Master of Technology

Computational Intelligence II

GA Workshop 2: Rule Induction

Dr. Zhu Fangming
Institute of Systems Science,
National University of Singapore
Email: isszfm@nus.edu.sg

Problem Description

- A supermarket chain has just completed a customer survey and would like to understand customer spending behaviour.

- Your team has been assigned to review the survey results for carpet cleaners.

- Data available
  - **Name** of product
  - **Design** of the package
  - **Price**
  - **Seal** of approval (for safety)
  - 30-day money-back guarantee (**MBG**)

Problem Description

No	Product Name	Package Design	Product Price	Seal	MBG	Acceptable?
1	K2R	A	119	No	No	No
2	Glory	A	139	No	Yes	No
3	Bissell	A	159	Yes	No	No
4	K2R	B	119	Yes	Yes	Yes
5	Glory	B	139	No	No	No
6	Bissell	B	159	No	No	Yes
7	K2R	C	119	No	Yes	No
8	Glory	C	139	Yes	No	No
9	Bissell	C	159	No	No	No
10	K2R	A	119	Yes	No	No
11	Glory	A	139	No	Yes	No
12	Bissell	A	159	No	No	No
13	K2R	B	119	No	No	Yes
14	Glory	B	139	Yes	No	Yes
15	Bissell	B	159	No	Yes	Yes
16	K2R	C	119	No	No	No
17	Glory	C	139	No	No	Yes
18	Bissell	C	159	Yes	Yes	No

Problem Model

1. Representation of solution

- It really depends on how sophisticated we want our rule to be. In this exercise, we'll keep it simple.
- The solution will comprise one chromosome representing the parameterisable fields of this rule.
- Let's decide the lower-level details a little later.

Problem Model

2. Constraints

- Application-specific
  - Each field has its own set of constraints

    e.g. name $\in$ {Glory, Bissell, K2R}

    money back guarantee $\in$ {yes, no}
- Model-specific
  - Depends on how precise / general you want the rule to be

    e.g. can some fields be ignored?

Problem Model

3. Fitness function

- Our goal is to find a rule that matches our survey results most closely.
- We can characterise our fitness function as the number of matches our proposed rule has against the survey results.

Rule Form

Let's start with an example:

if name = Glory *and*

package-design = B *and*

price >= 159 *and*

seal of approval = yes *and*

MBG = yes

then product is acceptable to customer

Rule Form

Two levels of parameterisation: operators and values

- Operators
  - Discrete values e.g. name, MBG → equality

    e.g. name **= or !=** Glory
  - Continuous values e.g. price → inequality

    e.g. price **<= or >=** 159
- Values
  - Dependent on application-specific requirements

    e.g. name ∈ {Glory, Bissell, K2R}

Rule Form

The simplest form of the rule is

if name (= *or* !=) (Glory, Bissell *or* K2R) *and*

package-design (= *or* !=) (A, B *or* C) *and*

price (= *or* <= *or* >=) (119, 139 *or* 159) *and*

seal of approval (= *or* !=) (yes *or* no) *and*

MBG (= *or* !=) (yes *or* no)

then product is acceptable to customer

Tabular Rule Form

Attribute	Operator	Value
Name	= *or* !=	Glory, Bissell or K2R
Design	= *or* !=	A, B *or* C
Price	= *or* <= *or* >=	119, 139 *or 159*
Seal	= *or* !=	yes *or* no
MBG	= *or* !=	yes *or* no

Problem with This Approach

- Requiring the rule to make a statement about *every* attribute is unrealistic.
- For example, brand-conscious car buyers will ignore price e.g. Rolls-Royce or Bentley, whereas price conscious buyers will not care for the brand at all.
- We will allow a “don’t care” symbol written *.

Updated Tabular Rule Form

Attribute	Operator	Value
Name	= *or* !=	Glory, Bissell, K2R *or* *
Design	= *or* !=	A, B, C *or* *
Price	= *or* <= *or* >=	119, 139, 159 *or* *
Seal	= *or* !=	yes, no *or* *
MBG	= *or* !=	yes, no *or* *

Model-Specific Constraints

- One model-specific consideration we highlighted earlier was how precise /general we wanted our rule to be.
- We can set a hard constraint to restrict a maximum number of “don’t care” symbols in our rule – otherwise, it’s so general it tells us absolutely nothing!

Value Mapping

- As we need a chromosome representation using integer values, we should map the values (e.g. Glory, yes, !=) into the space of non-negative integers (e.g. 0, 1 or 2).

Equality operators -	1:equal, 2:not equal	(for non-numerical values)
Inequality operators -	1:equal, 2: <=, 3: >=	(only for numerical values)
Problem value -	0:don't care, 1:A, 2:B, 3:C	
	0:don't care, 1:K2R, 2:glory,3:bissell	
	0:don't care, 1:119, 2:139, 3:159	
	0:don't care, 1:yes, 2:no (for **seal** and **MBG**)	
	0: no, 1:yes (for **acceptable**)	