Rule-Based Classifier

MSc. Bui Quoc Khanh

khanhbq@hanu.edu.vn

Classification Rules

- A rule-based classifier is a set of propositional rules of the form
  - IF outlook=sunny and Humidity=normal
    - THEN PlayTennis=yes
  - IF Humidity=normal and wind=strong
    - THEN PlayTennis=yes

Classification Rules

- Alternative notation
  - outlook=sunny and Humidity=normal ➔ yes
  - Humidity=normal and wind=strong ➔ yes
- Equivalent to DNF
- (Outlook=sunny and Humidity=normal) OR (Humidity=normal and wind=strong)

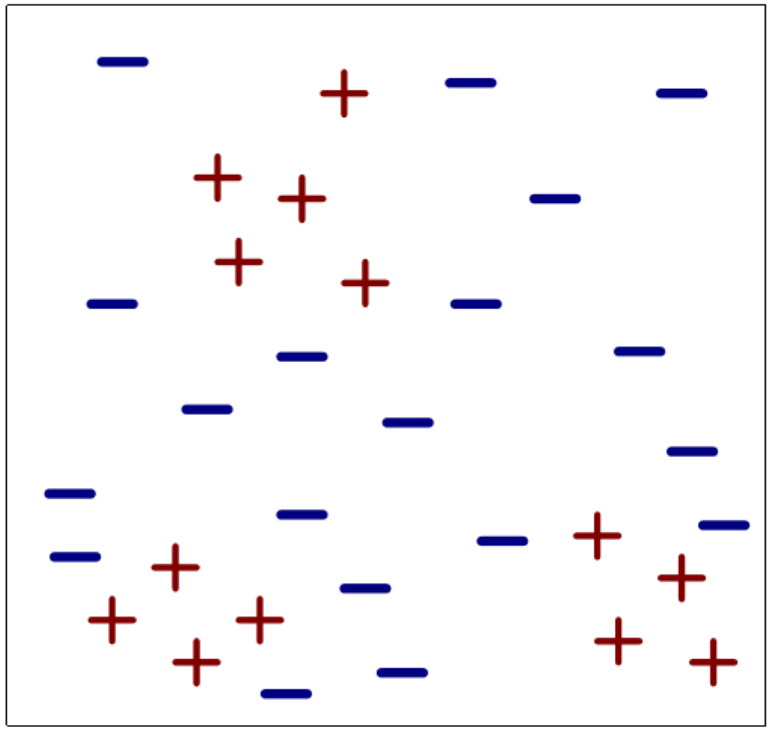
Classification Rules

- Rules can be generated
  - straight from training data (direct)
  - indirectly from a decision tree (indirect)
- RIPPER is the most well-known direct rule learner

