W8: REGUŁY DECYZYJNE

Nguyen Hung Son

Tematy

- Reguły decyzyjne: prosty opis klas decyzyjnych
- Problem wyszukiwania reguł decyzyjnych
- Algorytmy generowania reguł decyzyjnych
  - Sekwencyjne pokrywanie
  - Algorytm AQ
- Problemy badawcze

Klasyfikatory regułowe

- ***Reguły decyzyjne*** mają postać:

$$r : (Warunek) \rightarrow (Teza)$$

  - Warunek: koniunkcja testów na atrybutach
  - Teza: klasa decyzyjna
- ***Zbiór reguł:***

$$R = \{r_1, r_2, \ldots, r_n\}$$

- ***Jak przebiega klasyfikacja nowych obiektów (przypadków)?***

Pokrycie regułami

- Reguła *r* ***pokryje*** obiekt *x* jeśli wartości jego atrybutów spełniają warunek reguły
- Np.
  - Reguła:

    r : (Age < 35) $\wedge$ (Status = Married) $\rightarrow$ Cheat=No
  - Obiekty:

    x_1 : (Age=29, Status=Married, Refund=No)

    x_2 : (Age=28, Status=Single, Refund=Yes)

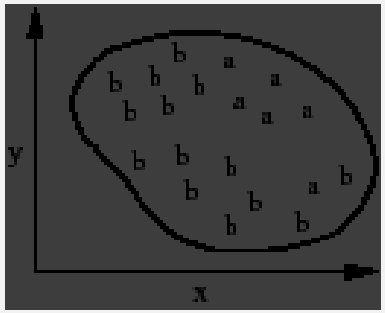
    x_3 : (Age=38, Status=Divorced, Refund=No)
  - Tylko x_1 jest pokryty przez regułę *r*

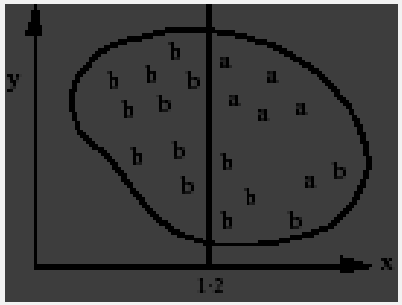
Klasyfikacja nowych przypadków za pomocą zbioru reguł

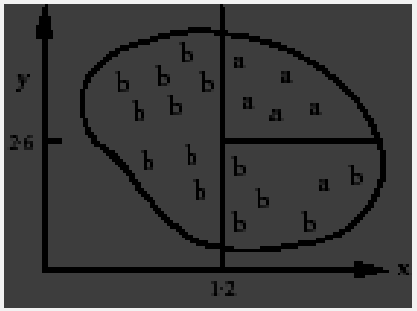
- **Reguły nie są rozłączne**
  - Obiekt może być pokryty przez kilka reguł
- **Strategie rozwiązywania problemu:**
  - Wymuszenie rozłączności reguł;
    - Np. reguły wydobywane z drzew decyzyjnych
  - Uporządkowanie reguł
    - Reguły są uporządkowane według ich priorytetów
  - Głosowanie

- **Reguły nie pokrywają wszystkich przypadków**
- **Strategie:**
  - Dodanie reguły domniemanej
    $r_d : ( ) \rightarrow y_d$
    do zbioru reguł

Przykład: reguły decyzyjne








```
If true then class = a
```

```
If x > 1.2 then class = a
```

```
If x > 1.2 and y > 2.6 then class = a
```

