Adversarial Search and Game Theory

CS 510 Lecture 4
October 22, 2018
Reminders

• Proposals due today
• Midterm next week (no in-class lecture)
• past midterms online
• Midterm online BBLearn
  • Covers readings (book/papers) and lectures
  • Available Sat noon - Tues night 11:59 pm, ~2 hours
Overview

- Game theory (Simultaneous games)
- Games (Stackelberg)
What is game theory?

- Game theory is a formal way to analyze strategic interactions among a group of rational players (or agents) who behave strategically

- Game theory has applications
  - Economics
  - Politics
  - Computer Science
What is game theory?

Games are a form of *multi-agent environment*

- Key question: How do actions of other agents affect me?
- Multi-agent environments can be cooperative or competitive
- *Games* are generally (but not always) applied in competitive/adversarial environments
- Each agent is completely self-interested
Relation of Games to Search

• Search – no adversary
  • Solution is (heuristic) method for finding goal
  • Evaluation function: estimate of cost from start to goal through node
  • Examples: path planning, scheduling activities

• Games – adversary
  • Solution is strategy (strategy specifies move for every possible opponent reply).
  • Evaluation function: evaluate “goodness” of game position
  • Examples: chess, checkers, Othello, backgammon
Types of Games

- **Perfect Information**
  - Deterministic: chess, checkers, go, othello
  - Chance: backgammon, monopoly

- **Imperfect Information**
  - Deterministic: 
  - Chance: bridge, poker, scrabble, nuclear war
Why Study Game Theory?

• Helps us in two ways:

• Agent Design
  • Help design agents that reason strategically and perform optimally

• Mechanism (Game) Design
  • Design Multiagent Systems that maximize collective (global) goals
    • Internet routing, robot teams, traffic congestion
Assumptions

• Features of a game:
  • There are at least two rational players
  • Each player has more than one choice
  • The outcome depends on the strategies chosen by all players; there is strategic interaction

• Example: Six people go to a restaurant.
  • Each person pays his/her own meal – a single-agent decision problem
  • Before the meal, every person agrees to split the bill evenly among them – a game
Assumptions (cont)

• Simultaneous-move
  • Each player chooses his/her strategy without knowledge of others’ choices.
    No cooperation
  • Each player receives his/her payoff at end of game

• Complete information
  • Each player’s strategies and payoff function are common knowledge among all the players.

• Assumptions on the players
  • Rationality
Formal Definition of a Game

- Players $P: \{P1, P2, \ldots, Pn\}$
- Actions $S: \{S1, S2, \ldots, Sn\}$
- Payoff Matrix $M$:
  - Each player chooses an action $s1 \in S1, s2 \in S2, \ldots, sn \in Sn$
  - $M(s1, s2, \ldots, sn) \rightarrow \{u1, u2, \ldots, un\}$ where $ui$ is payoff for Player $Pi$
Game Representations

**Extensive form**
- Player 1: Up, Down
- Player 2: Left, Right

**Matrix form**
- Player 1’s strategy:
  - Up: 1,2
  - Down: 5,6,7,8
- Player 2’s strategy:
  - Left: 3,4
  - Right:
    - Up: 1,2
    - Down: 5,6,7,8

Potential combinatorial explosion
Example: Remote Control Wars

- Players: Chris and Pat
- Actions: Watch soccer game or watch soap opera
  - Chris prefers soap opera
  - Pat prefers soccer
  - Both want to hang out together
- Complete information: both know the matrix

<table>
<thead>
<tr>
<th></th>
<th>Soap</th>
<th>Soccer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris</td>
<td>2,1</td>
<td>0,0</td>
</tr>
<tr>
<td>Pat</td>
<td>0,0</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td>Pat</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Soap</td>
<td>2, 1</td>
<td></td>
</tr>
<tr>
<td>Soccer</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Pat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soap</td>
<td>0, 0</td>
</tr>
<tr>
<td>Soccer</td>
<td>1, 2</td>
</tr>
</tbody>
</table>
Example: Rock, Paper, Scissors

- Two players, each simultaneously chooses Rock, Paper or Scissors.
- When $\Sigma u_i = 0$, we call this a zero-sum game. Otherwise, general-sum.
Example: Rock, Paper, Scissors

- Two players, each simultaneously chooses Rock, Paper or Scissors.
- When $\sum u_i = 0$, we call this a zero-sum game. Otherwise, general-sum.
Definition: Strategy

• An action selection strategy for a given game specifies (probabilistically) the action player should take.