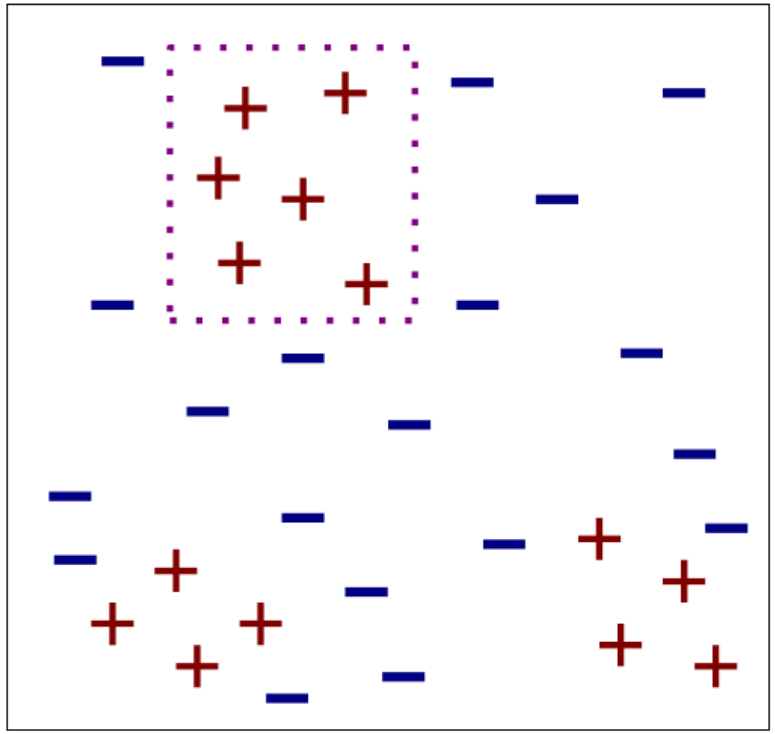
Sequential Covering

- Greedy approach which reduces the problem of learning a set of rules to a sequence of simpler problems, each requiring that a single rule is learned
- Once a rule r is learned, all covered examples (both positive and negative) are removed from the training set, so that the next generated rule is different from r
- It learns rules until it can no longer learn a rule whose performance is above the given Threshold

