Induction of Rules

JERZY STEFANOWSKI
Institute of Computing Sciences
Poznań University of Technology

Doctoral School , Catania-Troina, April, 2008

Outline of this lecture

1. Rule representation
2. Various algorithms for rule induction.
3. MODLEM $\rightarrow$ exemplary algorithm for inducing a minimal set of rules.
4. Classification strategies
5. Descriptive properties of rules.
6. Explore $\rightarrow$ discovering a richer set of rules.
7. Association rules
8. Logical relations
9. Final remarks.

Rules - preliminaries

- **Rules** → popular symbolic representation of knowledge derived from data;
  - Natural and easy form of representation → possible inspection by human and their interpretation.
- Standard form of rules

  IF *Conditions* THEN *Class*
- Other forms: Class IF Conditions; Conditions → Class

Example: The set of decision rules induced from PlaySport:

if outlook = overcast **then** Play = yes

if temperature = mild **and** humidity = normal **then** Play = yes

if outlook = rainy **and** windy = FALSE **then** Play = yes

if humidity = normal **and** windy = FALSE **then** Play = yes

if outlook = sunny **and** humidity = high **then** Play = no

if outlook = rainy **and** windy = TRUE **then** Play = no

Rules – more preliminaries

- A set of rules – a disjunctive set of conjunctive rules.
- Also DNF form:
  - *Class* IF *Cond*_1 OR *Cond*_2 OR … *Cond*_m
- Various types of rules in data mining
  - Decision / classification rules
  - Association rules
  - Logic formulas (ILP)
  - Other → action rules, …
- **MCDA** → attributes with some additional preferential information and ordinal classes.

Why Decision Rules?

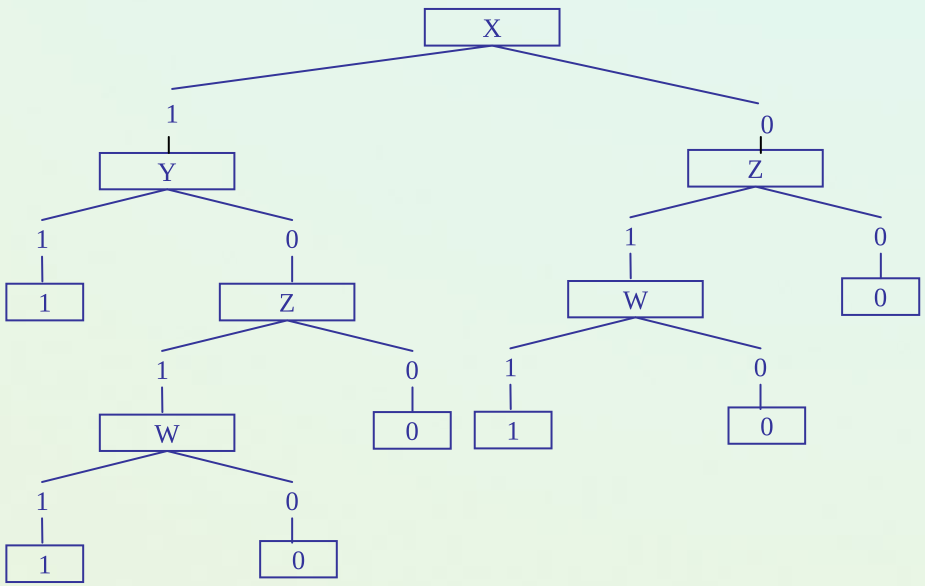
- Decision rules are more compact.
- Decision rules are more understandable and natural for human.
- Better for descriptive perspective in data mining.
- Can be nicely combined with background knowledge and more advanced operations, …

Example: Let $X \in \{0,1\}$, $Y \in \{0,1\}$, $Z \in \{0,1\}$, $W \in \{0,1\}$. The rules are:

if X=1 and Y=1 **then** 1

if Z=1 and W=1 **then** 1

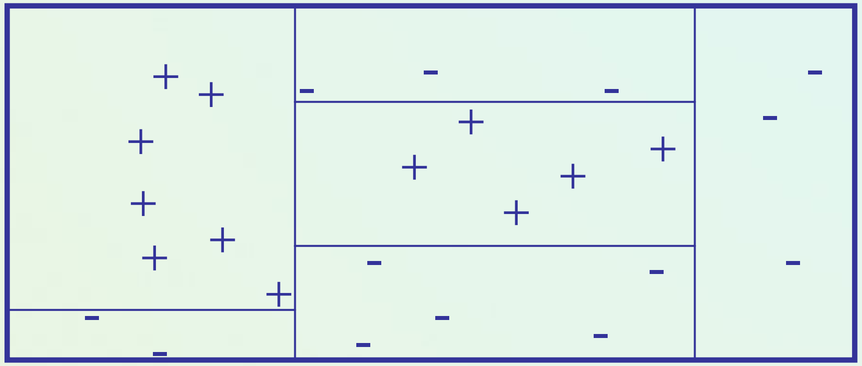
Otherwise 0;



Decision rules vs. decision trees:

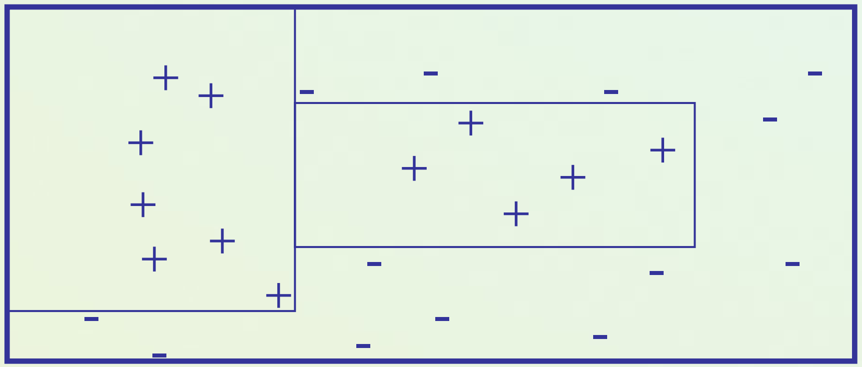
- Trees – splitting the data space (e.g. C4.5)

Decision boundaries of decision trees



- Rules – covering parts of the space (AQ, CN2, LEM)

Decision boundaries of decision rules



Rules – more formal notations

- A rule corresponding to class K_j is represented as

  if P then Q

  where $P = w_1$ and w_2 and … and w_m is a condition part and Q is a decision part (object x satisfying P is assigned to class K_j)

- Elementary condition w_i (a *rel* v), where $a \in A$ and v is its value (or a set of values) and *rel* stands for an operator as $=, <, \leq, \geq, >$.

- [P] is a cover of a condition part of a rule $\rightarrow$ a subset of examples satisfying P.

  - *if* ($a2$ = small) *and* ($a3 \leq 2$) *then* (d = C1) $\{x1, x7\}$

Rules - properties

- B $\rightarrow$ a set of examples from K_j.
- A rule if P then Q is discriminant in *DT* iff
  $[P]=\bigcap [w_i] \subseteq B$,
- otherwise ($P \cap B \neq \emptyset$) the rule is partly discriminating
  - Rule accuracy (or confidence) $|[P \cap K]|/|[P]|$
- Rule cannot not have a redundant condition part,
  i.e. there is no other $P^* \subset P$ such that $[P^*] \subseteq B$.

- Rule sets induced from DT
  - Minimal set of rules
  - Other sets of rules (all rules, satisfactory)

An example of rules induced from data table

Minimal set of rules

- *if* ($a2$ = s) ∧ ($a3$ ≤ 2) *then* (d = C1) {$x1$,$x7$}
- *if* ($a2$ = n) ∧ ($a4$ = c) *then* (d = C1) {$x3$,$x4$}
- *if* ($a2$ = w) *then* (d = C2) {$x2$,$x6$}
- *if* ($a1$ = f) ∧ ($a4$ = a) *then* (d = C2) {$x5$,$x8$}

Partly discriminating rule:

- *if* (*a1*=m) *then* (*d*=C1) {*x1*,*x3*,*x7* | *x6*} 3/4

id.	a_1	a_2	a_3	a_4	d
x_1	m	s	1	a	C1
x_2	f	w	1	b	C2
x_3	m	n	3	c	C1
x_4	f	n	2	c	C1
x_5	f	n	2	a	C2
x_6	m	w	2	c	C2
x_7	m	s	2	b	C1
x_8	f	s	3	a	C2

How to learn decision rules?

- Typical algorithms based on the scheme of a sequential covering and heuristically generate a minimal set of rule covering examples:
  - see, e.g., AQ, CN2, LEM, PRISM, MODLEM, Other ideas – PVM, R1 and RIPPER).
- Other approaches to induce „richer” sets of rules:
  - Satisfying some requirements (Explore, BRUTE, or modification of association rules, „Apriori-like”).
  - Based on local „reducts” $\rightarrow$ boolean reasoning or LDA.
- Specific optimization, eg. genetic approaches.
- Transformations of other representations:
  - Trees $\rightarrow$ rules.
  - Construction of (fuzzy) rules from ANN.

Covering algorithms

- A strategy for generating a rule set directly from data:
  - for each class in turn find a rule set that covers all examples in it (excluding examples not in the class).
- The main procedure is iteratively repeated for each class.
  - Positive examples from this class vs. negative examples.
- This approach is called a *covering* approach because at each stage a rule is identified that covers some of the instances.
- A sequential approach.
- For a given class it conducts in a stepwise way a general to specific search for the best rules (learn-one-rule) guided by the evaluation measures.

