Unit 3: Classification

Section 2:
Alternative methods and Advanced Classification

Rule-Based Classifier

- ✗Classify records by using a collection of "if…then…" rules
- ✗Rule: (*Condition*) → *y*
  - ✗where
    - ✗ *Condition* is a conjunction of attributes
    - ✗ *y* is the class label
  - ✗*LHS*: rule antecedent or condition
  - ✗*RHS*: rule consequent
  - ✗Examples of classification rules:
    - ✗ (Blood Type=Warm) ^ (Lay Eggs=Yes) → Birds
    - ✗ (Taxable Income < 50K) ^ (Refund=Yes) → Evade=No

Application of Rule-Based Classifier

- ✗A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → 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 => Bird
The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

- ✗COVERAGE OF A RULE **R:** A → Y IN A DATASET D
  - ✗Fraction of records that satisfy the antecedent of a rule

$$Coverage(D) = \frac{|A|}{|D|}$$

- ✗ACCURACY OF A RULE:
  - ✗Fraction of records that satisfy both the antecedent and consequent of a rule

$$Accuracy(D) = \frac{|A \cap y|}{|A|}$$

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

Tid	**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**

How does Rule-based Classifier Work?

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → 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 criterion must be stablished
A dogfish shark triggers none of the rules: Classification is unknown

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

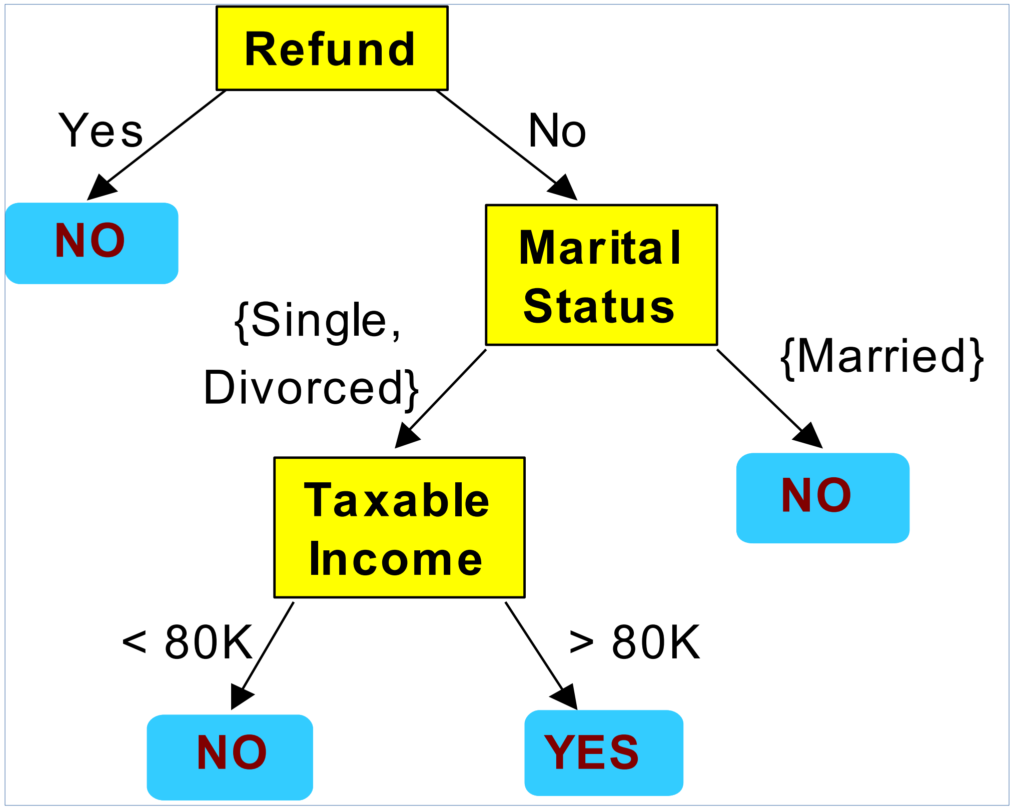
Classification Rules

(Refund=Yes) ==> No

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

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

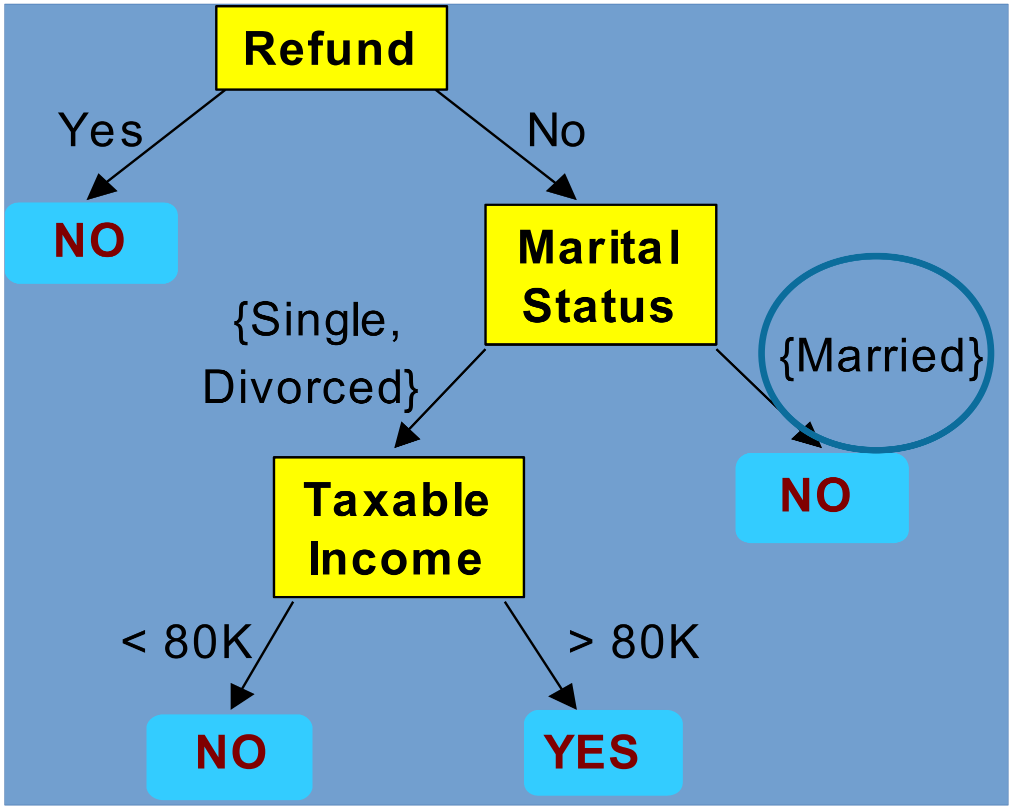
(Refund=No, Marital Status={Married}) ==> No



RULES ARE MUTUALLY EXCLUSIVE AND EXHAUSTIVE

RULE SET CONTAINS AS MUCH INFORMATION AS THE TREE

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: (Refund=No) $\wedge$ (Status=Married) → No

Simplified Rule: (Status=Married) → 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) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → 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* (a criterion needed)
  - ✗Lower-ranked rules more difficult to interpret: They imply the negation of the previous ones
- ✗Class-based ordering
  - ✗Rules that belong to the same class appear together
  - ✗Rules are collectively sorted by group of classes (internal order irrelevant)

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.  Let $E$ be the training records and $A$ de the set of attribute-value
    pairs $\{(A_j, v_j)\}$
2.  Let $Y_o$ be an ordered set of classes $\{y_1, y_2, \ldots, y_k\}$
3.  Let $R = \{\}$ be the initial rule list
4.  for each class $y$ in $Y_o - \{y_k\}$ do
5.      while stopping condition is not met do
6.          $r \leftarrow$ Learn-One-Rule($E, A, y$)
7.          Remove training records from $E$ that are covered by $r$
8.          Add $r$ to the bottom of the rule list: $R \rightarrow R \vee r$
9.      end while
10. end for
11. Insert the default rule, $\{\} \rightarrow y_k$, to the bottom of the rule list $R$
```

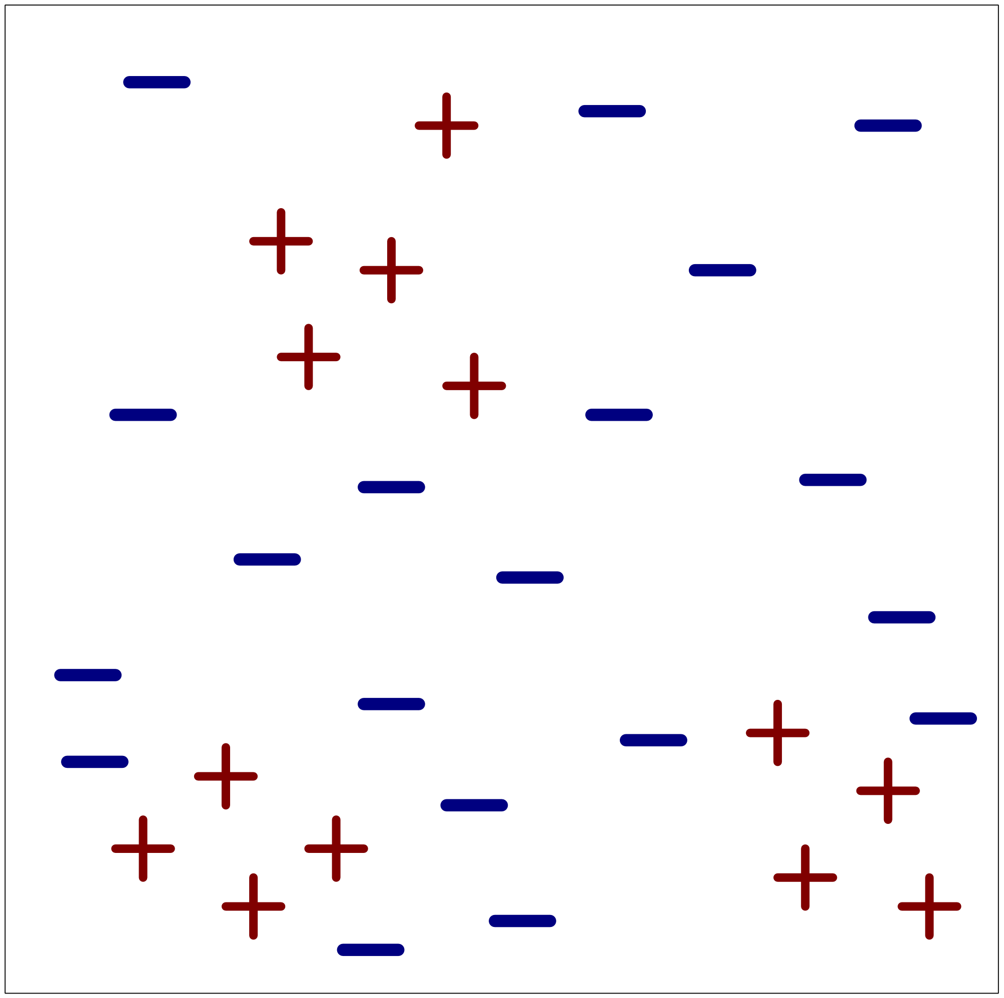
Learn-one-rule algorithm

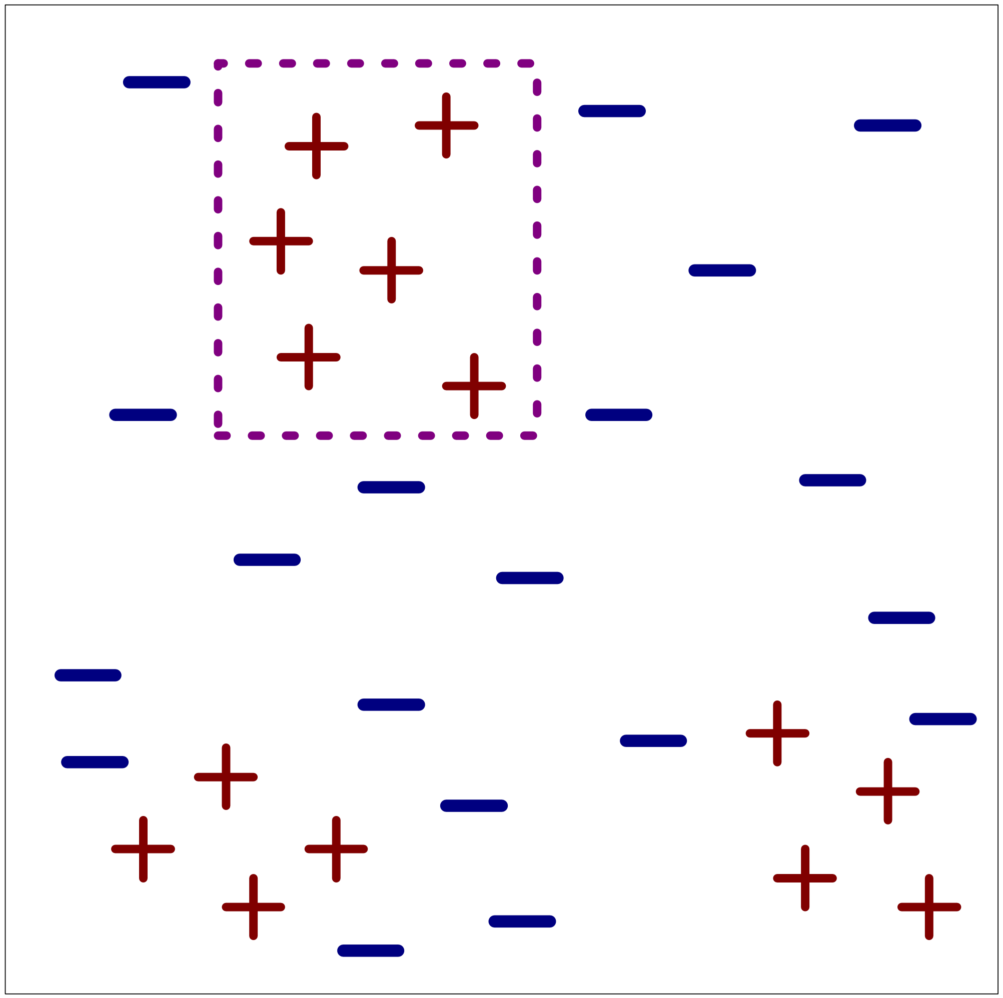
```
Learn-One-Rule(target_attribute, attributes, examples, k)
# Returns a single rule that covers some of the Examples
  best-hypothesis = the most general hypothesis
  candidate-hypotheses = {best-hypothesis}
  while candidate-hypothesis     ;Generate the next more specific candidate-hypotheses
    all-constraints = all "att.=val." contraints
    new-candidate-hypotheses = all specializations of candidate-hypotheses by adding all-constraints
    remove from new-candidate-hypotheses any that are duplicates, inconsistent, or not maximally specific
    # Update best-hypothesis
    best-hypothesis = argmax h ∈new-candidate-hypotheses Performance(h,examples,target_attribute)
    # Update candidate-hypotheses
    candidate-hypotheses = the k best from new-candidate-hypotheses according to Performance.
  prediction = most frequent value of target_attribute from examples that match best-hypothesis
  return IF best-hypothesis THEN prediction