Sequential Covering

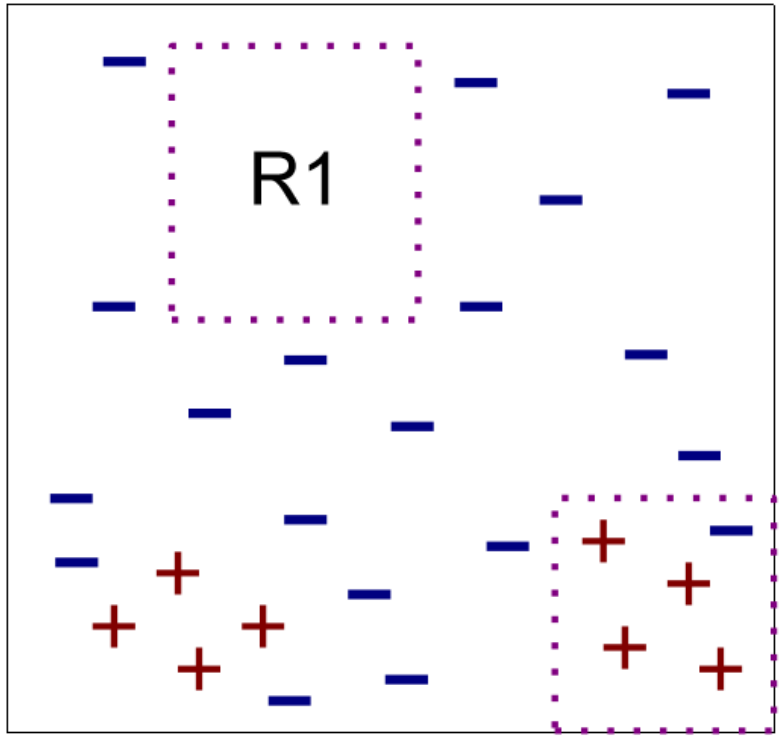


(i) Original Data

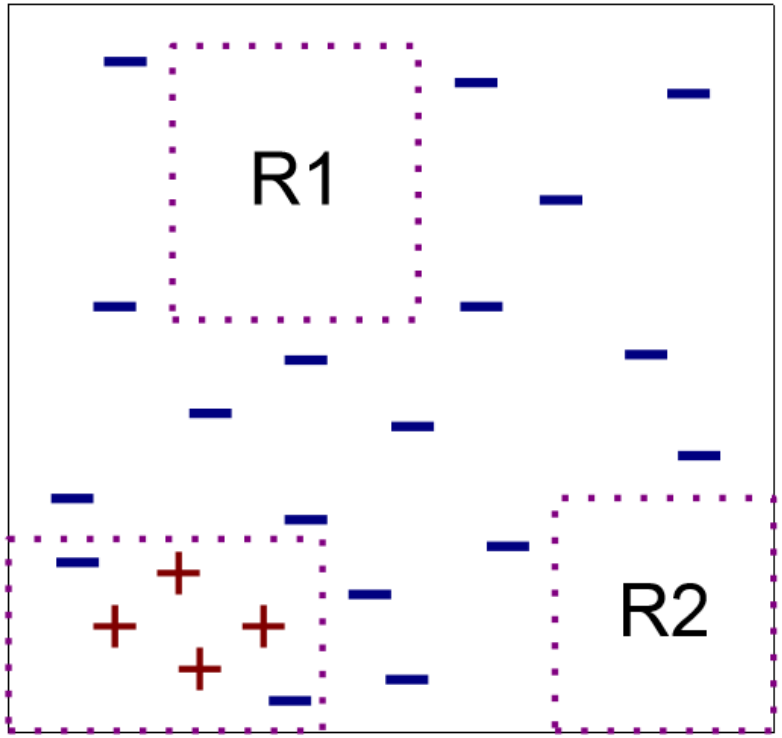


(ii) Step 1

Sequential covering



(iii) Step 2



(iv) Step 3

Sequential covering

- **Seq-Cov**(Attributes, Examples, Threshold)
  - **for** each class c in {c_1, …, c_k}
    - $Classifier_c$ = {};
    - $PosExam_c$ is the set of Examples with label = c;
    - $NegExam_c$ is the set of Examples with label <> c
    - $Exam_c$ = $PosExam_c$ ∪ $NegExam_c$;
    - Rule = Learn-One-Rule(Attributes, Examples)
    - **while** performance(Rule, Examples) > Threshold do
      - $Classifier_c$ = $Classifier_c$ ∪ { Rule}
      - $Exam_c$= $Exam_c$ – {examples covered by Rule}
      - Rule = Learn-One-Rule(Attributes, Examples)
    - **endWhile**
  - **endFor**
  - Sort classifiers according to their performance on Examples
  - Add default rule {} -> default class (with minimum priority)
- **return**

Sequential covering

- **Seq-Cov**(Attributes, Examples, Threshold)
  - **for** each class c in {c_1, …, c_k}
    - $Classifier_c$ = {};
    - $PosExam_c$ is the set of Examples with label = c;
    - $NegExam_c$ is the set of Examples with label <> c
    - $Exam_c$ = $PosExam_c$ ∪ $NegExam_c$;
    - Rule = Learn-One-Rule(Attributes, Examples)
    - **while** performance(Rule, Examples) > Threshold do
      - $Classifier_c$ = $Classifier_c$ ∪ { Rule}
      - $Exam_c$= $Exam_c$ – {examples covered by Rule}
      - Rule = Learn-One-Rule(Attributes, Examples)
    - **endWhile**
  - **endFor**
  - Sort classifiers according to their performance on Examples
  - Add default rule {} -> default class (with minimum priority)
- **return**

