
Data Mining:

3. Klassifikation

C) Alternative Techniques I:
Rule-based Classifiers, NN-based Classifiers

Rule-Based Classifier

- Classify records by using a collection of “if...then...” rules
- Rule: $(Condition) \rightarrow y$
 - where
 - ◆ *Condition* is a conjunction of (attribute,value)-comparisons
 - ◆ y is the class label
 - *Left hand side*: rule antecedent or (pre-)condition
 - *Right hand side*: rule consequent
 - Examples of classification rules:
 - ◆ $(\text{Blood Type}=\text{Warm}) \wedge (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}$
 - ◆ $(\text{Taxable Income} < 50\text{K}) \wedge (\text{Refund}=\text{Yes}) \rightarrow \text{Cheat}=\text{No}$

Rule-based Classifier (Example)

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.
- Then r is said to be fired or **triggered** by x

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 in class Birds

The rule R3 covers the grizzly bear \Rightarrow in class Mammals

Rule Quality: Coverage and Accuracy

- **Coverage** of a rule:
 - Fraction of records that satisfy the antecedent of a rule
- **Accuracy** of a rule:
 - Fraction of the former records that satisfy both the antecedent and consequent of a rule (=confidence)

<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) \rightarrow No *has*

Coverage = 40%, Accuracy = 50%

How does a 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

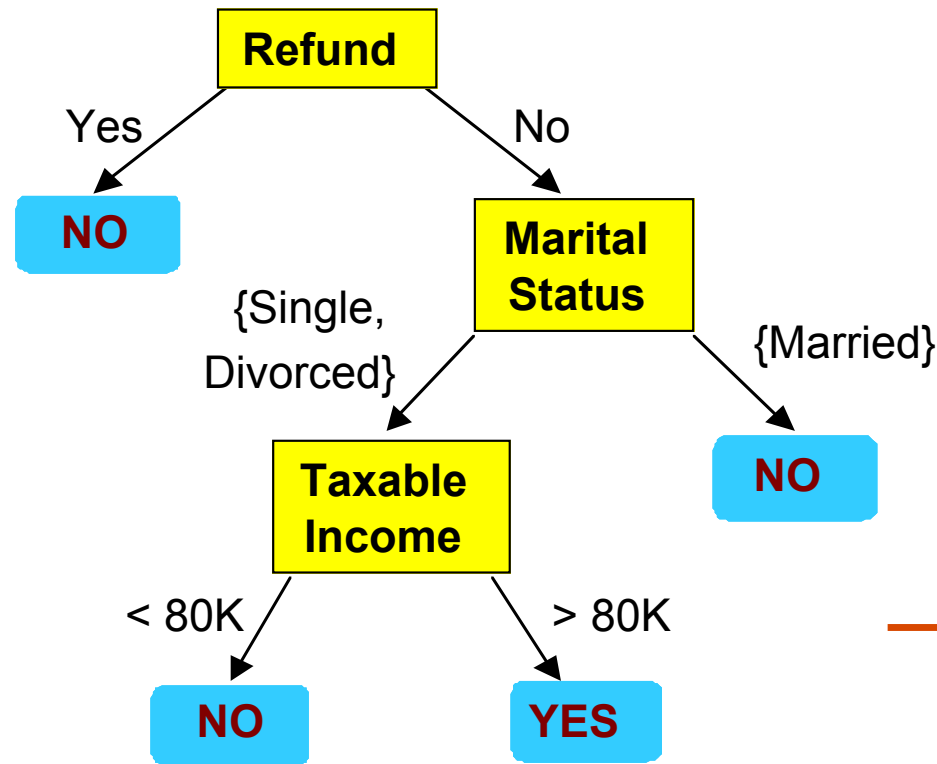
A turtle triggers both R4 and R5; *contradictory classes!*

A dogfish shark triggers none of the rules; *non-covered instance!*

Characteristics of Rule-Based Classifier

- **Mutually exclusive** rules
 - Classifier contains mutually exclusive rules if the rules are independent of each other
 - 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

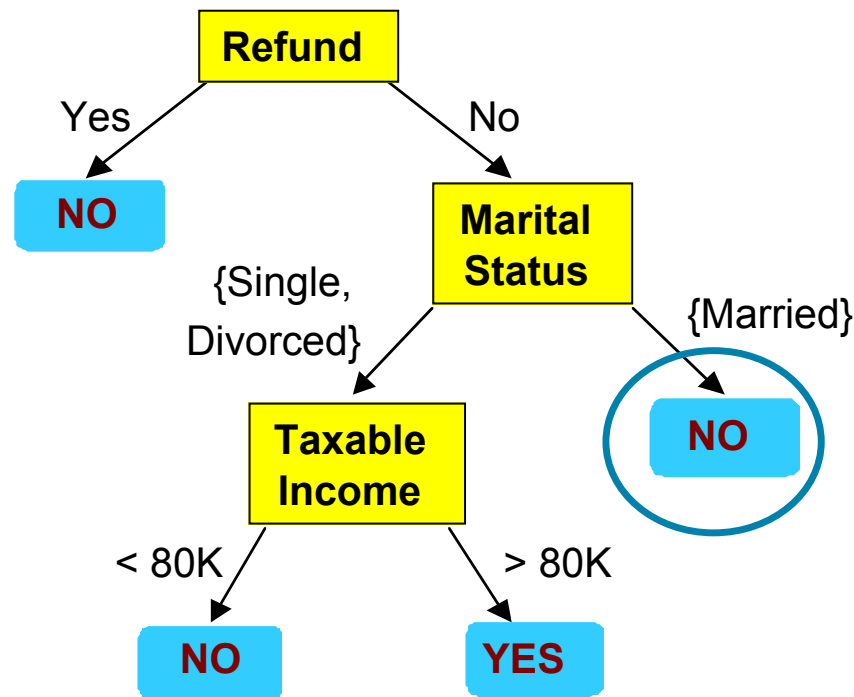
(Refund=No, Marital Status={Single, Divorced}, Taxable Income>80K) ==> Yes

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

Rules are mutually exclusive and exhaustive

Rule set contains as much information as the tree (4 branches, thus 4 rules)

Rules Can Be Simplified



Tid	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 – count rule “votes”, take maximum
- 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 gets the class label of the highest ranked rule it has triggered
 - If none of the rules fired, it is assigned to the default class

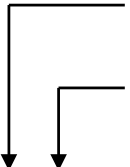
R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

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	?