Performance(h, examples, target_attribute)
  h-examples = the set of examples that match h
  return - Entropy(h-examples) wrt target_attribute
```

Rule extraction

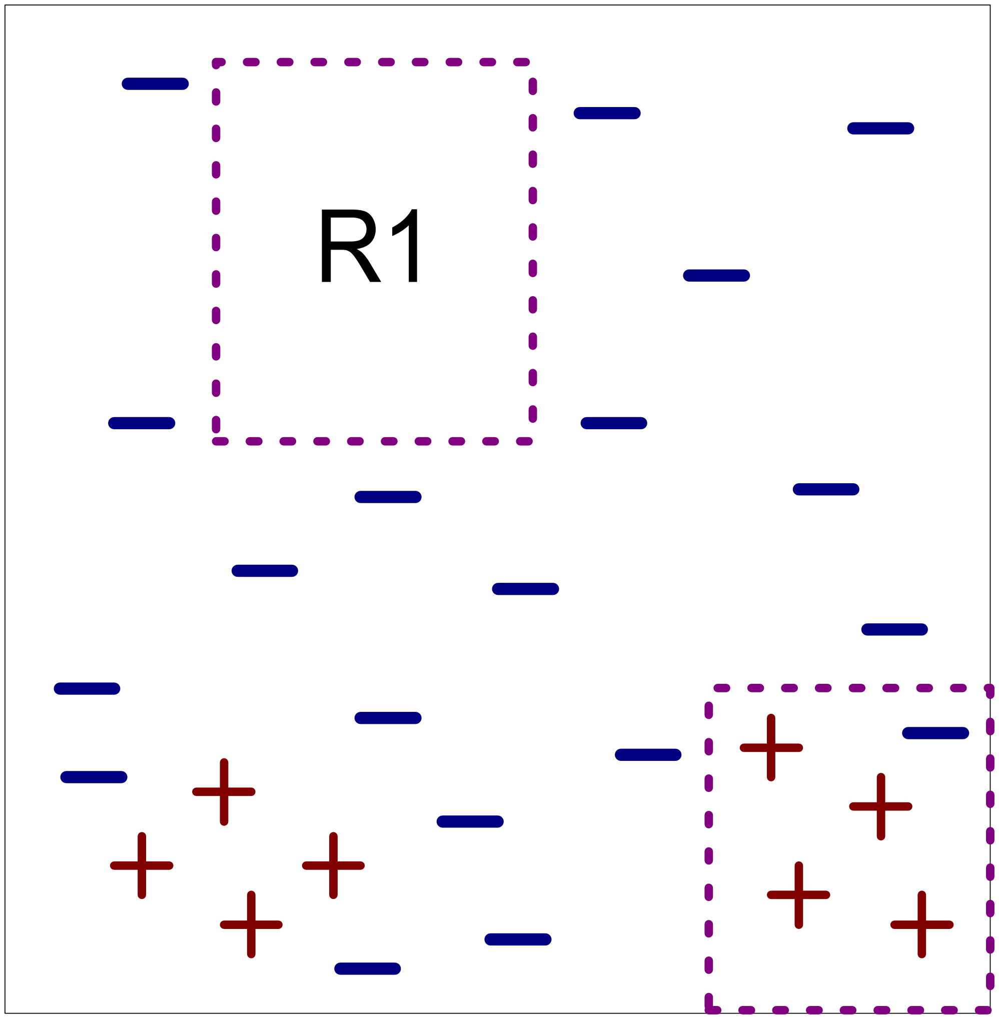
- ✗One rule $A \rightarrow y$ is desirable if
  - ✗Covers most of the positive examples of class y
  - ✗Covers none, or just a few, negative examples
- ✗Finding an optimal rule is computationally expensive
  - ✗The search space grows exponentially
- ✗Approximate solutions

Example of Sequential Covering

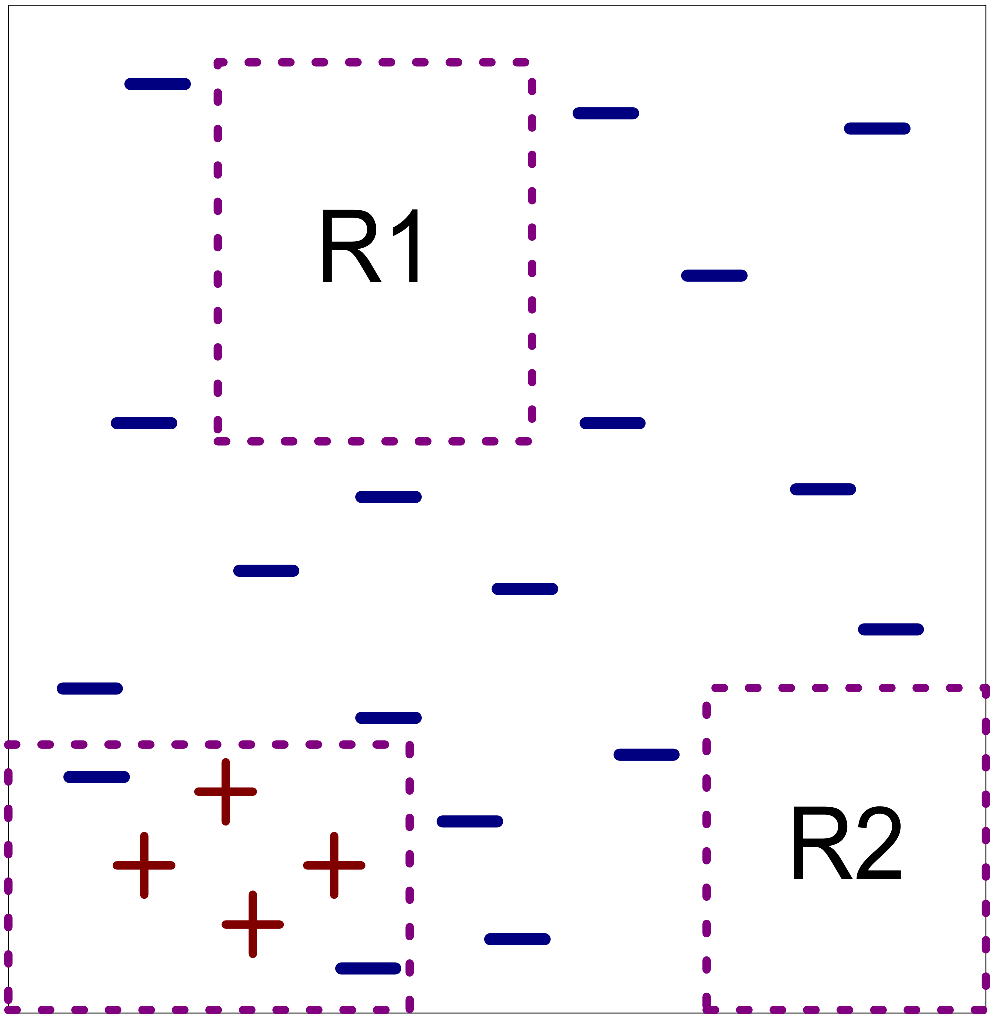

(i) Original Data


(ii) Step 1

Example of Sequential Covering…



(iii) Step 2



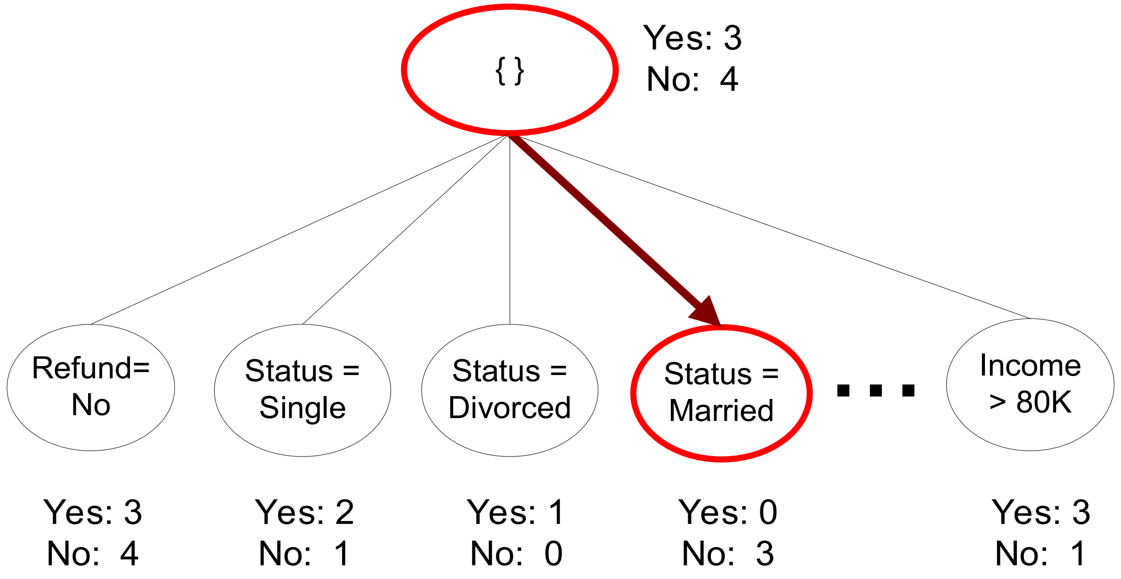
(iv) Step 3

Aspects of Sequential Covering

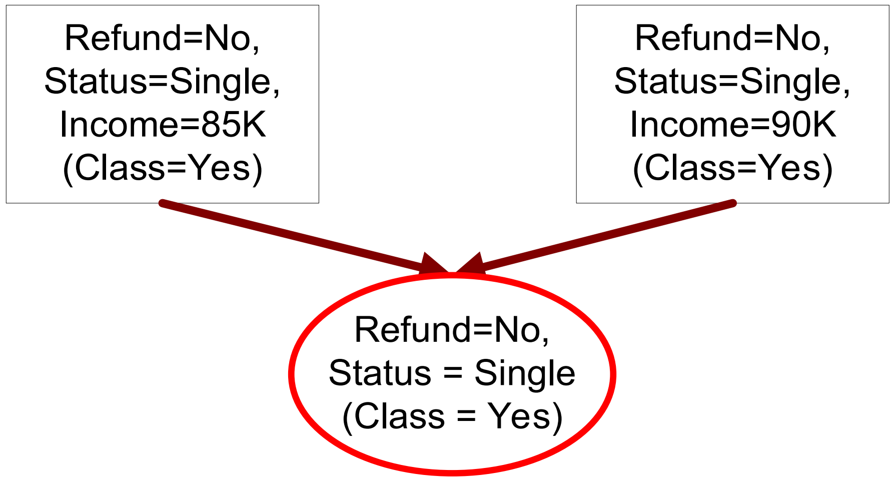
- ✗Rule Growing
- ✗Instance Elimination
- ✗Rule Evaluation
- ✗Stopping Criterion
- ✗Rule Pruning

Rule Growing

✗Two common strategies



(a) General-to-specific


Refund=No, Status=Single, Income=85K (Class=Yes)
Refund=No, Status=Single, Income=90K (Class=Yes)
Refund=No, Status = Single (Class = Yes)

(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: {} => class (initial rule)
    - R1: {A} => class (rule after adding conjunct)
    - 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

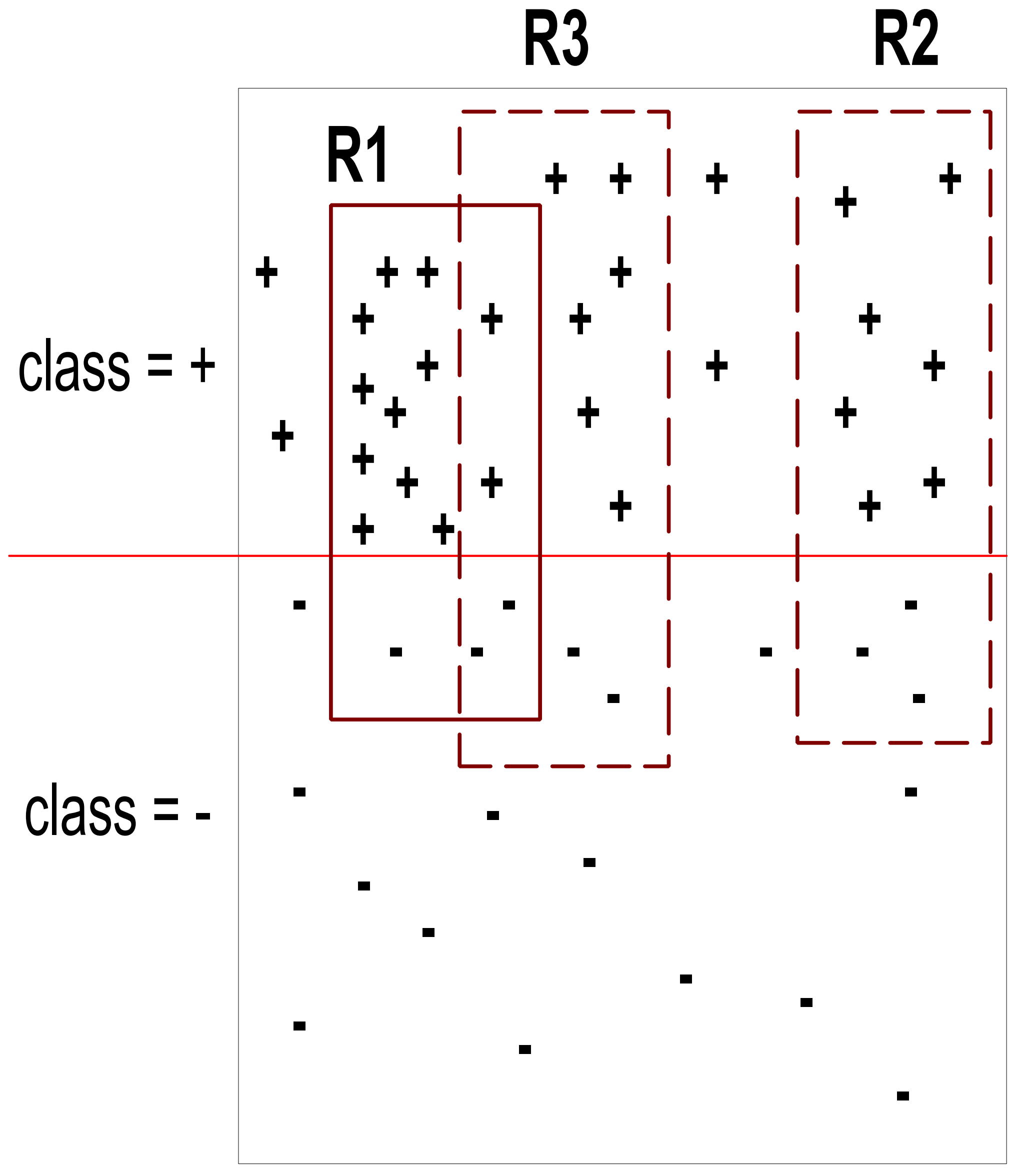
      n0: number of negative instances covered by R0

      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 a bad estimation of the accuracy of a rule
  - ✗Compare rules R2 and R3 in the diagram



Rule Evaluation

- Metrics:
  - Accuracy $= \dfrac{n_c}{n}$
  - Laplace $= \dfrac{n_c + 1}{n + k}$
  - M-estimate $= \dfrac{n_c + kp}{n + k}$

n : Number of instances covered by rule

n_c : Number of instances covered by rule correctly classified

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

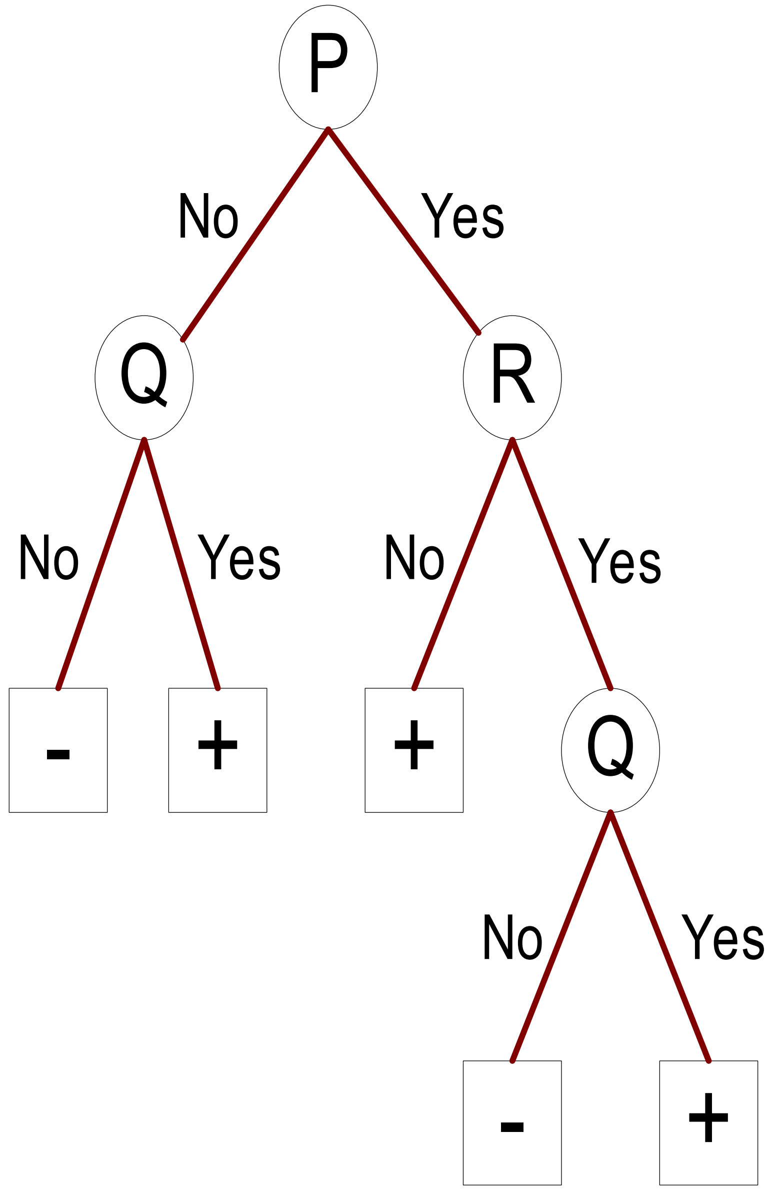
Direct Method: RIPPER

- ✗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 → y,
  - ✗consider an alternative rule r′: A′ → 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

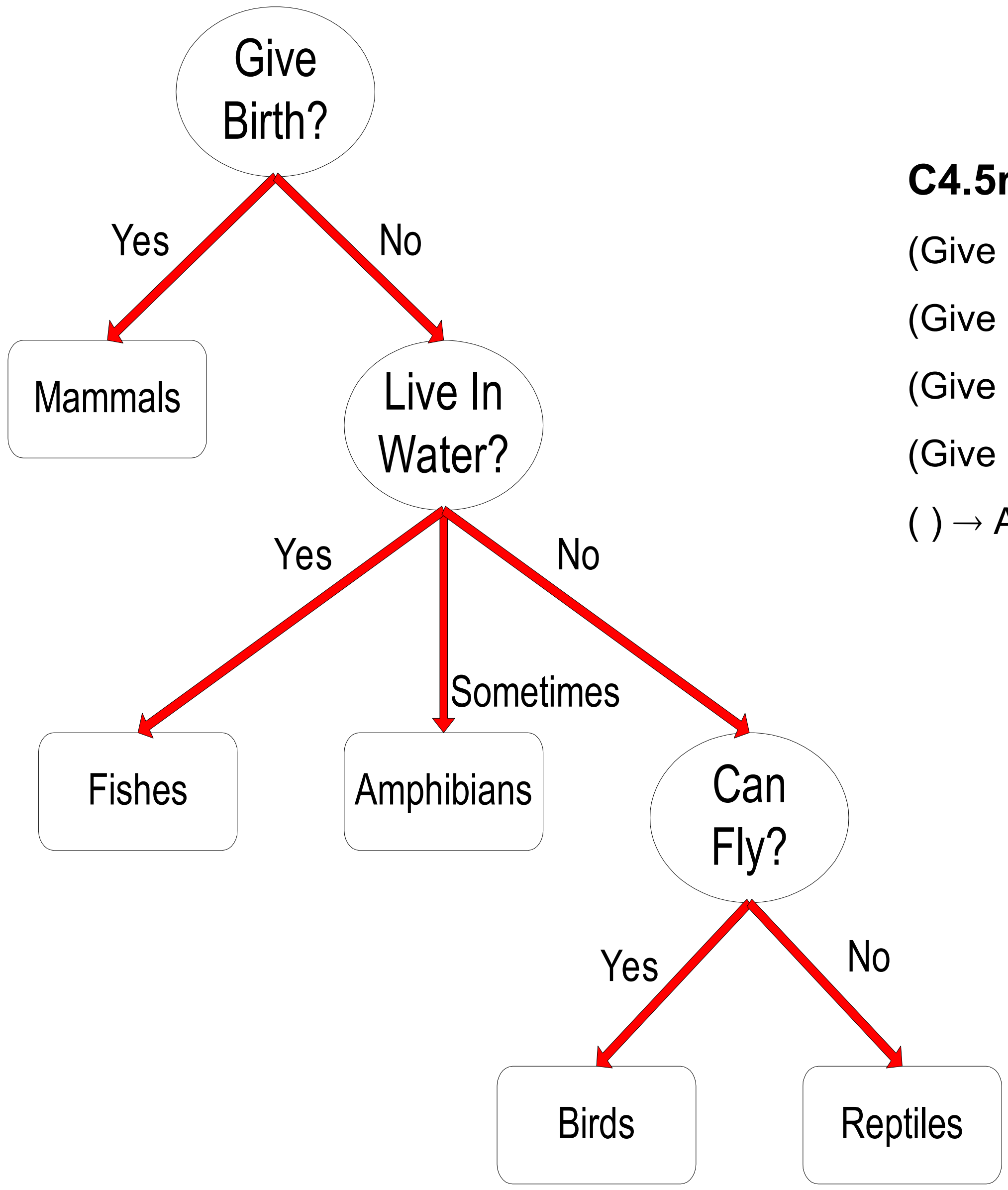
Indirect Method: C4.5rules

- ✗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(error) + g L(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

C4.5 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	Amphibians	0	0	0	0	2
CLASS	Fishes	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

