Introduction to Learning Classifier Systems (mostly XCS)

Stewart W. Wilson
Prediction Dynamics
On the original classifier system...


- Lashon Booker
- Larry Bull
- Stephanie Forrest
- John Holmes
- Tim Kovacs
- Rick Riolo
- Robert Smith
- Stewart Wilson
- Many others
What is it?

- Learning machine (program).
- Minimum *a priori*.
- “On-line”.
- Capture regularities in environment.
To get reinforcements ("rewards", "payoffs")

(Not “supervised” learning—no prescriptive teacher.)
Inputs:
Now binary, e.g., 100101110
—like thresholded sensor values.
Later continuous, e.g., <43.0 92.1 7.4 ... 0.32>

Outputs:
Now discrete decisions or actions,
e.g., 1 or 0 (“yes” or “no”),
“forward”, “back”, “left”, “right”
Later continuous, e.g., “head 34 degrees left”
XCS contains rules (called *classifiers*), some of which will match the current input. An action is chosen based on the predicted payoffs of the matching rules.

\[ \text{<condition>} : \text{<action>} \Rightarrow \text{<prediction>} . \]

Example: 01#1## : 1 \Rightarrow 943.2

Note this rule matches more than one input string:

- 010100
- 010110
- 010101
- 011111
- 011100
- 011101
- 011110
- 011111.

This adaptive “rule-based” system contrasts with “PDP” systems such as NNs in which knowledge is distributed.
For each action in [M], classifier predictions $p$ are weighted by fitnesses $F$ to get system’s net prediction in the prediction array.

Based on the system predictions, an action is chosen and sent to the environment.

Some reward value is returned.
How do rules acquire their predictions?

1. By “updating” the current estimate.
   For each classifier $C_j$ in the current $[A]$, 
   
   $$ p_j \leftarrow p_j + \alpha (R - p_j), $$
   
   where $R$ is the current reward and $\alpha$ is the learning rate.

   This results in $p_j$ being a “recency weighted” average of previous reward values:
   
   $$ p_j(t) = \alpha R(t) + \alpha (1-\alpha)R(t-1) + \alpha (1-\alpha)^2 R(t-2) + \ldots + (1-\alpha)^t p_j(0). $$

2. And by trying different actions, according to an explore/exploit regime.

   A typical regime chooses a random action with probability 0.5.

   Exploration (e.g., random choice) is necessary in order to learn anything. But exploitation—picking the highest-prediction action is necessary in order to make best use of what is learned.

   There are many possible explore/exploit regimes, including gradual changeover from mostly explore to mostly exploit.
• Usually, the “population” [P] is initially empty. (It can also have random rules, or be seeded.)

• The first few rules come from “covering”: if no existing rule matches the input, a rule is created to match, something like imprinting.

Input: 11000101

Created rule: 1##0010# : 3 => 10
Random #’s and action, low initial prediction.

• But primarily, new rules are derived from existing rules.
• Besides its prediction $p_j$, each classifier’s error and fitness are regularly updated.

Error:

$$\varepsilon_j \leftarrow \varepsilon_j + \alpha(|R - p_j| - \varepsilon_j).$$

Accuracy:

$$\kappa_j \equiv \varepsilon_j^{-n} \text{ if } \varepsilon_j > \varepsilon_0, \text{ otherwise } \varepsilon_0^{-n}$$

Relative accuracy:

$$\kappa_j' \equiv \frac{\kappa_j}{\left(\sum_i \kappa_i\right)}, \text{ over } [A].$$

Fitness:

$$F_j \leftarrow F_j + \alpha(\kappa_j' - F_j).$$

• Periodically, a genetic algorithm (GA) takes place in [A].

Two classifiers $C_i$ and $C_j$ are selected with probability proportional to fitness. They are copied to form $C_i'$ and $C_j'$.

With probability $\chi$, $C_i'$ and $C_j'$ are crossed to form $C_i''$ and $C_j''$, e.g.,

$$1 0 # # 1 1 : 1 \implies 1 0 # # 1 # : 1$$
$$# 0 0 0 1 # : 1 \implies # 0 0 0 1 1 : 1$$

$C_i''$ and $C_j''$ (or $C_i'$ and $C_j'$ if no crossover occurred), possibly mutated, are added to [P].
Can I see the overall process?

