Performance and Population State Metrics for Rule-Based Learning Systems

Tim Kovacs
Department of Computer Science
The University of Bristol
kovacs@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~kovacs
Outline

This work:

- Proposes a distinction between performance and population state metrics.
- Demonstrates the superiority of the latter in some cases.
- Proposes some new population state metrics.
- Uses them to study the pressure against overlapping rules in XCS.
Learning Classifier Systems (LCS)

- Incremental machine learning systems suitable for both Supervised Learning and Reinforcement Learning.

- Use a population of if-then rules (called classifiers) to make decisions.

- Use a Genetic Algorithm (GA) to evolve the rules.
Basic LCS Operation

Trials consist of:

1. Input to the LCS.
2. Find matching rules.
3. Conflict resolution.
4. Output action.
5. Receive feedback on action.
6. Update rule utility estimates.
7. Occasionally trigger GA to update rule population.
Performance: Proportion of last 50 actions which were correct.
Population size: Number of macroclassifiers in the population.

[Curves are averages of 10 runs with XCS on the 6 multiplexer.]
Measuring Performance and Measuring State

Performance metrics:

- Directly measure performance.

- Indirectly measure the state of the population.

Population state metrics:

- Indirectly measure performance.

- Directly measure the state of the population.
Representing Solutions

Normally a set of rules is required to represent the solution.

What properties should this set have?

**Completeness:** All inputs are matched by some rule.

**Correctness:** The rules actually represent the target function.

**Minimality:** The solution consists of the minimal number of rules needed to represent it completely and accurately.

The XCS classifier system tends towards a 4th property:

**Irredundancy:** No input is matched by more than one rule.
In XCS terminology:

- We call a solution with all 4 properties an *optimal solution* \([O]\).
- We denote by \%[O] the percentage of \([O]\) in the population.
- \%[O] is a population state metric:
  - To determine \%[O] we make a pairwise comparison of two sets of rules: \([O]\) and the population.
The 6 Multiplexor

- A popular benchmark for LCS. In Disjunctive Normal Form:

\[ \overline{x_0x_1x_2} \lor \overline{x_0x_1x_3} \lor x_0\overline{x_1x_4} \lor x_0x_1x_5 \]

- The task is to approximate the function from input/output examples.

- [O]: The 8 conditions below each map to 2 actions, producing 16 rules.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>2.</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>3.</td>
<td>0</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>4.</td>
<td>0</td>
<td>1</td>
<td>#</td>
<td>1</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>5.</td>
<td>1</td>
<td>0</td>
<td>#</td>
<td>#</td>
<td>0</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>6.</td>
<td>1</td>
<td>0</td>
<td>#</td>
<td>#</td>
<td>1</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>7.</td>
<td>1</td>
<td>1</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>0</td>
<td>#</td>
</tr>
<tr>
<td>8.</td>
<td>1</td>
<td>1</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>1</td>
<td>#</td>
</tr>
</tbody>
</table>
Using $\%[O]$ to Evaluate Deletion Schemes

Note:

- Performance is similar.
- Population size is better for $t_2$.
- ...but $t_2$'s $\%[O]$ is much worse!

[Curves are averages of 10 runs with XCS on the 6 multiplexer. Population size limit = 400.]
Measuring XCS's Bias Against Overlapping Rules

XCS prefers irredundant solutions: it is biased against overlapping rules.

It employs a niche GA: only rules which match the current input can be selected.

- More general rules match more inputs.
  - And so get an implicit fitness bonus.

- But rules which match the same inputs (overlap) compete against each other.
  - And so get an implicit fitness penalty.
Prime Implicants

We define \([\Pi]\) as the set of rules whose conditions are Prime Implicants for the target function.

I.e. accurate rules which cannot be generalised and yet remain accurate.

These are the most general, accurate rules possible.

- ...so they should have high fitness!

\([\Pi]\) for the 6 multiplexer consists of 36 rules:

- \([O]\) is a 16-rule subset of \([\Pi]\).
XCS’s [%[PI]] on the 6 Multiplexer

[Curves are averages of 10 runs with XCS on the 6 multiplexer. Population size limits of 400, 800 and 1600 were used.]
Why didn’t XCS find all of [PI]?

- Even with a population size limit of 1600 rules, XCS does not find all 36 rules in [PI].

- But we saw earlier that with just 400 rules it found the 16-rule subset [O].

Why did it not find the other 20?

- [O] is irredundant.

- [PI] is not.

Conclusion: XCS’s bias against overlapping rules is very strong.
Conclusions

Re: Metrics

- Performance metrics tell us *how well* the system is doing.
- Population state metrics tell us *what* the system is doing.

Re: XCS

- XCS is very biased against overlapping rules.