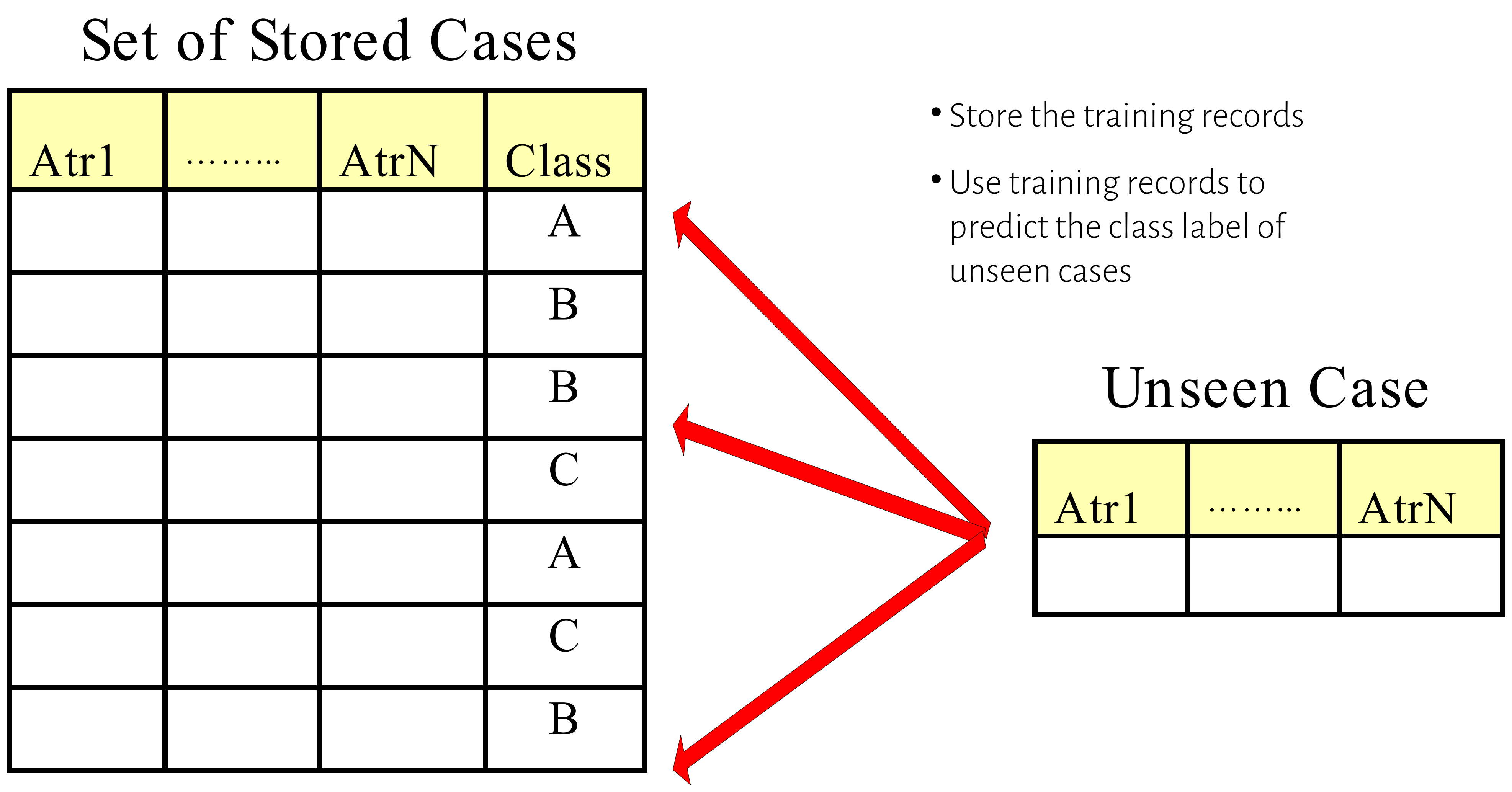
RIPPER:

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

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

Instance-Based Classifiers



Set of Stored Cases

Atr1	………	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

Unseen Case

Atr1	………	AtrN

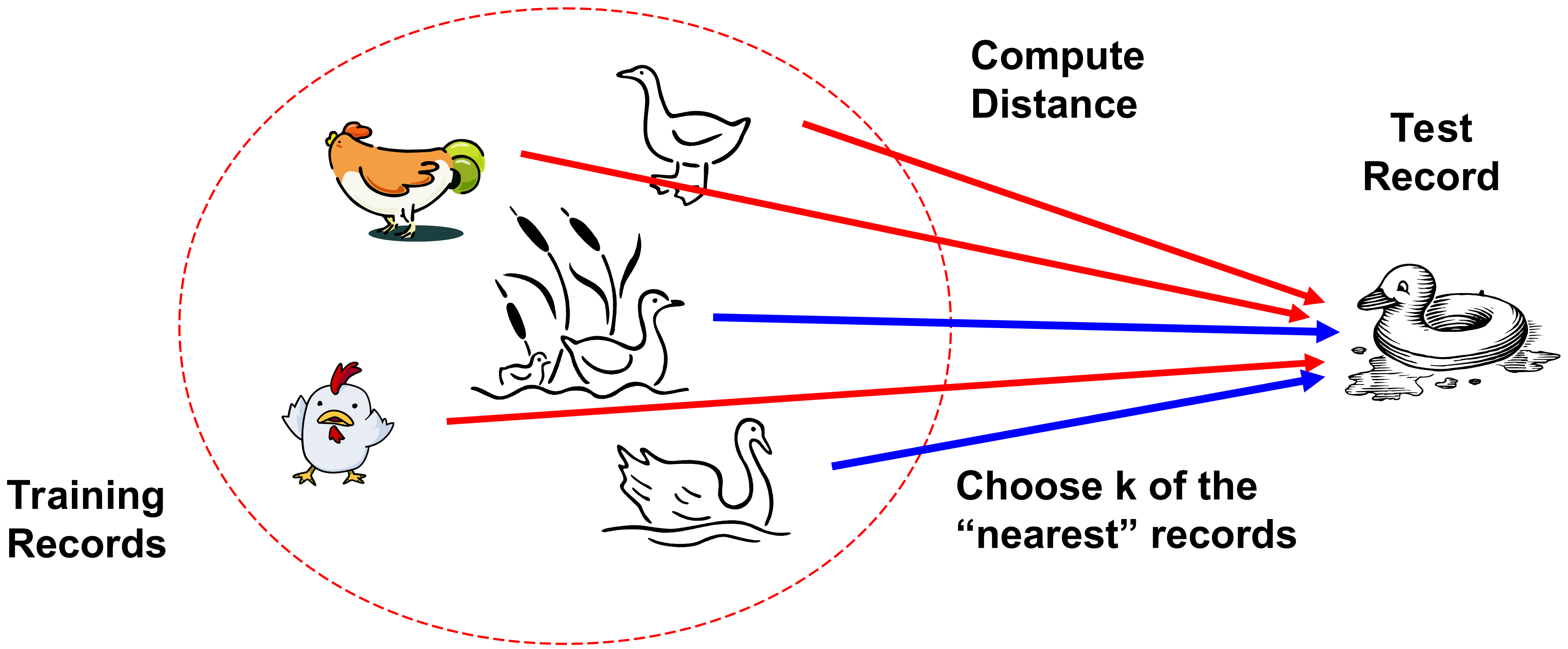
- Store the training records
- Use training records to predict the class label of unseen cases

Instance Based Classifiers

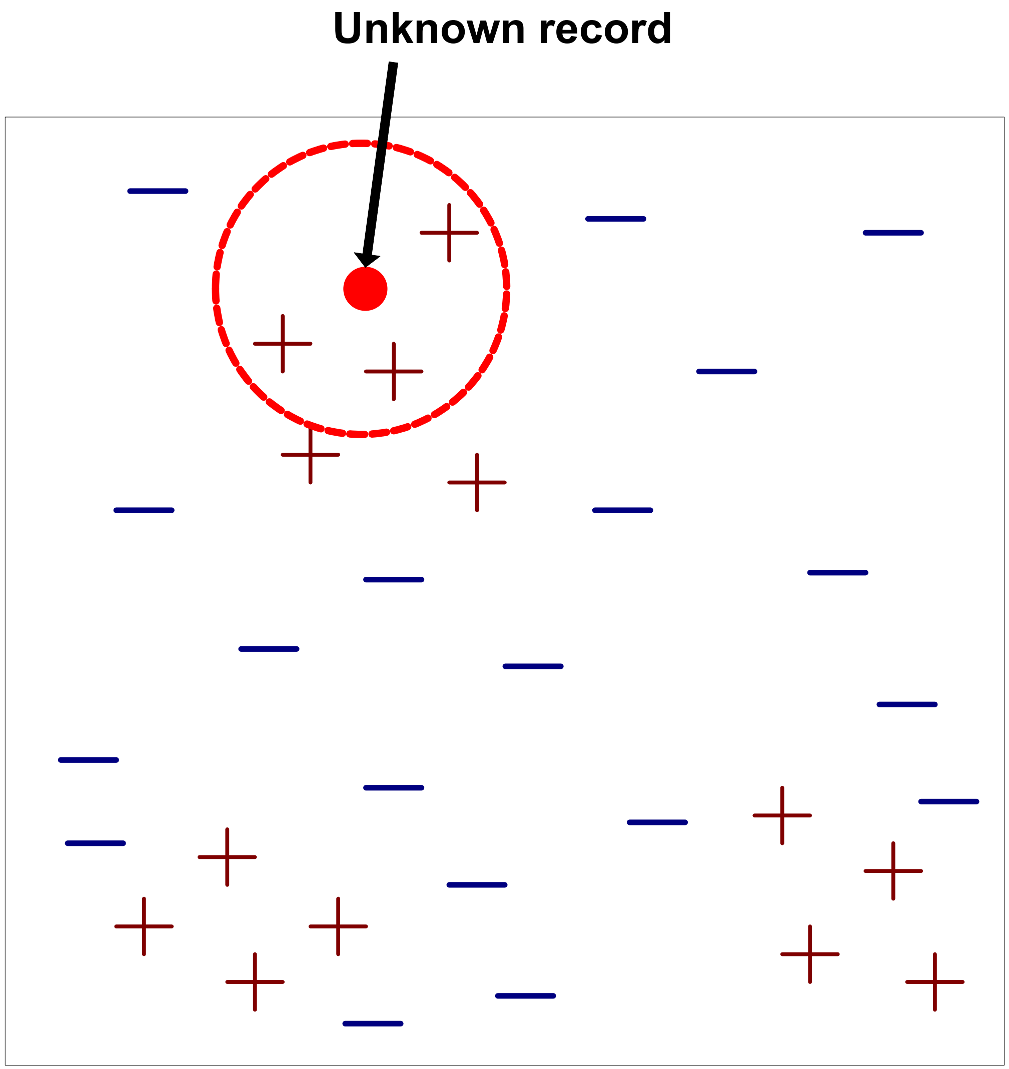
- ✗Examples:
  - ✗Rote-learner
    - ✗ Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - ✗k Nearest neighbor
    - ✗ Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

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



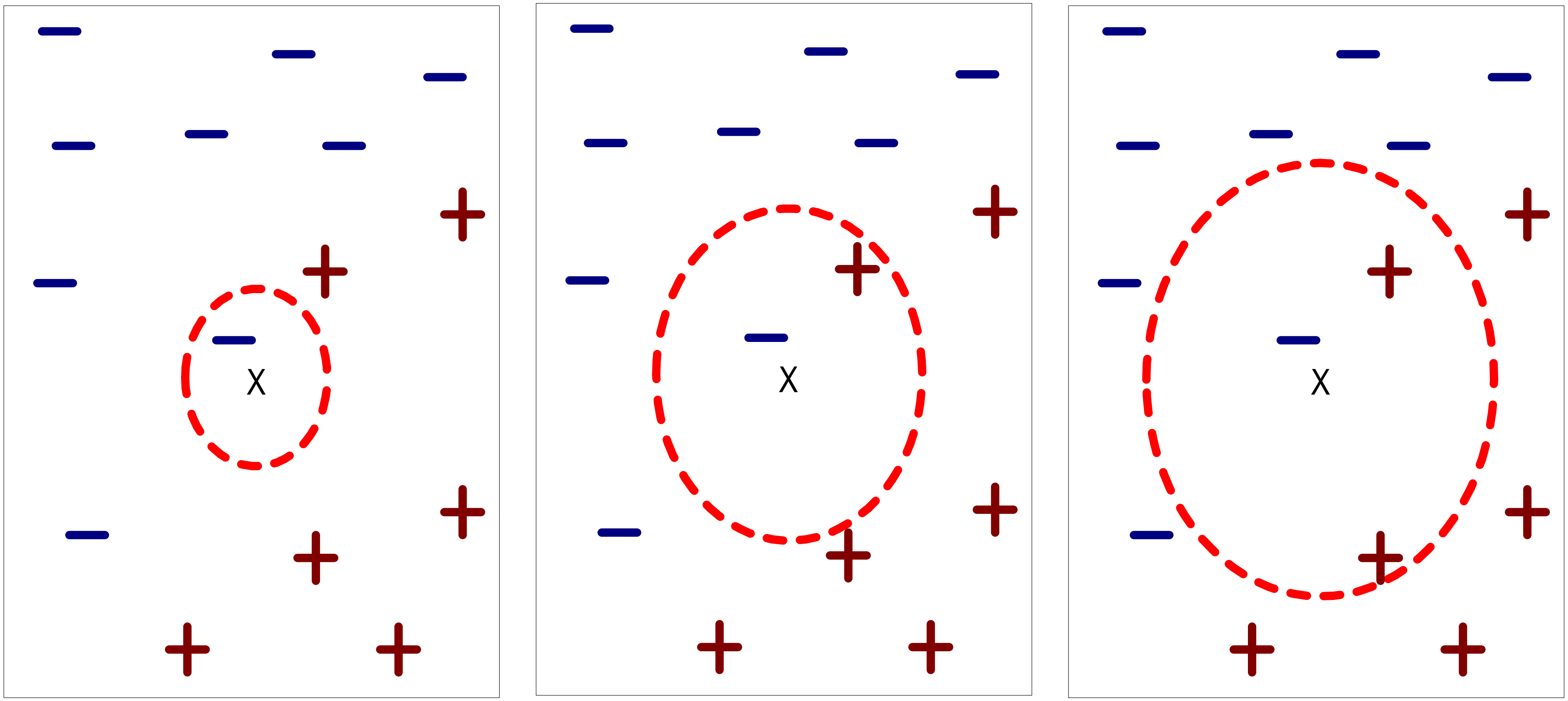
Nearest-Neighbor Classifiers



- 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

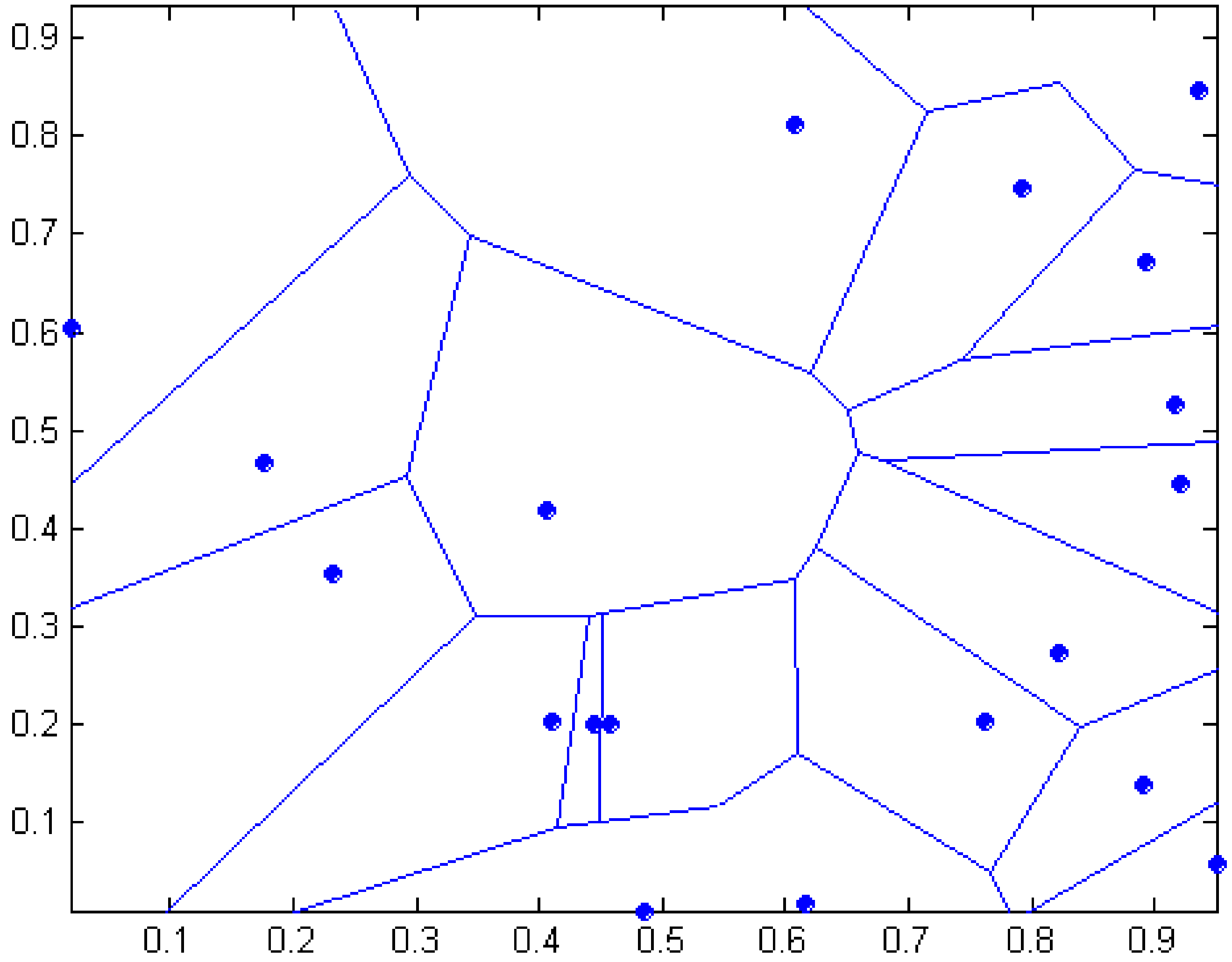


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

1 nearest-neighbor

Voronoi Diagram



Nearest Neighbor Classification

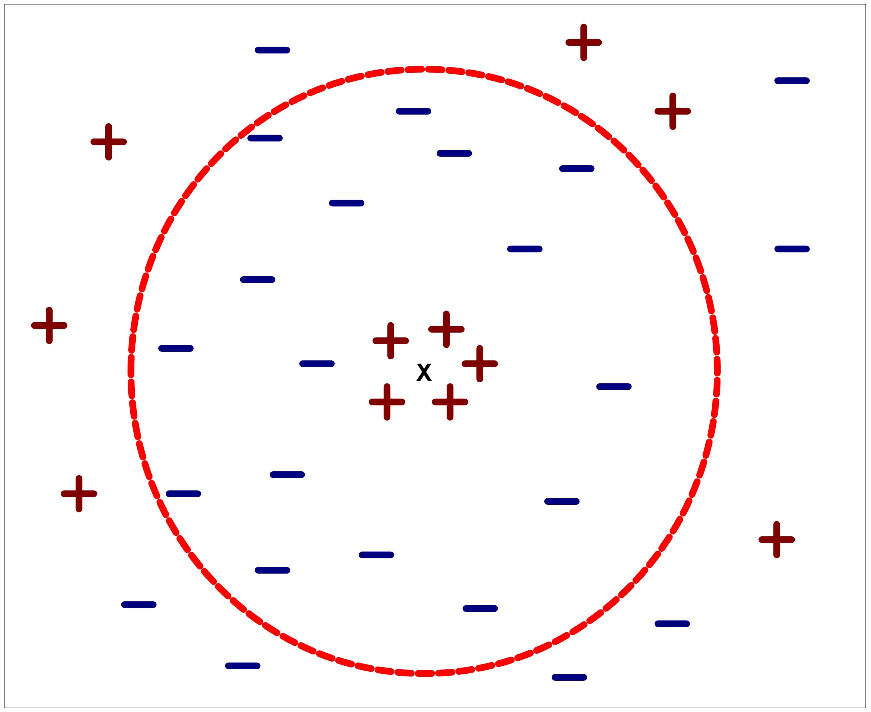
- ✗Compute distance between two points:
  - ✗Euclidean distance

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

- ✗Determine the class from nearest neighbor list
  - ✗take the majority vote of class labels among the k-nearest neighbors
  - ✗Weigh the vote according to distance?
    - ✗ weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- ✗Choosing the value of k:
  - ✗If k is too small, sensitive to noise points
  - ✗If k is too large, neighborhood may include 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

Nearest Neighbor Classification…

- ✗Problem with Euclidean measure:
  - ✗High dimensional data
    - ✗ curse of dimensionality
  - ✗Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		d = 1.4142

Solution: Normalize the vectors to unit length

Nearest neighbor Classification…

- ✗K-NN CLASSIFIERS ARE LAZY LEARNERS
  - ✗It does not build models explicitly
  - ✗Unlike eager learners such as decision tree induction and rule-based systems
  - ✗Classifying unknown records are relatively expensive

Example: PEBLS

- ✗PEBLS: Parallel Examplar-Based Learning System (Cost & Salzberg)
  - ✗Works with both continuous and nominal features
    - ✗For nominal features, distance between two nominal values is computed using modified value difference metric (MVDM)
