Learning Classifier Systems

Dr. Jason Lines
j.lines@uea.ac.uk

UEA University of East Anglia

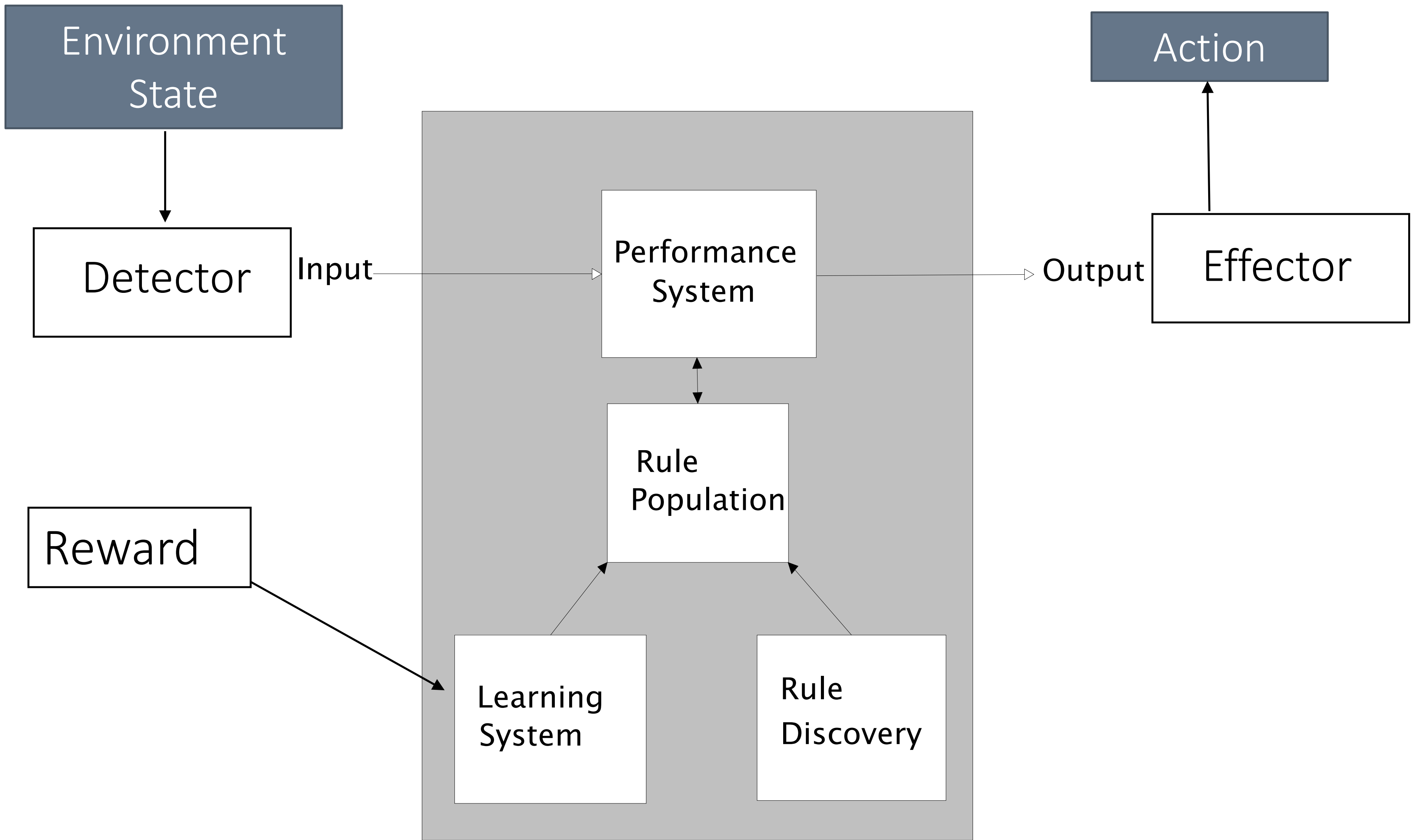
Learning Classifier Systems (LCS)

A Learning Classifier System is a rule based algorithm that can be used for machine learning and reinforcement learning. It is a type of Genetics Based Machine Learning

The basic components of an LCS are:

- A Rule Set: A set of rules (with associated parameters) that encapsulates the current state of the agent knowledge
- A Production System: A means of mapping the detector signal to the action space using the rule set
- A Learning System: A mechanism for altering the parameters of the rules to reflect performance
- A Rule Discovery System: An algorithm to alter the rule set

Learning Classifier Systems (LCS)



Rule Based Classifiers

Decision trees are an example of a type of a **rule based classifier**

They use a greedy algorithm to search for a **non-overlapping rule set**

The greedy solution to obtain a non-overlapping rule set may not always lead to the best possible solution however:

1. The greedy decision may not be the best because it ignores interactions between the attributes
2. The best rule set may contain overlapping rules

1. The greedy decision may not be the best

Greedy decision making ignores possible interactions between attributes

A1	A2	A3	C
1	1	0	0
1	1	0	0
1	0	1	1
1	0	1	1
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	1

A1=sunny (Yes/No)
A2=Hot (Yes/No)
A3 = Humid (Yes/No)
C= Play golf

InfoGain(A1)=0

InfoGain(A2)=0

InfoGain(A3)=0.188

The greedy solution would be to split on A3

However, using A1 and A2 together can completely classify C

if(A1==A2)	Don't play
else	Play

2. The best rule set may contain overlapping rules that interact

The most concise rule set may be achieved with overlapping rules with different priorities

Sunny	Hot	Play
1	1	0
1	1	0
1	0	1
1	0	1
1	1	0
0	1	0
0	0	1
1	0	1

if Sunny **then** play is only 40% accurate

But if we combine two overlapping rules, giving priority to the first, we get a concise accurate description

if Sunny and Hot **then** Don't Play (Priority 10)

if Sunny **then** Play (Priority 5)

Rule Representation

It is common to include wild cards in the rule representation to indicate “don’t care”

Sunny	Hot	Humid	Play

if Sunny **then** Play	1 * * /1
if Not Sunny and Hot **then** Don’t Play	0 1 * /0
Play	* * * /1
if Hot and Humid **then** Don’t Play	* 1 1 /0

This representation does not allow for compact expression of OR rules

if Sunny or Hot **then** Play	1 * * /1
	* 1 * /1

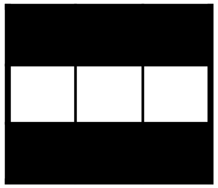
Detectors

The agent detectors take the environment signal and map it onto the internal rule based representation

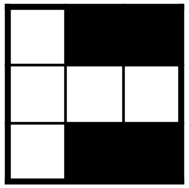
Environment state:

Detector

Internal Message



$\longrightarrow$ 11101110



$\longrightarrow$ 01101100

We can use some form of encoding for discrete and continuous variables

Outlook =Sunny
Temp=78.9
Humidity=65

$\longrightarrow$ 00 1011 0011

Rule Set

At any point the agent maintains a set of rules which are Condition/Action pairs

Condition				Action
1	*	1	1	1
*	*	0	*	0
1	1	1	1	1
*	0	0	*	1
1	*	1	1	0
0	*	*	1	0
1	*	1	*	1
0	0	*	0	1

The condition part is in the syntax of the detector

The rules do not have to be consistent

Each rule has a set of parameters that are influence how the rule is used to choose an action

Rule Parameters

Prediction: *P*

Each rule has a prediction parameter. This is the predicted expected return for taking the proscribed action in environments that match the condition

Error: *e*

The error estimates the variation in reward (important if rewards are at different levels)

Fitness: *f*

Measures the quality of a rule in relation to the other rules in the rule set

Production System

The agent uses the environment message and the rule set to choose an action:

1. Form a match set

   Find all rules that match the environment message

2. Form a prediction array

   Estimate the expected reward for each action

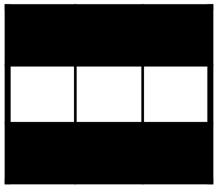
3. Choose an action

   Choose based on the agent exploit/explore trade off strategy

4. Form the action set

   Rewards will be allocated to the rules with the chosen action

Production System Example



Env State

Detector

$$\begin{tikzcd}\text{Detector} \arrow[d] \\ 11101110\end{tikzcd}$$

Env Message

Rule Set

Cond								Act	Pred	Err	Fit
1	*	1	0	1	*	1	*	1	75	22	0.1
*	*	0	*	*	*	0	*	0	73	11	0.25
1	1	1	0	*	*	1	0	1	70	0	0.3
*	0	0	*	*	0	0	*	1	67	15	0.67
1	*	1	1	1	*	1	1	0	65	33	0.13
0	*	*	1	0	*	*	1	0	63	4	0.22
1	*	1	*	1	*	1	*	0	10	10	0.09
0	0	*	0	0	0	*	0	1	0	0	0.01