Reptiles !

Ordered Rule Set: 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
 - Preferred in well-known rule-based classifiers

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:

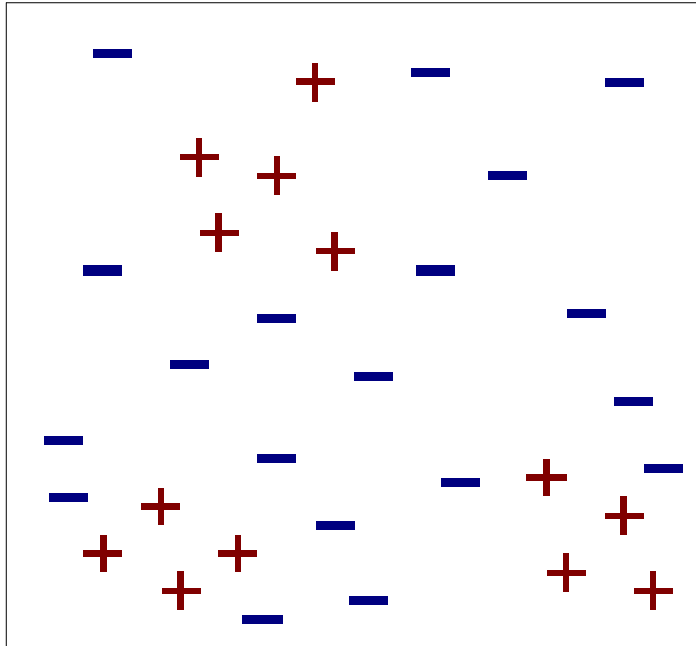
- ◆ Extract rules from other classification models
(e.g. decision trees, neural networks, etc).
- ◆ e.g.: C4.5rules

Direct Method: Sequential Covering

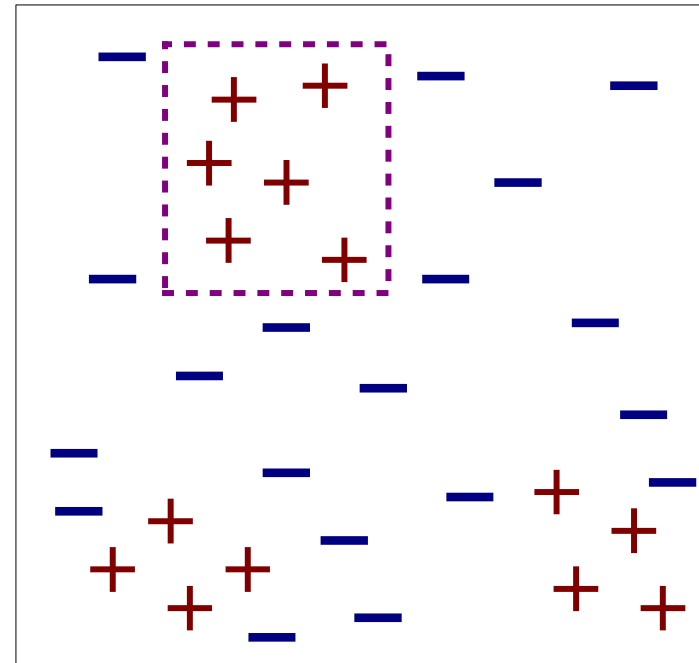
Proceed class by class (y):

- Start from an empty rule set
- “Grow a rule” until rule stopping criterion is met in order to (greedily) extract the best rule for class y that covers the current set of training records
- Prune the rule if necessary
- Remove training records covered by the rule
- Add the new rule to the rule set
- Repeat steps (Grow...) until no more training records for y are left or until heuristic class stopping criterion is met

