# Constructing Features to Learn to Play Hearts

Nathan Sturtevant and Adam White
University of Alberta

Computers and Games
May 31, 2006

# Challenge

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

# 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

# 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:
  - $Max^n$ algorithm; generalization of minimax
  - Less efficient search/pruning
  - Weaker theoretical properties

# Imperfect Information

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

# Learning

- Learning algorithms not yet "plug and play"
- Significant tuning often needed to learn

# Previous Work

- Search-based Hearts program
  - Hand-tuned evaluation function
  - Monte-Carlo search
- Plays as well (better than) best computers?


3D Hearts Deluxe 7.5 0506
Your lead...

# Average Scores

| | Per Game | Per Hand |
|---|---|---|
| Expert Program | 56.1 | 5.16 |
| Opponent Avg. | 76.3 | 6.97 |

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)

# General Approach

- Define perfect information features
  - Linearly weighted
- Monte-Carlo sampling
  - $Max^n$ search in perfect-information game
- Use TD(λ) with linear regression to train

# 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

# 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♥]]
  - …

# Feature Abstraction

- We defined basic 'atomic' features
- Sample Features
  - Which suits do we hold low/high cards
  - Which suits are we 'short'
  - Which suits does the 'leader' have

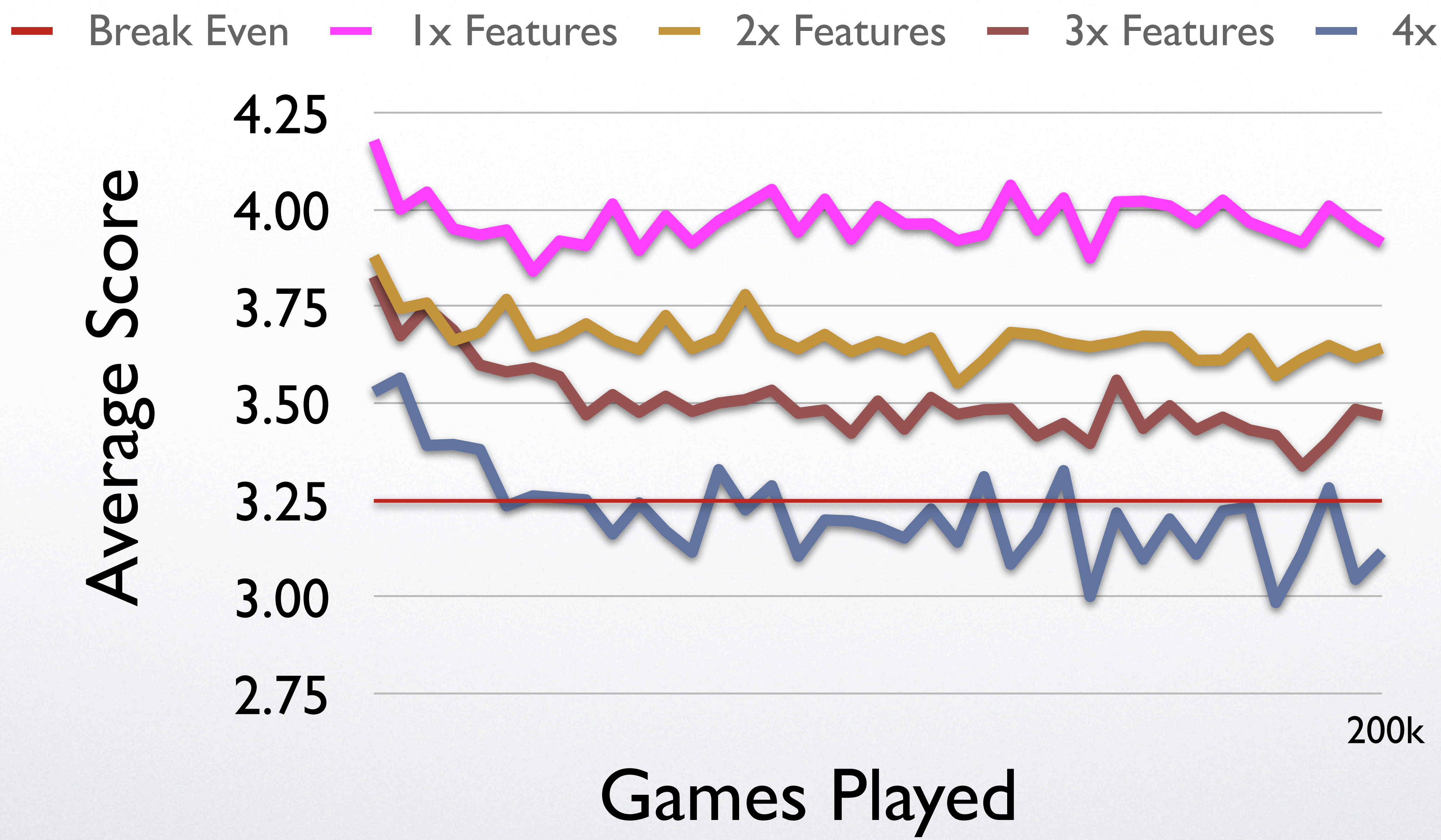
# Even More Features

- 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’
  - Predict expected points in game
  - Train against previous program

# QoS Features



# Analysis

- What is the network learning
  - Easily understand by examining weights assigned to feature sets

# Features - Avoid Q♠

| Rank | Weight | We have | We have | We have | Opponent |
|---|---|---|---|---|---|
| 1 | -0.103 | 1 low ♠ | | Lead | Q♠ no ♠ |
| 2 | -0.097 | 1 low ♠ | No ♥ | Lead | Q♠ no ♠ |
| 3 | -0.096 | 2 low ♠ | K♠ | | Q♠ two ♠ |
| 4 | -0.093 | 1 low ♠ | No ♣ | Lead | Q♠ no ♠ |
| 5 | -0.090 | 1 low ♠ | No ♦ | Lead | Q♠ no ♠ |
| 148 | -0.040 | 1 low ♠ | Q♠ | | Lead no ♠ |

