1 Game Representation

1.1 Tabular Form and the Problem of Succinctness

In the previous lecture, we generally dealt with games represented in tabular or normal form in which each player’s payoff is listed for all choices of strategies by all players. Table 1 represents the Prisoner’s Dilemma in this form.

<table>
<thead>
<tr>
<th></th>
<th>silent</th>
<th>defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>silent</td>
<td>3,3</td>
<td>0,4</td>
</tr>
<tr>
<td>defect</td>
<td>4,0</td>
<td>1,1</td>
</tr>
</tbody>
</table>

If there are $n$ players, each with $s$ strategies to choose from, then the number of entries in the table is $n \cdot s^n$. With a representation this large, the goal of algorithmic efficiency becomes meaningless, as simply reading the game into memory takes exponential time. It is for this reason that alternative game representations have been developed.

1.2 Graphical Games

A graphical game is given by a graph $G = (V, E)$ in which the vertices correspond to the players and the presence of an edge $e = (p_1, p_2)$ between players 1 and 2 means that the strategies that each player chooses (potentially) affects the other’s payoff. For example, in Figure 1, the payoff of $p_2$ is given by a function $U_2(s_1, s_2, s_3, s_4)$ whose value depends only on the strategies of players 1, 2, 3, and 4. For a graphical game with $n$ players, each of which have $s$ strategies each, where the graph has maximum degree $d$, the representation of a graphical game requires $n \cdot s^{d+1}$ entries. For a sparse graph, this is a far smaller representation than normal form.

Figure 1: Graphical Game

Each player $p_i$ plays strategy $s_i$. 
A decomposable graphical game is a graphical game with the further restriction that the payoff for the player is affected by the strategies that each neighbor plays independently. For example, if the game in Figure 1 was a decomposable graphical game, the payoff of \( p_2 \) is given by a function

\[
U_2(p_1, p_2, p_3, p_4) = U_{2,1}(p_2, p_1) + U_{2,3}(p_2, p_3) + U_{2,4}(p_2, p_4).
\]

We will see below (Sec. 2) that zero-sum decomposable graphical games can be solved easily with linear programming.

1.3 Congestion Games

A congestion game is given by a graph where each of the \( n \) players is assigned 2 vertices and each edge is assigned an \( n \)-tuple defining a delay function for that edge. In the game, each player must choose a set of edges forming a path between his 2 vertices. Each edge of the graph has some delay, which is a function of only the number of players that choose to use that edge. A player’s total delay is the sum of the delays of all edges used on the path. See Figure 2 for an example. For a congestion game formed by a graph with \( m \) edges, representing the game requires roughly \( m \cdot n \) entries.

![Figure 2: Congestion Game](image)

In the congestion game above, suppose Player A uses edge 1 and Player B uses edge 2, and that Players D and E use neither edge 1 nor edge 2. If Player C uses edge 1, the delay on that edge for Players A and C will be 2 whereas the delay for Player B (on edge 2) will be 1. Alternatively, if Player C uses edge 2, the delays on both edge 1 and edge 2 will be 1.

1.4 Symmetric Games

A game in which all players are indistinguishable is a symmetric game. Examples of such games include Prisoner’s Dilemma, Rock-Paper-Scissors, and 2/3 of the Majority. In a symmetric game all \( n \) players have the same strategies and same payoffs, which are a function only of how many players chose each strategy, not which players chose them. Nash proved that every symmetric game has a symmetric Nash equilibrium - one in which every player shares a single strategy (1951).

To represent a symmetric game, we only need to store a single payoff value for each possible set of strategies played. Since only the number of players who choose a particular strategy matters, \( s \cdot \binom{n+s-1}{s-1} \) entries are needed.

1.4.1 Anonymous Games

A generalization of a symmetric game is an anonymous game, one in which the players are distinguishable, but the payoffs (possibly different for each player) are still a function only of how many players choose each strategy rather than which players. In other words, each player sees all the other players as anonymous, or indistinguishable. An anonymous game requires \( n \cdot s \cdot \binom{n+s-1}{s-1} \) entries to represent.
1.5 Extensive Form (Bayesian Games)

The following description of extensive-form games is taken from Wikipedia:

An extensive-form game is a specification of a game in game theory, allowing (as the name suggests) explicit representation of number of important aspects, like the sequencing of players’ possible moves, their choices at every decision point, the (possibly imperfect) information each player has about the other player’s moves when he makes a decision, and his payoffs for all possible game outcomes. Extensive-form games also allow representation of incomplete information in the form of chance events encoded as "moves by nature".