Sequential Covering: Example



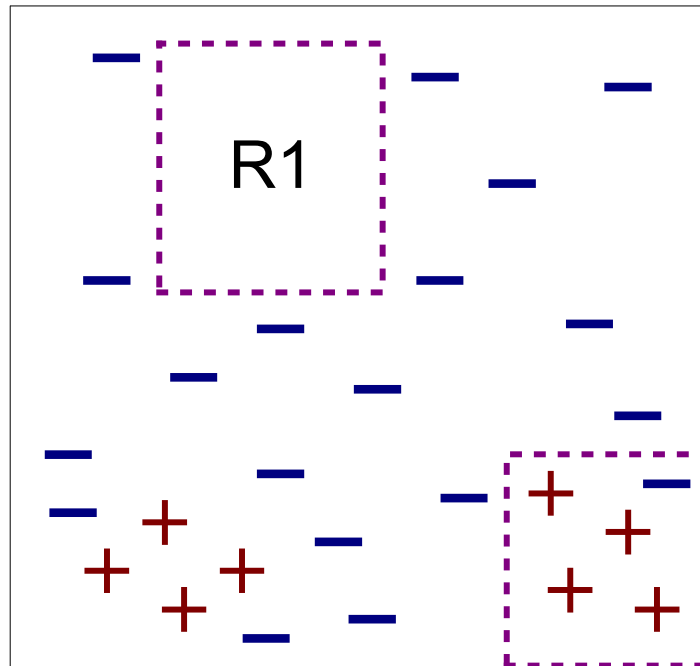
(i) Original Data



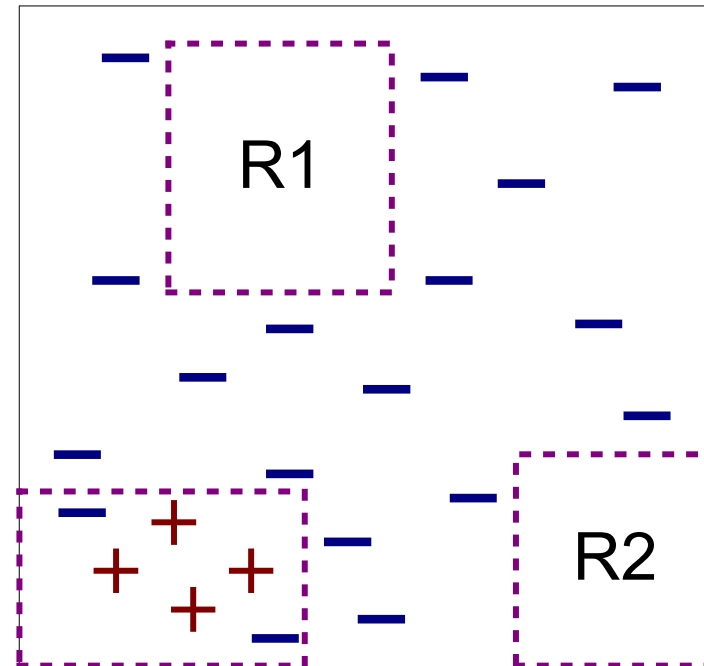
(ii) Step 1

A rule is desirable if it covers most of the positive examples (instances of current class) and none of (or very few) negative examples (other classes).

Sequential Covering: Example...

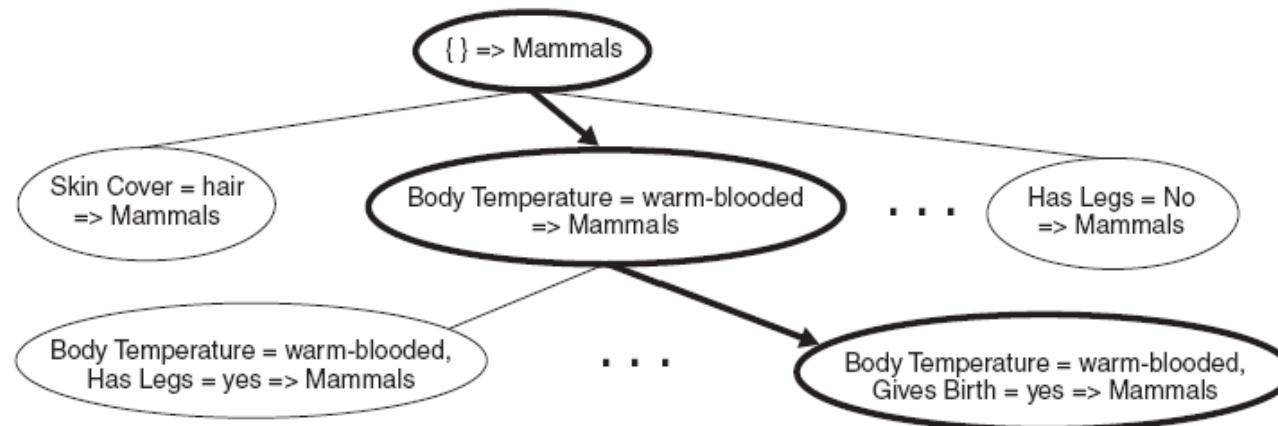


(iii) Step 2

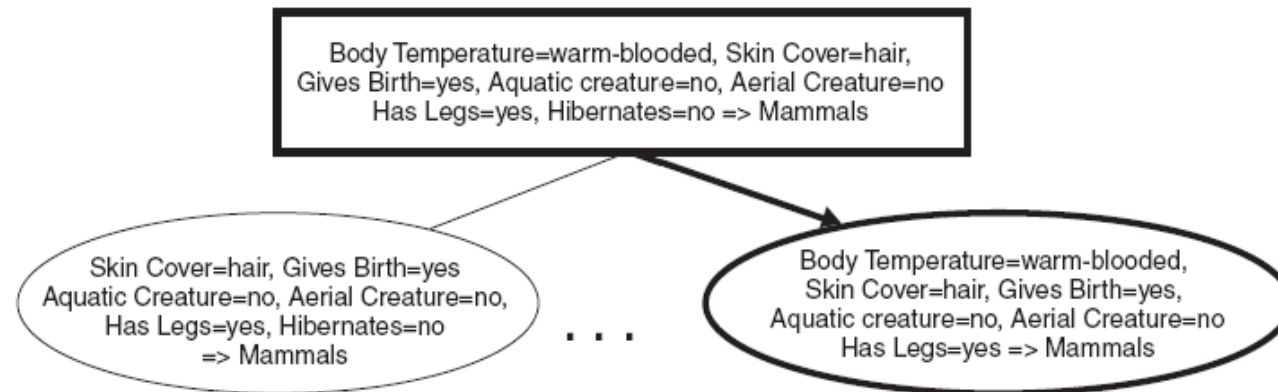


(iv) Step 3

Sequential Covering: Rule Growing



(a) General-to-specific



(b) Specific-to-general

Figure 5.3. General-to-specific and specific-to-general rule-growing strategies.

- Start with
 - a) empty antec.
 - b) a positive example as seed
- Modify rule until rule stopping criterion is met, e.g.
 - a) added conjunct does not improve rule quality
 - b) rule starts covering negative examples

Sequential Covering: Rule Growing/Evaluation

- Rule Growing and Rule Evaluation in the **RIPPER** Algorithm:

- Start from an empty rule: $\{\} \Rightarrow \text{class}$
- Add conjunct that maximizes “FOIL’s information gain” measure:
 - ◆ $R_0: \{\} \Rightarrow \text{class}$ (initial rule)
 - ◆ $R_1: \{A\} \Rightarrow \text{class}$ (rule after adding conjunct)
 - ◆ $\text{Gain}(R_0, R_1) = p_1 [\log(p_1/(p_1+n_1)) - \log(p_0/(p_0 + n_0))]$
 - ◆ where p_0 : number of positive instances covered by R_0
 n_0 : number of negative instances covered by R_0
 p_1 : number of positive instances covered by R_1
 n_1 : number of negative instances covered by R_1

FOIL = first order inductive learner, ripper = Trennmaschine

Sequential Covering: Rule Evaluation

- Other Measures:

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

- 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 (positive) class instances covered by rule

k : Number of classes

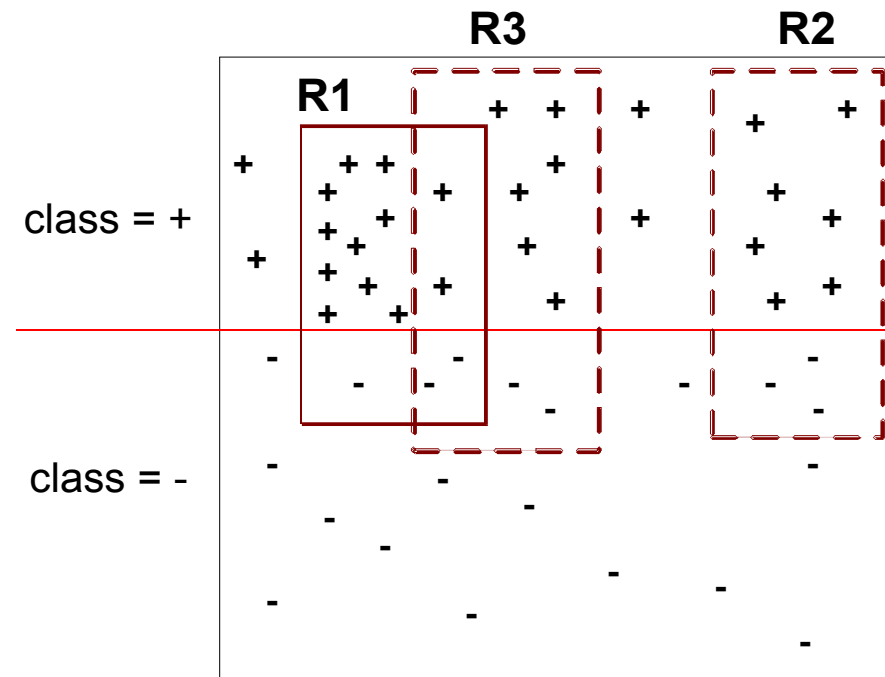
p : Prior probability of (positive) class

Sequential Covering: Rule Evaluation

- For example, assume a training set with 60 positive vs. 100 negative examples.
- Assume we have two rules R1/R2 covering 50 vs. 5 / 8 vs. 0 positive vs. negative examples
 - Accuracy(R1) = $50/55 = 90.9\%$
 - Laplace(R1) = $51/57 = 89.4\%$
 - M-estimate(R1) = $(50 + 2 \cdot 60/160)/57 = 92.2\%$
 - Accuracy(R2) = $8/8 = 100\%$
 - Laplace(R2) = $9/10 = 90\%$
 - M-estimate(R2) = $(8 + 2 \cdot 60/160)/10 = 87.50\%$
 - => prefer R1 !
- Higher accuracy might be misleading due to low coverage
- FOIL's information gain emphasizes support count:
 - Gain($\{\} \Rightarrow \text{pos}$, R1) = 63.9, Gain($\{\} \Rightarrow \text{pos}$, R2) = 11.3

Sequential Covering: Instance Elimination

- Why do we need to remove instances?
 - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
 - Prevent overestimating accuracy of rule
- Why do we remove negative instances?
 - Prevent underestimating accuracy of rule
- Compare rules R2 and R3 in the diagram with/without removals for rule R1



Sequential Covering: Rule Pruning/Stopping

- Rule Pruning (Simplification)
 - Similar to post-pruning of decision trees
 - Validation-based 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
- Typical heuristic stopping criteria
 - Compute a kind of “gain” for a rule / class
 - Stop growing of a rule/generation of new rules, if “gain” is not significant or negative

... in the RIPPER Algorithm

- 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
 - End with default class

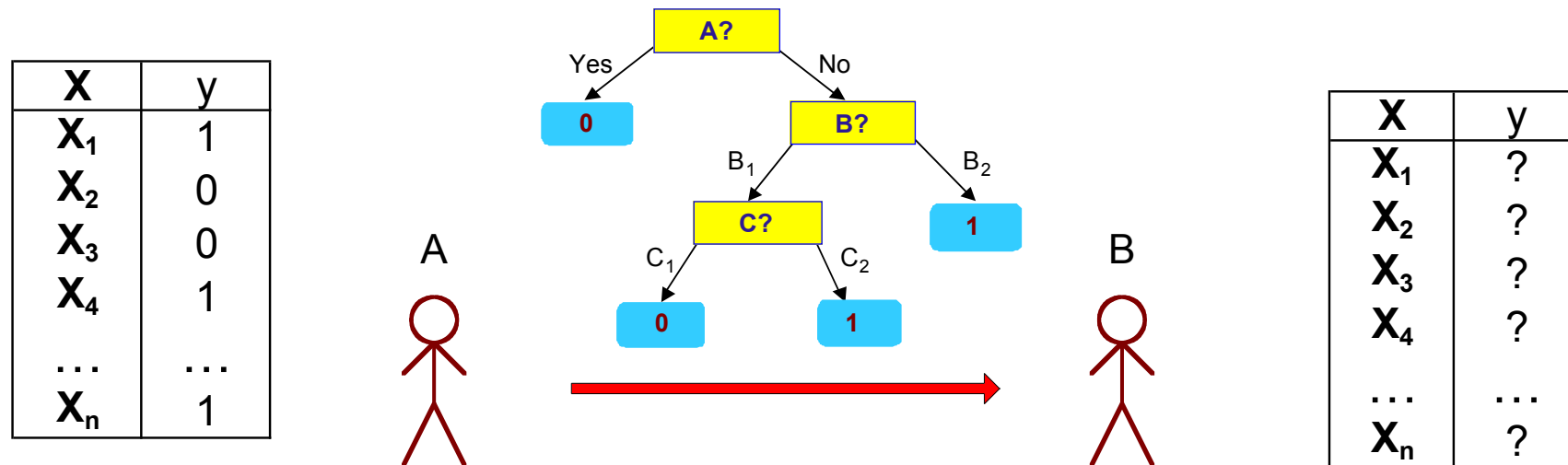
... in the RIPPER Algorithm

- Growing/Pruning 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 validation-based 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 conjuncts (in the condition) whose deletion improves v

