Opponent Exploitation

Sam Ganzfried
Carnegie Mellon University
Computer Science Department

15-780 Graduate AI Spring 2013
Outline

• **Deviation-Based Best Response**: scalable, domain-independent, game-theoretic algorithm for opponent exploitation in imperfect-information games [AAMAS ‘11]

• **Safe Opponent Exploitation**: robust approach for exploiting weak opponents while guaranteeing the value of the game against strong, adaptive opponents [EC’12]
Sequential imperfect-information games

- Most real-world games are sequential & imperfect info
  - Almost any economic situation in which the other participants possess private information (e.g., valuations, quality information)
    - Negotiation
    - Multi-stage auctions (e.g., English, FCC ascending, combinatorial auctions)
    - Sequential auctions of multiple items
  - Many military settings (don’t know exactly what opponents have or their preferences)
  - Card games in which the other players’ cards are hidden, e.g., poker
    - ...

- Challenges
  - Imperfect information
  - Techniques for complete-info games (like chess) don’t apply

- Our techniques are domain-independent
  - We assume players’ actions are observable (but not all nature’s actions)
Game theory

• **Definition.** *Strategy* is a mapping from known history to action

• **In multi-agent systems, an agent’s outcome depends on the actions of others’**

  =>*Agent’s optimal* strategy depends on others’ strategies

• **Definition:** A *Nash equilibrium* is a strategy for each agent such that no agent benefits from using a different strategy
Basics about finding equilibria

• In 2-person zero-sum games,
  – Any equilibrium guarantees at least value of the game in expectation

• Any finite sequential game (satisfying perfect recall) can be converted into a matrix game
  – Exponential blowup in #strategies (even in reduced normal form)

• *Sequence form*: More compact representation based on sequences of moves rather than pure strategies [Romanovskii 62, Koller & Megiddo 92, von Stengel 96]
  – 2-person 0-sum games with perfect recall can be solved in time polynomial in size of game tree using LP
  – Cannot solve Rhode Island Hold’em (3.1 billion nodes) or Texas Hold’em ($10^{18}$ nodes)
Our approach [Gilpin & Sandholm EC’06, JACM’07…]

Now used by all competitive Texas Hold’em programs

Original game

Automated abstraction

Abstracted game

Custom equilibrium-finding algorithm

Nash equilibrium

Reverse model

Nash equilibrium
Traditionally two approaches

• Game theory approach (abstraction+equilibrium finding)
  – Safe in 2-person 0-sum games
  – Doesn’t maximally exploit weaknesses in opponent(s)

• Opponent modeling
  – *Get-taught-and-exploited problem* [Sandholm AIJ-07]
  – Needs prohibitively many repetitions to learn in large games (loses too much during learning)
    • Crushed by game theory approach in Texas Hold’em…even with just 2 players and limit betting
    • Same tends to be true of no-regret learning algorithms
Let’s hybridize the two approaches

• Start playing based on game theory approach

• As we learn opponent(s) deviate from equilibrium, start adjusting our strategy to exploit their weaknesses

• **Motivation:** Start playing well right away and learn to exploit as we collect information about the opponent’s weaknesses during play
Differences from prior research

• Prior research on opponent modeling in imperfect-information games
  – Small games
    • Kuhn poker [Hoehn et al. AAAI-05]
    • Rock-paper-scissors [McCracken & Bowling AAAI Fall Symp.-04]
    • Iterated prisoners’ dilemma [E.g., Chakraborty & Stone ICML-10]
  – Used massive prior datasets of human poker play [Davidson et al IJCAI-00, Ponsen et al AAAI-08]
  – Used expert-generated features or priors [Hoehn et al. AAAI-05, Southey et al UAI-05]
  – Assumed data about opponent’s prior games [Johanson et al NIPS-07, Ponsen et al IDTGT-10]
    • Also assumed that the algorithm has access to opponent’s private info
Main idea of our approach

• Find opponent’s strategy that is “closest” to a pre-computed approximate equilibrium strategy and consistent with his actions so far.
  – E.g., equilibrium raises 50% of the time when first to act, but the opponent raises 30% of the time.
  – This is our opponent model.

