Genetic algorithms

Outline:

1. What are genetic algorithms (GA)?
2. The components of GA.
3. GA: first example.
4. GA: second example.
5. Genetic programming (GP).
6. Applications of GA.

1. What are genetic algorithms (GA)?

- GA are search algorithms based on the mechanic of natural selection and natural genetics.
- They involve randomised but structured information exchange and survival of the fittest.
- How do they differ from other optimisation and search algorithms?
    - GA work with a coding of the parameter set, not the parameters themselves.
    - GA search from a population of points, not a single point.
    - GA use payoff (objective function) information, not derivatives or other auxiliary knowledge.
    - GA use probabilistic transition rules, not deterministic rules.

Outline of GA.

- GA start with a set of individuals and apply selection and reproduction operators to evolve an individual that is most successful as measured by the fitness function.
- Outline of the algorithm:

```
function GA(population, fitness-function)
• inputs: population - a set of inviduals
          fitness-function - the fitness of an individual
• output: fittest individual
   – repeat
       o    parents <- select(population, fitness-function)
       o    population <- reproduce(parents)
     until some individual is fit enough
   – return the best individual in population according to fitness-function.
```

Parameters in GA.

- The size of the population.
- The first generation.
- The selection for reproduction.
- The pairing between fit parents.
- The crossover site for each pair selected for reproduction.
- The mutation site.

2. The components of GA

The are four components of GA: fitness function, gene, selection strategy and reproduction operators.

1. **Fitness function:**

   • It takes an individual as input and returns a real number as output.

   • The type of fitness function depends on the individuals.

   • The fitness function can be a performance measure or reward function, or a critic, or anything at all that can be framed as an optimisation problem.

The components of GA (continued)

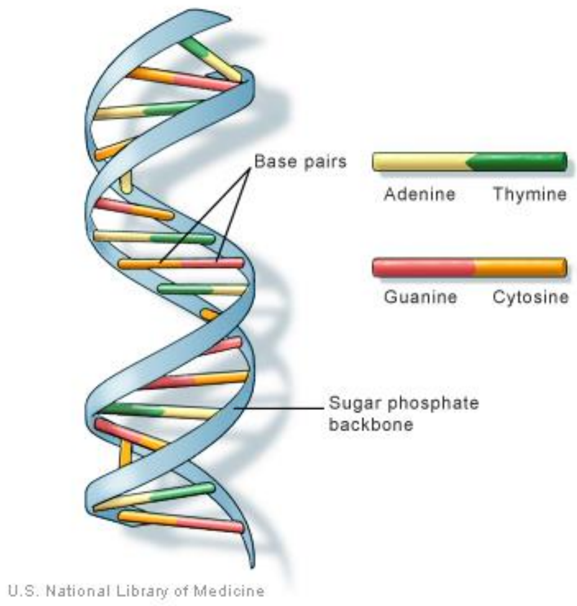
2. **<u>Gene</u>:**

- The basic unit of an individual who is represented as a string over a finite alphabet.
- Each element of the string is called a gene.
- In real DNA, the alphabet consists of

A (adenin), G (guanine), T (thymine), or

C (cytosine).

- In GA, the binary alphabet (0,1) is usually used.



The components of GA (continued)

3. **Selection strategy:**

- It is usually randomised, with the probability of selection proportional to the fitness.
- If individual X scores twice as high as individual Y on the fitness function, then X is twice more likely to be selected for reproduction than Y.
- Usually, selection is done with replacement, so that a very fit individual will get to reproduce several times.

The components of GA (continued)

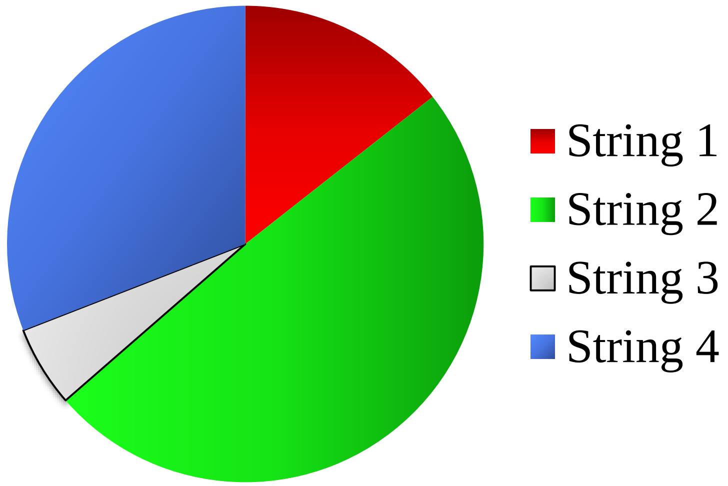
4. **Reproduction operators:**

- **reproduction:** a process in which individuals are copied according to their fitness values. The fitter an individual, the more copies it has. This operator is an artificial version of natural selection. An easy way to implement reproduction is to create a biased roulette wheel where each individual in the population has a roulette wheel slot sized in proportion to its fitness.
- **crossover:** all the individuals that have been selected for reproduction are randomly paired. For each pair, a crossover point is randomly chosen. The crossover point is the point where the string representing an individual is split into 2 parts. Two new individuals are created by swapping the second parts of the pair.

The components of GA : reproduction operator

Sample problem strings and fitness values

No	String	Fitness	% of total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0



- To reproduce, spin the weighted roulette wheel and get the reproduction candidate.
- More highly fit strings have a higher number of offspring in the next generation.
- Once a string is selected, an exact replica is made and it is entered into a mating pool for further genetic operator.

The components of GA: crossover operators

Different crossover operators:

- Single point crossover

$$\begin{tikzcd}
11101001000 \arrow[dr, dash] & \text{Crossover mask} & 11101010101 \\
 & 11111000000 \arrow[ur] \arrow[dr] & \\
00001010101 \arrow[ur, dash] & & 00001001000
\end{tikzcd}$$

Initial strings | Offspring

- Two-point crossover

$$\begin{tikzcd}
11101001000 \arrow[dr, dash] & \text{Crossover mask} & 11001011000 \\
 & 00111110000 \arrow[ur] \arrow[dr] & \\
00001010101 \arrow[ur, dash] & & 00101000101
\end{tikzcd}$$

Initial strings | Offspring

- Uniform crossover

$$\begin{tikzcd}
11101001000 \arrow[dr, dash] & \text{Crossover mask} & 10001000100 \\
 & 10011010011 \arrow[ur] \arrow[dr] & \\
00001010101 \arrow[ur, dash] & & 01101011001
\end{tikzcd}$$

Initial strings | Offspring

The components of GA: crossover operators

Single point crossover.

- The crossover mask always begins with a string of contiguous 1's followed by the necessary number of 0's to complete the string.
- The first offspring takes its bits from the first parent corresponding to 1's in the mask, and its remaining bits from the second parent corresponding to 0's in the mask.

Two-point crossover.

- The crossover mask begins with n_0 zeros, followed by n_1 ones, and followed by the necessary number of zeros.

Uniform crossover.

- The mask consists of bits that are generated randomly and independent from one another.

