Odkrywanie reguł klasyfikacyjnych bezpośrednio z danych

JERZY STEFANOWSKI
Institute of Computing Sciences
Poznań University of Technology

TPD wykład ZED, 2008

Źródła

- Wykład częściowo oparty na moim wykładzie szkoleniowym dla COST Action Spring School on Data Mining and MCDA – Troina 2008 oraz wcześniejszych wystąpieniach konferencyjnych.
- Proszę także przeczytać stosowane rozdziały z mojej rozprawy habilitacyjnej – dostępna na stronie WWW.

Indukcja reguł decyzyjnych

- Podstawowa idea - reguły poszukuje się bezpośrednio z danych
  - potencjalnie większa zrozumiałość wiedzy
  - ale więcej różnych podejść:
    - opis danych z wykorzystaniem minimalnego zbioru reguł o dobrych własnościach.
    - poszukiwanie bardziej wyczerpujących zbiorów reguł o dobrych własnościach interpretacyjnych
  - więcej parametrów do sterowania w metodach indukcji reguł

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

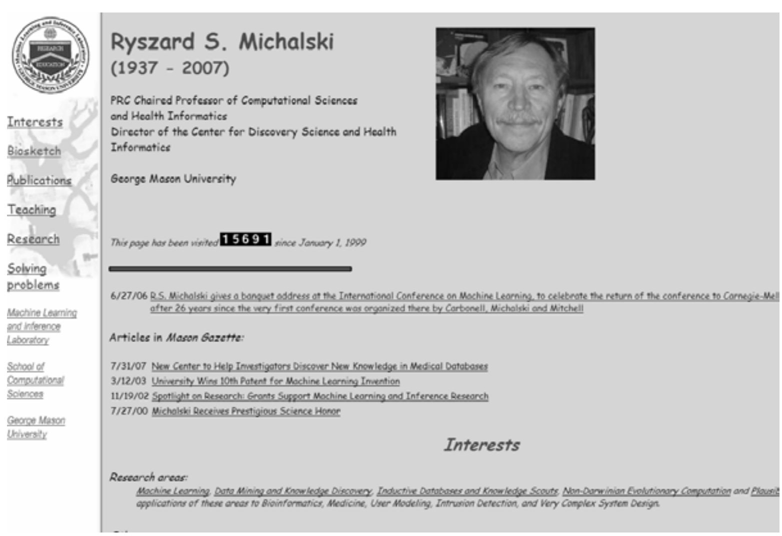
if outlook = rainy **and** windy = TRUE **then** Play = no

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.

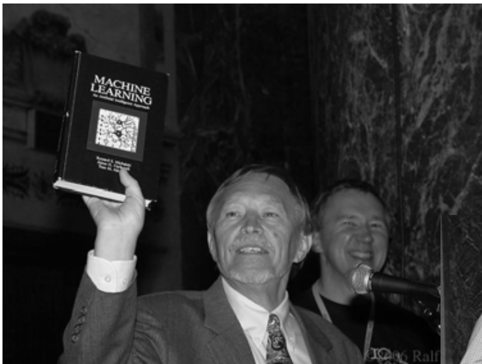
Polski wątek – prof. Ryszard Michalski

- Father of Machine Learning and rule induction



Trochę więcej o „ojcach założycielach”

- J.Carbonel, R.Michalski, T.Mitchell



Covering algorithms

- A strategy for generating a rule set directly from data:
  - for each class in turn find rule set that covers all instances in it (excluding instances 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}

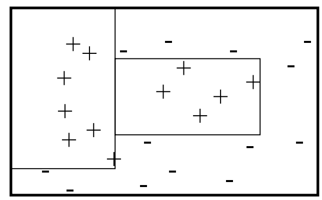
find-one-rule R covering some positive examples

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 pruning 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).