Environment

0011

Detectors

$[P]$ match

$\begin{align*}
#011:01 & \quad 43.01 \quad 99 \\
11##:00 & \quad 32.13 \quad 9 \\
#0##:11 & \quad 14.05 \quad 52 \\
001#:01 & \quad 27.24 \quad 3 \\
#01:11 & \quad 18.02 \quad 92 \\
1#01:10 & \quad 24.17 \quad 15 \\
\text{...etc.}
\end{align*}$

Effectors

$\text{"left"}$

01

Reward

Match Set

$[M]$

Prediction Array

$\begin{align*}
\text{action} & \quad \text{selection} \\
nil & \quad 42.5 & \quad \text{nil} & \quad 16.6
\end{align*}$

Action Set

$[A]$

$\begin{align*}
#011:01 & \quad 43.01 \quad 99 \\
001#:01 & \quad 27.24 \quad 3 \\
\end{align*}$

GA

Update:

predictions, errors, fitnesses

$01$

$\text{GA(cover)}$

Can I see the overall process?
They remain in [P], in competition with their offspring.

But two classifiers are *deleted* from [P] in order to maintain a constant population size.

Deletion is probabilistic, with probability proportional to, e.g.:

- A classifier’s average action set size $a_j$—estimated and updated like the other classifier statistics.

- $a_j/F_j$, if the classifier has been updated enough times, otherwise $a_j/F_{ave}$, where $F_{ave}$ is the mean fitness in [P].

—And other arrangements, all with the aim of balancing resources (classifiers) devoted to each niche ([A]), but also eliminating low fitness classifiers rapidly.
Basic example for illustration: Boolean 6-multiplexer.

$$F_6 = x_0'x_1'x_2 + x_0'x_1x_3 + x_0x_1'x_4 + x_0x_1x_5$$

$$l = k + 2^k \quad k > 0$$

$$F_{20} = x_0'x_1'x_2'x_3'x_4 + x_0'x_1'x_2x_3x_5 + x_0'x_1'x_2x_3'x_6 + x_0'x_1x_2'x_3x_7 + x_0'x_1x_2'x_3'x_8 + x_0'x_1x_2'x_3x_9 + x_0'x_1x_2'x_3x_10 + x_0'x_1x_2'x_3x_11 + x_0x_1'x_2'x_3'x_12 + x_0x_1'x_2'x_3x_13 + x_0x_1'x_2'x_3'x_14 + x_0x_1'x_2x_3x_15 + x_0x_1'x_2'x_3'x_16 + x_0x_1'x_2'x_3x_17 + x_0x_1'x_2'x_3'x_18 + x_0x_1'x_2'x_3x_19$$

$$01100010100100001000 \rightarrow 0$$
What are the results like?— 2
Population at 5,000 problems in descending order of numerosity (first 40 of 77 shown).

