

Rule Based Classification

Rule-Based Classifier

- Classify records by using a collection of “if...then...” rules
- Rule: $(Condition) \rightarrow y$
 - where
 - *Condition* is a conjunctions of attributes
 - y is the class label
 - *LHS*: rule antecedent or pre condition
 - *RHS*: rule consequent
- Rules : $R=(R1 \vee R2 \vee \dots \vee Rk)$
 - Examples of classification rules:
 - $(\text{Blood Type}=\text{Warm}) \wedge (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}$
 - $(\text{Taxable Income} < 50\text{K}) \wedge (\text{Refund}=\text{Yes}) \rightarrow \text{Evade}=\text{No}$

Rule-based Classifier (Example) for Vertebrate Classification Problem

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

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians

Application of Rule-Based Classifier

- A rule r **covers** a record x if the attributes of the record satisfy the condition of the rule. Rule r is also said to be **triggered or fired** whenever it covers a given record.

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk \Rightarrow Bird

The rule R3 covers the grizzly bear \Rightarrow Mammal

Rule Coverage and Accuracy

- Coverage of a rule:
 - Fraction of records that satisfy the antecedent of a rule
- Accuracy of a rule:
 - Fraction of records that satisfy both the antecedent and consequent of a rule

<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal

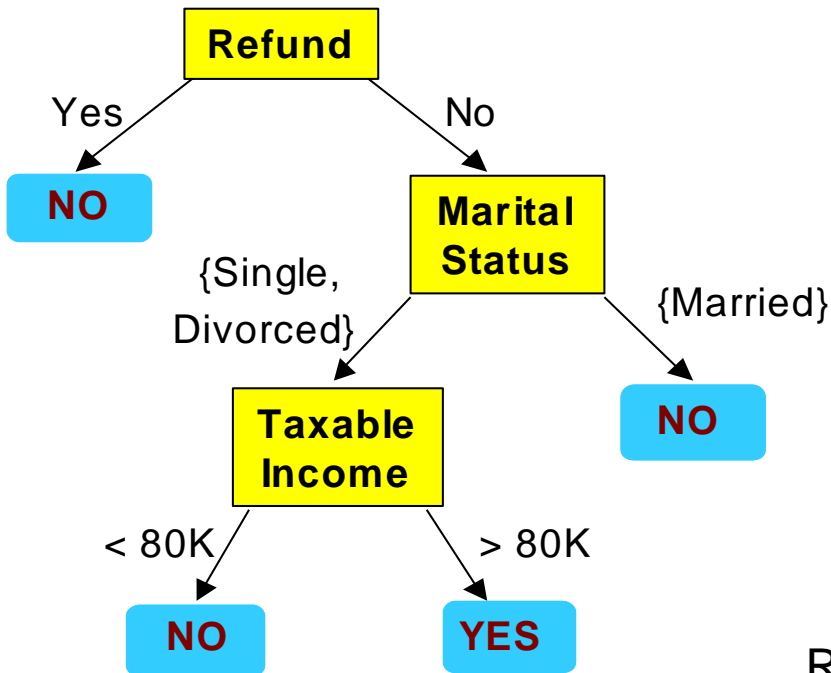
A turtle triggers both R4 and R5

A dogfish shark triggers none of the rules

Characteristics of Rule-Based Classifier

- Mutually exclusive rules
 - Classifier contains mutually exclusive rules if no two rules are triggered by the same record.
 - Every record is covered by at most one rule
- Exhaustive rules
 - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
 - Each record is covered by at least one rule