The components of GA : mutation

<u>Point mutation.</u>

• It produces offspring from a single parent by introducing small random changes to the parent.

$$111010\underline{0}1000 \xrightarrow{\hspace{6cm}} 111010\underline{1}1000$$

• Mutation is often performed after crossover has been applied.

3. GA: first example

Maximise the function $f(x) = x^2$ on the integer interval [0,31].

• GA requires the natural parameter set of the optimisation to be coded as a finite-length string over some alphabet.

• Let us use a 5-bit binary representation to code the parameter x.

• The next step is to decide the population size, say N = 4.

• We randomly generate four initial strings:

01101

11000

01000

10011

• Successive populations are generated by the algorithm.

GA: first example (continued)

Initial string	01101	11000	01000	10011
x	13	24	8	19
$F(x) = x^2$	169	576	64	361
F/sum(F(x))	0.14	0.49	0.06	0.31
F/ave(F(x))	0.58	1.97	0.22	1.23
Roulette wheel count	1	2	0	1

sum(F(x)) = 1170

ave(F(x)) = 293

Randomly select pairs and crossover site:

Generation 1	01101	11000	11000	10011
Generation 2	01100	11001	11011	10000
x	12	25	27	16
F(x)	144	625	729	256

sum(F(x)) = 1754

4. GA: second example

- Consider the hypothesis consisting of two rules:

  If $a_1 = T \wedge a_2 = F$ then $c = T$; If $a_2 = T$ then $c = F$

  which is represented by the string:

a_1	a_2	c	a_1	a_2	c
10	01	1	11	10	0

- Note the first 2 binary bits correspond to hypothesis a_1:

  $(b_1,b_2) =$ (1,0) iff $a_1 = T$, (0,1) iff $a_1 = F$

  (1,1) iff $a_1 = T$ or F (don't care)

  $(b_3,b_4) =$ (1,0) iff $a_2 = T$, (0,1) iff $a_2 = F$

  (1,1) iff $a_2 = T$ or F (don't care)

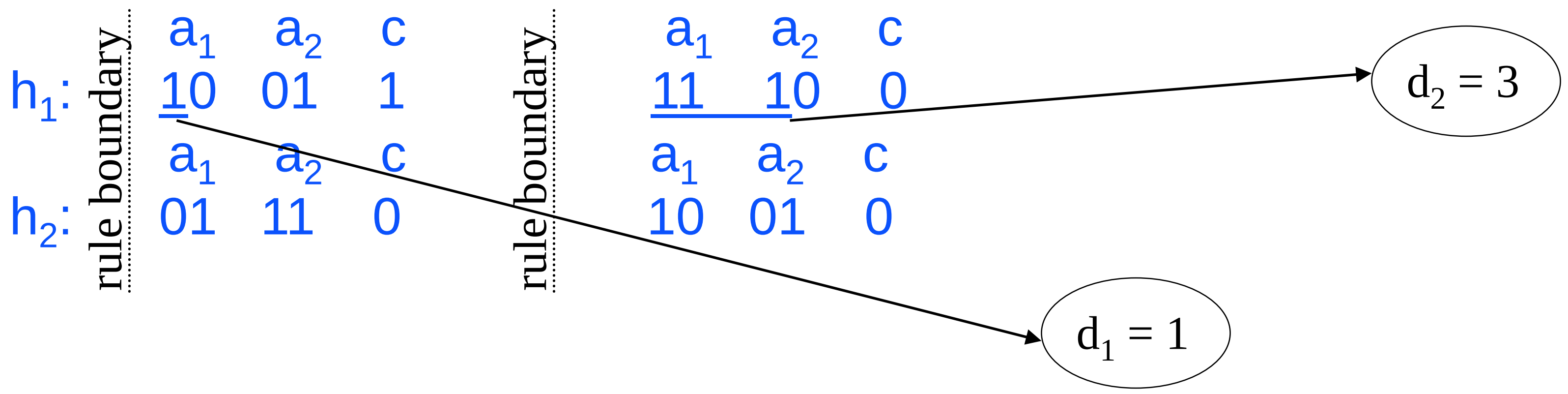
  $b_5 = 0$ iff the predicted target value of attribute c is F

  $b_5 = 1$ iff the predicted target value of attribute c is T

GA: second example (continued)

Genetic operators.

- Assume that the two parents are:



h1: a1 a2 c 10 01 1 | rule boundary | a1 a2 c 11 10 0
h2: a1 a2 c 01 11 0 | rule boundary | a1 a2 c 10 01 0
rule boundary
rule boundary
d2 = 3
d1 = 1

- The crossover points for the first parent are the points following bit positions 1 and 8.
- Let d_1 be the distance from the leftmost of the two crossover points to the rule boundary immediately to its left, $d_1 = 1$.
- Let d_2 be the distance from the rightmost of the two crossover points to the rule boundary immediately to its left, $d_2 = 3$.

GA: second example (continued)

- The allowed pairs of crossover points for the second parent include the pairs in bit positions <1,3>, <1,8>, and <6,8>. This restriction is imposed so that crossover occurs only between sections of the bit strings that encode rules.
- Suppose the pair <1,3> is chosen:

	a_1	a_2	c	a_1	a_2	c
h_1:	10	01	1	11	10	0
h_2:	01	11	0	10	01	0

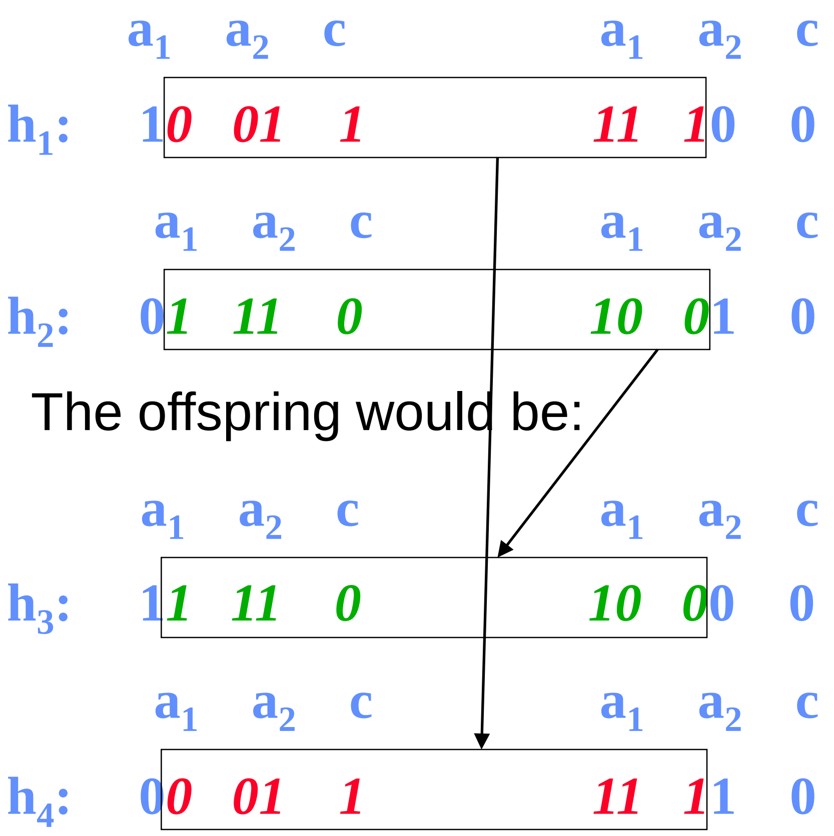
- The offspring would be:

	a_1	a_2	c	a_1	a_2	c	a_1	a_2	c
h_3:	11	10	0						
h_4:	00	01	1	11	11	0	10	01	0

GA: second example (continued)

