Classification: Alternative Techniques

Tom Claassen

Radboud University Nijmegen

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 condition
  - *RHS*: rule consequent
- Examples of classification rules:
  - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
  - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Evade=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 a 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) $\wedge$ (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 => Bird

The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

- Coverage of a rule:
  - Fraction (relative to the total number) of records that satisfy the antecedent of a rule

- Accuracy of a rule:
  - Fraction (relative to those satisfying the antecedent) of records that satisfy both the antecedent and consequent of a rule

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

(Status=Single) → No

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

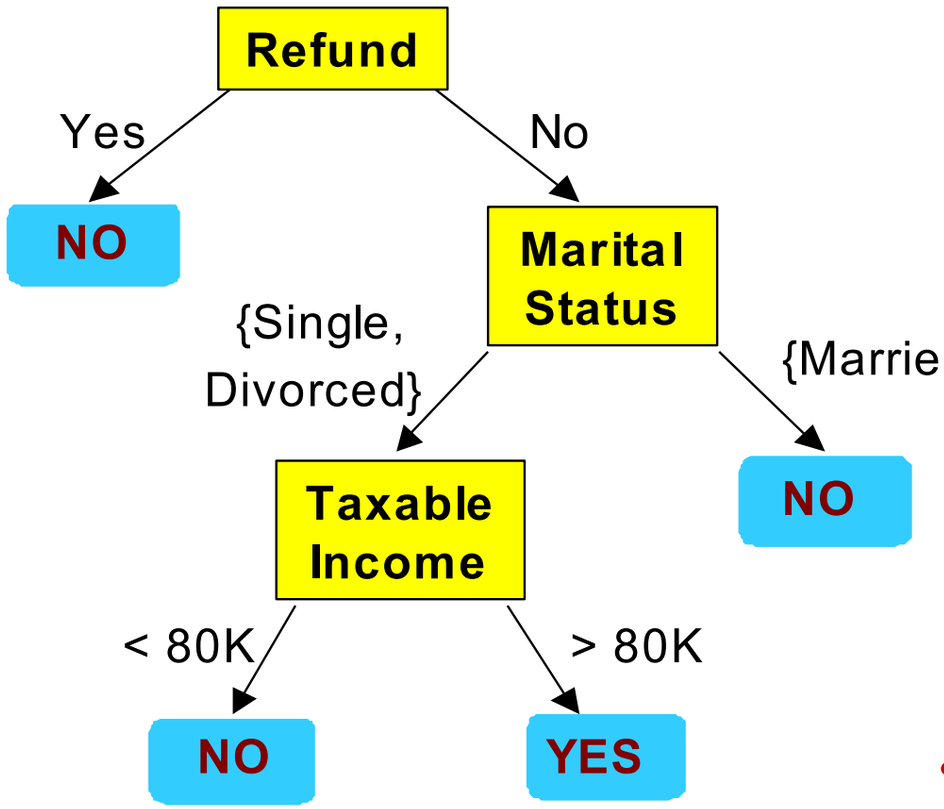
A turtle triggers both R4 and R5

A dogfish shark triggers none of the rules

Characteristics of Rule-based Classifiers

- 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

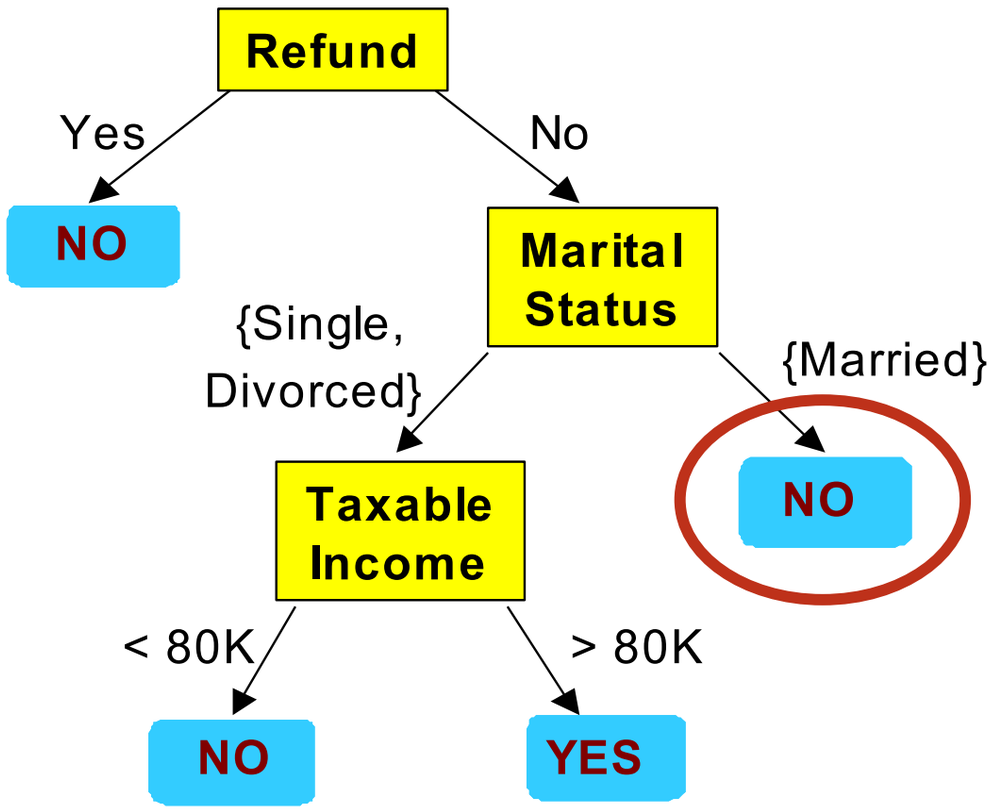
(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(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