<table>
<thead>
<tr>
<th>PRED</th>
<th>ERR</th>
<th>FITN</th>
<th>NUM</th>
<th>GEN</th>
<th>ASIZ</th>
<th>EXPER</th>
<th>TST</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>.00</td>
<td>884</td>
<td>30</td>
<td>.50</td>
</tr>
<tr>
<td>1.</td>
<td>00</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>0.</td>
<td>819</td>
<td>24</td>
</tr>
<tr>
<td>2.</td>
<td>01</td>
<td>1</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>856</td>
</tr>
<tr>
<td>3.</td>
<td>01</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>0.</td>
<td>840</td>
<td>20</td>
</tr>
<tr>
<td>4.</td>
<td>11</td>
<td>#</td>
<td>1</td>
<td>0</td>
<td>0.</td>
<td>719</td>
<td>20</td>
</tr>
<tr>
<td>5.</td>
<td>00</td>
<td>1</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>698</td>
</tr>
<tr>
<td>6.</td>
<td>01</td>
<td>0</td>
<td>#</td>
<td>0</td>
<td>0.</td>
<td>664</td>
<td>18</td>
</tr>
<tr>
<td>7.</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>712</td>
<td>18</td>
</tr>
<tr>
<td>8.</td>
<td>00</td>
<td>0</td>
<td>#</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>674</td>
</tr>
<tr>
<td>9.</td>
<td>10</td>
<td>0</td>
<td>#</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>706</td>
</tr>
<tr>
<td>10.</td>
<td>11</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>539</td>
</tr>
<tr>
<td>11.</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>638</td>
<td>16</td>
</tr>
<tr>
<td>12.</td>
<td>01</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>0.</td>
<td>522</td>
<td>15</td>
</tr>
<tr>
<td>13.</td>
<td>00</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>0.</td>
<td>545</td>
<td>14</td>
</tr>
<tr>
<td>14.</td>
<td>10</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>0.</td>
<td>425</td>
<td>12</td>
</tr>
<tr>
<td>15.</td>
<td>11</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>458</td>
<td>11</td>
</tr>
<tr>
<td>16.</td>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>233</td>
<td>6</td>
</tr>
<tr>
<td>17.</td>
<td>#0</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>0.</td>
<td>210</td>
<td>6</td>
</tr>
<tr>
<td>18.</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>187</td>
<td>5</td>
</tr>
<tr>
<td>19.</td>
<td>01</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0.</td>
<td>168</td>
<td>4</td>
</tr>
<tr>
<td>20.</td>
<td>11</td>
<td>#1</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>114</td>
<td>4</td>
</tr>
<tr>
<td>21.</td>
<td>10</td>
<td>#1</td>
<td>0</td>
<td>0</td>
<td>0.</td>
<td>152</td>
<td>4</td>
</tr>
<tr>
<td>22.</td>
<td>10</td>
<td>#1</td>
<td>0</td>
<td>1</td>
<td>0.</td>
<td>131</td>
<td>3</td>
</tr>
<tr>
<td>23.</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>117</td>
</tr>
<tr>
<td>24.</td>
<td>11</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>0.</td>
<td>68</td>
<td>3</td>
</tr>
<tr>
<td>25.</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>46</td>
<td>3</td>
</tr>
<tr>
<td>26.</td>
<td>10</td>
<td>#1</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>81</td>
<td>3</td>
</tr>
<tr>
<td>27.</td>
<td>#1</td>
<td>0</td>
<td>#</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>86</td>
</tr>
<tr>
<td>28.</td>
<td>01</td>
<td>10</td>
<td>#</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>61</td>
</tr>
<tr>
<td>29.</td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>58</td>
<td>2</td>
</tr>
<tr>
<td>30.</td>
<td>10</td>
<td>0</td>
<td>#</td>
<td>0</td>
<td>1</td>
<td>0.</td>
<td>63</td>
</tr>
<tr>
<td>31.</td>
<td>11</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>63</td>
</tr>
<tr>
<td>32.</td>
<td>00</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>77</td>
</tr>
<tr>
<td>33.</td>
<td>10</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>1</td>
<td>0.</td>
<td>93</td>
</tr>
<tr>
<td>34.</td>
<td>11</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>59</td>
<td>2</td>
</tr>
<tr>
<td>35.</td>
<td>01</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>1000</td>
<td>.00</td>
<td>75</td>
</tr>
<tr>
<td>36.</td>
<td>01</td>
<td>0</td>
<td>#</td>
<td>1</td>
<td>0</td>
<td>.00</td>
<td>856</td>
</tr>
<tr>
<td>37.</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.</td>
<td>92</td>
<td>2</td>
</tr>
<tr>
<td>38.</td>
<td>10</td>
<td>#</td>
<td>1</td>
<td>1000</td>
<td>.00</td>
<td>703</td>
<td>.31</td>
</tr>
<tr>
<td>39.</td>
<td>#1</td>
<td>1</td>
<td>#</td>
<td>0</td>
<td>0</td>
<td>856</td>
<td>.22</td>
</tr>
</tbody>
</table>

What are the results like? — 3
Action sets \([A]\) for input 101001 and action 0 at several epochs.

Can you show the evolution of a rule?
Consider two classifiers C1 and C2 having the same action, and let C2 be a generalization of C1. That is, C2 can be obtained from C1 by changing some non-# alleles in the condition to #’s. Suppose that C1 and C2 are equally accurate. They will therefore have the same fitness. However, note that, since it is more general, C2 will occur in more action sets than C1. What does this mean? Since the GA acts in the action sets, C2 will have more reproductive opportunities than C1. This edge in reproductive opportunities will cause C2 to gradually drive C1 out of the population.

Example:

\[
\begin{array}{ccc}
 p & \varepsilon & F \\
 C1: & 1 0 # 0 0 1 : 0 & \Rightarrow & 1000 & .001 & 920 \\
 C2: & 1 0 # # 0 # : 0 & \Rightarrow & 1000 & .001 & 920 \\
\end{array}
\]

C2 has equal fitness but more reproductive opportunities than C1.

C2 will “drive out” C1.
Does XCS scale up?

![Graphs showing performance, error, and population size for XCS on different problem sizes](image-url)
What about complexity?

20m \(\sim\) 5x harder than 11m
11m \(\sim\) 5x harder than 6m.

\( \Rightarrow \ D = cg^p, \)

where \( D = \) “difficulty”, here learning time,
\( g = \) number of maximal generalizations,
\( p = \) a power, about 2.3
\( c = \) a constant about 3.2

Thus “\( D \) is polynomial in \( g \)”.