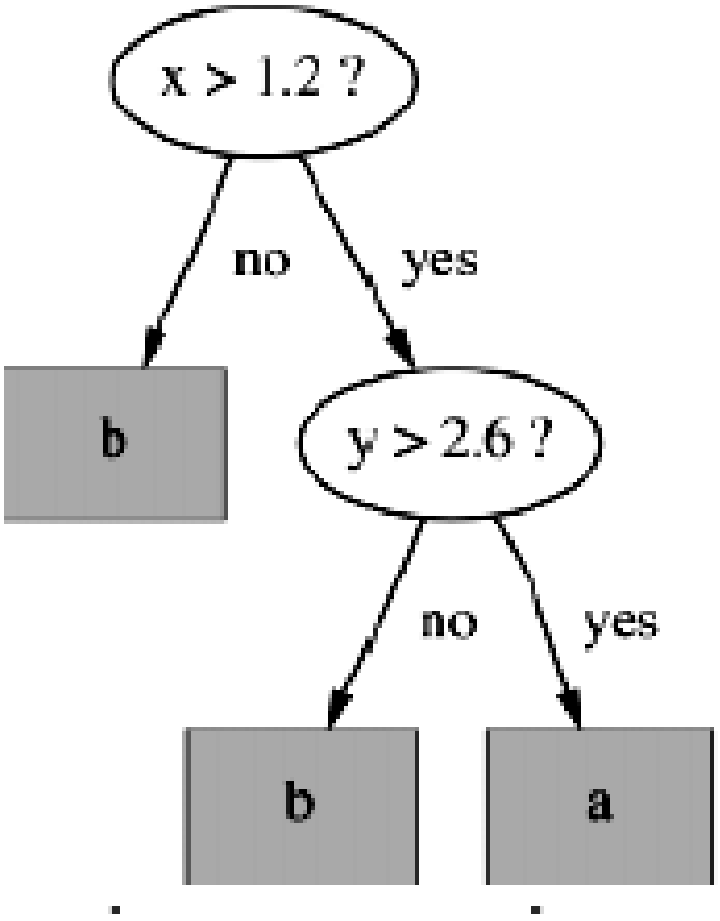
- Possible rule set for class “b”:

  ```
  If x ≤ 1.2 then class = b
  If x > 1.2 and y ≤ 2.6 then class = b
  ```

- More rules could be added for “perfect” rule set

Reguły decyzyjne vs. drzewo decyzyjne

- Drzewo decyzyjne to zbiór reguł rozłącznych.
- **Ograniczenie:** Metody drzew decyzyjnych są mniej wrażliwe dla danych o wielu klas decyzyjnych.



DEFINICJE I PROBLEMY

Reprezentacja reguł

- Logiczna postać reguły

$$R := (a_1 \in V_1) \wedge \ldots \wedge (a_k \in V_k) \rightarrow (decyzja = d)$$

- Rodzaje reguł
  - standardowe: $|V_i| = 1$

$$R := (a_1 = v_1) \wedge \ldots \wedge (a_k = v_k) \rightarrow (decyzja = d)$$

  - uogólnione:
    - $V_i = \{v_1, v_2, \ldots, v_m\}$

    (pref. dla atrybutów symbolicznych)
    - $V_i = (a, b) \subseteq R$

    (pref. dla atrybutów rzeczywistych)

Definicje

$$\boldsymbol{R}\text{: } (a_1 \in V_1) \wedge \ldots \wedge (a_k \in V_k) \rightarrow (decyzja = d)$$

- **Pokrycie:**
  - Obiekt *x spełnia* deskryptor $(a_i \in V_i)$ wtw, gdy
    $$(a_i(x) \in V_i).$$
  - Obiekt jest *pokryty* przez regułę ***R***, jeśli spełnia wszytkie deskryptory w ***R***.
- **Długość:** liczba deskrytorów w ***R***
- **Wsparcie:** liczba obiektów pokrytych przez ***R*** (ozn. *support*(***R***)).

Definicje (c.d.)