From Decision Trees To Rules



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

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

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

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

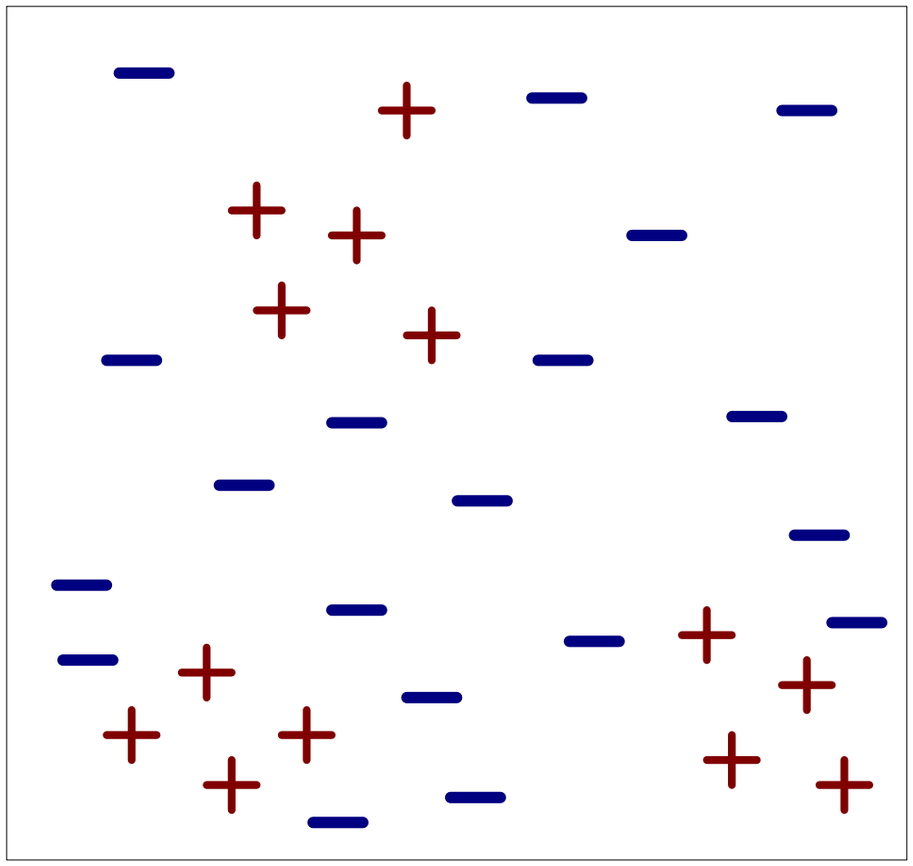
Building Classification Rules

- 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,).
  - e.g: C4.5rules

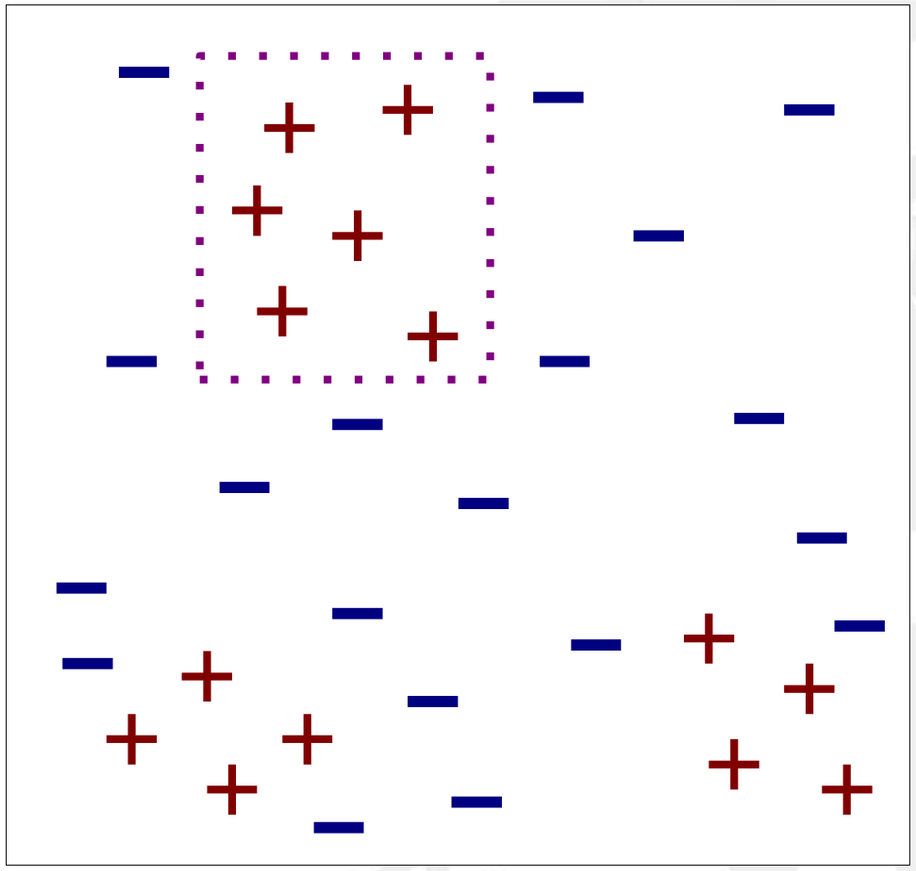
Direct Method: Sequential Covering

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

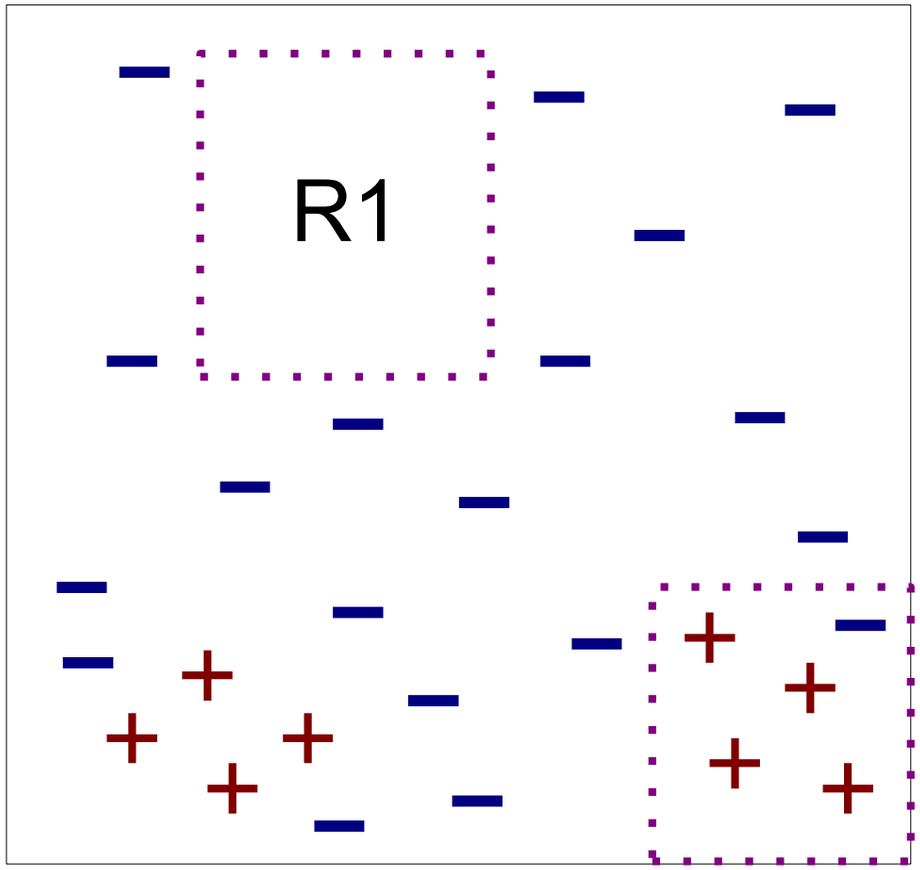


(i) Original Data

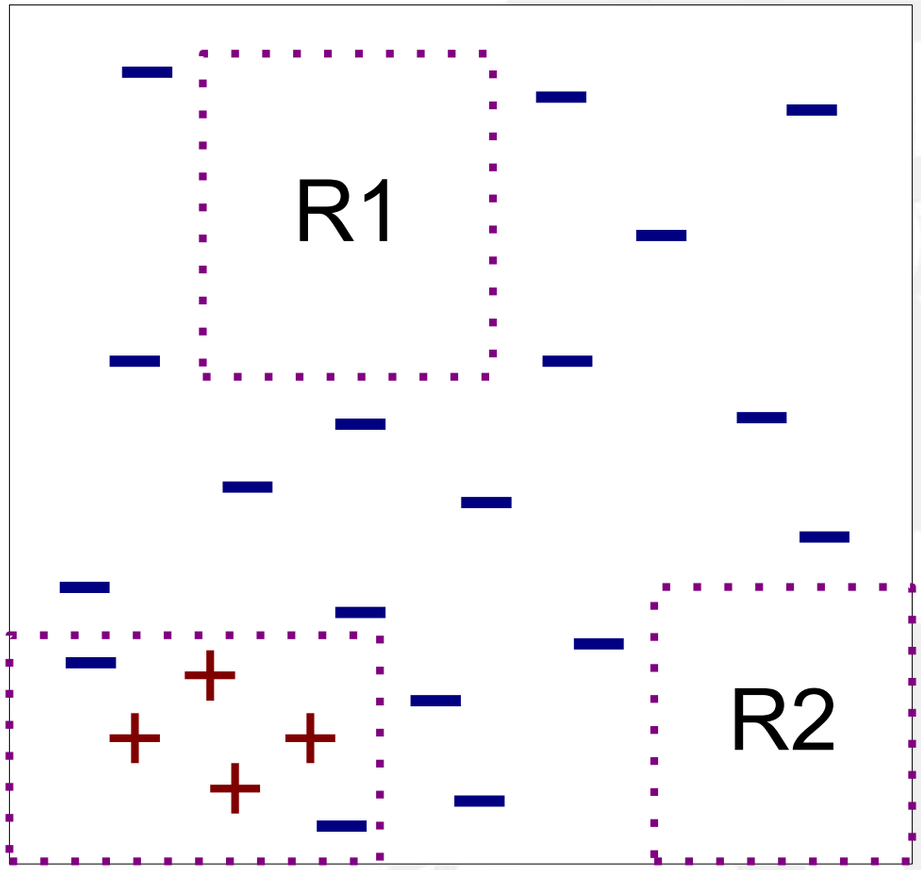


(ii) Step 1

Example of Sequential Covering…



(iii) Step 2



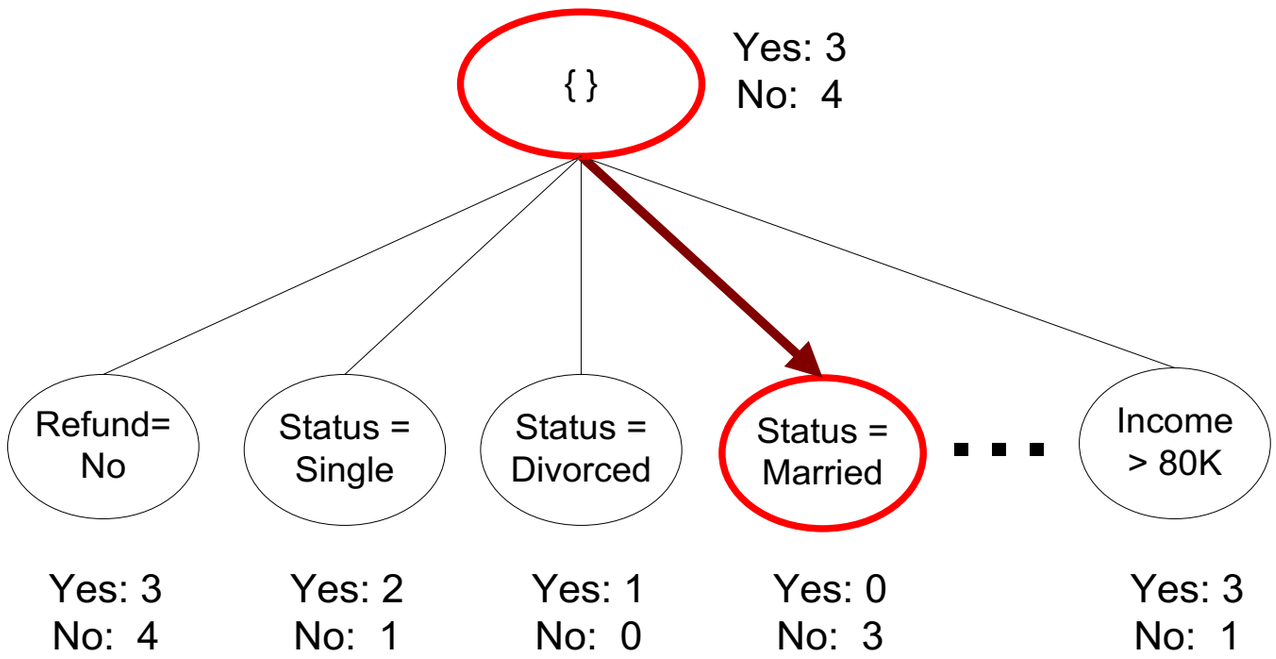
(iv) Step 3

Aspects of Sequential Covering

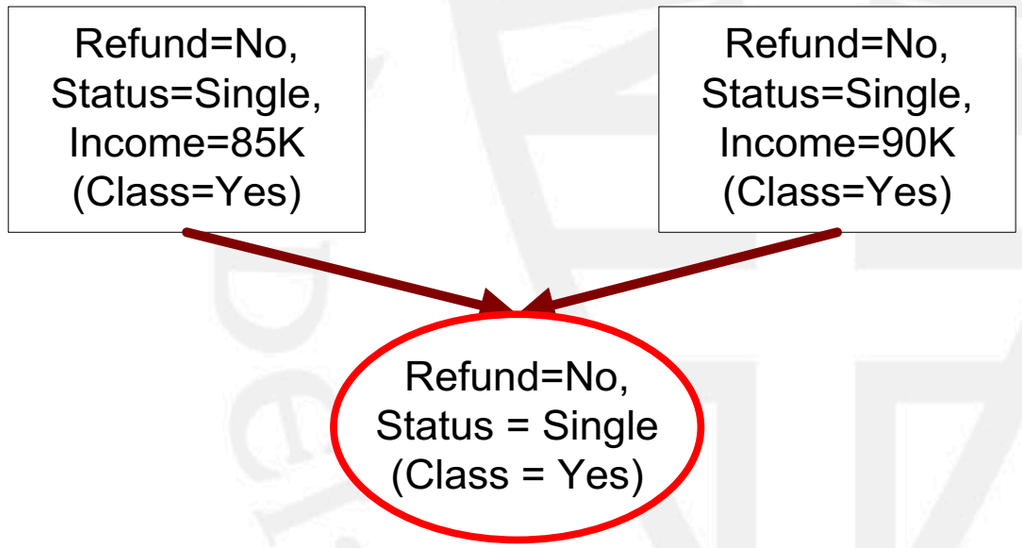
- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

Rule Growing: CN2

- Start from an empty conjunct: {}
- Greedily add conjuncts that minimize some entropy measure: {A}, {A,B}, …
- Determine the rule consequent by taking majority class of instances covered by the rule

Rule Growing: RIPPER

- Start from an empty rule: {} => class

- Add conjuncts that maximizes FOIL's information gain measure:
  - R0: {A} => class (initial rule)
  - R1: {A Λ B} => class (rule after adding conjunct: more specific)

  - **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
    (note: p1 ≤ p0 and n1 ≤ n0)

Rule Evaluation Metrics

- 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 instances covered by rule

k : Number of classes

p : Prior probability for positive class

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

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 conjunct if without it v is higher on a validation set

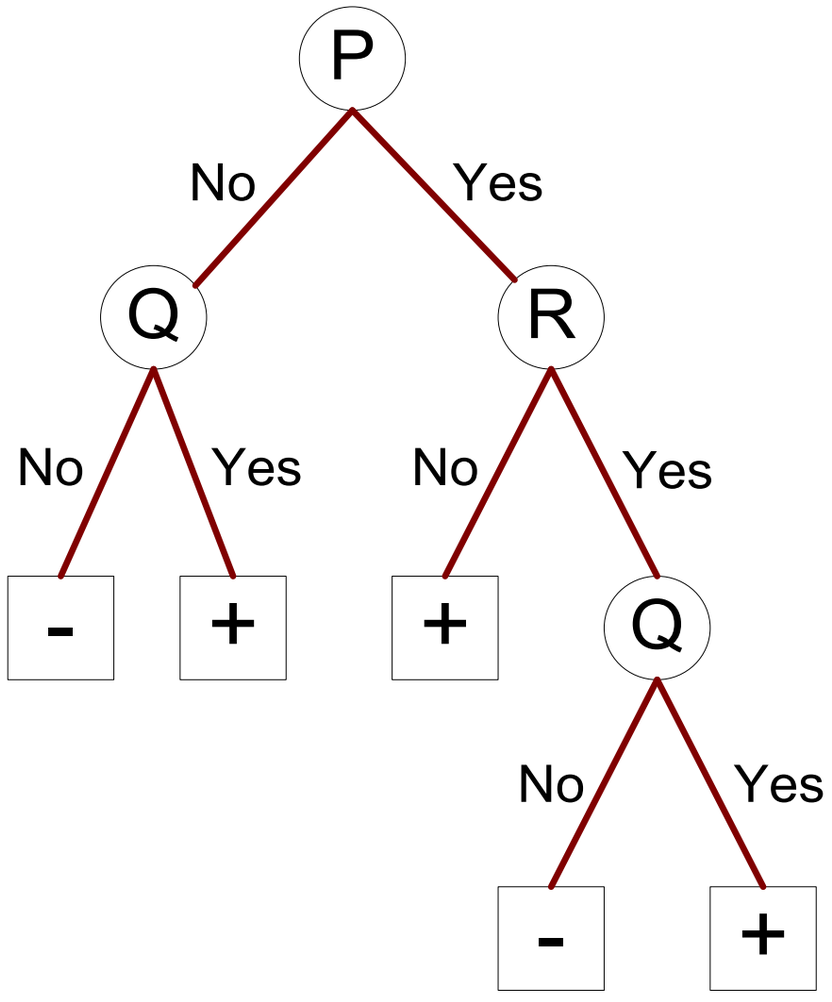
RIPPER: Building a Rule Set

- 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 (default: 64) longer than the smallest description length obtained so far

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 the description length
- 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) ==> +

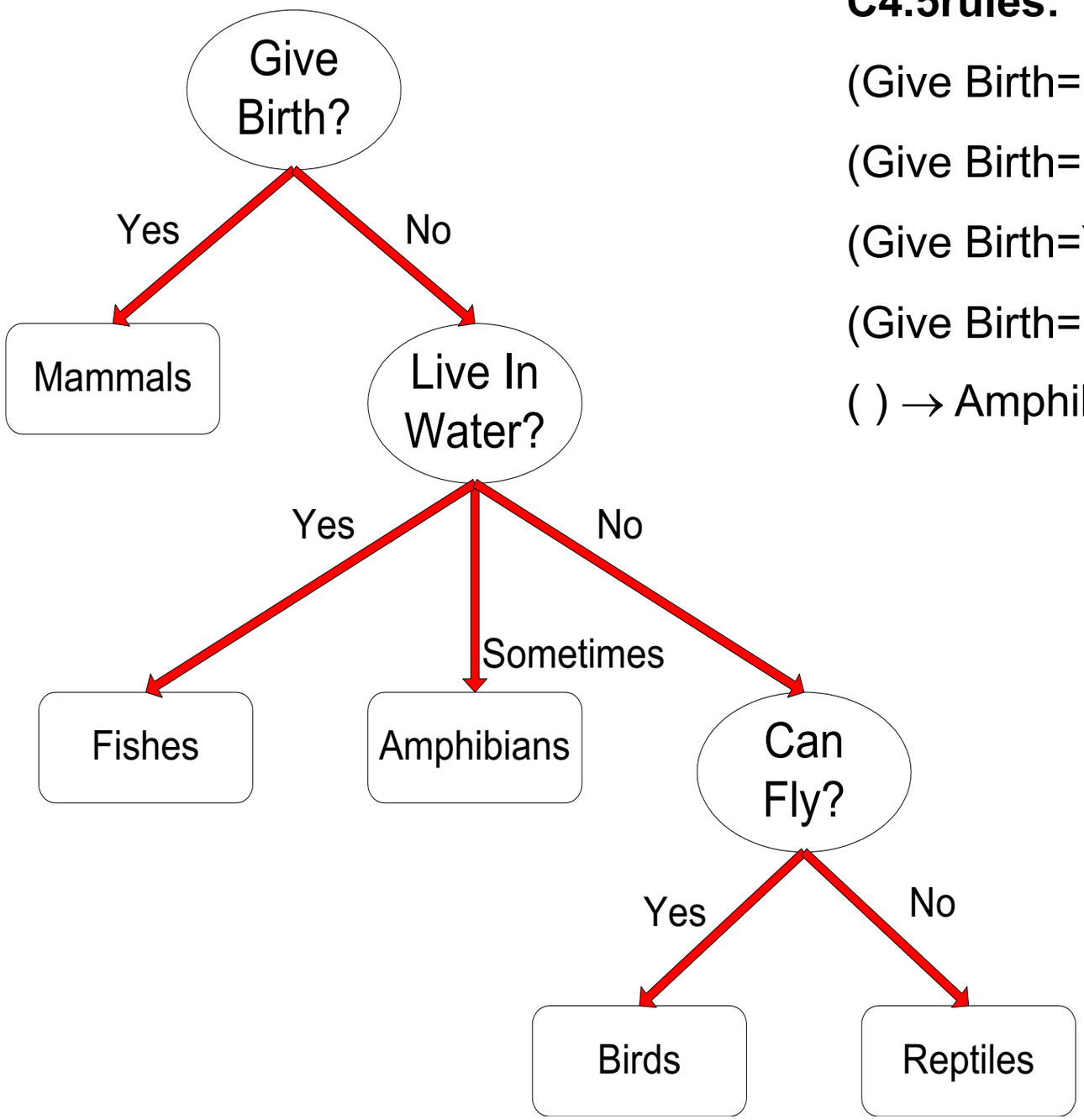
Indirect Method: C4.5rules

- 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
- Use class-based ordering: order subsets of rules that correspond to the same class