... in the RIPPER Algorithm

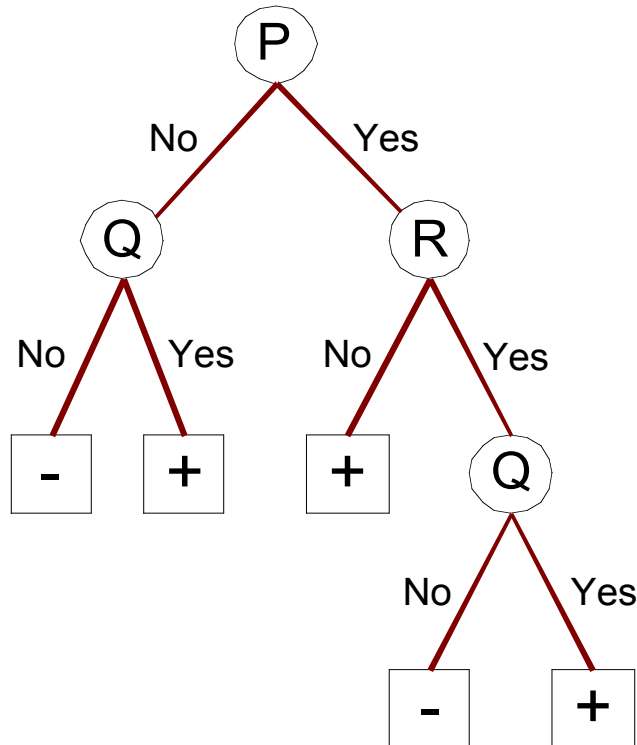
- Building a Rule Set:
 - Use sequential covering algorithm
 - ◆ Find the “best” rule that covers the current set of positive examples
 - ◆ Eliminate both positive and negative examples covered by the rule
 - Stopping is based on the MDL principle:
Each time a rule is added to the rule set, compute the new “description length” (of rule set + exceptions)
 - ◆ stop adding new rules when the new description length is d bits longer than the smallest description length obtained so far
 - Another stopping condition: error rate of rule on validation set must not exceed 50%
- Plus additional optimizations...

[Background] Minimum Description Length (MDL) Principle



- A knows the classification, B not. How much costs (transmission of) a description of the classification from A to B ?
- **Explicit: proportional to $n = \text{Cost}(\text{Data})$**
- **Or model-based: $\text{Cost}(\text{Model}) + \text{Cost}(\text{Data}|\text{Model})$ *Minimize this!***
- Cost is the number of bits needed for encoding.
- $\text{Cost}(\text{Model})$ is the cost of encoding the model, e.g. a decision tree: here nodes (number of children) plus splitting conditions.
- $\text{Cost}(\text{Data}|\text{Model})$ is the cost of encoding remaining explicit data, here misclassification errors.

Indirect Methods (Rule Extraction)



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) ==> +

The rules can be simplified to:

r1: *as above*
r2': (Q=Yes) ==> +
r3: *as above*
r4': () ==> -
r5: *omit*

Indirect Method: C4.5rules: Rule Generation

- Extract rules from an unpruned 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
 - Compare the (e.g. pessimistic) generalization error rate for r against all r 's
 - If there is an r' with lower error rate then prune r to r'
 - Repeat until error rate cannot be improved

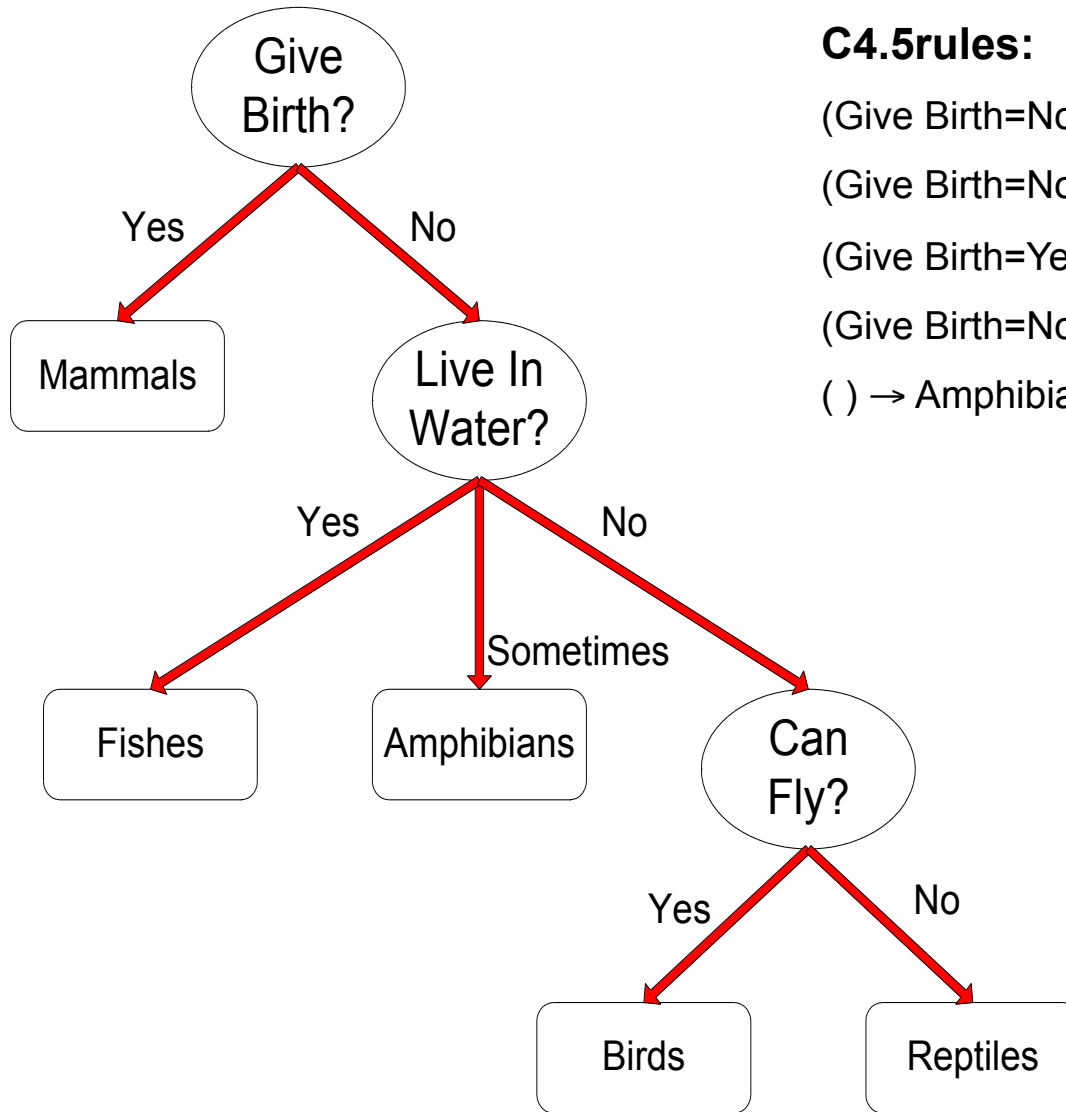
Indirect Method: C4.5rules: Rule Ordering