Learn-One-Rule

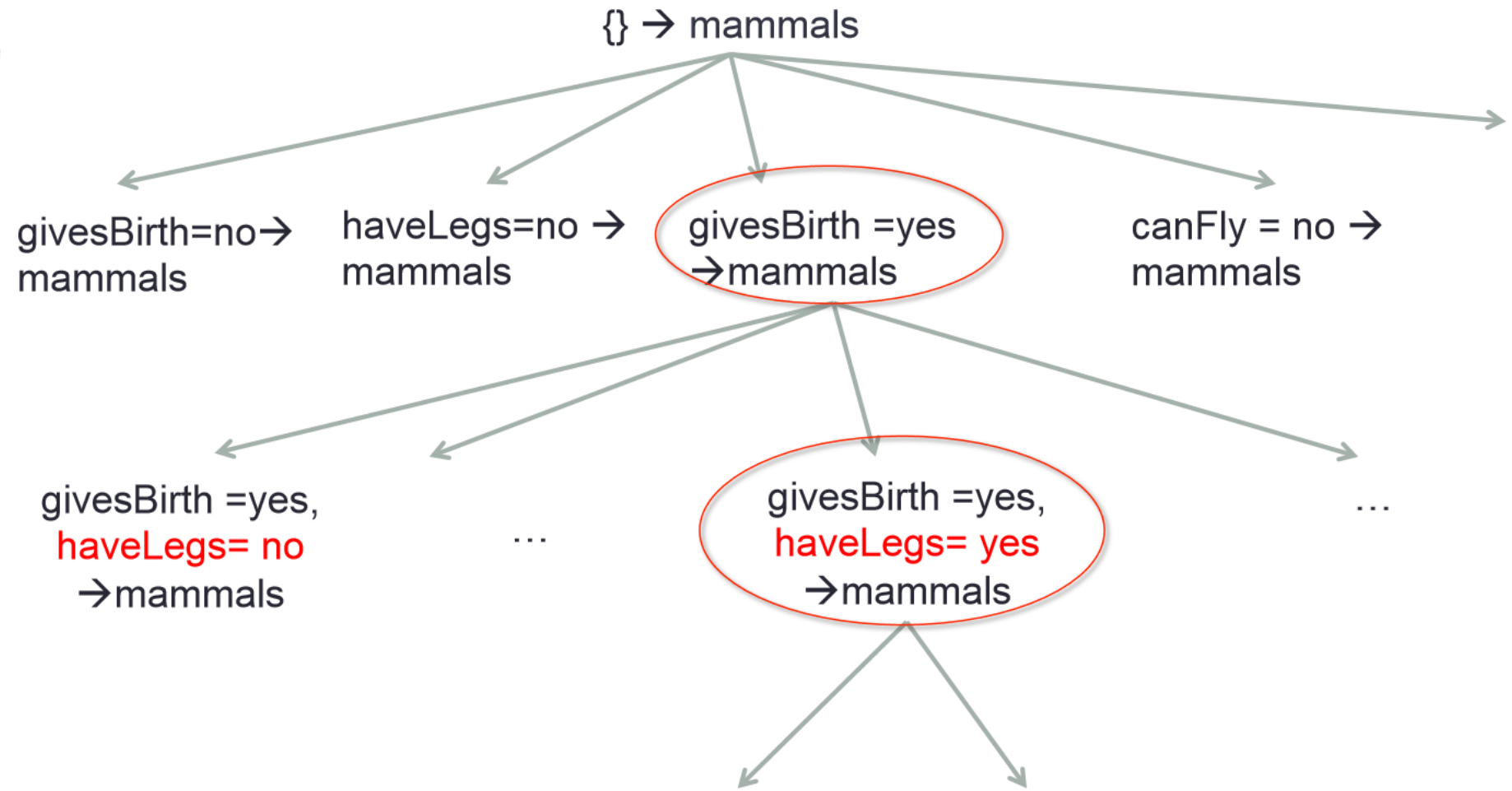
- **Objective**: learning a rule that covers many positive examples and few negative ones (possibly none)
- **Method**: Grows the rules in a greedy fashion based on a general-to-specific approach
  - It starts with the most general rule, i.e., one with the empty antecedent (it covers all the examples of the training set – poor performance)
  - Then greedily adds the attribute that most improves rule performance (e.g., accuracy) over the training set
  - The process is repeated by adding a second attribute, and so on and so forth
  - The process is repeated until the rule reaches an acceptable level of performance

Selecting an attribute

- Select the attribute that most increases rule performance (accuracy, or other measures, e.g., Laplace rank, FOIL's Information Gain) of the current rule
- Example
  - r0: a ➔ c
  - r1: a, b ➔ c
  - Select attribute b if
    - performance(r_1) > performance (r_0) and
    - performance (r_1) > performance (r_j)
  - for each rule r_j obtainable by adding in the antecedent of r_0 any attribute other than b

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

Learn-One-Rule



Rules Performance

- rule r: $A_1=a_1, \ldots, A_n=a_n \rightarrow c$
- r covers an example x if x= $<a_1, \ldots, a_n>$, i.e., the attributes of x satisfy the condition of the rule (antecedent)

Rules Performance

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

- Gives Birth = yes → Mammals

Rules Performance

- rule r: $A_1=a_1, \ldots, A_n=a_n \rightarrow c$
- An example x satisfies r if r covers x and the class of x is c

Rules Performance

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

- Gives Birth = yes → Mammals

Rules Performance

- Coverage(r) = cov(r)/D, where
  - D is the number of training examples
  - cov(r) is the number of examples covered by r
- Accuracy(r) = sat(r)/cov(r), where
  - Sat(r) is the number of examples satisfying r

Rules Performance

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

- **Gives Birth = yes → Mammals**
- **Coverage = 7/20**
- **Accuracy = 6/7**

Rules Performance

- Rule accuracy may not be a meaningful criterion
- r_1: covers 50 positive examples and 5 negative examples

$$Acc(r_1) = 50/55 = 90.9\%$$

- r_2: covers 2 positive examples and no negative examples

$$Acc(r_2) = 2/2 = 100\%$$

- However, r_1 is intuitively "more reliable" than r_2
- Other performance measures, e.g., Laplace rank, FOIL's Information Gain, etc.

