Classification: Alternative Techniques

Dr. Meng Qu
Rutgers University

THE STATE UNIVERSITY OF NEW JERSEY
RUTGERS

Classification: Alternative Techniques

Rule-based Classifier

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	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	Mammals
python	cold-blooded	scales	no	no	no	no	yes	Reptiles
salmon	cold-blooded	scales	no	yes	no	no	no	Fishes
whale	warm-blooded	hair	yes	yes	no	no	no	Mammals
frog	cold-blooded	none	no	semi	no	yes	yes	Amphibians
komodo dragon	cold-blooded	scales	no	no	no	yes	no	Reptiles
bat	warm-blooded	hair	yes	no	yes	yes	yes	Mammals
pigeon	warm-blooded	feathers	no	no	yes	yes	no	Birds
cat	warm-blooded	fur	yes	no	no	yes	no	Mammals
guppy	cold-blooded	scales	yes	yes	no	no	no	Fishes
alligator	cold-blooded	scales	no	semi	no	yes	no	Reptiles
penguin	warm-blooded	feathers	no	semi	no	yes	no	Birds
porcupine	warm-blooded	quills	yes	no	no	yes	yes	Mammals
eel	cold-blooded	scales	no	yes	no	no	no	Fishes
salamander	cold-blooded	none	no	semi	no	yes	yes	Amphibians

r_1: (Gives Birth = no) $\wedge$ (Aerial Creature = yes) $\longrightarrow$ Birds
r_2: (Gives Birth = no) $\wedge$ (Aquatic Creature = yes) $\longrightarrow$ Fishes
r_3: (Gives Birth = yes) $\wedge$ (Body Temperature = warm-blooded) $\longrightarrow$ Mammals
r_4: (Gives Birth = no) $\wedge$ (Aerial Creature = no) $\longrightarrow$ Reptiles
r_5: (Aquatic Creature = semi) $\longrightarrow$ Amphibians

Application of Rule-Based Classifier

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

r_1: (Gives Birth = no) $\wedge$ (Aerial Creature = yes) $\longrightarrow$ Birds
r_2: (Gives Birth = no) $\wedge$ (Aquatic Creature = yes) $\longrightarrow$ Fishes
r_3: (Gives Birth = yes) $\wedge$ (Body Temperature = warm-blooded) $\longrightarrow$ Mammals
r_4: (Gives Birth = no) $\wedge$ (Aerial Creature = no) $\longrightarrow$ Reptiles
r_5: (Aquatic Creature = semi) $\longrightarrow$ Amphibians

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates
hawk	warm-blooded	feather	no	no	yes	yes	no
grizzly bear	warm-blooded	fur	yes	no	no	yes	yes

The rule r1 covers a hawk => Bird

The rule r3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

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

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**

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

r_1: (Gives Birth = no) $\wedge$ (Aerial Creature = yes) $\longrightarrow$ Birds
r_2: (Gives Birth = no) $\wedge$ (Aquatic Creature = yes) $\longrightarrow$ Fishes
r_3: (Gives Birth = yes) $\wedge$ (Body Temperature = warm-blooded) $\longrightarrow$ Mammals
r_4: (Gives Birth = no) $\wedge$ (Aerial Creature = no) $\longrightarrow$ Reptiles
r_5: (Aquatic Creature = semi) $\longrightarrow$ Amphibians

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates
lemur	warm-blooded	fur	yes	no	no	yes	yes
turtle	cold-blooded	scales	no	semi	no	yes	no
dogfish shark	cold-blooded	scales	yes	yes	no	no	no

A lemur triggers rule r3, so it is classified as a mammal

A turtle triggers both r4 and r5

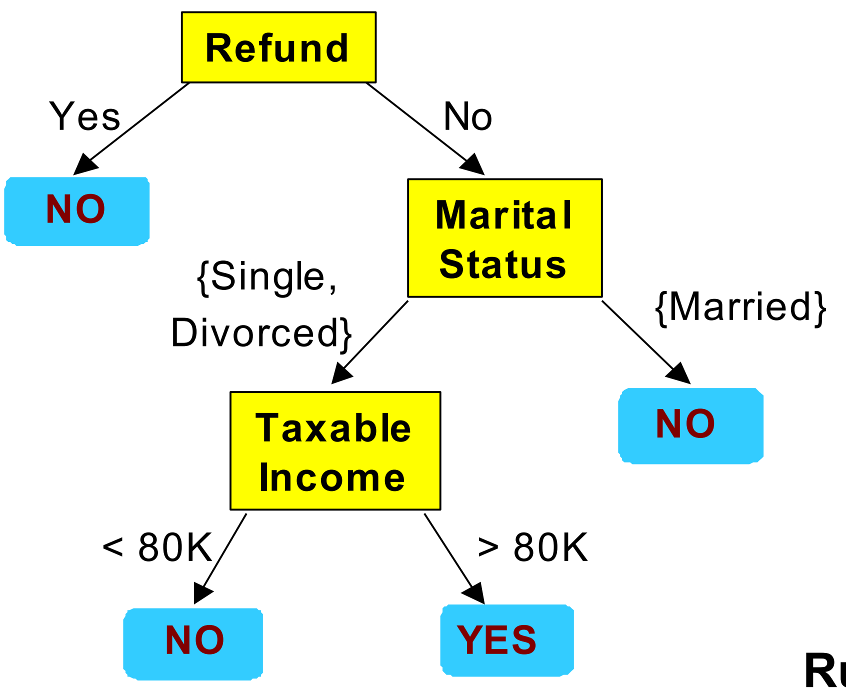
A dogfish shark triggers none of the rules

Characteristics of Rule-Based Classifier

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if 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

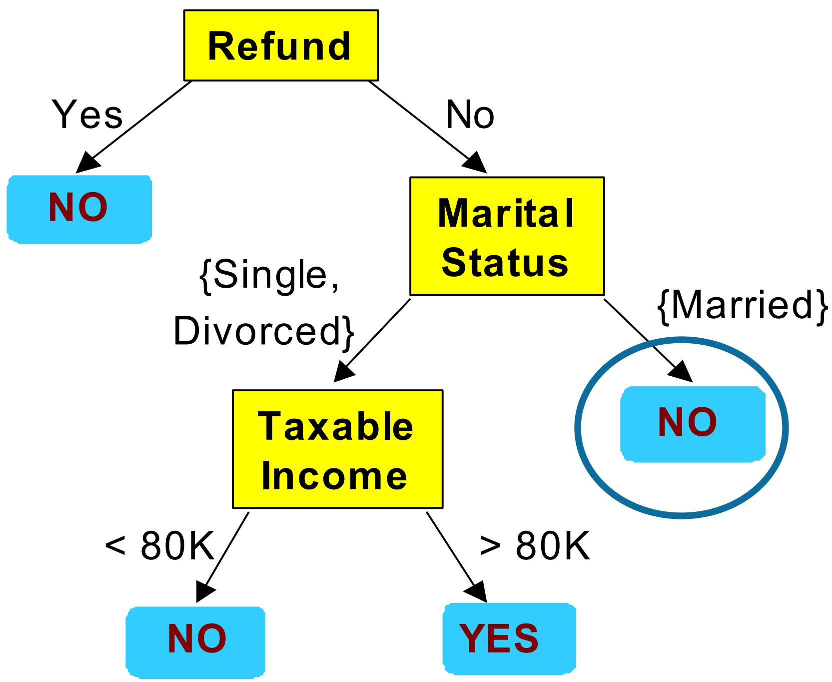
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(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) $\rightarrow$ No

Simplified Rule: (Status=Married) $\rightarrow$ 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

r_1: (Gives Birth = no) $\wedge$ (Aerial Creature = yes) $\longrightarrow$ Birds
r_2: (Gives Birth = no) $\wedge$ (Aquatic Creature = yes) $\longrightarrow$ Fishes
r_3: (Gives Birth = yes) $\wedge$ (Body Temperature = warm-blooded) $\longrightarrow$ Mammals
r_4: (Gives Birth = no) $\wedge$ (Aerial Creature = no) $\longrightarrow$ Reptiles
r_5: (Aquatic Creature = semi) $\longrightarrow$ Amphibians

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates
turtle	cold-blooded	scales	no	semi	no	yes	no