  - ✗Each record is assigned a weight factor
  - ✗Number of nearest neighbor, k = 1

Example: PEBLS

Class	Marital Status		
	Single	Married	Divorced
Yes	2	0	1
No	2	4	1

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

Class	Refund	
	Yes	No
Yes	0	3
No	3	4

Distance between nominal attribute values:

$$d(V_1,V_2)=\sum_i \left|\frac{n_{1i}}{n_1}-\frac{n_{2i}}{n_2}\right|$$

d(Single,Married)

$= |2/4-0/4| + |2/4-4/4| = 1$

d(Single,Divorced)

$= |2/4-1/2| + |2/4-1/2| = 0$

d(Married,Divorced)

$= |0/4-1/2| + |4/4-1/2| = 1$

d(Refund=Yes,Refund=No)

$= |0/3-3/7| + |3/3-4/7| = 6/7$

Example: PEBLS

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
X	Yes	Single	125K	**No**
Y	No	Married	100K	**No**

Distance between record X and record Y:

$$\Delta(X,Y) = w_X w_Y \sum_{i=1}^{d} d(X_i, Y_i)^2$$

where:

$$w_X = \frac{\text{Number of times X is used for prediction}}{\text{Number of times X predicts correctly}}$$

$w_X \cong 1$ if X makes accurate prediction most of the time

$w_X > 1$ if X is not reliable for making predictions

Bayes Classifier

- ✗A probabilistic framework for solving classification problems
- ✗Conditional Probability:

$$P(C|A)=\frac{P(A,C)}{P(A)}$$

$$P(A|C)=\frac{P(A,C)}{P(C)}$$

- ✗Bayes theorem:

$$P(C|A)=\frac{P(A|C)P(C)}{P(A)}$$

Example of Bayes Theorem

- ✗Given:
  - ✗A doctor knows that meningitis causes stiff neck 50% of the time
  - ✗Prior probability of any patient having meningitis is 1/50,000
  - ✗Prior probability of any patient having stiff neck is 1/20

- ✗ If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- ✗Consider each attribute and class label as random variables

- ✗Given a record with attributes ($A_1, A_2,\dots,A_N$)
  - ✗Goal is to predict class C
  - ✗Specifically, we want to find the value of C that maximizes $P(C| A_1, A_2,\dots,A_n)$

- ✗Can we estimate $P(C| A_1, A_2,\dots,A_N)$ directly from data?

Bayesian Classifiers

✗APPROACH:

✗compute the posterior probability P(C | A_1, A_2, …, A_n) for all values of C using the Bayes theorem

$$P(C|A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n|C)\, P(C)}{P(A_1 A_2 \ldots A_n)}$$

✗Choose value of C that maximizes
P(C | A_1, A_2, …, A_n)

✗Equivalent to choosing value of C that maximizes
P(A_1, A_2, …, A_n|C) P(C)

✗HOW TO ESTIMATE P(A_1, A_2, …, A_N | C)?