R: *warunek* $\rightarrow$ (*decyzja = d*)

- **Pozytywny przykład**: obiekt, który spełnia *warunek* i ma decyzję równą *d*.
- **Negatywne przykład**: obiekt, który spełnia *warunek*, ale ma decyzję różną od *d*.
- *p*(***R***): liczba pozytywnych przykładów
- *support*(***R***): wsparcie ***R***
- ***Dokładność reguły*** (*ang. accuracy ratio*):

$$accuracy(\boldsymbol{R}) = p(\boldsymbol{R})/support(\boldsymbol{R})$$

Jakość reguły decyzyjnej

- Funkcja jakości reguł mają postać:

$$Quality(\boldsymbol{R}) = F(accuracy(\boldsymbol{R}), support(\boldsymbol{R}))$$

gdzie F jest funkcją rosnącą wzg. każdej zmiennej

- Przykłady funkcji jakości:
  - $Quality_1(R) = accuracy\ (R)$
  - $Quality_2(R) = accuracy(R) \times Log(support(P))$
- $Quality_1(\boldsymbol{R})$ często jest zastosowane w praktyce.

Problemy wyszukiwania reguł decyzyjnych

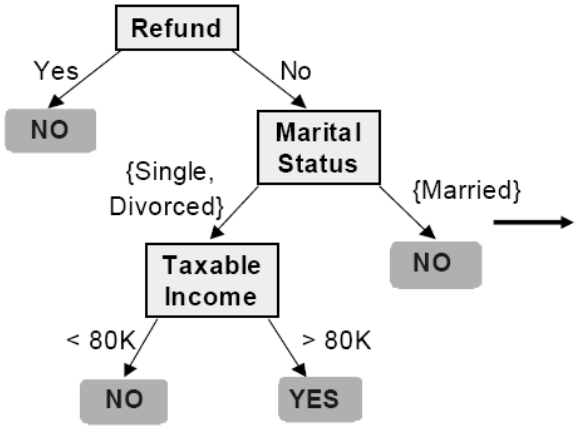
- Znaleźć *regułę* o maksymalnej dokładności.
- Znaleźć *regułę* o dokładności nie mniejszej niż acc_{min}.
- Znaleźć krótkie reguły.
- Znaleźć długie reguły.
- Znaleźć najmniejszy *zbiór reguł*, który klasyfikuje obiekty z dużą dokładnością

POSZUKIWANIE REGUŁ DECYZYJNYCH

Construction of a Rule Based Classifier from data

- Generate an initial set of rules
  - Direct Method:
    - Extract rules directly from data
    - Examples: RIPPER, CN2, Holte's 1R, Boolean reasoning
  - Indirect Method:
    - Extract rules from other classification methods (e.g. decision trees)
    - Example: C4.5 rules
- Rules are pruned and simplified
- Rules can be order to obtain a rule set *R*
- Rule set *R* can be further optimized

Indirect Method: Conversion from Decision Trees



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

- Rules are mutually exclusive and exhaustive
- Rule set contains as much information as the tree
- Rules can be simplified

$(\text{Refund}=\text{No}) \wedge (\text{Status}=\text{Married}) \rightarrow \text{No} \quad ==> \quad (\text{Status}=\text{Married}) \rightarrow \text{No}$

Indirect Method: C4.5 rules

- Creating an initial set of rules
  - Extract rules from an un-pruned decision tree
  - For each rule, $r : A \rightarrow y$
    - Consider alternative rules r': $A' \rightarrow y$, where A' is obtained by removing one of the conjuncts in A
    - Replace r by r' if it has a lower pessimistic error
    - Repeat until we can no longer improve the generalization error
- Ordering the rules
  - Instead of ordering the rules, order subsets of rules
    - Each subset is a collection of rules with the same consequent (class)
    - The subsets are then ordered in the increasing order of

      Description length = L(exceptions) + $g \cdot L$(model)
    - where g is a parameter that takes in to account the presence of redundant attributes in a rule set. Default value is 0.5

Algorytmy

- Można adaptować algorytmy wyszukiwania szablonu z ewentualną modyfikacją.
- Istniejące dwie strategie:
  - przeszukiwania przez „wydłużenia”
  - przeszukiwanie przez „skracanie”
- Dla każdej strategii są dwie metody
  - metoda globalna
  - metoda lokalna

