Robust Algorithms For Game Play Against Unknown Opponents

Nathan Sturtevant
University of Alberta
May 11, 2006
Introduction

• A lot of work has gone into two-player zero-sum games

• What happens in non-zero sum games and multi-player games?
  • Actual games
  • Robotic teams
  • Perfect-information extensive-form
Multi-Player Games

- Max\(^n\) algorithm
  - Luckhardt and Irani, 1986
- \(n\)-tuple of scores/utilities
- One value for each player, eg (3, 5, 7)
Max$^N$ Decision Rule

(3, 5, 2)
Max\(^n\) Computation

- Max\(^n\) computes an equilibrium strategy
- If all players were given the strategy, nobody would have incentive to change

Assumes:
- All utilities known exactly
- Tree analyzed completely
- Players choose common strategy
- Strategies cannot be changed
Sample Domain: Spades

- Spades
  - Trick-based card game
  - Use 3-player variation
  - Many similar card games
- Tricks \(\rightarrow\) Hands \(\rightarrow\) Game
Spades Rules - 1 Hand

• Cards dealt to players
• Players bid how many tricks they will take
• After playing the hand:
  • -10xbid if bid is missed (eg bid 5 take 4)
  • 10xbid if bid is made (eg bid 5 take 5 or 6)
  • -100 for taking 10 overtricks
Spades Strategies

• Players may play with different strategies:
  • Minimize overtricks (mOT)
  • Maximize tricks (MT)
• Players must model opponents’ strategies
Experimental Setup

- 100 games, played to 300 points
- 7 cards per player
- Perfect information
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>Score</th>
<th>%Win</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>mOT</td>
<td>MT</td>
<td>178.2</td>
<td>44.0</td>
<td>207.3</td>
</tr>
<tr>
<td>mOT</td>
<td>MT</td>
<td>198.2</td>
<td>53.5</td>
<td>191.4</td>
</tr>
<tr>
<td>mOT</td>
<td>MT</td>
<td>235.4</td>
<td>59.0</td>
<td>199.2</td>
</tr>
<tr>
<td>mOT</td>
<td>MT</td>
<td>248.6</td>
<td>74.7</td>
<td>163.8</td>
</tr>
</tbody>
</table>
Results - Discussion

- We must use *some* opponent model
- Don’t know opponents utilities
- Even in perfect-information games
- Payoffs ≠ utilities
- Model has large effect on quality of play
Spades Example

(30, 10, 10) (-30, 10, 11) (30, 10, 10) (30, 10, 10)
Max^n Deficiencies

- Max^n only calculates one of many equilibria
- Keeps no information about alternates
- Some alternates may be less risky in the face of uncertain opponents
**Soft-Max**$^N$

- Back up sets of $\text{max}^n$ values
- Each time there is a tie, return both values
- Calculates a superset of all equilibria
Spades Example

1

{(30, 10, 10)}

2

{(30, 10, 10), (-30, 10, 11)}

3

(30, 10, 10) (-30, 10, 11)

3

(30, 10, 10) (30, 10, 10)
**Soft-Max**$^n$ - Dominance

- Dominance relationship to compare $\text{max}^n$ sets with respect to a given player
  - $\{(10, 2, 7), (8, 7, 4)\}$ vs:
    - $\{(5, 10, 4)\}$ – strictly dominates
    - $\{(8, 4, 7)\}$ – weakly dominates
    - $\{(9, 1, 9)\}$ – no domination
  - Union all sets that are not dominated
Soft-Max\textsuperscript{N} - Outcomes

- How large can soft-max\textsuperscript{n} sets grow?
- In trick-based card games
  - \(n\) players, \(c\) cards
  - \(O(c^{n-1})\) possible game outcomes
- In other domains we can reduce number of outcomes
Opponent Modeling

- Represent opponent models as a graph
- Nodes are outcomes in the game
- Directed edges represent preferences
- Partial order over game outcomes
Opponent Models

Possible Outcomes

1: (0, 0, 2)
2: (0, 1, 1)
3: (0, 2, 0)
4: (1, 0, 1)
5: (1, 1, 0)
6: (2, 0, 0)

maximize tricks

minimize overtricks
Opponent Modeling

• We do not want to assume too much about our opponents
• Eliminating all ties would remove all ambiguities from $\text{max}^n$ analysis
• Analysis will be incorrect unless we have a perfect opponent model
• More or less accurate model?
Opponent Models

- Combine opponent models to form more generic opponent models
- Intersection of edges over each opponent model
- Builds a generic opponent model
Opponent Models

Possible Outcomes

1: (0, 0, 2)
2: (0, 1, 1)
3: (0, 2, 0)
4: (1, 0, 1)
5: (1, 1, 0)
6: (2, 0, 0)

maximize tricks

minimize overtricks
Generic Opponent Model

- bid made
- bid missed

 generic model

4 → 5 → 6 → 1 → 2 → 3
Soft-Max\textsuperscript{n} Performance

- Run same experiments as before
  - Use soft-max\textsuperscript{n} with generic opponent models
## Experimental Results

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Score</th>
<th>%Win</th>
<th>%Gain</th>
<th>%Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>mOT</td>
<td>MT</td>
<td>241.7</td>
<td>68.6</td>
<td>15.0</td>
<td>6.8</td>
</tr>
<tr>
<td>mOT</td>
<td>MT</td>
<td>218.2</td>
<td>53.5</td>
<td>9.5</td>
<td>5.5</td>
</tr>
<tr>
<td>mOT</td>
<td>mOT</td>
<td>242.2</td>
<td>54.8</td>
<td>4.8</td>
<td>8.0</td>
</tr>
<tr>
<td>mOT</td>
<td>mOT</td>
<td>230.6</td>
<td>46.0</td>
<td>8.8</td>
<td>4.0</td>
</tr>
</tbody>
</table>
Learning in Soft-Max\textsuperscript{N}

- We observe players’ actions during the game
- Sometimes we can distinguish between models based on their moves
- Similar to version space learning
- Used 3 player models and did inference
  - In 900 hands, 423 (correct) inferences
  - Identify player type in 1/6 hands
**Soft-Max^N Summary**

- It is better to under-assume than over-assume about our opponents
- Need a bigger picture of what is happening in the game
- Can observe players to learn their models
- Only use a partial ordering of outcomes
- No utilities actually used
Thanks

• Joint work with Michael Bowling

• See also:
  • ProbMax^n : Opponent Modeling in N-Player Games, Nathan Sturtevant, Michael Bowling, and Martin Zinkevich, AAAI-06.