• Compute and play an (approximate) best response to the opponent model.
Deviation-Based Best Response (DBBR) algorithm (can be generalized to multi-player non-zero-sum)

- Many ways to determine opponent’s “best” strategy that is consistent with bucket probabilities
  - Weighted $L_1$ or $L_2$ distance to the approx. equilibrium strategy
  - Custom weight-shifting algorithm
  - ...

Compute an approximate equilibrium of the game.
Maintain counters from observing opponent’s play throughout the match.

\[
\text{for } n = 1 \text{ to } \lvert PH_{-i} \rvert \text{ do}
\]
- Compute posterior action probabilities at $n$.
- Compute posterior bucket probabilities at $n$.
- Compute full model of opponent’s strategy at $n$.

\[
\text{return } \text{Best response to the opponent model.}
\]
Geometric ways to construct the opponent model

- Computed separately for each public history set \( n \) (using CPLEX)
- L1-based:
  
  \[
  \begin{align*}
  \text{minimize} & \quad x \\
  \text{subject to} & \quad \sum_{b \in B_n} \sum_{a \in A_n} [\beta_{n,b} \cdot |x_{n,b,a} - \sigma_{n,b,a}^*|] \\
  & \quad \sum_{b \in B_n} [\beta_{n,b} \cdot x_{n,b,a}] = \alpha_{n,a} \text{ for all } a \in A_n \\
  & \quad \sum_{a \in A_n} x_{n,b,a} = 1 \text{ for all } b \in B_n \\
  & \quad 0 \leq x_{n,b,a} \leq 1 \text{ for all } a \in A_n, b \in B_n
  \end{align*}
  \]

- In L2-based, change objective to
  
  \[
  \begin{align*}
  \text{minimize} & \quad x \\
  \text{subject to} & \quad \sum_{b \in B_n} \sum_{a \in A_n} [\beta_{n,b} \cdot (x_{n,b,a} - \sigma_{n,b,a}^*)^2]
  \end{align*}
  \]
Custom weight-shifting algorithm for constructing the opponent model

- Simple greedy algorithm
  - Faster than $L_1$-based and $L_2$-based
  - With this component algorithm, entire DBBR algorithm is linear in the size of the game tree
- E.g.,
  - Opponent raises 30% of time when first to act (i.e., this is our model that combines prior and observed)
  - Equilibrium raises 50% of the time
  - We sort the buckets by how often equilibrium opponent raises in them
  - Greedily remove buckets from his raising range until probability is 30%
- Details:
  - One bucket can be “removed” partially
  - Doing this for all actions one at a time
  - Renormalizing
Texas Hold’em poker

• 2-player Limit Texas Hold’em has $\sim 10^{18}$ leaves in game tree

Nature deals 2 cards to each player

Round of betting

Nature deals 3 shared cards

Round of betting

Nature deals 1 shared card

Round of betting

Nature deals 1 shared card

Round of betting

On NBC:
Experimental setup

- Parameters in our algorithm (not carefully tuned):
  - $N_{\text{prior}} = 5$
  - Start exploiting after 1000 iterations
  - Recompute strategy every 50 iterations (not every time to save time)
  - Much coarser abstraction than $GS5$ so strategy can be recomputed in few seconds
    - Bucket branching factors 8, 12, 4, 4 instead of 15, 40, 6, 6

- Each pairing of bots contains multiple matches of 3000 duplicate hands each
Experimental results

- All 3 variants significantly outperform the game-theory-based base strategy (GS5) against trivial opponents and weak opponents from AAAI computer poker competitions
- Selective superiority
- DBBR-WS performs best against the real opponents
- DBRR-WS is by far the fastest

Table 1: Win rate in small bets/hand of the bot listed in the row. The ± given is the standard error (standard deviation divided by the square root of the number of hands).
Examples of our win rate (sb/hand) evolution
• Deviation-Based Best Response: scalable, domain-independent, game-theoretic algorithm for opponent exploitation in imperfect-information games [AAMAS ‘11]

• Safe Opponent Exploitation: robust approach for exploiting weak opponents while guaranteeing the value of the game against strong, adaptive opponents [EC’12]
Your friend challenges you to a poker game

- If he is bad, you would like to crush him
- If he is good, you would like to ensure that you still beat him
- Can you crush him if he is bad while guaranteeing victory even if he is good?
Overview