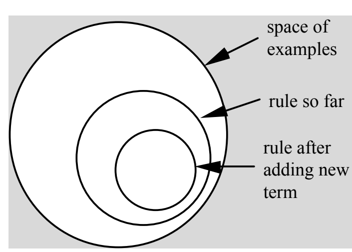
General schema of inducing minimal set of rules

- 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 (Kj Class; A attributes; E examples,
τ - acceptance threshold);
begin
   R := ∅;     {set of induced rules}
   r := learn-one-rule(Yj Class; A attributes; E examples)
   while evaluate(r,E) > τ do
   begin
     R := R ∪ r;
     E := E \ [R];      {remove positive examples covered by R}
     r := learn-one-rule(Kj Class; A attributes; E examples);
   end;
return R
end.
```

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 (*a*1 = m) ***and*** (*a*2 ≤ 2.6) ***then*** (*D* = C1)   {*x*1,*x*3,*x*7}

if (*a*2 ∈ [1.45, 2.4]) ***and*** (*a*3 ≤ 2) ***then*** (*D* = C1)
{*x*1,*x*4,*x*7}

if (*a*2 ≥ 2.4) ***then*** (*D* = C2)   {*x*2,*x*6}

if (*a*1 = f) ***and*** (*a*2 ≤ 2.15) ***then*** (*D* = C2)   {*x*5,*x*8}

Procedure Modlem

Procedure MODLEM
(**input** B - a set of positive examples from a given decision concept;
    criterion - an evaluation measure;
output $\mathcal{T}$ – single local covering of B, treated here as rule condition parts)
begin
    $G := B$; {A temporary set of rules covered by generated rules}
    $\mathcal{T} := \emptyset$;
    while $G \neq \emptyset$ **do** {look for rules until some examples remain uncovered}
    begin
        $T := \emptyset$; {a candidate for a rule condition part}
        $S := U$; {a set of objects currently covered by T}
        while $(T = \emptyset)$ or $(\text{not}([T] \subseteq B))$ **do** {stop condition for accepting a rule}
        begin
            $t := \emptyset$; {a candidate for an elementary condition}
            for each attribute $q \in C$ **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;
            $T := T \cup \{t\}$; {add the best condition to the candidate rule}
            $S := S \cap [t]$; {focus on examples covered by the candidate}
        end; { while not$([T] \subseteq B$ }
        for each elementary condition $t \in T$ **do**
            if $[T - t] \subseteq B$ then $T := T - \{t\}$; {test a rule minimality}