From Decision Trees To Rules



Classification Rules

(Refund=Yes) ==> No

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

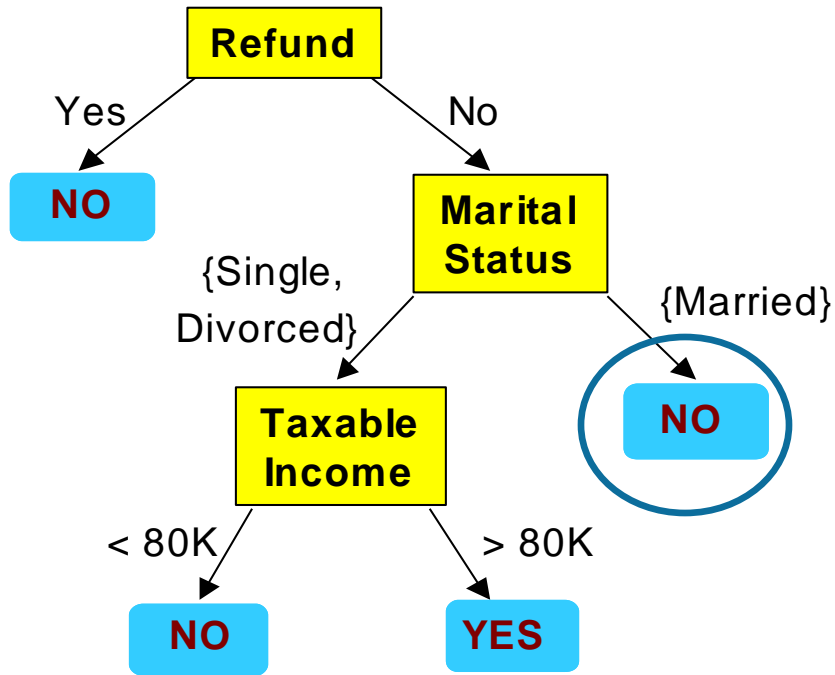
(Refund=No, Marital Status={Single, Divorced}, Taxable Income>80K) ==> Yes

(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



<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Initial Rule: $(\text{Refund}=\text{No}) \wedge (\text{Status}=\text{Married}) \rightarrow \text{No}$

Simplified Rule: $(\text{Status}=\text{Married}) \rightarrow \text{No}$

Effect of Rule Simplification

- Rules are no longer mutually exclusive
 - A record may trigger more than one rule
 - Solution?
 - Ordered rule set
 - Unordered rule set – use voting schemes
- Rules are no longer exhaustive
 - A record may not trigger any rules
 - Solution?
 - Use a default class

Ordered Rule Set

- Rules are rank ordered according to their priority
 - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
 - It is assigned to the class label of the highest ranked rule it has triggered
 - If none of the rules fired, it is assigned to the default class

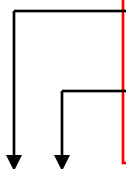
R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians



Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Rule Ordering Schemes

- Rule-based ordering
 - Individual rules are ranked based on their quality
- Class-based ordering
 - Rules that belong to the same class appear together

Rule-based Ordering

(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

Class-based Ordering

(Refund=Yes) ==> No

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

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

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

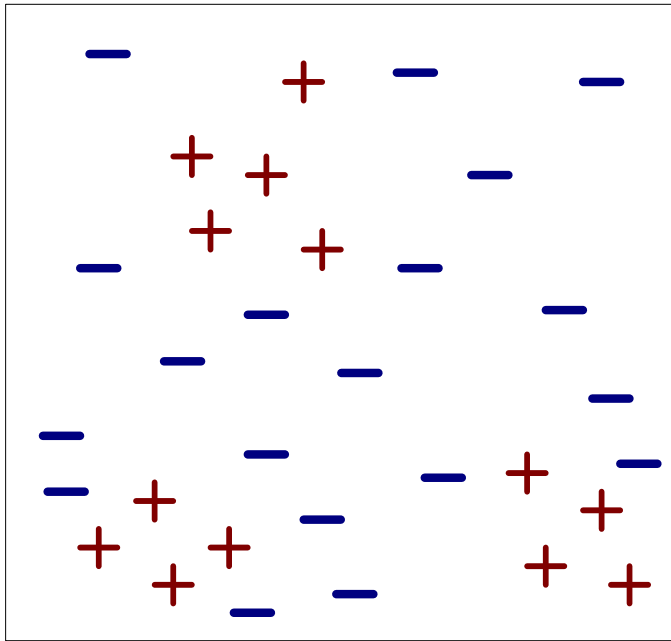
Building Classification Rules

- Direct Method:
 - Extract rules directly from data
 - e.g.: RIPPER, CN2, Holte's 1R
- Indirect Method:
 - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
 - e.g: C4.5rules