- Background
- Safe exploitation
- Algorithms for safe exploitation
- Experiments
Game theory

<table>
<thead>
<tr>
<th></th>
<th>rock</th>
<th>paper</th>
<th>scissors</th>
</tr>
</thead>
<tbody>
<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>

- Players
- Actions (aka pure strategies)
- Strategy profile: e.g., (R, p)
- Utility function: e.g., $u_1(R, p) = -1$, $u_2(R, p) = 1$
### Zero-sum game

<table>
<thead>
<tr>
<th></th>
<th>rock</th>
<th>paper</th>
<th>scissors</th>
</tr>
</thead>
<tbody>
<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>

- Sum of payoffs is zero at each strategy profile: e.g., \( u_1(R,p) + u_2(R,p) = 0 \)
- Models purely adversarial settings
Mixed strategies

• Probability distributions over pure strategies
• E.g., R with prob. 0.6, P with prob. 0.3, S with prob. 0.1
Best response (aka nemesis)

• Any strategy that maximizes payoff against opponent’s strategy
• If P2 plays (0.6, 0.3, 0.1) for r,p,s, then a best response for P1 is to play P with probability 1
Nash equilibrium

- Strategy profile where all players simultaneously play a best response
- Standard solution concept in game theory
  - Guaranteed to always exist in finite games [Nash 1950]
- In Rock-Paper-Scissors, the unique equilibrium is for both players to select each pure strategy with probability $1/3$
Minimax Theorem

- Minimax theorem: For every two-player zero-sum game, there exists a value $v^*$ and a mixed strategy profile $\sigma^*$ such that:
  a. $P_1$ guarantees a payoff of at least $v^*$ in the worst case by playing $\sigma^*_1$
  b. $P_2$ guarantees a payoff of at least $-v^*$ in the worst case by playing $\sigma^*_2$

- $v^*$ is the value of the game
- All equilibrium strategies for player $i$ guarantee at least $v_i$ in the worst case
- For RPS, $v^* = 0$
Exploitability

• Exploitability of a strategy is difference between value of the game and performance against a best response
  – Every equilibrium has zero exploitability
• Always playing rock has exploitability 1
  – Best response is to play paper with probability 1
Exploitation-exploitability tradeoff

- Want to achieve high levels of exploitation against weak opponents
- Want a low exploitability so that we can guarantee a good worst-case payoff against strong opponents
- Can we achieve both of these simultaneously?
Exploitation-exploitability tradeoff

- Exploitation
  - Exploitability
    - Nash equilibrium
    - ????
    - Full opponent exploitation

?-?
Overview

- Background
- Safe exploitation
- Algorithms for safe exploitation
- Experiments
Repeated game

- Repeat *stage game* for T iterations
- Overall payoff is cumulative total
- Strategies can be contingent on play in prior rounds
Safe exploitation

• A strategy is *safe* if it obtains payoff of at least $v_i$ per iteration in expectation regardless of the strategy used by the opponent.

• Playing an equilibrium strategy at each iteration is safe since it guarantees at least $v_i$ in each iteration.

• Do there exist any other safe strategies that deviate from stage-game equilibrium?
Rock-Paper-Scissors

• Suppose the opponent has played Rock in each of the first 10 iterations, while we have played the equilibrium $\sigma^*$

• Can we exploit him by playing pure strategy Paper in the 11$^{th}$ iteration?
  – Yes, but this would not be safe!

• By similar reasoning, any deviation from $\sigma^*$ will be unsafe

• So safe exploitation is not possible in Rock-Paper-Scissors
Rock-Paper-Scissors-Toaster

<table>
<thead>
<tr>
<th></th>
<th>rock</th>
<th>paper</th>
<th>scissors</th>
<th>toaster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>0,0</td>
<td>-1, 1</td>
<td>1, -1</td>
<td>4, -4</td>
</tr>
<tr>
<td>Paper</td>
<td>1,-1</td>
<td>0, 0</td>
<td>-1,1</td>
<td>3, -3</td>
</tr>
<tr>
<td>Scissors</td>
<td>-1,1</td>
<td>1,-1</td>
<td>0,0</td>
<td>3, -3</td>
</tr>
</tbody>
</table>