- Instead of ordering individual 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
$$= g \cdot \text{Cost}(\text{model}) + \text{Cost}(\text{errors}|\text{model})$$
 - ◆ g is a parameter that takes into account the presence of redundant attributes in a model=rule set (default value = 0.5)
 - Arrange classes in increasing order of their description length

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

C4.5 versus 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

() → 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

C4.5 versus 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
 - Easier to interpret than decision trees
 - Easy to generate
 - Can classify new instances rapidly
 - Performance comparable to decision trees
-
- ...and now from *eager* to *lazy* learning ...

Instance-Based Classifiers

Set of Stored Cases

Atr1	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

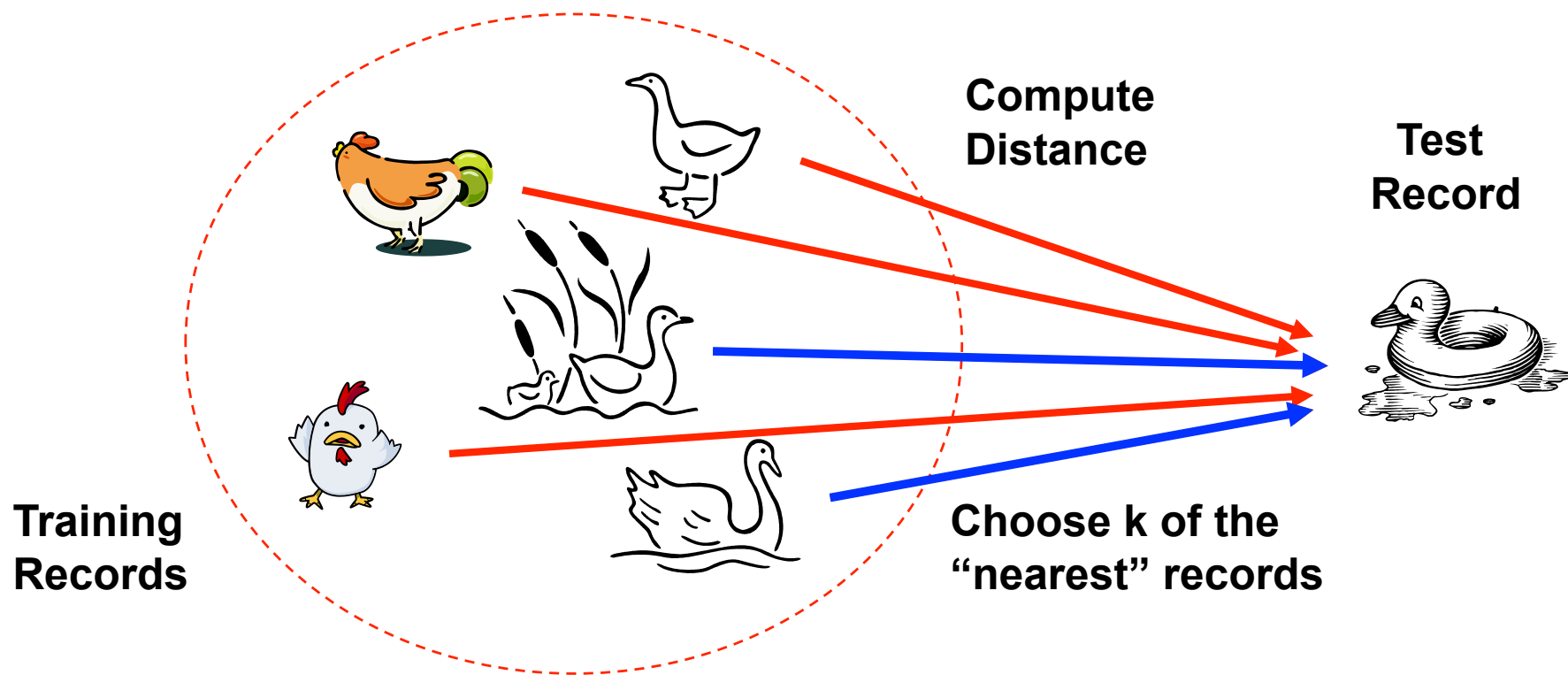