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**	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

RIPPER:

			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

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

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

Set of Stored Cases

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

Unseen Case

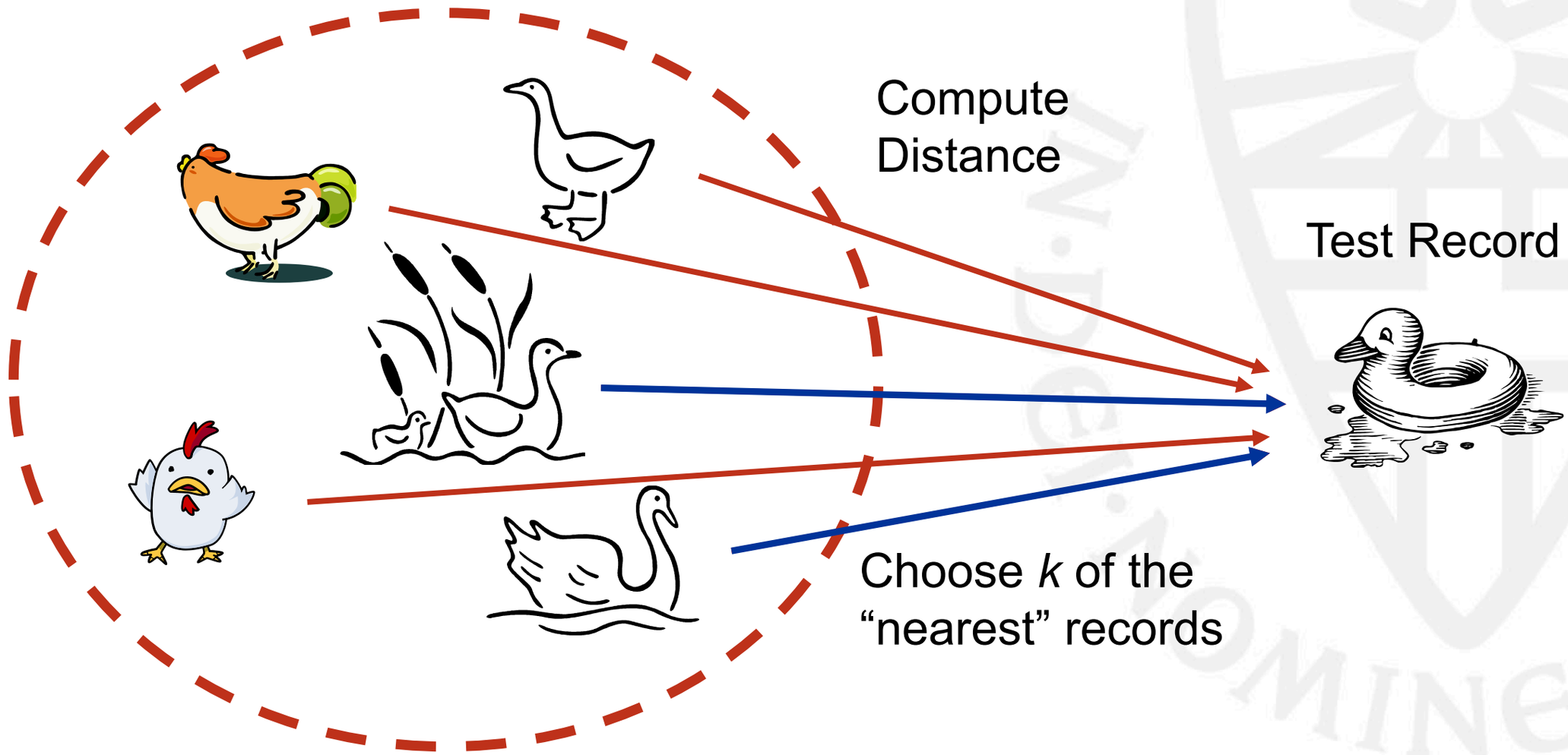
Atr1	…….	AtrN

Examples of Instance Based Classifiers

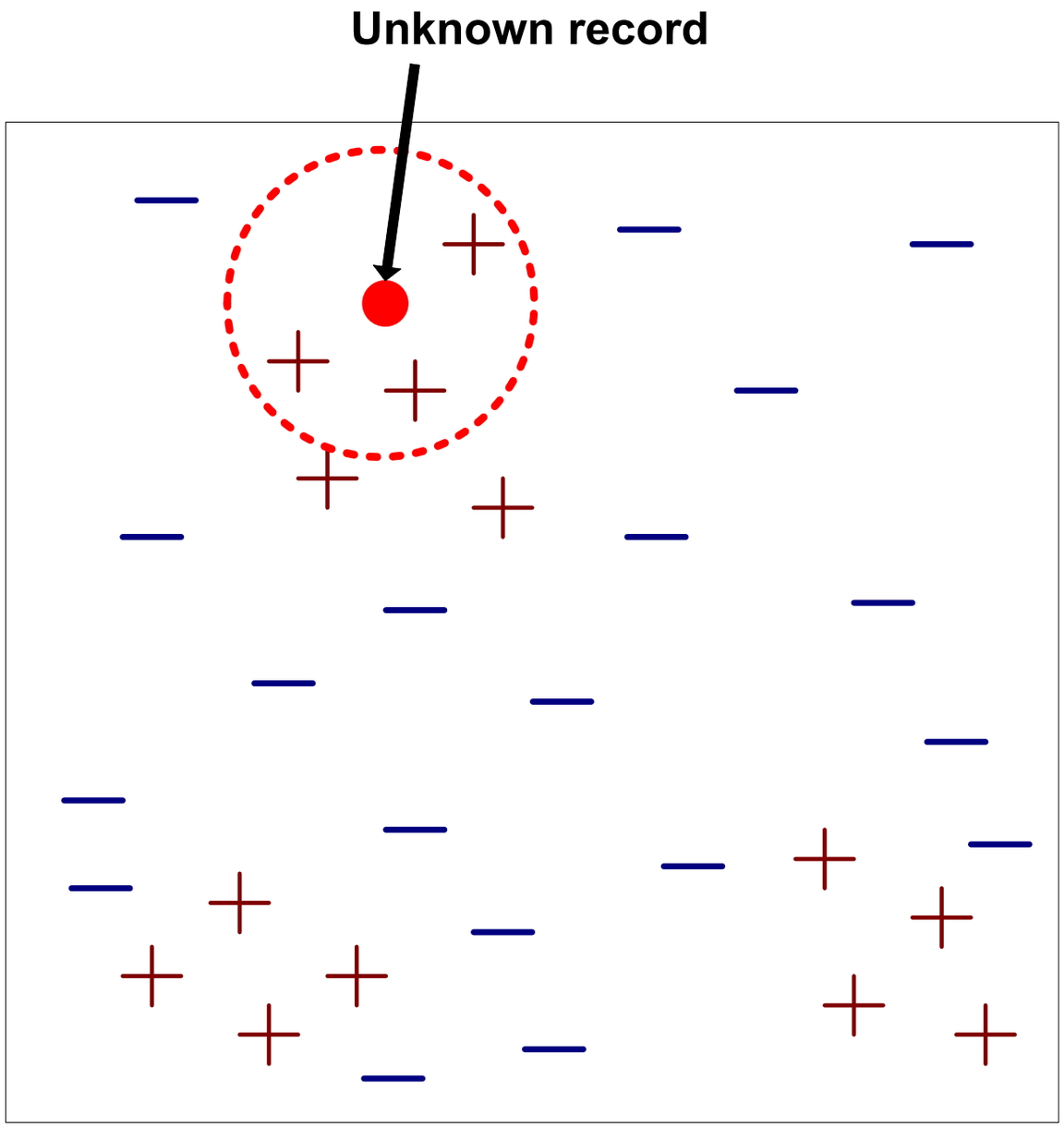
- Rote-learner
  - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
- Nearest neighbor
  - Uses *k* “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

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



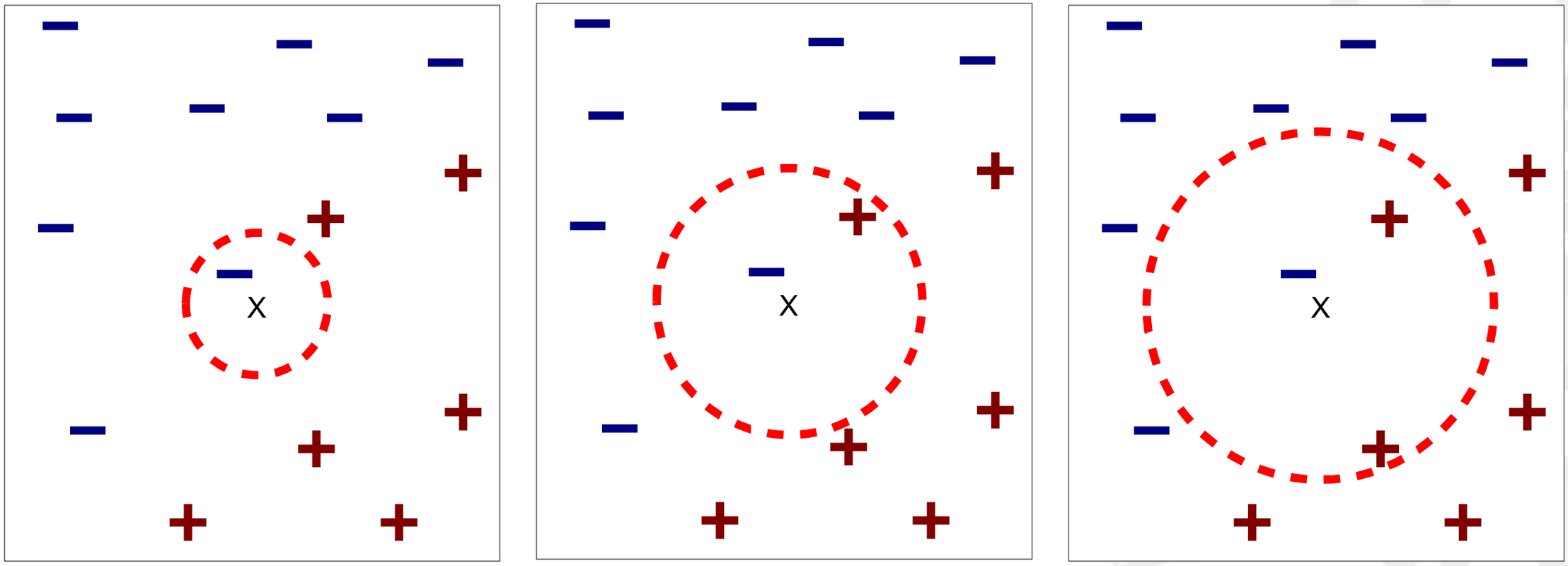
Nearest-Neighbor Classifiers



- Requires:
  1. The set of stored records
  2. A distance metric to compute distance between records
  3. 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)

