Computing Solution Concepts

CMPT 882
Computational Game Theory
Simon Fraser University
Spring 2010
Instructor: Oliver Schulte

Notation

- a_i^j : action (pure strategy) number *j* of player *i*.
- a_{-i}: generic tuple of actions for players other than *i*.
- s_i^j : probability that mixed strategy s_i assigns to action number *j* of player *i*.
- $\Sigma_k\, u_1(a_1^j, a_2^k) \cdot s_2^k$ is the utility for player 1 of playing pure strategy a_1^j against mixed strategy s_2^k.

Linear Programming

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

Linear Programming

Given n variables and m constraints, variables x and constants w, a and b:

$$\text{maximize} \quad \sum_{i=1}^{n} w_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{n} a_{ij} x_i \leq b_j \qquad \forall j = 1 \ldots m$$

$$x_i \in \{0, 1\} \qquad \forall i = 1 \ldots n$$

- 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

Computing equilibria of zero-sum games

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

- First, identify the variables:
  - U_1^* is the expected utility for player 1
  - $s_2^{a_2}$ is player 2's probability of playing action a_2 under his mixed strategy
- each $u_1(a_1, a_2)$ is a constant.

Computing equilibria of zero-sum games

Now let's interpret the LP:

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

- s_2 is a valid probability distribution.

Computing equilibria of zero-sum games

Now let's interpret the LP:

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

- U_1^* is as small as possible.

Computing equilibria of zero-sum games

Now let's interpret the LP:

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

- Player 1's expected utility for playing each of his actions under player 2's mixed strategy is no more than U_1^*.
  - Because U_1^* is minimized, this constraint will be tight for some actions: the support of player 1's mixed strategy.

Computing equilibria of zero-sum games

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

- This formulation gives us the minmax strategy for player 2.
- To get the minmax strategy for player 1, we need to solve a second (analogous) LP.

Computing Maxmin Strategies in General-Sum Games

Let's say we want to compute a maxmin strategy for player 1 in an arbitrary 2-player game G.

Constraints for determining whether s_i is strictly dominated by any mixed strategy

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) > u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

Constraints for determining whether s_i is strictly dominated by any mixed strategy

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) > u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

- What's wrong with this program?

Constraints for determining whether s_i is strictly dominated by any mixed strategy

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) > u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

- What's wrong with this program?
  - strict inequality in the first constraint means we don't have an LP

LP for determining whether s_i is strictly dominated by any mixed strategy

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in A_i} p_j \\ \text{subject to} \quad & \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\ & p_j \geq 0 && \forall j \in A_i \end{aligned}$$

- This is clearly an LP. Why is it a solution to our problem?

LP for determining whether s_i is strictly dominated by any mixed strategy

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in A_i} p_j \\ \text{subject to} \quad & \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i} \\ & p_j \geq 0 \qquad \forall j \in A_i \end{aligned}$$

- This is clearly an LP. Why is it a solution to our problem?
  - if a solution exists with $\sum_j p_j < 1$ then we can add $1 - \sum_j p_j$ to some p_k and we'll have a dominating mixed strategy (since utility was assumed to be positive everywhere)
- Our original approach works for very weak domination
- For weak domination we can use that program with a different objective function trick.

Identifying strategies that survive iterated elimination

- This can be done by repeatedly solving our LPs: solving a polynomial number of LPs is still in $\mathcal{P}$.
  - Checking whether every pure strategy of every player is dominated by any other mixed strategy requires us to solve at worst $\sum_{i \in N} |A_i|$ linear programs.
  - Each step removes one pure strategy for one player, so there can be at most $\sum_{i \in N} (|A_i| - 1)$ steps.
  - Thus we need to solve $O((n \cdot \max_i |A_i|)^2)$ linear programs.

Computing CE

$$\sum_{a \in A | a_i \in a} p(a) u_i(a) \geq \sum_{a \in A | a_i \in a} p(a) u_i(a'_i, a_{-i}) \quad \forall i \in N, \forall a_i, a'_i \in A_i$$

$$p(a) \geq 0 \quad \forall a \in A$$

$$\sum_{a \in A} p(a) = 1$$

- variables: $p(a)$; constants: $u_i(a)$

Computing CE

$$\sum_{a \in A | a_i \in a} p(a) u_i(a) \geq \sum_{a \in A | a_i \in a} p(a) u_i(a'_i, a_{-i}) \quad \forall i \in N, \forall a_i, a'_i \in A_i$$

$$p(a) \geq 0 \qquad \forall a \in A$$

$$\sum_{a \in A} p(a) = 1$$

- variables: $p(a)$; constants: $u_i(a)$
- we could find the social-welfare maximizing CE by adding an objective function

$$\text{maximize:} \quad \sum_{a \in A} p(a) \sum_{i \in N} u_i(a).$$

Why are CE easier to compute than NE?