1. Form Match Set

1	1	1	0	*	*	1	1	1	70	0	0.3
1	*	1	*	1	*	1	*	0	10	10	0.09
1	*	1	0	1	*	1	*	1	75	22	0.1

Production System Example

Match Set M

Cond								Act	Pred	Err	Fit
1	1	1	0	*	*	1	1	1	70	0	0.3
1	*	1	*	1	*	1	*	0	10	10	0.09
1	*	1	0	1	*	1	*	1	75	22	0.1

2. Form a prediction array P

One technique is simply to *average the predictions of the rules advocating each action*

P(0)=10
P(1)=(70+75)/2=72.5

3. Choose an action a

Greedy Action is argmax(P)=1

4. Form the action set A

1	1	1	0	*	*	1	1	1	70	0	0.3
1	*	1	0	1	*	1	*	1	75	22	0.1

Learning/Reinforcement System

Once the action has been made, a reward is received from the environment.

This could be a food signal for reinforcement learning or an indicator of correct classification for machine learning

This signal is used to adjust the parameters of the rules advocating the action

There are several alternative ways of updating. It is common to use the Widrow-Hoff update for the prediction

$$P_{t+1} = (1-\beta)P_t + \beta \cdot R_t$$

$$e_{t+1} = (1-\beta)e_t + \beta(|R_t - P_t|)$$

$$\downarrow$$

$$P_{t+1} = P_t + \beta(R_t - P_t)$$

We will see an example of reinforcing the action set later

Rule Discovery Mechanism

The LCS also needs to be able to discover new rules

There are two scenarios when the rule set is altered:

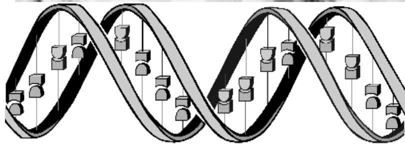
1. There are no matching rules

   In this situation a new rule is created to match the current environment. Wildcards are randomly assigned and the rule is given a random action

2. The current rule set is not good enough

   The rule conditions may only be too general or too specific. Certain areas of the attribute space may be misclassified. New rules are created with a **genetic algorithm**

Genetic Algorithms



Genetic Algorithms (GA) were invented in mid-1970s by John Holland

GAs are based on the principles of:

Evolution via survival of the fittest and

Genetic transfer of information via mating

A genetic algorithm maintains a population of solutions. Each solution has an associated fitness;

1. At each iteration of the GA (each *generation*) solutions (parents) are selected for mating according to fitness;
2. New solutions (offspring) are created to combine elements of both parents;
3. These offspring are merged back into the population, possibly replacing existing solutions.

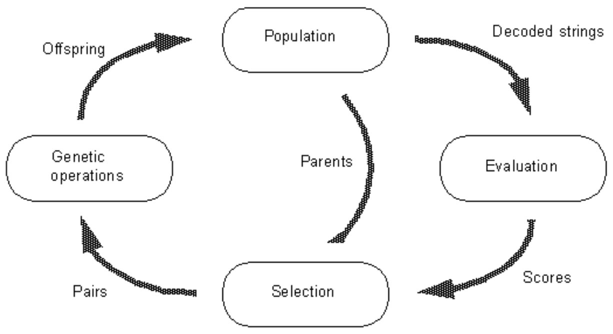
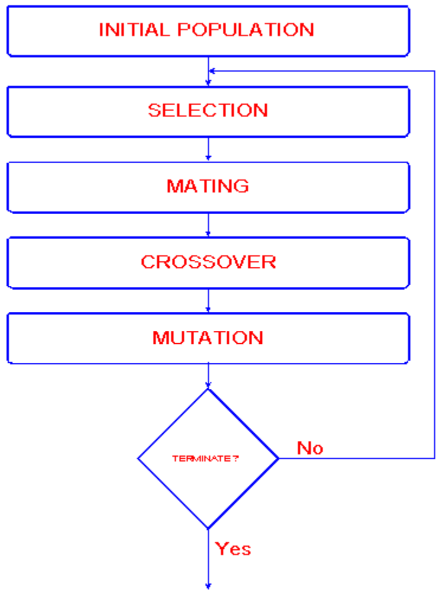


Figure 5.2: The "reproduction" cycle.