- Suppose the pair <1,8> is chosen as the cross over points for the second parent:

- The offspring would be:



a_1 a_2 c a_1 a_2 c
h_1: 10 01 1 11 10 0
a_1 a_2 c a_1 a_2 c
h_2: 01 11 0 10 01 0
a_1 a_2 c a_1 a_2 c
h_3: 11 11 0 10 00 0
a_1 a_2 c a_1 a_2 c
h_4: 00 01 1 11 11 0

GA: second example (continued)

Suppose the pair <6,8> is chosen as the cross over points for the second parent:

a_1 a_2 c a_1 a_2 c

h_1: 1*0* *01* *1* *11* *1*0 0

a_1 a_2 c a_1 a_2 c

h_2: 01 11 0 1*0* *0*1 0

- The offspring would be:

a_1 a_2 c

h_3: 1*0* *0*0 0

a_1 a_2 c a_1 a_2 c a_1 a_2 c

h_4: 01 11 0 1*0* *01* *1* *11* *1*1 0

GA: second example (continued)

- How do we compute the fitness of these individuals (h_3 and h_4)?

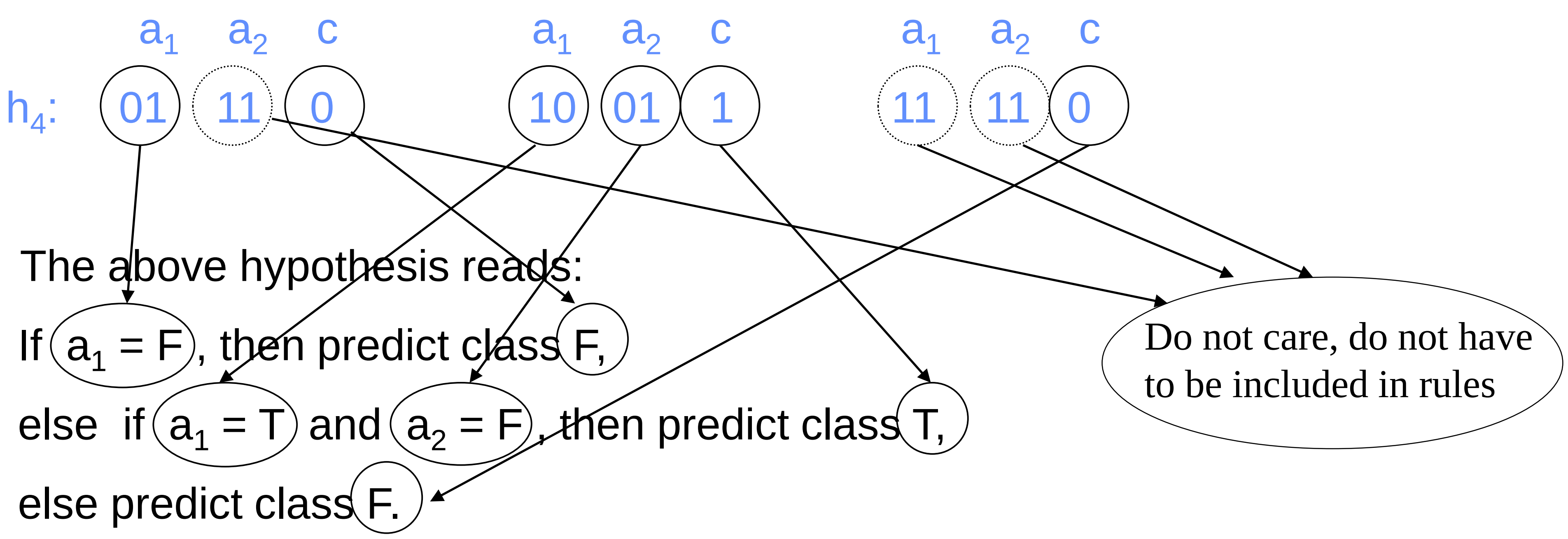
$$\begin{array}{llll} & a_1 & a_2 & c \\ h_3: & 10 & 00 & 0 \end{array}$$

- If we assume that at least one of the two bits (b_1,b_2) must be 1 and similarly at least one of the two bits (b_3,b_4) must be 1 since a_1 must be T or F and a_2 must also be either T or F.
- No sample satisfies the conditions of h_3.
- In this case, we can use the majority class as the predicted class, i.e., if there are more samples from class T, we always predict T, otherwise we always predict F.
- Say 6 of 10 samples from the training data set are from class T, then

$$\text{Fitness}(h_3) = 6^2 = 36$$

GA: second example (continued)

- How do we compute the fitness of h_4?



a_1 a_2 c a_1 a_2 c a_1 a_2 c
h_4: 01 11 0 10 01 1 11 11 0
Do not care, do not have to be included in rules

- The above hypothesis reads:

  If a_1 = F, then predict class F,

  else if a_1 = T and a_2 = F, then predict class T,

  else predict class F.

- We can now compute how many of the 10 training samples are correctly predicted by the above rule/hypothesis and the corresponding fitness(h_4).