Sequential covering

- **Seq-Cov**(Attributes, Examples, Threshold)
  - **for** each class c in {c_1, …, c_k}
    - $Classifier_c$ = {};
    - $PosExam_c$ is the set of Examples with label = c;
    - $NegExam_c$ is the set of Examples with label <> c
    - $Exam_c$ = $PosExam_c$ ∪ $NegExam_c$;
    - Rule = Learn-One-Rule(Attributes, Examples)
    - **while** performance(Rule, Examples) > Threshold do
      - $Classifier_c$ = $Classifier_c$ ∪ { Rule}
      - $Exam_c$= $Exam_c$ – {examples covered by Rule}
      - Rule = Learn-One-Rule(Attributes, Examples)
    - **endWhile**
  - **endFor**
  - Sort classifiers according to their performance on Examples
  - Add default rule {} -> default class (with minimum priority)
- **return**

Sequential covering

- **Seq-Cov**(Attributes, Examples, Threshold)
  - **for** each class c in {c_1, …, c_k}
    - $Classifier_c$ = {};
    - $PosExam_c$ is the set of Examples with label = c;
    - $NegExam_c$ is the set of Examples with label <> c
    - $Exam_c$ = $PosExam_c$ ∪ $NegExam_c$;
    - Rule = Learn-One-Rule(Attributes, Examples)
    - **while** performance(Rule, Examples) > Threshold do
      - $Classifier_c$ = $Classifier_c$ ∪ { Rule}
      - $Exam_c$= $Exam_c$ – {examples covered by Rule}
      - Rule = Learn-One-Rule(Attributes, Examples)
    - **endWhile**
  - **endFor**
  - Sort classifiers according to their performance on Examples
  - Add default rule {} -> default class (with minimum priority)
- **return**

Why do we need to eliminate instances?

Otherwise, the next rule is identical to previous rule

How a classifier works

- c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) ⟶ Bird
- c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) ⟶ Fish
- c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) ⟶ Mammal
- $c_{amphibian}$: (Lives in Water = sometimes) ⟶ Amphibian
- $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) ⟶ Reptile

Name	Gives birth	Can fly	Live in water	Blood type	class
hawk	no	yes	no	warm	?
Grizzly bear	yes	no	no	warm	?
turtle	no	no	sometimes	cold	?

How a classifier works

- c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) → Bird
- c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) → Fish
- c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) → Mammal
- $c_{amphibian}$: (Lives in Water = sometimes) → Amphibian
- $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) → Reptile

Name	Gives birth	Can fly	Live in water	Blood type	class
hawk	no	yes	no	warm	bird
Grizzly bear	yes	no	no	warm	?
turtle	no	no	sometimes	cold	?

How a classifier works

- c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) → Bird
- c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) → Fish
- c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) → Mammal
- $c_{amphibian}$: (Lives in Water = sometimes) → Amphibian
- $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) → Reptile

Name	Gives birth	Can fly	Live in water	Blood type	class
hawk	no	yes	no	warm	bird
Grizzly bear	yes	no	no	warm	mammal
turtle	no	no	sometimes	cold	?

How a classifier works

- c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) → Bird
- c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) → Fish
- c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) → Mammal
- $c_{amphibian}$: (Lives in Water = sometimes) → Amphibian
- $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) → Reptile

Ambiguity!!

Name	Gives birth	Can fly	Lives in water	Blood type	class
hawk	no	yes	no	warm	bird
Grizzly bear	yes	no	no	warm	mammal
turtle	no	no	sometimes	cold	?????

Rules are not mutually exclusive Ordered Rule Sets

- To solve the ambiguity, we order classifiers according to their reliability
- The less mistakes over the training data a classifier makes, the more reliable it is

Rules are not mutually exclusive Ordered Rule Sets

- Assume $c_{bird} > c_{fish} > c_{mammal} > c_{amphibian} > c_{rept}$
- Classifiers are ordered in decreasing order of reliability
- When a new instance is presented, it is classified by the highest-ranked classifier triggered by the instance

Rules are not mutually exclusive Ordered Rule Sets

- Classifiers are ordered
  - c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) → Bird
  - c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) → Fish
  - c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) → Mammal
  - $c_{amphibian}$: (Lives in Water = sometimes) → Amphibian
  - $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) → Reptile