        $\mathcal{T} := \mathcal{T} \cup \{T\}$; {store a rule}
        $G := B - \bigcup_{T \in \mathcal{T}} [T]$; {remove already covered examples}
    end; { while $G \neq \emptyset$ }
    for each $T \in \mathcal{T}$ do
        if $\bigcup_{T' \in \mathcal{T} - T} [T'] = B$ **then** $\mathcal{T} := \mathcal{T} - T$ {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
    $best\_t := \emptyset$;
    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 $(c \geq v)$;
            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∈[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∈[1050,2550)) then (Dec=p) [2]
6. if (Job=e) then (Dec=p) [1]
7. if (Age=sr) and (Period≥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 gilled 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.

Applying rule set to classify objects

- **Matching** 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 rule set 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

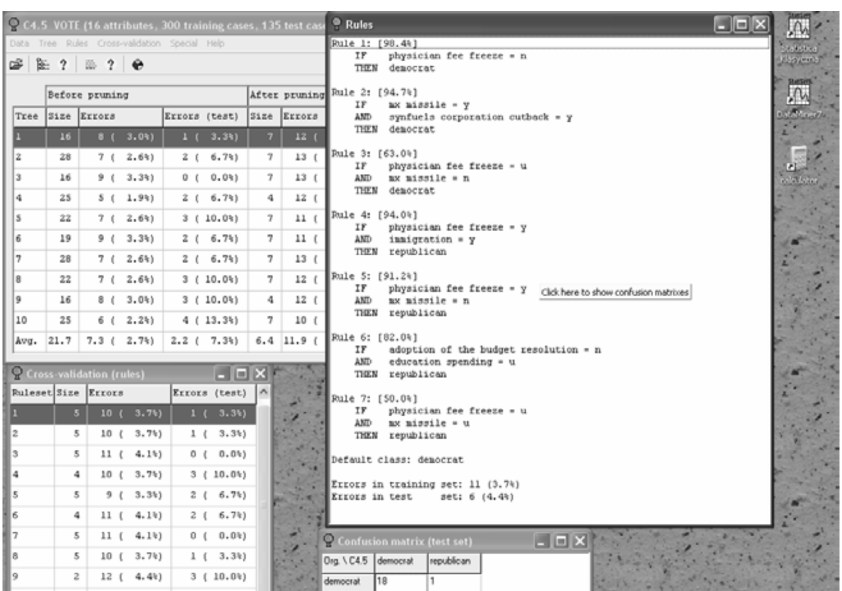
But: ordered execution when classifying a new instance: rules are sequentially tried and the first rule that ‘fires’ (covers the example) is used for final decision

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

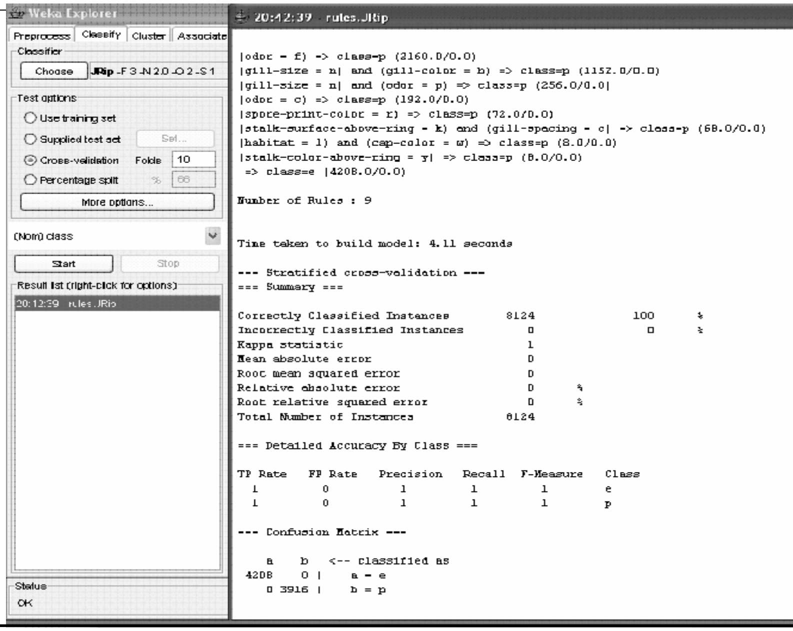
interpreted as **if-then-else** rules

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

Priority decision list (C4.5 rules)



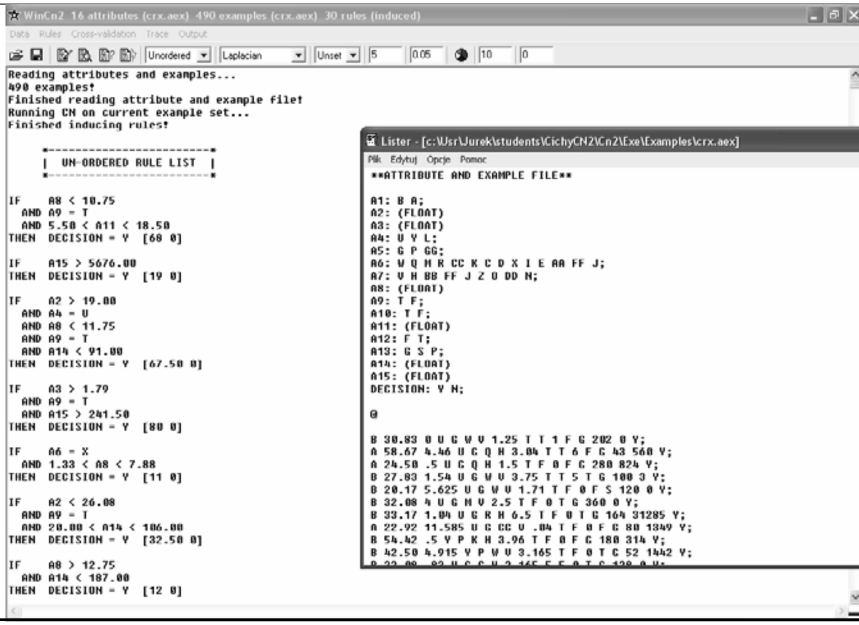
Specific solution RIPPER (Mushroom data)



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 → three situations:
  - Matching to rules indicating the same class.
  - Multiple matching to rules from different classes.
  - No matching to any rule.
- <u>An example:</u>
- 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. → 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 → 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!

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 ← ∅;

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

Q ← *LS;* {create a queue *Q*}

while *Q* ≠∅ **do**

begin

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

Q← *Q*\{*C*};

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

Good_Candidates(*LC*,*R*);

Q ← *Q*∪*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 $c_{h+i} \in LS$ 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

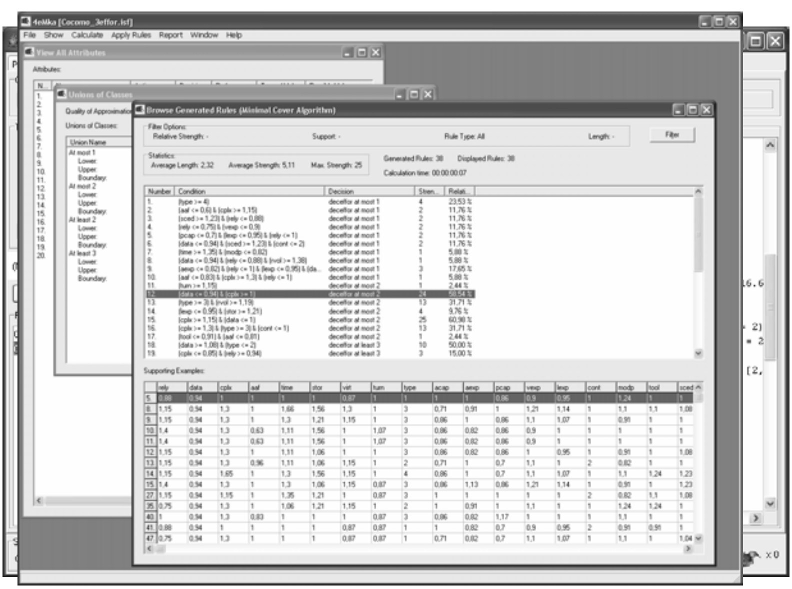
No.	q_1	q_2	q_3	q_4	q_5	q_6	d
1	2	3	1	3	1	2	1
2	2	3	1	1	1	1	1
3	2	3	1	3	2	1	1
4	2	1	1	1	1	1	1
5	2	2	1	1	2	2	1
6	1	3	2	3	1	2	1
7	1	3	2	3	2	1	1
8	2	1	2	1	2	2	1
9	3	1	1	3	1	2	2
10	3	1	2	2	2	1	2
11	3	1	1	3	2	2	2
12	2	1	2	1	2	1	2
13	3	2	4	2	1	1	2
14	3	2	4	2	2	1	2
15	2	2	3	2	1	2	2
16	2	2	2	1	1	1	1
17	2	2	2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

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		>300000	---	---	---
	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.

Our Software (PUT Poznań)



More about applications - see

Applications of Machine Learning and Rule Induction

Pat Langley*

Robotics Laboratory, Computer Science Dept.
Stanford University, Stanford, CA 94305

Herbert A. Simon

Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

An important area of application for machine learning is in automating the acquisition of knowledge bases required for expert systems. In this paper, we review the major paradigms for machine learning, including neural networks, instance-based methods, genetic learning, rule induction, and analytic approaches. We consider rule induction in greater detail and review some of its recent applications, in each case stating the problem, how rule induction was used, and the status of the resulting expert system. In closing, we identify the main stages in fielding an applied learning system and draw some lessons from successful applications.

Introduction

Machine learning is the study of computational methods for improving performance by mechanizing the acquisition of knowledge from experience. Expert performance requires much domain-

- P.Langley, H.Simon paper in Michalski, Bratko, Kubat book on Machine Learning and Data Mining

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?

?