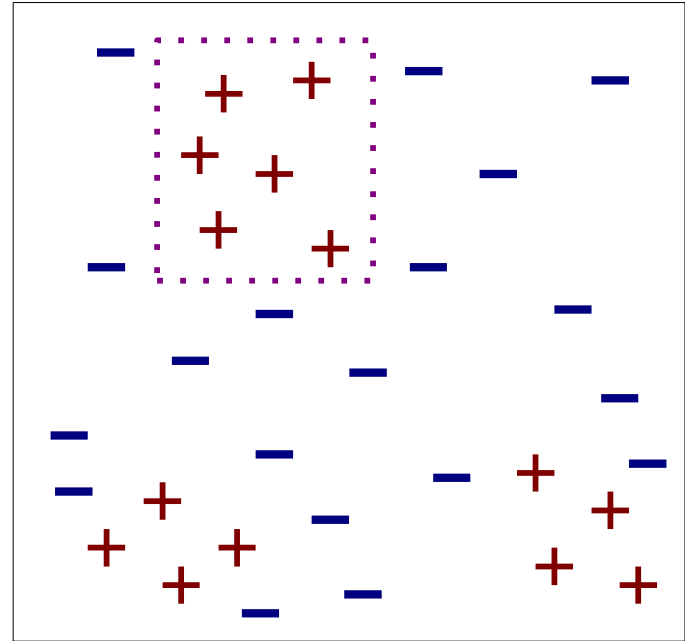
Direct Method: Sequential Covering

1. Start from an empty rule
2. Grow a rule using the Learn-One-Rule function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Example of Sequential Covering