Naïve Bayes Classifier

- ✗Assume independence among attributes A_I when class is given:
  - ✗$P(A_1, A_2, \ldots, A_n | C) = P(A_1| C_j)\, P(A_2| C_j) \ldots P(A_n| C_j)$
  - ✗Can estimate $P(A_i| C_j)$ for all A_i and C_j.
  - ✗New point is classified to C_j if $P(C_j) \Pi P(A_i| C_j)$ is maximal.

How to Estimate Probabilities from Data?

- ✗CLASS: $P(C) = N_c/N$
  - ✗e.g., $P(No) = 7/10$, $P(Yes) = 3/10$

- ✗FOR DISCRETE ATTRIBUTES:

  $P(A_I \mid C_K) = |A_{IK}| / N_C$

  - ✗where $|A_{ik}|$ is number of instanc belongs to class C_k
  - ✗Examples:

    $P(Status=Married|No) = 4/7$
    $P(Refund=Yes|Yes)=0$

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
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**

How to Estimate Probabilities from Data?

- ✗FOR CONTINUOUS ATTRIBUTES:
  - ✗Discretize the range into bins
    - ✗ one ordinal attribute per bin
    - ✗ violates independence assumption
  - ✗Two-way split: (A < v) or (A > v)
    - ✗ choose only one of the two splits as new attribute
  - ✗Probability density estimation:
    - ✗ Assume attribute follows a normal distribution
    - ✗ Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - ✗ Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

k

How to Estimate Probabilities from Data?

✗NORMAL DISTRIBUTION:

$$P(A_i|c_j)=\frac{1}{\sqrt{2\pi\sigma_{ij}^2}}e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

✗One for each (A_i,c_i) pair

✗FOR (INCOME, CLASS=NO):

✗If Class=No

✗ sample mean = 110

✗ sample variance = 2975

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
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**

$$P(Income=120|No)=\frac{1}{\sqrt{2\pi}(54.54)}e^{-\frac{(120-110)^2}{2(2975)}}=0.0072$$

Example of Naïve Bayes Classifier

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:

If class=No:	sample mean=110
	sample variance=2975
If class=Yes:	sample mean=90
	sample variance=25

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

- $P(X|\text{Class=No}) = P(\text{Refund=No}|\text{Class=No}) \times P(\text{Married}|\text{Class=No}) \times P(\text{Income=120K}|\text{Class=No})$
  $= 4/7 \times 4/7 \times 0.0072 = 0.0024$
- $P(X|\text{Class=Yes}) = P(\text{Refund=No}|\text{Class=Yes}) \times P(\text{Married}|\text{Class=Yes}) \times P(\text{Income=120K}|\text{Class=Yes})$
