AAAI-17 Tutorial on Computer Poker
Part 3: Local Search and Reinforcement Learning

Johannes Heinrich

Computer Science
University College London

February 4th 2017
Topics

Scope

- Local Tree Search
  - IS-MCTS
  - Online Outcome Sampling
  - Continuous Resolving

- Reinforcement Learning from Self-Play
  - Extensive-Form Fictitious Play
  - (Neural) Fictitious Self-Play
Local Tree Search

Problems with full-game equilibria

- Memory (Space requirements)
  - How do we obtain an equilibrium if we cannot fit the whole game tree in memory?

- Computation (Time requirements)
  - How do we compute an equilibrium in a reasonable amount of time?

- Locality
  - How do we map a real information state (sequence) to our full-game abstraction?
Local Tree Search

Approach

Compute or refine an approximate equilibrium locally for the situation at hand.

- Standard approach in large-scale perfect-information games
  - e.g. MCTS in Go
Local Tree Search

IS-MCTS

- Extension of common MCTS to imperfect-information games:
  - One information-state subtree per player
  - Sampling from local belief state
  - Information-state policy regret minimizing or fictitious play

- Achieved practical successes
  - E.g. Imperfect-information board&card games, including LHE

- Lacking equilibrium guarantees
  - Exploitable in local search
  - May plateau in full-game search
Local Tree Search

Equilibrium recovery

**Figure:** Bluff catching game
Local Tree Search

Online Outcome Sampling
Refs: Lisy et al. 2015

- Focus Outcome Sampling (MCCFR) on relevant information states
- Achieves equilibrium guarantees
- Does not fully address memory, computation and abstraction-mapping problems
Local Tree Search

Continuous resolving
Refs: Burch et al. 2014, Moravcik et al. 2017

- Solve gadget game that hashes essential information of the global equilibrium in counterfactual values
  - Recovers global equilibrium with strictly local search
  - Continuous resolving carries forward the counterfactual values

- Resolving may be intractable except at the end of the game
  - DeepStack bootstraps a limited-depth search from learned counterfactual values
Topics

Scope

- Local Tree Search
- Reinforcement Learning from Self-Play
  - Extensive-Form Fictitious Play
  - (Neural) Fictitious Self-Play
Reinforcement Learning from Self-Play

Problems

► Handcrafted abstractions
  - Do you have the domain expertise and time to handcraft an abstraction?
    ▶ Different variants may require different kinds of abstraction, e.g. limit, no-limit, stud, draw poker variants
  - How do we abstract action sequences for multi-player variants?

► Static abstractions
  - Did your abstraction lose essential information for achieving a desired equilibrium quality?
  - Does it have any deficiencies or result in pathologies?

► Scalability
  - Does your approach scale well with the number of players?
Reinforcement Learning from Self-Play

Approach

Learn strategies, $\pi$, that optimize action-values

$$Q_\pi(u, a) = \mathbb{E}_\pi \left[ \sum_{k=t}^{T} R_{k+1} \middle| U_t = u, A_t = a \right]$$
Reinforcement Learning from Self-Play

Generalised weakened fictitious play
Refs: (Brown, 1951; Leslie & Collins, 2006)

At each iteration each player

1. Computes a best response to fellow players’ average strategies
2. Updates own average strategy with computed best response

\[ \Pi_{k+1} \in \Pi_k + \alpha_{k+1} \left( \text{BR}_{\epsilon_{k+1}} [\Pi_k] - \Pi_k + M_{k+1} \right) \]

Original fictitious play: \( \alpha_k = \frac{1}{k}, \epsilon_k = 0, M_k = 0 \)
Fictitious play & reinforcement learning

- Fictitious players choose best (payoff-maximizing) strategies in hindsight.
- Reinforcement learners learn optimal (reward-maximizing) policies from their experience.
Reinforcement Learning from Self-Play

Extensive-Form Fictitious Play (XFP)
Refs: Koller et al. 1994; Von Stengel 1996; Heinrich et al. 2015

► Compute best responses with dynamic programming

\[ \beta_{t+1} \in \text{BR}(\pi_t), \]

► Update average strategies

\[ \pi_{t+1}^i(u) = \pi_t^i(u) + w_{t+1}(u) (\beta_{t+1}^i(u) - \pi_t^i(u)) \]

weighted by conditional probability (of playing best response at respective information state)

\[ w_{t+1}^i(u) = \frac{\alpha_{t+1} x_{\beta_{t+1}^i} (\sigma_u)}{(1 - \alpha_{t+1}) x_{\pi_t^i} (\sigma_u) + \alpha_{t+1} x_{\beta_{t+1}^i} (\sigma_u)} \]
## Reinforcement Learning from Self-Play