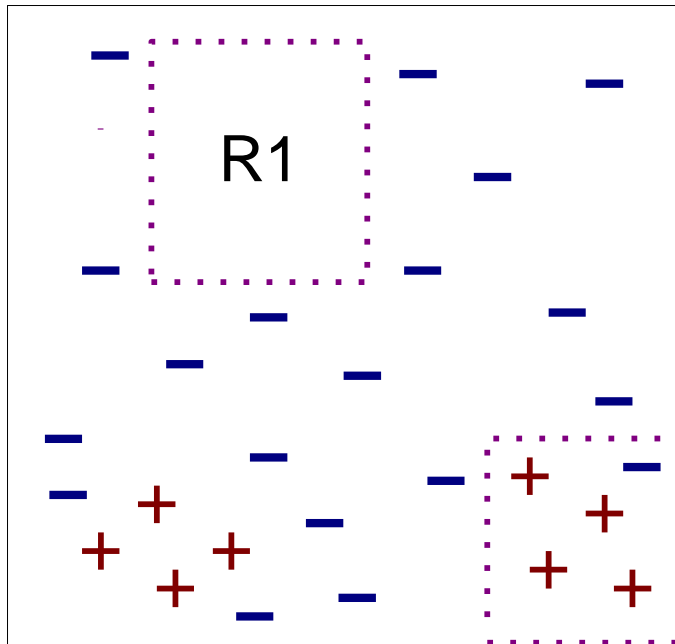


(i) Original Data

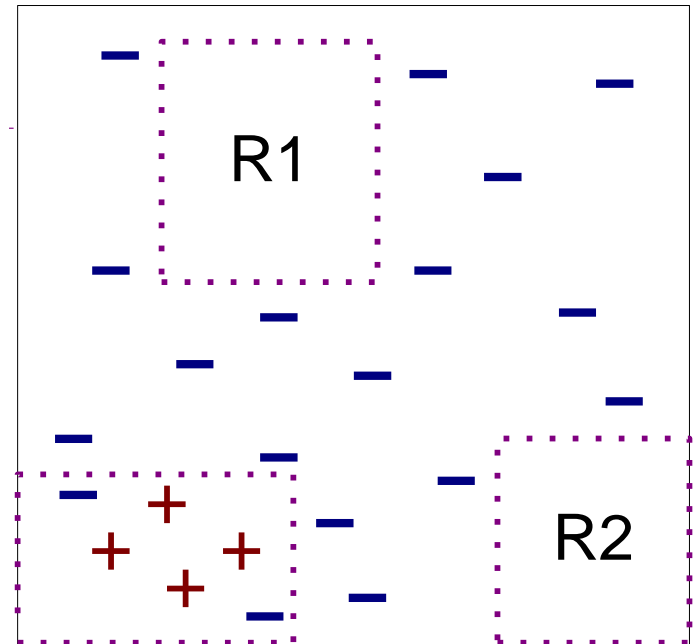


(ii) Step 1

Example of Sequential Covering...



(iii) Step 2



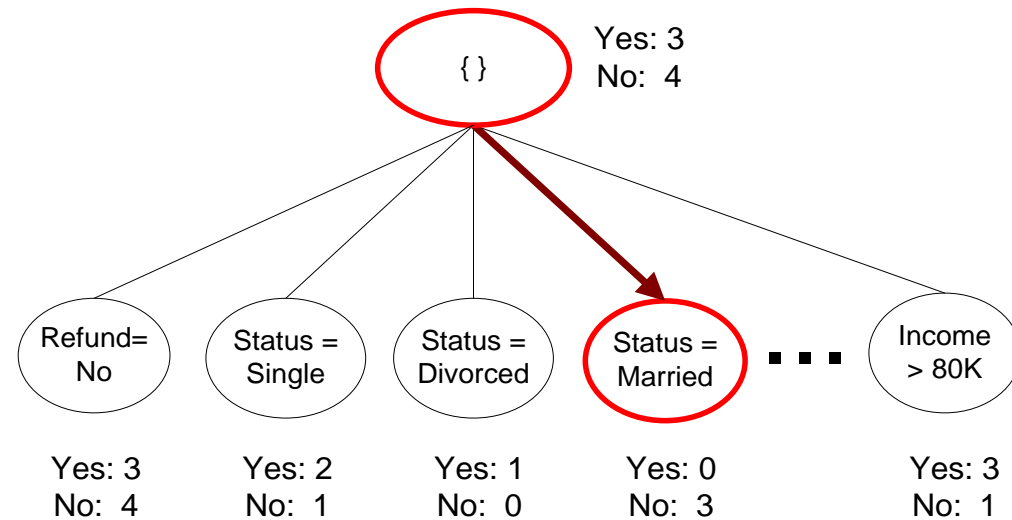
(iv) Step 3

Aspects of Sequential Covering

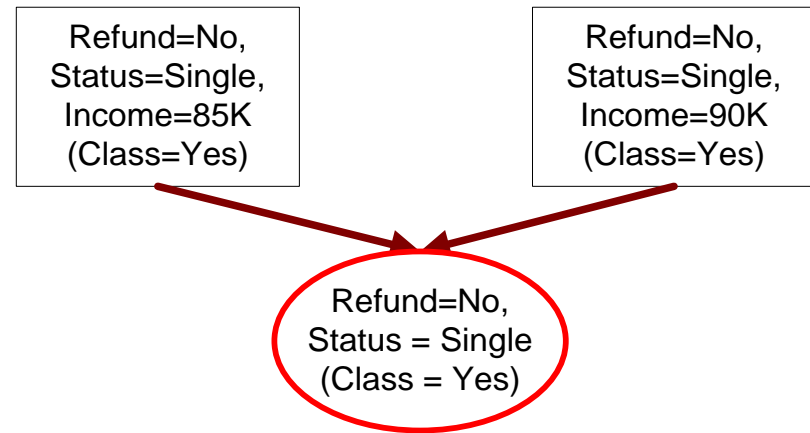
- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