Rule Ordering Schemes

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

Rule-Based Ordering

(Skin Cover=feathers, Aerial Creature=yes)
==> Birds

(Body temperature=warm-blooded,
Gives Birth=yes) ==> Mammals

(Body temperature=warm-blooded,
Gives Birth=no) ==> Birds

(Aquatic Creature=semi)) ==> Amphibians

(Skin Cover=scales, Aquatic Creature=no)
==> Reptiles

(Skin Cover=scales, Aquatic Creature=yes)
==> Fishes

(Skin Cover=none) ==> Amphibians

Class-Based Ordering

(Skin Cover=feathers, Aerial Creature=yes)
==> Birds

(Body temperature=warm-blooded,
Gives Birth=no) ==> Birds

(Body temperature=warm-blooded,
Gives Birth=yes) ==> Mammals

(Aquatic Creature=semi)) ==> Amphibians

(Skin Cover=none) ==> Amphibians

(Skin Cover=scales, Aquatic Creature=no)
==> Reptiles

(Skin Cover=scales, Aquatic Creature=yes)
==> Fishes

Building Classification Rules

- Direct Method:
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, 1R, and AQ

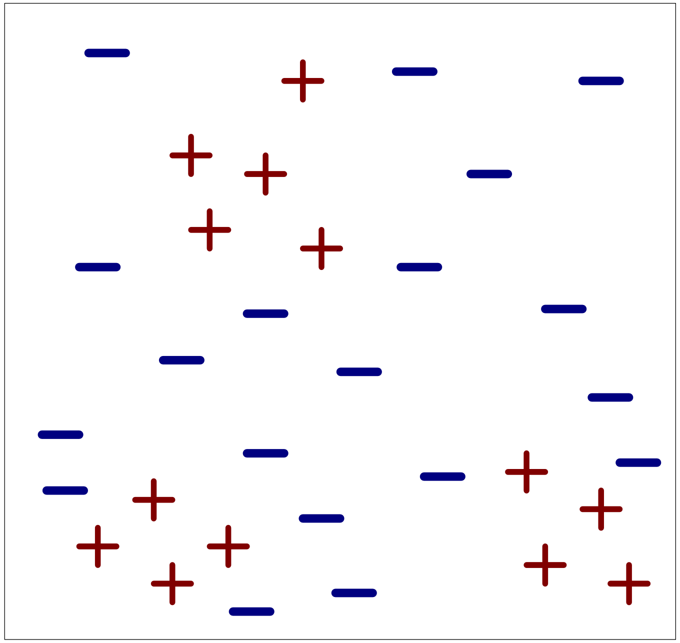
- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, SVM, etc).
  - e.g: C4.5rules

Direct Method: Sequential Covering

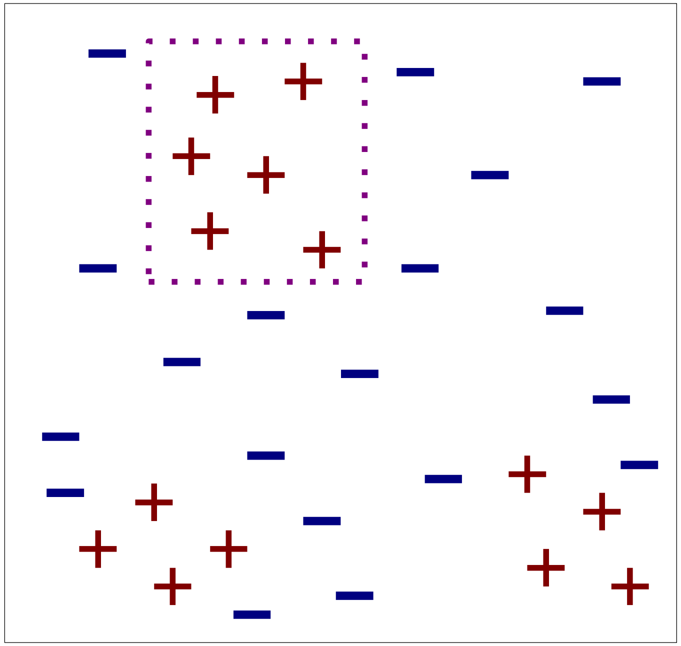
Algorithm 1.1 Sequential covering algorithm.

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

Example of Sequential Covering

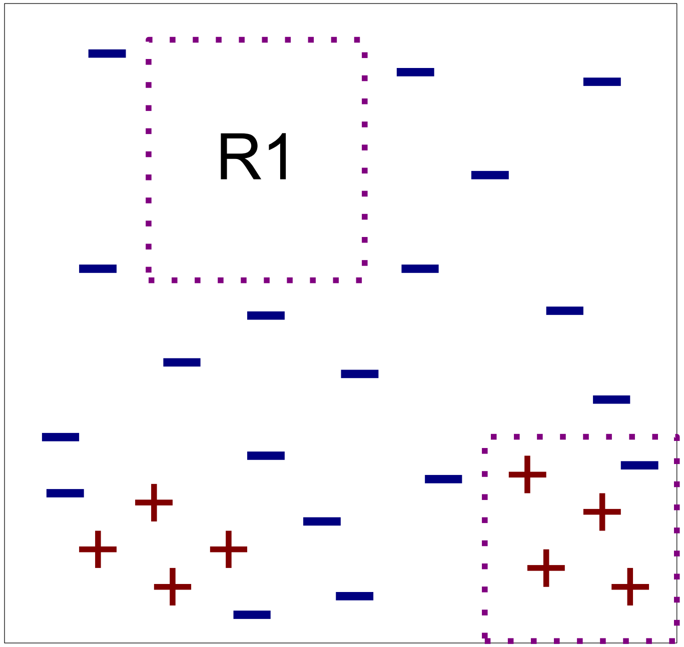


(i) Original Data

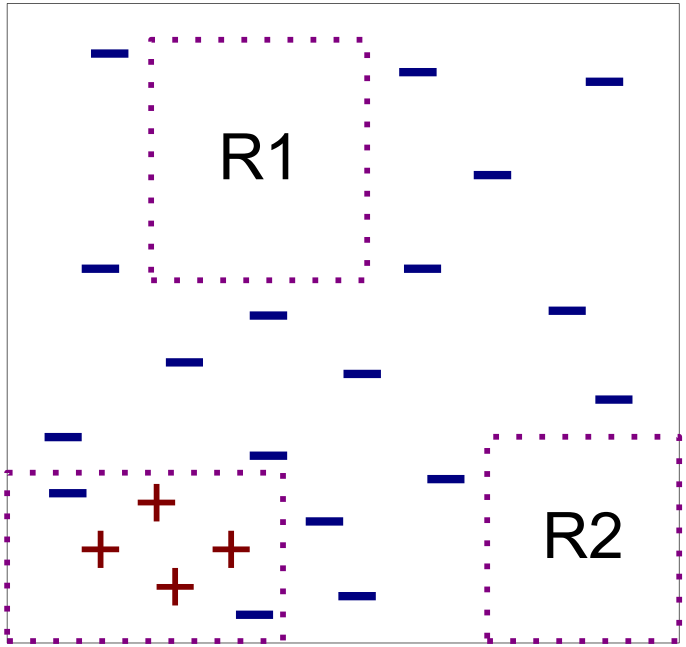


(ii) Step 1

Example of Sequential Covering...



(iii) Step 2



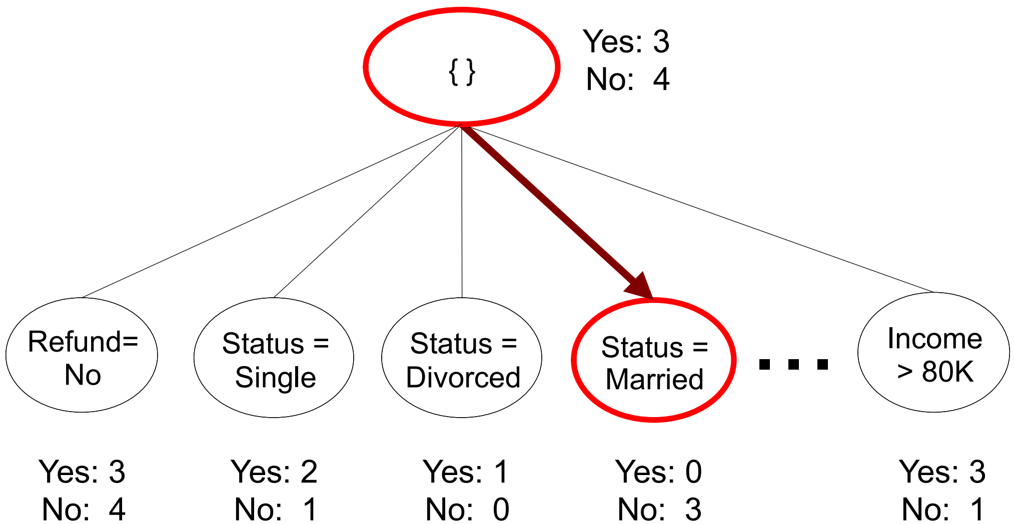
(iv) Step 3

Aspects of Sequential Covering

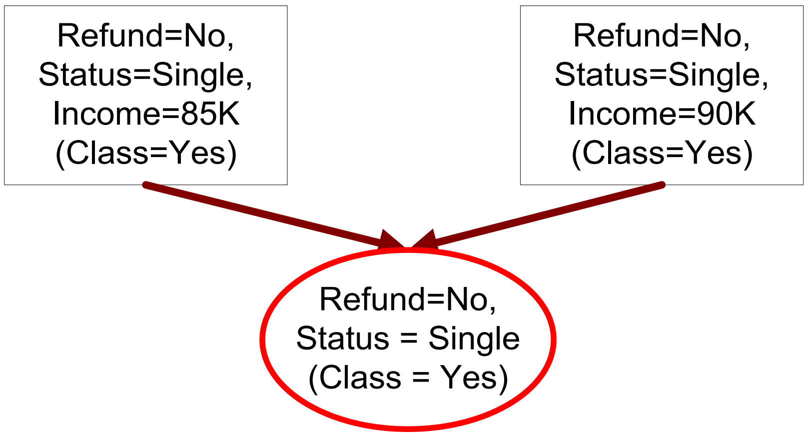
- Rule Growing
  - Rule evaluation
- Instance Elimination
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

