technische universiteit eindhoven

Genetic Algorithms

Genetic Algorithms provide an approach to learning based loosely on simulated evolution

a.j.m.m. (ton) weijters

Genetic Algorithm (GA)

- The search for an appropriate hypothesis begins with a population of initial *hypotheses strings*.
- Members of the current population gives rise to the next generation by means of operations such as *crossover* and *mutation*.
- At each step, the hypotheses in the current population are evaluated by a *fitness function*.
- The most fit hypotheses are selected probabilistically for producing the next generation.

Example Applications

- Planning
- Scheduling
- Optimization

General Characterization

- Searching for an optimal solution is difficult
  - large search space
  - simple more traditional algorithm not available
- Measurement of the quality of a given solution is relative simple
- Local optimization versus local optimization

Example GA; rule induction

- Motivation: Decision trees have sometimes problems with finding combinations of informative features.
- How to present rules (sets)
- Quality measurement

  – We don't search in one big step for THE rule set, but search step by step for good rules! Remove the cases covered by a rule out of the learning material and start searching for a next rule!

Illustration:

(10 learning examples):

Hair	Length	Weight	Suntan cream	Burned
blond	medium	light	yes	no
blond	medium	light	no	yes
red	long	light	yes	yes
brown	medium	heavy	yes	no
blond	long	medium	yes	no
brown	long	light	no	no
red	small	heavy	no	yes
brown	long	light	yes	no
blond	medium	heavy	no	yes
brown	small	heavy	no	no

New (test) examples:

red	medium	light	yes	yes
blond	medium	medium	no	yes
brown	small l	light	yes	nee

Representing hypotheses

- **Assume we are looking to rules for the burning-example:**
- **hair: red, blond, brown**
  length: short, medium, long
  weight: light, medium, heavy
  suntan cream: yes, no
  burning: yes, no
- **IF hair=blond AND suntan_cream=yes THEN burning=no**
- **010 111 111 10 01**

Genetic operators

single-point crossover:

1110100101110 → 1110010111101

0001010111101 → 0001100101110

two-point crossover:

1110100101110 → 1110010101110

0001010111101 → 0001100111101

point mutation:

0001100101110 0001100111110

Fitness Function

- We are interested in population elements (rules) that are accurate and are supported by many examples.
- Example fitness function for a classification rule: *Nc/N+1* with
  - *Nc* the number of correct classified cases
  - *N* is the number of cases covered by the rule

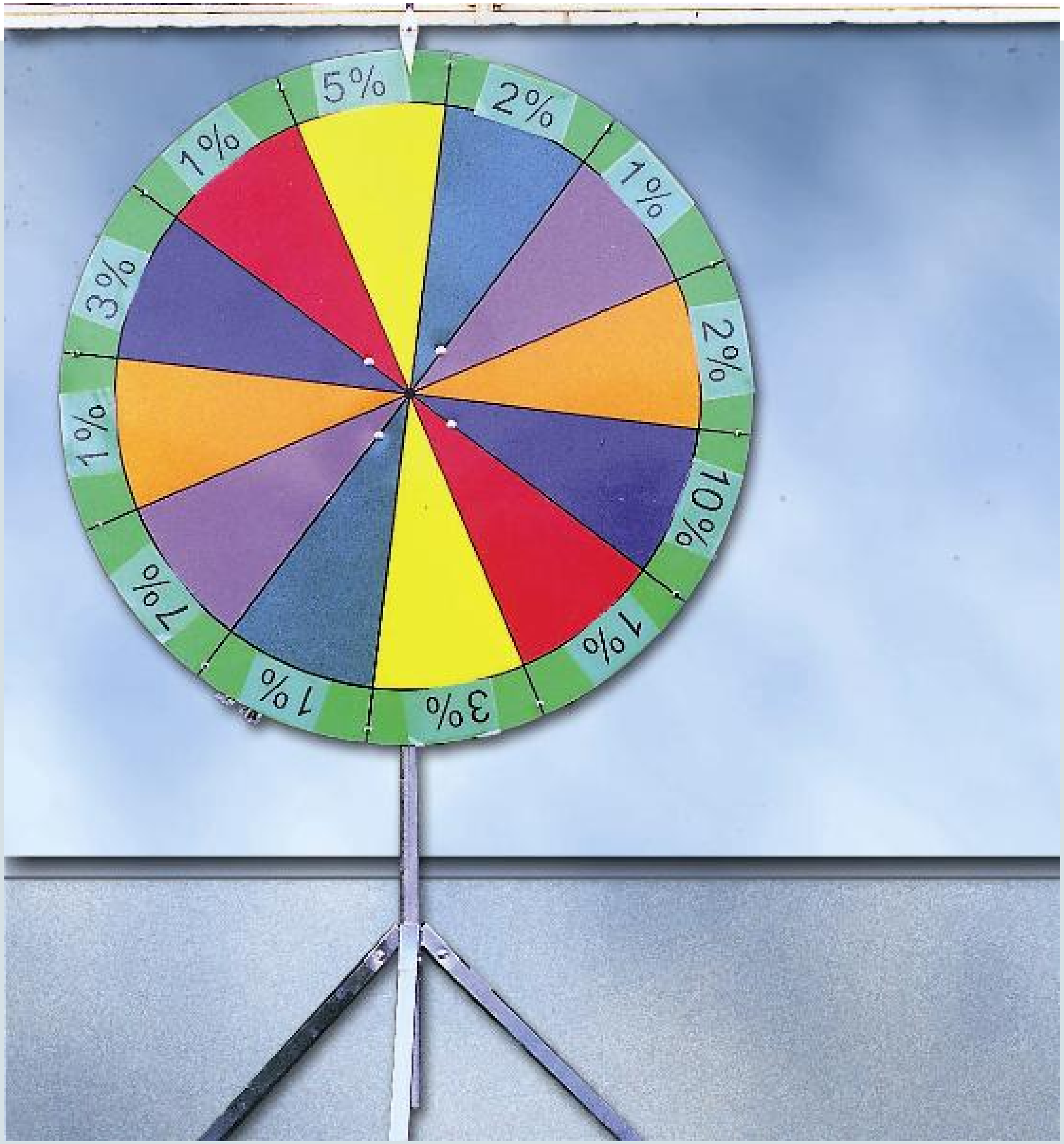
Examples (*Nc/N+1*)