(a) General-to-specific



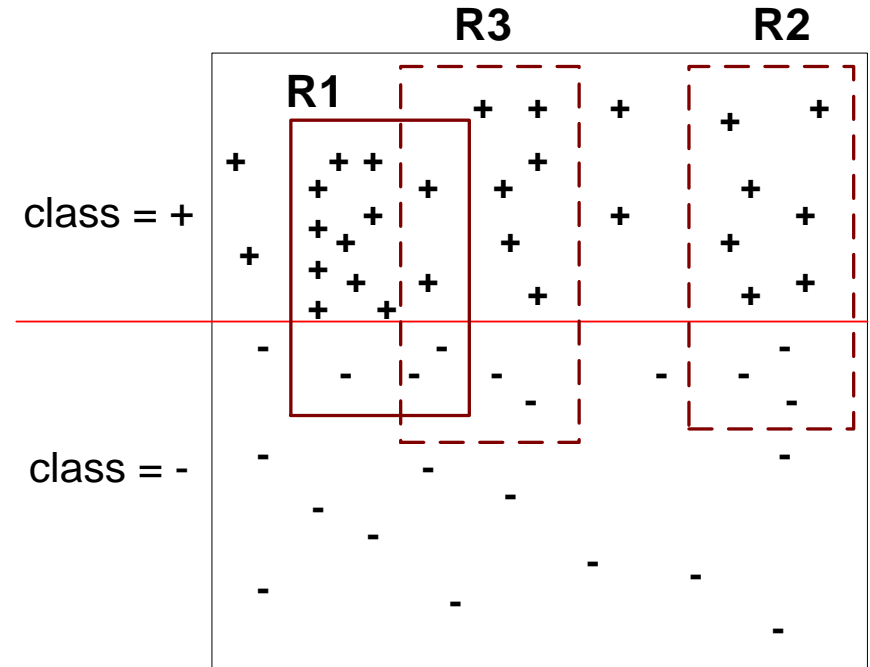
(b) Specific-to-general

Rule Growing (Examples)

- CN2 Algorithm:
 - Start from an empty conjunct: {}
 - Add conjuncts that minimizes the entropy measure: {A}, {A,B}, ...
 - Determine the rule consequent by taking majority class of instances covered by the rule
- RIPPER Algorithm:
 - Start from an empty rule: {} => class
 - Add conjuncts that maximizes FOIL's information gain measure:
 - $R0: \{\} \Rightarrow \text{class}$ (initial rule)
 - $R1: \{A\} \Rightarrow \text{class}$ (rule after adding conjunct)
 - $\text{Gain}(R0, R1) = t [\log(p1/(p1+n1)) - \log(p0/(p0 + n0))]$
 - where t : number of positive instances covered by both $R0$ and $R1$
 $p0$: number of positive instances covered by $R0$
 $n0$: number of negative instances covered by $R0$
 $p1$: number of positive instances covered by $R1$
 $n1$: number of negative instances covered by $R1$

Instance Elimination

- Why do we need to eliminate instances?
 - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
 - Ensure that the next rule is different
- Why do we remove negative instances?
 - Prevent underestimating accuracy of rule
 - Compare rules R2 and R3 in the diagram