5. Genetic programming (GP)

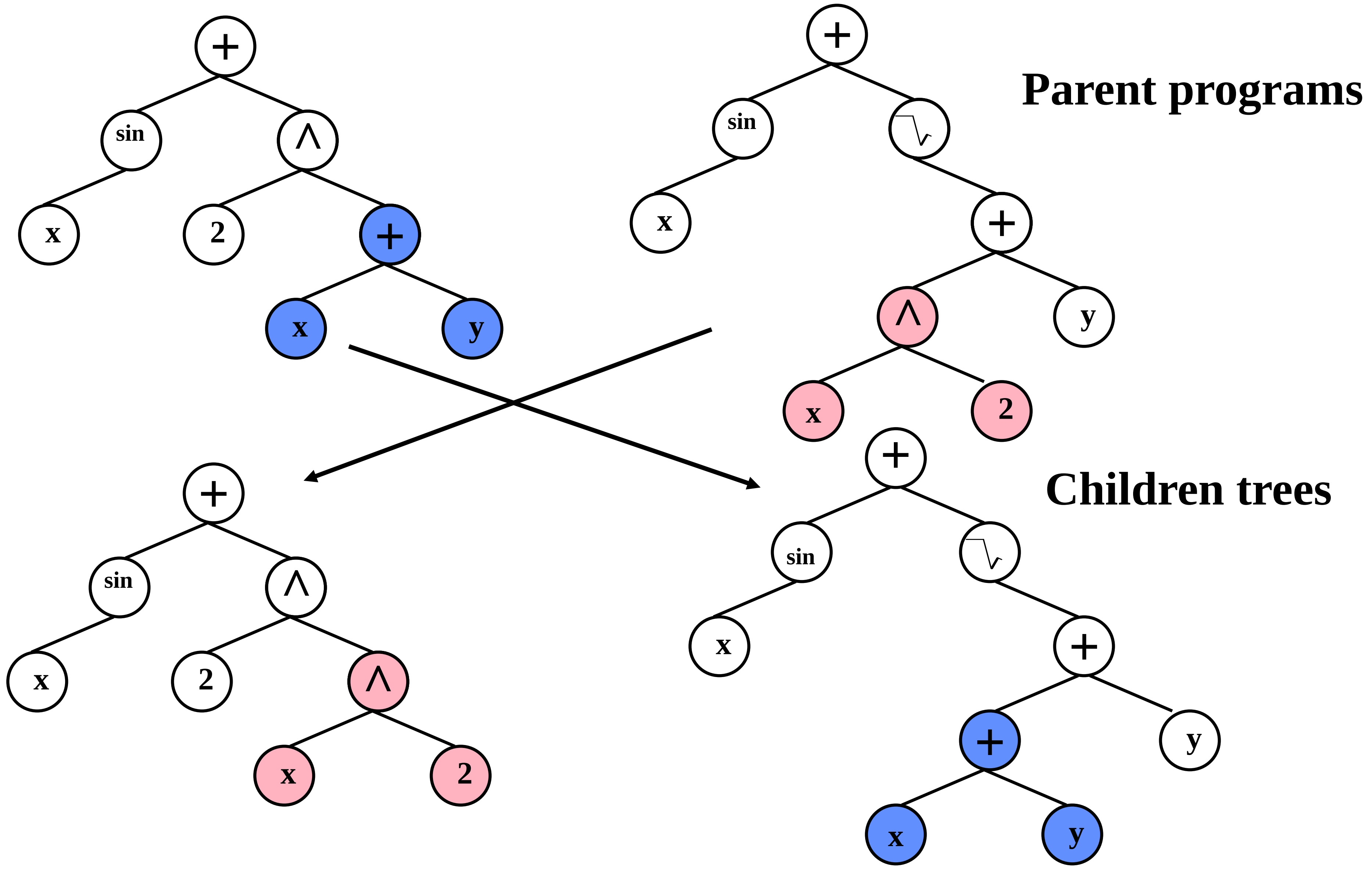
• GP is a form of evolutionary computation in which the individuals in the evolving population are computer programs.

• On each iteration it produces new generation of individuals using selection, crossover and mutation.

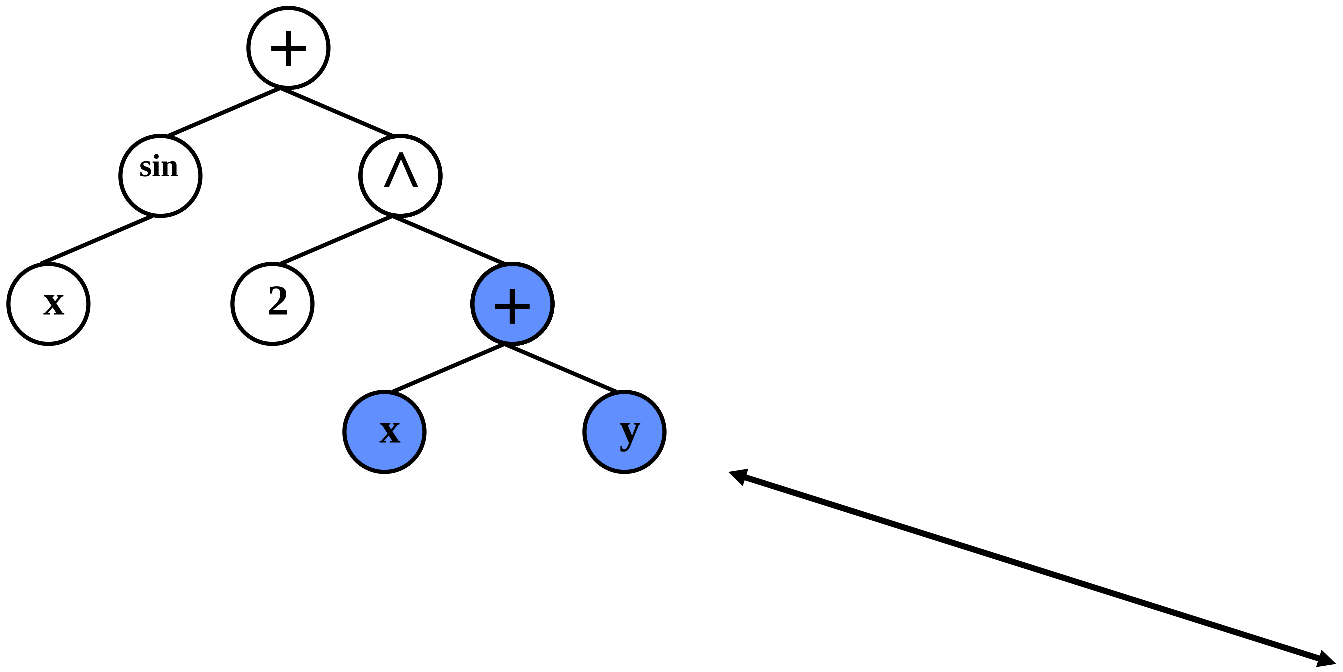
•The fitness of a given individual program in the population is determined by executing the program on a set of training data.

• Crossover operations are performed by replacing a randomly chosen subtree of one parent program by a subtree from the other parent program.

Genetic programming: Example.



Genetic programming: Example.



$\sin(x) + 2^{(x+y)}$

6. Applications of GA

They are commonly applied to optimisation/modelling problems outside machine learning:

- biology: simulation of the evolution of single-celled organism populations.
- computer science: clustering algorithm, adaptive document description.
- engineering: steady-state and transient optimisation of gas pipe line, VLSI circuit layout, VLSI layout compaction.
- operations research: bin packing and graph colouring problems, classroom scheduling, communication network link size optimisation.
- image processing and pattern recognition: searching for image feature detector, explicit pattern class recognition using partial matching.
- social science: GA-like adaptation in model of prehistoric hunter-gatherer behaviour, calibration of population migration model using GA search.

Applications of GA: bankruptcy prediction

- A paper that describes an application of genetic programming:

  Bankruptcy theory development and classification via genetic programming by T. Lensberg, A. Eilifsen and T.E. Mc Kee, European Journal of Operational Research, 169 (2006), pp. 677-697.