$$\sum_{a \in A | a_i \in a} p(a) u_i(a) \geq \sum_{a \in A | a'_i \in a} p(a) u_i(a'_i, a_{-i}) \quad \forall i \in N, \forall a_i, a'_i \in A_i$$

$$p(a) \geq 0 \quad \forall a \in A$$

$$\sum_{a \in A} p(a) = 1$$

- intuitively, correlated equilibrium has only a single randomization over outcomes, whereas in NE this is constructed as a product of independent probabilities.
- To change this program so that it finds NE, the first constraint would be

$$\sum_{a \in A} u_i(a) \prod_{j \in N} p_j(a_j) \geq \sum_{a \in A} u_i(a'_i, a_{-i}) \prod_{j \in N \setminus \{i\}} p_j(a_j) \quad \forall i \in N, \forall a'_i \in A_i.$$

- This is a nonlinear constraint!

We start with the feasibility program. Given a support profile $\sigma = (\sigma_1, \sigma_2)$ as input (where each $\sigma_i \subseteq A_i$), feasibility program TGS (for "test given supports") finds a NE p consistent with σ or proves that no such strategy profile exists. In this program, v_i corresponds to the expected utility of player i in an equilibrium, and the subscript $-i$ indicates the player other than i as usual. The complete program follows.

$$\sum_{a_{-i} \in \sigma_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) = v_i \qquad \forall i \in \{1,2\}, a_i \in \sigma_i \tag{4.26}$$

$$\sum_{a_{-i} \in \sigma_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) \leq v_i \qquad \forall i \in \{1,2\}, a_i \notin \sigma_i \tag{4.27}$$

$$p_i(a_i) \geq 0 \qquad \forall i \in \{1,2\}, a_i \in \sigma_i \tag{4.28}$$

$$p_i(a_i) = 0 \qquad \forall i \in \{1,2\}, a_i \notin \sigma_i \tag{4.29}$$

$$\sum_{a_i \in \sigma_i} p_i(a_i) = 1 \qquad \forall i \in \{1,2\} \tag{4.30}$$

Linear Program for Weak Dominance

- Question: Is pure strategy s_i weakly dominated by some mixed strategy with probs $p_1,..,p_j,\ldots$?
- As with strict dominance, need to use objective function to check a strict inequality.
- Maximize $\Sigma_{a_{-i}} [(\Sigma_{a_j} p_j \cdot u_i (a_j,a_{-i})) - u_i(s_i,a_{-i})]$

  subject to $(\Sigma_{a_j} p_j \cdot u_i (a_j,a_{-i})) \geq u_i(s_i,a_{-i})$ for all a_{-i}

  $p_j \geq 0$ for all j

  $\Sigma_j p_j = 1$

Linear Program for 2-player Nash Equilibrium that doesn't work

- Solve the following constraints.

$\Sigma_k\, u_1(a_1^j, a_2^k) \cdot s_2^k + r_1^j = U_1$ for all j indexing an action of player 1

$\Sigma_j\, u_2(a_1^j, a_2^k) \cdot s_1^j + r_2^k = U_2$ for all k indexing an action of player 2

$\Sigma_j\, s_1^j = 1$, $\Sigma_k\, s_2^k = 1$

$s_1^j \geq 0$, $s_2^k \geq 0$, $r_1^j \geq 0$, $r_2^k \geq 0$ for all j,k

- Why doesn't this work?
- Can make the slack variables r and U arbitrarily large.

Linear Complementarity Program

$\Sigma_k u_1(a_1^j, a_2^k) \cdot s_2^k + r_1^j = U_1$ for all j indexing an action of player 1

$\Sigma_j u_2(a_1^j, a_2^k) \cdot s_1^j + r_2^k = U_2$ for all k indexing an action of player 2

$\Sigma_j s_1^j = 1, \Sigma_k s_2^k = 1$

$s_1^j \geq 0, s_2^k \geq 0, r_1^j \geq 0, r_2^k \geq 0$ for all j,k

$r_1^j \cdot s_1^j = 0, r_2^k \cdot s_2^k = 0$ here $\cdot$ is the dot product

- Why does this work?
- If s_1^j is positive (in the support of s_1), then $r_1^j = 0$.
- Think of r_1^j as the incentive to switch from a_1^j to a best reply against s_2^k.
- The bad news: this is no longer a linear program.
  The book discusses the famous Lemke-Howson algorithm for solving the LCP.

Ficticious Play

- A simple intuitive way to learn in a repeated game is also often a good way to compute a Nash equilibrium.
- In **ficticious play**, we start at a random point.
- Each player observes the average frequency with which each action was chosen by their opponents.
- Then the player chooses a best reply to the average frequency.
- If this process converges, it settles into a Nash equilibrium.
- It converges in many situations, e.g. zero-sum games.