...[A]n n-player extensive-form game thus consists of the following:

- A finite set of n (rational) players
- A rooted tree, called the game tree
- Each terminal (leaf) node of the game tree has an n-tuple of payoffs, meaning there is one payoff for each player at the end of every possible play
- A partition of the non-terminal nodes of the game tree in n+1 subsets, one for each (rational) player, and with a special subset for a fictitious player called Chance (or Nature). Each player’s subset of nodes is referred to as the "nodes of the player". (A game of complete information thus has an empty set of Chance nodes.)
- Each node of the Chance player has a probability distribution over its outgoing edges.
- Each set of nodes of a rational player is further partitioned in information sets, which make certain choices indistinguishable for the player when making a move, in the sense that:
  - there is a one-to-one correspondence between outgoing edges of any two nodes of the same information set—thus the set of all outgoing edges of an information set is partitioned in equivalence classes, each class representing a possible choice for a player’s move at some point—, and
  - every (directed) path in the tree form the root to a terminal node can cross each information set at most once the complete description of the game specified by the above parameters is common knowledge among the players

A play is thus a path through the tree from the root to a terminal node. At any given non-terminal node belonging to Chance, an outgoing branch is chosen according to the probability distribution. At any rational player’s node, the player must choose one of the equivalence classes for the edges, which determines precisely one outgoing edge except (in general) the player doesn’t know which one is being followed.

So far, every game we have seen in the above representations can easily be expanded into tabular form. But how can we reconcile Chance nodes in an extensive form game? In the example in Figure 3, each player is assigned a type which can affect the choices made and payoffs earned. If each player gets a type $t \in T$, then the Chance node has a probability distribution on $T^n$, the set of all type assignments. Now we can generalize the idea of a player’s strategy (ex: do A with some probability $p_a$, do B with some probability $p_b$, etc) to a function $f : T \to strategy$ that assigns a strategy to each type. Given an assignment of types, the game can take a tabular form.

1.6 Others

There are many other types of games and game representations, including but certainly not limited to Scheduling, Facility Location, and Network Design games.
In this game, Players 1 and 2 each pay $1 to play. Then Nature (the dealer) gives Player 1 a card that’s High or Low. Player 1 can either fold (in which case Player 2 gets the pot) or bet an additional $1. If Player 2 bets, then Player 1 can either fold (in which case Player 2 gets the pot) or call by placing another $1 in the pot. If both players bet then Player 1 wins if his card was High loses if it was Low. After Player 1 bets, Player 2 has a single Information Set - the two circled nodes are indistinguishable from his point of view.

This example was taken from http://www.u.arizona.edu/~mwalker/PokerGame.pdf.
2 Decomposable Zero-Sum Games

In the homework for week 1, we saw a simple case of a decomposable graphical game. In that scenario, Player B was playing 2 zero-sum games simultaneously, one against Player A and one against Player C. The catch was that Player B had to a single strategy to apply to both games and his payoff was to be the sum of his payoffs from the two separate games. As an exercise, we showed that this could be solved by a linear program. It turns out that this approach can be used more generally.

Theorem 1. In any zero-sum decomposable graphical game, the Nash Equilibrium can be found in polynomial time.

Proof: First, we must prove that a Nash Equilibrium exists at all. To do this, we simply use the sledge hammer that is Nash’s Theorem - namely that every game has a Nash Equilibrium. Now our task is to find one.