  - 1136 companies were analysed (538 bankrupt, 568 non-bankrupt companies)
  - 900 training samples, 236 test (validation) samples
  - variable selection reduces the number of variables from 28 to 6:

    V_1. audit opinion

    V_2. firm size

    V_3. cash position II

    V_4. age I

    V_5. age II

    V_6. interest paying ability

6. Applications of GA: bankruptcy prediction

Machine code Genetic Program:

Program		A		B
	1	$R_0 = X_1$		$R_0 = X_2$
	2	$R_1 = R_0$		$R_0 = R_0 - 2$
	3	$R_1 = R_1 * R_0$		$R_0 = R_0 * R_0$
	4	$R_0 = R_0/3$	$\Leftrightarrow$	$R_0 = R_0 + 1$
	5	Return R_0	$\Leftrightarrow$	$R_0 = R_1/R_0$
	6		$\Leftrightarrow$	Return R_0
Return value		$X_1/3$		0

Applications of GA: bankruptcy prediction

Machine code Genetic Program – after crossover :

Program		A		B
	1	$R_0 = X_1$		$R_0 = X_2$
	2	$R_1 = R_0$		$R_0 = R_0 - 2$
	3	$R_1 = R_1 * R_0$		$R_0 = R_0 * R_0$
	4	$R_0 = R_0 + 1$	$\Leftrightarrow$	$R_0 = R_0/3$
	5	$R_0 = R_1/R_0$	$\Leftrightarrow$	Return R_0
	6	Return R_0	$\Leftrightarrow$	
Return value		X_1^2/(X_1+1)		(X_2-2)^2/3

Applications of GA: bankruptcy prediction

Fitness function: $d(y_i - b_i) = (y_i - b_i)^2 + 0.25 \times |y_i - C(b_i)|$

where

- $y_i = 1$ (bankrupt), $y_i = 0$ (non-bankrupt)
- b_i: predicted output from the model
- $C(b_i) = 1$ if $b_i > ½$, 0 otherwise

- Accuracy for 500 models:

  training samples: 0.818 ± 0.386

  validation samples: 0.805 ± 0.397

  all samples: 0.815 ± 0.388

- Accuracy for one selected model:

  training samples: 0.809

  validation samples: 0.817

  all samples: 0.815

Applications of GA: bankruptcy prediction

- 2,020,000 (+10,100,000) tournaments to select 500 models.
- One selected model:

$$F(V_1,V_2,V_3,V_4,V_5,V_6) = X^2/(X^2 + Y)$$

where

- $X = 2 + (V_1*V_2) – V_5*(3*V_2 – 5) + 2 * f(V_1,V_4,V_5,V_6)*(V_2 – 2)$
- $Y = (V_2)^2 * (1 + V_3)^2 * V_4$
- $F(V_1,V_2,V_5,V_6) = x^2 /(x^2 + V_6*x + y)$
- $x = 4 + V_5*(2*V_1*V_6 – 1)$
- $y = (V_4 * V_6^2)/2$

Applications of GA: bankruptcy prediction

- 2,020,000 (+10,100,000) tournaments to select 500 models.
- One selected model:

$$F(V_1,V_2,V_3,V_4,V_5,V_6) = X^2/(X^2 + Y)$$

where

- $X = 2 + (V_1*V_2) - V_5*(3*V_2 - 5) + 2 * f(V_1,V_4,V_5,V_6)*(V_2 - 2)$
- $Y = (V_2)^2 * (1 + V_3)^2 * V_4$
- $F(V_1,V_2,V_5,V_6) = x^2 /(x^2 + V_6*x + y)$
- $x = 4 + V_5*(2*V_1*V_6 - 1)$
- $y = (V_4 * V_6^2)/2$

Applications of GA: predicting customer default

- A paper that describes the application of GP for predicting customer default:

An artificial intelligence system for predicting customer default in e-commerce by L. Vanneschi, D.M. Horn, M. Castelli and A Popovic, Expert Systems with Applications, 104 (2018), pp. 1-21.

Default payment: customer fails to settle a bill within 90 days upon receipt.

Credit scoring is employed to identify customers' default probability.

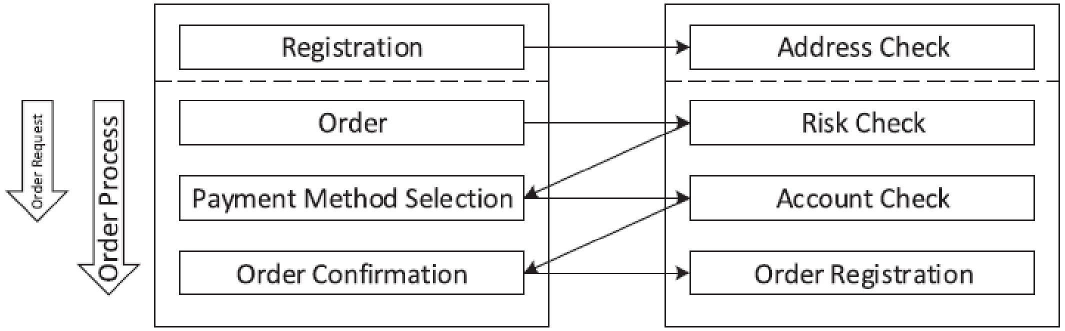


Fig. 4. RSS risk management services.

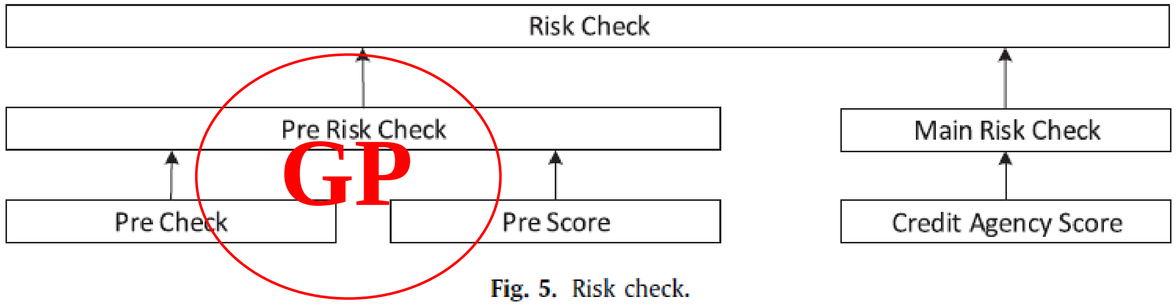


Fig. 5. Risk check.

Applications of GA: predicting customer default

- Data samples:

  15,535 bad orders (≈27.4%) , 41,134 good orders (≈72.6%) .

- Attributes:

  IN0: binary valued, whether the customer is known to the client

  IN1: binary valued, whether shipping address and billing address match

  IN2: elapsed time in days since customer was registered by the system for the first time.

  IN3: validation of billing address, whether the address exists and customer lives at stated address

  IN4, IN5, IN10, IN11, IN13, IN14, IN15, IN18: the customer’s order history

  IN6, IN7, IN8: information about the dunning history of the customer

  IN9: discretized order value in relation to the clients’ average order value

  IN12: variable designed as means of fraud prevention.

  IN16: clock hours of order time

  IN17: age of customer in years.