Atr1	AtrN

Instance Based Classifiers

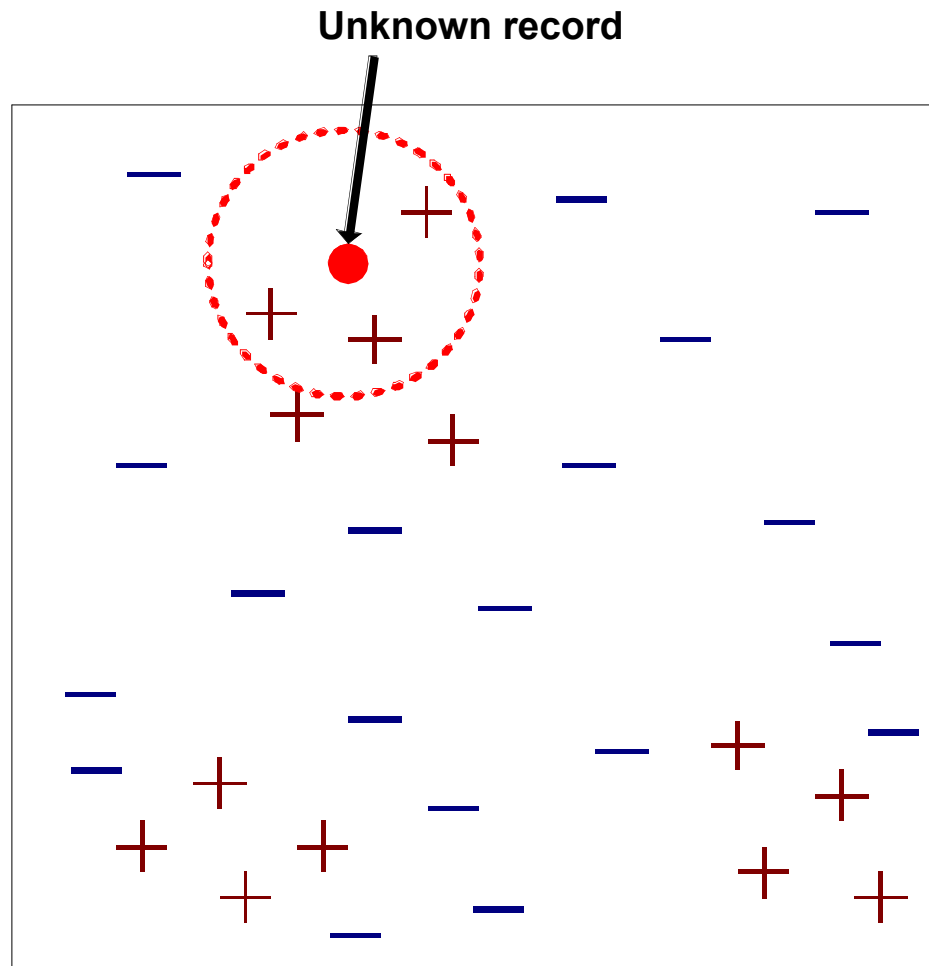
- Examples:
 - Rote learner
 - ◆ Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
 - ◆ Drawback: records may be not classified at all
 - Nearest neighbor
 - ◆ Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

- Basic idea:
 - If it walks like a duck, quacks like a duck, then it's probably a duck

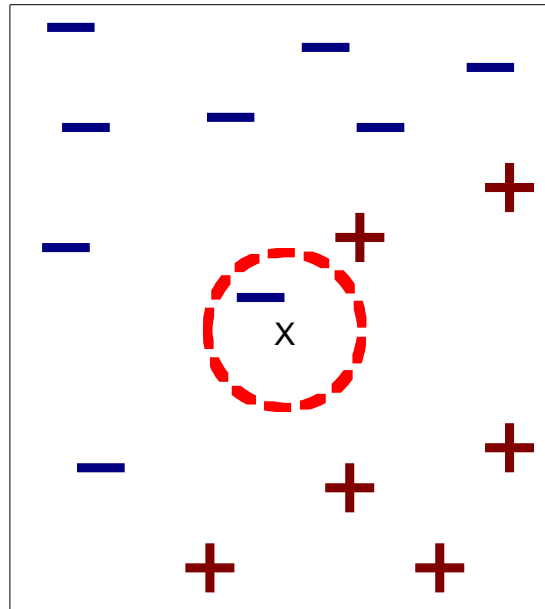


Nearest-Neighbor Classifiers

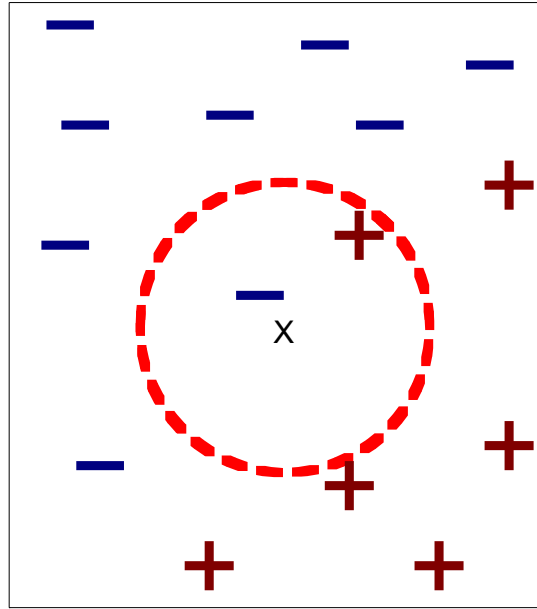


- Requires three things
 - The set of stored records
 - Distance Metric to compute distance between records
 - The value of k , the number of nearest neighbors to retrieve
- To classify an unknown record:
 - Compute distance to other training records
 - Identify k nearest neighbors
 - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

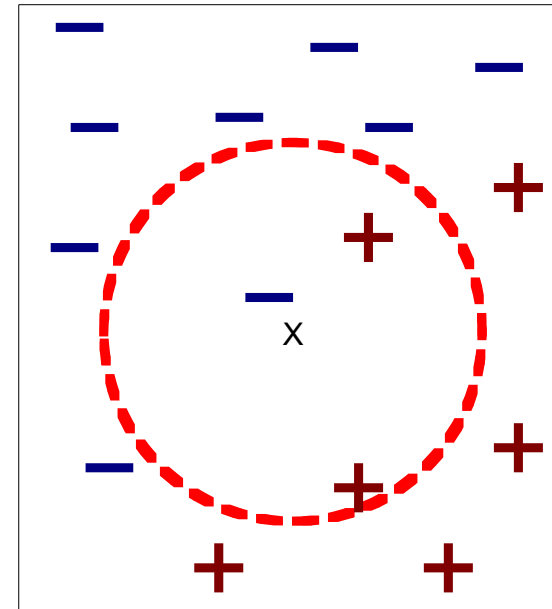
Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

