Online Feature Discovery in Relational Reinforcement Learning

Scott Sanner

University of Toronto
Overview

1. Use well-known techniques:
   - Monte Carlo RL (i.e., $TD(\lambda = 1)$)
   - Naïve Bayes classifier
   - Locally-weighted regression
   - Apriori data mining algorithm

2. Combine them in a novel way that...
   - Is space/time efficient for large relational state spaces
   - Achieves encouraging empirical results in game domains
     (TicTacToe, Othello, Backgammon)
RRL: Advantages and Challenges

- RRL is a natural representation/learning paradigm:
  - Describe state using relational features: \( \{At(O, 1, 1), At(X, 2, 3)\} \)
  - Admits compact descriptions:
    * **Closed-world assumption (CWA):** If not inferred true, assume false
    * **Quantifiers/Connectives:** \( \exists p, r. At(p, r, 1) \land At(p, r, 2) \land At(p, r, 3) \)

- But, benefits are not without drawbacks:
  - Very large ground relational state space:
    40 ground atoms \( = 2^{40} \) states
  - Need robust learning for sparse data:
    few samples per state \( \implies \) high variance
  - Must focus on time/space efficient approximations
RRL: Addressing these Challenges

• General solution difficult, focus on restricted setting:
  – Goal-oriented tasks (e.g., planning, games w/ stationary opp.)
  – Indefinite horizon, undiscounted MDP domains
  – Single terminal reward of success/failure

• ⇒ Value function = probability of success

• Allows us to address previous RRL challenges:
  – Very large state spaces: Naïve Bayes repr. of value function
  – Robust learning: Augment with high-freq. joint features (Apriori alg.)
  – Efficient approximation: Use ML estimate of value fun. (closed-form)
Theoretic Preliminaries

- Under a fixed policy $\pi$, MDP reduces to a Markov chain:

- Undiscounted, only non-zero reward is on success trans.

- Value function is prob. of reaching success in $\infty$ limit:

$$V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} r^t | S^t=0 = s] = P(S^t=\infty = success | S^t=0 = s, \pi)$$
Relational State Representation

- \( \{R_1, \ldots, R_i\} \): Set of relations used to describe a state
- \( \{A_1, \ldots, A_j\} \): Set of relation attribute types
  - TicTacToe: \( At(Mark, Pos, Pos) \); \( Mark = \{X, O\}, Pos = \{1, 2, 3\} \)
  - 18 ground atoms: \{\( At(X, 1, 1) \), \( At(X, 1, 2) \), \ldots, \( At(O, 3, 2) \), \( At(O, 3, 3) \)\}
  - \( 2^{18} \) possible truth assignments = 262,144 states
- \( F = \{F_1, \ldots, F_n\} \): Ground rel. atoms (boolean features)
- \( f = \{f_1, \ldots, f_p, \overline{f}_{p+1}, \ldots, \overline{f}_n\} \): Feature truth assignment
  - Order true/positive features first, false/negative features last
  - Represent state \( f \) as \( \{f_1, \ldots, f_p\} \), make CWA
  - Space efficient because typically \( p \ll n \)
Value Function Representation

- Computational and representational issues aside:
  - Let $W$ be a boolean variable denoting eventual win/success
  - Optimal value function under a fixed policy is $P(W|F_1, \ldots, F_n)$
  - Learning = direct estimate of $P(W|F_1, \ldots, F_n)$ from trial data

- Unfortunately, $P(W|F_1, \ldots, F_n)$ is intractably large... so approximate it with a naïve Bayes net, e.g.,

- ML parameters are just observed frequencies
Efficient Policy Evaluation

● Still many features, need to eval policy efficiently:
  – Focus on policy evaluation via after-state analysis
  – Policy eval. is just choice of highest valued after-state
  – This is state that maximizes log winning odds $\log\left(\frac{P(w|f)}{P(w|\bar{f})}\right)$

\[
\log \frac{P(w|f)}{P(w)} = \log \frac{P(w)}{P(w)} + \sum_{i=1}^{p} \log \frac{P(f_i|w)}{P(f_i|\bar{w})} + \sum_{i=p+1}^{n} \log \frac{P(f_i|w)}{P(f_i|\bar{w})}
\]

Let $C = \log \frac{P(w)}{P(w)} + \sum_{i=1}^{n} \log \frac{P(f_i|w)}{P(f_i|\bar{w})}$ (common to all states)

\[
\log \frac{P(w|f)}{P(w)} = C + \sum_{i=1}^{p} \left( \log \frac{P(f_i|w)}{P(f_i|\bar{w})} - \log \frac{P(f_i|w)}{P(f_i|\bar{w})} \right)
\]

● Find best after-state by looking at only positive features!
Exploiting Relational Structure

- **Example:** Predicting feature odds given nearby features...