- \( t \) is *strictly dominated*
  - \( s \) does strictly better than \( t \) regardless of \( P1 \)’s strategy
- Suppose we play NE in the first round, and he plays \( t \)
  - Expected payoff of 10/3
- Then we can play R in the second round and guarantee at least 7/3 between the two rounds
- Safe exploitation is possible in RPST!
  - Because of presence of ‘gift’ strategy \( t \)
Characterizing ‘gifts’

• In the preceding example, \( t \) was a strictly dominated pure strategy. What about other forms of dominance?
  – Weak dominance
  – Iterated dominance
  – Dominance by mixed strategies, etc.
Characterizing ‘gifts’

• Proposition: all non-weakly-iteratively-dominated strategies achieve the value of the game against all equilibrium strategies of the opponent [Waugh ‘09]
• Unique equilibrium: P1 plays U and D with prob. \( \frac{1}{2} \), and P2 plays L and M with prob. \( \frac{1}{2} \). Value to P1 is 2.5
• If P1 plays NE and P2 plays R, P1 gets 5
• So R is a gift, and P1 can safely deviate from NE to exploit
• But R is not dominated under any form of dominance!
Characterizing ‘gifts’ cont’d

- Definition: A strategy $\sigma_{-i}$ is a gift strategy if there exists an equilibrium strategy $\sigma^*_i$ for player $i$ such that $\sigma_{-i}$ is not a best response to $\sigma^*_i$.

- Proposition: Non-stage-game-equilibrium safe strategies exist if and only if there exists at least one gift strategy for the opponent.
Overview

• Background
• Safe exploitation
• Algorithms for safe exploitation
• Experiments
Safe best responses

- Define $\text{SAFE}(\epsilon)$ to be the set of strategies with exploitability at most $\epsilon$.
- Define the $\epsilon$-safe best response of player $i$ to $\sigma_{-i}$ to be the strategy in $\text{SAFE}(\epsilon)$ obtaining highest payoff against $\sigma_{-i}$.
Risk What You’ve Won (RWYW)

• Set $k^1 = 0$

• for $t = 1$ to $T$ do
  – Set $\pi^t_i$ to be $k^t$-safe best response to $M$
  – Play action $a^t_i$ according to $\pi^t_i$
  – Update $M$ with opponent’s action $a^t_{-i}$
  – Set $k^{t+1} = k^t + u_i(a^t_i, a^t_{-i}) - v^*$

• Is RWYW safe?
Risk What You’ve Won (RWYW)

- Proposition: RWYW is not safe!
- RWYW does not adequately differentiate between whether profits due to skill (i.e., from gifts) or luck
Risk What You’ve Won in Expectation (RWYWE)

• Set $k^1 = 0$

• for $t = 1$ to $T$ do
  – Set $\pi^t_i$ to be $k^t$-safe best response to $M$
  – Play action $a^t_i$ according to $\pi^t_i$
  – Update $M$ with opponent’s action $a^t_{-i}$
  – Set $k^{t+1} = k^t + u_i(\pi^t_i, a_{-i}) - v^*$

• Proposition: RWYWE is safe
Best Equilibrium Followed by Full Exploitation (BEFFE)

- Play a full best response if
  \[ k^t \geq \epsilon (T - t + 1) \]
  - \( \epsilon \) is exploitability of a full best response
- Otherwise play best equilibrium

- BEFFE plays best equilibrium, then full best response at the end
Best Equilibrium Followed by Full Exploitation (BEFFE)

• Advantage of BEFFE over RWYWE:
  – Saves up accumulated gifts until the end, when it has most accurate info on opponent

• Disadvantage:
  – Possibly misses out on additional rounds of exploitation by waiting until the end
Best Equilibrium and Full Exploitation When Possible (BEFEWP)

- Similar to prior algorithms, but only exploits when the exploitability of a full best response is below $k^t$; otherwise plays best Nash equilibrium
- Alternates between best Nash equilibrium and full best response
Summary of our safe exploitation algorithms

• From most aggressive to least aggressive:
  1. RWYWE
     – Plays $k^t$-safe best response at each iteration
  2. BEFEWP
     – Alternates between full best response and best NE
  3. BEFFE
     – Plays best NE for several iterations, then full best response at the end
Full characterization of safe strategies in matrix games