• Let $\pi$ denote the strategy for player $i$
  
  • $\pi_i(s)$ denotes probability with which player $i$ should choose action $s$
  
  • If exists $s$ such that $\pi_i(s) = 1$, $\pi_i$ called a pure strategy
  
  • Else, $\pi_i$ called a mixed strategy

• Example:
  
  • Pure strategy $\pi_i$: $\pi_i(\text{rock}) = 1$, $\pi_i(\text{scissors}) = 0$, $\pi_i(\text{paper}) = 0$
  
  • Mixed strategy $\pi_i$: $\pi_i(\text{rock}) = 0.3$, $\pi_i(\text{scissors}) = 0.3$, $\pi_i(\text{paper}) = 0.4$
Definition: Strategy Profile

- Strategy profile \( \prod \): collection of strategies \( \pi_i \) for each player \( i \)
- Example:
  
  Strategy Profile \( \prod \): \( \langle \pi_i, \pi_j \rangle \)
  
  - \( \pi_i(\text{rock}) = 0.5, \pi_i(\text{scissors}) = 0.5, \pi_i(\text{paper}) = 0.0 \)
  
  - \( \pi_j(\text{rock}) = 0.2, \pi_j(\text{scissors}) = 0.6, \pi_j(\text{paper}) = 0.2 \)
Definition: Expected Value

- The expected value (reward) of a game for player i is given by:
  \[ \sum_{s_i \in S_i} \sum_{s_j \in S_j} \text{Prob}(s_i,s_j) \times u_i(s_i,s_j) \]

- Given strategy profile \( \langle \pi_1, \pi_2 \rangle \), what is the expected value for player 1?
Definition: Expected Value

- The expected value (reward) of a game for player $i$ is given by:
  \[ \sum_{(i,j)} \text{Prob}(s_i,s_j) \times u_i(s_i,s_j) \]

- Given strategy profile $< \pi_1, \pi_2 >$, what is the expected value for player 1?

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Rock</th>
<th>Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rock</td>
<td>0,0</td>
<td>-1,1</td>
<td>1,-1</td>
</tr>
<tr>
<td>Paper</td>
<td>1,-1</td>
<td>0,0</td>
<td>-1,1</td>
</tr>
<tr>
<td>Scissors</td>
<td>-1,1</td>
<td>1,-1</td>
<td>0,0</td>
</tr>
</tbody>
</table>

$\pi_1(\text{rock}) = 1/3, \pi_1(\text{scissors}) = 1/3, \pi_1(\text{paper}) = 1/3$

$\pi_2(\text{rock}) = 1/3, \pi_2(\text{scissors}) = 1/3, \pi_2(\text{paper}) = 1/3$
Definition: Best Response

Strategy

- $\pi_i$ is Best Response for agent $i$ if, given strategies for other agents, $\pi_i$ maximizes expected value for agent $i$.

- What is best response for agent $i$ when agent $j$ plays the following strategy?
  - $\pi_j(b_0) = 0.2$, $\pi_j(b_1) = 0.8$

<table>
<thead>
<tr>
<th>Player i</th>
<th>a0</th>
<th>a1</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0</td>
<td>10,10</td>
<td>0,0</td>
</tr>
<tr>
<td>b1</td>
<td>0,0</td>
<td>12,12</td>
</tr>
</tbody>
</table>
Dominated Strategies

- Strategy $\pi_i$ is strictly dominated by $\pi_i'$ if
  - $u_i(\pi_i, \pi_j) < u_i(\pi_i', \pi_j)$ for all $\pi_j$
Two suspects held in separate cells are charged with a major crime. However, there is not enough evidence.

Both suspects are told the following policy:

- If neither confesses then both will be convicted of a minor offense and sentenced to one month in jail.

- If both confess then both will be sentenced to jail for 3 months.

- If one confesses but the other does not, then the confessor will be released but the other will be sentenced to jail for 5 months.