Rule Evaluation

- Metrics:
$$= \frac{n_c}{n}$$
 - Accuracy

- Laplace
$$= \frac{n_c + 1}{n + k}$$

- M-estimate
$$= \frac{n_c + kp}{n + k}$$

n : Number of instances covered by rule

n_c : Number of instances covered by rule

k : Number of classes

p : Prior probability

Stopping Criterion and Rule Pruning

- Stopping criterion
 - Compute the gain
 - If gain is not significant, discard the new rule
- Rule Pruning
 - Similar to post-pruning of decision trees
 - Reduced Error Pruning:
 - Remove one of the conjuncts in the rule
 - Compare error rate on validation set before and after pruning
 - If error improves, prune the conjunct

Summary of Direct Method

- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

Direct Method: 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

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

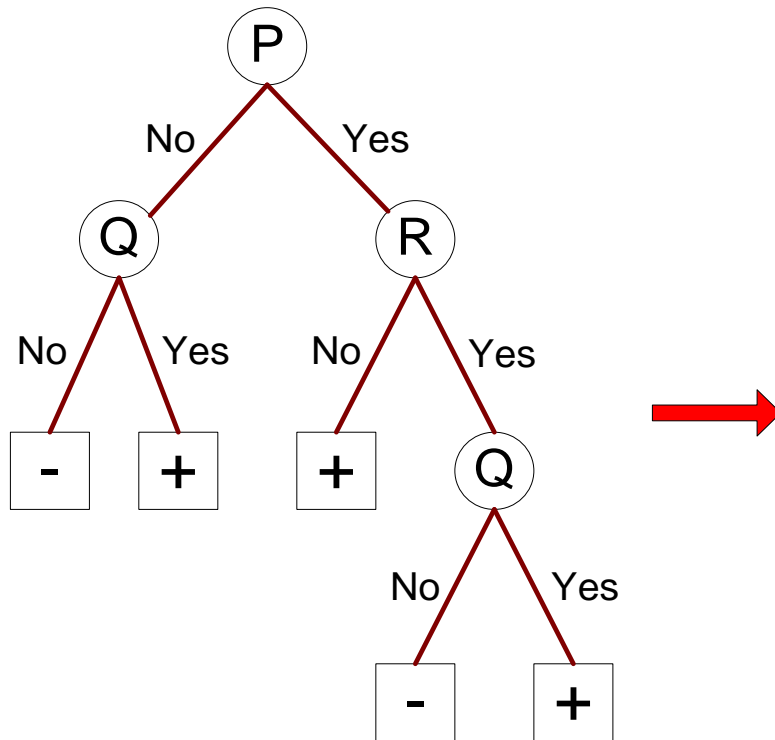
Direct Method: RIPPER

- 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 new description length
 - stop adding new rules when the new description length is d bits longer than the smallest description length obtained so far

Direct Method: 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

Indirect Methods



Rule Set

r1: (P=No,Q=No) ==> -

r2: (P=No,Q=Yes) ==> +

r3: (P=Yes,R=No) ==> +

r4: (P=Yes,R=Yes,Q=No) ==> -

r5: (P=Yes,R=Yes,Q=Yes) ==> +

Indirect Method: C4.5rules

- Extract rules from an unpruned decision tree
- For each rule, $r: A \rightarrow y$,
 - consider an alternative rule $r': A' \rightarrow y$ where A' is obtained by removing one of the conjuncts in A
 - Compare the pessimistic error rate for r against all r 's
 - Prune if one of the r 's has lower pessimistic error rate
 - Repeat until we can no longer improve generalization error