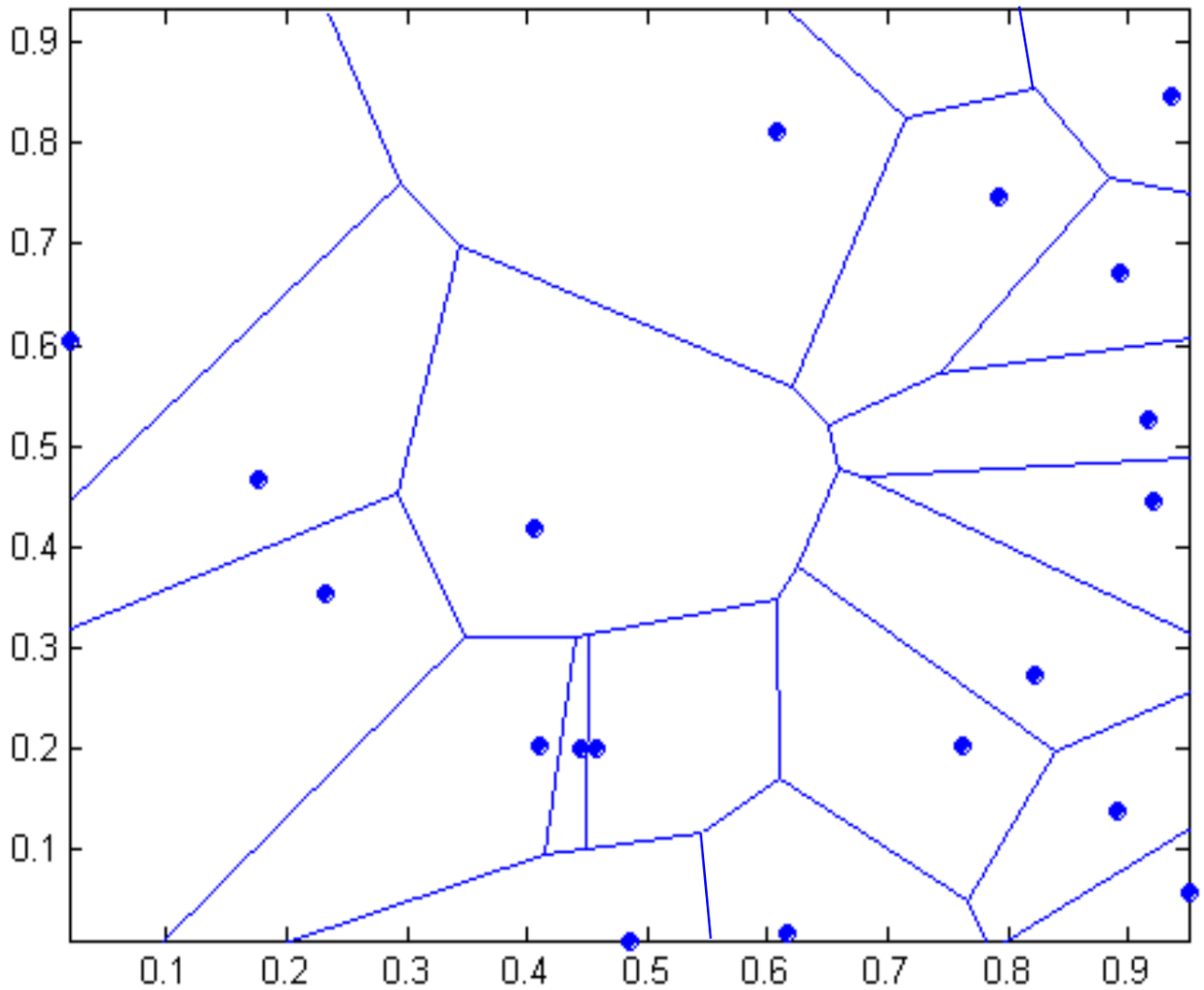
Definition of Nearest Neighbor



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

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

1 nearest-neighbor



Voronoi Diagram

Nearest Neighbor Classification

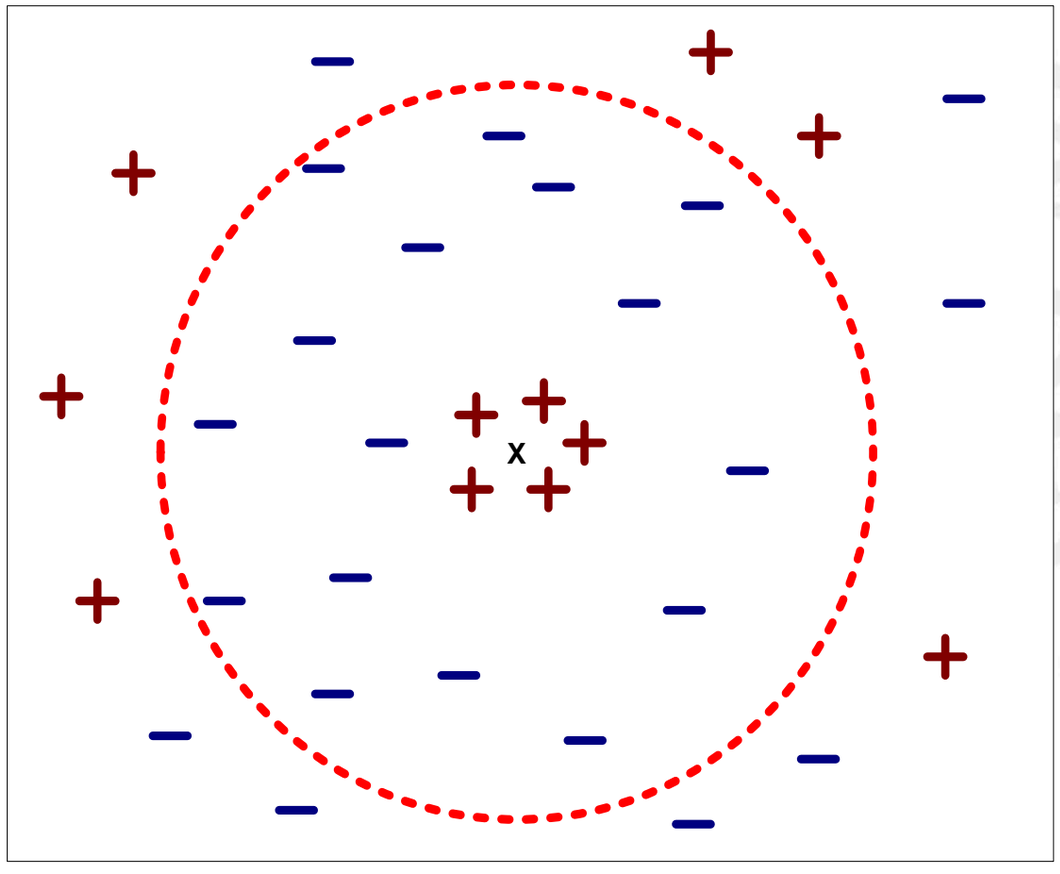
- Compute distance between two points, e.g., using Euclidean distance:

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

- 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, e.g., with weight factor, $w = 1/d^2$

Nearest Neighbor Classification: Choice 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

- Can also work with other distance measures

Nearest neighbor Classification: Lazy Learning

- “Lazy learners” do not build models explicitly
- To be contrasted with “eager learners” such as decision tree induction, rule-based systems, neural networks, …
- Classifying unknown records is relatively expensive

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional probability:

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

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

- Bayes theorem:

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

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - People without meningitis have a stiff neck with probability 1/20
  - Prior probability of any patient having meningitis is 1/50,000

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

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)}$$

$$P(S) = P(S \mid M)P(M) + P(S \mid \overline{M})P(\overline{M}) = 0.5 \times 1/50000 + 0.05 \times (1 - 1/50000) = 0.050009$$

$$P(M \mid S) = \frac{0.5 \times 1/50000}{0.050009} \approx 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

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

- Can we estimate $P(C| A_1, A_2,\ldots,A_n)$ directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using Bayes theorem

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

  - Choose the value of C that maximizes $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes $P(A_1, A_2, \ldots, A_n|C)\ P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:

$$P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C)\, P(A_2 \mid C) \ldots P(A_n \mid C)$$

  - Can estimate $P(A_i \mid C)$ for all A_i and C from training data.
  - New point is classified to $C=j$ if $P(A_1, A_2, \ldots, A_n \mid C=j)\, P(C=j)$ is maximal.

How to Estimate Probabilities from Data: Discrete Data

Tid	Home Owner	Marital Status	Taxable Income	Default
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

Loan classification data

- Class: $P(C=k) = N_k / N$
  - e.g., $P(\text{No}) = 7/10$, $P(\text{Yes}) = 3/10$
- For discrete attributes:

$$P(A_i=a \mid C=k) = |A_i=a;\ C=k| \ / \ N_k$$

  - where $|A_i=a;\ C=k|$ is the number of instances having attribute $A_i=a$ and belonging to class k
  - Examples:

P(Status=Married|No) = 4/7
P(Home Owner=Yes|Yes) = 0

How to Estimate Probabilities from Data: Continuous Data

- Discretize the range into bins
  - One ordinal attribute per bin
  - May violate independence assumption...
- Two-way split: ($A < v$) or ($A > v$)
  - Choose one of the two splits as new attribute
- Probability density estimation:
  - Assume attribute follows , for example, a normal distribution
  - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
  - Once probability distribution is known, use it to estimate the conditional probability $P(A_i|C)$

How to Estimate Probabilities from Data?

Tid	Home Owner	Marital Status	Taxable Income	Default
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

Loan classification data

- Normal distribution:

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

  - One for each $(A_i, C=j)$ pair

- For (Income, Home Owner=No):

  - If Home Owner =No:
    - sample mean = 82
    - sample variance = 207

$$P(\text{Income} = 120 \mid \text{No}) = \frac{1}{\sqrt{2\pi \times 207}} e^{-\frac{(120-82)^2}{2(207)}}$$

$$= 0.000872$$

Example of Naïve Bayes Classifier

- Given a Test Record: X = (HomeOwner = No, Married, Income = 120K)

naive Bayes Classifier:

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

- P(X | Class=No) = P(Owner=No | Class=No)
  × P(Married | Class=No)
  × P(Income=120K | Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Owner=No | Class=Yes)
  × P(Married | Class=Yes)
  × P(Income=120K | Class=Yes)
  = 1 × 0 × 1.2e-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: Handling Zeroes

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

$$\text{Original}: P(A_i = a \mid C = j) = \frac{|A_i = a; C = j|}{N_j}$$

$$\text{Laplace}: P(A_i = a \mid C = j) = \frac{|A_i = a; C = j| + 1}{N_j + c}$$

$$\text{M-estimate}: P(A_i = a \mid C = j) = \frac{|A_i = a; C = j| + mp_a}{N_j + m}$$

c: number of options *a*

p_a: prior probability of *a*

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

A: all attributes together

M: mammals

N: non-mammals

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

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

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

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

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

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

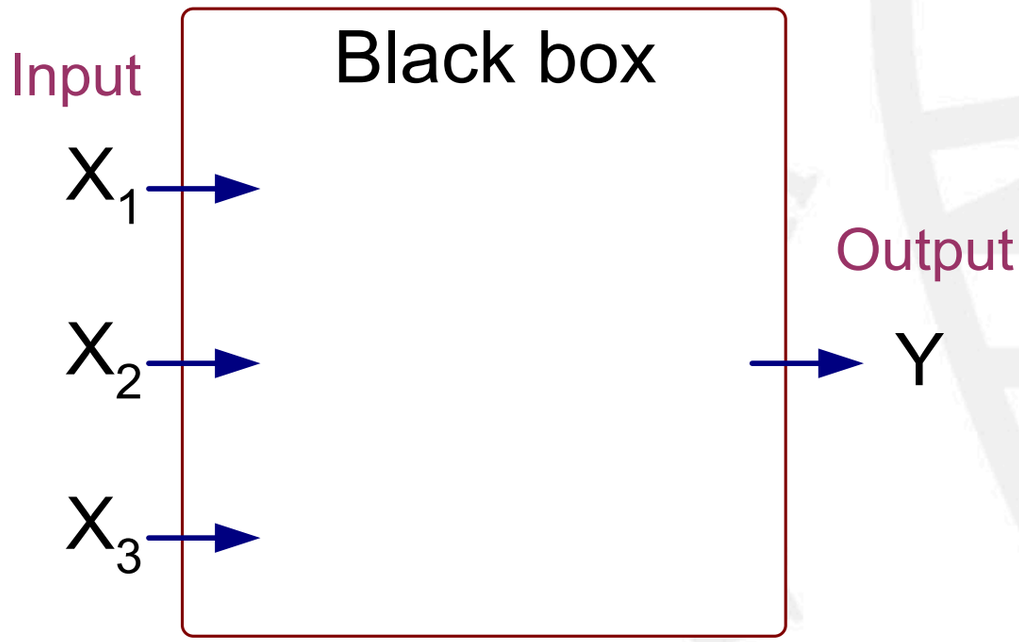
=> Mammals

Naïve Bayes Summary

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Conditional independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian networks to model the probability distribution of the attributes given the class

Artificial Neural Networks (ANN)