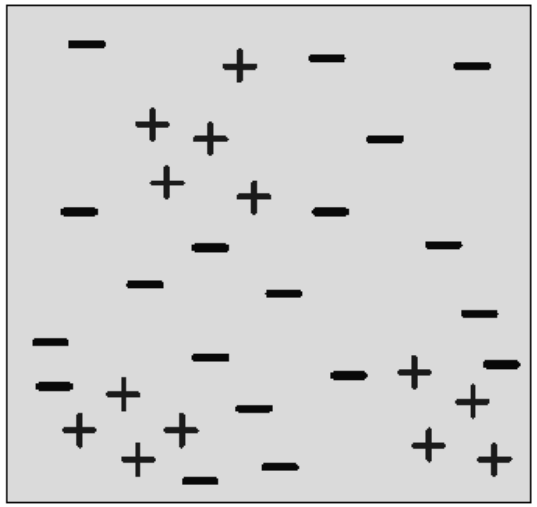
Sekwencyjne pokrywanie - metoda globalna

- **Wejście**: *T* - tablica treningu;
- **Wyjście**: *R* - (stadardowa) reguła decyzyjna o maksymalnej dokładności;
- **Schemat**:

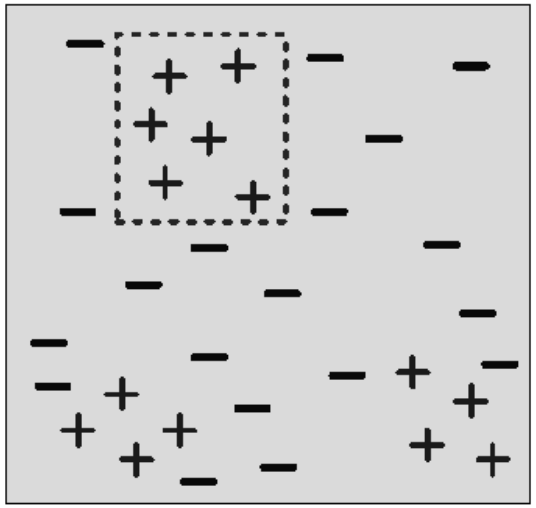
  1: R = ∅;

  2: Generuj R przez dodawanie do niej nowego deskryptora, który maksymalizuje *Quality*(*R*);

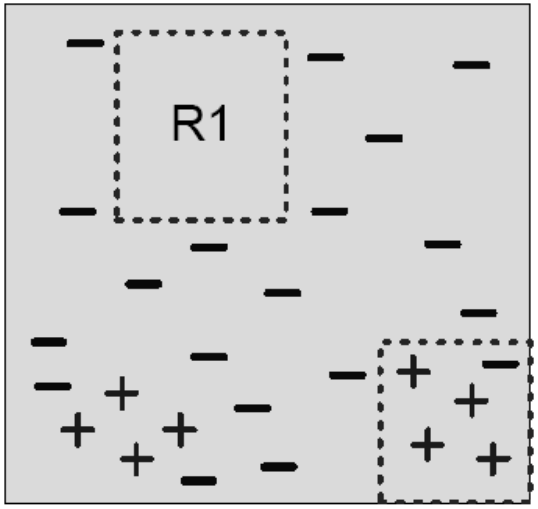
  3: Powtórz krok 2 dopóki warunek *stopu* zachodzi.



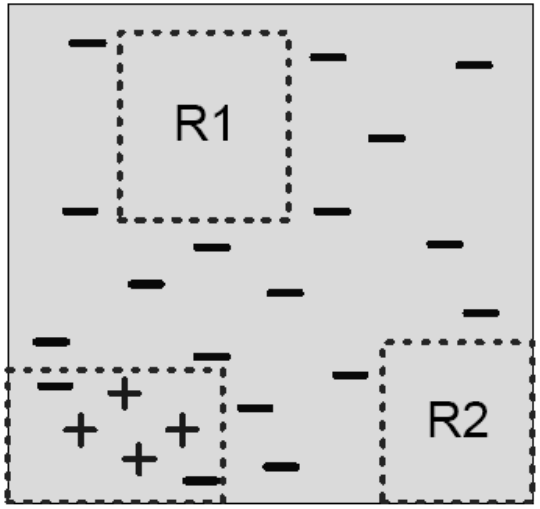
(i) Original Data



(ii) Step 1



(iii) Step 2



(iv) Step 3

Algorytm

- **Wejście:** $T = (U, A)$; d - decyzja dla reguły;
- **Wyjście:** stadardowa reguła $R = (P \rightarrow d)$;