Rule Evaluation

- Evaluation metric determines which conjunct should be added during rule growing
  - 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 of class *c* covered by rule

k : Number of classes

p : Prior probability

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

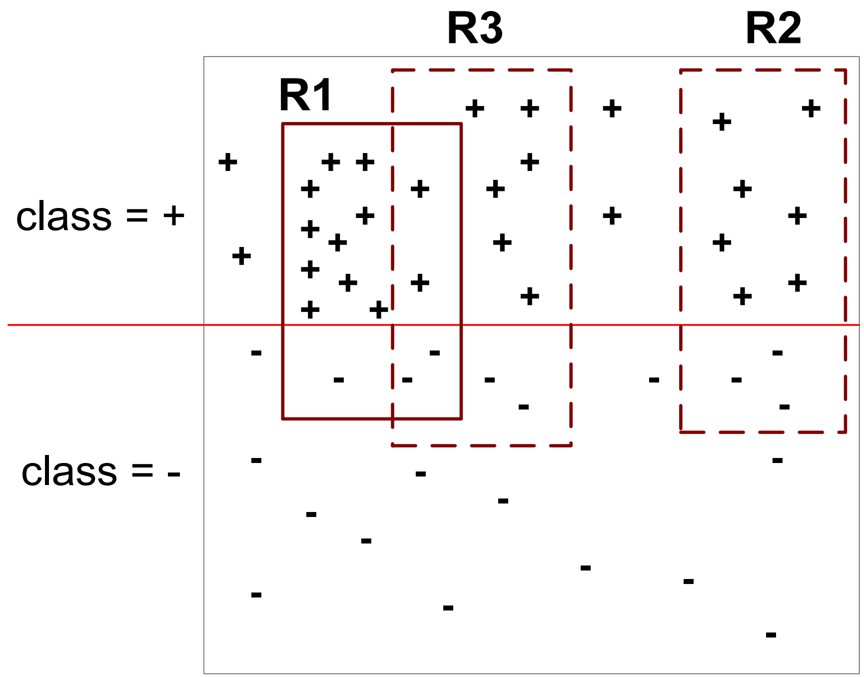
      n0: number of negative instances covered by R0

      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 underestimating accuracy of rule
  - Compare rules R2 and R3 in the diagram



Stopping Criterion and Rule Pruning

- Examples of stopping criterion:
  - If rule does not improve significantly after adding conjunct
  - If rule starts covering examples from another class

- Rule Pruning
  - Similar to post-pruning of decision trees
  - Example: using validation set (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

- Initial rule set is empty
- Repeat
  - Grow a single rule
  - Remove Instances covered by the rule
  - Prune the rule (if necessary)
  - Add rule to the current rule set

Direct Method: RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn the rules for positive class
  - Use negative class as default

- 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

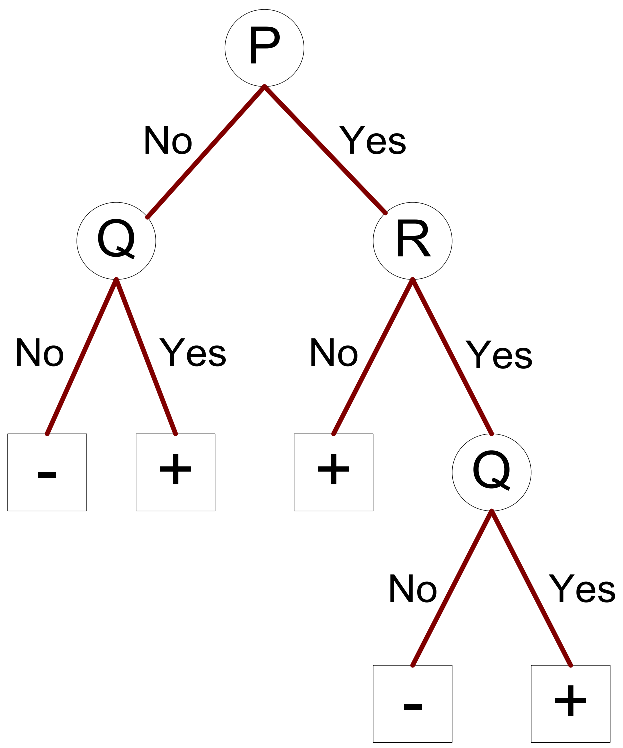
Direct Method: RIPPER

- Rule growing:
  - Start from an empty rule: $\{\} \rightarrow +$
  - 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
    - Grow a rule to cover 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

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) ==> +

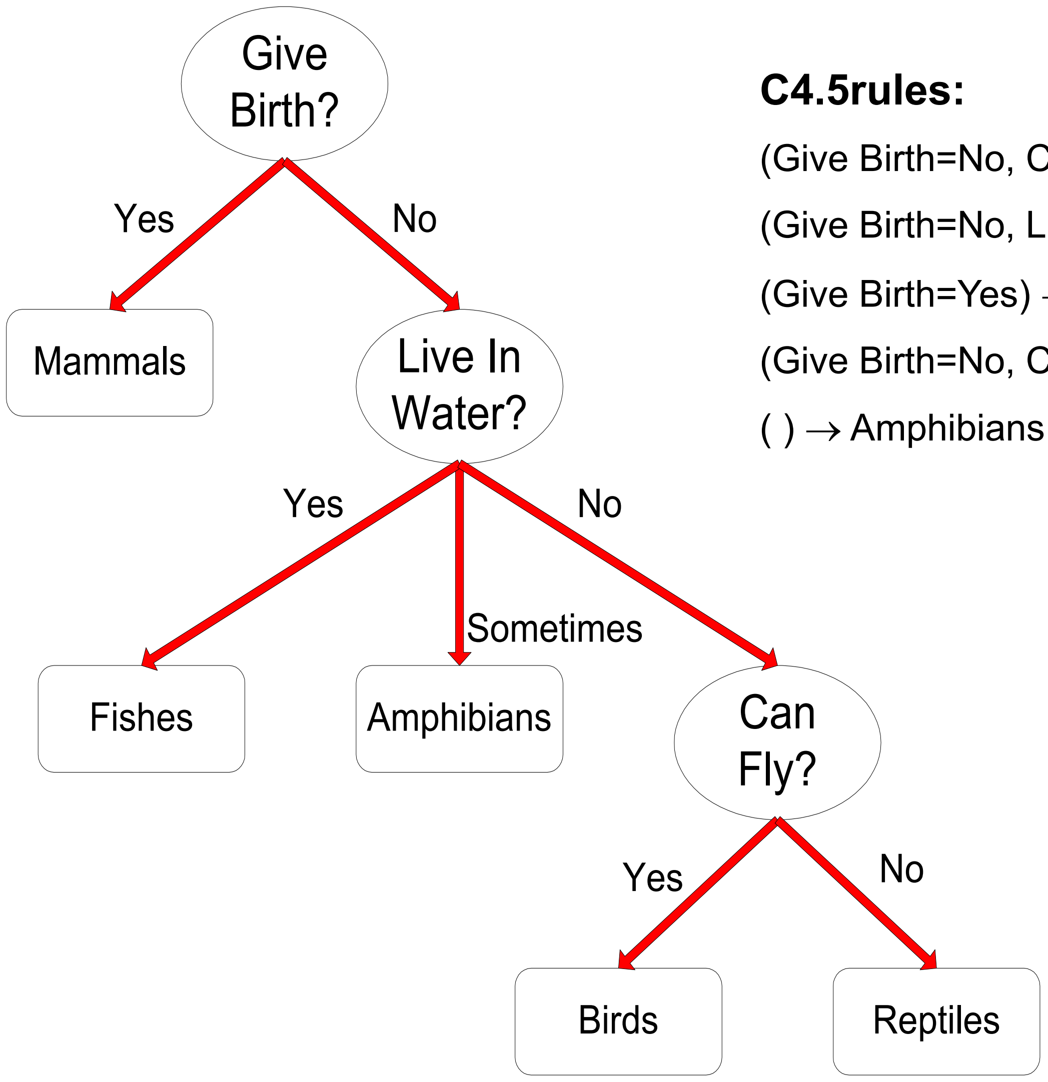
Indirect Method: C4.5rules

- Extract rules for every path from root to leaf nodes
- For each rule, r: A → y,
  - consider 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 pessimistic error rate can no longer be improved