- **Idea:** Locally weighted regression in $n$-D feature attr. space
  - Take Euclidean metric of user-defined attribute distances
  - Compute odds of target feat. as weighted combination of “nearby” feats.

- **Advantages:** Generalization, reduced storage, fast lookup
Training Example

- On each trial, apply policy \( \pi \) for current value function:

<table>
<thead>
<tr>
<th>Before Move</th>
<th>State Description</th>
<th>After Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 1, Turn X</td>
<td>{ }</td>
<td>X</td>
</tr>
<tr>
<td>State 2, Turn O</td>
<td>{At(X,2,2)}</td>
<td>X</td>
</tr>
<tr>
<td>State 6, Turn O</td>
<td>{At(X,2,2), At(O,1,3), At(X,3,3)}, {At(O,1,1), At(X,1,2)}</td>
<td>X</td>
</tr>
<tr>
<td>State 7, Turn X</td>
<td>{At(X,2,2), At(O,1,3), At(X,3,3)}, {At(O,1,1), At(X,1,2), At(O,3,1)}</td>
<td>X</td>
</tr>
</tbody>
</table>

Outcome: X Wins, O Loses

- End of trial: Update win/loss counts for \( P(W) \), \( P(F_i|W) \) CPTs
Learning Structure

- Linear expressiveness of naïve Bayes often inadequate
- Join nodes to learn nonlinear structure, e.g.,

\[ \Delta l^*(\theta|D) = C + M \cdot I(F_a, F_b|W) \] (\(M\) is number of samples)

- But for \(n\) features, must keep track of \(O(n^2)\) calculations

- Instead, use Apriori to mine features w/ freq. > threshold
  - Efficient; maximizes \(\sim\) VOI; frequent joint features \(\Rightarrow\) low variance
Empirical Results

- **Evaluation of Described RRL Approach:**
  - **Domains:** TicTacToe (18 gf), Othello (13,200), Backgammon (786,816)
  - **Opponent:** TicTacToe (opt.), Othello (interm.), Backgammon (pubeval)
  - **Structure Learning:** None; Apriori w/ 2 freq. thresh. → cap at 2000
  - **Training:** 5000 games vs. opp. in < 20 min, < 3Mb on 1 GhZ PIII

<table>
<thead>
<tr>
<th>Structure Learning</th>
<th>Win/Draw %</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>28.3 %</td>
<td>Tic-Tac-Toe</td>
</tr>
<tr>
<td>Apriori (Freq=1)</td>
<td>100.0 %</td>
<td></td>
</tr>
<tr>
<td>Apriori (Freq=50)</td>
<td>45.8 %</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>61.3 %</td>
<td>Othello</td>
</tr>
<tr>
<td>Apriori (Freq=1)</td>
<td>49.4 %</td>
<td></td>
</tr>
<tr>
<td>Apriori (Freq=50)</td>
<td>99.1 %</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>46.5 %</td>
<td>Backgammon</td>
</tr>
<tr>
<td>Apriori (Freq=1)</td>
<td>45.4 %</td>
<td></td>
</tr>
<tr>
<td>Apriori (Freq=50)</td>
<td>51.5 %</td>
<td></td>
</tr>
</tbody>
</table>
Future Work I

- **Better feature discovery:**
  - Directly mine frequent and informative features (e.g., LargeBayes)

- **Avoiding local minima:**
  - Only exploration due to “optimistic” priors, better explore/exploit?
  - Policy constantly changing $\Rightarrow$ value shift; use param decay?
  - Switch to a more direct policy gradient method?

- **POMDPs/PSRs:**
  - Relational representation often an abstraction $\Rightarrow$ state aliasing
  - Features may just be observations on actual state!
  - Optimal evaluation may require representation of history or future
Future Work II

- **Relational Bayes net structure learning:**
  - Probabilistic Relational Models: Retain efficient policy evaluation?

- **First-order feature discovery:**
  - Nodes can be general first-order formulae:

  \[ \exists p \exists x \forall y \, \text{At}(p,x,y) \land \text{At}(p,x,1) \]

\[ \forall p \exists x \forall y \, \text{At}(p,x,y) \lor (\text{At}(p,y,x) \land \text{At}(p,1,1)) \]

- How to generate structure: (n)FOIL? What about feature overlap?
- MRF or Factor Graph? How to est. parameters efficiently? \( \triangle \)-rule?