & 9 & 0 & 21 & 15 & 6 \end{pmatrix}$$

Code:

```
>>> dropped_label = nash.integer_pivoting.pivot_tableau(col_tableau,
...                                                     column_index=3)
>>> col_tableau
array([[ 21, -15,   3,   0,   0,  54,   9],
       [  0,   3,  -2,   7,   0,  -1,   1],
       [  0,  -3,   9,   0,  21,  15,   6]])
```

This has labels $\{1, 2, 5\}$ so we need to drop 1 by pivoting the second column of T_r. The minimum ratio test implies that we pivot on the third row:

$$T_r = \begin{pmatrix} 0 & 0 & 378 & 147 & -105 & 21 & 63 \\ 1134 & 0 & 0 & 63 & 441 & -315 & 189 \\ 0 & 54 & 0 & -15 & 3 & 21 & 9 \end{pmatrix}$$

Code:

```
>>> dropped_label = nash.integer_pivoting.pivot_tableau(row_tableau,
...                                                     column_index=1)
>>> row_tableau
array([[   0,    0,  378,  147, -105,   21,   63],
       [1134,    0,    0,   63,  441, -315,  189],
       [   0,   54,    0,  -15,    3,   21,    9]])
```

This has labels $\{3, 4, 5\}$ so we need to drop 5 by pivoting the sixth column of T_c. The minimum ratio test implies that we pivot on the first row:

$$T_c = \begin{pmatrix} 21 & -15 & 3 & 0 & 0 & 54 & 9 \\ 21 & 147 & -105 & 378 & 0 & 0 & 63 \\ -315 & 63 & 441 & 0 & 1134 & 0 & 189 \end{pmatrix}$$

Code:

```
>>> dropped_label = nash.integer_pivoting.pivot_tableau(col_tableau,
...                                                     column_index=5)
>>> col_tableau
array([[  21,  -15,    3,    0,    0,   54,    9],
       [  21,  147, -105,  378,    0,    0,   63],
       [-315,   63,  441,    0, 1134,    0,  189]])
```

This has labels $\{0, 1, 2\}$ so we have a Nash equilibrium with vertices:

$$((189/1134, 9/54, 63/378), (63/378, 189/1134, 9/54)) = ((1/6, 1/6, 1/6, (1/6, 1/6, 1/6))$$

Which in turn corresponds to the expected equilibrium:

$$((1/3, 1/3, 1/3, (1/3, 1/3, 1/3))$$

- Mention how different pivoting methods are fine, doing it this way ensures everything is an integer which is also more efficient for a computer.

1.7 06 Repeated games

1.7.1 Corresponding chapters

- Repeated games

1.7.2 Objectives

- Define repeated games and strategies in repeated games
- Understand proof of theorem for sequence of stage Nash
- Understand that other equilibria exist

1.7.3 Notes

Playing repeated games in pairs

- Explain that we are about to play a game twice.
- Explain that this has to be done **SILENTLY**

In groups we are going to play:

$$A = \begin{pmatrix} \underline{12} & 6 \\ 0 & \underline{24} \\ \underline{12} & 23 \end{pmatrix} \qquad B = \begin{pmatrix} \underline{2} & 1 \\ \underline{5} & 4 \\ \underline{0} & 0 \end{pmatrix}$$

In pairs:

- Decide on row/column player (recall you don't care about your opponents reward).
- We are going to play the game TWICE and write down both players *cumulative* scores.
- Define a strategy and ask players to write down a strategy that must describe the what they do in both stages by answering the following question: - What should the player do in the first stage? - What should the player do in the second stage given knowledge of what both

players did in the first period?

- SILENTLY, after having written down a strategy: show each other your strategies and SILENTLY agree on the pair of utilities. If you are unable to agree on a utility this indicates that the strategies were not descriptive enough. SILENTLY start again :)

As a challenge: repeat this (so repeatedly play a repeated game, repeatedly write down a new strategy) and make a note when you arrive at an equilibria (where no one has a reason to write a different strategy down)

If anyone arrives at an equilibria where the row player scores more than 24 and the column player more than 4 stand up as a pair.

Following this, assuming a pair has arrived at such an equilibrium discuss this.