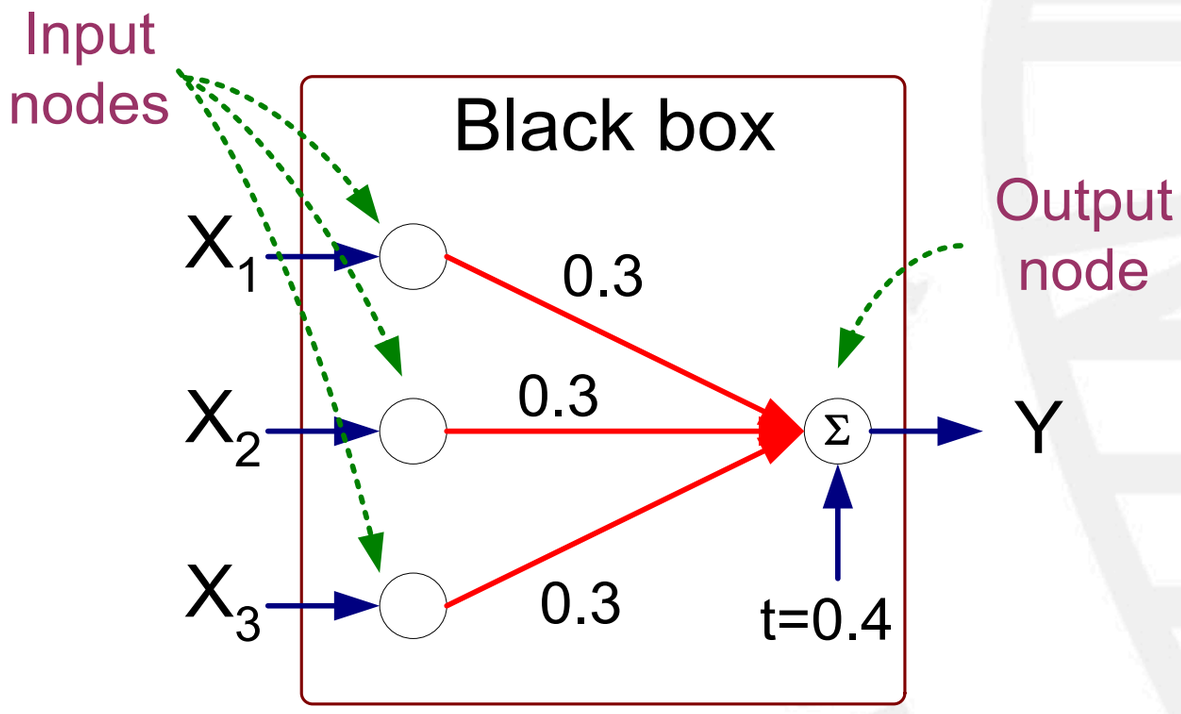
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 iff 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