Original covering idea (AQ, Michalski 1969, 86)

for each class Ki **do**

Ei := Pi U Ni (Pi positive, Ni negative example)

RuleSet(Ki) := empty

repeat {find-set-of-rules}

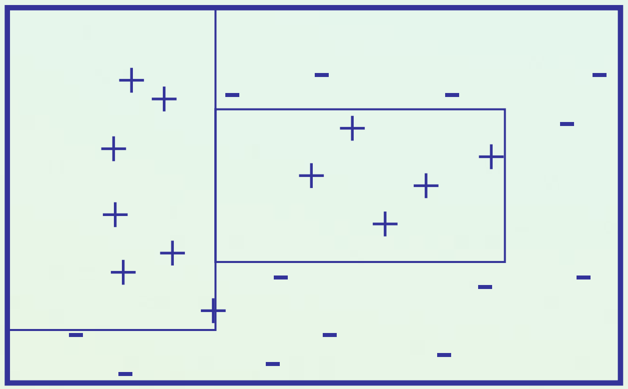
find-one-rule R covering some positive examples

and no negative ones

add R to RuleSet(Ki)

delete from Pi all pos. ex. covered by R

until Pi (set of pos. ex.) = empty



Find one rule:

Choosing a positive example called a seed.

Find a limited set of rules characterizing the seed → **STAR**.

Choose the best rule according to LEF criteria.

Another variant – CN2 algorithm

- Clark and Niblett 1989; Clark and Boswell 1991
- Combine ideas AQ with TDIDT (search as in AQ, additional evaluation criteria or prunning as for TDIDT).
  - AQ depends on a seed example
  - Basic AQ has difficulties with noise handling
    - Latter solved by rule truncation (pos-pruning)
- Principles:
  - Covering approach (but stopping criteria relaxed).
  - Learning one rule – not so much example-seed driven.
  - Two options:
    - Generating an unordered set of rules (First Class, then conditions).
    - Generating an ordered list of rules (find first the best condition part than determine Class).

General schema of inducing minimal set of rules

- The procedure conducts a general to specific (greedy) search for the best rules (**learn-one-rule**) guided by the evaluation measures.
- At each stage add to the current condition part next elementary tests that optimize possible rule’s evaluation (no backtracking).

Procedure Sequential covering (K_j Class; *A* attributes; *E* examples,
τ - acceptance threshold);
begin
  $R := \emptyset$; {set of induced rules}
  r := ***learn-one-rule***(Y_j Class; *A* attributes; *E* examples)
  while ***evaluate***(r,E) > τ **do**
  begin
    $R := R \cup r$;
    $E := E \setminus [R]$; {remove positive examples covered by *R*}
    r := ***learn-one-rule***(*K*j Class; *A* attributes; *E* examples);
  end;
return *R*
end.

The contact lenses data

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	No	Reduced	None
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Reduced	None
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Reduced	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Example: contact lens data 2

- Rule we seek:
- Possible conditions:

```
If ?
  then recommendation = hard
```

Age = Young	2/8
Age = Pre-presbyopic	1/8
Age = Presbyopic	1/8
Spectacle prescription = Myope	3/12
Spectacle prescription = Hypermetrope	1/12
Astigmatism = no	0/12
Astigmatism = yes	4/12
Tear production rate = Reduced	0/12
Tear production rate = Normal	4/12

ACK: slides coming from witten&eibe WEKA

Modified rule and covered data

- Condition part of the rule with the best elementary condition added:

```
If astigmatism = yes
    then recommendation = hard
```

- Examples covered by condition part:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Further specialization, 2

- Current state:

```
If astigmatism = yes
     and ?
   then recommendation = hard
```

- Possible conditions:

Age = Young	2/4
Age = Pre-presbyopic	1/4
Age = Presbyopic	1/4
Spectacle prescription = Myope	3/6
Spectacle prescription = Hypermetrope	1/6
Tear production rate = Reduced	0/6
Tear production rate = Normal	4/6

Two conditions in the rule

- The rule with the next best condition added:

```
If astigmatism = yes
    and tear production rate = normal
  then recommendation = hard
```

- Examples covered by modified rule:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Further refinement, 4

- Current state:

```
If astigmatism = yes
    and tear production rate = normal
    and ?
  then recommendation = hard
```

- Possible conditions:

```
Age = Young                                  2/2
Age = Pre-presbyopic                         1/2
Age = Presbyopic                             1/2
Spectacle prescription = Myope               3/3
Spectacle prescription = Hypermetrope        1/3
```

- Tie between the first and the fourth test
  - We choose the one with greater coverage

The result

- Final rule:

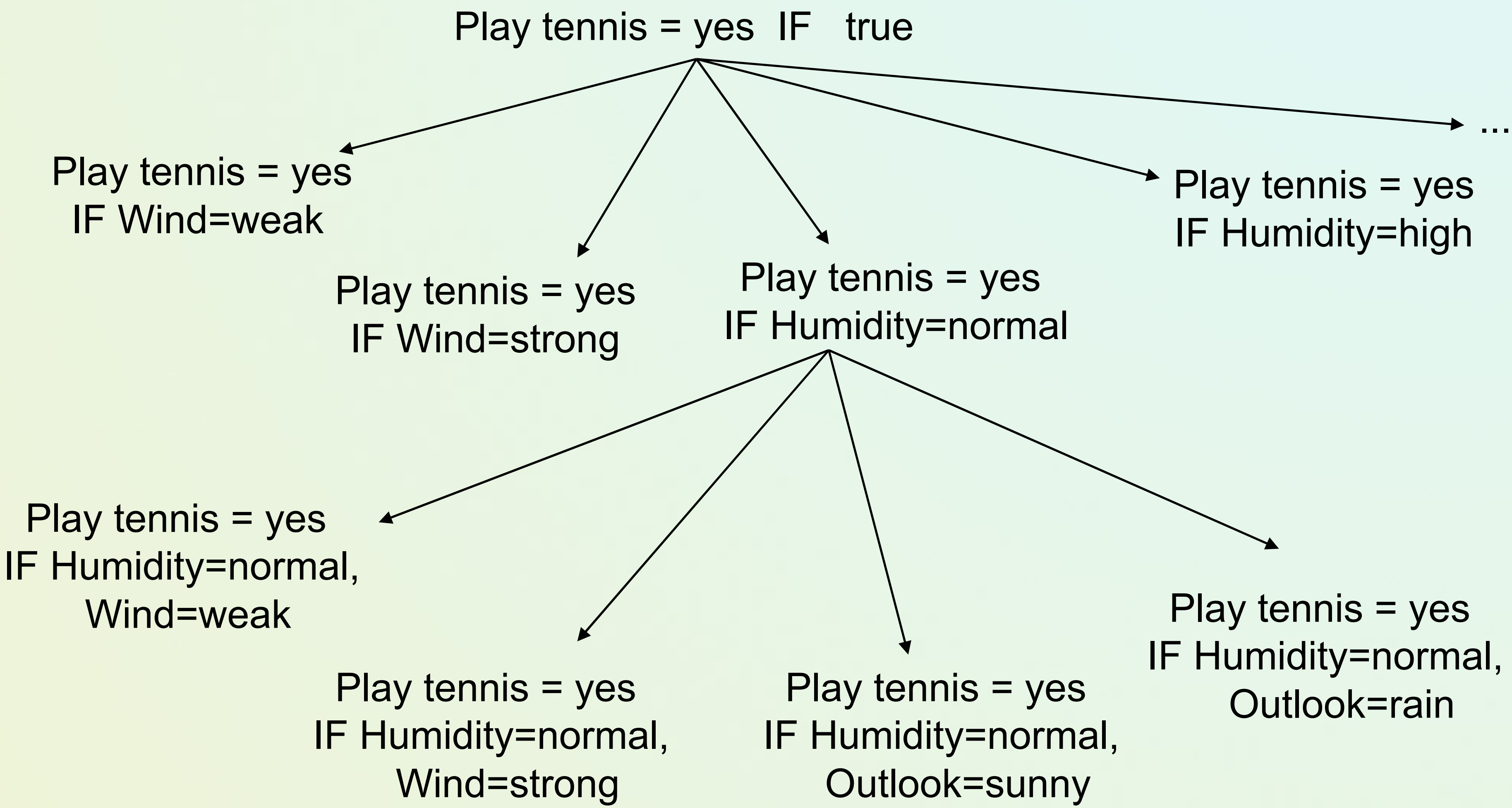
```
If astigmatism = yes
   and tear production rate = normal
   and spectacle prescription = myope
   then recommendation = hard
```

- Second rule for recommending “hard lenses”:
  (built from instances not covered by first rule)

```
If age = young and astigmatism = yes
   and tear production rate = normal
   then recommendation = hard
```

