Value Function Representation: Rete for Reinforcement Learning

Mitchell Keith Bloch

University of Michigan
2260 Hayward Street
Ann Arbor, MI. 48109-2121
bazald@umich.edu

June 19, 2014
Primary objective is to learn how to act, or to derive an optimal policy.

Prefer actions that lead to positive (or large) rewards to actions that lead to negative (or small) rewards.

Outcomes are characterized as a discounted return, \( \sum_{t=0}^{\infty} \gamma^t r_t \).

Deriving good estimates of these returns for different actions is essential for many RL algorithms.

See Sutton and Barto (1998) for an excellent primer.
Temporal Difference Method: Q-Learning

Given

- a discount rate, $\gamma$
- a Q-function, $Q(s, a)$, to represent value estimates for state-action pairs, and
- an immediate reward, $r$,

the update rule is expressed:

$$Q(s, a) \leftarrow r + \gamma \max_{a^*} Q(s', a^*)$$

- Conditions on RL-rules encode which features to test and how to discretize continuous state, defining the mapping $S \times A \Rightarrow Q$
- The presence of multiple RL-rules/weights for an operator results in linear function approximation
Research Goal

Efficient feature selection for relational reinforcement learning domains
Research Goal

Efficient feature selection for relational reinforcement learning domains

- Given a description of the environment, which descriptors are most essential?

**Descriptors:**

- **Blocks World:** \(^{\text{in-place}} \{<\text{block}> \text{ false}\}\)
- **Puddle World:** \(^{\text{x}} \{<\text{x}> 0.23235\}\)
- **Tetris:** \(^{\text{type}} \{<\text{type}> \text{ line}\}\)
Research Goal

Efficient feature selection for relational reinforcement learning domains

- Given a description of the environment, which descriptors are most essential?
- **Relational RL:** \((<s> ^{\text{block}} @B)\)
  \((<b1> ^{\text{in-place}} \text{false})\)
  \((<b1> ^{\text{on-top}} <b2>)\)

### Blocks World:

1. Given: A, B, C
2. Goal: A, B, C
Infinite Mario

Features:

- button-dpad [released/down/left/right]
- button-jump [up/down]
- button-speed [up/down]
- distance-to-right-pit $d$
- is-above-pit [true/false]
- is-in-pit [true/false]
- obstacle-right [true/false]

http://julian.togelius.com/mariocompetition2009/
Carli ≠ Soar – https://github.com/bazald/carli

What’s offered:

- A Soar-like execution cycle
Carli ≠ Soar – https://github.com/bazald/carli

What’s offered:

- A Soar-like execution cycle

Meaning:

1. \(^{\text{io.input-link}}\)
2. elaboration cycle
3. numeric preferences (and implicit operator proposal)
4. decide
   - impasses
5. act
What’s offered:
- A Soar-like execution cycle
- Soar-RL-like reinforcement learning support
- Architectural support for efficiently creating more specific RL-rules over time – a generative model for a value function

What’s missing or different:
- Manipulating WMEs from the RHS has not been tested yet
- Operators (as you know them) and impasses do not exist
- SMem, EpMem, and SVS do not exist
Carli ≠ Soar – https://github.com/bazald/carli

What’s offered:
- A Soar-like execution cycle
- Soar-RL-like reinforcement learning support
- Architectural support for efficiently creating more specific RL-rules over time – a generative model for a value function

What’s missing or different:
- Manipulating WMEs from the RHS has not been tested yet
- Operators (as you know them) and impasses do not exist
- SMem, EpMem, and SVS do not exist
What does an architecture need to support efficiently creating more specific RL-rules over time?
Architectural Support?

What does an architecture need to support efficiently creating more specific RL-rules over time?

- A fringe of possible more-specific RL-rules, each adding one condition
Fringe RL-Rules (A Possible Syntax)

sp {rl-rule-1
  "A general RL-rule"
  # No flags
  (<s> ^operator <o> +)
  (<o> ...)  
  # No other conditions
  # This is 100% general
  -- >
  ( <s> ^operator <o> = 0)
}

gp {rl-rule-1f1
  "A fringe RL-rule"
  :fringe
  (<s> ^operator <o> +)
  (<o> ...  
     ^attr-a [value-a1
     value-a2])
  -- >
  ( <s> ^operator <o> = 0)
}

Next we’ll refine the value function

Assume rl-rule-1f2,
rl-rule-1f3,
::
What does \texttt{fringe} accomplish?