1: $P_0 := \emptyset$; $k := 0$;

2. **While** (*not stop*()) **do**

3: $D_{k+1} = znajdź\_deskryptor(T, P_k, d)$;

4: $P_{k+1} := P_k \wedge D_{k+1}$;

5: $k := k+1$;

7: **end;**

9: return $R = P_n \rightarrow d$

Warunek stopu

- Jeśli celem jest generowanie reguły R o maksymalnej dokładności:
  - $acccuracy(R) = 1$ lub
  - dokładności nie da się poprawiać
- Jeśli celem jest generowanie reguły R o dokładności niemniejszej niż acc_{min}
  - $acccuracy(R) \geq acc_{min}$ lub
  - dokładności nie da się poprawiać

Wybór deskryptora

- Funkcja *znajdź_deskryptor*(T, P, d) zwraca deskryptor $D = (a = v)$, który najlepiej wydłuży szablon P, t.j. maksymalizuje $Quality(P \wedge D)$.
- Jeśli celem jest generowanie reguły o maksymalnej dokładności, to

$$Quality(P \wedge D) = |pos(P \wedge D)| / support(P \wedge D)$$

$pos(P \wedge D)$: zbiór obiektów pokrywanych przez $(P \wedge D)$ i mających decyzję d.

Przykład: generowanie reguły

Example: contact lenses data

- Rule we seek: `If ? then recommendation = hard`
- Possible tests:

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

Modified rule and resulting data

- Rule with best test added:

```
If astigmatics = yes then recommendation = hard
```

- Instances covered by modified rule:

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

Przykład: generowanie reguły (c.d.)

Further refinement

- Current state: `If astigmatism = yes and ? then recommendation = hard`
- Possible tests:

`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

Modified rule and resulting data

- Rule with best test added:

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

- Instances 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

- Current state:

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

- Possible tests:

```
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

ALGORYTM AQ

Algorytm AQ - Metoda lokalna

- **Idea:** Algorytm generuje regułę R, która pokrywa wybrany obiekt x_s (zwany *ziarnem* lub *jądrem*) i tylko obiekty o decyzji takiej samej jak x_s.
- *Negatywne ziarno*: obiekt, który spełnia regułę R i ma decyzję różną od decyzji x_s.
- Deskryptor $(a = v)$ odróżnia x_s od x_n jeśli

$$(a(x_s) = v) \text{ i } (a(x_n) \neq v)$$

Algorytm AQ

- **Dane:** T - tablica treningu;
- **Szukanee:** $R = (P \rightarrow d)$ - reguła decyzyjna o maksymalnej dokładności;

1. *Wybierz* ziarno x_s;
2. $P_0 = \emptyset$; $covered(R) = T$; $k = 0$;
3. **While** (*not stop*()) **do**
4. *Wybierz* negatywne ziarno $x_n \in covered(R)$;
5. $D_{k+1} = znajdź\_deskryptor(T, P_k)$;
4: $P_{k+1} = P_k \wedge D_{k+1}$;
5: $k := k+1$;
6. Usuń z $covered(R)$ obiekty niespełniające R
7. **endwhile**

Warunek stopu

- Algorytm kończy się, gdy wśród obiektów pokrywanych przez R nie istnieje obiekt o różnej decyzji niż decyzję x_s
- Formalny zapis:

$$\exists x : x \in covered(R) \wedge d(x) \neq d(x_s)$$

Wybór ziaren

- Ziarno pozytywnwe:
  - reprezentatywne dla klasy (podobne do więszości obiektów z klasy)
  - zróżnicowane od istniejących ziaren (w przypadku generowania zbioru reguł)
- Ziarno negatywne:
  - podobne do ziarna pozytywnego
- Odległość Hamminga lub (modyfikowana) odległość Euklidesowa.

Wybór deskryptora

function *znajdź_deskryptor* (P,x_s,x_n)

1. $K = \emptyset$;
2. **forall** (*atrybut a, który nie występuje w P*) **do**
3. **forall** ($v \in V_a$) **do**
   if (($a = v$) odróżnia x_s od x_n) **then**
4. $K = K \cup (a = v)$;
5. D = *najlepszy deskryptor* w K;
6. **return** D;

Wybór deskryptora (c.d.)

- Deskryptora D dla bieżącej reguły *R* jest dobry, jeśli reguła po wydłużeniu o D:
  - pokrywa dużo obiektów o decyzji zgodnej z x_s
  - pokrywa mało obiektów o decyzji różnej od x_s.
- **Kryterium oceny:**

$$w_{=} = |x \in covered(R): d(x) = d(x_s)|$$

$$w_{\neq} = |x \notin covered(R): d(x) \neq d(x_s)|$$

$$Quality\ (R, x_s) = w_{=} + w_{\neq}$$

Własność algorytmu AQ

- Algorytm można łatwo modyfikować, żeby otrzymać *zbiór m najlepszych reguł* pokrywających wybrane ziarno (*przeszukiwanie wiązkowe*).
- Algorytm AQ generuje dokładne reguły (accuracy = 100%).

Problem: nadmiar dopasowania (overfitting)

Generowanie zbioru reguł

