Mixed Strategies and Nash Equilibrium

Game Theory Course:
Jackson, Leyton-Brown & Shoham
Mixed Strategies

• It would be a pretty bad idea to play any deterministic strategy in matching pennies
• Idea: confuse the opponent by playing randomly
• Define a strategy \( s_i \) for agent \( i \) as any probability distribution over the actions \( A_i \).
  - pure strategy: only one action is played with positive probability
  - mixed strategy: more than one action is played with positive probability
    - these actions are called the support of the mixed strategy

• Let the set of all strategies for \( i \) be \( S_i \)
• Let the set of all strategy profiles be \( S = S_1 \times \ldots \times S_n \).
Utility under Mixed Strategies

• What is your *payoff* if all the players follow mixed strategy profile $s \in S$?
  • We can’t just read this number from the game matrix anymore: we won’t always end up in the same cell
Utility under Mixed Strategies

- What is your payoff if all the players follow mixed strategy profile \( s \in S \)?

  - We can’t just read this number from the game matrix anymore: we won’t always end up in the same cell

- Instead, use the idea of expected utility from decision theory:

\[
    u_i(s) = \sum_{a \in A} u_i(a) Pr(a|s)
\]

\[
    Pr(a|s) = \prod_{j \in N} s_j(a_j)
\]
Best Response and Nash Equilibrium

Our definitions of best response and Nash equilibrium generalize from actions to strategies.

Definition (Best response)

\( s_i^* \in BR(s_{-i}) \) iff \( \forall s_i \in S_i, u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}) \)

Definition (Nash equilibrium)

\( s = \langle s_1, \ldots, s_n \rangle \) is a Nash equilibrium iff \( \forall i, s_i \in BR(s_{-i}) \)
Our definitions of best response and Nash equilibrium generalize from actions to strategies.

**Definition (Best response)**

\[ s_i^* \in BR(s_{-i}) \text{ iff } \forall s_i \in S_i, u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}) \]

**Definition (Nash equilibrium)**

\[ s = \langle s_1, \ldots, s_n \rangle \text{ is a Nash equilibrium iff } \forall i, s_i \in BR(s_{-i}) \]

**Theorem (Nash, 1950)**

*Every finite game has a Nash equilibrium.*
• It’s hard in general to compute Nash equilibria, but it’s easy when you can guess the support
• For BoS, let’s look for an equilibrium where all actions are part of the support
Computing Mixed Nash Equilibria

Battle of the Sexes

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2, 1</td>
<td>0, 0</td>
</tr>
<tr>
<td>F</td>
<td>0, 0</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

- Let player 2 play $B$ with $p$, $F$ with $1 - p$.
- If player 1 best-responds with a mixed strategy, player 2 must make him indifferent between $F$ and $B$ (why?)
Computing Mixed Nash Equilibria

Battle of the Sexes

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2, 1</td>
<td>0, 0</td>
</tr>
<tr>
<td>F</td>
<td>0, 0</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

- Let player 2 play $B$ with $p$, $F$ with $1 - p$.
- If player 1 best-responds with a mixed strategy, player 2 must make him indifferent between $F$ and $B$ (why?)

$$u_1(B) = u_1(F)$$

$$2p + 0(1 - p) = 0p + 1(1 - p)$$

$$p = \frac{1}{3}$$
Computing Mixed Nash Equilibria

Battle of the Sexes

\[
\begin{array}{cc}
B & F \\
B & 2,1 & 0,0 \\
F & 0,0 & 1,2 \\
\end{array}
\]

- Likewise, player 1 must randomize to make player 2 indifferent.
- Why is player 1 willing to randomize?
Likewise, player 1 must randomize to make player 2 indifferent.

Why is player 1 willing to randomize?

Let player 1 play $B$ with $q$, $F$ with $1 - q$.

$$u_2(B) = u_2(F')$$

$$q + 0(1 - q) = 0q + 2(1 - q)$$

$$q = \frac{2}{3}$$

Thus the mixed strategies $(\frac{2}{3}, \frac{1}{3})$, $(\frac{1}{3}, \frac{2}{3})$ are a Nash equilibrium.
Interpreting Mixed Strategy Equilibria

What does it mean to play a mixed strategy? Different interpretations:

- Randomize to *confuse* your opponent
  - consider the matching pennies example

- Randomize when *uncertain* about the other’s action
  - consider battle of the sexes

- Mixed strategies are a concise description of what might happen in *repeated play*: count of pure strategies in the limit

- Mixed strategies describe *population dynamics*: 2 agents chosen from a population, all having deterministic strategies. MS gives the probability of getting each PS.
## Example - Soccer Penalty Kicks

<table>
<thead>
<tr>
<th>Kicker/Goalie</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>0, 1</td>
<td>1, 0</td>
</tr>
<tr>
<td>Right</td>
<td>1, 0</td>
<td>0, 1</td>
</tr>
</tbody>
</table>