The dominant strategy is clearly not the "best"!
Dominant strategy equilibrium

- Does not always exist but if it does, irrational to not play it
- Inferior strategies are called dominated
- Dominant strategy equilibrium is a strategy profile where each agent has picked its dominant strategy
  - Requires no counterspeculation
  - But doesn’t always exist, so
    - Nash Equilibrium (The “Beautiful Mind” Guy)
Nash Equilibrium

<table>
<thead>
<tr>
<th></th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L</td>
</tr>
<tr>
<td>Player 1</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>0 , 4</td>
</tr>
<tr>
<td>M</td>
<td>4 , 0</td>
</tr>
<tr>
<td>B</td>
<td>3 , 5</td>
</tr>
</tbody>
</table>

Realize: The strategy profile (B, R) has the following property:

- Player 1 CANNOT do better by choosing a strategy different from B, given that player 2 chooses R.
- Player 2 CANNOT do better by choosing a strategy different from R, given that player 1 chooses B.
Nash equilibrium

• A strategy profile is a Nash equilibrium if no player has incentive to deviate from his strategy given that others do not deviate.

• Or equivalently,

• A set of strategies, one for each player, such that each player’s strategy is best for her, given that all other players are playing their equilibrium strategies

• Note: Dominant strategy equilibria are Nash equilibria but not vice versa
Alternating move games

- Chess (deep blue) - 1997 not quite 1957
  - $b = 35, d=100$
- Checkers (Chinook) - solved
- Backgammon, Othello, Go
- Poker? add uncertainty
Game Trees

- Games as search
- Initial State
- Successor function
  - (move, state) pairs
- Terminal test
- Utility Function
Perfect play for deterministic games

- Assumption: My opponent will make the best possible move
- Solution: Minimax “minimize the maximum possible loss”
- Thm: For every two-person, zero-sum game with finite strategies, there exists a value $V$ and a mixed strategy for each player, such that (a) Given player 2's strategy, the best payoff possible for player 1 is $V$, and (b) Given player 1's strategy, the best payoff possible for player 2 is $-V$.
- Same as mixed-strategy Nash equilibrium for zero-sum games
Minimax value for a node

- Minimax value: Utility (for MAX) of reaching given state

- Minimax-value(n) =
  - Utility(n), if n is a terminal node
  - max over all successors(n), if n is a max node
  - min over all successors(n), if n is a min node
**Minimax Algorithm**

```plaintext
function Minimax-Decision(state) returns an action
    v ← Max-Value(state)
    return the action in Successors(state) with value v

function Max-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← −∞
    for a, s in Successors(state) do
        v ← Max(v, Min-Value(s))
    return v

function Min-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← ∞
    for a, s in Successors(state) do
        v ← Min(v, Max-Value(s))
    return v
```
Class exercise:
Fill in values
Properties of minimax

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes (against an optimal opponent)
- **Time complexity?** $O(b^m)$
- **Space complexity?** $O(bm)$ (depth-first exploration)

- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  $\rightarrow$ exact solution completely infeasible
Alpha-beta pruning

• Same result as minimax but more efficient
• Insight: Do not need to look at all nodes to find minimax value at the root of a game tree
• $\alpha$ - minimum score of maximizing player (-inf)
• $\beta$ - maximum score of minimizing player (inf)
• if $\beta < \alpha$ no need to explore further
Alpha beta example

- When we reach the 5 we know root $R \geq 5$
  - $\alpha = 5$
- $N$ is a min, so $N \leq 4$
  - $\beta = 4$
- But $4 < 5$, so no need to continue looking here (R never chooses N)
α-β pruning example
α-β pruning example
$\alpha$-$\beta$ pruning example
α-β pruning example
$\alpha$-$\beta$ pruning example
Alpha-Beta Pruning

- Algorithm:
  - Explore game tree in Depth First manner
  - Record and update alpha, beta values
  - Discontinue search when alpha > beta (for max nodes) or beta < alpha (for min nodes)
Class exercise:
Redo with alpha-beta
Monte Carlo Tree Search

• Heuristic for end-state for a node
• Monte Carlo Rollouts - simulations with random play from a node to the end
• Use back propagation to estimate the value of intermediate nodes based on the sims
Steps of MCTS

Run continuously in the allotted time

Selection → Expansion → Simulation → Back-propagation
Readings for 11/5

- Rodney A. Brooks. *Intelligence without representation*

- Tambe. *Beliefs, Desires, Intentions (BDI)*, Chapter 2 of CS 499 course reader