- Informs the system that it suggests a new condition
- Indicates that it should not contribute to value function
- Allows gathering of metrics about the new condition
  - Q-value
  - Update count
  - Firing count
Creating More Specific RL-Rules Over Time

Assume a refinement procedure just ran, choosing rl-rule-1f1**

\[
\text{gp \{ rl-rule-2 }
\]

"A bit more specific"

# No flags

\(<s> \ ^{\text{operator}} <o> + \)
\(<o> \ ...
\)

\(^{\text{attr-a}} [ \text{value-a1} \text{value-a2}]\)

# One new condition

# But only one

\(-- >
\)

\(<s> \ ^{\text{operator}} <o> = 0 \)
\}

\[
\text{gp \{ rl-rule-2f1 }
\]

"New fringe RL-rule"

:fringe

\(<s> \ ^{\text{operator}} <o> + \)
\(<o> \ ...
\)

\(^{\text{attr-a}} [ \text{value-a1} \text{value-a2}]\)

\(^{\text{attr-b}} [ \text{value-b1} \text{value-b2}]\)

\(-- >
\)

\(<s> \ ^{\text{operator}} <o> = 0 \)
\}
Enumerable and Ranged Conditions

We are not restricted to enumerable conditions

```
gp { rl-rule-2f1
  "For Blocks World"
  :fringe
  (<s> ^operator <o> +)
  (<o> ^block <b>
      ^dest <d>)
  (<b> ^in-place [true
      false])
  -- >
  (<s> ^operator <o> = 0)
}
```

```
gp { rl-rule-2f1
  "For Puddle World"
  :fringe
  (<s> ^operator <o> +)
  (<o> ^x <x>
      ^y <y>)
  (<o> ^x <x>
      {[< >]= 0.5})
  -- >
  (<s> ^operator <o> = 0)
}
```

Note the deliberate ordering
Nodes in the first row (the $\alpha$ network) match on WMEs
Lower nodes (the $\beta$ network) join tokens and/or perform tests
Actions simply add Q-values/weights to operators and decide whether to modify the value function representation
Within any box, all weights but one represent a fringe
Later Retes for Blocks World

- Fringe weights/actions are removed
- Predicate tests move lower in the rete
- Other tests are joined from one layer to the next
- The number of fringe nodes is generally reduced over time
  - Numeric predicates may or may not be “infinitely” divisible
  - Not sure how, syntactically, to describe this to the architecture
Ongoing Research

Architectural Support? – The Bigger Picture

What does an architecture need to support efficiently creating more specific RL-rules over time?

- A fringe of possible more-specific RL-rules, each adding one condition
- A method for determining, with some confidence, which fringe RL-rules to promote to actual RL-rules
- Reverse methods to allow for corrections
What does an architecture need to support efficiently creating more specific RL-rules over time?

- A fringe of possible more-specific RL-rules, each adding one condition
- A method for determining, with some confidence, which fringe RL-rules to promote to actual RL-rules
- Reverse methods to allow for corrections
What does an architecture need to support efficiently creating more specific RL-rules over time?

- A fringe of possible more-specific RL-rules, each adding one condition
- A method for determining, with some confidence, which fringe RL-rules to promote to actual RL-rules
- Reverse methods to allow for corrections
What does an architecture need to support efficiently creating more specific RL-rules over time?

- A fringe of possible more-specific RL-rules, each adding one condition
- A method for determining, with some confidence, which fringe RL-rules to promote to actual RL-rules
- Reverse methods to allow for corrections

This is my current research
Refinement Criteria

When and how to best refine the value function?

- Cumulative absolute temporal difference error
  - Focus on regions of high activity and error
  - Seems to work better for Blocks World than the value criterion
- Value criterion (Whiteson, 2007)
  - Focus on improving value estimates
  - Early implementation
  - Already works better for Infinite Mario
- Policy criterion (Whiteson, 2007)
  - Focus on modifying policy
  - Coming soon
Nuggets:
- Another rete implementation taking advantage of C++11 features
- Progress on the development of a generative model for a value function
- It appears to be implementable as an extension to Soar-RL

Coal:
- As part of Soar-RL, it would involve excising rules over time, which does not appear to be common practice
- Some syntax details to work out before implementing in Soar-RL
- Good, general criteria for deciding when to refine/collapse/... the value function are not yet settled