Applications of GA: predicting customer default

Some ideas:

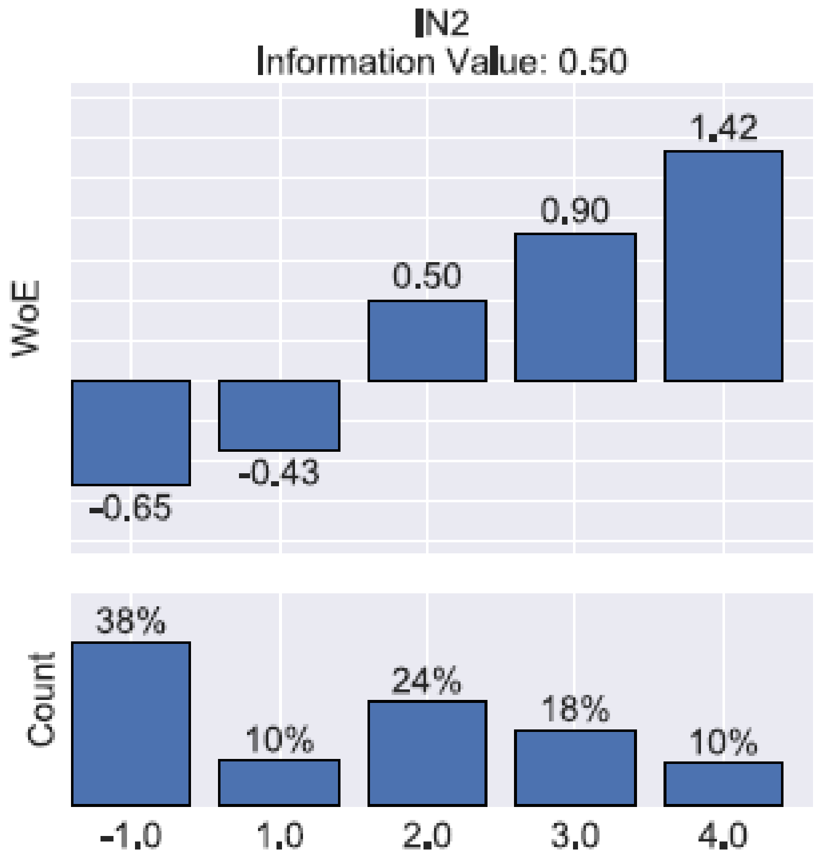
- Weight of evidence (WoE) measure to assess the predictive power of an attribute:

  WoE = [ln(Distribution Goods/Distribution Bads)] × 100
- The values of a continuous attribute are partitioned into bins.
- A bin with WoE around 0 has the same probability of observing a default payment as the sample average.
- A bin with WoE above or below 0, the probability of observing a default payment is higher/lower than sample average.
- Compute Information Value (IV):

$$\mathrm{IV} = \sum_i (\text{Distribution Goods}_i - \text{Distribution Bads}_i) \times \ln\left(\frac{\text{Distribution Goods}_i}{\text{Distribution Bads}_i}\right)$$

Applications of GA: predicting customer default

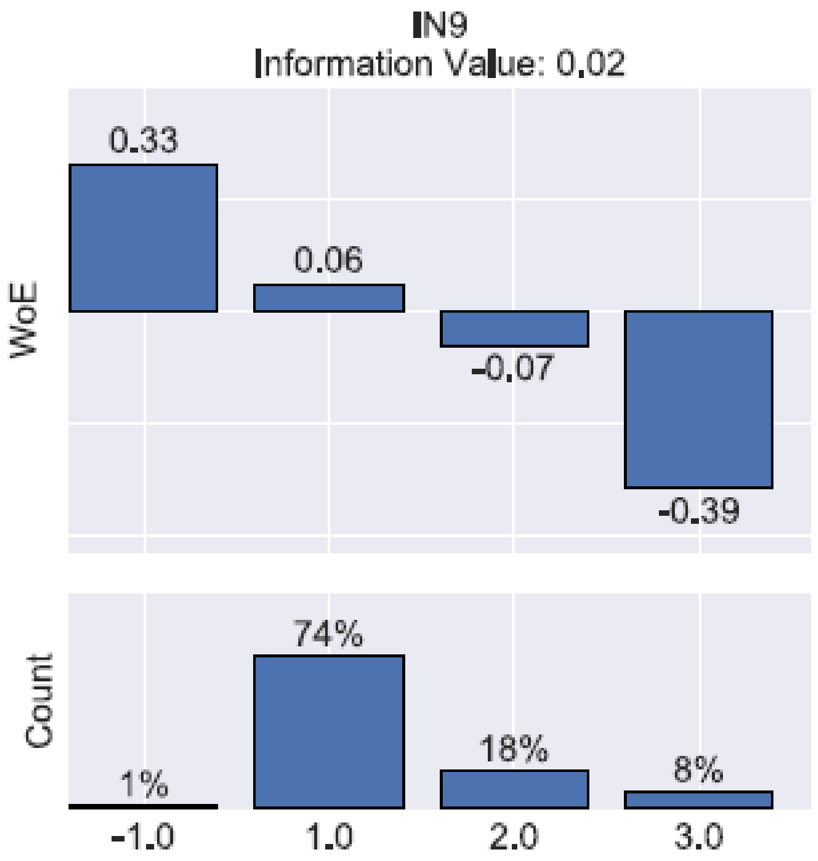
Feature selection:



- IN2 is a continuous variable (elapsed time since the customer was registered with the system)
- There are 5 bins:
  - Missing values
  - [-∞,90)
  - [90,380)
  - [380,600)
  - [600,∞)
- This variable is expected to be useful for classification (IV > 0.05)