Then work through the notes:

- Definition of a repeated game;
- Definition of a strategy;
- Theorem of sequence of stage Nash (relate this back to the utilities of our game):
  - (r_1r_1, c_1c_1) with utility: (24, 4).
  - (r_1r_3, c_1c_1) with utility: (24, 2).
  - (r_3r_1, c_1c_1) with utility: (24, 2).
  - (r_3r_3, c_1c_1) with utility: (24, 0).

Now discuss the potential of a different equilibrium:

1. For the row player:

$$(\emptyset, \emptyset) \to r_2$$
$$(r_2, c_1) \to r_3$$
$$(r_2, c_2) \to r_1$$

2. For the column player:

$$(\emptyset, \emptyset) \to c_2$$
$$(r_1, c_2) \to c_1$$
$$(r_2, c_2) \to c_1$$
$$(r_3, c_2) \to c_1$$

This corresponds to the following scenario:

> Play (r_2, c_2) in first stage and (r_1, c_1) in second stage unless the column player does not cooperate in which case play (r_3, c_1).

This gives a utility of $(36, 6)$. Is this an equilibrium?

1. If the row player deviates, they would do so in the first round and gain no utility.
2. If the column player deviates, they would only be rational to do so in the first stage, if they did they would gain 1 but lose 2 in the second round.

Thus this is Nash equilibrium.

- Discuss how to identify such an equilibria: which player has an incentive to build a reputation? (The column player want to prove to be trustworthy to gain 2 in the final round).
- Mention how this shows how game theory studies the emergence of unexpected behaviour.

1.8 07 Prisoners Dilemma

1.8.1 Corresponding chapters

- Prisoners Dilemma

1.8.2 Objectives

- Explore the Iterated Prisoners Dilemma

1.8.3 Notes

Playing team based Iterated Prisoner's Dilemma

Show this video: https://www.youtube.com/watch?v=p3Uos2fzIJ0

Ask class to form 4 teams of equal size.

Write the following on the board:

Name	Score vs 1st opp	$\sum$ score vs 2nd opp	$\sum$ score vs 3rd opp

Explain that teams will play the iterated Prisoners dilemma:

$$A = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix}$$

Discuss NE of stage game.

Discuss "Cooperation" versus "Defection" and explain that goal is to maximise overall score (not necessarily beat direct opponent).

For every dual write the following on the board (assuming 5 turns):

Name	Score	$\sum$ score	$\sum$ score	$\sum$ score	$\sum$ score

Instructions for a dual:

- Give both teams copies of a `C and a D`.
- In your teams: discuss plans for a strategy.
- Before every stage invite both teams to talk to each other.
- Get teams to "face away", after a count down "show" (either a C or a D).

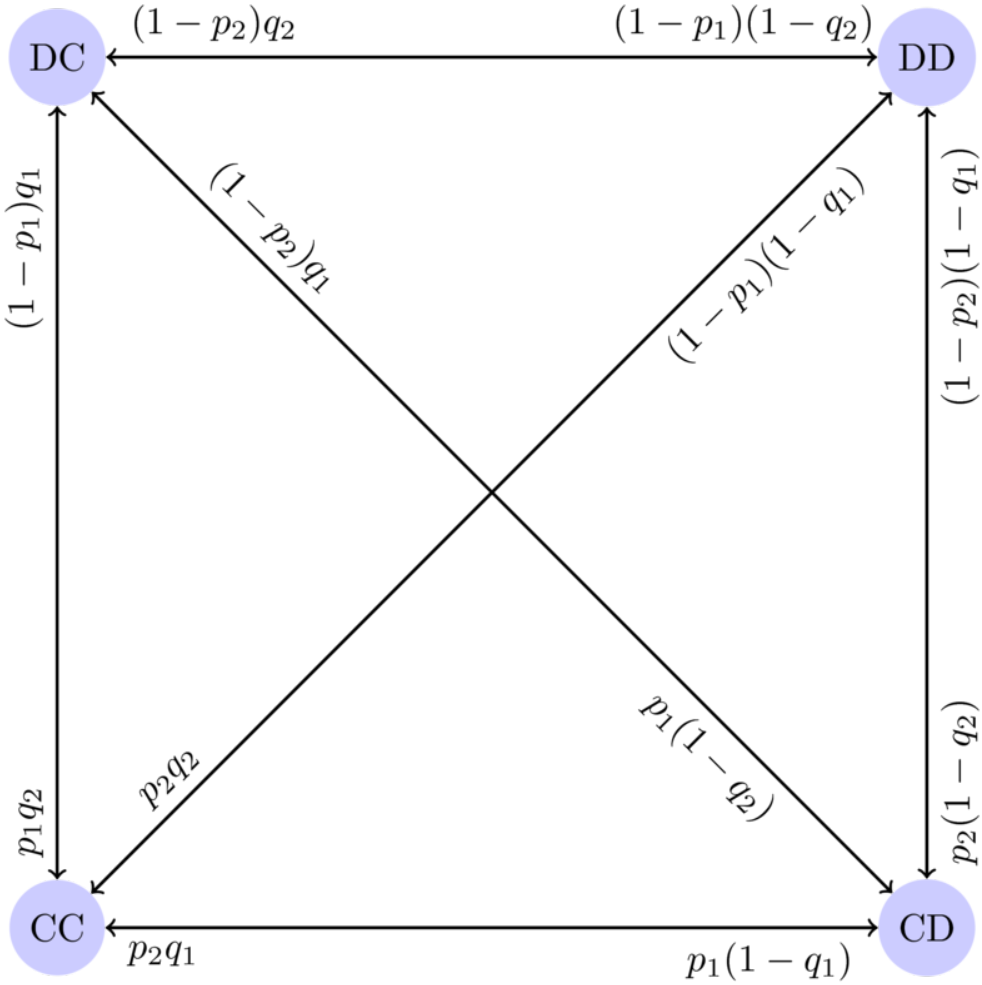
After the tournament:

- Discuss winning strategy and other interesting strategies. Discuss potential coalitions that arose.

- Discuss how teams had more information that usual.
- Invite the possibility of modifications (prob end, noise and lack of information).

Consider Reactive strategies

Go over theory.



In the case of $(p_1, q_1) = (1/4, 4/5)$ and $(p_2, q_2) = (2/5, 1/3)$ we have:

$$r_1 = \frac{11}{20} \qquad r_2 = \frac{1}{15}$$

$$s_1 = \frac{185}{311} \qquad s_2 = \frac{116}{311}$$

The steady state probabilities are given by:

$$\pi = (21460/96721, 36075/96721, 14616/96721, 24570/96721)$$

Here is some sympy code to illustrate this:

```
>>> import sympy as sym
>>> p_1, p_2 = sym.S(1) / sym.S(4), sym.S(4) / sym.S(5)
>>> q_1, q_2 = sym.S(2) / sym.S(5), sym.S(1) / sym.S(3)
```

(continues on next page)

(continued from previous page)

```
>>> r_1 = p_1 - p_2
>>> r_2 = q_1 - q_2
>>> r_1, r_2
(-11/20, 1/15)
>>> s_1 = (q_2 * r_1 + p_2) / (1 - r_1 * r_2)
>>> s_2 = (p_2 * r_2 + q_2) / (1 - r_1 * r_2)
>>> s_1, s_2
(185/311, 116/311)
>>> pi = sym.Matrix([[s_1 * s_2, s_1 * (1 - s_2), (1 - s_1) * s_2, (1 - s_1) * (1 - s_
↪2)]])
>>> pi
Matrix([[21460/96721, 36075/96721, 14616/96721, 24570/96721]])
```

We can verify that this is a steady state vector:

$$M = \begin{pmatrix} 1/10 & 3/20 & 3/10 & 9/20 \\ 8/25 & 12/25 & 2/25 & 3/25 \\ 1/12 & 1/6 & 1/4 & 1/2 \\ 4/15 & 8/15 & 1/15 & 2/15 \end{pmatrix}$$

$$\pi M = (21460/96721, 36075/96721, 14616/96721, 24570/96721)$$

Sympy code:

```
>>> M = sym.Matrix([[p_1*q_1, p_1*(1-q_1), (1-p_1)*q_1, (1-p_1)*(1-q_1)],
...                 [p_2 * q_1, p_2 * (1-q_1), (1-p_2) * q_1,  (1-p_2) * (1-q_1)],
...                 [p_1 * q_2, p_1 * (1-q_2),  (1-p_1) * q_2, (1-p_1) * (1-q_2)],
...                 [p_2 * q_2, p_2 * (1-q_2), (1-p_2) * q_2, (1-p_2)*(1-q_2)]])
>>> M
Matrix([
[1/10,  3/20, 3/10, 9/20],
[8/25, 12/25, 2/25, 3/25],
[1/12,   1/6,  1/4,  1/2],
[4/15,  8/15, 1/15, 2/15]])
>>> pi * M
Matrix([[21460/96721, 36075/96721, 14616/96721, 24570/96721]])
>>> pi * M == pi
True
```

The utility is then given by:

$$3s_1s_2 + 0s_1(1-s_2) + 5(1-s_1)s_2 + (1-s_1)(1-s_2) = 162030/96721 \approx 1.675$$

Sympy code:

```
>>> rstp = sym.Matrix([[sym.S(3), sym.S(0), sym.S(5), sym.S(1)]])
>>> score = pi.dot(rstp)
>>> score, float(score)
(162030/96721, 1.675...)
```

1.9 08 Evolutionary dynamics

1.9.1 Corresponding chapters

- Evolutionary dynamics

1.9.2 Objectives

- Go over basic differential models for population dynamics.

1.9.3 Notes

Explain difference from games so far:

- Players were discrete entities (and we only consider 2 player games);
- Now there are *no* players: just strategies in a population.

Our population will be *stand* ers or *sit* ers.

Say we are going to use the following differential equation to denote the rate at which people stand:

$$\frac{dx}{dt} = 2x$$

Use seconds (artificially) as a time unit, $\frac{dx}{dt}$ corresponds to the speed with which people stand up. We will instead consider it as *the rate at which people become standing up*. Using a continuous population these two things are equivalent.

Get one student to stand up.

The speed at which other individuals are becoming *standing up* is at that exact point 2. Get someone to slowly stand up ... until they become stood up etc...

Note how we will run out of people very quickly as the solution to our equation is:

$$x(t) = x_0 e^{2t}$$

Show how we can use Python to solve this differential equation numerically:

```
>>> import numpy as np
>>> from scipy.integrate import odeint
>>> t = np.linspace(0, 10, 100)  # Obtain 100 time points
>>> def dx(x, t, a):
...     """Define the derivate of x"""
...     return a * x
>>> a = 10
>>> xs = odeint(func=dx, y0=1, t=t, args=(a,))
>>> xs
array([[  1.000...
```

Now consider population of two individuals (see notes).

Discuss having students neither sitting or standing (who knows how? :)). What happens over time:

$$\frac{dx}{dt} = 2x \qquad \frac{dx}{dt} = 3x$$

This means that the we would very quickly run out of sitters and standers but what would happen to the limit of the ratio? (Go over mathematics in notes)

Now, finally consider a constant population:

Consider .5 population standing and .5 sitting.

What must we have for our rates?

One population increases at a rate of 2 and the other increases at a rate of 3.

So our population also, somehow decrease by 5.

Ask students to suggest how this can be done. Have a discussion.

Solution: share this between both populations:

- One populations "increases" at a rate of 2 - 2.5 = -0.5
- One populations "increases" at a rate of 3 - 2.5 = 0.5

Go over notes.

Discuss what happens at following starting conditions:

- x=1, y=0
- x=0, y=1
- x=0.5, y=0.5 but changes rates to be equal.

Show how can obtain this numerically:

```
>>> def dxy(xy, t, a, b):
...     """
...     Define the derivate of x and y.
...     It takes `xy` as a vector
...     """
...     x, y = xy
...     phi = a * x + b * y
...     return x * (a - phi), y * (b - phi)
>>> a, b = 10, 5
>>> xys = odeint(func=dxy, y0=[.5, .5], t=t, args=(a, b))
>>> xys
array([[  5.00000000e-01,   5.000...
```

Discuss how this basic notion of fitness will now be extended to consider game theoretic models where the fitness comes out game theoretic models.

1.10 09 Evolutionary game theory

1.10.1 Corresponding chapters

- Evolutionary game theory

1.10.2 Objectives

- Go over the logical progression from dynamics to games;
- Explain ESS

1.10.3 Notes

Recall that there are *no* players: just strategies in a population.

Go over the notes, explaining the mathematics.

Now consider the Matching pennies game:

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

This gives:

$$f_1 = x_1 - x_2 \qquad f_2 = -x_1 + x_2$$

which is equivalent to:

$$f = Ax = \begin{pmatrix} x_1 - x_2 \\ -x_1 + x_2 \end{pmatrix}$$

and so:

$$phi = fx = x_1(x_1 - x_2) + x_2(x_2 - x_1) = (x_1 - x_2)^2$$

thus:

$$\frac{dx}{dt} = \begin{pmatrix} x_1((x_1 - x_2) - (x_1 - x_2)^2) \\ x_2((x_2 - x_1) - (x_1 - x_2)^2) \end{pmatrix}$$

$$\frac{dx}{dt} = \begin{pmatrix} x_1(x_1 - x_2)(1 - (x_1 - x_2)) \\ x_2(x_1 - x_2)(-1 - (x_1 - x_2)) \end{pmatrix}$$

however recall that $x_1 + x_2 = 1$:

$$\frac{dx_1}{dt} = x_1(2x_1 - 1)2(1 - x_1))$$

Verification of above calculations:

```
>>> import sympy as sym
>>> x_1, x_2 = sym.symbols("x_1, x_2")
>>> f_1 = x_1 - x_2
>>> f_2 = x_2 - x_1
>>> phi = x_1 * f_1 + x_2 * f_2
>>> dx_1 = x_1 * (f_1 - phi)
>>> sym.factor(dx_1.subs({x_1: 1 - x_2}))
2*x_2*(x_2 - 1)*(2*x_2 - 1)
```

We have the stable points as expected:

1. $x_1 = 0$
2. $x_1 = 1$
3. $x_1 = 1/2$

Relate this to the notes and discuss notion of mutated population and stable ESS.

Show output of the following:

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from scipy.integrate import odeint
>>> t = np.linspace(0, 10, 100)  # Obtain 100 time points
>>> def dx(x, t, A):
...     """
...     Define the derivate of x.
...     """
...     f = np.dot(A, x)
...     phi = np.dot(f, x)
...     return x * (f - phi)
```

Slight deviation from $x_1 = 0$:

```
>>> A = np.array([[1, -1], [-1, 1]])
>>> epsilon = 10 ** -1
>>> xs = odeint(func=dx, y0=[epsilon, 1 - epsilon], t=t, args=(A,))
>>> plt.plot(xs);
[...
```

Slight deviation from $x_1 = 1$:

```
>>> xs = odeint(func=dx, y0=[1 - epsilon, epsilon], t=t, args=(A,))
>>> plt.plot(xs);
[...
```

Slight deviation from $x_1 = 1/2$:

```
>>> xs = odeint(func=dx, y0=[1 / 2 - epsilon, 1 / 2 + epsilon], t=t, args=(A,))
>>> plt.plot(xs);
[...
```

Look at theorem and discuss proof.

Carry out Nash equilibria computation (which corresponds to the above case):

```
>>> import nash
>>> game = nash.Game(A, A.transpose())
>>> list(game.support_enumeration())
[(array([ 1.,  0.]), array([ 1.,  0.])), (array([ 0.,  1.]), array([ 0.,  1.])),␣
↪(array([ 0.5,  0.5]), array([ 0.5,  0.5]))]
```

Now look at cases:

1. $x_1 = 0$: we are in the first case of the theorem so we have an ESS.
2. $x_1 = 1$: we are in the first case of the theorem so we have an ESS.
3. $x_1 = 1/2$: we now have the second case of the theorem $u(x, y) = u(x, x)$ (indeed the row strategy aims to make the column strategy indifferent - according to the best response condition).

To deal with this case we need to look at the next part of the second condition:

$$u(x, y) = 0$$

$$u(y, y) = y_1^2 + (1 - y_1)^2 - 2(1 - y_1)y_1 = (y_1 - (1 - y_1))^2 = (2y_1 - 1)^2$$

And as long as $y_1 \neq x_1 = 1/2$ then $u(x, y) < u(y, y)$ thus this is *not* an ESS:

```
>>> A = sym.Matrix(A)
>>> y_1, y_2 = sym.symbols("y_1, y_2")
>>> y = sym.Matrix([y_1, y_2])
>>> sym.factor((y.transpose() * A * y)[0].subs({y_2: 1 - y_1}))
1.0*(2.0*y_1 - 1.0)**2
```

Discuss the work John Maynard Smith in 1973 who formalised this work. Mention that he was not actually aware of Nash equilibria at the time.

1.11 10 Moran Processes

1.11.1 Corresponding chapters

- Moran Processes

1.11.2 Objectives

- Play a class activity of a Moran process;
- Define a Moran process;
- Prove theorem for formula of fixation probabilities;
- Numeric calculations.

1.11.3 Activity

Ask students to form groups of 4: this is a population.

Explain that players/individuals are either Hawks or Doves and that they will play the Hawk-Dove game with row matrix:

$$A = \begin{pmatrix} 0 & 3 \\ 1 & 2 \end{pmatrix}$$

Recall, this corresponds to sharing of 4 resources:

- Two hawks both get nothing;
- A hawk meeting a dove gets 3 out of 4.
- A dove meeting a hawk gets 1 out of 4.
- A dove meeting a dove gets 2 out of 4.

Hand out a 10 and 20 sided dice to each group.

Explain that one student is to be a Hawk and the others Doves.

Every student players each other and write down their total scores:

- The Hawk gets: 9 (3×3).
- The Doves all get: 5 ($1 \times 1 + 2 \times 2 = 5$).

One of the individuals is chosen to reproduce:

$$P(H) = \frac{9}{24} = 3/8 \qquad P(D) = \frac{9}{24} = 5/24$$

Use the dice (invite the students to figure out a way to do this, various different ways - ignore none throws etc. . .).

Then randomly choose a player to eliminate: **this can be the same player chosen to reproduce** (life can suck).

Now recalculate the fitness (every player plays everyone else).

Repeat until all players are of the same type.

Count from groups to obtain mean fixation rate of a Hawk.

Depending on time, potentially repeat this using Doves.

Now work through the notes: culminating in the proof of the theorem for the absorption probabilities of a birth death process.

Discuss and use code from chapter to show the fixation with the Hawk Dove game:

```
>>> import numpy as np
>>> A = np.array([[0, 3], [1, 2]])
```

Calculate theoretic value using formula from theorem:

$$f_{1i} = \frac{3(N-i)}{N-1} = 3\frac{N-i}{N-1} \tag{1.25}$$

$$f_{2i} = \frac{i + 2(N-i-1)}{N-1} = \frac{2N-2-i}{N-1} \tag{1.26}$$

(1.27)

This gives (for $N = 4$):

	$i=1$	$i=2$	$i=3$
f_{1i}	3	2	1
f_{2i}	5/3	4/3	1
γ_i	5/9	2/3	1

Thus:

$$x_1 = \frac{1}{1 + 5/9 + 5/9 \times 2/3 + 5/9 \times 2/3 \times 1} = \frac{1}{62/27} = \frac{27}{62} \approx .44$$

- Discuss work of Maynard smith but that this actually used Hawk Dove game in infinite population games.
- Discussion possibility for using a utility model on top of fitness.
- A lot of current work looks at Moran processes: a good model of invasion of a specifies etc...
- The Prisoners dilemma can also be included, there is documentation about simulating this with Axelrod is here: http://axelrod.readthedocs.io/en/stable/tutorials/getting_started/moran.html

1.12 11 Contemporary research

1.12.1 Corresponding chapters

- Contemporary research

1.12.2 Objectives

- Discuss different papers that will be looked at.
- Answer queries about assessment of these papers.

Explain that in future sessions we will discuss the papers, students will be expected to have read them. This will allow me to answer any questions they might have.

TODO: Write my own notes on each paper.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

CHAPTER 3

Indices and tables

- genindex
- modindex
- search