- 4/5 rule (covers 5 examples out of which it classifies 4 correctly) = 4/5+1 = 0.667
- 40/50 rule (covers 50 examples out of which it classifies 40 correctly) = 40/50+1 = 0.784

This is what we want because the 4/5 rule is based on less data then the 40/50 rule.

The fitness function defines the criterion for probabilistically selecting a hypothesis for inclusion in the next generation. For example:

$$P(h_i) = Fitness\,(h_i) \,/ \sum_{j=1}^{p} Fitness\,(h_j)$$


5%
2%
1%
2%
10%
1%
3%
1%
7%
1%
3%
1%

Prototypical genetic algorithm

- Generate P with 500 random hypotheses
- Calculate the fitness of all 500 members
- Repeat while max-fitness<threshold:
  - Select probabilistically 200 members of P (high fitness high chance)
  - Apply the **crossover** operator to the 200 members
  - Choose one example of the 200 new members and apply **mutation**
  - Update P (300 most fit elements + 200 new)
  - Calculate the **fitness** of the new members

Conclusions

- Genetic algorithms can be viewed as general optimization method for searching a large solution space.
- Although not guaranteed to find an optimal solution, GAs has been successfully applied to a number of optimization problems.

Demo: hospital-planning

```
Classes 1 2 3_5 6_9 10_  (5) s
diagnose D1 D2 D3 D4 D5  (5) s
geslacht M V  (2) s
leeftijd continuous 0 100  (10) i
verzekerin Z P  (2) s
Class =  1#4 P=  0.01
Class =  2#6 P=  0.01
Class =  3#389 P=  0.52
Class =  4#301 P=  0.40
Class =  5#50 P=  0.07
Default class = 3
# examples: 750
Stem of the training and test data:
C:\Data\delphi\geseco\Opnamepl
```

```
UseDefault = TRUE
Seed = 1
BitString1Chance: 0.50
Number of rules in the population: 500
Maximum number of generations: 500
Next generation if max fitness is # times
equal: 25
Covering Weight: 0.00
RuleReliabilityThres <: 40.00
```

New random population 1
Generation 43
R1: 000100111011111111100010 OK=143 Match=143
Total performance: (143/#143) 19.07%
Default class=3 (P=0.64)

New random population 2
Generation 40
R2: 010001001110111111100010 OK=43 Match=43
Total performance: (186/#186) 24.80%
Default class=3 (P=0.69)

```
IF                                   (R1 143/143)
diagnose=D4
geslacht=V
THEN class=6_9

IF                                   (R2 43/43)
diagnose=D2
geslacht=M
leeftijd in[10..40][50..100]
THEN class=6_9
```

```
Performance: (650/#729) #examples=750
Score=86.67%

Confusion matrix:

-------- target classification -------->
classified as      1     2     3     4     5
            1      0     0     0     0     0
            2      0     0     0     0     0
            3      4     3   361    24     7
            4      0     0    18   251     4
            5      0     0     0    19    38
```

```
Test Performance: (202/#241) #examples=250
Score=80.80%

Confusion matrix:

-------- target classification -------->
classified as      1     2     3     4     5
            1      0     0     0     0     0
            2      0     0     0     0     0
            3      4     0   127    13     4
            4      0     0     7    62     5
            5      0     0     0     6    13
```