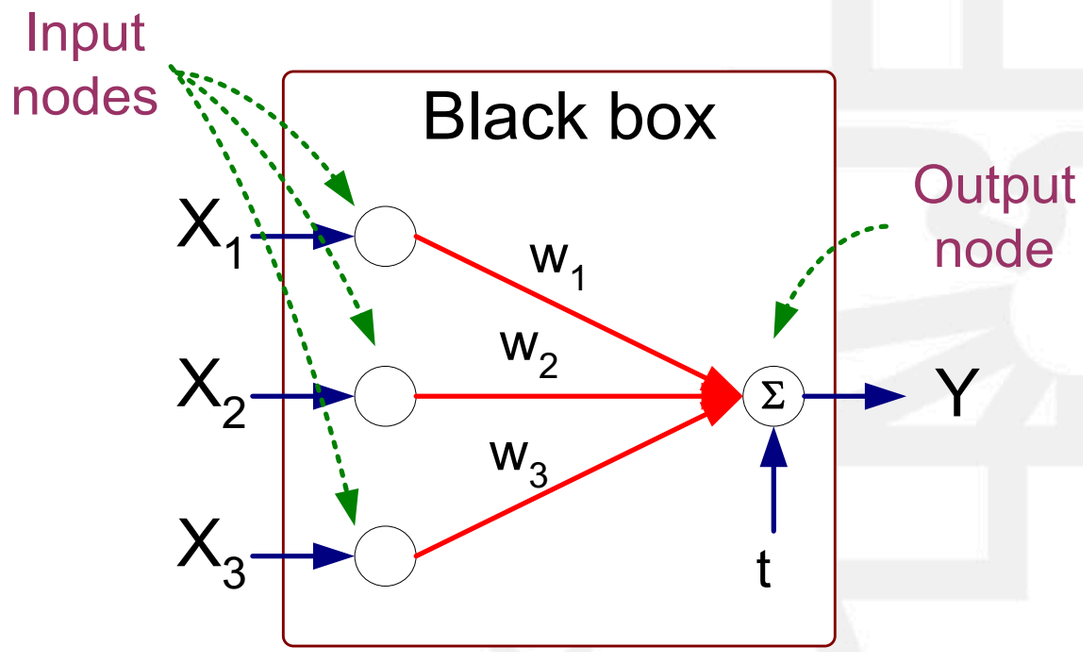


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

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*

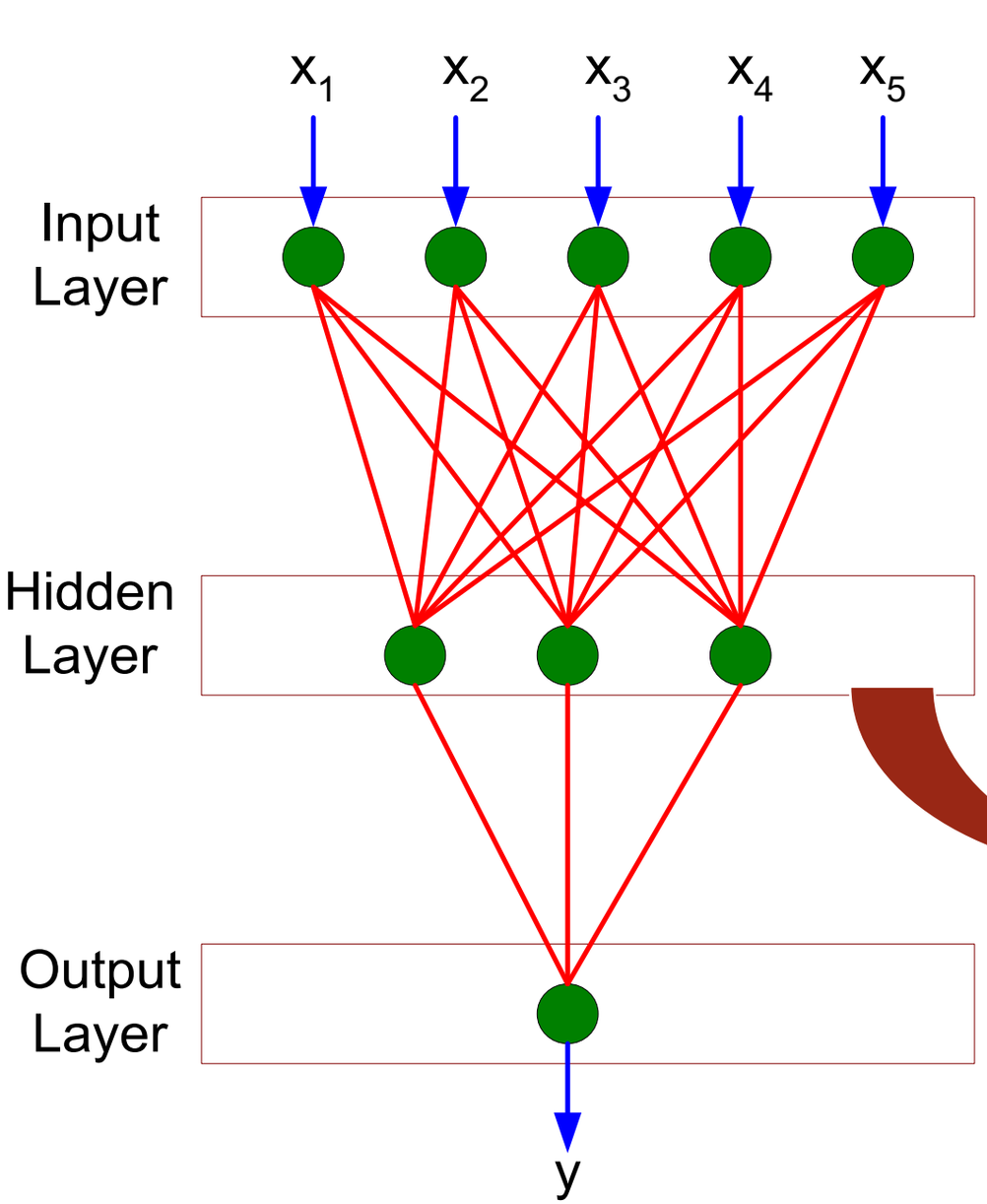


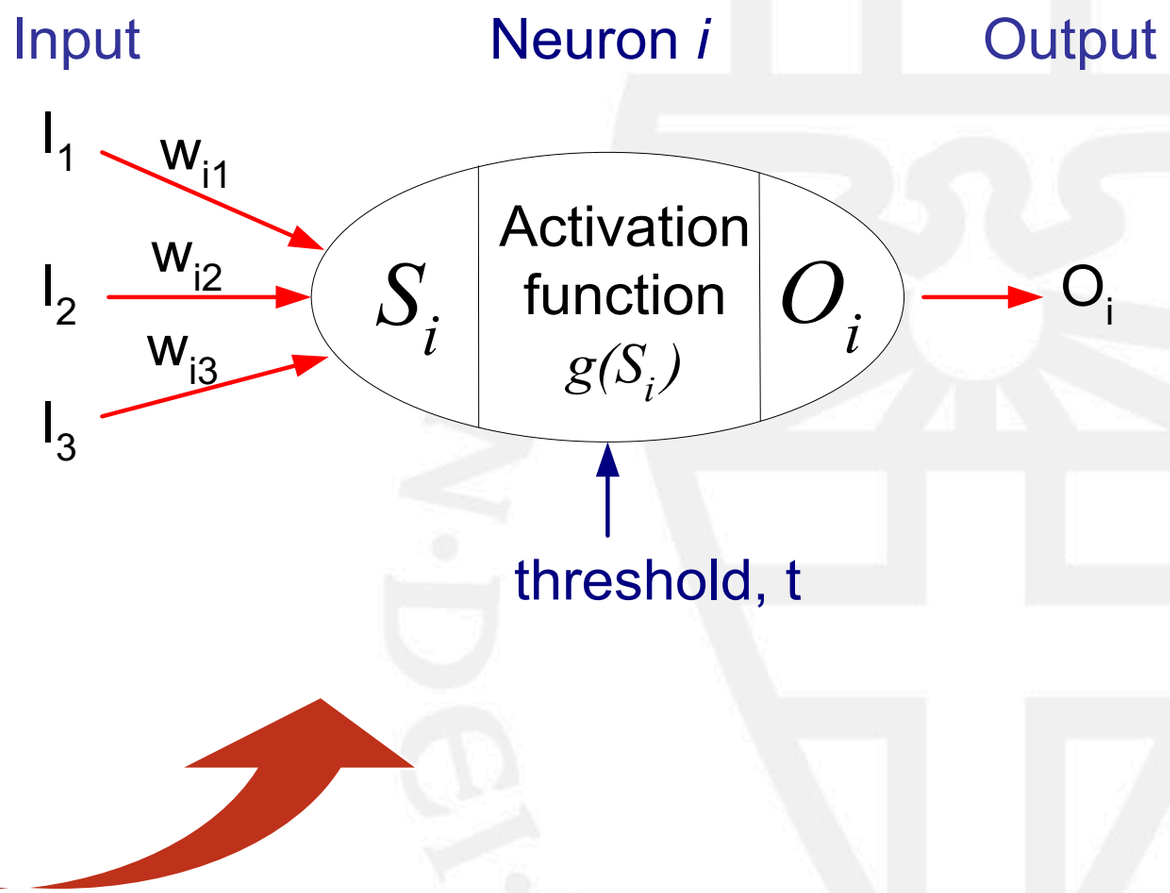
$$Y = I(\sum_i w_i X_i - t)$$

or

$$Y = \text{sign}(\sum_i w_i X_i - t)$$

General Structure of ANN



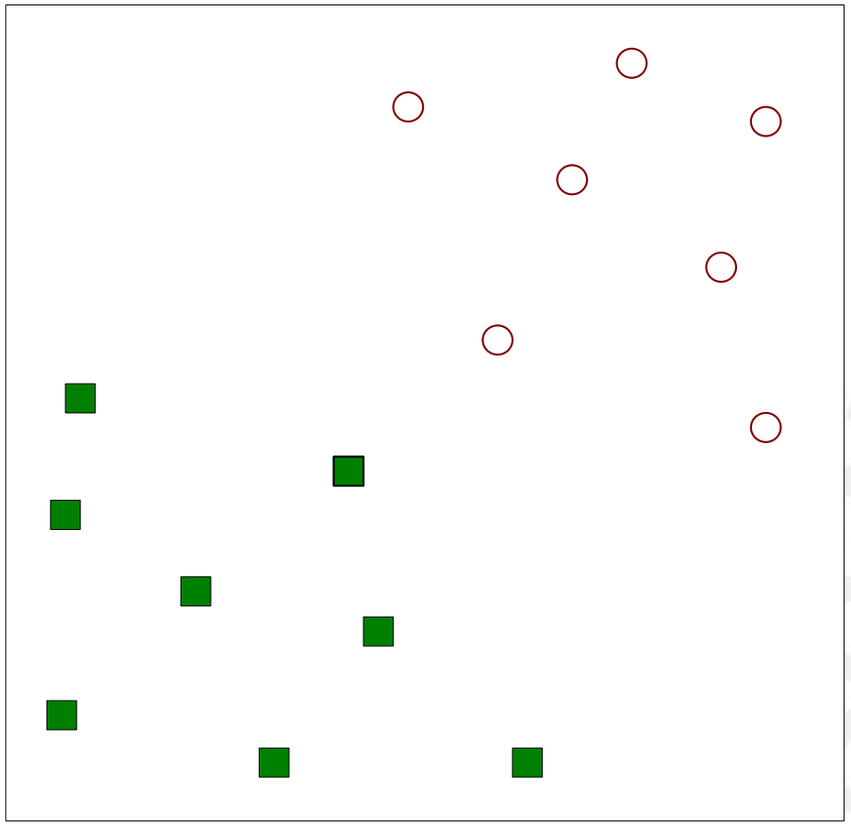


Training ANN means changing the weights

Algorithm for learning ANN

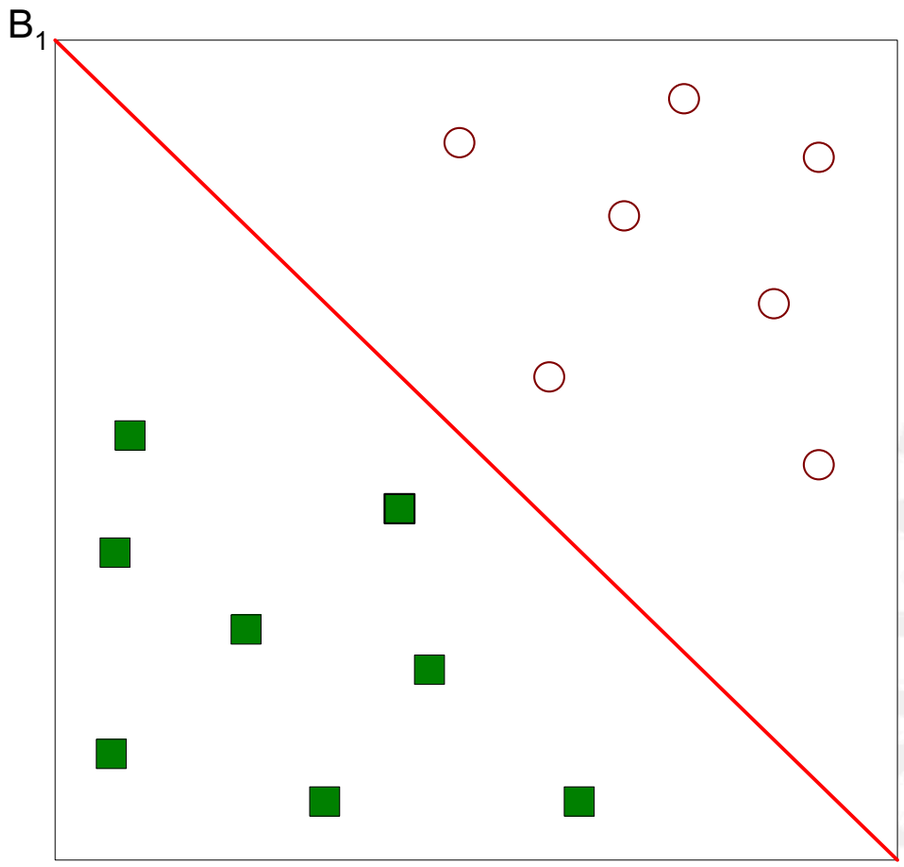
- Initialize the weights (w_0, w_1, …, w_k)
- Adjust the weights in such a way that the output of ANN is consistent with the class labels of the training examples
  - Objective function: $E = \sum_i \left[Y_i - f((w_0, \cdots w_k), X_i) \right]^2$
  - Find the weights that minimize the above objective function e.g., using the so-called backpropagation algorithm
- Revival of “deep learning”: clever ways to initialize the weights in combination with lots of data to learn huge neural networks

Support Vector Machines



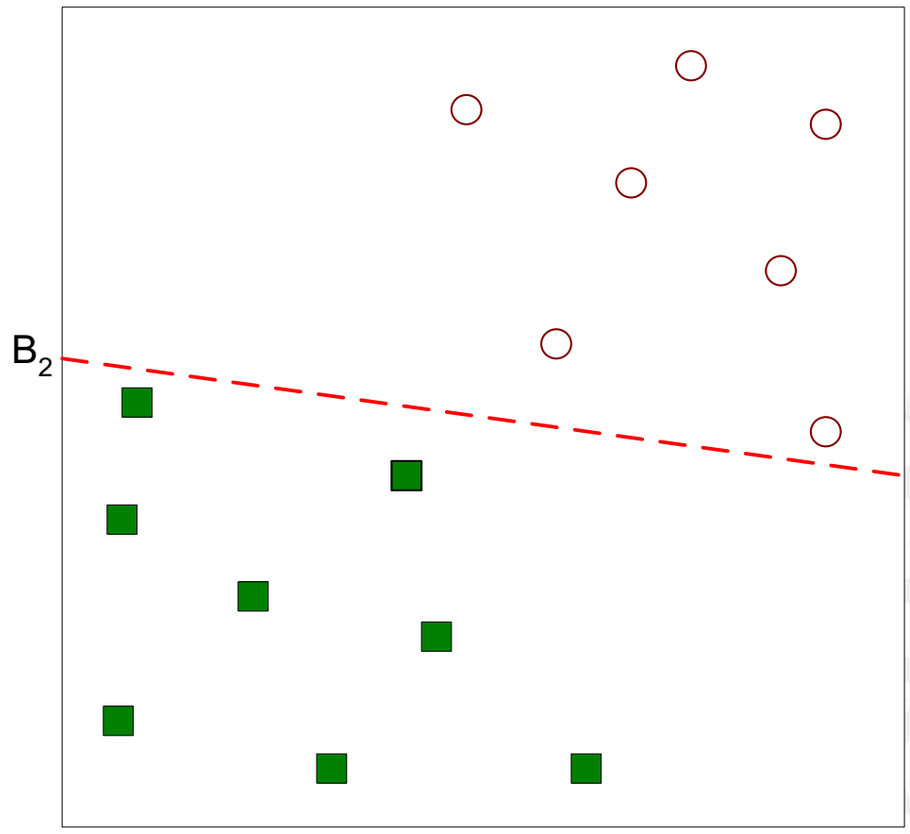
Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