  $= 1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since $P(X|No)P(No) > P(X|Yes)P(Yes)$

Therefore $P(No|X) > P(Yes|X)$

=> Class = No

Naïve Bayes Classifier

- ✗If one of the conditional probability is zero, then the entire expression becomes zero
- ✗Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

Example of Naïve Bayes Classifier

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

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

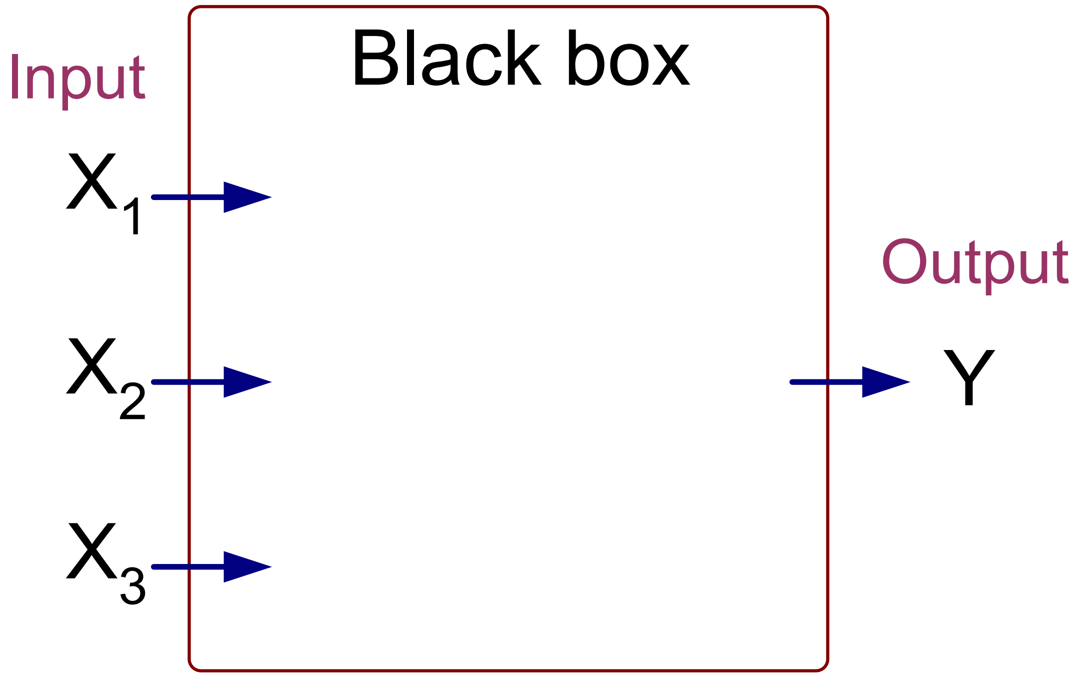
=> Mammals

Naïve Bayes (Summary)

- ✗Robust to isolated noise points
- ✗Handle missing values by ignoring the instance during probability estimate calculations
- ✗Robust to irrelevant attributes
- ✗Independence assumption may not hold for some attributes
  - ✗Use other techniques such as Bayesian Belief Networks (BBN)

Artificial Neural Networks (ANN)

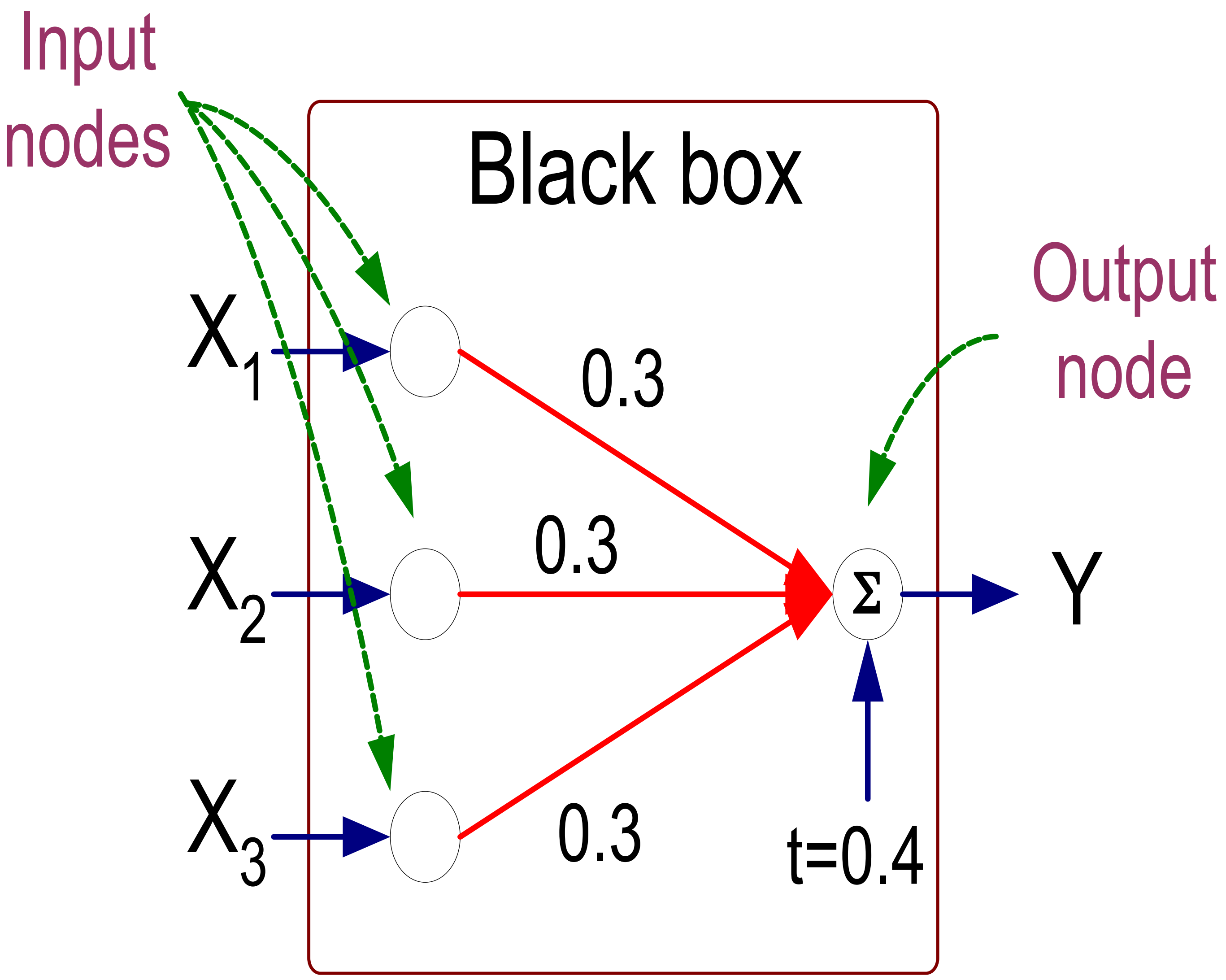
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

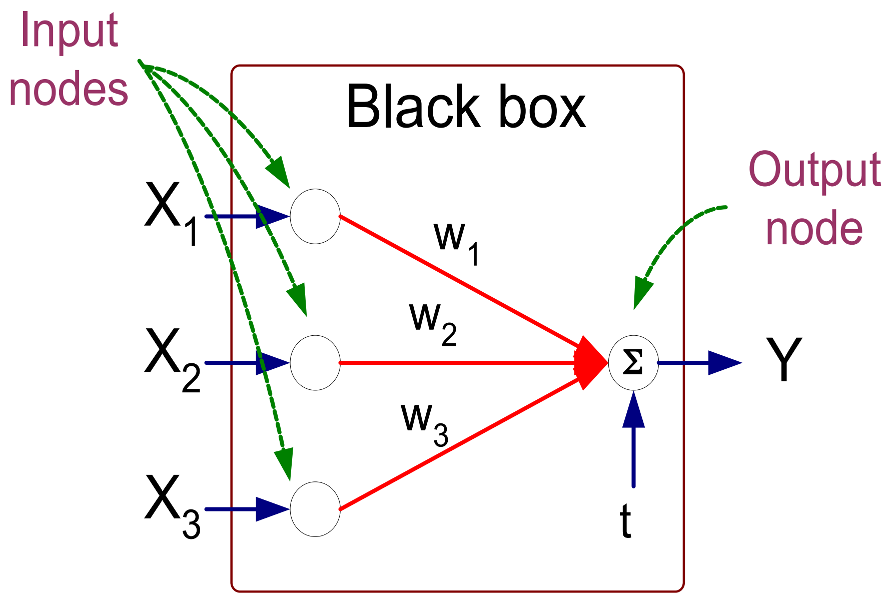
Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Artificial Neural Networks (ANN)

- ✗Model is an assembly of inter-connected nodes and weighted links
- ✗Output node sums up each of its input value according to the weights of its links
- ✗Compare output node against some threshold t

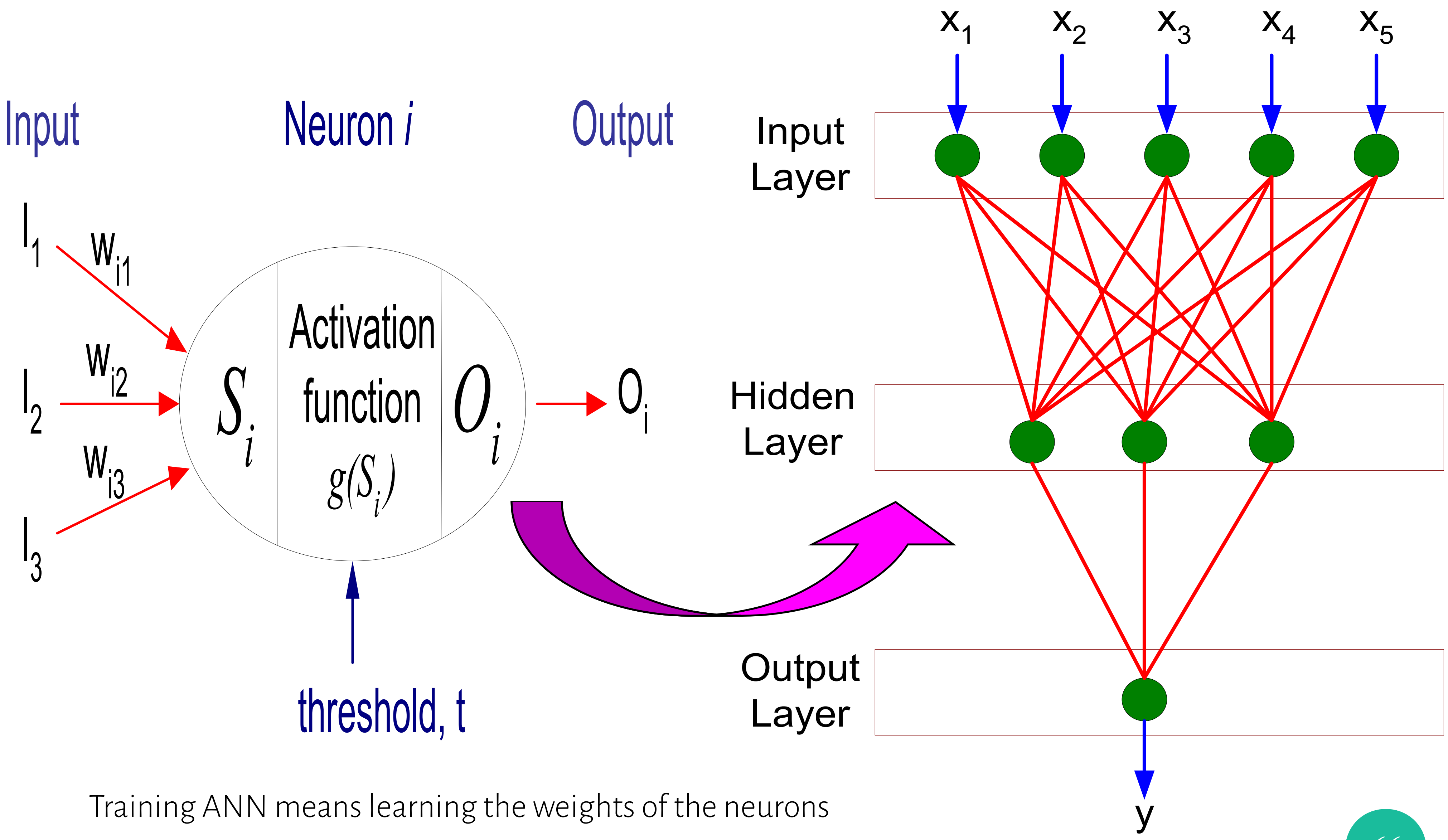


Perceptron Model

$$Y = I\left(\sum_i w_i X_i - t\right) \quad \text{or}$$

$$Y = sign\left(\sum_i w_i X_i - t\right)$$

General Structure of ANN



Training ANN means learning the weights of the neurons

Algorithm for learning ANN

- ✗Initialize the weights ($w_0, w_1, \ldots, w_k$)

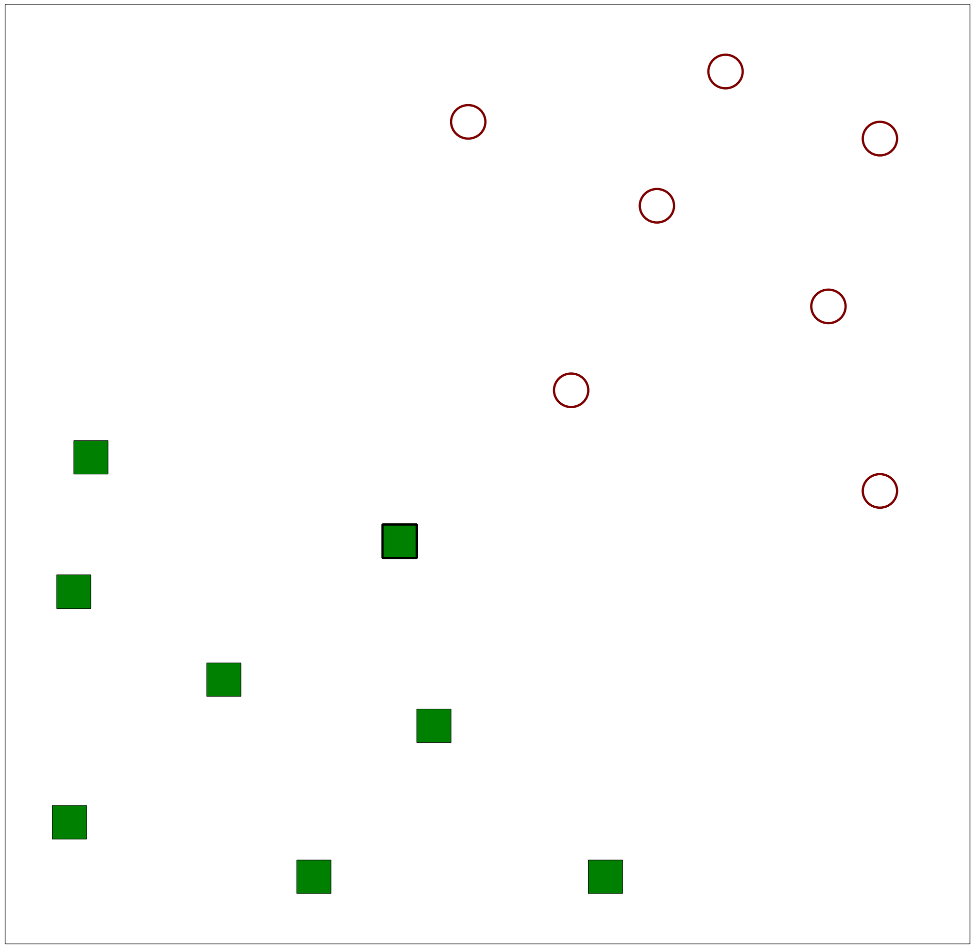
- ✗Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - ✗Objective function:

$$E = \sum_i \left[Y_i - f(w_i, X_i) \right]^2$$

  - ✗Find the weights w_i's that minimize the above objective function
    - ✗ e.g., backpropagation algorithm (see lecture notes)

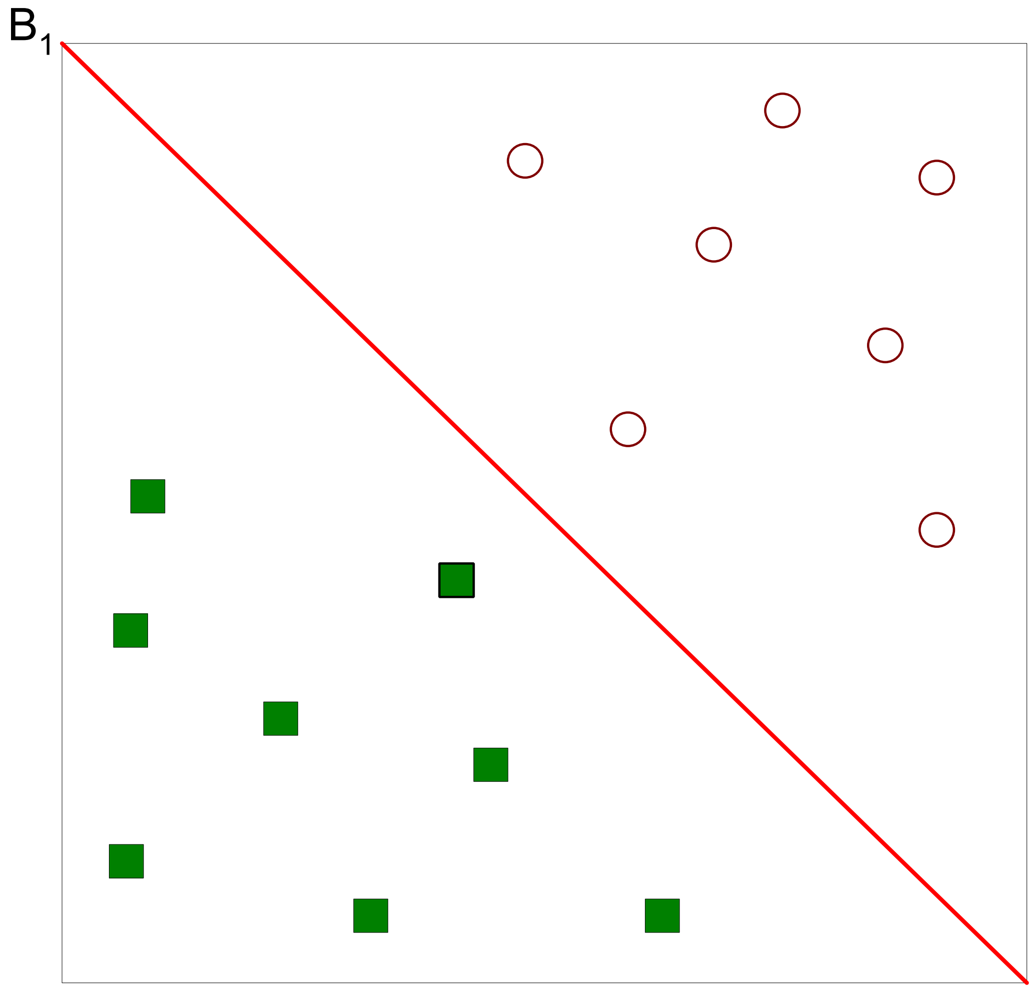
Support Vector Machines

✗Find a linear hyperplane (decision boundary) that will separate the data



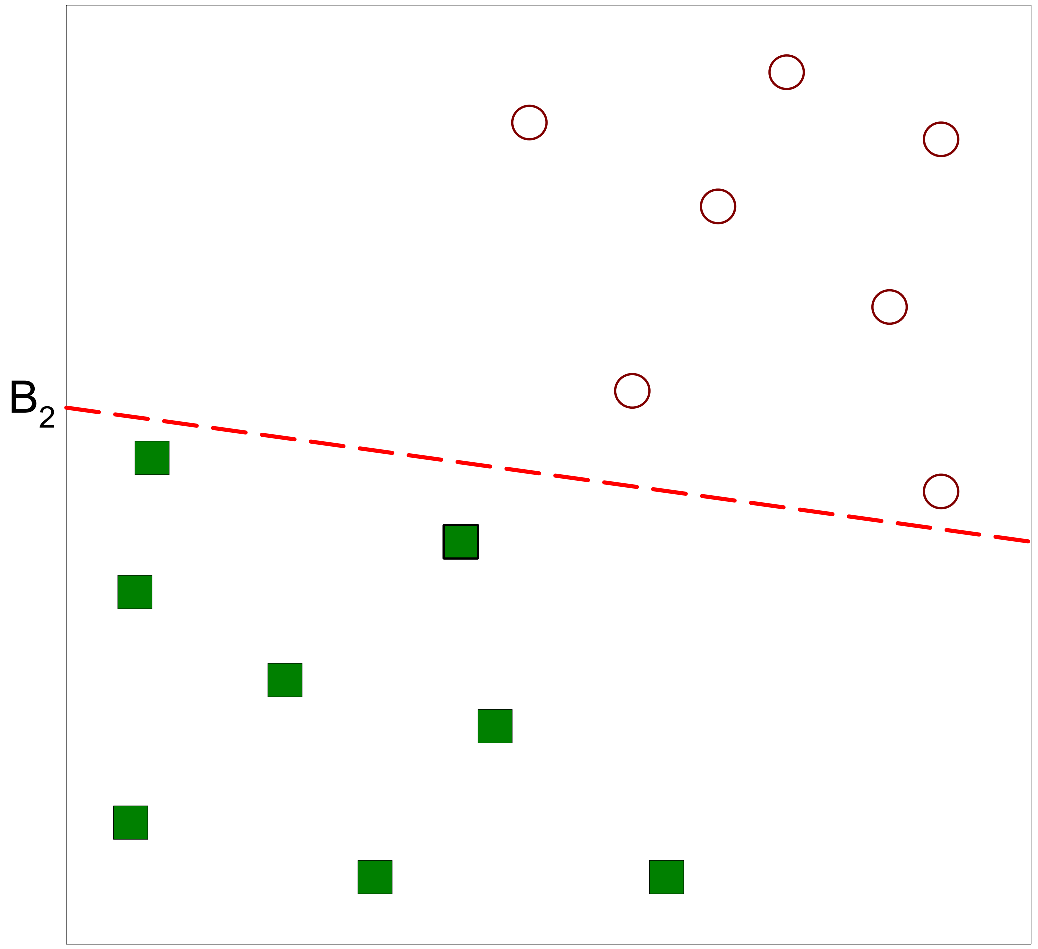
Support Vector Machines

✗One Possible Solution



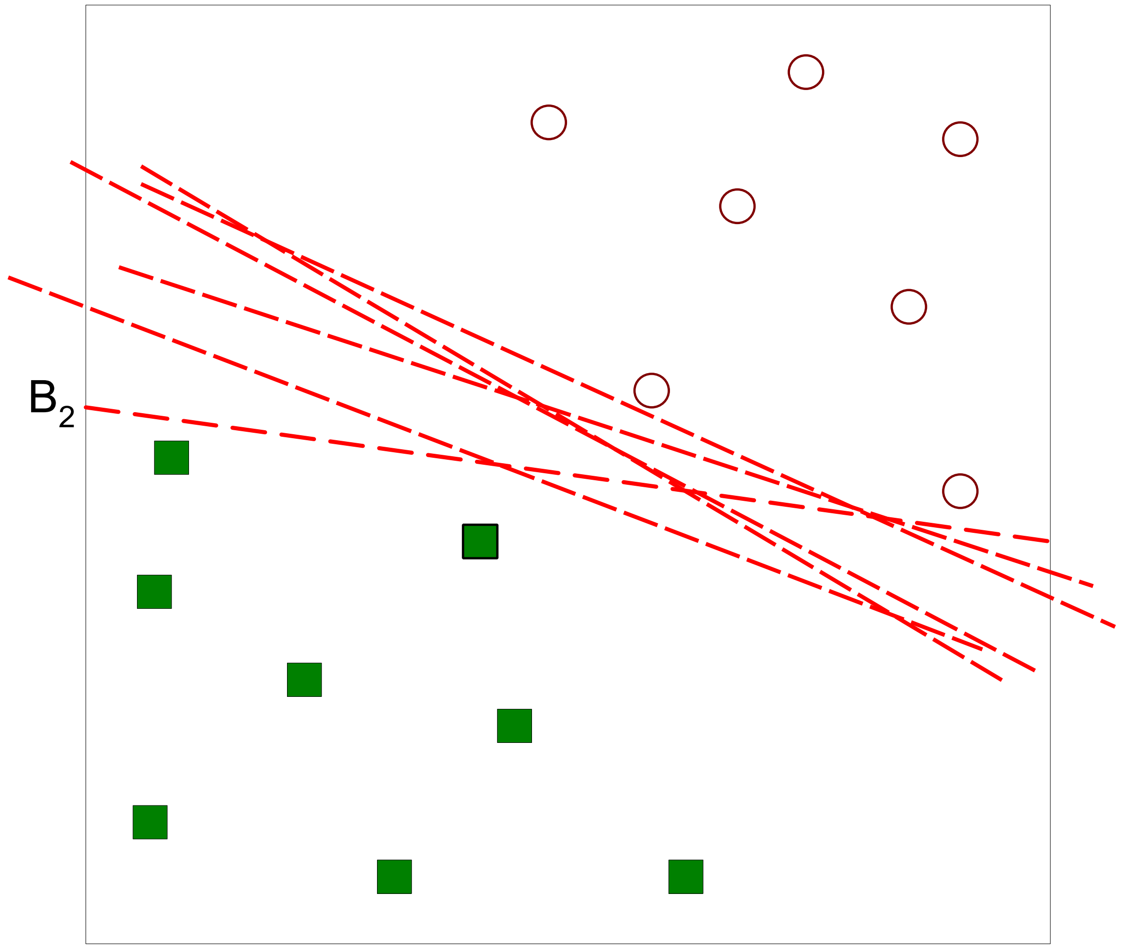
Support Vector Machines

✗Another possible solution



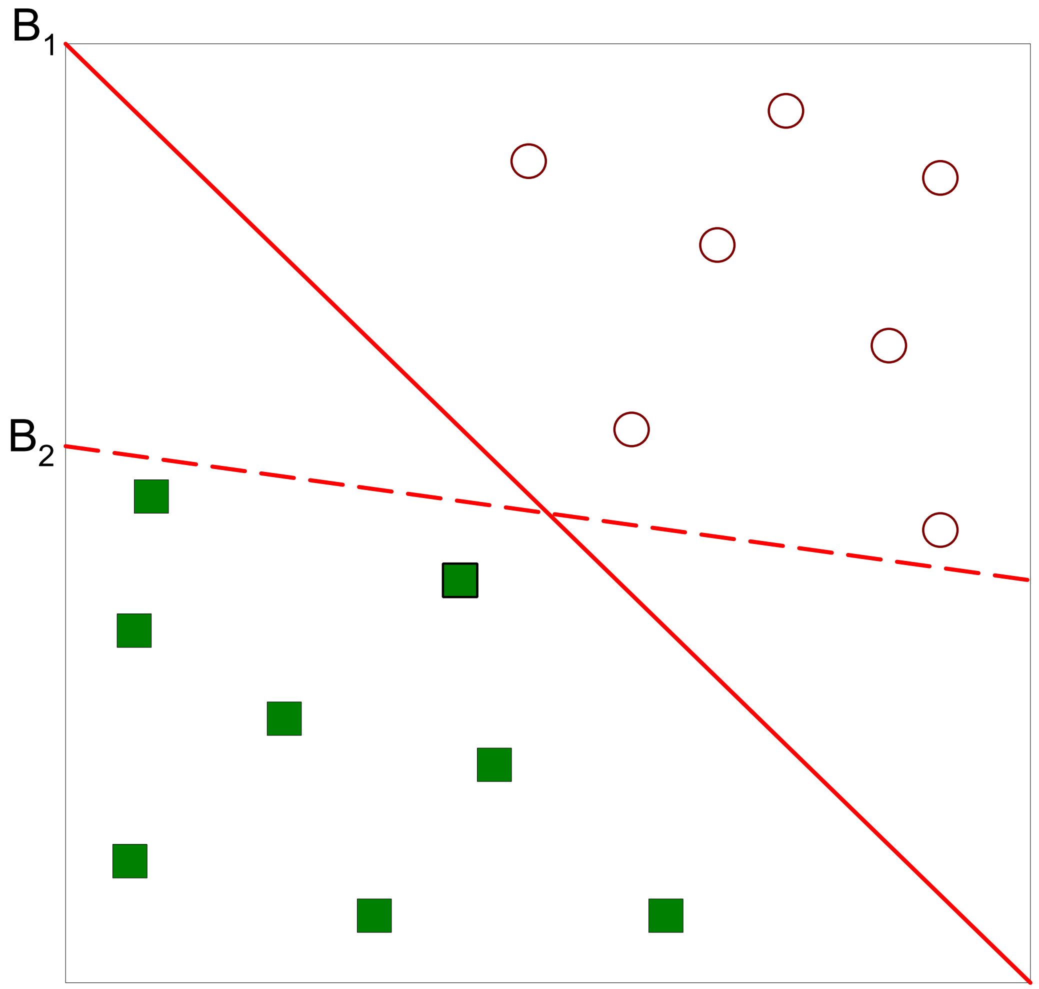
Support Vector Machines

✗Other possible solutions



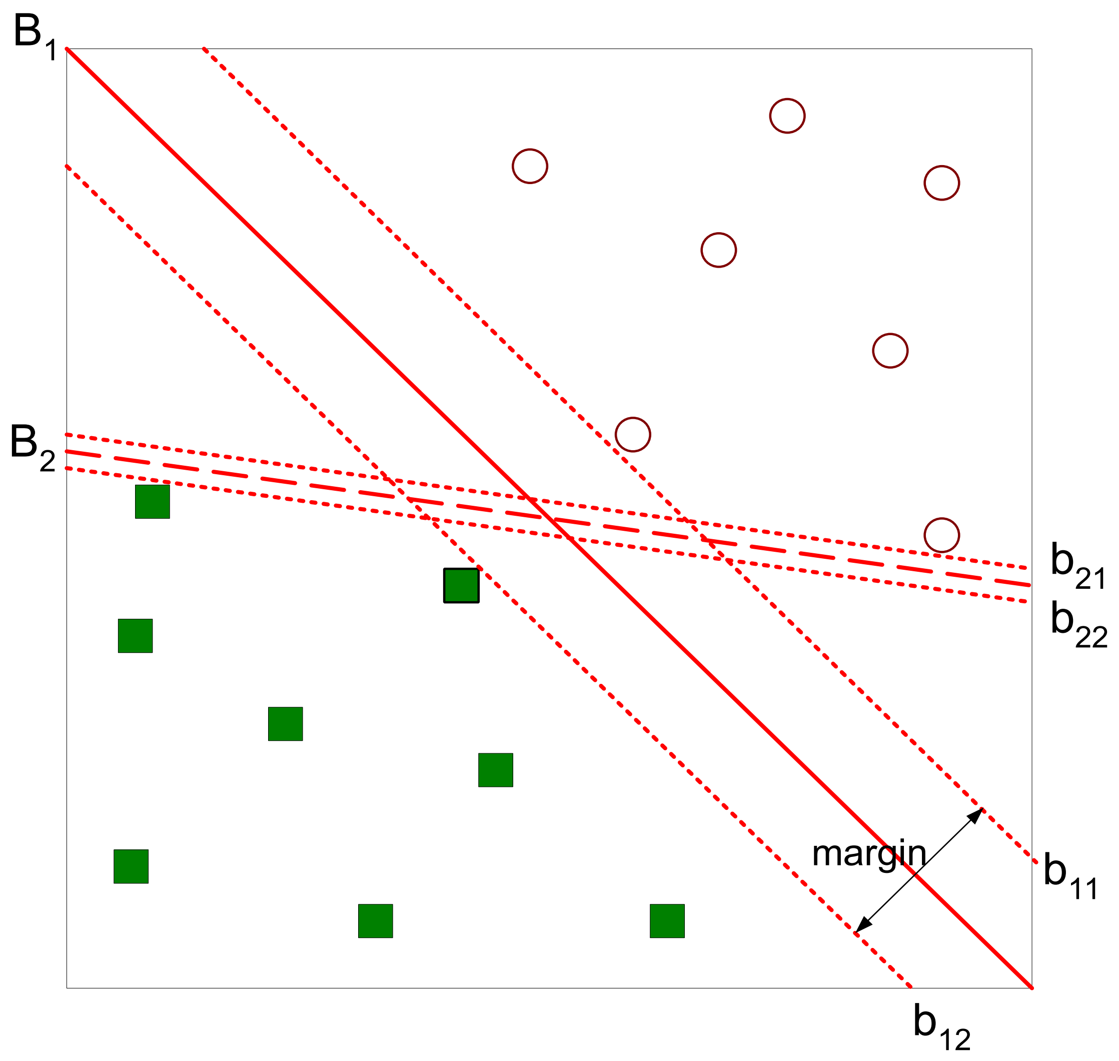