Indirect Method: C4.5rules

- Use class-based ordering
  - Rules that predict the same class are grouped together into the same subset
  - Compute total description length for each class
  - Classes are ordered in increasing order of their total description length

Example



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

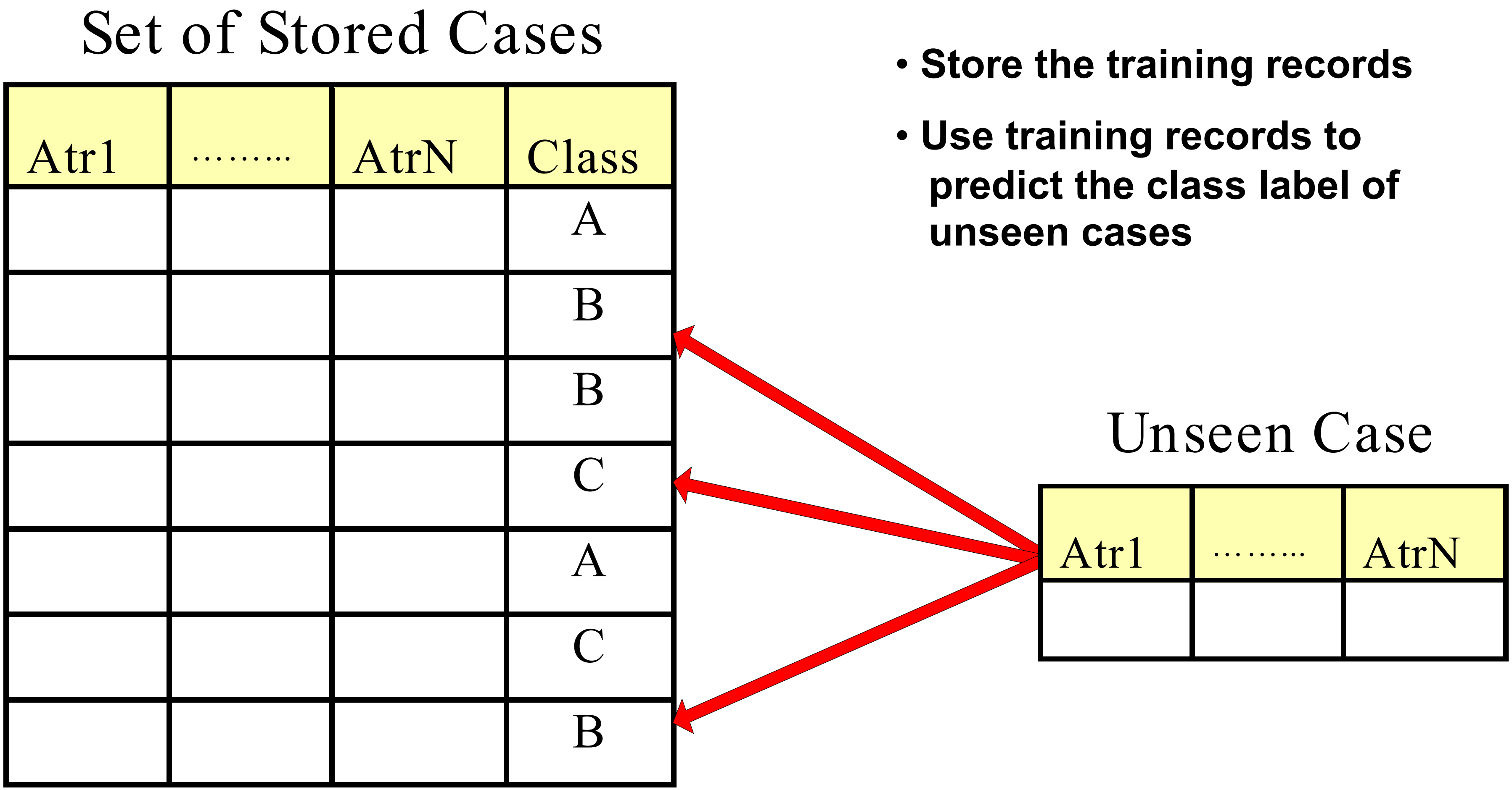
Characteristics of Rule-Based Classifiers

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

Classification: Alternative Techniques

Instance-Based Classifiers

Instance-Based Classifiers



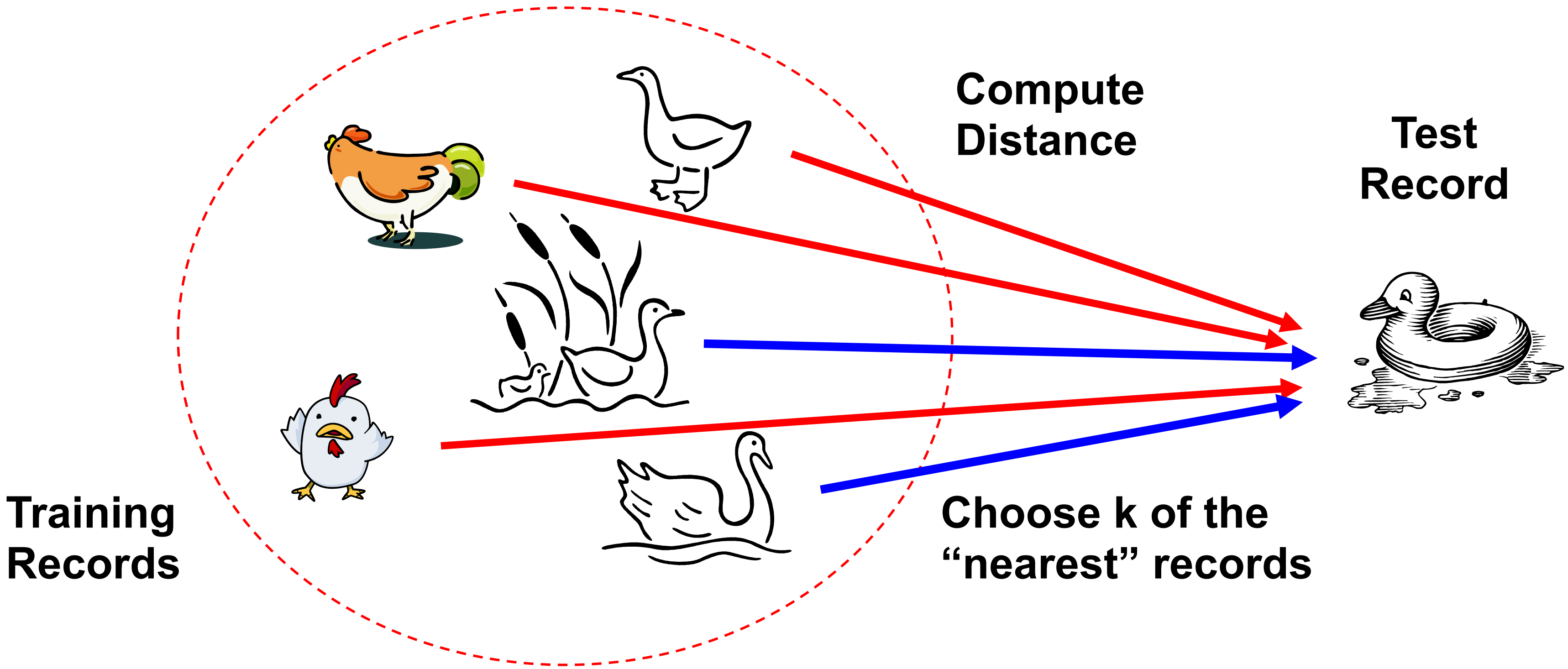
- Store the training records
- Use training records to predict the class label of unseen cases