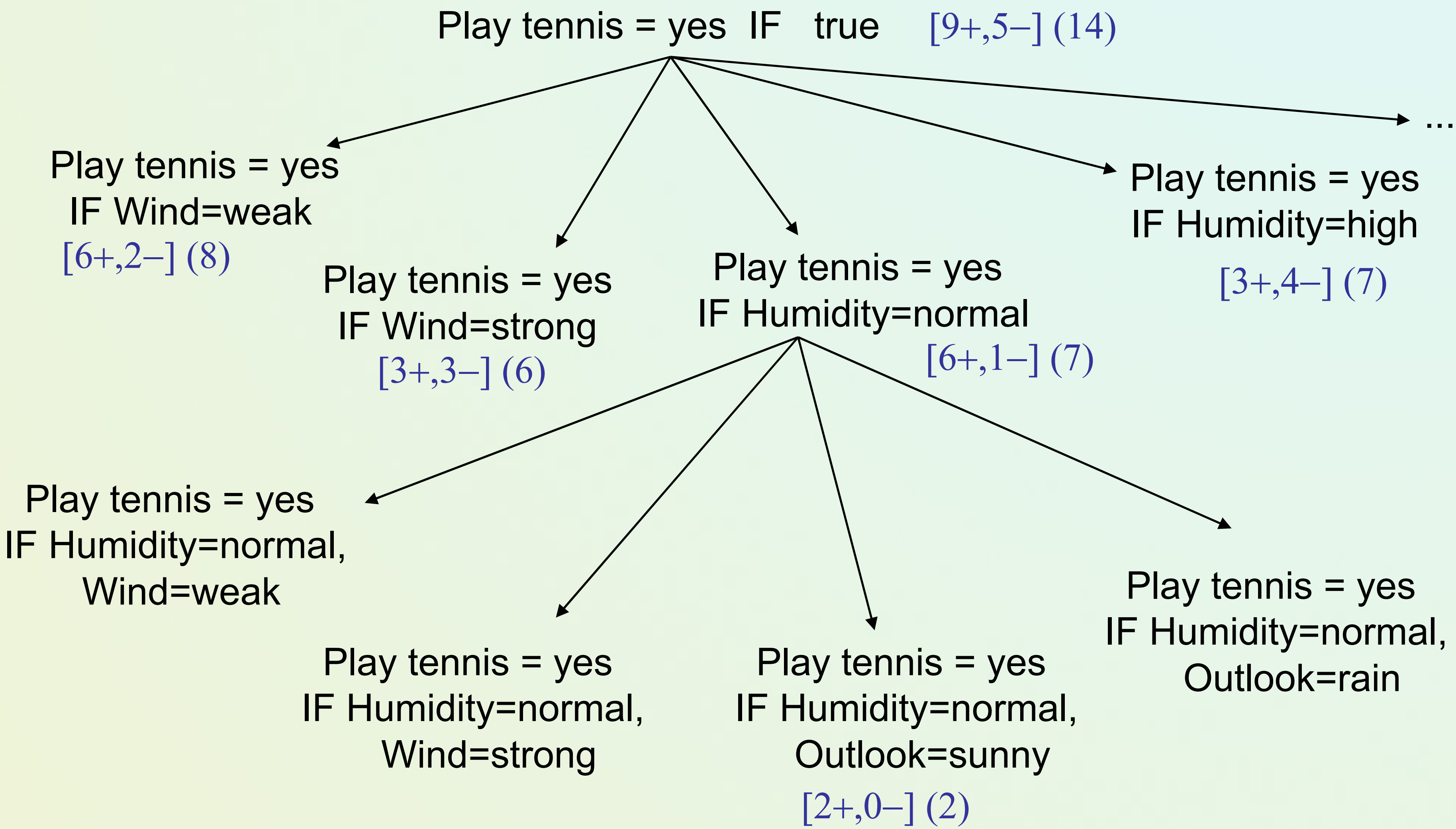
- These two rules cover all “hard lenses”:
  - Process is repeated with other two classes

Learn-one-rule as search (Play sport data)



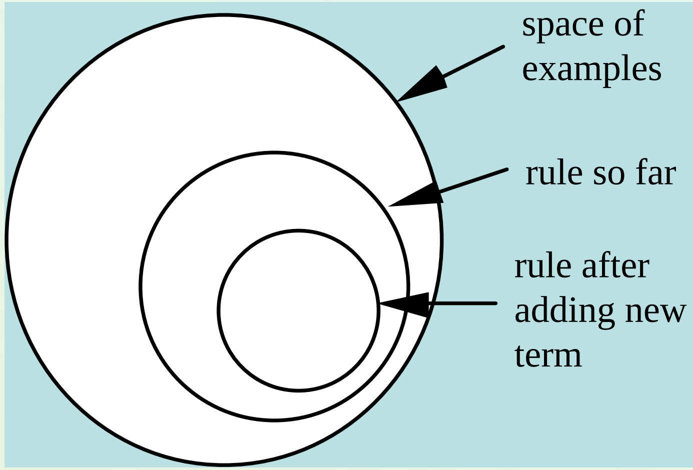
In Mitchell’s book – examples of weather / Play tennis decision

Learn-one-rule as heuristic search



A simple covering algorithm

- Generates a rule by adding tests that maximize rule's accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
  - But: decision tree inducer maximizes overall purity
- Each new term reduces rule's coverage:



Evaluation of candidates in Learning One Rule

- When is a candidate for a rule R treated as "good"?
  - High accuracy P(K|R);
  - High coverage |[P]I = *n*.
- Possible evaluation functions:
  - *Relative frequency*: $\frac{n_K(R)}{n(R)}$
    - where n_K is the number of correctly classified examples form class K, and *n* is the number of examples covered by the rule $\rightarrow$ problems with small samples;
  - Laplace estimate: $\frac{n_K(R)+1}{n(R)+k}$

    Good for uniform prior distribution of k classes
  - *m-estimate of accuracy*: $(n_K(R)+mp)/(n(R)+m)$,

    where n_K is the number of correctly classified examples, *n* is the number of examples covered by the rule, *p* is the prior probablity of the class predicted by the rule, and *m* is the weight of *p* (domain dependent – more noise / larger *m*).

Other evaluation functions of rule R and class K

Assume rule R specialized to rule R'

- Entropy (Information gain and others versions).
- Accuracy gain (increase in expected accuracy)

  P(K|R') – P(K|R)
- Many others
- Also weighted functions, e.g.

$$WAG(R', R) = \frac{n_K(R')}{n_K(R)} \cdot (P(K \mid R') - P(K \mid R))$$

$$WIG(R', R) = \frac{n_K(R')}{n_K(R)} \cdot (\log_2(K \mid R') - \log_2(K \mid R))$$

MODLEM – Algorithm for rule induction

- MODLEM [Stefanowski 98] generates a minimal set of rules.
- Its extra specificity – handling directly numerical attributes during rule induction; elementary conditions, e.g. ($a \geq v$), ($a < v$), ($a \in [v_1,v_2)$) or ($a = v$).
- Elementary condition evaluated by one of three measures: class entropy, Laplace accuracy or Grzymala 2-LEF.

obj.	*a*1	*a*2	*a*3	*a*4	*D*
*x*1	m	2.0	1	a	C1
*x*2	f	2.5	1	b	C2
*x*3	m	1.5	3	c	C1
*x*4	f	2.3	2	c	C1
*x*5	f	1.4	2	a	C2
*x*6	m	3.2	2	c	C2
*x*7	m	1.9	2	b	C1
*x*8	f	2.0	3	a	C2

if ($a1$ = m) *and* ($a2 \leq 2.6$) *then* (D = C1) $\{x1,x3,x7\}$

if ($a2 \in [1.45, 2.4]$) *and* ($a3 \leq 2$) *then* (D = C1) $\{x1,x4,x7\}$

if ($a2 \geq 2.4$) *then* (D = C2) $\{x2,x6\}$

if ($a1$ = f) *and* ($a2 \leq 2.15$) *then* (D = C2) $\{x5,x8\}$

Procedure Modlem

```
Procedure MODLEM
(input B - a set of positive examples from a given decision concept;
      criterion - an evaluation measure;
output 𝒯 – single local covering of B, treated here as rule condition parts)
begin
      G := B; {A temporary set of rules covered by generated rules}
      𝒯 := ∅;
      while G ≠ ∅ do {look for rules until some examples remain uncovered}
      begin
         T := ∅; {a candidate for a rule condition part}
         S := U; {a set of objects currently covered by T}
         while (T = ∅) or (not([T] ⊆ B)) do {stop condition for accepting a rule}
         begin
            t := ∅; {a candidate for an elementary condition}
            for each attribute q ∈ C do {looking for the best elementary condition}
            begin
               new_t :=Find_best_condition(q, S);
               if Better(new_t, t, criterion) then t := new_t;
               {evaluate if a new condition is better than previous one
               according to the chosen evaluation measure}
            end;
            T := T ∪ {t}; {add the best condition to the candidate rule}
            S := S ∩ [t]; {focus on examples covered by the candidate}
         end; { while not([T] ⊆ B }
         for each elementary condition t ∈ T do
            if [T − t] ⊆ B then T := T − {t}; {test a rule minimality}
         𝒯 := 𝒯 ∪ {T}; {store a rule}
         G := B − ⋃_{T∈𝒯} [T] ; {remove already covered examples}
      end; { while G ≠ ∅ }
      for each T ∈ 𝒯 do
         if ⋃_{T'∈𝒯−T} [T'] = B then 𝒯 := 𝒯 − T {test minimality of the rule set}
end {procedure}
```

Set of positive examples

Looking for the best rule

Testing conjunction

Finding the most discrimantory single condition

Extending the conjunction

Testing minimality

Removing covered examples

Find best condition

```
function Find_best_condition
(input c - given attribute; S - set of examples; output best_t - bestcondition)
begin
    best_t := ∅;
    if c is a numerical attribute then
    begin
        H:=list of sorted values for attribute c and objects from S;
        { H(i) - ith unique value in the list }
        for i:=1 to length(H)-1 do
        if object class assignments for H(i) and H(i+1) are different then
        begin
            v := (H(i) + H(i+1))/2;
            create a new_t as either (c < v) or (c ≥ v);
            if Better(new_t, best_t, criterion) then best_t := new_t ;
        end
    end
    else { attribute is nominal }
    begin
        for each value v of attribute c do
        if Better((c = v), best_t, criterion) then best_t := (c = v) ;
    end
end {function}.
```

Preparing the sorted value list

Looking for the best cut point between class assignments

Testing each candidate

Return the best evaluated condition

An Example (1)