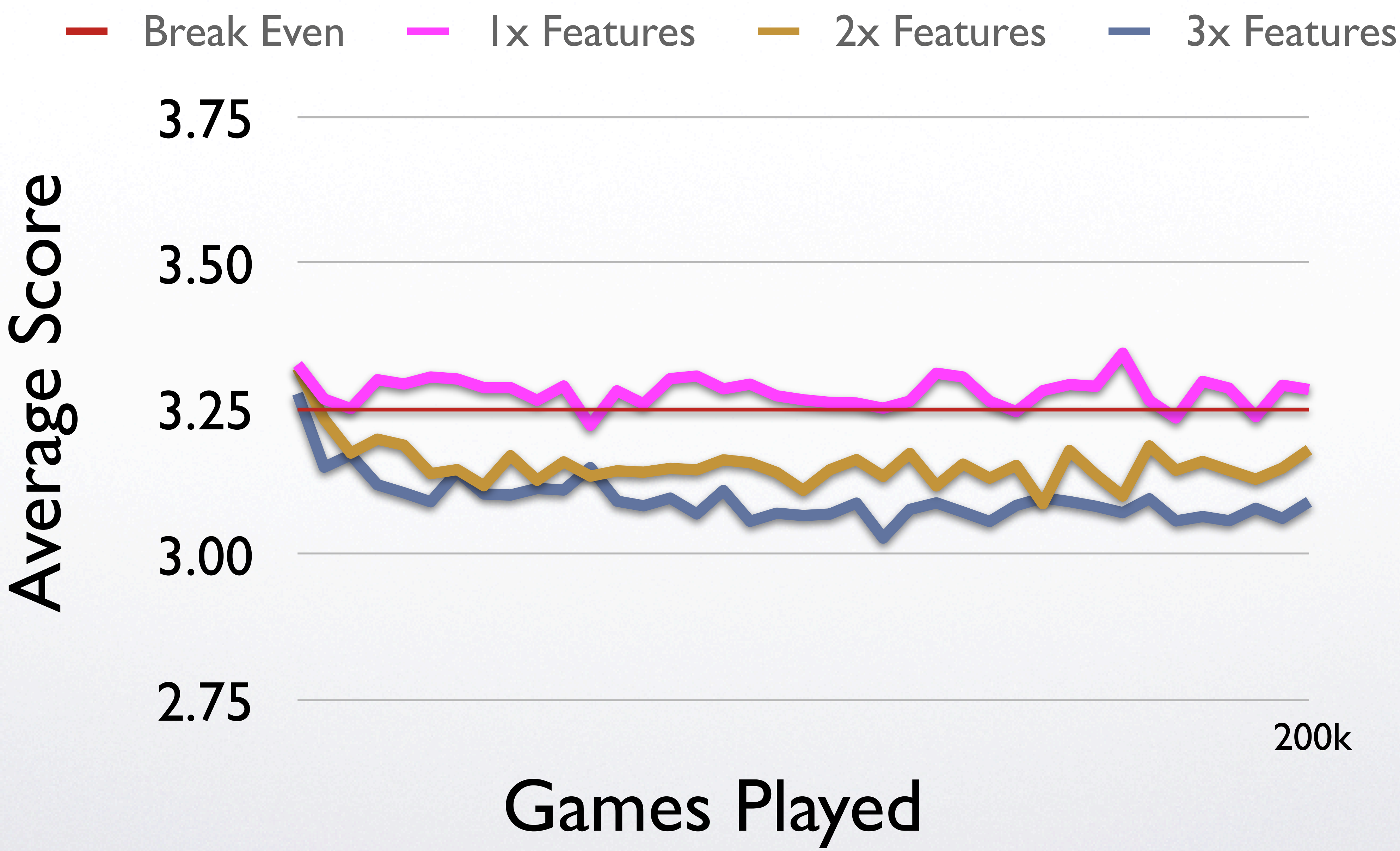
# Features - Take Q♠

| Rank | Weight | We Have | We have | We have | We have |
|---|---|---|---|---|---|
| 1 | 0.125 | Q♠ | 1 low ♠ | | Lead |
| 2 | 0.123 | Q♠ | 1 low ♠ | | |
| 3 | 0.117 | Q♠ | No ♣ | No ♥ | Lead |
| 4 | 0.116 | A/K/Q♠ | | | Lead |
| 5 | 0.112 | Q♠ | No ♣ | No ♥ | No ♦ |

# Learning Part II

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

# Hearts Features



# Learning Part III

- Learn to play the perfect-information game
  - 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

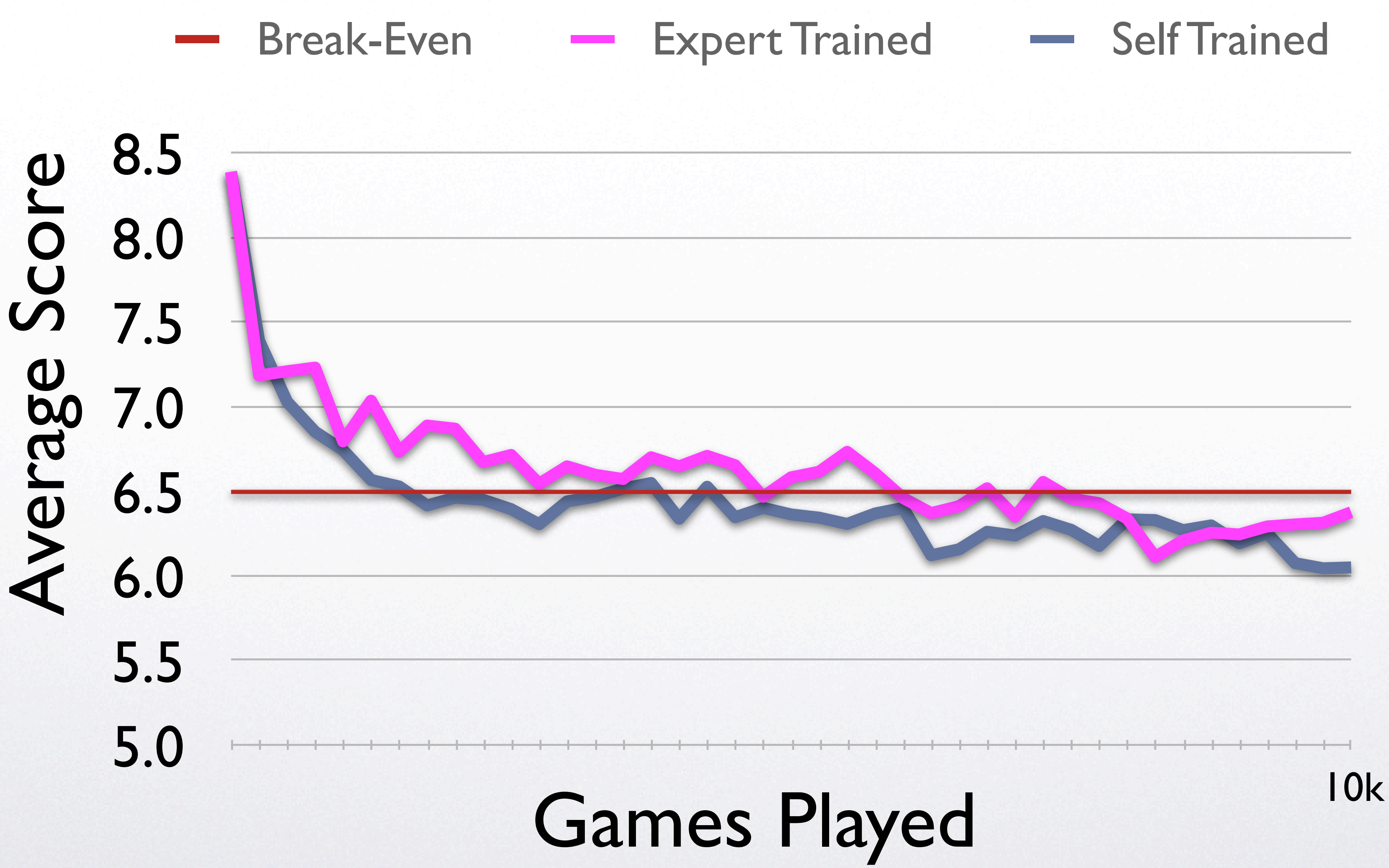
# Steady-State Evaluation

- Test the learned networks
  - Play trained network against expert
  - Play 100 hands
  - 4 players, 2 player types
  - Repeat each hand $2^4$ - 2 times

# Arrangement

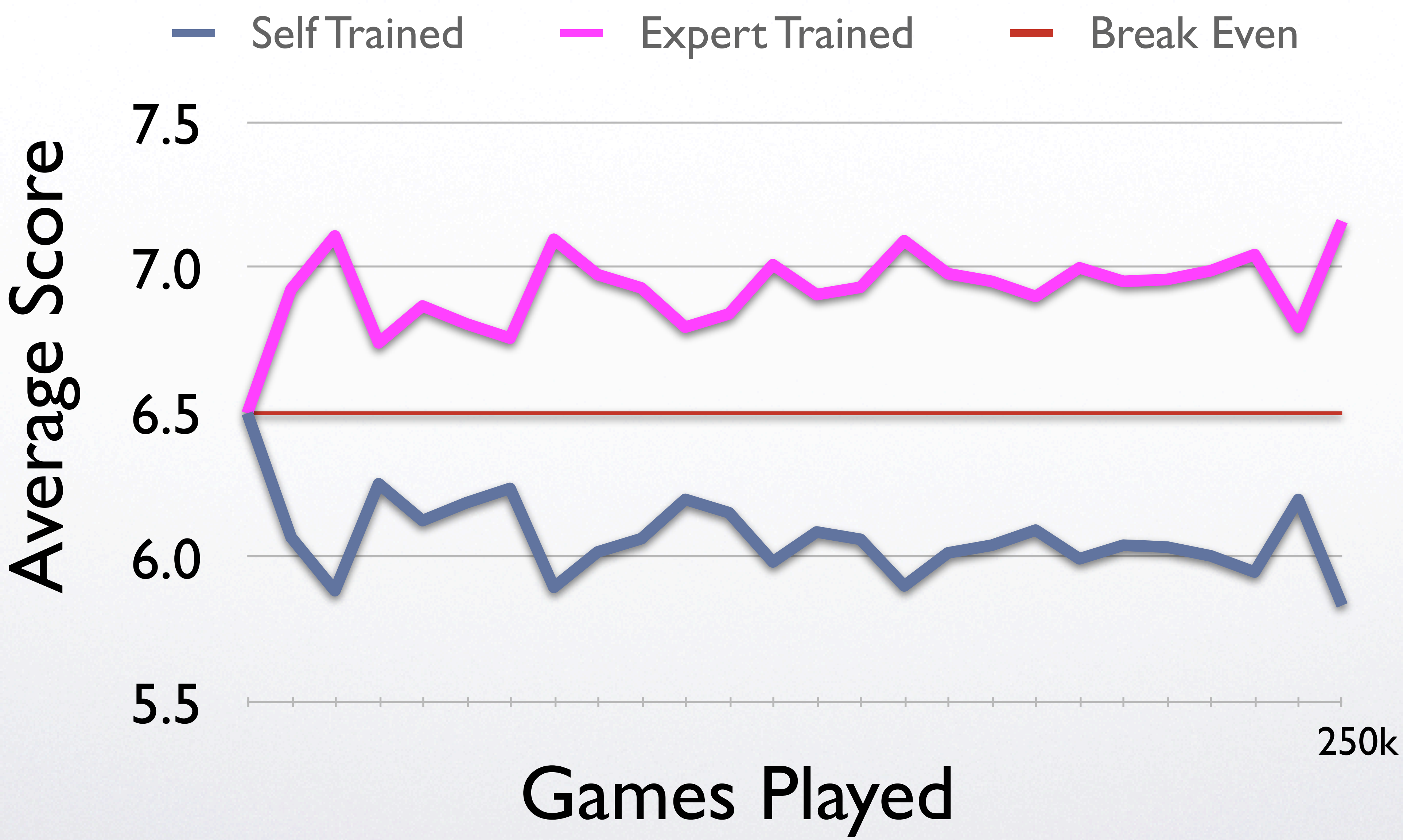
| Player 1 | Player 2 | Player 3 | Player 4 |
|---|---|---|---|
| Expert | Trained | Trained | Trained |
| Trained | Expert | Trained | Trained |
| Expert | Expert | Trained | Trained |
| Trained | Trained | Expert | Trained |
| Expert | Trained | Expert | Trained |
| Trained | Expert | Expert | Trained |
| Expert | Expert | Expert | Trained |

# Games Against Expert



# Games Against Expert

# Trained Play



# Imperfect Info. Play

- Played against expert program
- Single hands
  - 56.9% of hands, 6.35 v. 7.30 average score
- Games to 100 points
  - 63.8% of hands, 69.8 v. 81.1 average score

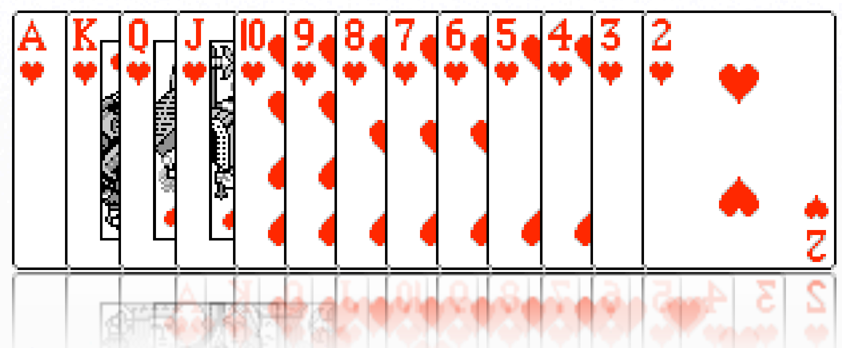
# Summary

- Learned to beat 'expert' by a large margin
  - Program plays well, but sometimes lacks deep analysis of game
- Not a trivial result

# Future Work

- Different algorithms than $max^n$
- Other ways of combining/building features
- Better handling of shooting the moon
- Play against other opponents

# Thank You



INFORMATICS iCORE CIRCLE OF RESEARCH EXCELLENCE