Name	Gives birth	Can fly	Live in water	Blood type	class
hawk	no	yes	no	warm	bird
Grizzly bear	yes	no	no	warm	mammal
turtle	no	no	sometimes	cold	Amphib

Turtle is NOT a reptile according to the above model, as $c_{amphibian}$ is more reliable than $c_{reptile}$

Sequential covering

- **Seq-Cov**(Attributes, Examples, Threshold)
  - **for** each class c in {c_1, …, c_k}
    - $Classifier_c$ = {};
    - $PosExam_c$ is the set of Examples with label = c;
    - $NegExam_c$ is the set of Examples with label <> c
    - $Exam_c$ = $PosExam_c$ ∪ $NegExam_c$;
    - Rule = Learn-One-Rule(Attributes, Examples)
    - **while** performance(Rule, Examples) > Threshold do
      - $Classifier_c$ = $Classifier_c$ ∪ { Rule}
      - $Exam_c$= $Exam_c$ – {examples covered by Rule}
      - Rule = Learn-One-Rule(Attributes, Examples)
    - **endWhile**
  - **endFor**
  - Sort classifiers according to their performance on Examples
  - Add default rule {} -> default class (with minimum priority)
- **return**

Rules are not exhaustive Default rule

- Rules are not exhaustive: an instance may not trigger any rule
  - c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) → Bird
  - c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) → Fish
  - c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) → Mammal
  - $c_{amphibian}$: (Lives in Water = sometimes) → Amphibian
  - $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) → Reptile

Name	Gives birth	Can fly	Lives in water	Blood type	class
dogfish shark	yes	no	yes	cold	?

- Dogfish shark does not trigger any rule

Rules are not exhaustive Default rule

- The instance is assigned to a default class, i.e., the class assigned by the default rule
  - {} is default class
  - this is triggered when all other rules have failed
  - Default class: majority class of training examples not covered by any rule

Rules are not exhaustive Default rule

- c_{bird}: (Gives Birth = no) ∧ (Can Fly = yes) → Bird
- c_{fish}: (Gives Birth = no) ∧ (Lives in Water = yes) → Fish
- c_{mammal}: (Gives Birth = yes) ∧ (Blood Type = warm) → Mammal
- $c_{amphibian}$: (Lives in Water = sometimes) → Amphibian
- $c_{reptile}$: (Gives Birth = no) ∧ (Can Fly = no) → Reptile
- Default rule: {} → Mammal

Name	Gives birth	Can fly	Lives in water	Blood type	class
dogfish shark	yes	no	yes	cold	mammal