No.	Age	Job	Period	Income	Purpose	Dec.
1	m	u	0	500	K	r
2	sr	p	2	1400	S	r
3	m	p	4	2600	M	d
4	st	p	16	2300	D	d
5	sr	p	14	1600	M	p
6	m	u	0	700	W	r
7	sr	b	0	600	D	r
8	m	p	3	1400	D	p
9	sr	p	11	1600	W	d
10	st	e	0	1100	D	p
11	m	u	0	1500	D	p
12	m	b	0	1000	M	r
13	sr	p	17	2500	S	p
14	m	b	0	700	D	r
15	st	p	21	5000	S	d
16	m	p	5	3700	M	d
17	m	b	0	800	K	r

Class (Decision = r)

$E = \{1, 2, 6, 7, 12, 14, 17\}$

List of candidates

(Age=m) {1,6,12,14,17+; 3,8,11,16-}
(Age=sr) {2,7+; 5,9,13-}

(Job=u) {1,6+; 11-}
(Job=p) {2+, 3,4,8,9,13,15,16-}
(Job=b) {7,12,14,17+; ∅}

(Pur=K) {1,17+; ∅}
(Pur=S) {2+;13,15-}
{Pur=W} {6+, 9-}
{Pur=D} {7,14+; 4,8,10,11-}
{Pur=M} {12+;5,16-}

An Example (2)

- Numerical attributes: Income

500	600	700	800	1000	1100	1400	1500	1600	2300	2500	2600	3700	5000
1+	7+	6+ 14+	17+	12+	10-	2+ 8-	11-	9- 5-	4-	13-	3-	10-	15-

(Income < 1050) {1,6,7,12,14,17+;∅}

(Income < 1250) {1,6,7,12,14,17+;10-}

(Income < 1450) {1,2,6,7,12,14,17+;8,10-}

Period

(Period < 1) {1,6,7,14,17+;10,11-}

(Period < 2.5) {1,2,6,7,12,14,17+;10,11-}

Example (3) - the minimal set of induced rule

1. if (Income<1050) then (Dec=r) [6]
2. if (Age=sr) and (Period<2.5) then (Dec=r) [2]
3. if (Period$\in$[3.5,12.5)) then (Dec=d) [2]
4. if (Age=st) and (Job=p) then (Dec=d) [3]
5. if (Age=m) and (Income$\in$[1050,2550)) then (Dec=p) [2]
6. if (Job=e) then (Dec=p) [1]
7. if (Age=sr) and (Period$\geq$12.5) then (Dec=p) [2]

- For inconsistent data:
    - Approximations of decision classes (rough sets)
    - Rule post-processing (a kind of post-pruning) or extra testing and earlier acceptance of rules.

Mushroom data (UCI Repository)

- Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981).
- This data set includes descriptions of hypothetical samples corresponding to 23 species of mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility.
- Number of examples: 8124.
- Number of attributes: 22 (all nominally valued)
- Missing attribute values: 2480 of them.
- Class Distribution:
  - -- edible: 4208 (51.8%)
  - -- poisonous: 3916 (48.2%)

MOLDEM rule set (Implemented in WEKA)

=== Classifier model (full training set) ===

Rule 1.(odor is in: {n, a, l})&(spore-print-color is in: {n, k, b, h, o, u, y, w})&(gill-size = b) => (class = e); [3920, 3920, 93.16%, 100%]

Rule 2.(odor is in: {n, a, l})&(spore-print-color is in: {n, h, k, u}) => (class = e); [3488, 3488, 82.89%, 100%]

Rule 3.(gill-spacing = w)&(cap-color is in: {c, n}) => (class = e); [304, 304, 7.22%, 100%]

Rule 4.(spore-print-color = r) => (class = p); [72, 72, 1.84%, 100%]

Rule 5.(stalk-surface-below-ring = y)&(gill-size = n) => (class = p); [40, 40, 1.02%, 100%]

Rule 6.(odor = n)&(gill-size = n)&(bruises? = t) => (class = p); [8, 8, 0.2%, 100%]

Rule 7.(odor is in: {f, s, y, p, c, m}) => (class = p); [3796, 3796, 96.94%, 100%]

Number of rules: 7

Number of conditions: 14

Approaches to Avoiding Overfitting

- **Pre-pruning:** stop learning the decision rules before they reach the point where they perfectly classify the training data

- **Post-pruning:** allow the decision rules to overfit the training data, and then post-prune the rules.

Pre-Pruning

The criteria for stopping learning rules can be:

- **minimum purity** criterion requires a certain percentage of the instances covered by the rule to be positive;
- **significance test** determines if there is a significant difference between the distribution of the instances covered by the rule and the distribution of the instances in the training sets.

Post-Pruning

1. Split instances into *Growing Set* and *Pruning Set*;
2. Learn set *SR* of rules using *Growing Set*;
3. Find the best simplification *BSR* of *SR*.
4. **while** (Accuracy(*BSR*, *Pruning Set*) > Accuracy(*SR*, *Pruning Set*)) **do**

4.1 *SR* = *BSR*;

4.2 Find the best simplification *BSR* of *SR*.

5. **return** *BSR*;

Applying rule set to classify objects

- **Matching** a new object description *x* to condition parts of rules.
  - Either object's description satisfies all elementary conditions in a rule, or not.

  IF (a1=L) and (a3$\geq$ 3) THEN Class +

  x $\rightarrow$ (a1=L),(a2=s),(a3=7),(a4=1)
- Two ways of assining x to class K depending on the set of rules:
  - Unordered set of rules (AQ, CN2, PRISM, LEM)
  - Ordered list of rules (CN2, c4.5rules)

Applying rule set to classify objects

- The rules are ordered into priority decision list!

  Another way of rule induction – rules are learned by first determining Conditions and then Class (CN2)

Notice: mixed sequence of classes K1,…, K in a rule list

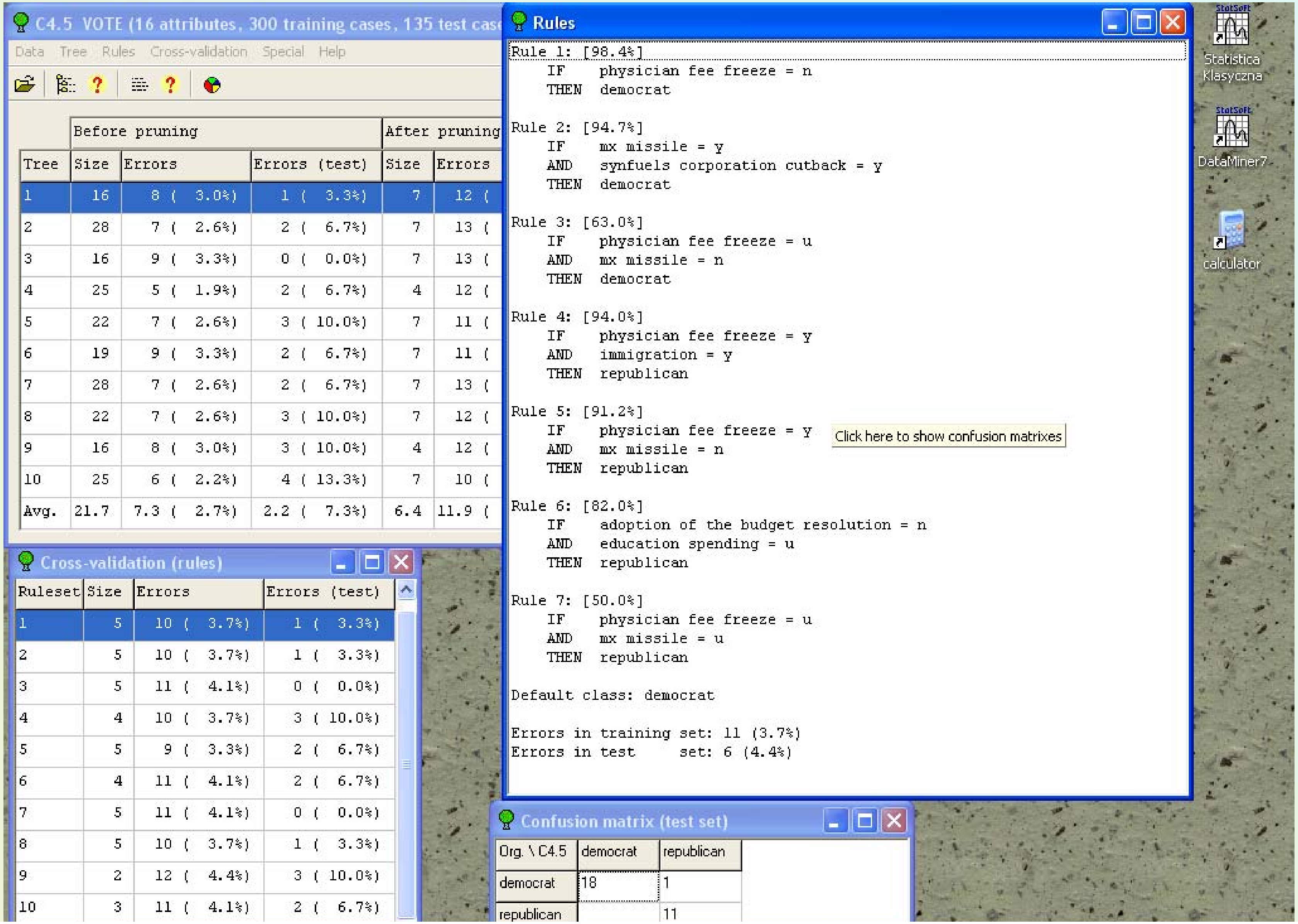
But: ordered execution when classifying a new instance: rules are sequentially tried and the first rule that 'fires' (covers the example) is used for final decision