- Cel: Generować zbiór reguł, który
  - pokryje wszystkie przykłady w tablicy trenującej,
  - dobrze klasyfikuje obiekty w zbiorze trenującym,
  - ma mały rozmiar.

Algorytm

- **Wejście:** *T* - tablica trenująca;
- **Wyjście:** **S** - zbiór reguł pokrywających *T*;

1.**S** = ∅;

2.**while** (istnieje obiekt *x* niepokrywany przez **S**) **do**

3. *Znajdź optymalną regułę R;*

4. Dodaj *R* do **S;**

5. Usuń z T obiekty pokrywane przez **S;**

6. **end;**

7. **return S;**

INNE ALGORYTMY

Learn one rule (1R)

- The objective of this function is to extract the best rule that covers the current set of training instances
  - What is the strategy used for rule growing
  - What is the evaluation criteria used for rule growing
  - What is the stopping criteria for rule growing
  - What is the pruning criteria for generalizing the rule

Learn One Rule: Rule Growing Strategy

- General-to-specific approach
  - It is initially assumed that the best rule is the empty rule, $r : \{\} \rightarrow y$, where y is the majority class of the instances
  - Iteratively add new conjuncts to the LHS of the rule until the stopping criterion is met
- Specific-to-general approach
  - A positive instance is chosen as the initial seed for a rule
  - The function keeps refining this rule by generalizing the conjuncts until the stopping criterion is met

Rule Evaluation and Stopping Criteria

- Evaluate rules using rule evaluation metric
  - Accuracy
  - Coverage
  - Entropy
  - Laplace
  - M-estimate
- A typical condition for terminating the rule growing process is to compare the evaluation metric of the previous candidate rule to the newly grown rule

Learn 1R

- Rule Pruning
  - – Each extracted rule can be pruned to improve their ability to generalize beyond the training instances
  - Pruning can be done by removing one of the conjuncts of the rule and then testing it against a validation set
- Instance Elimination
  - – Instance elimination prevents the same rule from being generated again
  - Positive instances must be removed after each rule is extracted
  - Some rule based classifiers keep negative instances, while some remove them prior to generating next rule

RIPPER

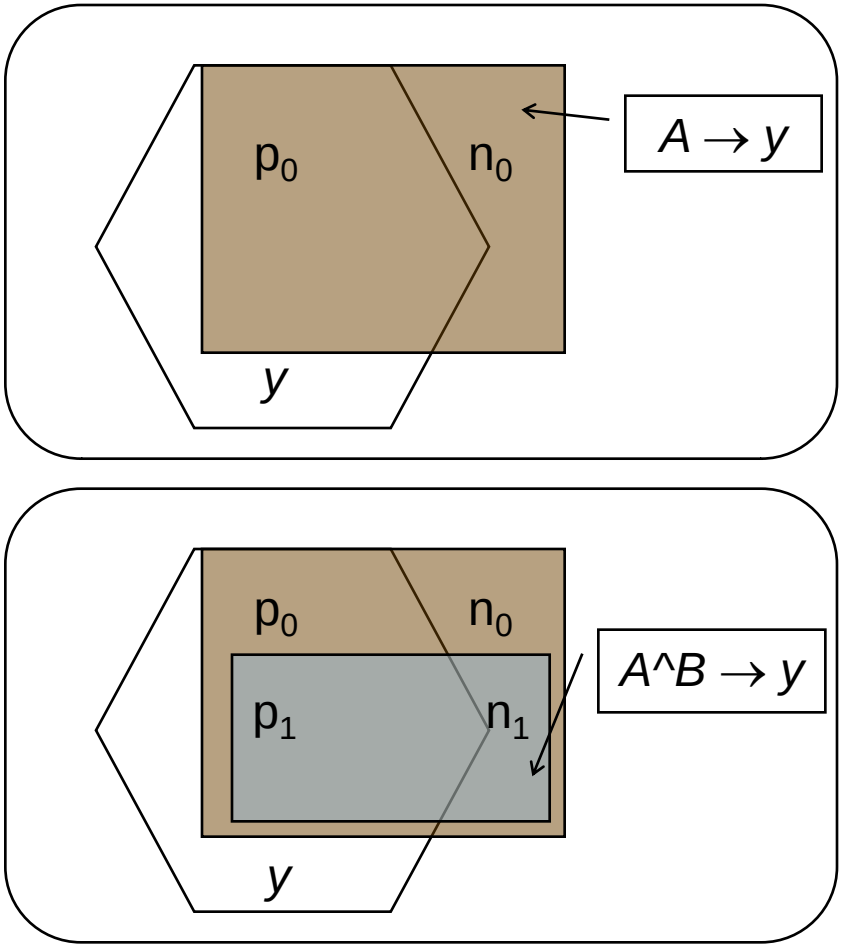
- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn rules for positive class
  - Negative class will be default class
- For multi-class problem
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class

Foil's Information Gain