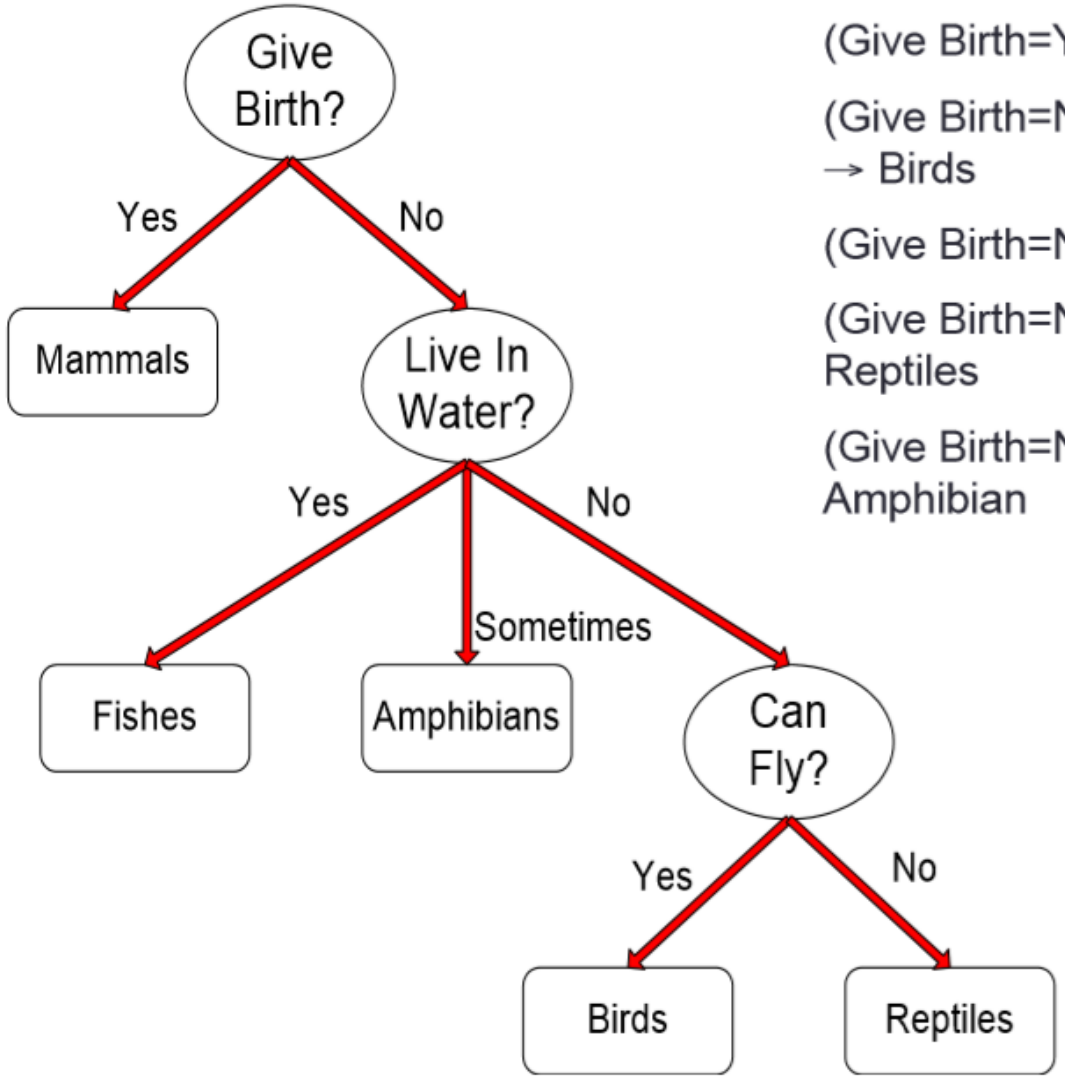
Sequential covering – rule-based ordering

- **Seq-Cov**(Attributes, Examples, Threshold)
  - **for** each class c in {c_1, …, c_k}
    - $Classifier_c$ = {};
    - $PosExam_c$ is the set of Examples with label = c;
    - $NegExam_c$ is the set of Examples with label <> c
    - $Exam_c$ = $PosExam_c$ ∪ $NegExam_c$;
    - Rule = Learn-One-Rule(Attributes, Examples)
    - **while** performance(Rule, Examples) > Threshold do
      - $Classifier_c$ = $Classifier_c$ ∪ { Rule}
      - $Exam_c$= $Exam_c$ – {examples covered by Rule}
      - Rule = Learn-One-Rule(Attributes, Examples)
    - **endWhile**
  - **endFor**
  - Sort classifiers according to their performance on Examples
  - Add default rule {} -> default class (with minimum priority)
- **return**

How a rule-based classifier works Summary

- When a new instance 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 is fired, it is assigned to the default class

C4.5 rules versus RIPPER



(Give Birth=Yes) → Mammals

(Give Birth=No, Lives In Water = no, Can Fly=Yes) → Birds

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

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

(Give Birth=No, Live in Water=sometimes) → Amphibian

C4.5 rules versus RIPPER

- C4.5:
  - (Give Birth=Yes) → Mammals
  - (Give Birth=No, Lives In Water = no, Can Fly=Yes) → Birds
  - (Give Birth=No, Live in Water=Yes) → Fishes
  - (Give Birth=No, Live in Water=No, Can Fly=No) → Reptiles
  - (Give Birth=No, Live in Water=sometimes) → Amphibian
- RIPPER:
  - (Live in Water=Yes) → Fishes
  - (Can Fly=No) → Reptiles
  - (Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles
  - (Can Fly=Yes, Give Birth=No) → Birds
  - () → Mammals

C4.5 rules versus RIPPER

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
- Can classify new instances rapidly
- Performance comparable to decision trees