



# ***Artificial Intelligence Programming Classifier Systems***

Chris Brooks

Department of Computer Science  
University of San Francisco

## 4-0: Classifier Systems

- ⑥ Genetic Algorithms work very well for optimization problems
  - △  $f(x_1, x_2, x_3, \dots, x_n) = R$
- ⑥ What about less well-defined problems?
- ⑥ Can we use these ideas to build a vacuum-cleaner agent?
- ⑥ (yes, but we'll need to be a bit more careful)
- ⑥ *Classification* is the process of assigning an input to one of several classes.
  - △ We'll see much more of this later on.

## 4-1: *Production Systems*

- ⑥ Production systems are a common AI technique
- ⑥ if-then, or condition-action rules
- ⑥ if [1,1] has not been vacuumed and is adjacent, then move to [1,1]
- ⑥ A *Production system* will consist of a large body of rules.
  - △ Some rules may cause other rules to fire.
- ⑥ In traditional systems, the problem is what to do when more than one rule is matched.
  - △ (We'll return to this in a few weeks.)

## 4-2: *Classifiers*

- ⑥ Genetic Algorithms potentially give us a way to evolve and select rules.
  - △ Pick the rule with the highest fitness.
- ⑥ Encode rules as a bitstring, with 0,1,\*
- ⑥ condition bits : action bits
- ⑥ Issues:
  - △ How to assign fitness to a rule?
  - △ How to maintain high online performance?

## 4-3: *Components of a classifier system*

- ⑥ A Classifier system has three components:
  - △ A rule and message system
  - △ A credit assignment system
  - △ A GA for generating new rules

## 4-4: *Rule and message system*

- ⑥ The agent's sensors receive information, which is encoded as a bitstring.
  - △ This information is a *message* from the environment.
- ⑥ This message activates classifiers (rules) with matching conditions.
- ⑥ These classifiers post their messages to a message list.
  - △ These messages may activate other classifiers, or the agent's effectors.

## 4-5: Example

⑥ Assume our system has the following classifiers:

- 1) 01## : 0000
- 2) 00#0 : 1100
- 3) 11## : 1000
- 4) ##00 : 0001

⑥ Message 0111 arrives from the environment.

- △ Rule 1 fires, placing 0000 in the message list.
- △ Rules 2 and 4 fire, placing 1100 and 0001 in the message list.
- △ Rule 3 fires, placing 1000 in the message list.
- △ This matches rule 4, whose message is in the message list.

⑥ There are several messages now in the message list - which is sent to the effectors?

## 4-6: *Bucket Brigade*

- ⑥ The bucket brigade algorithm allows rules to “bid” on firing based on past performance.
- ⑥ When a rule is matched, it participates in an “auction.”
- ⑥ Each rule has a strength, based on past performance.
  - △ A rule bids a constant proportion of its strength.
- ⑥ Highest bidding rules win out.
- ⑥ This bid is sent to the classifier(s) that activated it.

## 4-7: Example

### 6 Initially ( $t = 0$ ): $M = 0111$

- 1) 01## : 0000     $S = 200$      $B = 20$
- 2) 00#0 : 1100     $S = 200$
- 3) 11## : 1000     $S = 200$
- 4) ##00 : 0001     $S = 200$

Environment     $S = 0$

## 4-8: Example

6 **t = 1**

1) 01## : 0000    S = 180                    0000

2) 00#0 : 1100    S = 200    B = 20

3) 11## : 1000    S = 200

4) ##00 : 0001    S = 200    B = 20

Environment        S = 20

## 4-9: Example

6 t = 2

1) 01##	:	0000	S = 220		1100
2) 00#0	:	1100	S = 180		0001
3) 11##	:	1000	S = 200	B = 20	
4) ##00	:	0001	S = 180	B = 18	

Environment      S = 20

## 4-10: Example

6  $t = 3$

1) 01##	:	0000	S = 220	1000
2) 00#0	:	1100	S = 218	0001
3) 11##	:	1000	S = 180	
4) ##00	:	0001	S = 162	B = 16
Environment			S = 20	

## 4-11: Example

6 **t = 4**

1) 01##	:	0000	S = 220	0001
2) 00#0	:	1100	S = 208	
3) 11##	:	1000	S = 196	
4) ##00	:	0001	S = 156	
Environment			S = 20	

## 4-12: Example

- ⑥  $t = 5$  - payment comes into system, assigned to last active classifier.

1) 01## : 0000      S = 220

2) 00#0 : 1100      S = 208

3) 11## : 1000      S = 196

4) ##00 : 0001      S = 206

Environment      S = 20

## 4-13: *Generating new rules*

- ⑥ Bucket brigade provides a way of selecting rules and assigning credit.
- ⑥ How to get new rules?
- ⑥ Our basic GA uses a *nonoverlapping* population model.
  - △ All the population at time  $t$  is replaced at time  $t + 1$
- ⑥ This works well for optimization, but not so much for learning.
- ⑥ Instead, we use elitism to retain some rules.
- ⑥ Use roulette selection to retain rules.
- ⑥ Also, evolve more slowly - every  $T$  iterations rather than 1.

## 4-14: Genetic Programming - beyond rules

- ⑥ Evolutionary techniques have also been applied to the construction of programs.
- ⑥ Idea: A program can be represented as an *S-expression*
  - △  $(+ 3 (* 4 x))$
- ⑥ We can draw this as a tree.
- ⑥ We then perform evolution on a population of programs.

# 4-15: Genetic Programming - beyond rules

- ⑥ Fitness is the performance of a program on a given task.
- ⑥ Crossover swaps subtrees.
- ⑥ Mutation replaces operators.
- ⑥ Used to evolve:
  - △ Robot soccer programs
  - △ Analog control circuits
  - △ Filters
  - △ Complex circuits
- ⑥ Challenges:
  - △ Vast (infinite) space of programs
  - △ Lack of higher-level program structures