• An algorithm is expected-profit-safe if it selects $\pi_t$ in $\text{SAFE}(k^t)$ for each $t$, where $k^1 = 0$ and $k$ is updated using the rule:
  - $k^{t+1} \leftarrow k^t + u_i(\pi^t, a_{-i}) - v^*$

• Proposition: a strategy is safe if and only if it is expected-profit-safe
Extensions to more complex game representations

- Analogous results in sequential games of perfect and imperfect information
- Must be pessimistic about how the opponent would play off the path of play and with unobserved private information
Overview

• Background
• Safe exploitation
• Algorithms for safe exploitation
• Experiments
Kuhn poker [Kuhn 1950]

- Two-player zero-sum game, consisting of a three-card deck and a single round of betting
- Value of the game to P1 is $-1/18 = -0.0556$
- P2 has unique NE strategy, while P1 has infinitely many
Experimental setup

- We experimented with RWYWE, BEFFE, BEFEWP, Best Nash, and Full Best Response
- For all algorithms, we used a natural opponent modeling algorithm
  - Assumes opponent plays according to observed frequencies so far, where we observe his hand after each iteration
- Adapted all algorithms to the imperfect-information setting using pessimistic update rule
- 1000 hands/match
- Multiple matches for each algorithm/opponent class combination
Opponent classes

• Random
  – Static strategy with probabilities chosen uniformly at random at each information set

• Sophisticated static
  – Static strategy with probabilities chosen randomly within 0.2 of equilibrium probabilities

• Dynamic
  – Plays static random strategy for 100 iterations, then plays a best response to our strategy

• Equilibrium
Results

- (Game value is -0.055)
- All the exploitative safe algorithms outperform Best Nash against the static opponents
- RWYWE did best against static opponents
- Against dynamic opponents, best response does much worse than value of the game

<table>
<thead>
<tr>
<th>Opponent</th>
<th>Random</th>
<th>Sophisticated static</th>
<th>Dynamic</th>
<th>Equilibrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWYWE</td>
<td>0.363 ± 0.003</td>
<td>-0.0104 ± 0.0013</td>
<td>-0.021 ± 0.003</td>
<td>-0.055 ± 0.001</td>
</tr>
<tr>
<td>BEFEWP</td>
<td>0.353 ± 0.003</td>
<td>-0.0111 ± 0.0013</td>
<td>-0.020 ± 0.003</td>
<td>-0.054 ± 0.001</td>
</tr>
<tr>
<td>BEFFE</td>
<td>0.199 ± 0.003</td>
<td>-0.0121 ± 0.0013</td>
<td>-0.041 ± 0.003</td>
<td>-0.054 ± 0.001</td>
</tr>
<tr>
<td>Best Nash</td>
<td>0.143 ± 0.003</td>
<td>-0.0142 ± 0.0013</td>
<td>-0.035 ± 0.003</td>
<td>-0.054 ± 0.001</td>
</tr>
<tr>
<td>Best response</td>
<td>0.470 ± 0.003</td>
<td>0.0545 ± 0.0014</td>
<td>-0.121 ± 0.003</td>
<td>-0.055 ± 0.001</td>
</tr>
</tbody>
</table>
Gift accumulation of RWYWE

- In some matches, RWYWE steadily accumulates gifts along the way, and $k^t$ increases throughout the match.
- When this happens, we play a full best response in most iterations.
Gift accumulation of RWYWE

- In other matches, $k^t$ remains very close to 0 throughout, despite the fact that profits steadily increase.
- In this situation, we are frequently playing an equilibrium and only occasionally playing a full best response.
- Note that $k^t$ falling to 0 does not mean that we are losing; just that we are erring on side of caution to ensure safety.
Conclusions

• Safe opponent exploitation is possible in certain games
• We presented several new safe exploitative algorithms
• We provided a full characterization of safe strategies
• Experiments show that safe exploitation is feasible and potentially effective in realistic settings
• Our most aggressive safe exploitation algorithm (RWYWE) performed best
Recap

• **Deviation-Based Best Response**: scalable, domain-independent, game-theoretic algorithm for opponent exploitation in imperfect-information games [AAMAS ‘11]

• **Safe Opponent Exploitation**: robust approach for exploiting weak opponents while guaranteeing the value of the game against strong, adaptive opponents [EC’12]
Questions?