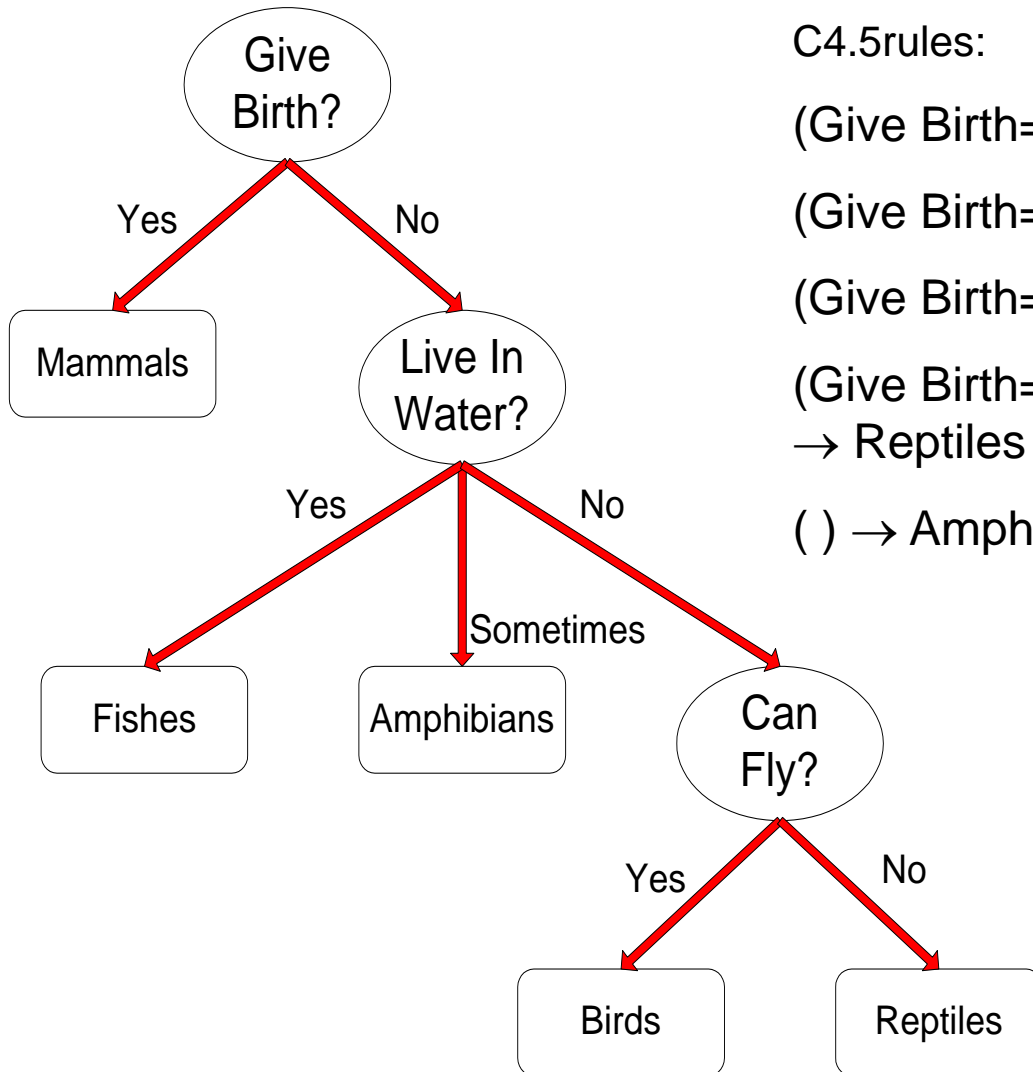
Indirect Method: C4.5rules

- Instead of ordering the rules, order subsets of rules (**class ordering**)
 - Each subset is a collection of rules with the same rule consequent (class)
 - Compute description length of each subset
 - Description length = $L(\text{error}) + g L(\text{model})$
 - g is a parameter that takes into account the presence of redundant attributes in a rule set (default value = 0.5)

Example

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

C4.5rules versus 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

RIPPER:

() → Amphibians

(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

C4.5rules versus RIPPER

C4.5 and C4.5rules:

		PREDICTED CLASS				
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL CLASS	Amphibians	2	0	0	0	0
	Fishes	0	2	0	0	1
	Reptiles	1	0	3	0	0
	Birds	1	0	0	3	0
	Mammals	0	0	1	0	6

RIPPER:

		PREDICTED CLASS				
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL CLASS	Amphibians	0	0	0	0	2
	Fishes	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees