Computing Nash Equilibrium in 2-Player Games

CMPUT 355: Games, Puzzles, and Algorithms

Lecture Outline

1. Logistics & Recap
2. Computing 2-player Nash equilibrium by hand
3. Linear programming
4. Computing 2-player Nash equilibrium by linear programming

Logistics

- **Practice questions #5** released last Friday
  - Solutions were posted yesterday
- **Quiz #5** is **Friday** (Mar 27)
  - Covers up to the end of **last Friday's lecture** (AND/OR & rock-paper-scissors)
- **Quiz #4** marks are posted

Recap: Normal Form Games

- Rock Paper Scissors is an example of a **normal form game**
- Players choose actions **simultaneously**
- Each possible **combination** of actions maps to scores for the players ("**utilities**")
- To specify a normal form game, list:
  - **Who** the players are (need not be just 2)
  - What **actions** each player has available (need not be the same per player)
  - The **utility** to each player for each combination of actions (often a table for 2-player games)
- Unlike **perfect information alternating moves**, the optimal strategy for a normal form game may require **randomization** (i.e., **mixed strategies**)

	Rock	Paper	Scissor
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissor	-1,1	1,-1	0,0

Recap: Expected Utility

- Suppose column plays a **mixed strategy** s_c
  - i.e., a **distribution** that chooses action b with probability $s_c(b)$
- Row's aim is to maximize their **expected utility**:

$$u_r(a, s_c) = \mathbb{E}_{b \sim s_c} u_r(a, b) = \sum_b \Pr(b) u_r(a, b) = \sum_b s_c(b) u_r(a, b).$$

- Row might **themselves** be playing a mixed strategy s_r:

$$u_r(s_r, s_c) = \mathbb{E}_{\substack{a \sim s_r \\ b \sim s_c}} \left[u_r(a, b) \right] = \sum_a \Pr(a) u_r(a, s_c)$$

$$= \sum_a s_r(a) \mathbb{E}_{b \sim s_c} u_r(a, b) = \sum_a s_r(a) \sum_b s_c(b) u_r(a, b)$$

Recap: Nash Equilibrium

- A **Nash equilibrium** is a profile of strategies in which every player simultaneously best responds to all the other players
- In the two-player case, s, t is a Nash equilibrium when:
  1. s is a best response to t; i.e., $u_r(s, t) \geq u_r(a, t)$ for all actions a
  2. t is a best response to s; i.e., $u_c(s, t) \geq u_c(s, b)$ for all actions b

Recap: Best Response & Indifference

Theorem: Suppose that s is a **best response** for player i to strategy t, and there exist actions $a \neq b$ such that $s(a) > 0$ and $s(b) > 0$ (i.e., s is a **mixed strategy**).

Then i is **indifferent** between a and b; that is,

$$u_i(s, t) = u_i(a, t) = u_i(b, t).$$

- *Implication:* If (s, t) is a Nash equilibrium, then every action that s plays with positive probability is a best response
- Mixed strategies in equilibrium only "mix over" best responses

Computing Pure Strategy Nash Equilibrium by Hand

- In a Nash equilibrium, every player simultaneously **best responds** to the other
- So you can check whether a given **pure strategy profile** is an **equilibrium** simply by looking for **deviations**:
  - Given Row's strategy, is there a different strategy that gives Column a **strictly greater** utility?
  - Given Column's strategy, is there a different strategy that gives a Row a **strictly greater** utility?
- In a sufficiently small game, you can just look at **every possible** strategy profile and **check** whether it is an equilibrium
- **Question:** What is the **time-complexity** of that approach?
- **Question:** What are the pure strategy Nash equilibria of Ballet-Soccer-Dishes?
- **Question:** What are the pure strategy Nash equilibria of Matching Pennies?

	Ballet	Soccer	Dishes
Ballet	2,1	0,0	0,0
Soccer	0,0	1,2	0,0
Dishes	0,0	0,0	1,1

	Head	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

Computing Mixed Strategy Nash Equilibrium by Hand

- Some games have **only mixed** strategy Nash equilibria
- Some games have **both** pure and mixed strategy Nash equilibria
- To compute a mixed equilibrium:
  1. Guess the **support** of the equilibrium (**what is support?**)
     - Recall that in every mixed equilibrium, every action in the support of one player's strategy must have equal utility given the other player's strategy
  2. Derive the strategy that Row would have to play to make Column indifferent among all of the actions in their support
     - and vice versa
  3. Verify that both are best responses (**how could they not be?**)

	Ballet	Soccer	Dishes
Ballet	2,1	0,0	0,0
Soccer	0,0	1,2	0,0
Dishes	0,0	0,0	1,1

	Head	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

Example: Computing Mixed Strategy Nash Equilibria by Hand for Matching Pennies

	Head	Tails
Heads	1, -1	-1, 1
Tails	-1, 1	1, -1

1. Let's guess that the **support** for both players is both actions (**why?**)
   - So Column plays $s_c(H) = p$ and $s_c(T) = 1 - p$
2. What value of p makes Row **indifferent** between Heads and Tails?

$$
\begin{aligned}
& u_r(H, s_c) = u_r(T, s_c) \\
\iff\ & p u_r(H,H) + (1-p) u_r(H,T) = p u_r(T,H) + (1-p) u_r(T,T) \\
\iff\ & p(1) + (1-p)(-1) = p(-1) + (1-p)1 \\
\iff\ & 1p - 1 + p = -p + 1 - p \\
\iff\ & 4p = 2 \quad \iff\ p = \frac{1}{2}.
\end{aligned}
$$

3. **Verify** best response: $u_r(H, s_c) = u_r(T, s_c) = 0$ ✅

Same procedure gives the same answer for Column; so it is a Nash equilibrium for both players to uniformly randomize over both actions.

Example: Computing Mixed Strategy Nash Equilibria by Hand for **Asymmetric** Matching Pennies

	Head	Tails
Heads	**7**, -1	-1, 1
Tails	-1, 1	1, -1

1. Let's guess that the **support** for both players is both actions
   - So Column plays $s_c(H) = p$ and $s_c(T) = 1 - p$
2. What value of p makes Row **indifferent** between Heads and Tails?

$$
\begin{aligned}
& u_r(H, s_c) = u_r(T, s_c) \\
\iff & p u_r(H,H) + (1-p) u_r(H,T) = p u_r(T,H) + (1-p) u_r(T,T) \\
\iff & p(7) + (1-p)(-1) = p(-1) + (1-p)1 \\
\iff & 7p - 1 + p = -p + 1 - p \\
\iff & 10p = 2 \iff p = \frac{1}{5}.
\end{aligned}
$$

3. **Verify** best response: $u_r(H, s_c) = u_r(T, s_c) = \frac{3}{5}$ ✅

- So (s_r, s_c) is a Nash equilibrium, where
  - $s_r(H) = 0.5, s_r(T) = 0.5$
  - $s_c(H) = 0.2, s_c(T) = 0.8$

Row plays $s_r(H) = q$ and $s_r(T) = 1 - q$

2. What value of q makes **Column** indifferent between Heads and Tails?

$$
\begin{aligned}
& u_c(s_r, H) = u_c(s_r, T) \\
\iff & q u_c(H,H) + (1-q) u_c(T,H) = q u_c(H,T) + (1-q) u_c(T,T) \\
\iff & q(-1) + (1-q)(1) = q(1) + (1-q)(-1) \\
\iff & -q + 1 - q = q - 1 + q \\
\iff & 2 = 4q \iff q = \mathbf{\frac{1}{2}}!
\end{aligned}
$$

3. Verify best response: $u_c(s_r, H) = u_c(s_r, T) = 0$ ✅

Notice that **Column's strategy** is different from the last slide, even though **Row's utilities** are the ones that changed! (**why?**)

Linear Programming

Definition:
A **linear program** consists of

- A set of real-valued **variables** $\{x_1, \ldots, x_n\}$
- A linear **objective function** defined by **weights** $\{w_1, \ldots, w_n\}$
- A set of linear **constraints** of the form $\Sigma_{j=1}^{n} a_j x_j \leq b$

Sample:

$$\text{maximize} \sum_{j=1}^{n} w_j x_j$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall 1 \leq i \leq m$$

$$x_j \geq 0 \qquad \forall 1 \leq j \leq n$$

Linear Program Properties

$$\text{maximize} \sum_{j=1}^{n} w_j x_j$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall 1 \leq i \leq m$$

$$x_j \geq 0 \qquad \forall 1 \leq j \leq n$$

- Linear programs can be solved in **polynomial time** by generic algorithms (e.g., ellipsoid algorithm)
  - So writing a problem as a linear program constitutes a **proof** that it is solvable in polynomial time
- Negating weights w_j allows us to **minimize** or **maximize** the objective
- Negating constraint coefficients a_{ij} allows for **greater-than-or-equal** constraints
- Providing both greater-than-or-equal and less-than-or-equal constraints allows for **equality constraints**
- **Cannot** always express **strict inequalities** (although there are tricks)

Computing Nash Equilibrium: Zero-Sum Games

$$\begin{aligned}
\text{minimize } & U_1^* \\
\text{subject to } & \sum_{a_2 \in A_2} u_1(a_1, a_2) s_2(a_2) \leq U_1^* && \forall a_1 \in A_1 \\
& \sum_{a_2 \in A_2} s_2(a_2) = 1 \\
& s_2(a_2) \geq 0 && \forall a_2 \in A_2
\end{aligned}$$

- This linear program computes U_1^*, player 1's **minmax value**, and s_2, player 2's **minmax strategy** against player 1
  - By the minimax theorem, this is player 2's **equilibrium strategy**
- Compute player 1's equilibrium strategy analogously

Computing Nash Equilibrium: Two-Player, General Sum Games

- If we already know the **support** σ_i, σ_{-i} of the equilibrium, then we can compute it efficiently in a two-player game
- In the following, let A_i be player i's set of actions, and let player $-i$ be player i's opponent

$$\sum_{a_{-i} \in \sigma_{-i}} s_{-i}(a_{-i}) u_i(a_i, a_{-i}) = v_i \qquad \forall i \in \{1,2\}, a_i \in \sigma_i$$

$$\sum_{a_{-i} \in \sigma_{-i}} s_{-i}(a_{-i}) u_i(a_i, a_{-i}) \leq v_i \qquad \forall i \in \{1,2\}, a_i \notin \sigma_i$$

$$s_i(a_i) \geq 0 \qquad \forall i \in \{1,2\}, a_i \in \sigma_i$$

$$s_i(a_i) = 0 \qquad \forall i \in \{1,2\}, a_i \notin \sigma_i$$

$$\sum_{a_i \in A_i} s_i(a_i) = 1 \qquad \forall i \in \{1,2\}$$

Questions:

1. Why can't we just set $\sigma_i = A_i$ for every agent and solve **once**?
2. Why can't we just try **every possible support**?
3. Why wouldn't this work for ***n*-player** games?

Summary

- In general, computing Nash equilibrium is **intractable** ("$PPAD$-complete")
- However, some **special cases** can be easier:
  - Zero-sum games
  - Small two-player games
- **Zero-sum games:** Find a minimax strategy for each player
  - Polynomial time using **linear programming**
- **Small two-player games:** Guess a support and solve indifference equations
  - Our by-hand examples had support size of two, yielding a single equation per player
  - In general you have to solve a **system of equations** (depending on number of actions in the support)
  - *For a given support*, can also solve this using **linear programming**