Decision list {R1, R2, R3, …, D}: rules Ri are

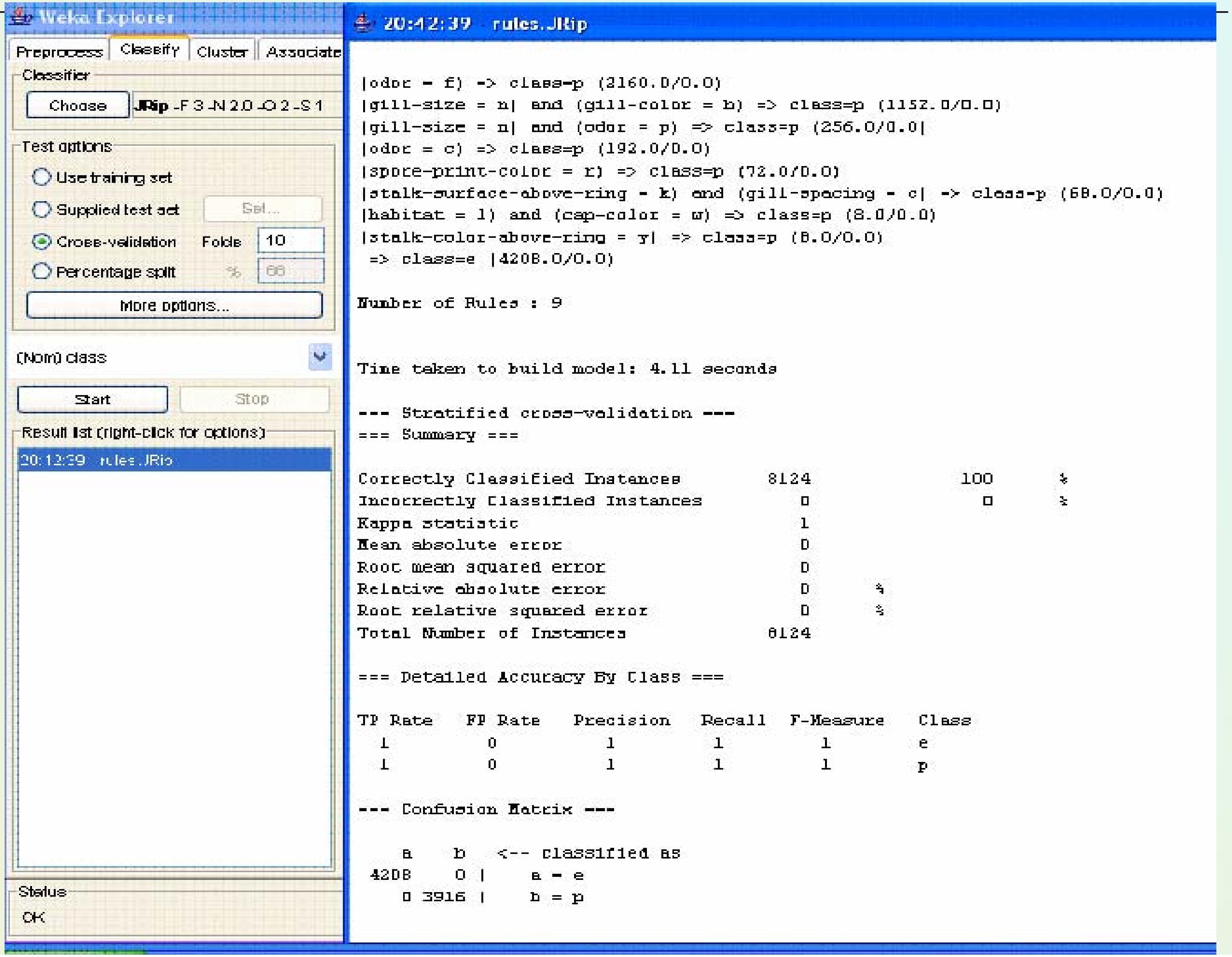
interpreted as **if-then-else** rules

If no rule fires, then DefaultClass (majority class in input data)

Priority decision list (C4.5 rules)



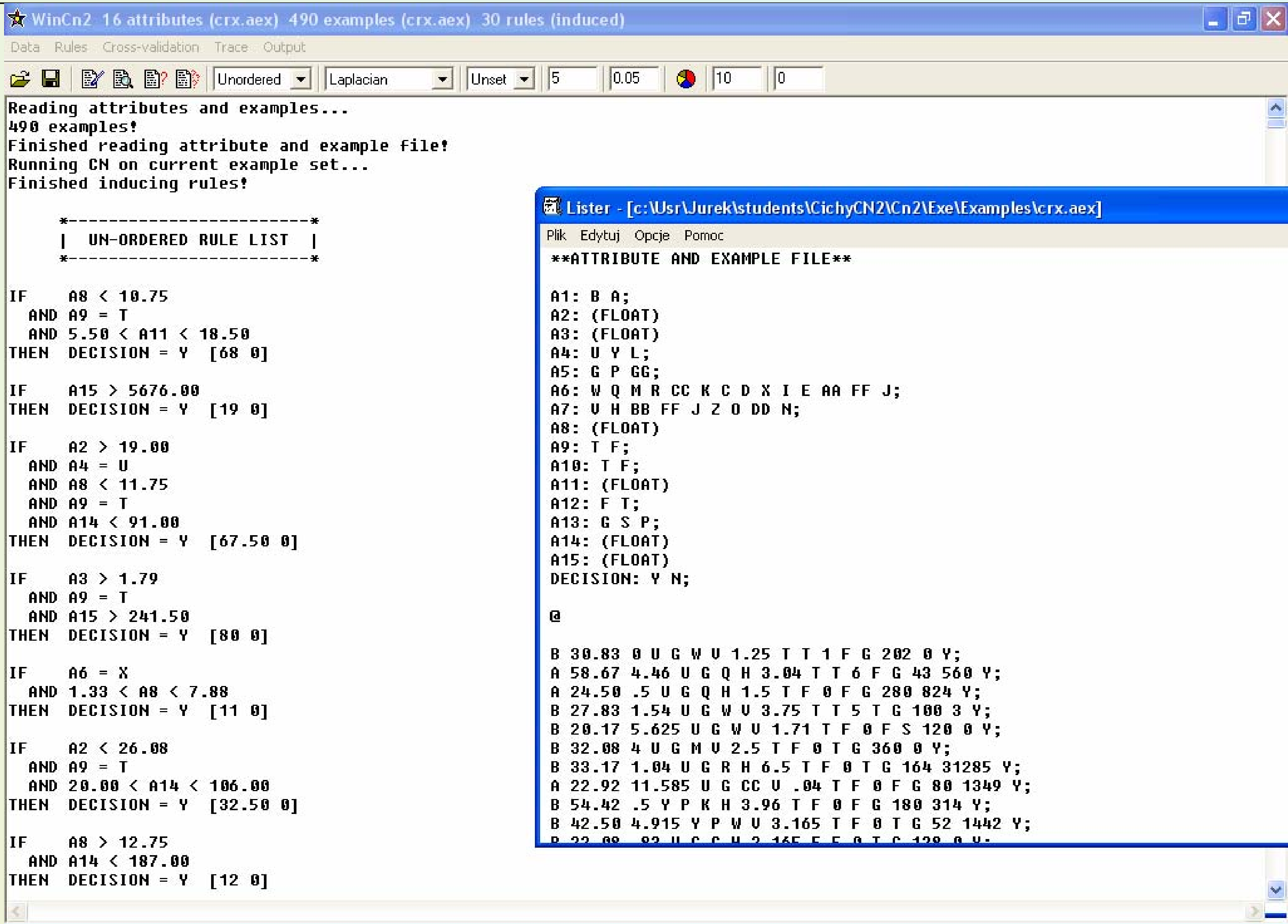
Specific list of rules - RIPPER (Mushroom data)



Learning ordered set of rules

- RuleList := empty; E_{cur}:= E
- **repeat**
  - learn-one-rule R
  - RuleList := RuleList ++ R
  - E_{cur} := E_{cur} - {all examples covered by R}
    (Not only positive examples !)
- **until** performance(R, E_{cur}) < ThresholdR
- RuleList := sort RuleList by performance(R,E)
- RuleList := RuleList ++ DefaultRule(E_{cur})

CN2 – unordered rule set



Applying unordered rule set to classify objects

- An unordered set of rules $\rightarrow$ three situations:
  - Matching to rules indicating the same class.
  - Multiple matching to rules from different classes.
  - No matching to any rule.
- An example:
- e1={(Age=m), (Job=p),(Period=6),(Income=3000),(Purpose=K)}
  - rule 3: if (Period$\in$[3.5,12.5)) then (Dec=d) [2]
  - Exact matching to rule 3. $\rightarrow$ Class (Dec=d)
- e2={(Age=m), (Job=p),(Period=2),(Income=2600),(Purpose=M)}
  - No matching!

Solving conflict situations

- LERS classification strategy (Grzymala 94)
  - Multiple matching
    - Two factors: *Strength*(*R*) – number of learning examples correctly classified by *R* and final class *Support*(*Y*i):

$$\sum_{\text{matching rules R for Yi}} Strength(R)$$

  - Partial matching
    - Matching factor *MF*(*R*) and

$$\sum_{\text{partially match. rules R for Yi}} MF(R) \cdot Strength(R)$$

- e2={(Age=m), (Job=p), (Period=2),(Income=2600),(Purpose=M)}
  - Partial matching to rules 2 , 4 and 5 for all with MF = 0.5
  - Support(r) = 0.5·2 =1 ; Support(d) = 0.5·2+0.5·2=2
- Alternative approaches – e.g. nearest rules (Stefanowski 95)
- Instead of MF use a kind of normalized distance *x* to conditions of *r*

Some experiments

- Analysing strategies (total accuracy in [%]):

data set	all	multiple	exact
large soybean	87.9	85.7	79.2
election	89.4	79.5	71.8
hsv2	77.1	70.5	59.8
concretes	88.9	82.8	81.0
breast cancer	67.1	59.3	51.2
imidasolium	53.3	44.8	34.4
lymphograpy	85.2	73.6	67.6
oncology	83.8	82.4	74.1
buses	98.0	93.5	90.8
bearings	96.4	90.9	87.3

- Comparing to other classification approaches
  - Depends on the data
  - Generally $\rightarrow$ similar to decision trees

Variations of inducing minimal sets of rules

- Sequential vs. simultaneous covering of data.
- General-to-specific vs. specific-to-general; begin search from single most general vs. many most specific starting hypotheses.
- Generate-and-test vs. example driven (as in AQ).
- Pre-pruning vs. post-pruning of rules
- What evaluation functions to use?
- …

Different perspectives of rule application

- In a descriptive perspective
  - To present, analyse the relationships between values of attributes, to explain and understand classification patterns
- In a prediction/classification perspective,
  - To predict value of decision class for new (unseen) object)

Perspectives are different;
Moreover rules are evaluated in a different ways!

Evaluating single rules

- rule *r* (*if P then Q*) derived from *DT*, examples *U*.

	Q	$\neg Q$	
P	n_{PQ}	$n_{P\neg Q}$	n_P
$\neg P$	$n_{\neg PQ}$	$n_{\neg P\neg Q}$	$n_{\neg P}$
	n_Q	$n_{\neg Q}$	n

- Reviews of measures, e.g.
- Yao Y.Y, Zhong N., An analysis of quantitative measures associated with rules, In: Proc. the 3rd Pacific-Asia Conf. on Knowledge Discovery and Data Mining, LNAI 1574, Springer, 1999, pp. 479-488.
- Hilderman R.J., Hamilton H.J, Knowledge Discovery and Measures of Interest. Kluwer, 2002.
- Support of rule r $$G(P \wedge Q) = \frac{n_{PQ}}{n}$$ Coverage $$AS(P \mid Q) = \frac{n_{PQ}}{n_Q}$$
- Confidence of rule r $$AS(Q \mid P) = \frac{n_{PQ}}{n_P}$$ and others …

Descriptive requirements to single rules

- In descriptive perspective users may prefer to discover rules which should be:
  - strong / general – high enough rule coverage $AS(P|Q)$ or support.
  - accurate – sufficient accuracy $AS(Q|P)$.
  - simple (e.g. which are in a limited number and have short condition parts).
  - Number of rules should not be too high.
- Covering algorithms biased towards minimum set of rules - containing only a limited part of potentially `interesting' rules.
  - We need another kind of rule induction algorithms!

Explore algorithm (Stefanowski, Vanderpooten)

- Another aim of rule induction
  - to extract from data set inducing **all rules** that *satisfy* some *user's requirements* connected with *his interest* (regarding, e.g. the strength of the rule, level of confidence, length, sometimes also emphasis on the syntax of rules).
- Special technique of exploration the space of possible rules:
  - Progressively generation rules of increasing size using in the most efficient way some 'good' pruning and stopping condition that reject unnecessary candidates for rules.
- Similar to adaptations of Apriori principle for looking frequent itemsets [AIS94]; Brute [Etzioni]

Explore – some algorithmic details

procedure Explore (*LS*: list of conditions; SC: stopping conditions; **var** *R*: set_of_rules);

begin

$R \leftarrow \emptyset$;

Good_Candidates(*LS,R*); {*LS* - ordered list of *c*1,*c*2,..,*c*n}

$Q \leftarrow LS$; {create a queue *Q*}

while $Q \neq \emptyset$ **do**

begin

select the first conjunction *C* from *Q* ;

$Q \leftarrow Q \backslash \{C\}$;

Extend(*C*,*LC*); {*LC* - list of extended conjunctions}

Good_Candidates(*LC,R*);

$Q \leftarrow Q \cup C$; {place all conjunctions from *LC* at the end of *Q*}

end

end.

procedure Extend(*C* : conjunction, **var** *L* : list of conjunctions);

{This procedure puts in list *L* extensions of conjunction *C* that are possible candidates for rules}

begin

Let *k* be the size of *C* and *h* be the highest index of elementary conditions involved in *C;*

$L \leftarrow \{C \wedge c_{h+i}$ where $ch+i \in LS$ and such that all the *k* subconjunctions of $C \wedge c_{h+i}$ of size *k* and involving c_{h+i} belong to *Q* , i=1,..,n-h}

end

procedure Good_Candidates(*LC* : ist of conjunctions, **var** *R* - set of rules);

{This procedure prunes list *LC* discarding:

- conjunctions whose extension cannot give rise to rules due to SC,
- conjunctions corresponding to rules which are already stored in *R*

Various sets of rules (Stefanowski and Vanderpooten 1994)

- A minimal set of rules (LEM2):

rule 1.	if $(q_1 = 2) \wedge (q_3 = 1)$ then $(d = 1)$	$\{1, 2, 3, 4, 5\}$	5/8
rule 2.	if $(q_1 = 1)$ then $(d = 1)$	$\{6, 7\}$	2/8
rule 3.	if $(q_3 = 2) \wedge (q_6 = 2)$ then $(d = 1)$	$\{6, 8\}$	2/8
rule 4.	if $(q_1 = 3)$ then $(d = 2)$	$\{9, 10, 11, 13, 14\}$	5/7
rule 5.	if $(q_3 = 3)$ then $(d = 2)$	$\{15\}$	1/7
rule 6.	if $(q_3 = 2) \wedge (q_4 = 1) \wedge (q_6 = 1)$ then $(d = 2)$	$\{12\}$	1/7

- A „satisfactory" set of rules (Explore):

Let us assume that the user's level of interest to the possible strength of a rule by assigning a value $l = 50\%$ in SC.

Explore gives the following decision rules:

rule 1.	if $(q_2 = 3)$ then $(d = 1)$	$\{1, 2, 3, 6, 7\}$	5/8
rule 2.	if $(q_1 = 2) \wedge (q_3 = 1)$ then $(d = 1)$	$\{1, 2, 3, 4, 5\}$	5/8
rule 3.	if $(q_1 = 3)$ then $(d = 2)$	$\{9, 10, 11, 13, 14\}$	5/7
rule 4.	if $(q_4 = 2)$ then $(d = 2)$	$\{10, 13, 14, 15\}$	4/7

Table 1: The illustrative set of learning exam

No.	q_1	q_2	q_3	q_4	q_5	q_6	d
1	2	3	1	3	1	2	1
2	2	3	1	1	1	1	1
3	2	3	1	3	2	1	1
4	2	1	1	1	1	1	1
5	2	2	1	1	2	2	1
6	1	3	2	3	1	2	1
7	1	3	2	3	2	1	1
8	2	1	2	1	2	2	1
9	3	1	1	3	1	2	2
10	3	1	2	2	2	1	2
11	3	1	1	3	2	2	2
12	2	1	2	1	2	1	2
13	3	2	4	2	1	1	2
14	3	2	4	2	2	1	2
15	2	2	3	2	1	2	2
16	2	2	2	1	1	1	1
17	2	2	2	1	1	1	2

A diagnostic case study

- A fleet of homogeneous 76 buses (AutoSan H9-21) operating in an inter-city and local transportation system.
- The following symptoms characterize these buses :

  s1 – maximum speed [km/h],

  s2 – compression pressure [Mpa],

  s3 – blacking components in exhaust gas [%],

  s4 – torque [Nm],

  s5 – summer fuel consumption [l/100lm],

  s6 – winter fuel consumption [l/100km],

  s7 – oil consumption [l/1000km],

  s8 – maximum horsepower of the engine [km].

Experts' classification of busses:

1. Buses with engines in a good technical state – further use (46 buses),
2. Buses with engines in a bad technical state – requiring repair (30 buses).

LEM2 algorithm – (sequential covering)

- A ***minimal*** set of *discriminating* decision rules

  1. if ($s2 \geq 2.4$ MPa) & ($s7 < 2.1$ l/1000km) then (technical state=good) [46]

  2. if ($s2 < 2.4$ MPa) then (technical state=bad) [29]

  3. if ($s7 \geq 2.1$ l/1000km) then (technical state=bad) [24]

- The prediction accuracy (‘leaving-one-out’ reclassification test) is equal to 98.7%.

Another set of rules (EXPLORE)

All decision rules with min. SC1 threshold (rule coverage > 50%):

1. if (*s1*>85 km/h) then (technical state=good) [34]
2. if (*s8*>134 kM) then (technical state=good) [26]
3. if (*s2*≥2.4 MPa) & (*s3*<61 %) then (technical state=good) [44]
4. if (*s2*≥2.4 MPa) & (*s4*>444 Nm) then (technical state=good) [44]
5. if (*s2*≥2.4 MPa) & (s7<2.1 *l*/1000km) then (technical state=good) [46]
6. if (*s3*<61 %) & (*s4*>444 Nm) then (technical state=good) [42]
7. if (*s1*≤77 km/h) then (technical state=bad) [25]
8. if (*s2*<2.4 MPa) then (technical state=bad) [29]
9. if (*s7*≥2.1 *l*/1000km) then (technical state=bad) [24]
10. if (*s3*≥61 %) & (*s4*≤444 Nm) then (technical state=bad) [28]
11. if (*s3*≥61 %) & (*s8*<120 kM) then (technical state=bad) [27]

The prediction accuracy - 98.7%

Descriptive vs. classification properties (Explore)

Data set	Stopping conditions		Number of rules	Average rule length [# cond.]	Average rule strength [# exam.]	classification accuracy [%]
	SC1	SC2				
Iris	All rules		80	2.1	6.03	92.67
	5%	---	35	1.89	12.23	92.67
	10%	---	22	1.86	17.27	92
	15%	---	20	1.85	18.4	90
	20%	---	15	1.8	21.6	83.33
	25%	---	14	1.79	22.36	78.67
	30%	---	6	1.83	33.83	60.67
	Minimum	rule set	23	1.91	11	95.33
Tic-tac-toe	All rules		2858	4.63	4.27	91.35
	5%	5	16	3	60.25	97.19
	10%	5	16	3	60.25	96.14
	15%	5	2	3	50	---
	20%	5	0	---	---	---
	30%	5	0	---	---	---
	Minimum	rule set	24	3.67	40.83	98.96
Voting	All rules		1502	4.723	10.61	95.87
	5%	4	231	3.6	45.86	94.51
	10%	4	138	3.3	66.96	94.5
	15%	4	104	3.1	79.61	93.8
	20%	4	82	3.1	89.87	94
	25%	4	67	3.1	96.99	93.32
	30%	4	50	3.1	104.7	93.31
	40%	4	21	2.76	133	80.23
	Minimum	rule set	26	3.69	43.77	95.87
Election	All rules		>500000	---	---	---
	10%	---	828	3.48	26.91	89.39
	15%	---	87	3.05	33.82	87.37
	20%	---	8	2.38	53.75	73.88
	25%	---	2	1.5	79	32.96
	30%	---	1	1	105	23.64
	Minimum	rule set	48	3.27	21.176	89.41

- Tuning a proper value of stopping condition SC (rule coverage) leads to sets of rules which are „satisfactory” with respect to a number of rules, average rule length and average rule strength without decreasing too much the classification accuracy.

Where are we now?

1. Rule representation
2. Various algorithms for rule induction.
3. MODLEM $\rightarrow$ exemplary algorithm for inducing a minimal set of rules.
4. Classification strategies
5. Descriptive properties of rules.
6. Explore $\rightarrow$ discovering a richer set of rules.
7. Association rules
8. Logical relations
9. Final remarks.

Association rules

- Transaction data
- Market basket analysis

TID	Produce
1	MILK, BREAD, EGGS
2	BREAD, SUGAR
3	BREAD, CEREAL
4	MILK, BREAD, SUGAR
5	MILK, CEREAL
6	BREAD, CEREAL
7	MILK, CEREAL
8	MILK, BREAD, CEREAL, EGGS
9	MILK, BREAD, CEREAL

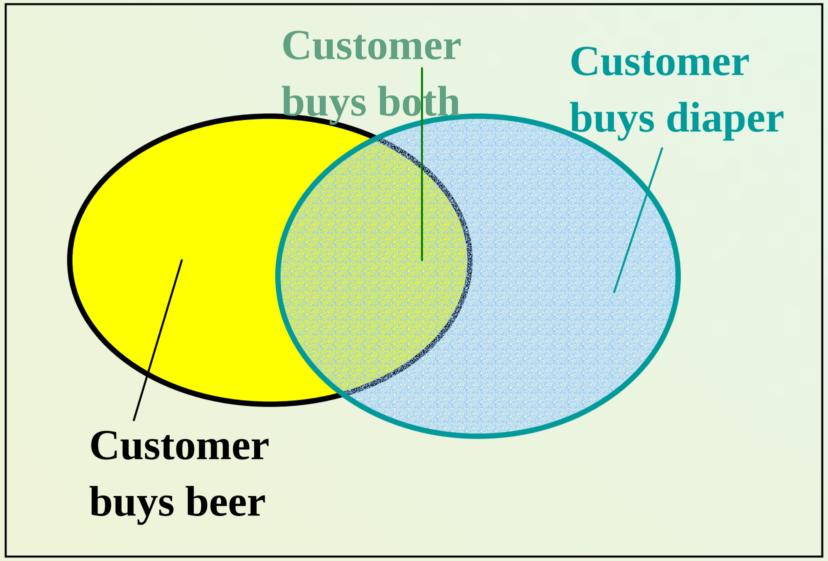
- {Cheese, Milk} → Bread [sup=5%, conf=80%]
- Association rule:
  „80% of customers who buy *cheese* and *milk* also buy *bread* and 5% of customers buy all these products together"

Why is Frequent Pattern or Association Mining an Essential Task in Data Mining?

- Foundation for many essential data mining tasks
  - Association, correlation, causality
  - Sequential patterns, temporal or cyclic association, partial periodicity, spatial and multimedia association
  - Associative classification, cluster analysis, fascicles (semantic data compression)
- DB approach to efficient mining massive data
- Broad applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis
  - Web log (click stream) analysis, DNA sequence analysis, etc

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F



- Itemset $X=\{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with min confidence and support
  - support, s, probability that a transaction contains $X \cup Y$
  - confidence, c, conditional probability that a transaction having X also contains Y.

Let min_support = 50%,
min_conf = 50%:

$A \rightarrow C$ (50%, 66.7%)

$C \rightarrow A$ (50%, 100%)

Mining Association Rules—an Example

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule $A \Rightarrow C$:

support = support($\{A\} \cup \{C\}$) = 50%

confidence = support($\{A\} \cup \{C\}$)/support($\{A\}$) = 66.6%

Generating Association Rules

- Two stage process:
  - Determine frequent itemsets e.g. with the Apriori algorithm.
  - For each frequent item set *I*
    - for each subset *J* of *I*
      - determine all association rules of the form: *I-J => J*
- Main idea used in both stages : subset property
- Focus on computational efficiency, access to data, scalability, …

Apriori: A Candidate Generation-and-test Approach

- Any subset of a frequent itemset must be frequent
  - if **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - generate length (k+1) candidate itemsets from length k frequent itemsets, and
  - test the candidates against DB
- The performance studies show its efficiency and scalability
- Agrawal & Srikant 1994, Mannila, et al. 1994

The Apriori Algorithm—An Example

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

Example: Generating Rules from an Itemset

- Frequent itemset from Play data:

  `Humidity = Normal, Windy = False, Play = Yes (4)`

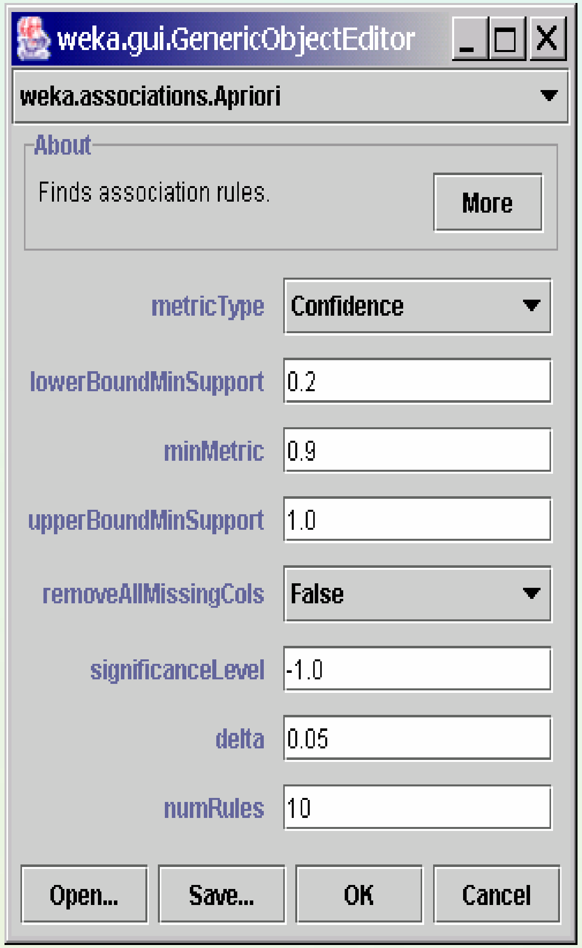
- Seven potential rules:

```
If Humidity = Normal and Windy = False then Play = Yes          4/4
If Humidity = Normal and Play = Yes then Windy = False          4/6
If Windy = False and Play = Yes then Humidity = Normal          4/6
If Humidity = Normal then Windy = False and Play = Yes          4/7
If Windy = False then Humidity = Normal and Play = Yes          4/8
If Play = Yes then Humidity = Normal and Windy = False          4/9
If True then Humidity = Normal and Windy = False and Play = Yes 4/12
```