### Fictitious Self-Play

Refs: Heinrich et al. 2015

<table>
<thead>
<tr>
<th></th>
<th>XFP</th>
<th>FSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best response</td>
<td>Dynamic programming in whole tree</td>
<td>Reinforcement learning from sampled trajectories of past best responses</td>
</tr>
<tr>
<td>Average strategy</td>
<td>Explicit update in whole tree</td>
<td></td>
</tr>
</tbody>
</table>
Fictitious Self-Play

Learning a best response

- Each agent plays its average strategy, $\pi$
- Each agent learns a best response strategy by maximizing and evaluating its Q values with off-policy reinforcement learning

$$Q^i(u, a) \approx \mathbb{E}_{\beta^i, \pi^{-i}} \left[ \sum_{k=t}^{T} R_{k+1} \bigg| U_t = u, A_t = a \right]$$

e.g. $\beta^i = \text{greedy}(Q^i)$
Fictitious Self-Play

Learning the average strategy

To learn

$$\Pi^i_k = \sum_{j=1}^{k} w_j B^i_j$$

1. Sample a proportion of $w_j$ episodes from each $\beta^i_j$ respectively
2. Store experienced state-action pairs, $(u_t, a_t)$, in a memory
3. Train the average strategy with imitation learning from the behaviour data in memory
Fictitious Self-Play

Memorizing the Average Strategy

Infinite stream of state-action pairs \((u_t, a_t)\) can be memorized in an online fashion and with finite memory capacity:

▶ Counting model, \(N_k(u, a)\)

\[
\pi_k(a | u) = \frac{N_k(u, a)}{N_k(u)}
\]

▶ Reservoir sampling

- Tracks a finite random sample of a possibly large or infinite stream of items (here, best responses)
Reinforcement Learning from Self-Play

 Neural Fictitious Self-Play
 Refs: Heinrich&Silver 2016

1. Train an action-value network, $Q(s, a | \theta^Q)$, that approximates best response with neural fitted Q-learning (DQN)
   - $\beta = \epsilon$-greedy ($Q$)

2. Train an average-policy network, $\Pi(s, a | \theta^\Pi)$, that approximates own average behaviour with imitation learning
   - $\pi = \Pi$
Neural Fictitious Self-Play

Sampling Experience

Each agent

- Uses policy $\sigma = \begin{cases} 
\epsilon\text{-greedy}(Q), & \text{with probability } \eta \\
\Pi, & \text{with probability } 1 - \eta 
\end{cases}$

- Stores its transitions $(u_t, a_t, r_{t+1}, u_{t+1})$ in reinforcement learning memory $\mathcal{M}_{RL}$

- Stores its behaviour tuples $(s_t, a_t)$ in supervised learning memory $\mathcal{M}_{SL}$, when following its best response strategy ($\epsilon\text{-greedy}(Q)$)
Neural Fictitious Self-Play

Reinforcement learning of best response

Refs: Mnih et al. 2015

Train $Q(s, a \mid \theta^Q)$ with SGD on

$$
\mathcal{L} (\theta^Q) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{M}_{RL}} \left[ \left( r + \max_{a'} Q(s', a' \mid \theta^{Q'}) - Q(s, a \mid \theta^Q) \right)^2 \right]
$$
Neural Fictitious Self-Play

Imitation learning of average strategy

Train $\Pi(s, a \mid \theta^\Pi)$ with SGD on

$$L(\theta^\Pi) = \mathbb{E}_{(s,a) \sim \mathcal{M}_{SL}} \left[ - \log \Pi(s, a \mid \theta^\Pi) \right]$$
Neural Fictitious Self-Play

Experiments in Limit Texas Hold’em

Figure: Win rates of NFSP against SmooCT in Limit Texas Hold’em. The estimated standard error of each evaluation is less than 10 mbb/h.
Neural Fictitious Self-Play

Experiments in Limit Texas Hold’em

<table>
<thead>
<tr>
<th>Match-up</th>
<th>Win rate (mbb/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>escabeche</td>
<td>-52.1 ± 8.5</td>
</tr>
<tr>
<td>SmooCT</td>
<td>-17.4 ± 9.0</td>
</tr>
<tr>
<td>Hyperboreean</td>
<td>-13.6 ± 9.2</td>
</tr>
</tbody>
</table>

Table: Win rates of NFSP’s greedy-average strategy against the top 3 agents of the ACPC 2014.
Neural Fictitious Self-Play

Visualization of a poker-playing neural network

Figure: t-SNE embedding of first player’s last hidden layer activations, coloured by A) action probabilities; B) round of the game; C) initiative feature; D) pot size in big bets (logarithmic scale).
Neural Fictitious Self-Play

Visualization of a poker-playing neural network

- A: Preflop
- B: Pairs on river, after check-calling down
- C: Pairs on flop, facing continuation bet after big-blind defense
- D: Straight draws on the turn
- E: Busted straight draws on the river, after bluffing on the turn
Reinforcement Learning from Self-Play

Why?

Self-play RL can be applied to a variety of poker games, including multi-player, without having to design abstractions. E.g. full ring stud poker or pot limit Omaha
Thanks!