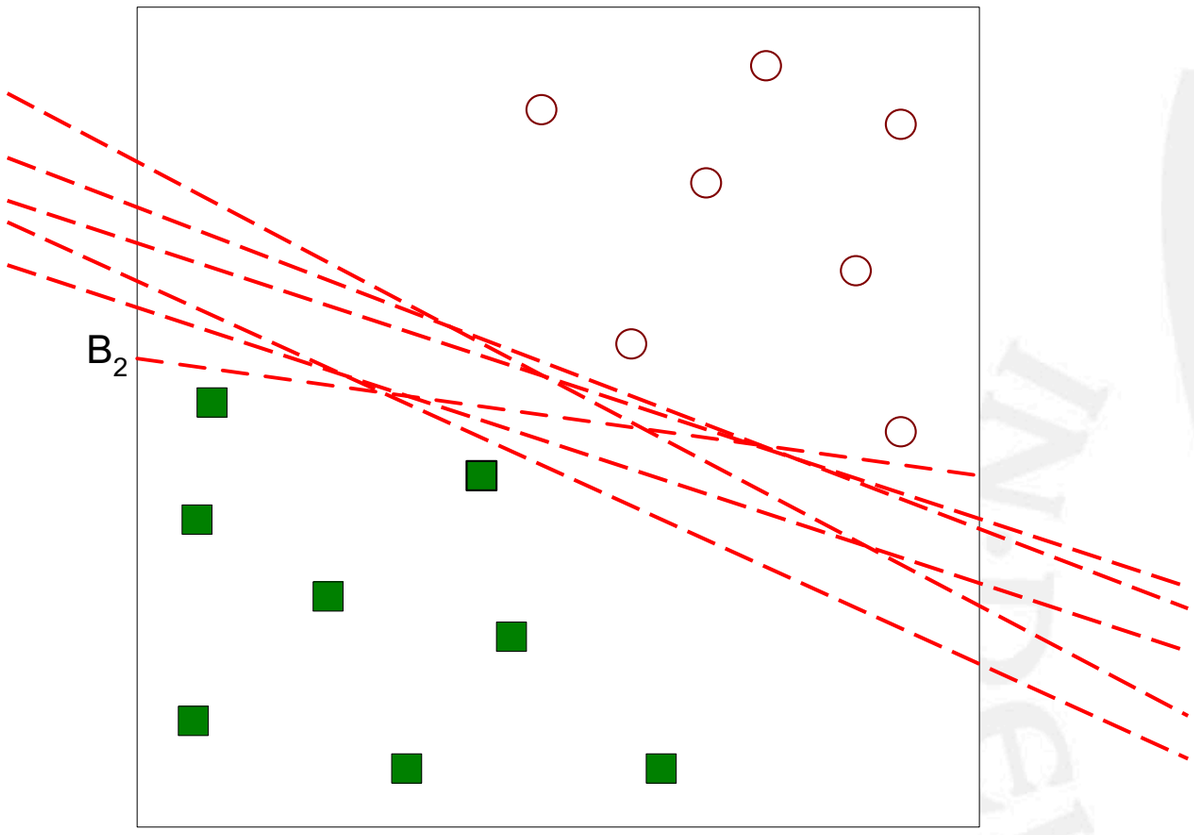
One possible solution

Support Vector Machines



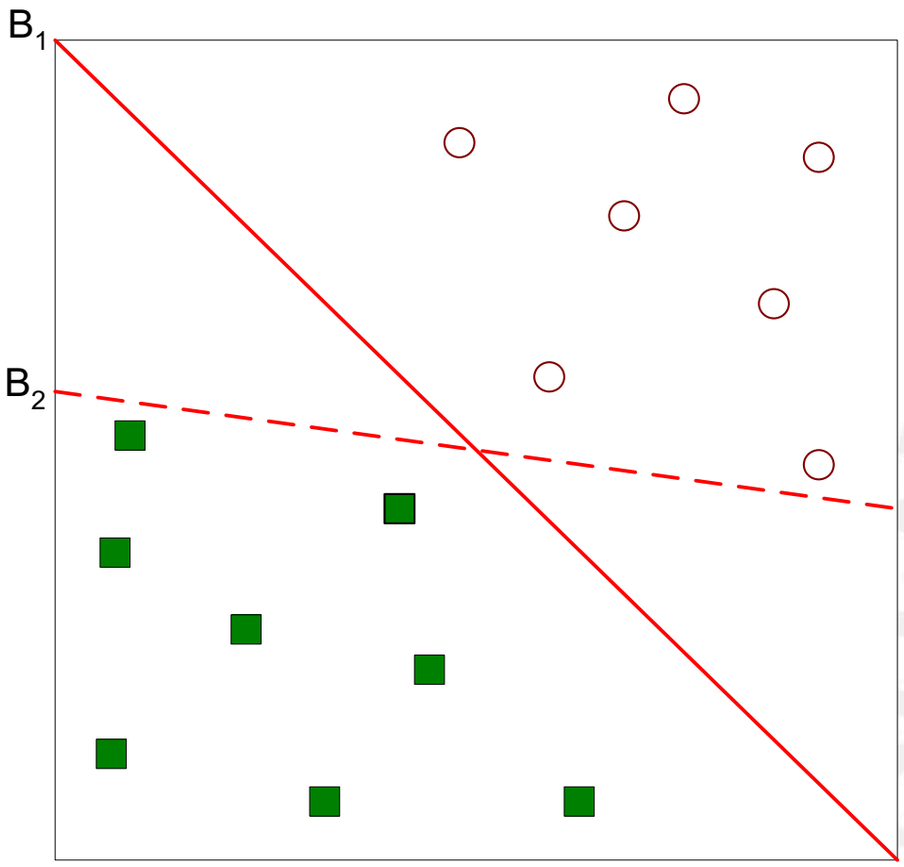
Another possible solution

Support Vector Machines



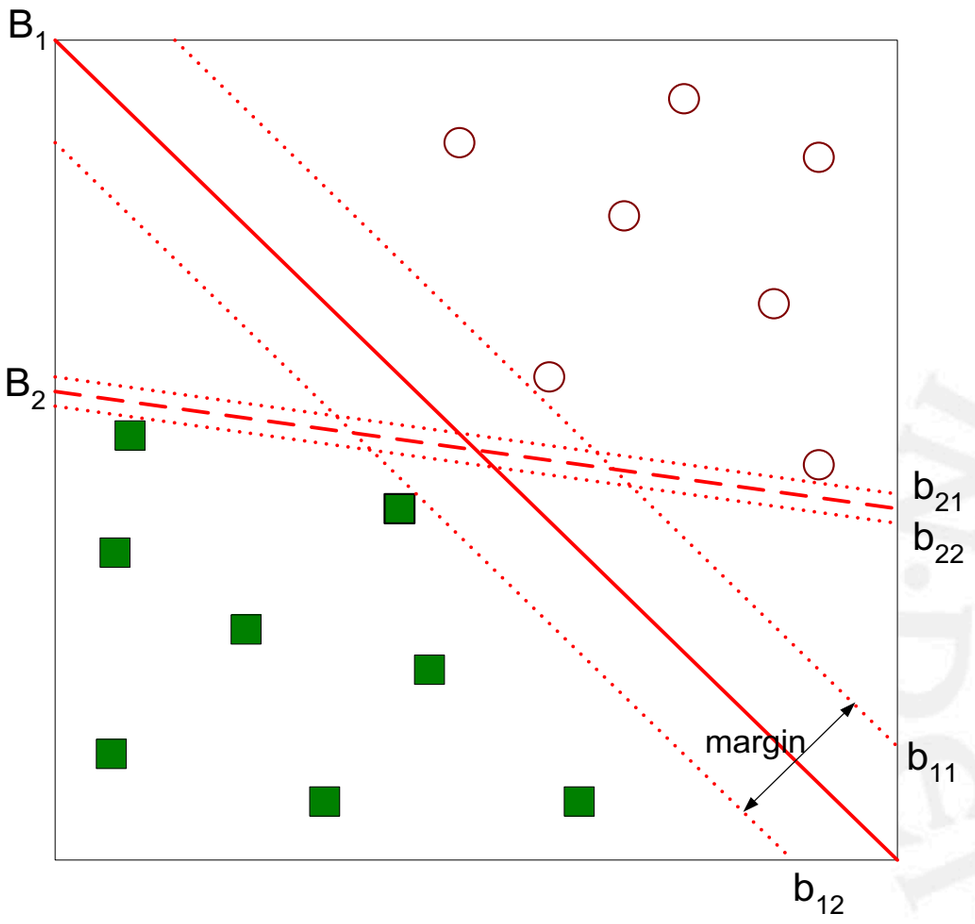
Other possible solutions

Support Vector Machines



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

Support Vector Machines

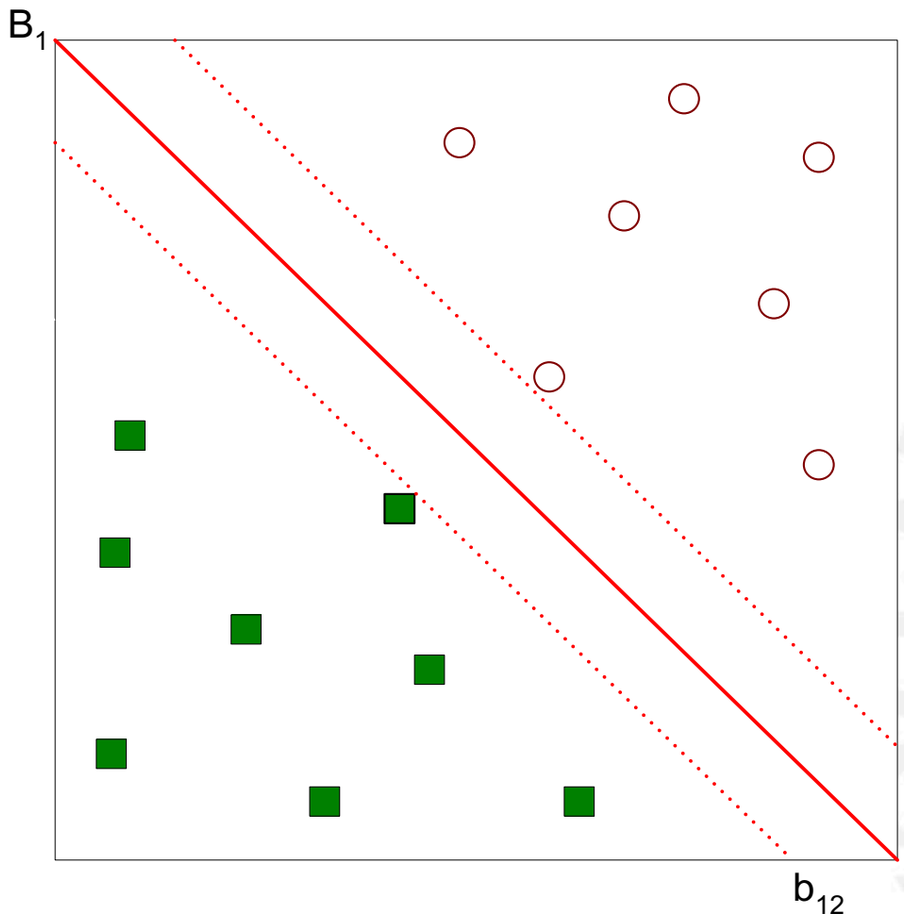


Find the hyperplane that maximizes the margin => B1 is better than B2

Support Vector Machines

$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$



$$\vec{w} \bullet \vec{x} + b = +1$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Support Vector Machines: Constrained Optimization

- We want to maximize: $$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

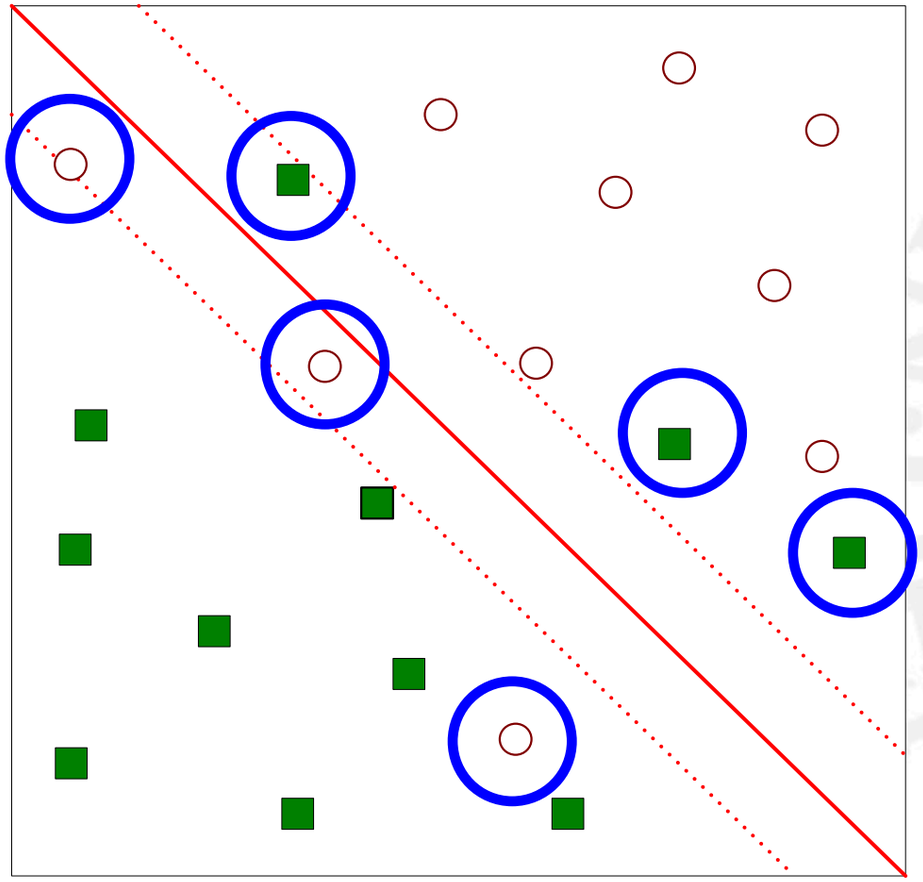
  which is equivalent to minimizing: $$L(w) = \frac{\| \vec{w} \|^2}{2}$$

- But we would like to satisfy the constraint that all data points are correctly classified:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

- This is a constrained optimization problem, with various numerical approaches for solving it (e.g., quadratic programming)

Support Vector Machines: Nonlinearly Separable Case



Support Vector Machines: Nonlinearly Separable Case

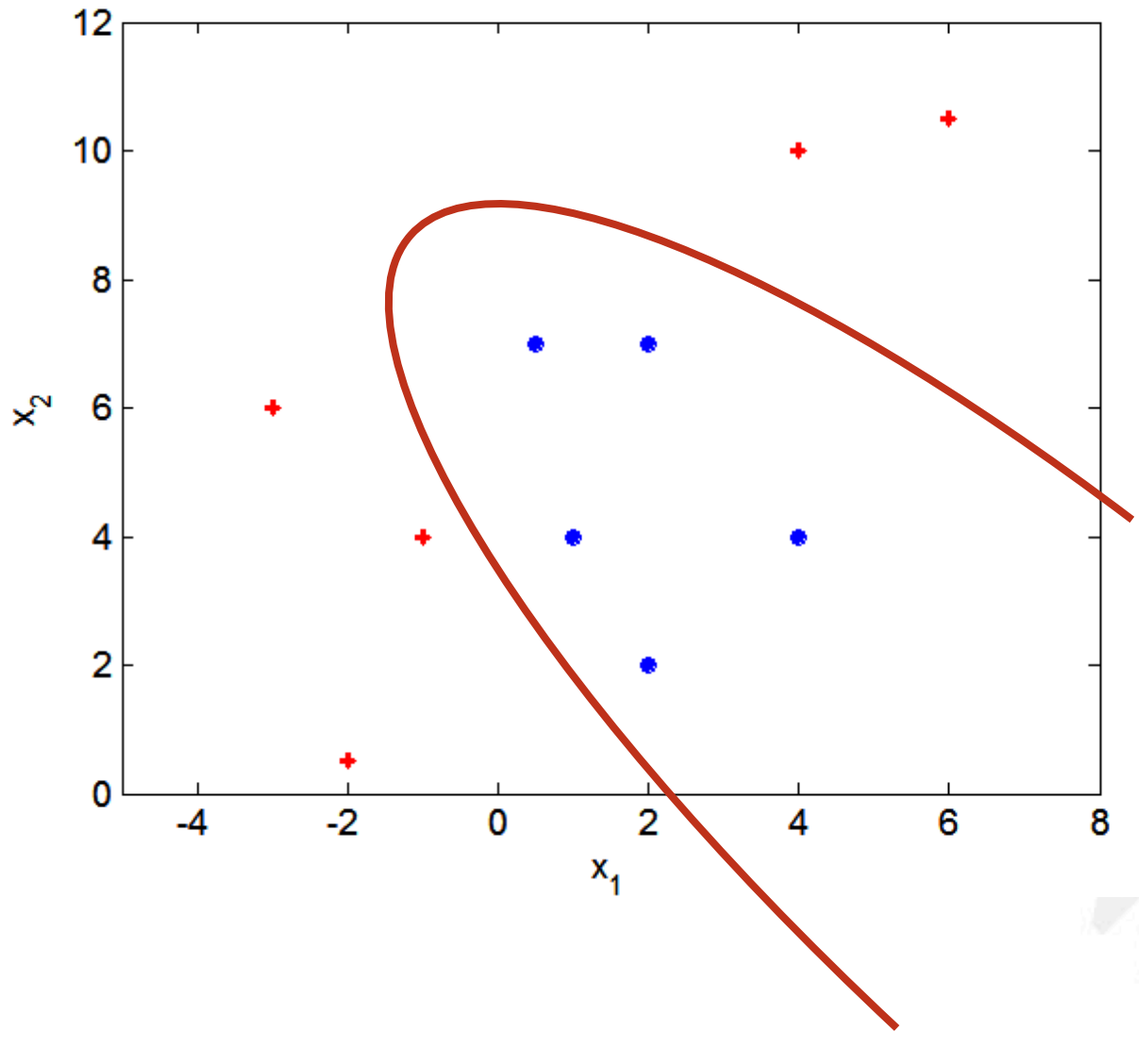
- Introduce "slack" variables ξ to allow for errors
- Constraints now become

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1\text{-}\xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

- Still want to minimize the norm, but also the amount of errors, so now:

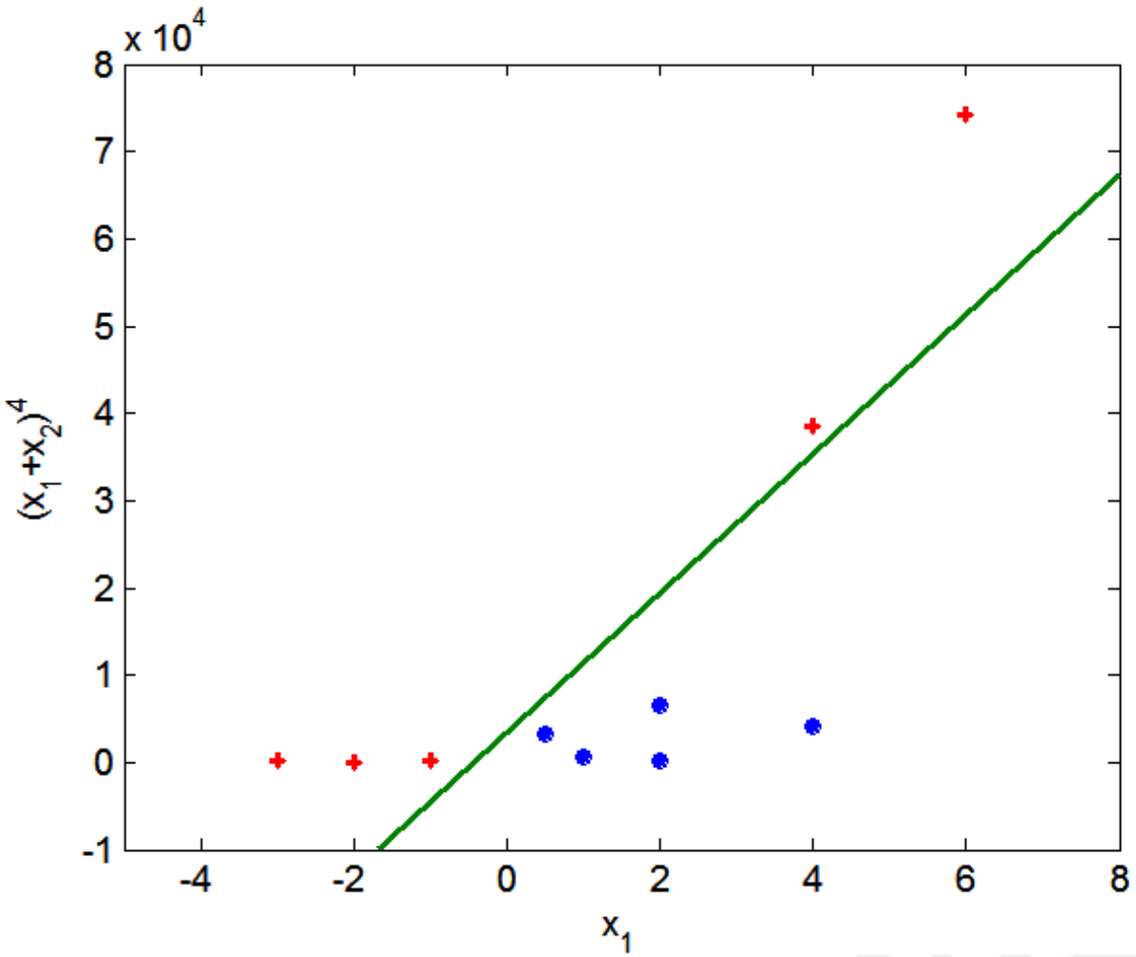
$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i \right)$$