- Compares the performance of a rule before and after adding a new conjunct
- Foil's information gain is defined as

  $= t \cdot [\ log_2(p_1/(p_1 + n_1)) - log_2(p_0/(p_0 + n_0))\ ]$

  where t is the number of positive instances covered by both r and r'



Direct Method: RIPPER

- Growing a rule:
  - Start from empty rule
  - Add conjuncts as long as they improve Foil's information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning: $v = (p - n) / (p + n)$
    - p: number of positive examples covered by the rule in the validation set
    - n: number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes v

RIPPER: Building a Rule Set

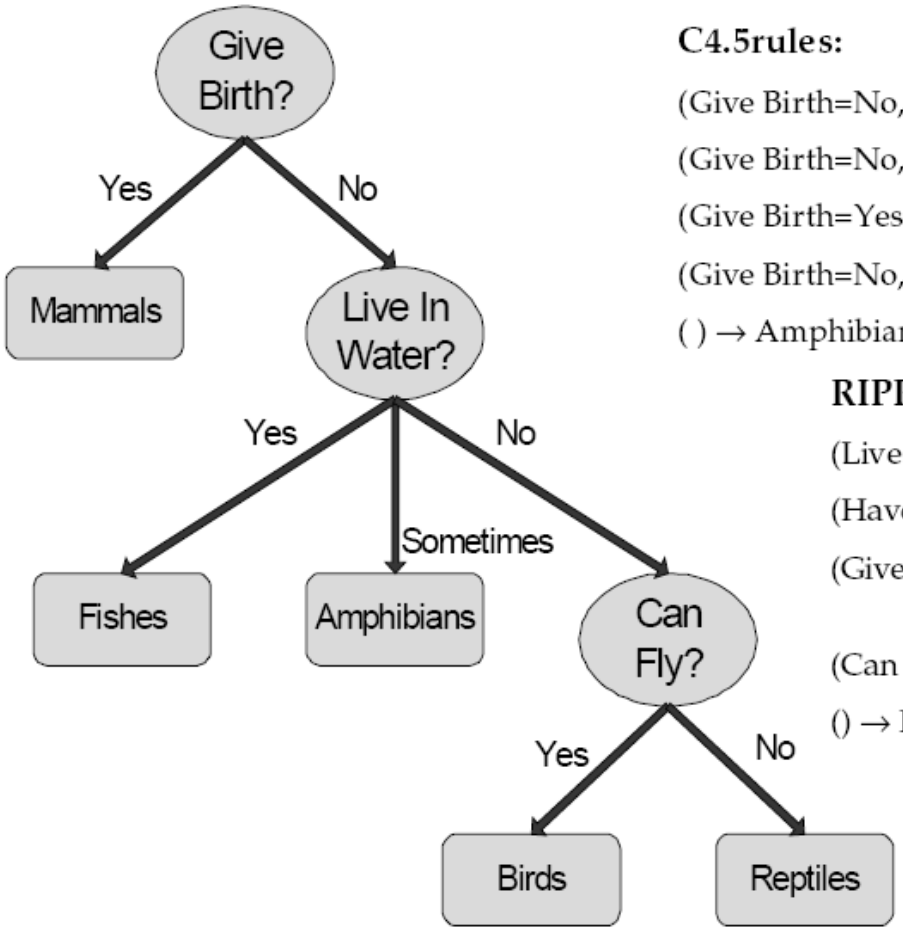
- Use sequential covering algorithm
  - Finds the best rule that covers the current set of positive examples
  - Eliminate both positive and negative examples covered by the rule
- Each time a rule is added to the rule set, compute the description length
  - Stop adding new rules when the new description length is *d* bits longer than the smallest description length obtained so far. *d* is often chosen as 64 bits

RIPPER: Optimize the rule set:

- For each rule *r* in the rule set ***R***
  - Consider 2 alternative rules:
    - Replacement rule (*r**): grow new rule from scratch
    - Revised rule (*r'*): add conjuncts to extend the rule *r*
  - Compare the rule set for *r* against the rule set for *r** and *r'*
  - Choose rule set that minimizes MDL principle
- Repeat rule generation and rule optimization for the remaining positive examples

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds

C 4.5 rules vs. RIPPER



C4.5rules:

(Give Birth=No, Can Fly=Yes) → Birds

(Give Birth=No, Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles

( ) → Amphibians

RIPPER:

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles

(Can Fly=Yes,Give Birth=No) → Birds

() → Mammals

Problemy

- Uproszczenie zbioru reguł
  - usuwanie zbędnych reguł
  - usuwanie zbędnych deskryptorów w regule
- Atrybuty rzeczywiste:
  - Problem generowania uogólnionych reguł
- Reguły dla danych zmieniających się dynamicznie.

Bibliografia

- Michalski R. S., Mozetic I.,Hong J., Lavrac N.(1986): *The multi-purpose incremental learning system AQ15 and its testing application to three medical domain*. Proc. of the 5-th National Conference on AI (AAAI-86).
- Hoa S. Nguyen, H. Son Nguyen (1998). *Pattern extraction from data.* Fundamenta Informaticae **34/1-2**, pp. 129-144.