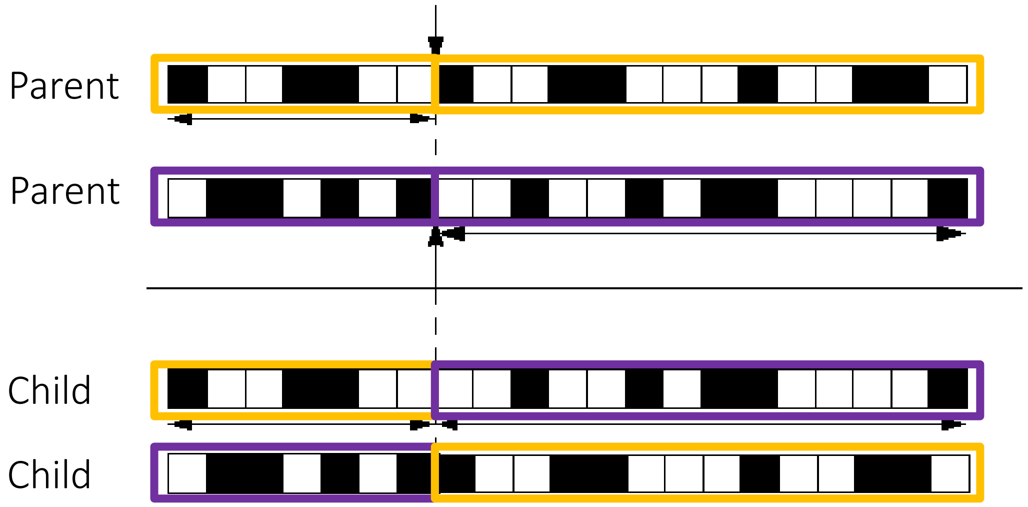
A Genetic Algorithm



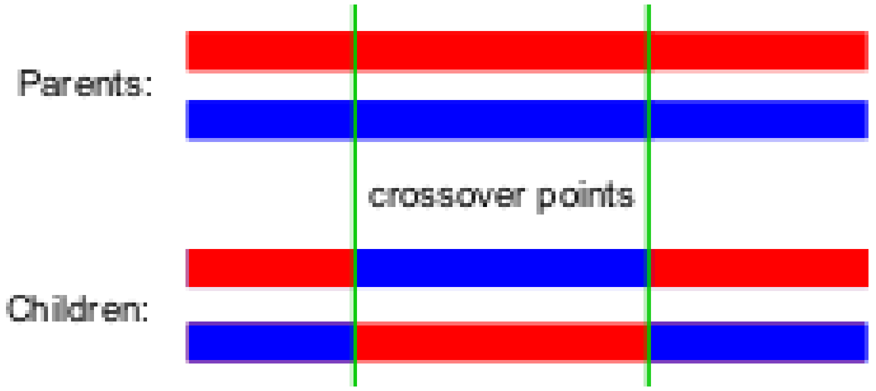
```
GA Algorithm
        Choose an initial population
        **while** not finished
            **Select** Parents
                *e.g select proportional to fitness*
            **Create** Offspring through crossover and mutation
            **Merge** offspring back into population
                *e.g. delete least fit rules*
```

Genetic Algorithm Crossover

One Point Crossover



Two Point Crossover



Genetic Algorithm Crossover

A population is a set of solutions, with an associated fitness

00011100001111	90
10110010100111	51
11011010001000	43
10101010111010	54

Suppose two parents are selected for crossover
00011100001111 and 11011010001000

Crossover involves taking part of the solution from each parent, and combining it to produce the offspring