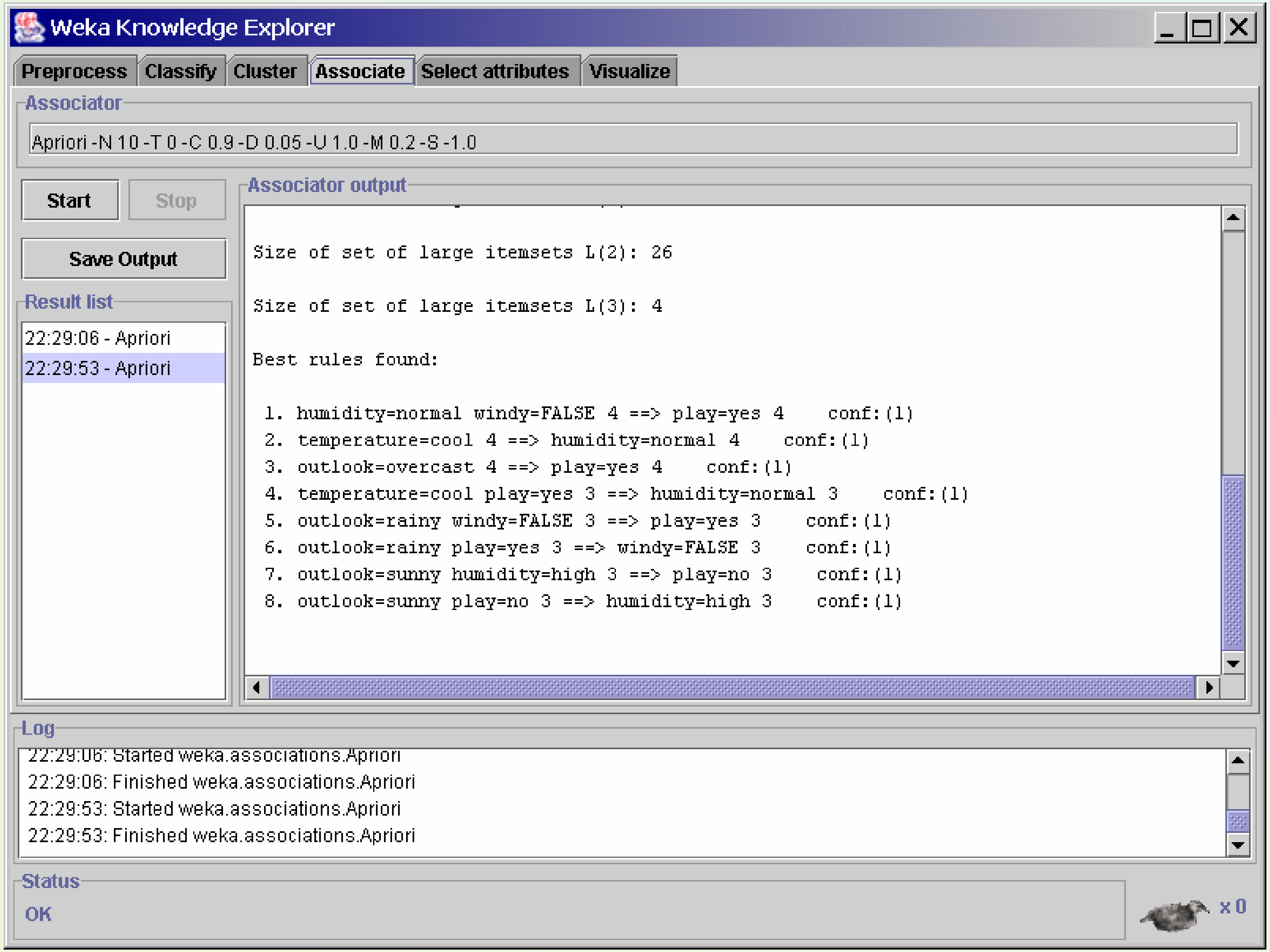
Weka associations

File: weather.nominal.arff

MinSupport: 0.2



Weka associations: output

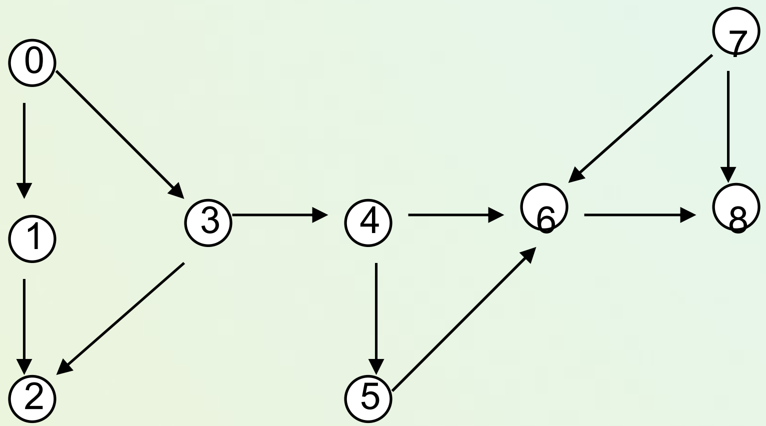


Learning First Order Rules

- Is object/attribute table sufficient data representation?
- Some limitations:
  - Representation expressivness – unable to express relations between objects or object elements. ,
  - *background knowledge* sometimes is quite complicated.
- Can learn sets of rules such as
  - $Parent(x,y) \rightarrow Ancestor(x,y)$
  - $Parent(x,z)$ and $Ancestor(z,y) \rightarrow Ancestor(x,y)$
- Research field of **Inductive Logic Programming**.

Why ILP? (slide of S.Matwin)

- **expressiveness of logic as representation** (Quinlan)



- can't represent this graph as a fixed length vector of attributes
- can't represent a "transition" rule:

A can-reach B if A link C, and C can-reach B

without variables

FINITE ELEMENT MESH DESIGN

Given a geometric **structure** and **loadings/boundary conditions**
Find an **appropriate resolution** for a finite element mesh

Examples: ten structures with appropriate meshes (cca. 650 edges)

Background knowledge

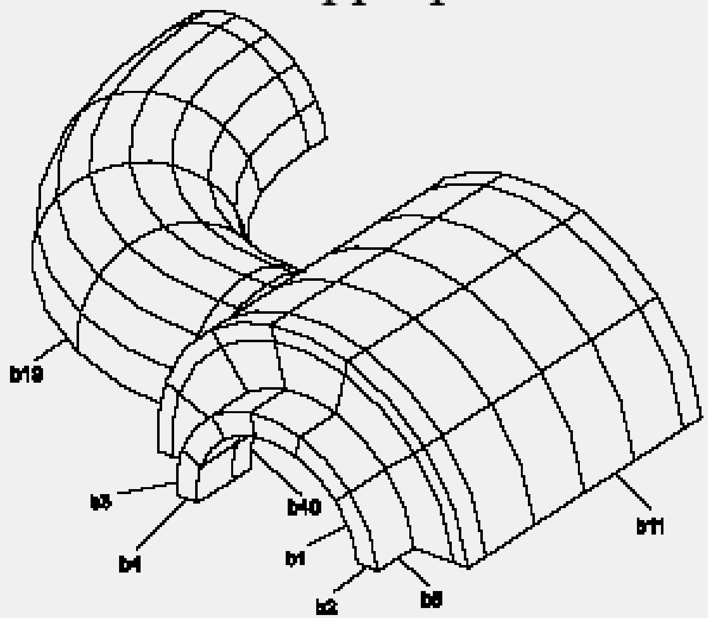
- Properties of edges (short, loaded, two-side-fixed, ...)
- Relations between edges (neighbor, opposite, equal)

ILP systems applied: GOLEM, CLAUDIEN

Many interesting rules discovered (according to expert evaluation)

Finite element mesh design (ctd.)

Example structure with an appropriate mesh



Example rules

$$mesh(Edge, 7) \leftarrow usual\_length(Edge),$$
$$neighbour\_xy(Edge, EdgeY), two\_side\_fixed(EdgeY),$$
$$neighbour\_zx(EdgeZ, Edge), not\_loaded(EdgeZ)$$
$$mesh(Edge, N) \leftarrow equal(Edge, Edge2), mesh(Edge2, N)$$

Application areas

- Medicine
- Economy, Finance
- Environmental cases
- Engineering
  - Control engineering and robotics
  - Technical diagnostics
  - Signal processing and image analysis
- Information sciences
- Social Sciences
- Molecular Biology
- Chemistry and Pharmacy
- …

Where to find more?

- T. Mitchell *Machine Learning* New York: McGraw-Hill, 1997.
- I. H. Witten & Eibe Frank *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations* San Francisco: Morgan Kaufmann, 1999.
- Michalski R.S., Bratko I., Kubat M. *Machine learning and data mining*; J. Wiley. 1998.
- Clark, P., & Niblett, T. (1989). The CN2 induction algorithm.*Machine Learning*, *3*, 261–283.
- Cohen W. Fast effective rule induction. *Proc. of the 12th Int. Conf. on Machine Learning 1995.* 115–123
- R.S. Michalski, I. Mozetic, J. Hong and N. Lavrac, The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, *Proceedings of i4AAI 1986, 1041-1045, (1986).*
- J.W. Grzymala-Busse, LERS-A system for learning from example-s based on rough sets, In Intelligent`*Decision* Support: Handbook *of* Applications and Advances *of Rough Sets Theory,* (Edited by R.Slowinski), pp. 3-18
- Michalski R.S.: A theory and methodology of inductive learning. W Michalski R.S, Carbonell J.G., Mitchell T.M. (red.) *Machine learning: An Artificiall Intelligence Approach,* Morgan Kaufmann Publishers, Los Altos (1983),.
- J.Stefanowski: On rough set based approaches to induction of decision rules, w: A. Skowron, L. Polkowski (red.), *Rough Sets in Knowledge Discovery Vol* 1, Physica Verlag, Heidelberg, 1998, 500-529.
- J.Stefanowski, The rough set based rule induction technique forclassification problems, w: *Proceedings of 6th European Conference on Intelligent Techniques and Soft Computing, Aachen, EUFIT 98*, 1998, 109-113.
- J. Furnkranz . Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

Where to find more - 2

- P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Working Session on Learning (EWSL-91)*, pp. 151–163, 1991.
- Grzymala-Busse J.W.: Managing uncertainty in machine learning from examples. Proceedings of 3rd Int. Symp. on Intelligent Systems, Wigry 1994 .
- Cendrowska J.: PRISM, an algorithm for inducing modular rules. *Int. J. Man-Machine Studies*, 27 (1987), 349-370.
- Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. *Proc. of the 15th Int. Conf. on Machine Learning (ICML-98)* (pp. 144–151).
- J. Furnkranz and P. Flach. An analysis of rule evaluation metrics. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 202–209,
- S. M. Weiss and N. Indurkhya. Lightweight rule induction. In *Proc. of the 17th Int. Conference on Machine Learning (ICML-2000)*, pp. 1135–1142,
- J.Stefanowski, D.Vanderpooten: Induction of decision rules in classification and discovery-oriented perspectives, *International Journal of Intelligent Systems*, vol. 16 no. 1, 2001, 13-28.
- J.W.Grzymala-Busse, J.Stefanowski: Three approaches to numerical attribute discretization for rule induction, *International Journal of Intelligent Systems*, vol. 16 no. 1, 2001, 29-38.
- P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.
- R. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

Any questions, remarks?

?