Support Vector Machines

✗Which one is better? B1 or B2?
✗How do you define better?

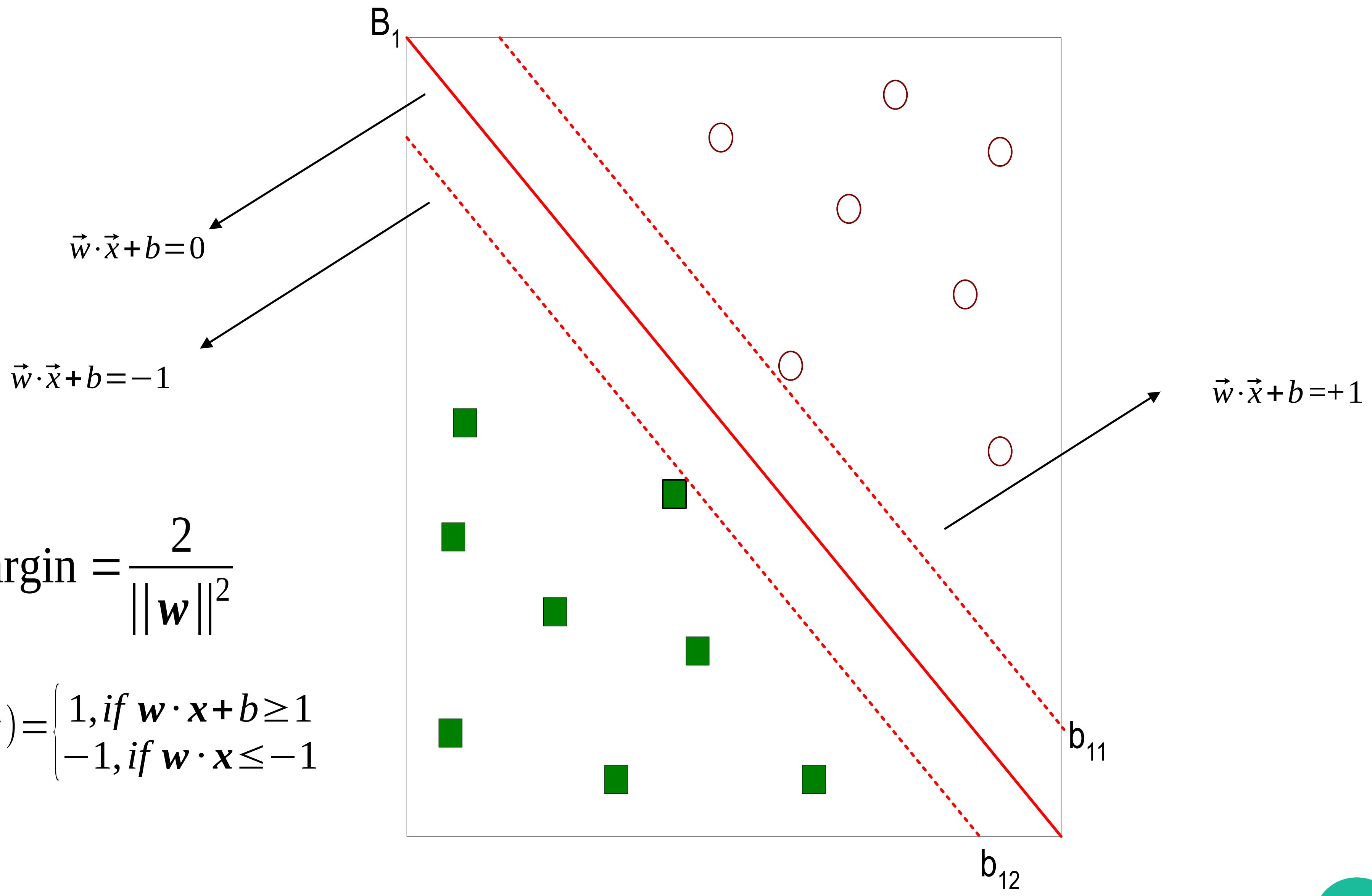


Support Vector Machines

✗Find hyperplane maximizes the margin => B1 is better than B2



Support Vector Machines



B_1
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=+1$
b_{11}
b_{12}

$$\text{Margin} = \frac{2}{\|\mathbf{w}\|^2}$$

$$f(\mathbf{x}) = \begin{cases} 1, \text{if } \mathbf{w}\cdot\mathbf{x}+b \geq 1 \\ -1, \text{if } \mathbf{w}\cdot\mathbf{x} \leq -1 \end{cases}$$

Support Vector Machines

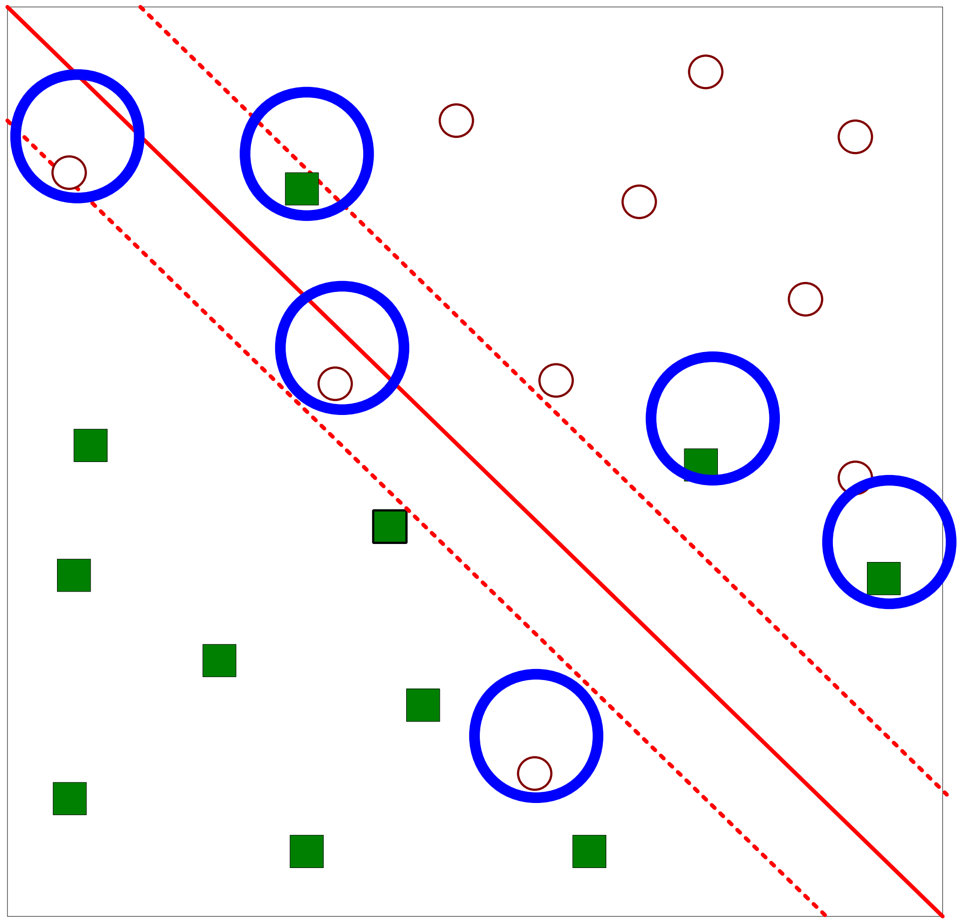
- ✗WE WANT TO MAXIMIZE: $$\text{Margin} = \frac{2}{\|\mathbf{w}\|^2}$$
  - ✗Which is equivalent to minimizing: $$L(w) = \frac{\|\mathbf{w}\|^2}{2}$$
  - ✗But subjected to the following constraints:

$$f(\mathbf{x}) = \begin{cases} 1, if\ \mathbf{w}\cdot\mathbf{x}+b \geq 1 \\ -1, if\ \mathbf{w}\cdot\mathbf{x} \leq -1 \end{cases}$$

    - ✗ This is a constrained optimization problem
      - ✗Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

✗What if the problem is not linearly separable?



Support Vector Machines

- ✗What if the problem is not linearly separable?
  - ✗Introduce slack variables
    - ✗ Need to minimize:

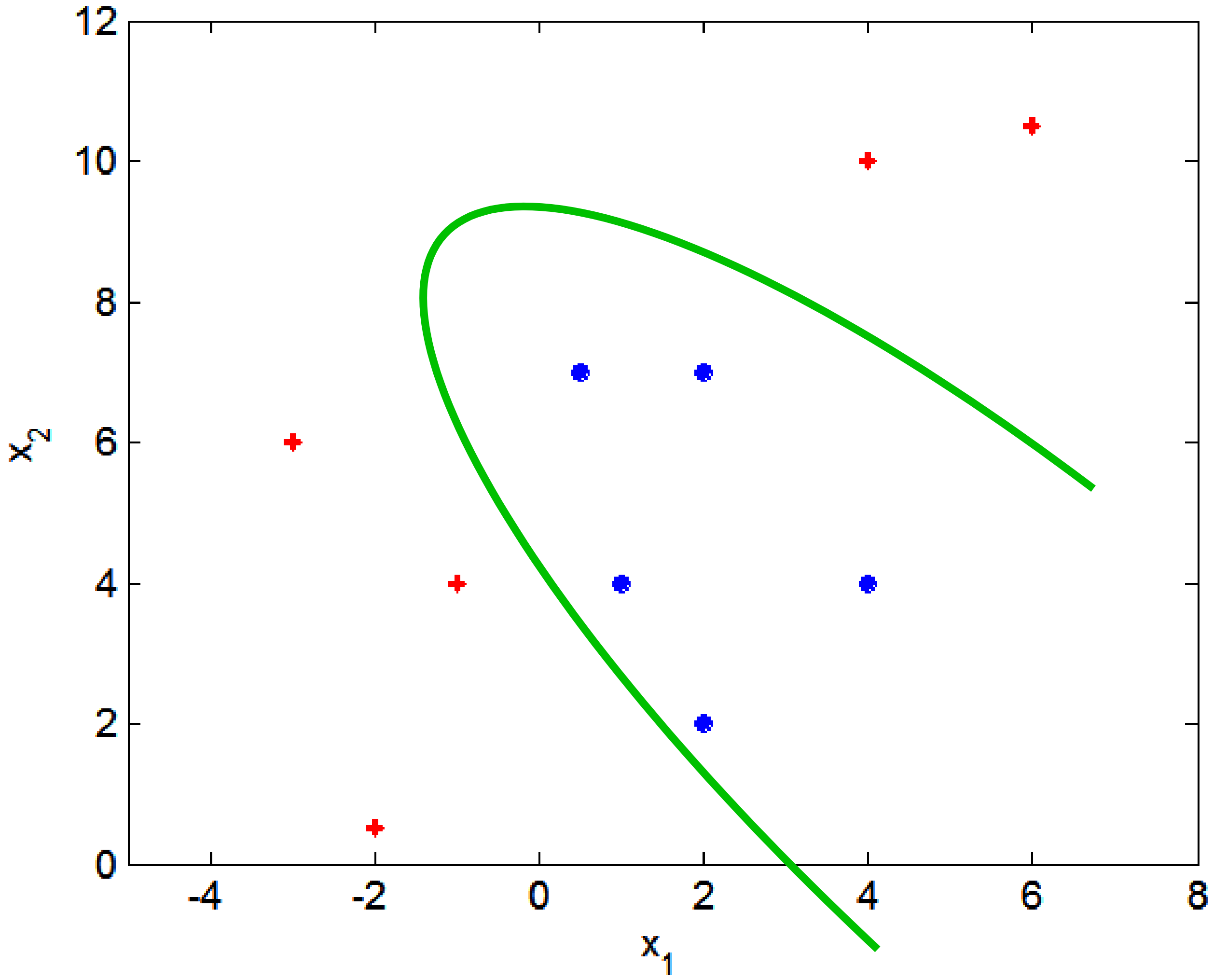
$$L(w)=\frac{\|\vec{w}\|^2}{2}+C\left(\sum_{i=1}^{N}\xi_i^k\right)$$

    - ✗ Subject to:

$$f(\boldsymbol{x})=\begin{cases}1, if\ \boldsymbol{w}\cdot\boldsymbol{x}+b\geq 1-\xi_i\\ -1, if\ \boldsymbol{w}\cdot\boldsymbol{x}\leq -1+\xi_i\end{cases}$$

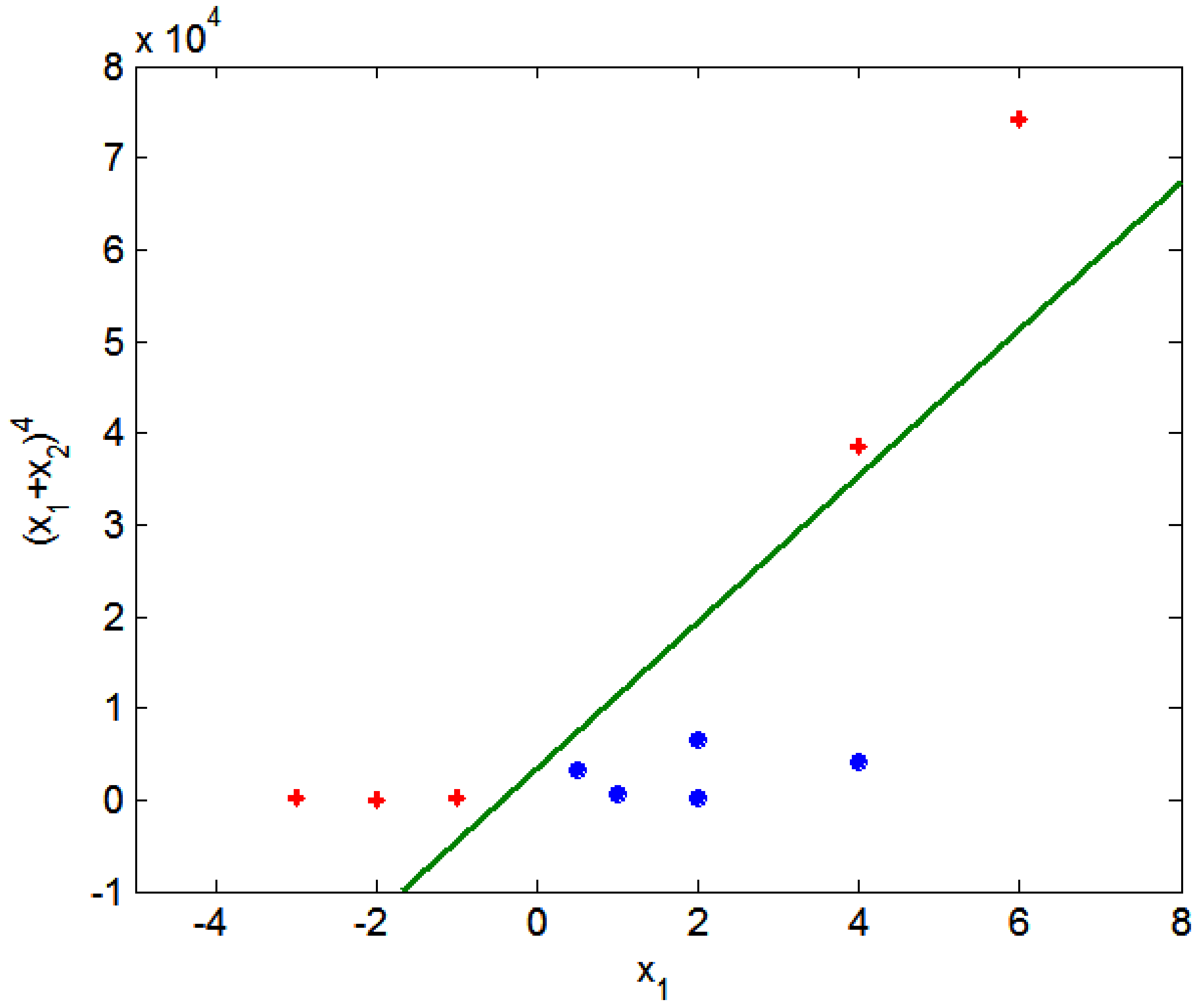
Nonlinear Support Vector Machines

✗What if decision boundary is not linear?



Nonlinear Support Vector Machines

✗Transform data into higher dimensional space: kernel trick



More than two classes

- ✗Transform a N class problem into M two-class problems
- ✗Output coding
  - ✗One – vs – all
  - ✗One – vs – one
  - ✗Error correcting output codes (ECOC)

Ensemble Methods

- ✗Construct a set of classifiers from the training data

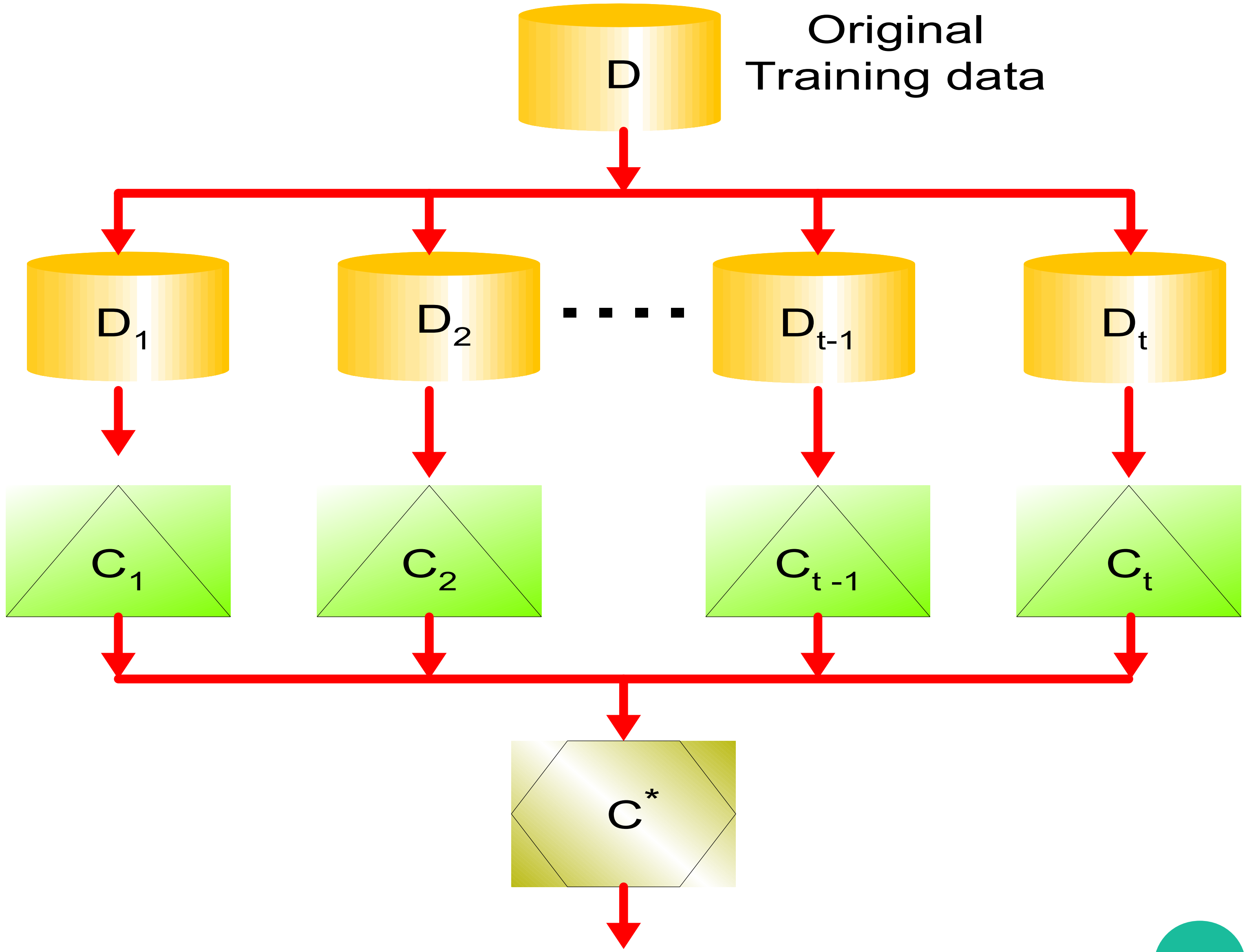
- ✗Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea

Step 1: Create Multiple Data Sets

Step 2: Build Multiple Classifiers

Step 3: Combine Classifiers



Why does it work?

- ✗SUPPOSE THERE ARE 25 BASE CLASSIFIERS
  - ✗Each classifier has error rate, $\varepsilon = 0.35$
  - ✗Assume classifiers are independent