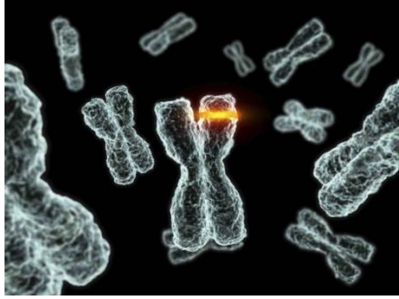
Parents 0001110|0001111
1101101|0001000

CrossPoint = 7

Offspring 1 : 0001110+0001000 = 00011100001000

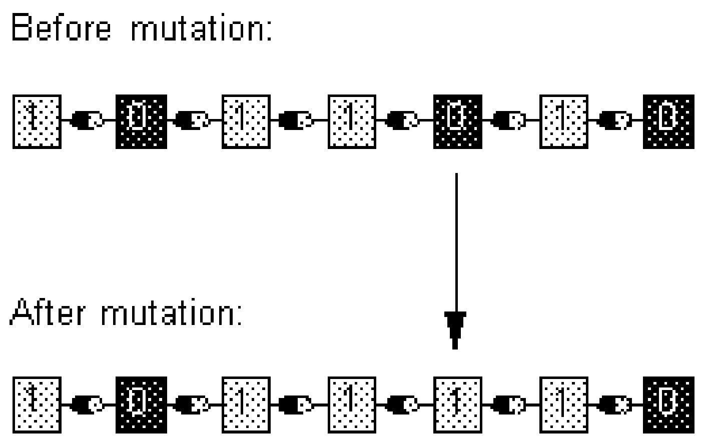
Offspring 2 : 1101101+0001111 = 11011010001111

Genetic Algorithm Mutation



Mutation is the second genetic operator that is applied

Randomly selected bits are flipped to the opposite value



After the application of genetic operators the fitness of each new solution is evaluated

Algorithm LearningClassifierSystem(dataSet D)

```
RuleSet R, M, A;
While ! Stop(R,D)
    d=getEnvironment(D)
    M:=formMatchSet(R,d);
    If(|M|=0)
            M:=coverCase(R,d) ;
    P:=formPredictionArray(M);
    a:=chooseAction(P);
    A:=formActionSet(M,a);
    r:=getReward(d,a);
    reinforce(A,r);
    if(triggerGA(R,A))
            R:=ruleCreation(R,A);
```

A Simple LCS (Bull's YCS)

A commonly used form of LCS is Wilson's error based classifier, XCS (1996).

It is too complex to teach in this time period. A simpler LCS based on the same principles was described in 2003 by Larry Bull.

Rule Parameters

Each rule has a Prediction, *p*, Error e and Fitness *f*.

In addition, each rule has an **estimate of the average action set size,** α

Yet another Classifier System (YCS)

1: Production System

A prediction array is formed by finding the average value for each class, weighted by the fitness.

Actions are selected either randomly (when learning) or greedily as the best predicted reward

Cond								Act	Pred	Err	Fit
1	1	1	0	*	*	1	1	1	70	0	0.3
1	*	1	*	1	*	1	*	0	10	10	0.09
1	*	1	0	1	*	1	*	1	75	22	0.1

P(0)=(0.09*10)/0.09=10
P(1)=(0.3*70+0.1*75)/(0.3+0.1)=71.25

Yet another Classifier System (YCS)

2. Reinforcement System. The rules in the action set are updated as follows

Prediction

$$P_{t+1} = (1-\beta)P_t + \beta \cdot R_t$$

Error

$$e_{t+1} = (1-\beta)e_t + \beta\left(|R_t - P_t|\right)$$

Action Set Size:

$$\alpha_{t+1} = (1-\beta)\cdot\alpha_t + \beta\,|A_t|$$

Fitness:

$$f_{t+1} = \frac{1}{\left(e_{t+1}+1\right)}$$

Yet another Classifier System (YCS)

3. Rule Discovery

1. **Trigger GA**: A Panmictic GA which triggers on each step with probability *g*
2. **GA Select**: Two parents are selected by roulette selection from the entire population
3. **GA Create:** One point crossover
4. **GA Merge:** Rules are deleted proportional to action set size. Rules belonging to larger action sets are *more* likely to be deleted

Example YCS Iteration

Learning rate = 0.2

Case 10011 (correct ==1)