Applications of GA: predicting customer default

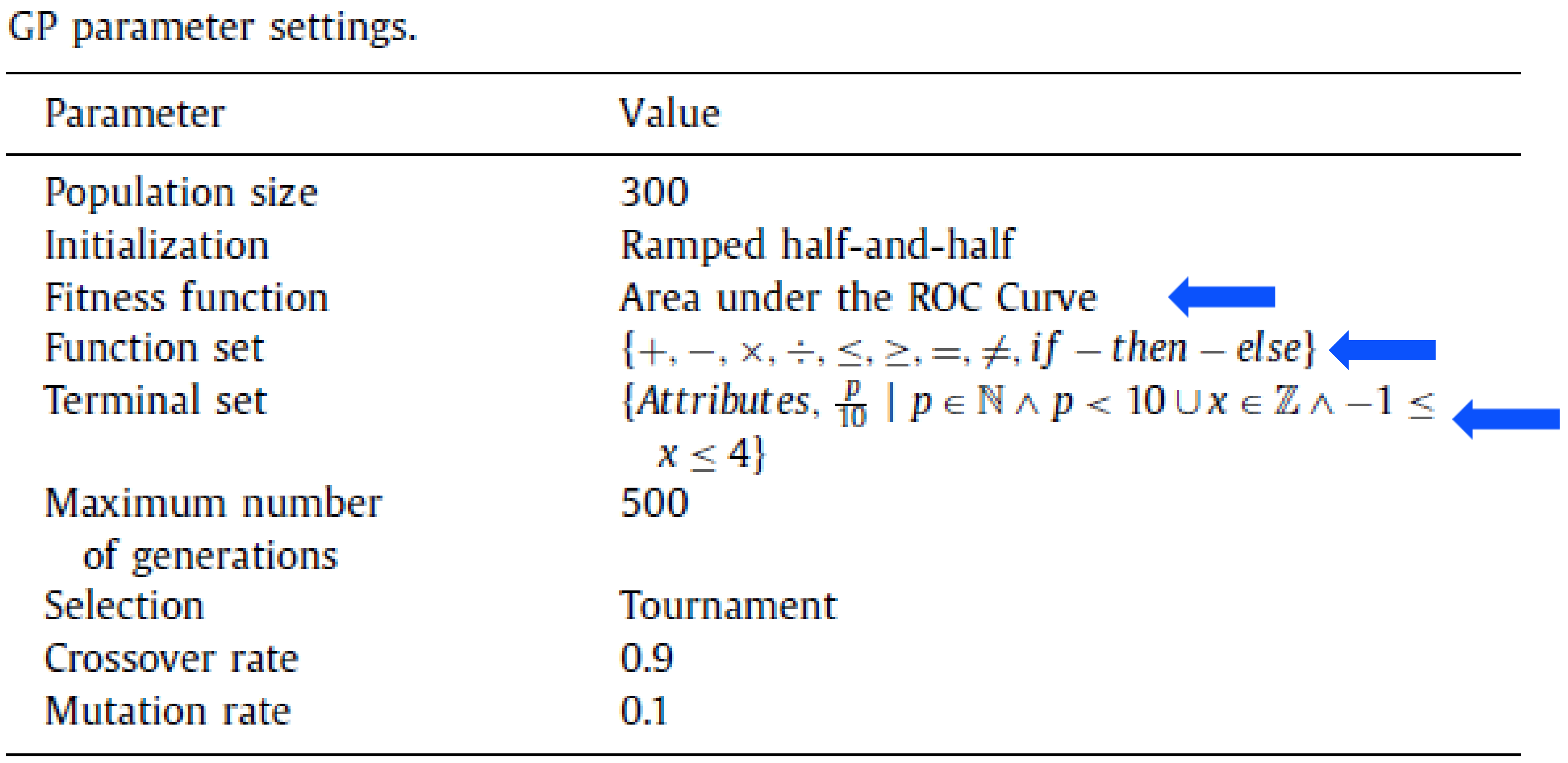
Feature selection:



- IN9 is a continuous variable (the clients' average order value)
- There are 4 bins:
  - Missing values
  - [-∞,150)
  - [150,300)
  - [300,∞)
- This variable is expected not to be useful for classification (IV $\leq$ 0.05)

Applications of GA: predicting customer default

GP parameter setting:

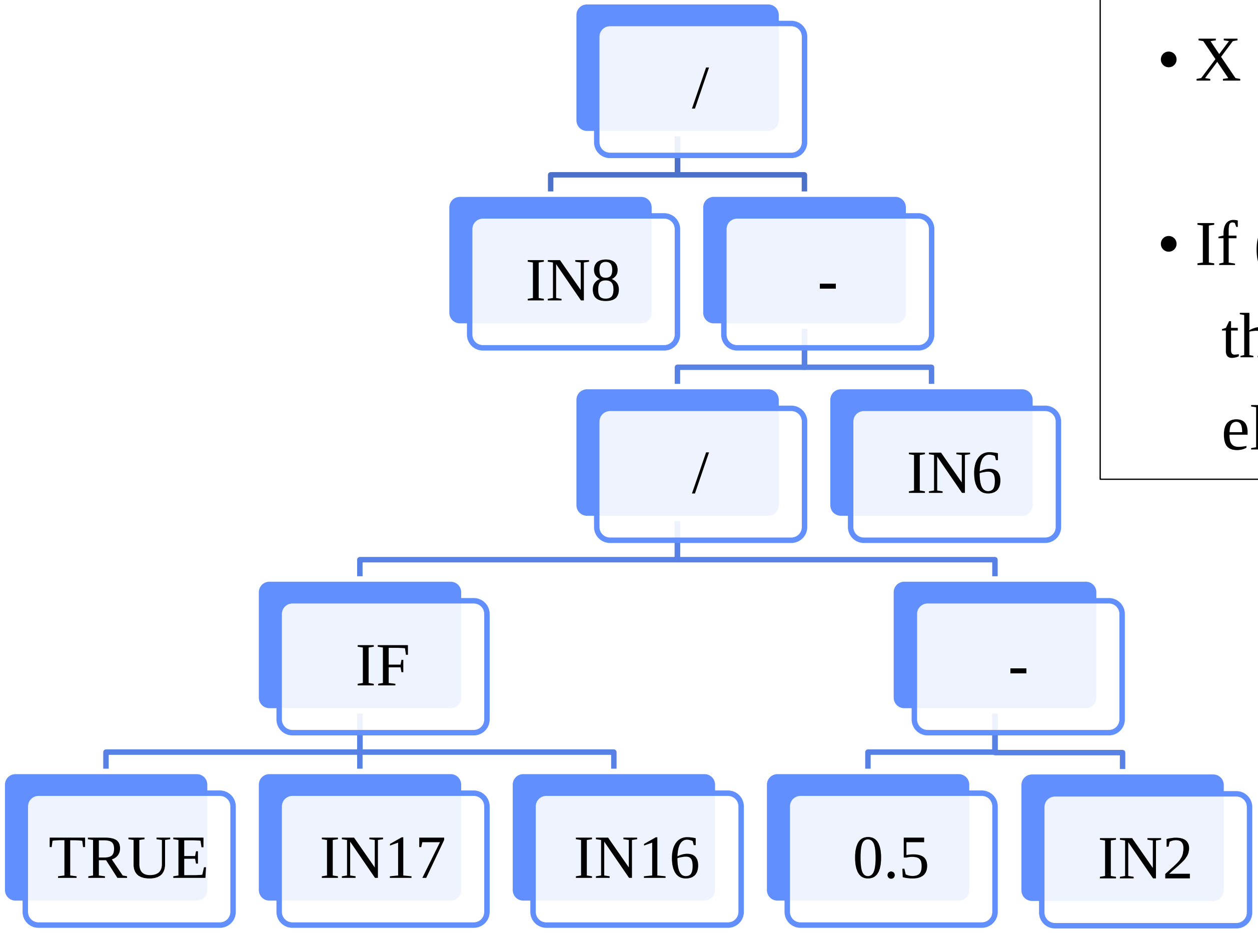


GP parameter settings.

