Learning to Play Hearts

Nathan Sturtevant
University of Alberta - AI Seminar
February 3, 2006
Hearts

- Trick-based card game
Sample Trick
Sample Trick
Hearts

- Trick-based card game
- Want to *minimize* your points
  - One point for every heart (♥)
  - 13 points for Q♠
  - If one player takes all 26 points (shoots the moon) others get 26 each
Challenge

- Learn to play the game of **Hearts** well:
  - Multi-Player Game
  - Imperfect Information
  - Learning
Multi-Player Games

• A lot of work in two-player games:
  • Checkers, chess, backgammon, scrabble, othello, go…

• Much less in multi-player games
Multi-Player Games

• Differences:
  • $\text{Max}^n$ algorithm; generalization of minimax
  • Less pruning possible
  • Weaker theoretical properties
Hearts Example

A♠ 3♣

K♠ Q♣ K♠ Q♣

3 3 3 3

(0, 0, 13) (13, 0, 0) (0, 0, 13) (0, 0, 13)
Imperfect Information

- In practice we can’t see opponents cards
- Monte-Carlo Sampling
  - Generate perfect-information sample hands for opponents
  - Analyze samples
  - Combine results
Previous Work

- Hearts program based on previous ideas
  - Hand-tuned evaluation function
  - Non-linear evaluation function
  - Somewhat slow
- Plays as well (better than) best computers?
## Average Scores

<table>
<thead>
<tr>
<th></th>
<th>Per Game</th>
<th>Per Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expert Program</strong></td>
<td>56.1</td>
<td>5.16</td>
</tr>
<tr>
<td><strong>Opponent Avg.</strong></td>
<td>76.3</td>
<td>6.97</td>
</tr>
</tbody>
</table>

Played 90 games, each to 100 points.
Learning in Hearts

• University of Mass. Course Project
  (Perkins, 1998)

• Operational Advice
  (Fürnkranz, et. al., 2000)

• State sampling with imperfect-information
  (Fujita and Ishii, 2005)
Our Approach

• Use search-based approach to train
  • Similar to what was used in Backgammon
  • TD-Gammon plays at the level of the best humans
Backgammon

- Why did learning in backgammon work well?
  - Already developed good neural networks
  - Stochastic element helps exploration
  - Good features
Hearts

• Promising domain for learning:
  • Game fixed length (13 moves)
  • Cards dealt randomly
  • Occasionally get good cards
Hearts Difficulty

- Cards have relative value
  - 5♣ is good when 2-4♣ already played
  - 5♣ is bad when 6-A♣ already played
Approach

- Define features for game
- Use perceptron (regression) to predict game score given features
- Use $\max^n$ to play given predicted score
- Use $\text{TD}(\lambda)$ to train
Features

- What features to use for each player?
  - 52 cards they could have in their hand
  - 52 cards they could have taken
  - 104 features per player
  - 416 total features
Valuable Feature

- Interesting feature: P1 has the lowest ♥
- [P1 has 2♥] or
- [P1 has 3♥] and
  [[P1 has taken 2♥] or [P2 has taken 2♥]]
  [[P3 has taken 2♥] or [P4 has taken 2♥]]
- ...
Features

• We defined basic ‘atomic’ features
• Only evaluate features on trick boundaries
• Sample Features
  • Which suits do we hold low/high cards
  • Which suits are we ‘short’
  • Which suits does the ‘leader’ have
Features

• These features still inadequate
  • Combinations of features more interesting than ‘atomic’ features
  • Combine features using AND operator
Learning Part I

• Learn to avoid the Q♠
• 60 ‘atomic features’
• \( \lambda = 0.75 \)
• \( \alpha = 1/[13 \times \# \text{ active features}] \)
• Predict expected points in game
• Train against previous program
• Randomly switch position each game
Analysis

- What is the network learning
  - Easily understand by examining weights assigned to feature sets
# Features - Avoid Q♠

<table>
<thead>
<tr>
<th>Rank</th>
<th>Weight</th>
<th>We have</th>
<th>We have</th>
<th>We have</th>
<th>Opponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.103</td>
<td>1 low ♠</td>
<td></td>
<td>Lead</td>
<td>Q♠ no ♠</td>
</tr>
<tr>
<td>2</td>
<td>-0.097</td>
<td>1 low ♠</td>
<td>No ♥</td>
<td>Lead</td>
<td>Q♠ no ♠</td>
</tr>
<tr>
<td>3</td>
<td>-0.096</td>
<td>2 low ♠</td>
<td>K♦</td>
<td></td>
<td>Q♠ two ♠</td>
</tr>
<tr>
<td>4</td>
<td>-0.093</td>
<td>1 low ♠</td>
<td>No ♣</td>
<td>Lead</td>
<td>Q♣ no ♠</td>
</tr>
<tr>
<td>5</td>
<td>-0.090</td>
<td>1 low ♠</td>
<td>No ♦</td>
<td>Lead</td>
<td>Q♠ no ♠</td>
</tr>
<tr>
<td>148</td>
<td>-0.040</td>
<td>1 low ♠</td>
<td>Q♠</td>
<td></td>
<td>Lead no ♠</td>
</tr>
</tbody>
</table>
# Features - Take Q♠