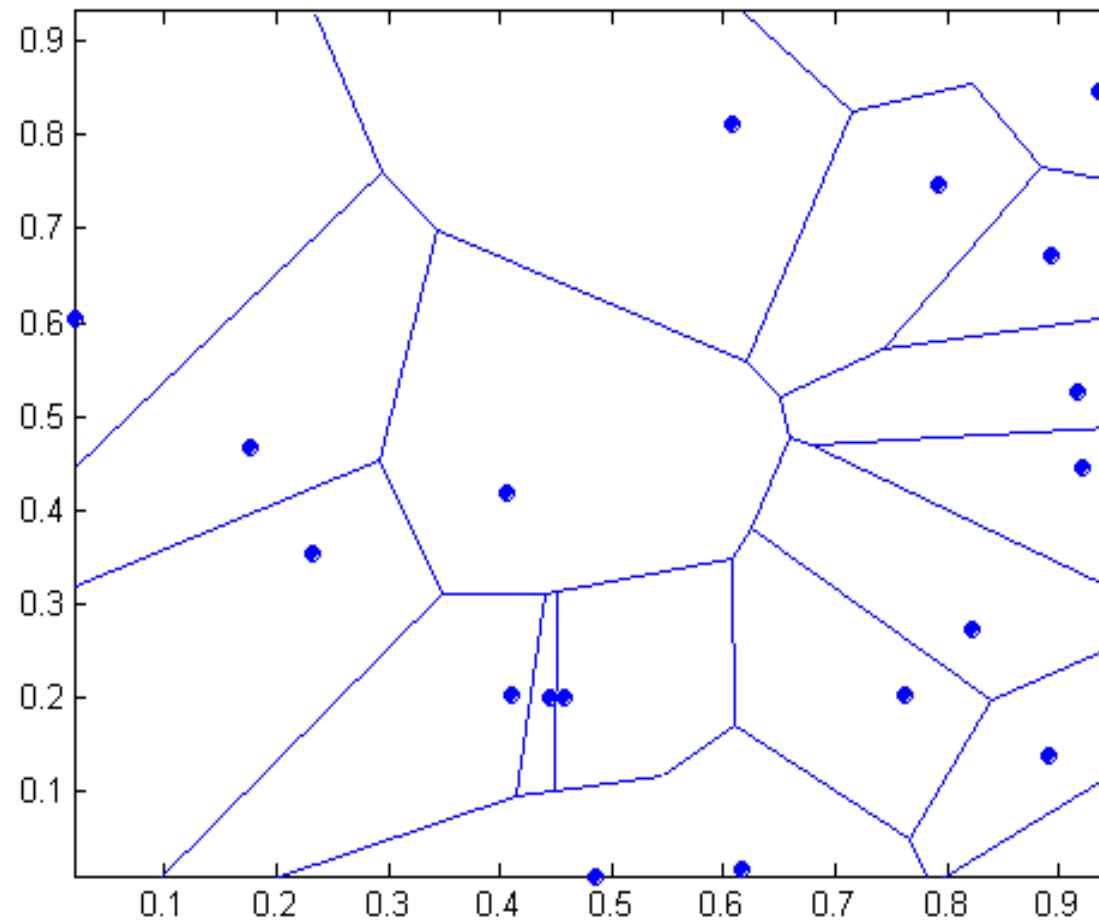


(c) 3-nearest neighbor

k-nearest neighbors of a record x are data points that have the k smallest distances to x

1 nearest-neighbors

can be visualized by a Voronoi Diagram:



Nearest Neighbor Classification

- Compute distance between two points:
 - e.g., Euclidean distance

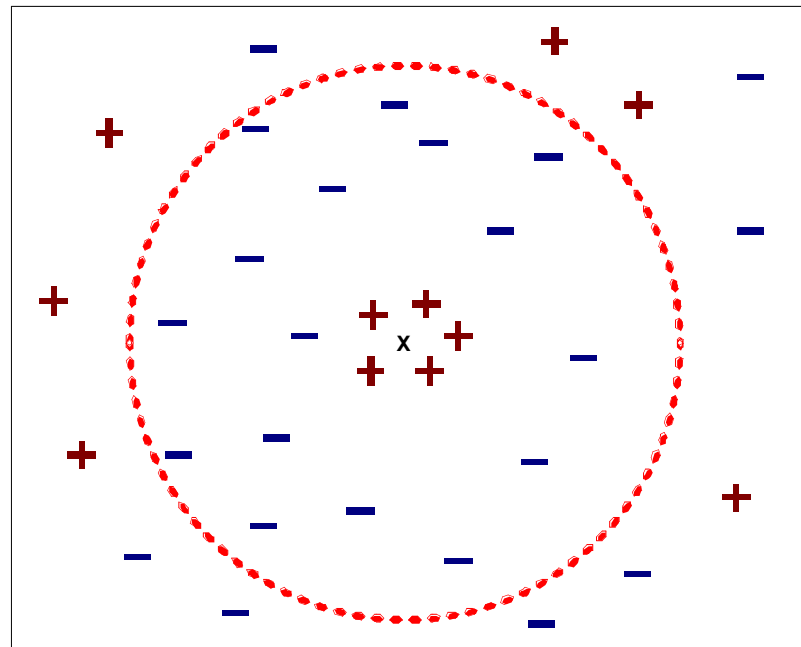
$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

(which, however, may be counterintuitive for high dimensions)

- Determine the class from nearest neighbor list
 - take the majority vote of class labels among the k-nearest neighbors
 - optional: weigh the votes according to distance
 - ◆ typical weight factor: $w = 1/d^2$

Nearest Neighbor Classification...

- Choosing the value of k :
 - If k is too small, sensitive to overfitting by noise points
 - If k is too large, neighborhood may include far away points from other classes



Nearest Neighbor Classification...

- Scaling issues
 - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
 - Example:
 - ◆ height of a person may vary from 1.5m to 1.8m
 - ◆ weight of a person may vary from 90lb to 300lb
 - ◆ income of a person may vary from \$10K to \$1M

Characteristics of NN Classifiers

- k-NN classifiers are lazy learners
 - They do not build models explicitly unlike eager learners such as decision tree induction and rule-based systems
 - Classifying unknown records is relatively expensive (compute distances)
 - Preprocessing needed
 - Decisions are made locally
 - + Decision boundaries can be arbitrarily shaped