Game Theory Course: Jackson, Leyton-Brown & Shoham
### Example - Soccer Penalty Kicks

<table>
<thead>
<tr>
<th>Kicker/Goalie</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>0, 1</td>
<td>1, 0</td>
</tr>
<tr>
<td>Right</td>
<td>.75, .25</td>
<td>0, 1</td>
</tr>
</tbody>
</table>

Game Theory Course: Jackson, Leyton-Brown & Shoham

Example: Mixed Strategy Nash.
Mixed Strategies and Nash Equilibrium

Game Theory Course: Jackson, Leyton-Brown & Shoham
Hardness beyond $2 \times 2$ games

Algorithms

Two example algorithms for finding NE

- LCP (Linear Complementarity) formulation
  - [Lemke-Howson ’64]
- Support Enumeration Method
  - [Porter et al. ’04]
The Lemke-Howson Algorithm

CPSC 532L
Lecture Overview

1. Linear Programming

2. Lemke-Howson Algorithm
A linear program is defined by:

- a set of real-valued variables
- a linear objective function
  - a weighted sum of the variables
- a set of linear constraints
  - the requirement that a weighted sum of the variables must be greater than or equal to some constant
Given $n$ variables and $m$ constraints, variables $x$ and constants $w$, $a$ and $b$:

\[
\text{maximize } \sum_{i=1}^{n} w_i x_i \\
\text{subject to } \sum_{i=1}^{n} a_{ij} x_i \leq b_j \quad \forall j = 1 \ldots m
\]

- These problems can be solved in polynomial time using interior point methods.
  - Interestingly, the (worst-case exponential) simplex method is often faster in practice.
Lecture Overview

1 Linear Programming

2 Lemke-Howson Algorithm
Two-player equilibrium constraints

\[
\sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \\
\sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \\
\sum_{a_i \in A_i} s_i(a_i) = 1 \\
s_i(a_i) \geq 0 \\
r_i(a_i) \geq 0 \\
r_i(a_i) \cdot s_i(a_i) = 0
\]

- We can write down a set of constraints that a two player strategy profile satisfies if and only if it is a Nash equilibrium.
Two-player equilibrium constraints

\[ \sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \quad \forall a_1 \in A_1 \]

\[ \sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \quad \forall a_2 \in A_2 \]

\[ \sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in N \]

\[ s_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i \]

\[ r_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i \]

\[ r_i(a_i) \cdot s_i(a_i) = 0 \quad \forall i \in N, a_i \in A_i \]

- \( U_i^* \) is the utility of \( i \)'s best responses.
- \( s_i(a_i) \) is the probability that \( i \) plays \( a_i \).
- \( r_i(a_i) \) is a "slack" variable.
- Each \( u_i(a_i, a_{-i}) \) is a constant.
Two-player equilibrium constraints

\[ \sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \quad \forall a_1 \in A_1 \]

\[ \sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \quad \forall a_2 \in A_2 \]

\[ \sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in N \]

\[ s_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i \]

\[ r_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i \]

\[ r_i(a_i) \cdot s_i(a_i) = 0 \quad \forall i \in N, a_i \in A_i \]

- \( s_1 \) and \( s_2 \) are valid probability distributions.
Two-player equilibrium constraints

\[
\sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \quad \forall a_1 \in A_1
\]

\[
\sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \quad \forall a_2 \in A_2
\]

\[
\sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in N
\]

\[
s_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \cdot s_i(a_i) = 0 \quad \forall i \in N, a_i \in A_i
\]

- Slack variables \( r_i(a_i) \) are non-negative.
- \( U_1^* \) is weakly greater than the EU of any of player 1’s actions, given \( s_2 \ldots \).
Two-player equilibrium constraints

\[
\sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \quad \forall a_1 \in A_1
\]

\[
\sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \quad \forall a_2 \in A_2
\]

\[
\sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in N
\]

\[
s_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \cdot s_i(a_i) = 0 \quad \forall i \in N, a_i \in A_i
\]

- Slack variables \( r_i(a_i) \) are non-negative.
- \( U_1^* \) is weakly greater than the EU of any of player 1’s actions, given \( s_2 \ldots \)
- and exactly equal to the EU of every action in the support.
Two-player equilibrium constraints

\[
\sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \quad \forall a_1 \in A_1
\]

\[
\sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \quad \forall a_2 \in A_2
\]

\[
\sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in N
\]

\[
s_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \cdot s_i(a_i) = 0 \quad \forall i \in N, a_i \in A_i
\]