What is \( D \) with respect to \( l \), string length?

For the multiplexers, \( l = k + 2^k \),
or \( l \rightarrow 2^k \) for large \( k \).

But \( g = 4 \cdot 2^k \), thus \( l \sim g \),
So that “\( D \) is polynomial in \( l \)” (not exponential).
XCS

What about deferred reward?

Apply ideas from multi-step reinforcement learning.

Need the *action-value* of each action in each state.

What is the action-value of a state more than one step from reward?

Intuitive sketch:

\[ p_j \leftarrow p_j + \alpha \left[ (r_{\text{imm}} + \gamma \max_{a' \in A} P(x',a')) - p_j \right] \]

where \( p_j \) is the prediction of a classifier in the current action set \([A]\),
\( x' \) and \( a' \) are the next state and possible actions,
\( P(x',a') \) is a system prediction at the next state,
and \( r_{\text{imm}} \) is the current external reward.
• Previous action set $[A]_{-1}$ is saved and updates are done there, using the current prediction array for “next state” system predictions.

• On the last step of a problem, updates occur in $[A]$. 

Can I see the overall process?
What are the results like?— 1

• Animat senses the 8 adjacent cells.
  
  F b b
  O * b
  Q b b

• Coding of each object:
  
  F = 110 “food1”
  G = 111 “food2”
  O = 010 “rock1”
  Q = 011 “rock2”
  b = 000 “blank”

• “Sense vector” for above situation: 00000000000000011010110

• A matching classifier: ####0#00####00001##101## : 7
Two generalizations discovered by XCS in Woods1.

(Food = 11 = “Tasty”, “Opaque”
Rock = 01 = “Bland”, “Opaque”
Blank = 00 = “Bland”, “Clear”)
What about non-binary inputs?

Each input variable $x_i$ may be real or integer valued.

Classifier: $<(x_{i1}, x_{1u}) \ldots (x_{ni}, x_{nu})>: <\text{action}> \Rightarrow p$

- Condition consists of “interval predicates”
  \[ \text{int}_i = (x_{il}, x_{iu}). \]

- Classifier matches iff $x_{il} \leq x_i < x_{iu}, \ \forall i$.

- Crossover occurs between and within predicates.

- Mutation adds $\pm \text{rand}(m_1)$ to allele.
  $[m_1$ is real or integer as appropriate.$]$

- Covering creates a classifier with condition in which
  \[ x_{il} = \max[(x_i - \text{rand}(m_2)), x_{iMIN}] \]
  \[ x_{iu} = \min[(x_i + \text{rand}(m_2)), x_{iMAX}] \]
Example: Wisconsin Breast Cancer dataset

- Sample instances (699 in all).
  
  1070935,3,1,1,1,1,2,1,1,2  
  1071760,2,1,1,1,2,1,3,1,1,2  
  1072179,10,7,7,3,8,5,7,4,3,4  
  1074610,2,1,1,2,2,1,3,1,1,2  
  1075123,3,1,2,1,2,1,2,1,1,2  
  1079304,2,1,1,1,2,1,2,1,1,2  
  1080185,10,10,10,8,6,1,8,9,1,4  
  1081791,6,2,1,1,1,1,7,1,1,2  
  1084584,5,4,4,9,2,10,5,6,1,4  
  1091262,2,5,3,3,6,7,7,5,1,4  
  1096800,6,6,6,9,6,?,7,8,1,2  

- Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses.

- 458 Benign + 241 Malignant = 699 Cases.

- Stratified 10-fold cross-validation result:

<table>
<thead>
<tr>
<th>Correct</th>
<th>Incorrect</th>
<th>Not Matched</th>
<th>Fraction Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td>2</td>
<td>0</td>
<td>0.9714</td>
</tr>
<tr>
<td>69</td>
<td>1</td>
<td>0</td>
<td>0.9857</td>
</tr>
<tr>
<td>65</td>
<td>5</td>
<td>0</td>
<td>0.9286</td>
</tr>
<tr>
<td>66</td>
<td>4</td>
<td>0</td>
<td>0.9429</td>
</tr>
<tr>
<td>65</td>
<td>3</td>
<td>2</td>
<td>0.9286</td>
</tr>
<tr>
<td>64</td>
<td>3</td>
<td>3</td>
<td>0.9143</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
</tr>
<tr>
<td>69</td>
<td>1</td>
<td>0</td>
<td>0.9857</td>
</tr>
<tr>
<td>65</td>
<td>3</td>
<td>1</td>
<td>0.9420</td>
</tr>
<tr>
<td>67</td>
<td>2</td>
<td>1</td>
<td>0.9571</td>
</tr>
</tbody>
</table>

MEAN ⇒ 0.9556

- Performance similar to best other systems.
What about generalization?

