Solving Games with Regret Estimation

Kevin Waugh, Dustin Morrill, Drew Bagnell, Michael Bowling

AAAI Spring Symposium 2015
Computational Game Theory
Extensive-form Games
zero-sum with imperfect information

- Limit Texas Hold’em — $3.59 \times 10^{13}$
- No-limit Texas Hold’em — $7.16 \times 10^{175}$

[Johanson14, Bowling et al.15]
Standard Approach

Abstraction

(i) Large Game → Abstract Game

(ii) Abstract Game → Equilibrium Computation

Reverse Mapping

(iii) Abstract Strategy → Full Game Strategy
Iterative dominance: (a) => not (b), cannot remove (a)
Regression
Counterfactual Regret Minimization
equilibrium computation using domain-specific features

Large Game

Full Game Strategy

Abstract Strategy

(i) → (ii)

(iii)
Extensive-form Game
Histories
/Jack/raise

1 — call — 2
-2 — fold — 1

-1 — fold — raise

Jack 1/2 — call — King 1/2

fold — raise — fold
Nature’s Strategy

- Jack
  - 1/2

- King
  - 1/2
Information Sets and Partitions

Jack

King
Behavioral Strategy
Terminal Histories
Equilibrium Computation

A pair of strategies, one for each player, s.t. neither can benefit by deviating

Zero-sum $\Rightarrow$ Equilibrium is minimax optimal
Green should never fold the King
Call $p$ the probability green bets the Jack

Value to red of calling:
\[
\frac{1}{2}(2p + 1(1 - p)) + \frac{1}{2}(-2) = \frac{p - 1}{2}
\]

Value to red of folding:
\[-1\]
Nash: Green folds Jack, Red always calls
Counterfactual Regret Minimization

Two no-regret learners in self-play converge to an equilibrium
(difference between learner and best in hindsight goes to zero)

Minimizing counterfactual regret at each information set minimizes overall regret
(utility given you play to reach an information set)
Counterfactual Regret Minimization using regret-matching  
\[ \text{time} = 1 \]

\[
\begin{array}{ccc}
\text{regret} & \text{policy} & \text{utility} \\
(0) & (0.5) & (-0.5) \\
(0) & (0.5) & (-0.25)
\end{array}
\]

\[ \sigma(a) \propto \max\{\text{regret}(a), 0\} \]

\[
\begin{array}{ccc}
(0) & (0.5) & (-0.5) \\
(0) & (0.5) & (0.75)
\end{array}
\]

\[
\begin{array}{ccc}
(0) & (0.5) & (-0.5) \\
(0) & (0.5) & (0)
\end{array}
\]

[Hart and Mas-Colell 2000]
Counterfactual Regret Minimization using regret-matching

<table>
<thead>
<tr>
<th>Jack</th>
<th>King</th>
</tr>
</thead>
<tbody>
<tr>
<td>time = 2</td>
<td>time = 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>regret</th>
<th>policy</th>
<th>utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-0.125)</td>
<td>(0)</td>
<td>(-0.5)</td>
</tr>
<tr>
<td>(0.125)</td>
<td>(1)</td>
<td>(-1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>regret</th>
<th>policy</th>
<th>utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-1.125)</td>
<td>(1)</td>
<td>(-0.5)</td>
</tr>
<tr>
<td>(1.125)</td>
<td>(0)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>regret</th>
<th>policy</th>
<th>utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-0.25)</td>
<td>(1)</td>
<td>(-0.5)</td>
</tr>
<tr>
<td>(0.25)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
</tbody>
</table>
Counterfactual Regret Minimization using regret-matching time = 10

average policy

\[
\begin{pmatrix}
0.825 \\
0.175
\end{pmatrix}
\]

\[
\begin{pmatrix}
0.005 \\
0.995
\end{pmatrix}
\]

\[
\begin{pmatrix}
0.05 \\
0.95
\end{pmatrix}
\]
Large Game

Abstract Game

Full Game Strategy

Abstract Strategy

Abstraction

Equilibrium Computation

(i)

(ii)
Cluster hands together using features of hand strength
\[
\tilde{r}^t = \begin{pmatrix} -4 \\ -4 \\ 1 \end{pmatrix}
\]

approximating regrets with piecewise constant function!
• Do we need a piecewise constant representation?
• Does the representation need to be static?
\[ \tilde{R}^{t+1} : \Phi \rightarrow \mathbb{R} \]

\[ \tilde{R}^{t+1}(x) \approx \tilde{R}^t(x) + \begin{pmatrix} -6 \\ -4 \\ 0 \end{pmatrix} \]

\[ \tilde{R}^{t+1}(y) \approx \tilde{R}^t(y) + \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \]

L$_2$ regression to the rescue
Theorem

Additional regret of regression CFR is bounded by the regressor’s expected $L_2$ error equivalent to an additive error introduced by abstraction.
Large Game

(i) Train Regressor (abstraction)

(ii) Compute Update (equilibrium computation)

Abstract Strategy
In the future

- Engineering details for large games:
  - online regression to avoid explicit training set ✓
  - approximate updates via sampling ✓
  - regularization? ✓ starting conditions? ✓
  - feature engineering?
  - fast-crude vs. expensive-precise updates?
Thank you!

waugh@cs.cmu.edu

mbowling@ualberta.ca

dbagnell@ri.cmu.edu

morrill@ualberta.ca