Define the following variables:
\[ x_{u,j} = \Pr[\text{player } u \text{ chooses action } j] \]
\[ U_{u,v}(i,j) = \text{the payoff for player } u \text{ if } u \text{ plays } i \text{ and } v \text{ plays } j. \]
\[ L_{u,i}(x) = \text{the expected payoff for player } u \text{ if he chooses action } i \text{ given that the other players fix their strategies } x_{v,j}. \]

Notice that \( L_{u,i} \) is a linear function because the game is decomposable. In particular, \( L_{u,i} = \sum_{v,j} U_{u,v}(i,j) \cdot x_{v,j}. \)

Consider the following linear program:

\[
\begin{align*}
\text{Minimize } & \sum_u w_u \\
\text{subject to: } & w_u \geq L_{u,i} \forall u, i \\
& \sum x_{u,i} = 1 \forall u \\
& x_{u,i} \geq 0 \forall u, i
\end{align*}
\]

The following claim completes the proof:

Claim 1. The minimum is achieved at 0 which is a Nash Equilibrium.

Proof: Define:
\[ \bar{w}_u = \sum x_{u,i} L_{u,i} \text{ - This is the average gain of player } u. \]

Obviously, \( \sum_u \bar{w}_u = 0 \) because this is a zero-sum game.

From the linear program we have \( w_u \geq \bar{w}_u. \) So to minimize \( w_u, \) we can set \( w_u = \bar{w}_u \forall u, \) yielding \( \sum_u w_u = 0, \) which is minimal. Thus, for every strategy \( i, \) \( L_{u,i} \leq w_u \) is no better than the average \( \bar{w}_u, \) proving that it is indeed a Nash Equilibrium.

□ □

3 An Ancient Algorithm: Fictitious Play

In addition to the linear programming method we learned about previously, there are several natural algorithms for solving zero sum games. Of these, we first discuss “fictitious play,” a strategy in which we imagine the two players repeatedly playing with a naive strategy.

The first round of fictitious play is played randomly. For each subsequent round, both players look at the history of plays by the other, and assume that these historical plays represent the strategy that player will use: a strategy that was used in p% of the past rounds is assumed to be played with probability p%. Each player then plays the best response to that assumed history-based strategy. Over time, these historical strategies will converge to an equilibrium strategy.

As an example, consider a zero-sum game with row player payoff matrix:

\[
R = \begin{pmatrix}
2 & 1 & 0 \\
2 & 0 & 3 \\
-1 & 3 & -3
\end{pmatrix}
\]
(and column player payoff matrix $C = -R$)

The fictitious play algorithm may then run as follows:

<table>
<thead>
<tr>
<th>t</th>
<th>row plays</th>
<th>col plays</th>
<th>$u^t$ (ave gain vector by row)</th>
<th>$v^t$ (ave loss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>-0.3, -3</td>
<td>2, 1, 0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>-0.3, -3</td>
<td>$\frac{2}{3}, \frac{3}{2}$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>$\frac{1}{2}, -0.2, -1$</td>
<td>$\frac{2}{3}, \frac{1}{2}$</td>
</tr>
<tr>
<td>etc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that this process bounds the equilibrium value $v$ as: $\max u^t \geq v \geq \min v^t$.

**Theorem 2.** *(Robinson, 1950)* Fictitious play converges to the Nash equilibrium.

Robinson’s proof of convergence indicates a rate of $t \sim \left(\frac{c}{\epsilon}\right)^{m+n}$ (in the worst case, not probabilistic). Karlin conjectured (1965) a faster convergence rate $t \sim \left(\frac{c}{\epsilon}\right)^2$ suffices.

### 4 Another Natural Algorithm: Experts

Another natural algorithm is based on a concept called “boosting,” or alternatively “no regret learning,” “hedging,” “experts,” or “multiplicative updates” (MU). We will first introduce this concept, and then show how it can be applied to zero sum games.

#### 4.1 Introduction to Experts Algorithms

In these algorithms, $n$ experts suggest strategies over time, and the player can decide which mix of expert strategies to use based on the historical losses observed from each expert’s strategies. Specifically, at each time step $t$, the player produces an $n$-vector of weights $w^t$ indicating how much to follow each expert’s strategy, and then nature produces an $n$-vector of losses $l^t$ caused by following each strategy. The player’s loss is then computed as $w^t \cdot l^t$. Cumulative loss over time is $L^T = \sum_{t=1}^{T} ||w^t||_1 \cdot l^t$.