Rule Set									
Cond					Action	Pred	Error	Fit	alpha
*	0	0	1	*	1	500	10	0.8	6
1	*	*	1	1	0	700	90	0.7	3
0	*	*	*	*	1	200	20	0.7	4
*	*	*	1	1	0	300	110	0.1	2
1	0	0	*	*	0	250	50	0.2	4
*	*	*	1	*	1	400	75	0.3	3
1	1	1	0	0	0	600	100	0.1	2
0	*	0	*	0	1	100	0	1	5
0	1	1	0	*	0	0	0	1	1

1. Production System

Match Set					Action	Pred	Error	Fit
*	0	0	1	*	1	500	10	0.8
*	*	*	1	*	1	400	75	0.3
1	*	*	1	1	0	700	90	0.7
*	*	*	1	1	0	300	110	0.1
1	0	0	*	*	0	250	50	0.2

Case 10011 (correct ==1)

Form Prediction Array

Remember: weighting by *fitness*

$$P(0) = \frac{0.7 \times 700 + 0.1 \times 300 + 0.2 \times 250}{0.7 + 0.1 + 0.2} = \frac{570}{1} = 570$$

$$P(1) = \frac{0.8 \times 500 + 0.3 \times 400}{0.8 + 0.3} = \frac{520}{1.1} = 427.73$$

Choose Action

So action = 0

570 > 427

2. Reinforcement

Assume a reward of 1000 for correct classification and 0 for incorrect

Form action set: action = 0

Case 10011 (correct ==1)

Action Set					Action	Pred	Error	Fit
1	*	*	1	1	0	700	90	0.7
*	*	*	1	1	0	300	110	0.1
1	0	0	*	*	0	250	50	0.2

1st Rule

$$P_{t+1} = (1-\beta)P_t + \beta \cdot R_t$$
$$e_{t+1} = (1-\beta)e_t + \beta\left(|R_t - P_t|\right)$$
$$f_{t+1} = \frac{1}{(e_{t+1}+1)}$$

Pred = $0.8 \times 700 + 0.2 \times 0 = 560$

Error = $0.8 \times 90 + 0.2 \times (|0 - 700|) = 212$

Fitness = $\frac{1}{(212+1)} = 0.004717$

2. Reinforcement

Assume a reward of 1000 for correct classification and 0 for incorrect

Form action set: action = 0

Case 10011 (correct ==1)

Action Set					Action	Pred	Error	Fit
1	*	*	1	1	0	560	212	0.004
*	*	*	1	1	0	300	110	0.1
1	0	0	*	*	0	250	50	0.2

2nd Rule

$$P_{t+1} = (1-\beta)P_t + \beta \cdot R_t$$
$$e_{t+1} = (1-\beta)e_t + \beta\left(|R_t - P_t|\right)$$
$$f_{t+1} = \frac{1}{(e_{t+1}+1)}$$

Pred = $0.8 \times 300 + 0.2 \times 0 =$240

Error = 0.8$\times$ $110 + 0.2 \times (|0 - 300|) = 148$

Fitness = $\frac{1}{(148+1)} = 0.00671$

And so on for the rest of the action set
(1 more in this case)

3. Rule Discovery

1. Trigger

Probability of GA = g. Suppose it is triggered on this iteration

2. Select:

					Action	Fitness	Relative	Cumula
*	0	0	1	*	1	0.8	0.16	0.16
1	*	*	1	1	0	0.7	0.14	0.31
0	*	*	*	*	1	0.7	0.14	0.45
*	*	*	1	1	0	0.1	0.02	0.47
1	0	0	*	*	0	0.2	0.04	0.51
*	*	*	1	*	1	0.3	0.06	0.57
1	1	1	0	0	0	0.1	0.02	0.59
0	*	0	*	0	1	1	0.20	0.80
0	1	1	0	*	0	1	0.20	1.00

3. Rule Discovery

Assume Rules 1 and 2 are chosen to mate

					Action
*	0	0	1	*	1
1	*	*	1	1	0

Let’s do a one point crossover at position 3

*	0	0	1	*
1	*	*	1	1

*	0	0	1	1
1	*	*	1	*

LCS: Further Issues

1. Internal memory

The original LCS proposed linkage of rules through internal memory and a “Bucket Brigade” message passing. More recent systems have a built in anticipation mechanism

2. GA in the action set

Does it make sense to mate rules with different actions?

3. LCS Evaluation

Joint function of ML and RL confuses things somewhat

4. Error or Prediction

Joint function of ML and RL confuses things somewhat