Parameter	Value
Population size	300
Initialization	Ramped half-and-half
Fitness function	Area under the ROC Curve
Function set	$\{+, -, \times, \div, \leq, \geq, =, \neq, if - then - else\}$
Terminal set	$\{Attributes, \frac{p}{10} \mid p \in \mathbb{N} \wedge p < 10 \cup x \in \mathbb{Z} \wedge -1 \leq x \leq 4\}$
Maximum number of generations	500
Selection	Tournament
Crossover rate	0.9
Mutation rate	0.1

Applications of GA: predicting customer default

GP model (a subtree):



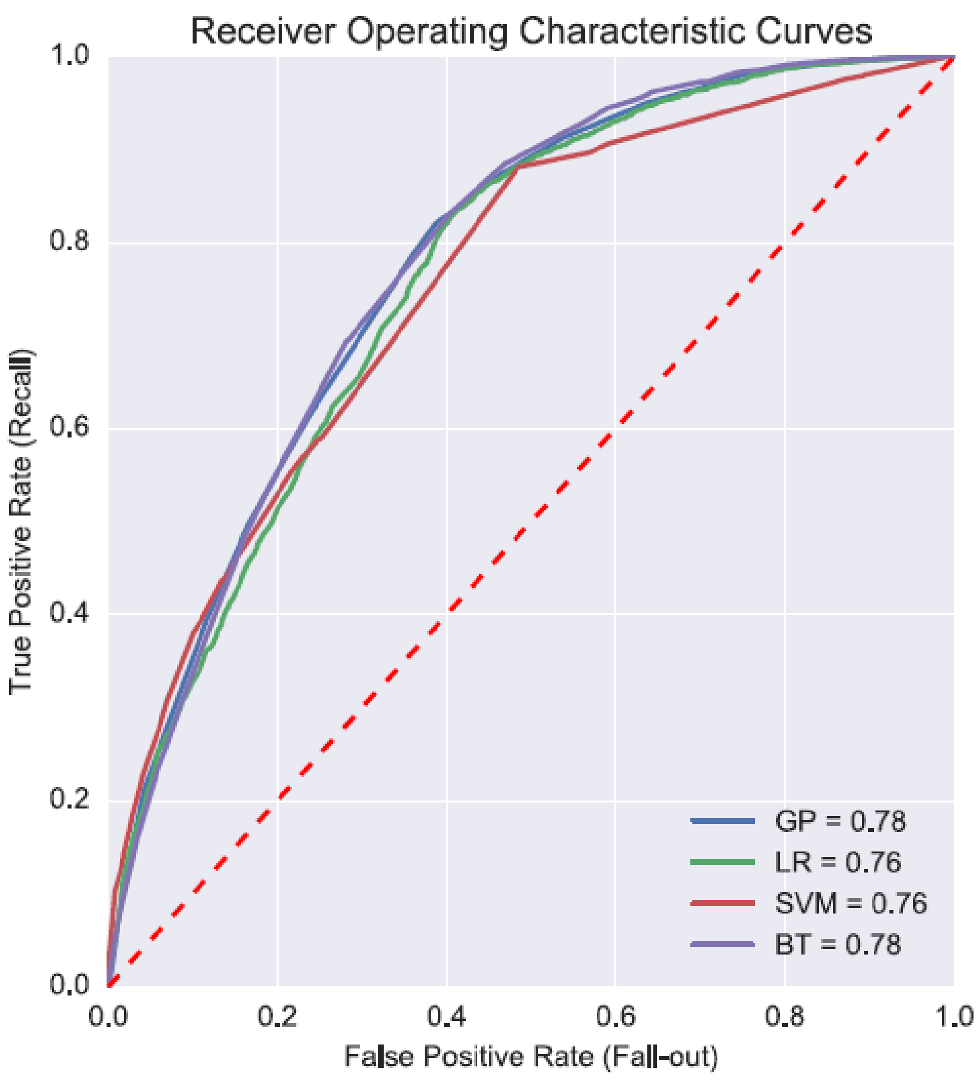
- IN8/ [X –IN16]
- X = IN17/(0.5 – IN2)

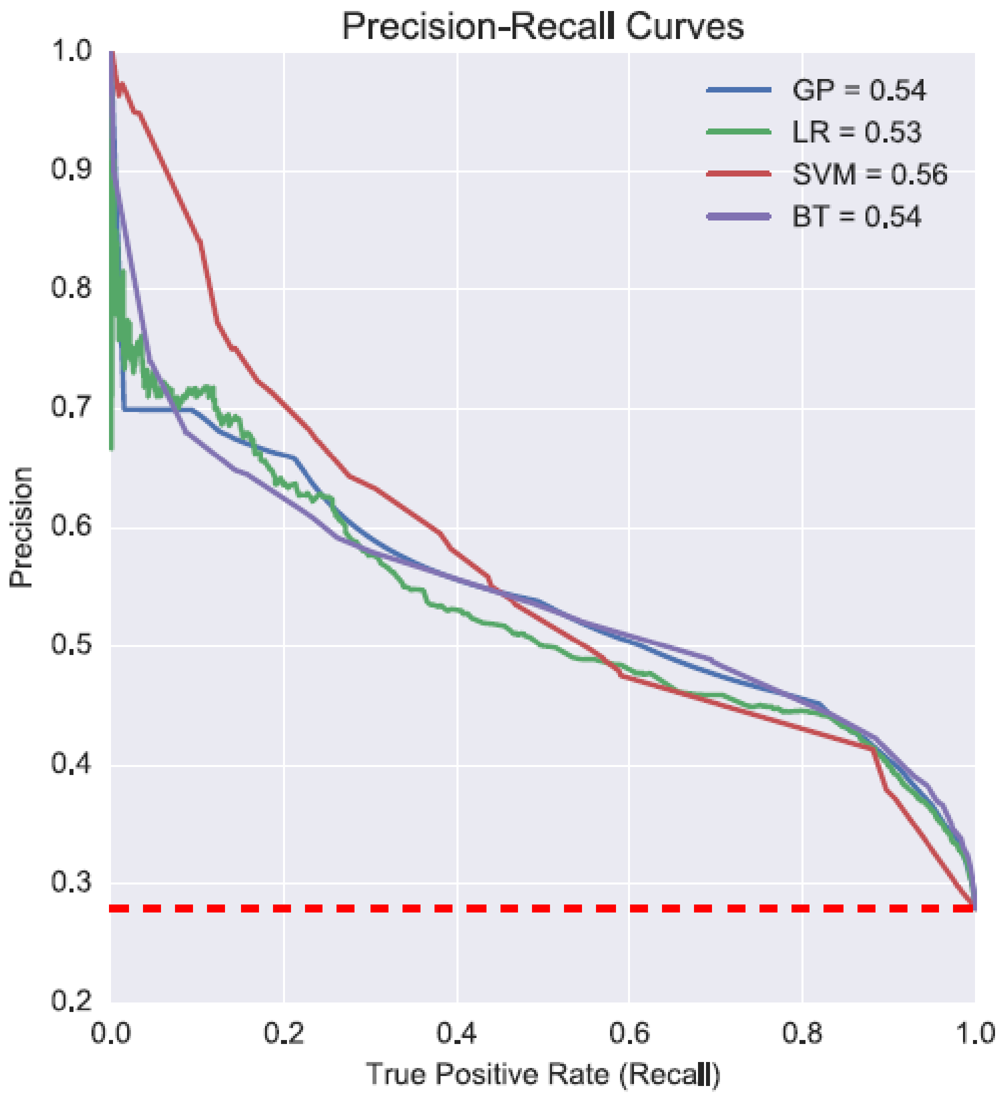
- If (condition = TRUE),
  then FIRST node,
  else SECOND node.

Positive output: Class 1
Other output: Class 2

Applications of GA: predicting customer default

Results:





Reference.

- T. M. Mitchell, Machine Learning, 1997, McGraw Hill.
- D. Goldberg, Genetic algorithm in search, optimization, and machine learning, 1989, Addison-Wesley.
- An excellent website for introduction to genetic algorithm:

  http://www.obitko.com/tutorials/genetic-algorithms