For example, we may imagine the following sequence of strategies, weights, and resulting losses from 3 experts:

<table>
<thead>
<tr>
<th>t= 1</th>
<th>t= 2</th>
<th>t= 3</th>
<th>t= 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight:</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>loss:</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The goal is to choose weights $w$ to do “well.” Doing well could mean getting the smallest cumulative loss $L^T$, or predicting the best row (best expert).

Given this setup, several algorithms for choosing the weight vectors are possible:

**Follow the leader**: For each round $t$ choose the expert $i$ who has the lowest loss on average so far, and set $w_i = 1$, $w_j \neq i = 0$. It is easy to construct a worst case scenario in which this algorithm incurs total loss which is $n$ times worse than following the true best expert.

**Multiplicative updates (MU)**: Punish $i$th expert by punishment function: $x \pi(l^t_i) = x(1 + \epsilon)^{l^t_i}$, with $\pi(l^t_i)$ bounded by $\alpha^x \leq x(x) \leq 1 - (1 - \alpha)x$.

**Theorem 3.** With multiplicative updates, cumulative loss $L^T \leq \min_i L_i^T \ln \frac{1}{1-\alpha} + \ln(n) \frac{1-\alpha}{1-\alpha}$

**Proof:**

$$\sum_{i=1}^{n} w_i^{t+1} = \sum_{i=1}^{n} w_i^t \cdot \pi(l^t_i)$$

1-6
\[ \leq \sum_i w_i^t (1 - (1 - \alpha)l_i^t) \]
\[ = (\sum_i w_i^t) (\sum_i p_i^t (1 - (1 - \alpha)l_i^t)) \]
\[ \ln(\sum_i w_i^{t+1}) \leq \ln(\sum_i w_i^t) + \ln(1 - (1 - \alpha)l_i^t) \]
\[ \ln(\sum_i w_i^{t+1}) \leq \ln(\sum_i w_i^t) - (1 - \alpha)l_i^t \]
\[ \ln(\sum_i w_i^{t+1}) \leq -(1 - \alpha)L^T \]
\[ L^T \leq -\ln(\sum_i w_i^{t+1}) \leq -\frac{\ln w_i^T}{1 - \alpha} \]
\[ w_i^{t+1} \geq w_i^t \alpha^t + \ldots + \sum_i \alpha^t \geq \frac{1}{n} \alpha L_i^T \]
\[ L^T \leq \ln(n) \frac{\ln(\alpha)}{1 - \alpha} + \ln(\alpha) \frac{\ln(\alpha)}{1 - \alpha} L_i^T \]

Further, note that setting \( \alpha = 1 - \sqrt{\frac{\ln n}{T}} \) implies \( L^T \leq \min_i L_i^T + 2\sqrt{T\ln n} \)

\[ \frac{1}{T} L^T \leq \min_i \frac{1}{T} L_i^T + 2\sqrt{\frac{\ln n}{T}} \]

### 4.2 Application to Zero Sum Games

Given a zero sum game \((A, -A)\), we can let both players play the game repeatedly, using multiplicative updates to update their strategies, \(x^t\) and \(y^t\).

Then we can analyze the convergence of this method as follows:

\[ L^T = \sum_{i=1}^T (x^t A y^t) \geq (e_i A (\sum y^t) - O(\sqrt{T}) \forall i(*) \]
\[ \sum x^t A y^t \leq \sum x^t A e_j + O(\sqrt{T}) \forall j \]
\[ \Rightarrow \sum x^t A y^t \leq \sum x^t A \frac{\sum y^t}{T} + O(\sqrt{T})(**) \]
\[ (*) \Rightarrow \sum x^t A (\frac{\sum y^t}{T}) \geq e_i A \sum y^t - O(\sqrt{\frac{1}{T}}) \]
\[ \Rightarrow (\frac{x^t A}{T}) \sum \frac{y^t}{T} \geq e_i A \sum \frac{y^t}{T} - O(\sqrt{\frac{1}{T}}) \]

Therefore, \( \frac{x^t}{T} \) approximates the Nash equilibrium for the row player.

And similarly the algorithm also converges for the column player.

Note that \( t \sim (\frac{2}{\alpha})^2 \), so this converges at the rate Karlin conjectured for fictitious play.