Instance Based Classifiers

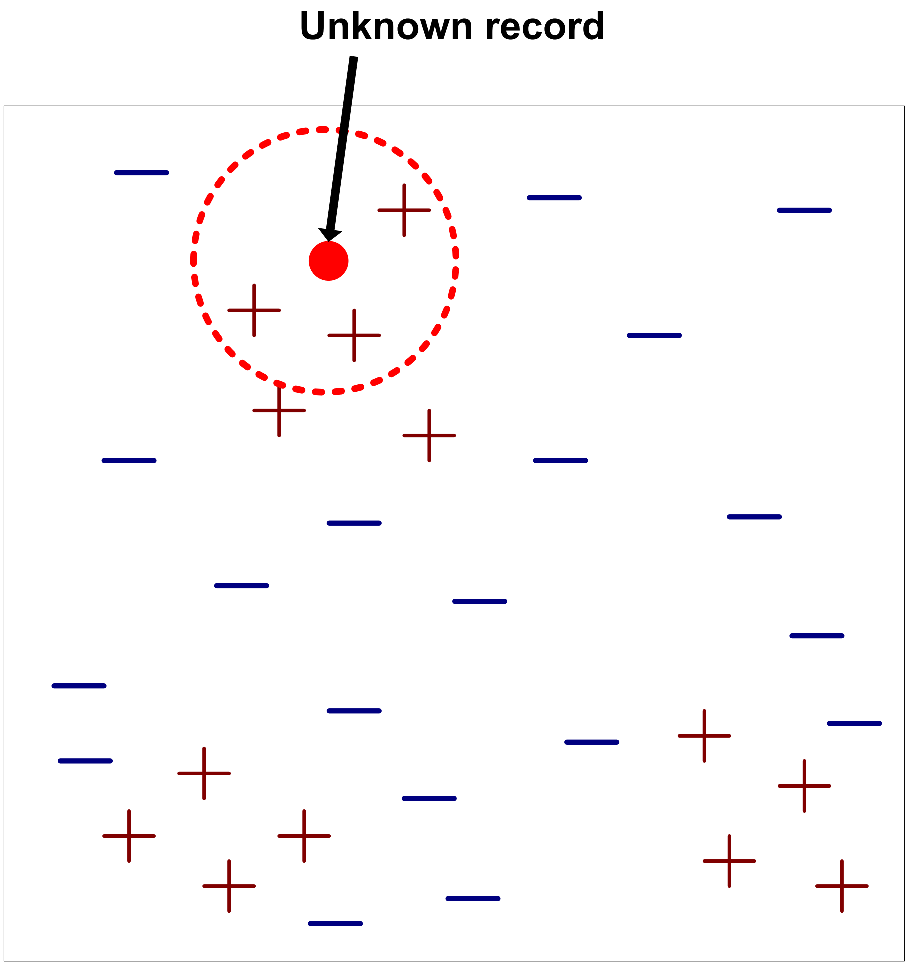
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses k "closest" points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