  - ✗Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=1}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

- ✗How to generate an ensemble of classifiers?
  - ✗Bagging: Constructs classifiers independently
  - ✗Boosting: Constructs ensemble incrementally

Bagging

- ✗Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- ✗Build classifier on each bootstrap sample
- ✗Each sample has probability $(1-1/n)^n$ of being selected

Boosting

- ✗An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - ✗Each instance is assigned a weight that measures its probability of being sampled
  - ✗Initially, all N records are assigned equal weights
  - ✗Unlike bagging, weights change at the end of boosting round

Boosting

- ✗Records that are wrongly classified will have their weights increased
- ✗Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Example: AdaBoost

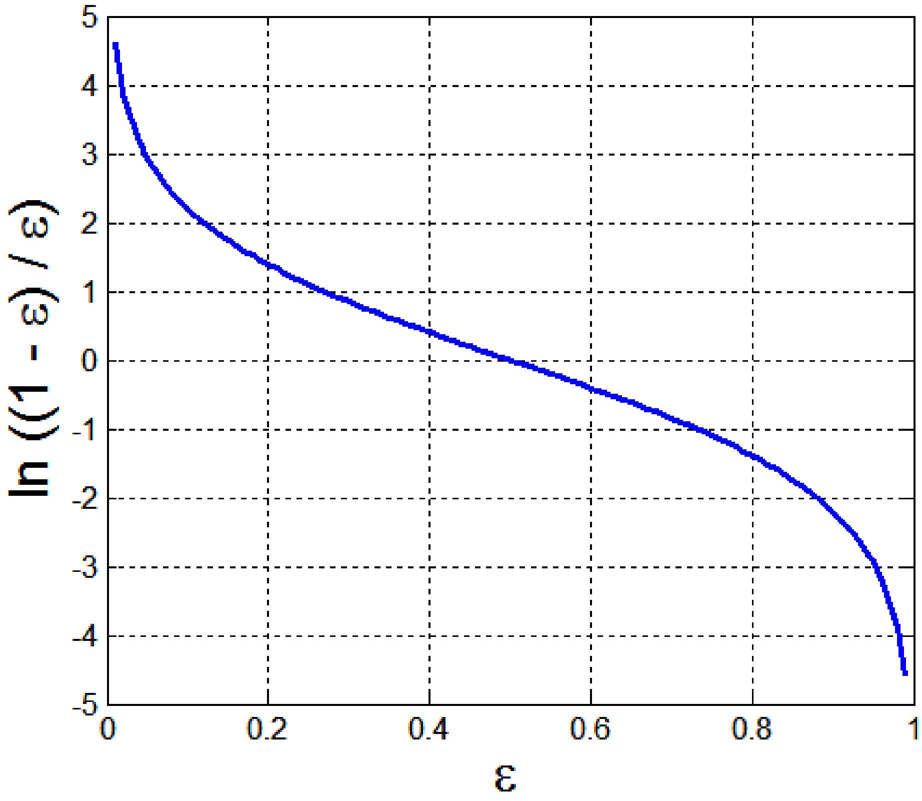
- ✗Base classifiers: $C_1, C_2, \ldots, C_T$

- ✗Error rate:

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- ✗Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Example: AdaBoost

- ✗Weight update:

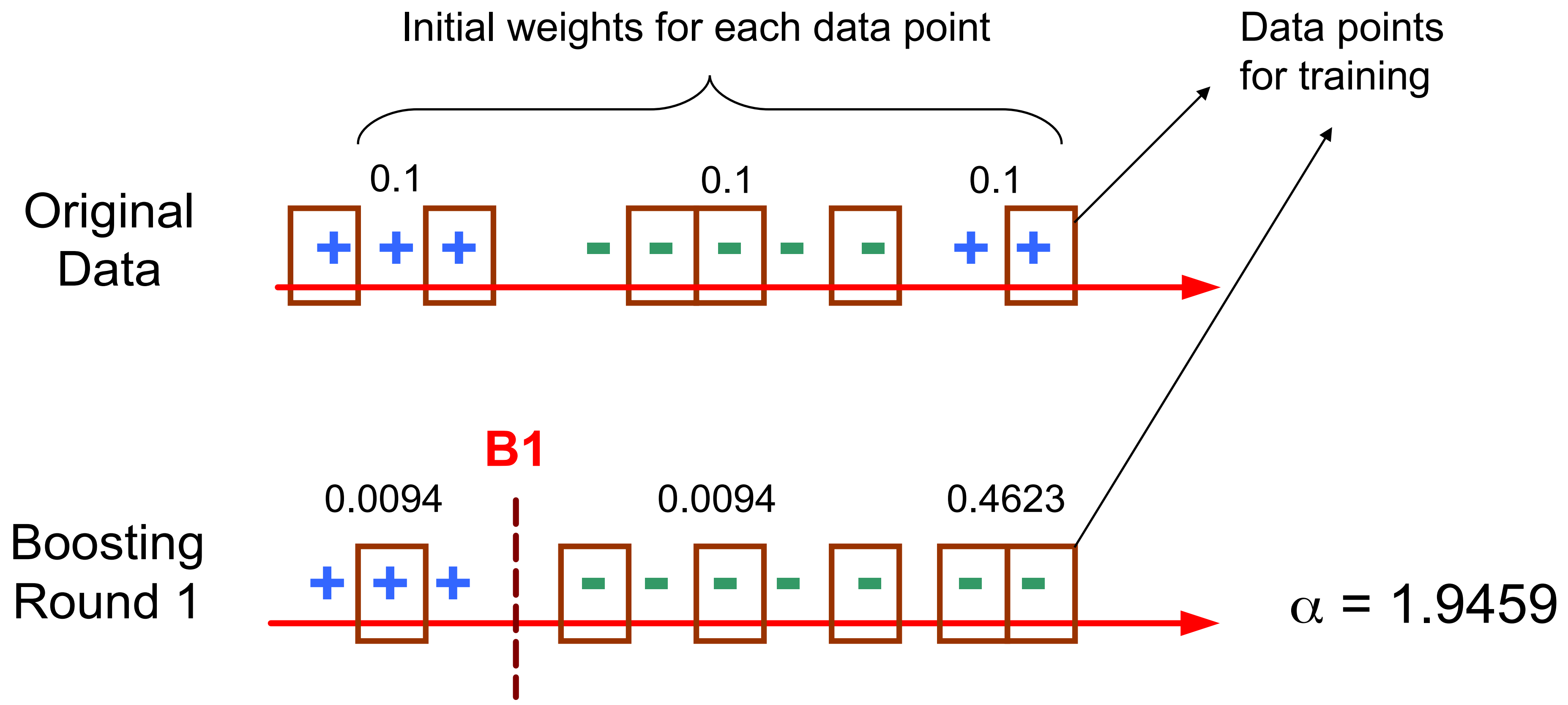
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

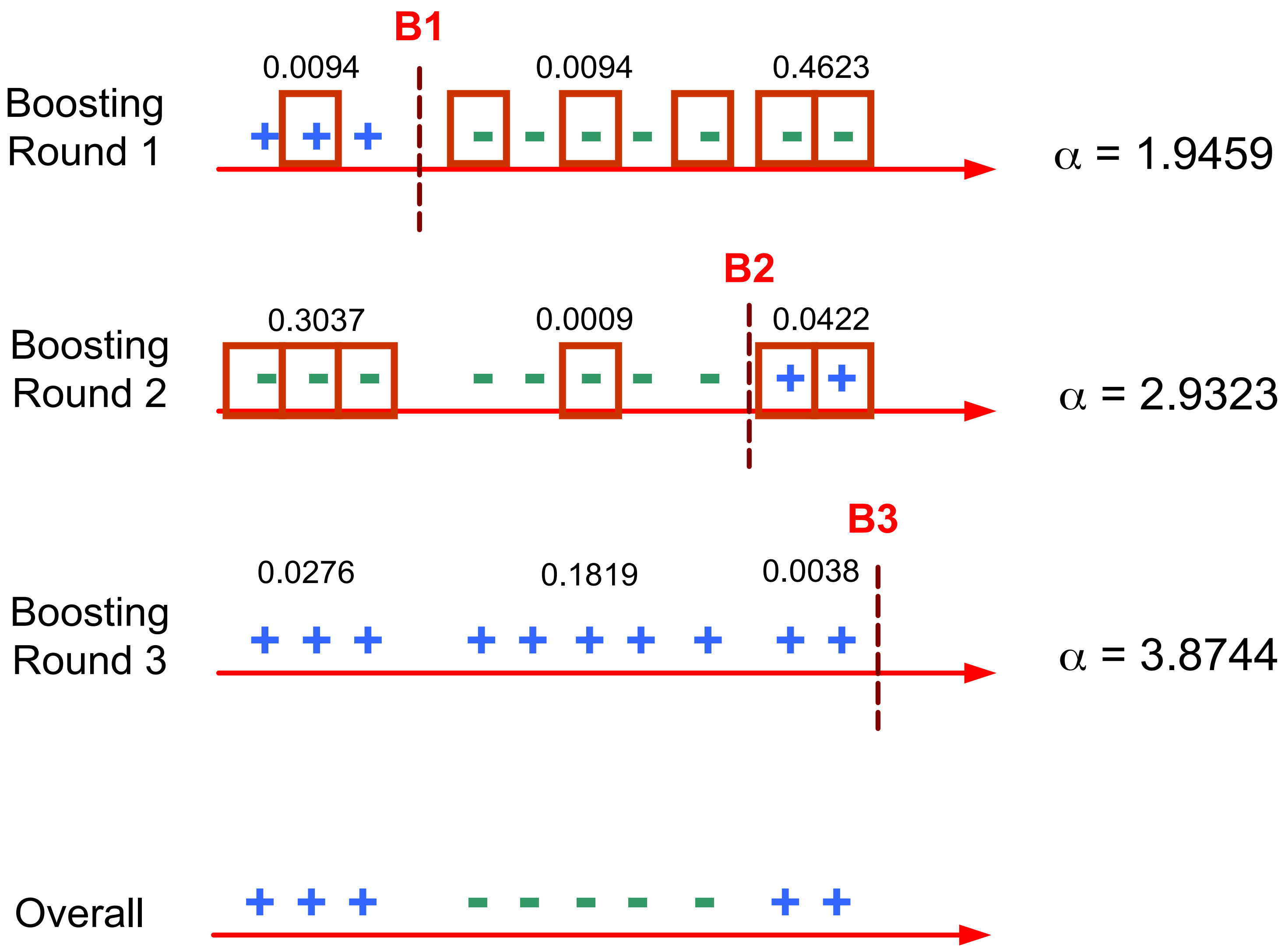
- ✗If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- ✗Classification:

$$C^*(x) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

Illustrating AdaBoost



Illustrating AdaBoost



Bias/variance of error

- ✗Error is usually divided into three parts
  - ✗Irreducible error
  - ✗Bias error: Error of the central tendency of the model
  - ✗Variance error: Error of the deviation from the central tendency

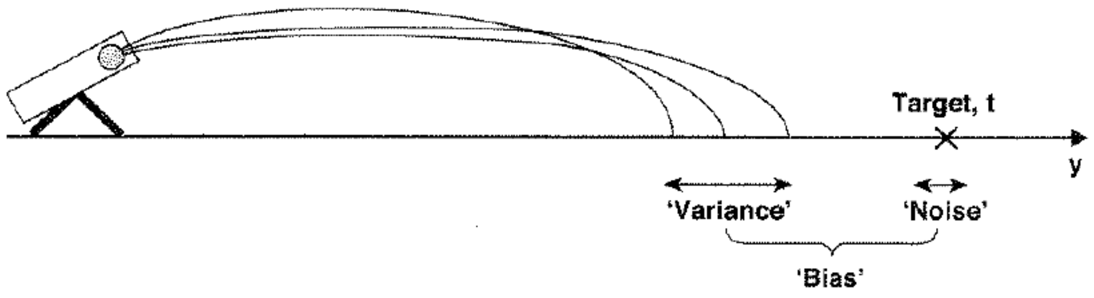


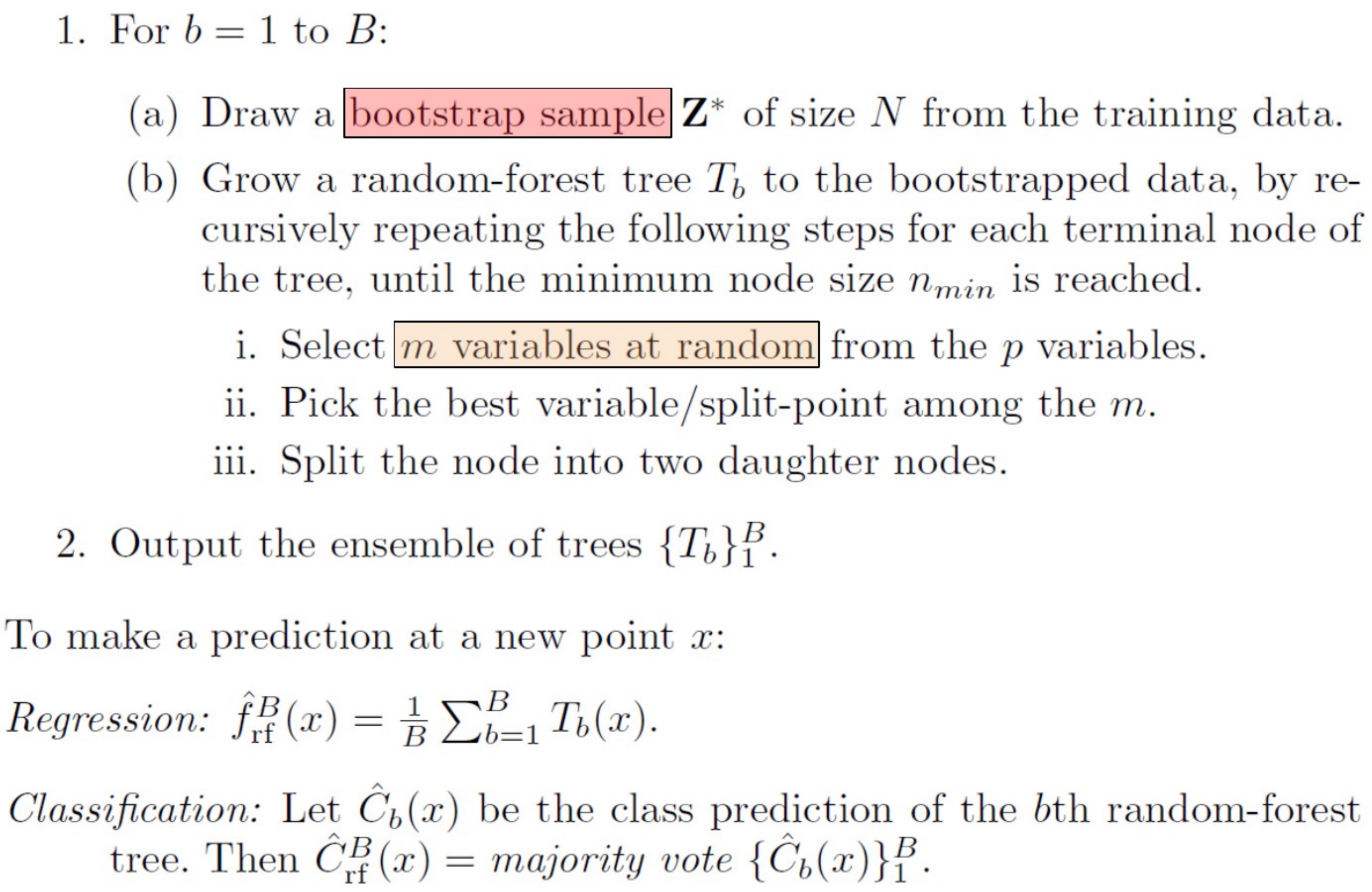
Figure 5.32. Bias-variance decomposition.

  - ✗Bagging: Reduces variance of the error
  - ✗Boosting: Reduces both bias and variance of the error

Random forest

- ✗Ensemble specifically designed for decision trees
- ✗Introduces randomness on each node calculation
  - ✗Boostrap sampling
  - ✗Random sampling of features on each node

Random forest algorithm



```
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
        i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

To make a prediction at a new point x:

Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.