Nonlinear Support Vector Machines



Nonlinear Support Vector Machines

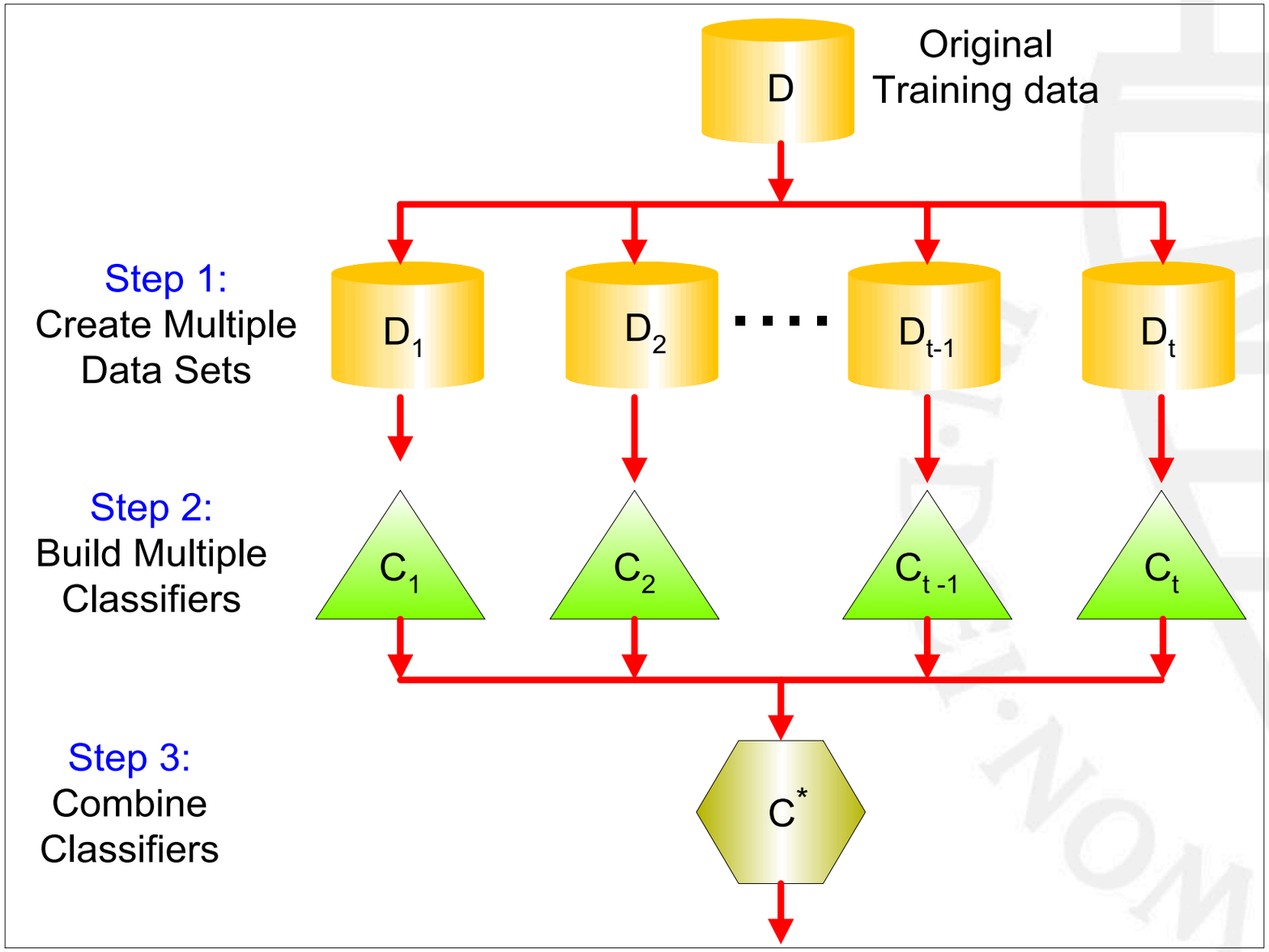
- Transform data into higher dimensional space and apply a linear SVM in this space
- The so-called "kernel trick" makes the transformation implicit instead of explicit
- Transformation back to the original space gives a nonlinear boundary



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
- Examples: bagging and boosting

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has an error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

- Catch: even classifiers trained on different parts of the data may be clearly dependent

- Basic argument especially applies for unstable classifiers such as decision trees and neural networks

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 $1 - (1 - 1/N)^N$ of being selected at least once

≈ 63.2% for large N

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
- Initially, all *N* records are assigned equal weights
- Unlike bagging, weights may change at the end of boosting round
  - 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

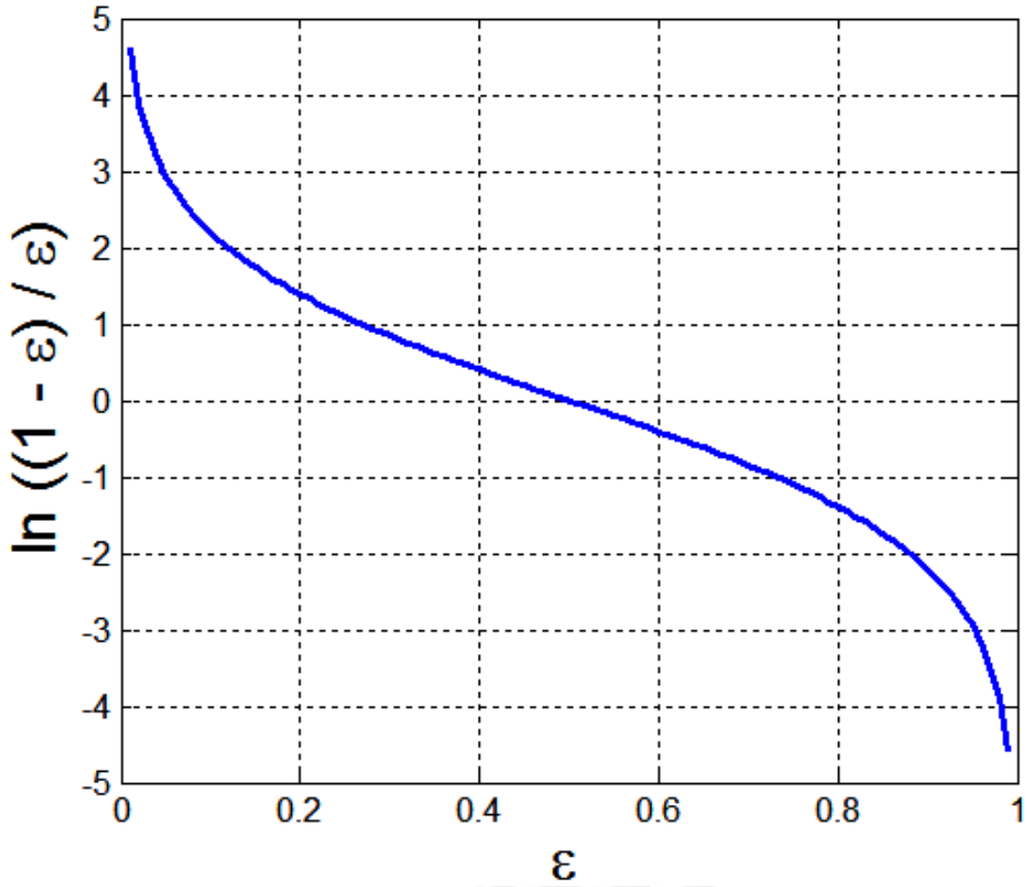
AdaBoost

- Base classifiers: $C_1, C_2, \ldots, C_T$
- Error rate:

$$\varepsilon_j = \frac{1}{N}\sum_{i=1}^{N} w_i \delta\left(C_j(x_i) \neq y_i\right)$$

- Importance of a classifier:

$$\alpha_j = \frac{1}{2}\ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$$



AdaBoost

- Weight update:

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

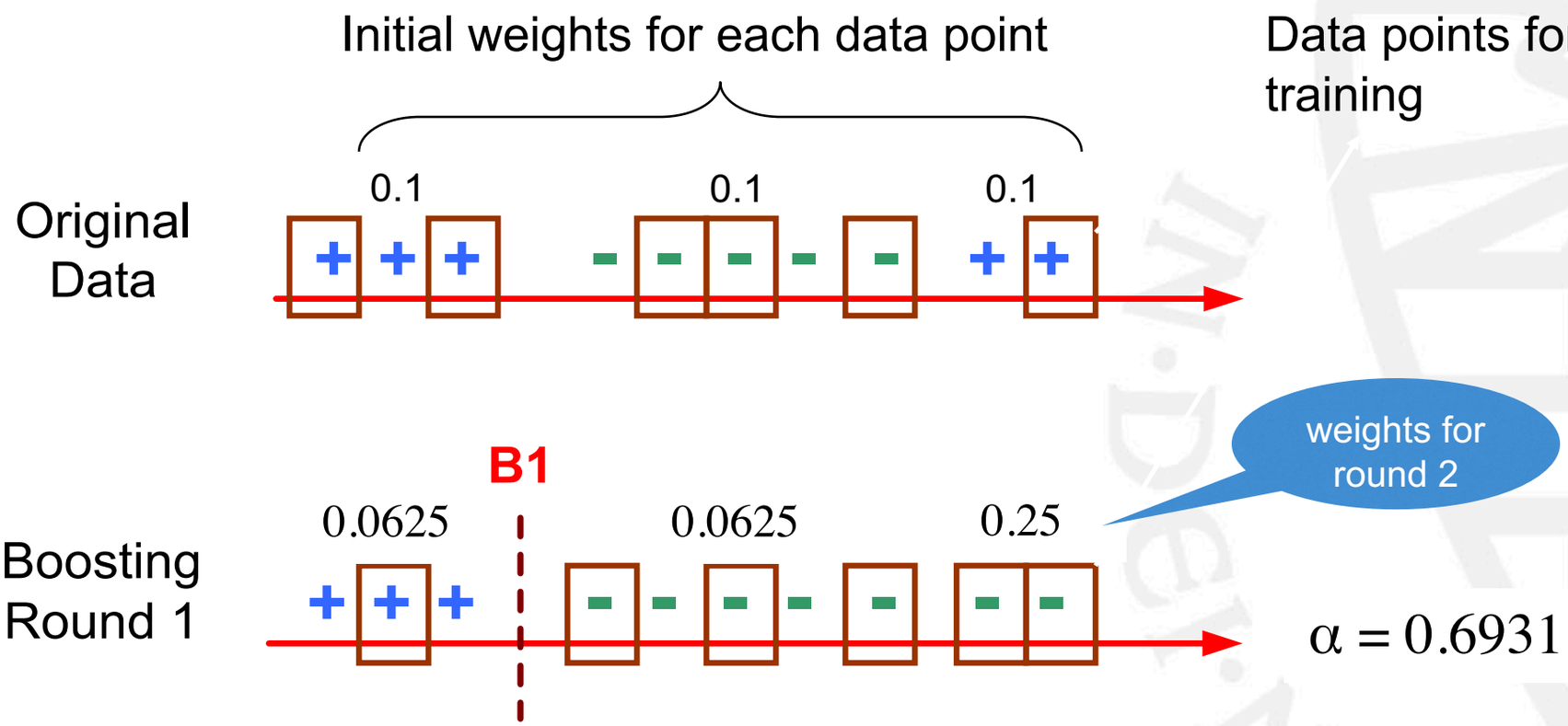
$$\text{where } Z_j \text{ is the normalization factor}$$

- Fallback: 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:

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

Illustrating AdaBoost



Illustrating AdaBoost