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



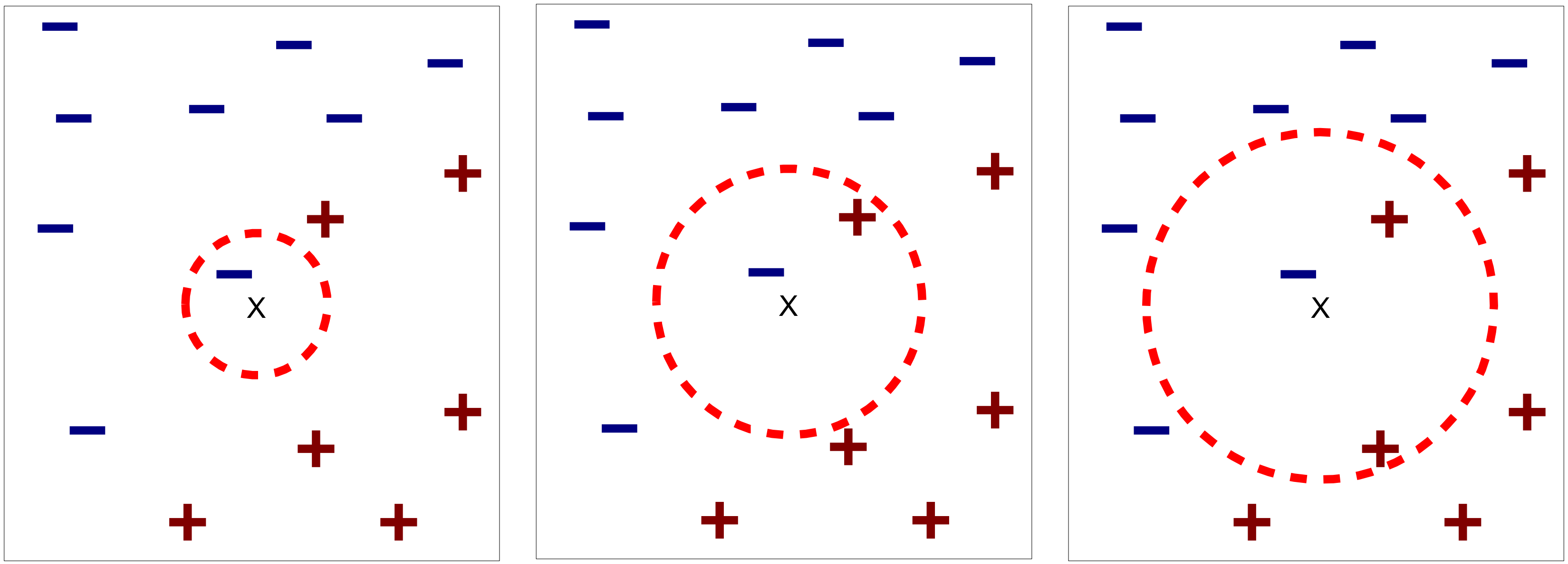
Nearest-Neighbor Classifiers



- 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

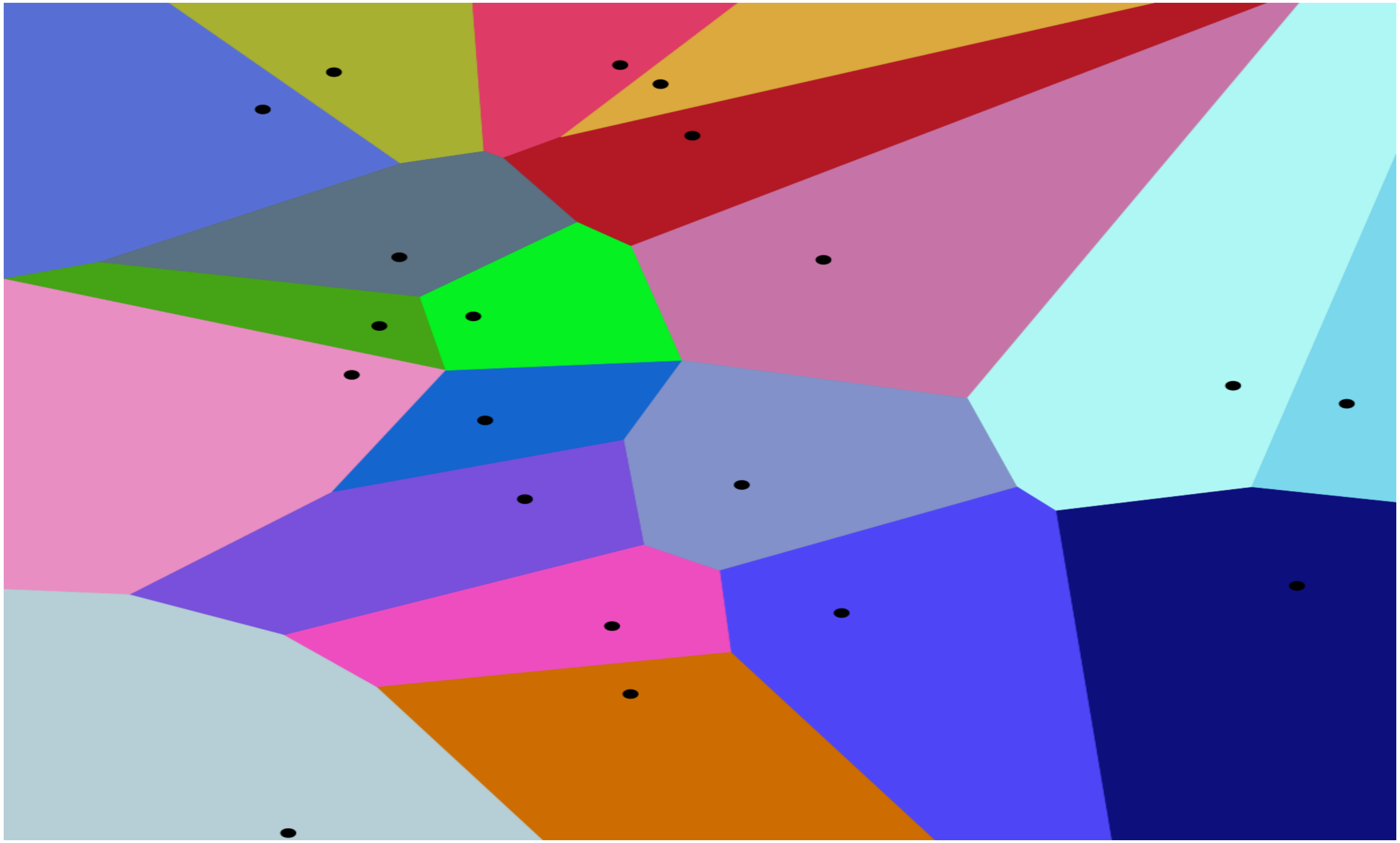


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

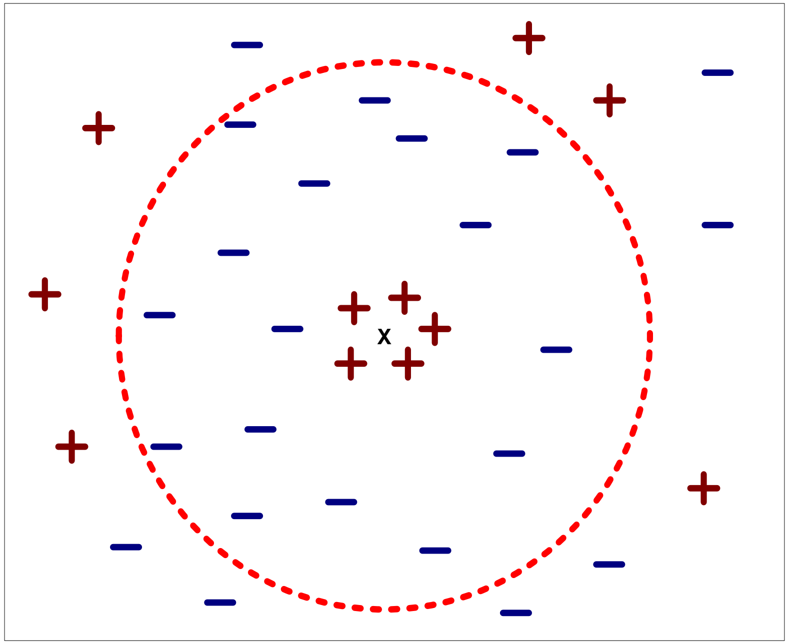
- Compute distance between two points:
  - Example: Euclidean distance

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

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weight 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

Classification: Alternative Techniques

Bayesian Classifiers

Classification: Alternative Techniques

Ensemble Methods

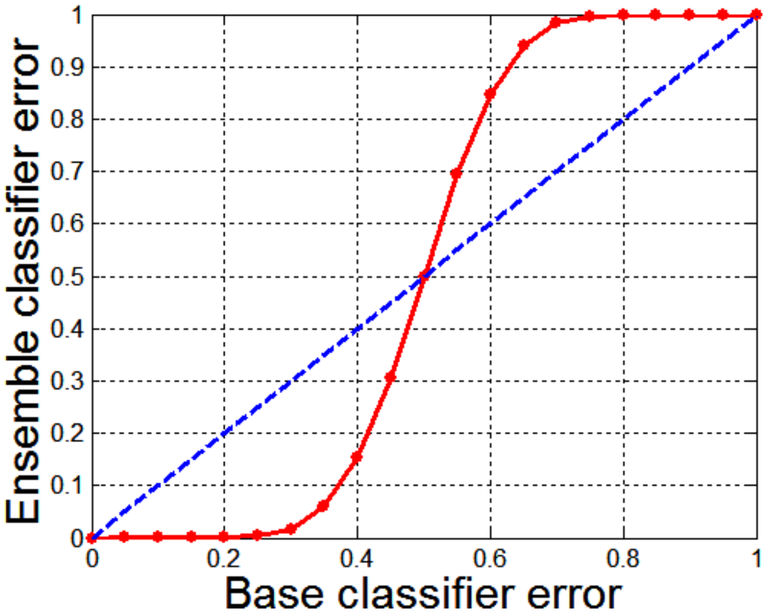
Ensemble Methods

- Construct a set of classifiers from the training data

- Predict class label of test records by combining the predictions made by multiple classifiers

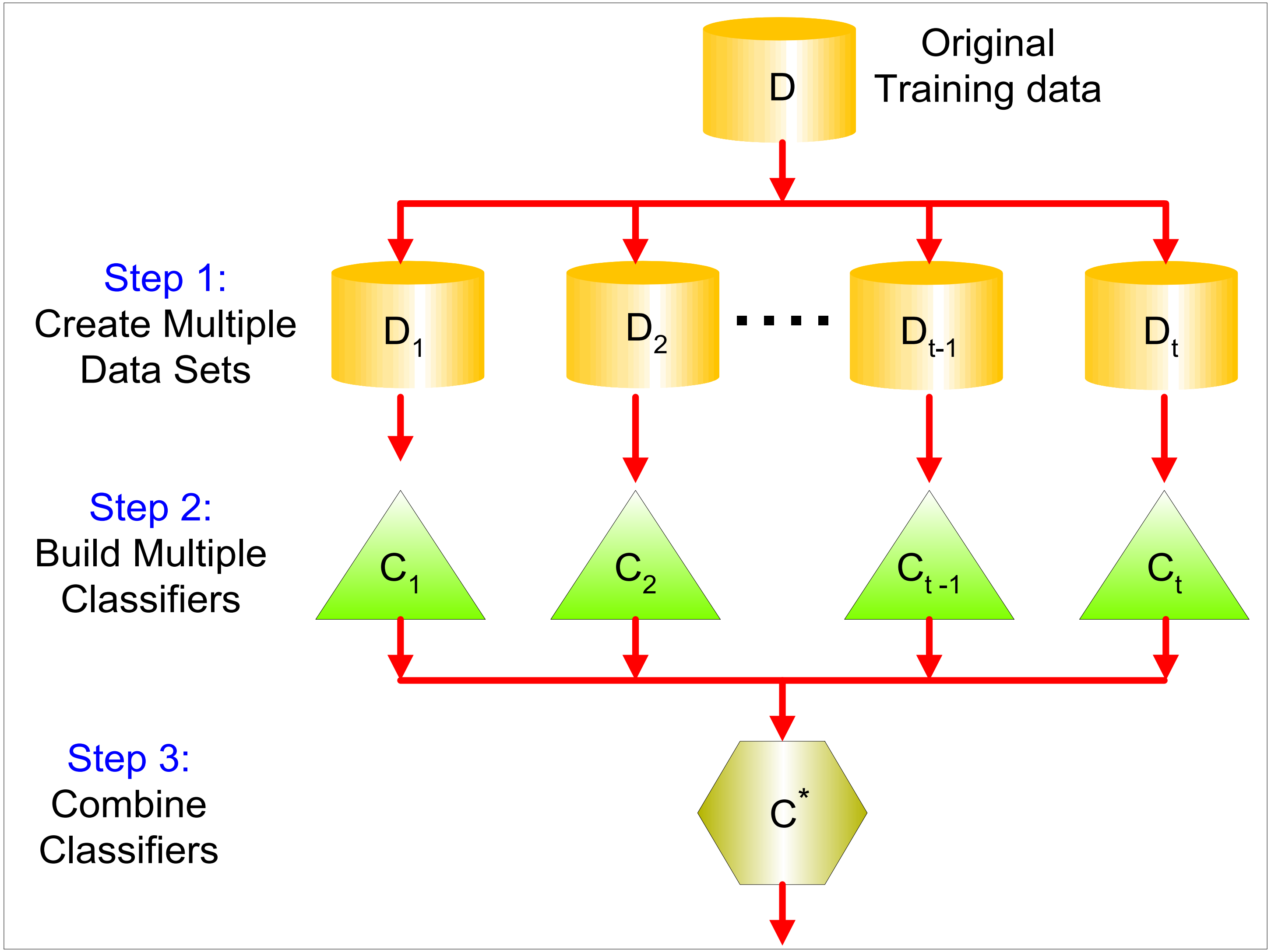
Why Ensemble Methods work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume errors made by classifiers are uncorrelated
  - Probability that the ensemble classifier makes a wrong prediction:



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

General Approach



Types of Ensemble Methods

- Bayesian ensemble
  - Example: Mixture of Gaussian
- Manipulate data distribution
  - Example: Resampling method
- Manipulate input features
  - Example: Feature subset selection
- Manipulate class labels
  - Example: error-correcting output coding
- Introduce randomness into learning algorithm
  - Example: Random forests

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

Bagging Algorithm

Algorithm 5.6 Bagging Algorithm

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size n, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

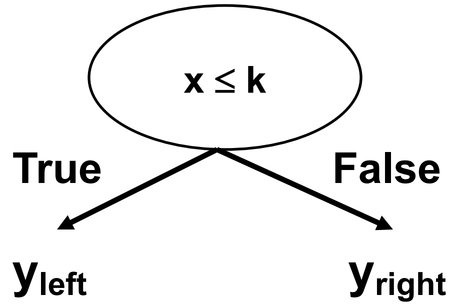
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

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 each 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

AdaBoost

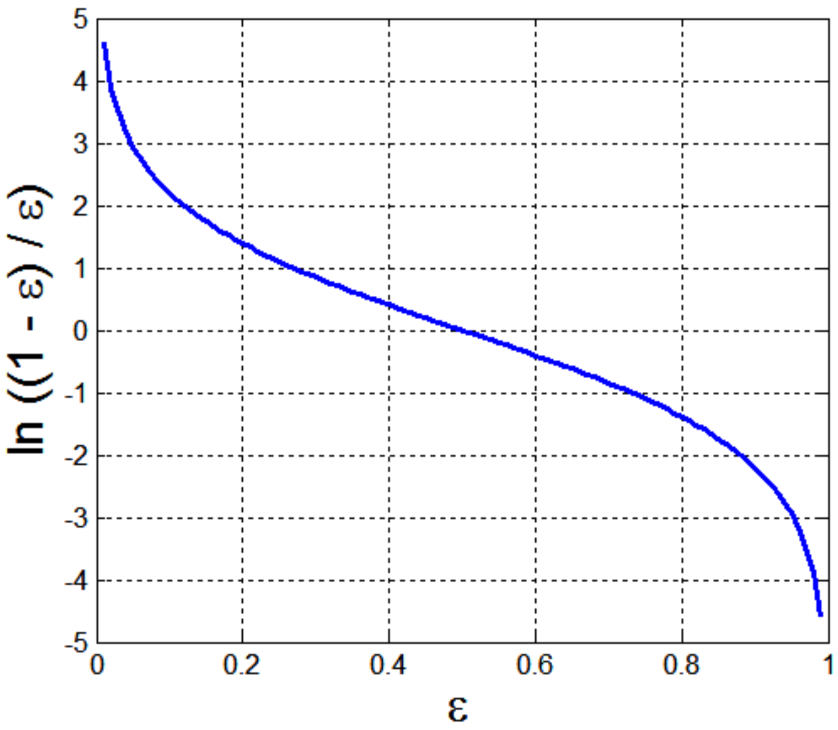
- Base classifiers: $C_1, C_2, \ldots, C_T$

- 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:

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



AdaBoost Algorithm

- 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:

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

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

1: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Initialize the weights for all n instances.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all instances in the original training set, D.
7: $\epsilon_i = \frac{1}{n}\left[\sum_j w_j\, \delta\big(C_i(x_j) \neq y_j\big)\right]$ {Calculate the weighted error}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Reset the weights for all n instances.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each instance according to equation (5.88).
14: **end for**
15: $C^*(\mathbf{x}) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\big(C_j(\mathbf{x}) = y\big)$.

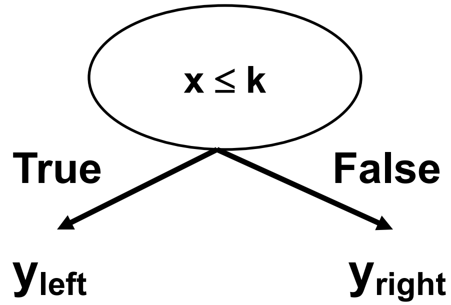
AdaBoost Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

AdaBoost Example

- Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

- Classification

	Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
	1	-1	-1	-1	-1	-1	-1	-1	1	1	1
	2	1	1	1	1	1	1	1	1	1	1
	3	1	1	1	-1	-1	-1	-1	-1	-1	-1
	Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Predicted Class	Sign	1	1	1	-1	-1	-1	-1	1	1	1

Classification: Alternative Techniques

Imbalanced Class Problem

Class Imbalance Problem

- Lots of classification problems where the classes are skewed (more records from one class than another)
  - Credit card fraud
  - Intrusion detection
  - Defective products in manufacturing assembly line

Challenges

- Evaluation measures such as accuracy is not well-suited for imbalanced class

- Detecting the rare class is like finding needle in a haystack

Confusion Matrix

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Problem with Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If a model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - This is misleading because the model does not detect any class 1 example
  - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

Alternative Measures

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b} = TPR$$

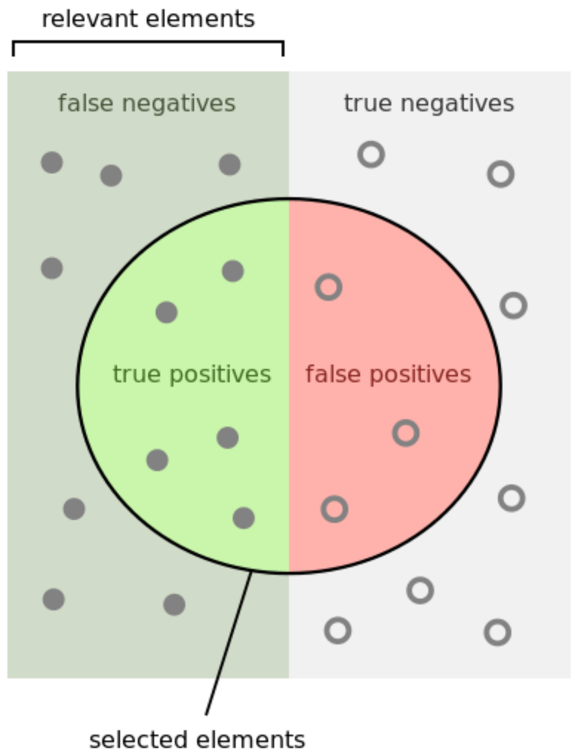
$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

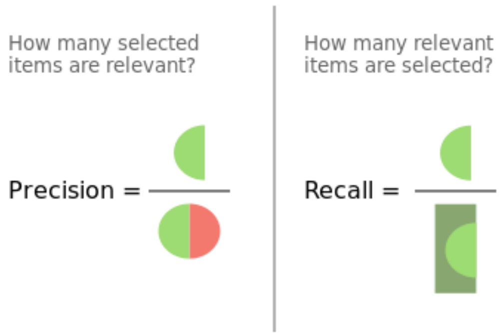
Alternative Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b} = TPR$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$





ROC (Receiver Operating Characteristic)

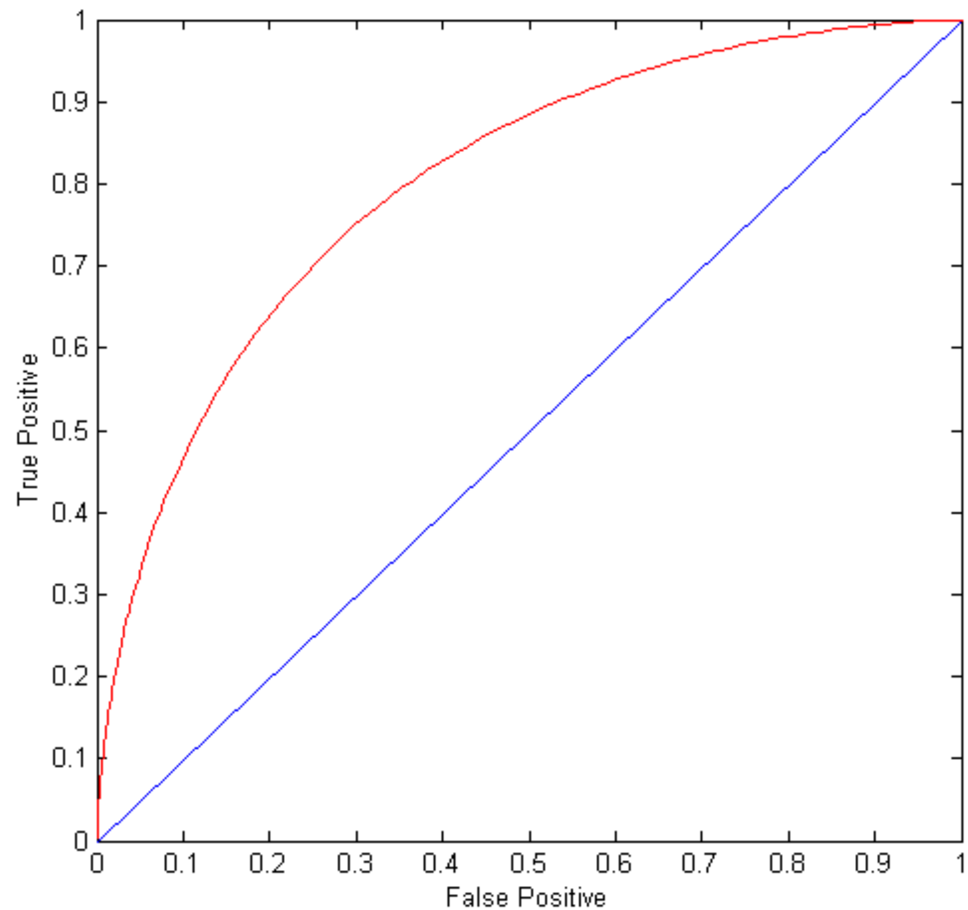
- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TPR against FPR
  - TPR = TP/(TP+FN), FPR = FP/(TN+FP)
  - Performance of a model represented as a point in an ROC curve
  - Changing the threshold parameter of classifier changes the location of the point