Increasingly general, accurate classifiers were found by continuing the evolution.

If clump thickness is 7 or above and uniformity of cell size is 5 or above, malignancy is indicated.

If normal nucleoli is 10, then malignant.

If uniformity of cell shape is 8 or above and marginal adhesion is not 1, then malignant.

If uniformity of cell size is 1 and bare nuclei is 4 or less, then benign.
What if generalizations are not conjunctive?

“Standard” classifier condition is a conjunction of variable values or ranges:

\#10\#1\# or (3,7)(0,2) ... (4,9) etc.

What about “if $x > y$ for any $x$ and $y$, and action $a$ is taken, payoff is predicted to be $p$”?

Cannot be represented by a single conjunctive condition, since it’s a relation.

However, it can be represented using an **S-classifier**:

$$(x > y) : a \Rightarrow p$$

I.e., a classifier whose condition is a Lisp S-expression.

With appropriate elementary functions, S-classifiers can encode an almost unlimited variety of conditions.

They can be evolved using techniques drawn from genetic programming.
What about Non-Markov environments?

Example (McCallum’s Maze):

```
  T T T
  T
```

Arrows indicate aliased states—each has the same local view. The optimal action is not determinable from the sensory input.

Approaches:
- “History window”—remember previous inputs
- Search for correlation with past input events
- **Adaptive internal state**
Adaptive internal state?

<Environmental condition> <Internal condition>:
  <Internal action> <External action> ⇒ p

Example: ###1##0# # : 1 0 ⇒ 504

Internal action modifies an internal register \( R \).
Internal condition reads (must match) \( R \).
**Internal state** = current contents of \( R \).

For a 1-bit register:

If internal action = 1, set \( R \) to 1
  = 0, set \( R \) to 0
  = #, leave \( R \) unchanged

Will classifiers evolve that set and read \( R \) so as to distinguish aliased states and achieve high performance?
Woods101 (= McCallum’s Maze)

\[
\begin{array}{c c c c}
T & T & T & T \\
T & T & T & T \\
T & T & T & T \\
T & T & T & T \\
T & T & T & T \\
\end{array}
\]

\[
\begin{array}{c c c c c c c c}
T & T & T & T & T & T & T & T \\
T & T & T & T & T & T & T & T \\
T & T & T & T & T & T & T & T \\
T & T & T & T & T & T & T & T \\
T & T & T & T & T & T & T & T \\
\end{array}
\]

\[
\begin{array}{c c c c c c c c}
0 & 1000 & 2000 & 3000 & 4000 & 5000 & 6000 & 7000 & 8000 \\
NUMBER OF STEPS TO GOAL \\
\end{array}
\]

\[
\begin{array}{c c c c c c c c}
0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 \\
NUMBER OF PROBLEMS \\
\end{array}
\]

OPTIMAL PERFORMANCE
Woods101.5

Optimum reached with register redundancy (4 bits vs. 2).
Woods102

(a)

(b)

(c)

Uses 8-bit register.
• **Generalized classifier** (GCL) architecture:

\[ t(x) : r(a) \Rightarrow p(x, a) \]

"For \( x \) in the subdomain given by \( t(x) \) and \( a \) satisfying the action restriction \( r(a) \), the prediction is given by \( p(x, a) \)"

GCL opens way to **continuous** (non-discrete) actions and maybe to continuous time.

• **Anticipatory** classifier systems that predict the next state. Individual classifiers predict entire state, or individual classifiers predict state components.

• Continue Non-Markov work to create **Hierarchical** LCS with sub-behaviors selected and controlled by higher behaviors. Based on extensions of the register idea.
• Theory of XCS **learning complexity**. Time to performance, memory required. Hypothesis is that complexity is a low-order polynomial in target function complexity—in contrast to other learning methods.

• Improvements to XCS **mechanisms**. More sophisticated accuracy measures. Tournament selection. Long-path techniques.

• Comparison of XCS and **strength-based** (traditional) classifier systems. Does the traditional system have a niche? Where is accuracy-based weak?
How is XCS different from other RL systems?

- Rule-based, not connectionist or rbf-like
- Structure is created as needed
- Learning may often be faster because classifiers are inherently non-linear
- Learning complexity tractable
- Classifiers can keep and use statistics; difficult in a network
- User can ”see the knowledge”
- Hierarchy and reasoning may be easier, since knowledge is in the form of discrete rules
- Powerful generalization ability, if syntax suits the problem domain