Strength or Accuracy?

Tim Kovacs
School of Computer Science
The University of Birmingham
T.Kovacs@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~tyk
Introduction

Goal: to compare two broad approaches to fitness calculation

- Strength-based (traditional, many forms)
  - Strength is the primary factor in fitness

- Accuracy-based (only XCS, Frey & Slate's letter recognizer)
  - Accuracy has a veto

Pure accuracy and pure strength were compared, but results should extend to more general case.

A small change in the fitness calculation can have many consequences, e.g. for exploration control & form of representation.
• We can think of strength as prediction of reward . . .
• . . . and accuracy as consistency of a rule’s strength over time

- Rule 1: low accuracy = low fitness
- Rule 2: high accuracy = high fitness

• With strength-based fitness the magnitude of reward influences fitness, but with pure accuracy-based fitness it does not.
1. **Strength (prediction):** $s_j \leftarrow s_j + \beta (R - s_j)$

where $0 < \beta \leq 1$ controls the learning rate and $R$ is the reward from the environment, and the update is for single step tasks.

2. **Prediction error:** $\varepsilon_j \leftarrow \varepsilon_j + \beta \left( \frac{|R - s_j|}{R_{\max} - R_{\min}} - \varepsilon_j \right)$

where $R_{\max}$ and $R_{\min}$ are the highest and lowest rewards possible in any state.

3. **Accuracy:** $\kappa_j = \begin{cases} 
1 & \text{if } \varepsilon_j \leq \varepsilon_o \\
 e^{(\ln \alpha)(\varepsilon_j - \varepsilon_o)/\varepsilon_o} \times 0.1 & \text{otherwise} 
\end{cases}$

where $\varepsilon_o$ controls the tolerance for prediction error and $\alpha < 1$ controls the rate of decline in accuracy when $\varepsilon_o$ is exceeded.
4. Relative accuracy: \( \kappa'_j = \frac{\kappa_j}{\sum_{x \in [A]} \kappa_x} \)

5. Fitness: \( F_j \leftarrow F_j + \beta(\kappa'_j - F_j) \)
Problems with strength-based LCS

- Overgeneral rules

- Rule allocation bias (esp. greedy classifier creation) → gaps in covering map

- Interaction: strong overgenerals
Strong overgenerals

- An overgeneral which is stronger than a correct accurate competitor
  - They influence action selection
  - They reproduce
  - They cannot be removed by deleting low strength rules
Unbiased Function – Rule Strengths

Fitness = Strength

Fitness based on Accuracy

Correct (A & D)

Incorrect (B & C)

Overgeneral (E & F)
Biased Function – Rule Strengths

Fitness = Strength

Fitness based on Accuracy

Correct (A)
Overgeneral (E)
Correct (D)
Overgeneral (F)
Incorrect (B & C)
All Accurate Rules (A,B,C,D)
Overgenerals (E & F)
Single step problems

- Strong overgeneralizations are possible in strength LCS when the reward function is sufficiently biased.

- Since we define the reward function, why bias it?
  - To focus allocation of rules to more important parts of the problem.

- Pure accuracy XCS is not influenced by reward function bias, but we could add a bias to the fitness function.
Multi step problems

- We define reward function, but now strengths are updated towards Q-values not rewards (Q-learning update vs. delta rule).

- In non-trivial multi step problems the Q-function is always sufficiently biased to produce strong overgenerals.
  - So pure strength seems unsuited to multi step problems.

- In discounted multi step problems rule allocation bias is towards the source of reward. I don’t know if this is appropriate.
Complete maps and best action maps

- Accuracy maintains complete maps.

- Strength *tends towards* best action only maps – and so (potentially) smaller populations.

- But without a complete map, how do we know what we’ve tried and what we haven’t?

- Adding an exploration bonus to the Q-values may help, but may not make best action maps completely safe.
Complete Map

Best Action Only Map
Complete Map  
Best Action Only Map  
... more of the environment
Summary

• Strength cannot get rid of strong overgenerals and cannot avoid them in multi step tasks – so it seems unsuited to them.

• In moderation, a rule allocation bias should be useful. But if it is too strong we can get:
  – partial maps, which make exploration control awkward.
  – gaps in the covering map.

• We don’t need to base fitness on strength to bias rule allocation.

• LCS search for accurate generalisations. To get them, use a fitness metric which insists that rules be accurate.