• So we’re done! Or are we?
Two-player equilibrium constraints

\[
\sum_{a_2 \in A_2} u_1(a_1, a_2) \cdot s_2(a_2) + r_1(a_1) = U_1^* \quad \forall a_1 \in A_1
\]

\[
\sum_{a_1 \in A_1} u_2(a_1, a_2) \cdot s_1(a_1) + r_2(a_2) = U_2^* \quad \forall a_2 \in A_2
\]

\[
\sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in N
\]

\[
s_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \geq 0 \quad \forall i \in N, a_i \in A_i
\]

\[
r_i(a_i) \cdot s_i(a_i) = 0 \quad \forall i \in N, a_i \in A_i
\]

- So we’re done! Or are we?
- This requirement changes the problem from a linear program to a linear complementarity program.
- Unfortunately, there is no general algorithm for solving LCPs.
The Lemke-Howson algorithm is a specialized algorithm for solving the previous LCP. It uses a concept of labels on mixed strategies.
The Lemke-Howson algorithm is a specialized algorithm for solving the previous LCP. It uses a concept of labels on mixed strategies.

**Definition (Labels)**

Every possible mixed strategy $s_i$ is given a set of labels $L(s_i) \subseteq A_1 \cup A_2$. The strategy $s_i$ has the following labels:

- Every action $a_i \in A_i$ satisfying $s_i(a_i) = 0$, and
- Every action $a_{-i} \in A_{-i}$ such that $a_{-i} \in BR_{-i}(s_i)$. 

A pair of strategies $(s_1, s_2)$ is a Nash equilibrium iff it is completely labelled: $L(s_1) \cup L(s_2) = A_1 \cup A_2$. 

The Lemke-Howson Algorithm CPSC 532L, Slide 7
The Lemke-Howson algorithm is a specialized algorithm for solving the previous LCP. It uses a concept of labels on mixed strategies.

**Definition (Labels)**

Every possible mixed strategy $s_i$ is given a set of labels $L(s_i) \subseteq A_1 \cup A_2$. The strategy $s_i$ has the following labels:

- Every action $a_i \in A_i$ satisfying $s_i(a_i) = 0$, and
- Every action $a_{-i} \in A_{-i}$ such that $a_{-i} \in BR_{-i}(s_i)$.

A pair of strategies $(s_1, s_2)$ is a Nash equilibrium iff it is completely labelled: $L(s_1) \cup L(s_2) = A_1 \cup A_2$. 
The Lemke-Howson algorithm can be understood as searching the two spaces of labelled strategies for a fully-labelled pair. When the game is nondegenerate*, there are no strategies with more labels than an agent has actions. So a completely labelled pair of strategies must consist of a pair that has no labels in common.
The LCP formulation allows us to define a pivot operation, which is able to take a labelled strategy and return a new one that differs in exactly one label.
The LCP formulation allows us to define a pivot operation, which is able to take a labelled strategy and return a new one that differs in exactly one label.

**Basic strategy:**
1. Start at the completely-labelled “synthetic equilibrium” \((0, 0)\).
2. Pivot to a new \(s_1\); its new label must duplicate a label of \(s_2\).
Pivoting

- The LCP formulation allows us to define a pivot operation, which is able to take a labelled strategy and return a new one that differs in exactly one label.

- Basic strategy:
  1. Start at the completely-labelled “synthetic equilibrium” \((0, 0)\).
  2. Pivot to a new \(s_1\); its new label must duplicate a label of \(s_2\).
  3. Repeat:
     1. Pivot to a new strategy to remove the duplicated label (the “leaving” label).
     2. If the new label (the “entering” label) is a duplicate, continue.
     3. Otherwise, the “missing” label must have been found. Halt.
Lemke-Howson properties

- Only works on 2-player games. (why?)
- Guaranteed to find at least one equilibrium.
- **Not** guaranteed to find all equilibria.
- May require exponentially many pivots.
- Quite fast in practice.
The basic idea behind SEM

- If you “guess” the right support, finding an equilibrium only requires solving a system of polynomial inequalities.
- In practice, tools like MINOS [Murtagh, Saunders, 2010] solve these systems quickly.
- To find one (or all) Nash equilibra, just enumerate supports.
Hardness beyond $2 \times 2$ games

Support Enumeration Method: Porter et al. 2004

- Step 1: Finding a NE with a specific support

\[
\begin{align*}
\sum_{a_{-1} \in \sigma_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) &= v_i & \forall i \in \{1, 2\}, a_i \in \sigma_i \\
\sum_{a_{-1} \in \sigma_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) &\leq v_i & \forall i \in \{1, 2\}, a_i \notin \sigma_i \\
p_i(a_i) &\geq 0 & \forall i \in \{1, 2\}, a_i \notin \sigma_i \\
p_i(a_i) &= 0 & \forall i \in \{1, 2\}, a_i \notin \sigma_i \\
\sum_{a_i \in \sigma_i} p_i(a_i) &= 1 & \forall i \in \{1, 2\}
\end{align*}
\]
The ideas that make SEM fast

(1) The size of the tree
(2) Dominance
(3) Test Given Support (TGS)