<table>
<thead>
<tr>
<th>Rank</th>
<th>Weight</th>
<th>We Have</th>
<th>We have</th>
<th>We have</th>
<th>We have</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.125</td>
<td>Q♠</td>
<td>1 low ♠</td>
<td></td>
<td>Lead</td>
</tr>
<tr>
<td>2</td>
<td>0.123</td>
<td>Q♠</td>
<td>1 low ♠</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.117</td>
<td>Q♠</td>
<td>No ♣</td>
<td>No ♥</td>
<td>Lead</td>
</tr>
<tr>
<td>4</td>
<td>0.116</td>
<td>A/K/Q♠</td>
<td></td>
<td></td>
<td>Lead</td>
</tr>
<tr>
<td>5</td>
<td>0.112</td>
<td>Q♠</td>
<td>No ♣</td>
<td>No ♥</td>
<td>No ♠</td>
</tr>
</tbody>
</table>
Learning Part II

• Learn to avoid taking ♥
• Removed 14 Q♣-specific features
• 42 new point (♥) related features (0-13)
• Same learning parameters
Hearts Features

- Break Even
- 1x Features
- 2x Features
- 3x Features

Games Played

Average Score

200k
Learning Part III

• Learn to play the full game of Hearts
  • No ‘shooting the moon’
  • Take best 10,000 features from the Q♠
  • Take best 1,000 features from ♥ points
• Train against expert and by self-play
Steady-State Evaluation

- Test the learned networks
- 4 players, 2 player types
- $2^4$ ways of assigning player types
- Repeat each hand $2^4$ - 2 times
- Play 100 hands
## Arrangement

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Player 2</th>
<th>Player 3</th>
<th>Player 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>Trained</td>
<td>Trained</td>
<td>Trained</td>
</tr>
<tr>
<td>Trained</td>
<td>Expert</td>
<td>Trained</td>
<td>Trained</td>
</tr>
<tr>
<td>Expert</td>
<td>Expert</td>
<td>Trained</td>
<td>Trained</td>
</tr>
<tr>
<td>Trained</td>
<td>Trained</td>
<td>Expert</td>
<td>Trained</td>
</tr>
<tr>
<td>Expert</td>
<td>Trained</td>
<td>Expert</td>
<td>Trained</td>
</tr>
<tr>
<td>Trained</td>
<td>Expert</td>
<td>Expert</td>
<td>Trained</td>
</tr>
<tr>
<td>Expert</td>
<td>Expert</td>
<td>Expert</td>
<td>Trained</td>
</tr>
</tbody>
</table>
Games Against Expert

Break-Even  Expert Trained  Self Trained

Games Played

Average Score

10k

Learning In Hearts

Nathan Sturtevant
Adam White
Games Against Expert

Average Score vs. Games Played for Break Even, Expert Trained, and Self Trained.
Summary

- Learned to beat ‘expert’ by a large margin
- Program plays well, but lacks deep analysis of game
Ongoing Work

• Learn with ‘shooting the moon’ turned on
  • Duplicate all features three times
    • Points are split
  • We have all the points
  • Someone else has all the points
• Compare steady-state play
### Steady-State Results

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>Score (1)</th>
<th>Score (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Trained</td>
<td>90k games</td>
<td>Expert</td>
<td>6.27</td>
<td>7.10</td>
</tr>
<tr>
<td>Expert-Trained</td>
<td>220k games</td>
<td>Expert</td>
<td>6.17</td>
<td>7.35</td>
</tr>
<tr>
<td>Self-Trained</td>
<td>90k games</td>
<td>Expert-Trained</td>
<td>6.35</td>
<td>7.03</td>
</tr>
<tr>
<td>Expert-Trained</td>
<td>220k games</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Work

- Optimize code
- Different algorithm than $\text{max}^n$
- Better ways of combining features
- Play against humans
- Passing cards
- Imperfect information
Thank You

• Joint work with Adam White

• Thanks to Rich Sutton and Mark Ring