ROC Curve

(TPR,FPR):

- (0,0): declare everything
  to be negative class
- (1,1): declare everything
  to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



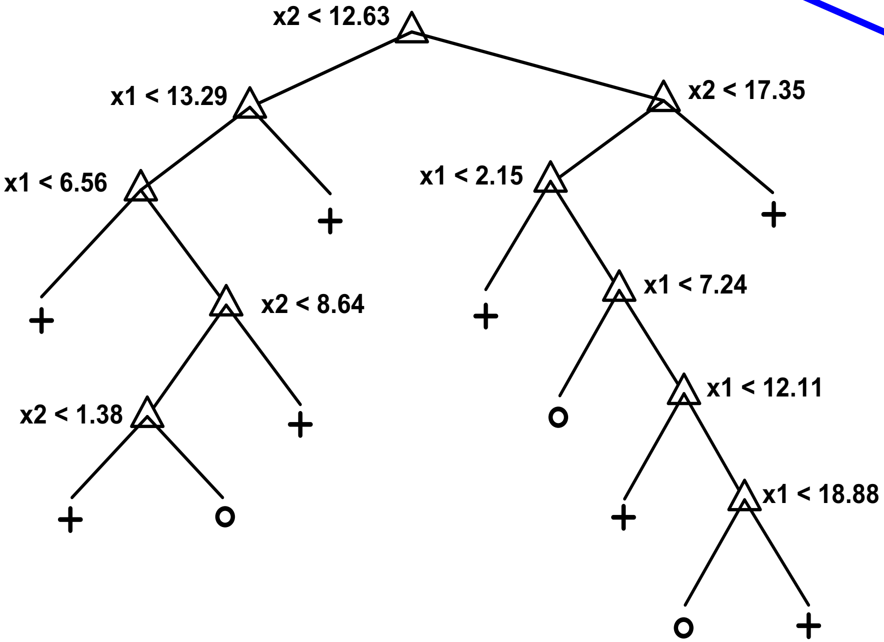
ROC (Receiver Operating Characteristic)

- To draw ROC curve, classifier must produce continuous-valued output
  - Outputs are used to rank test records, from the most likely positive class record to the least likely positive class record

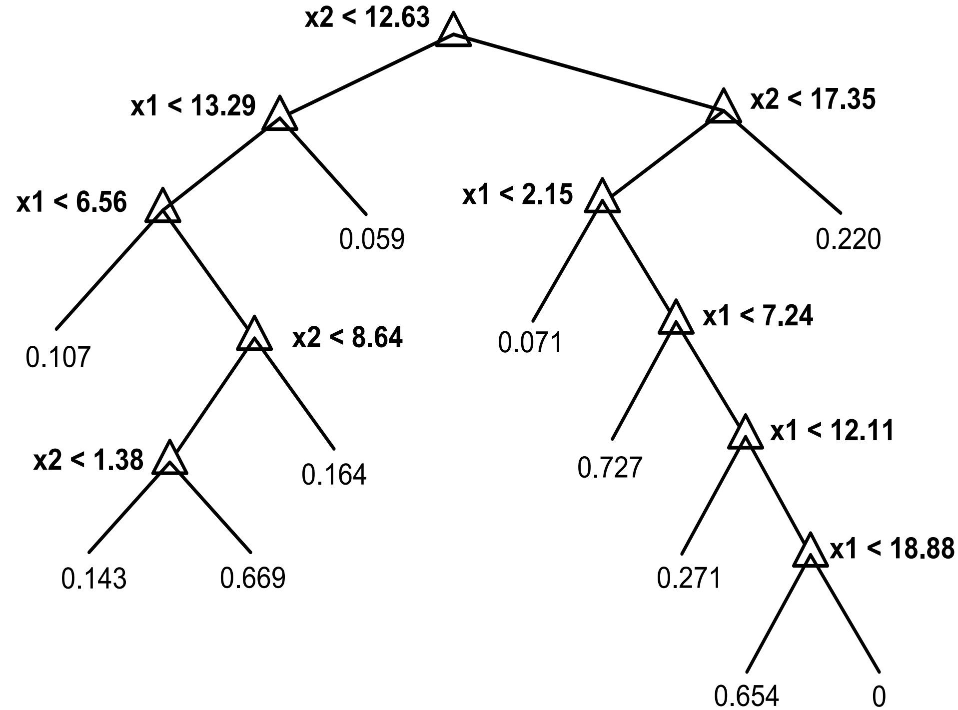
- Many classifiers produce only discrete outputs (i.e., predicted class)
  - How to get continuous-valued outputs?
    - Decision trees, rule-based classifiers, neural networks, Bayesian classifiers, k-nearest neighbors, SVM

Example: Decision Trees

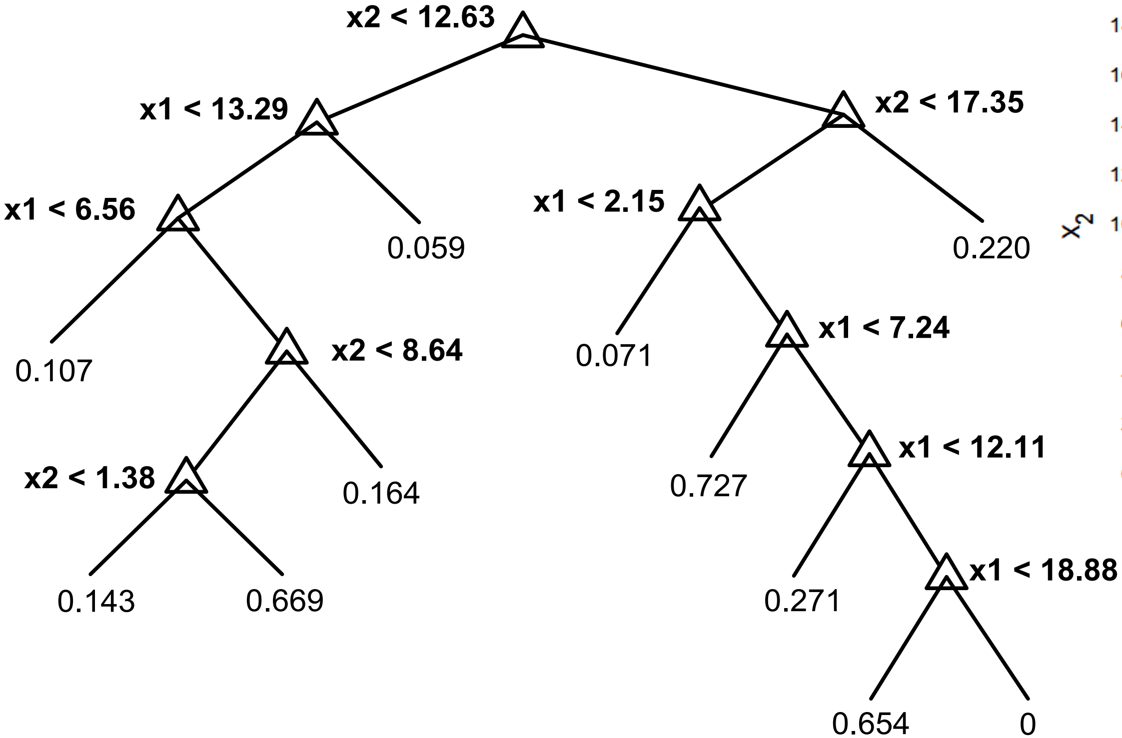
Decision Tree

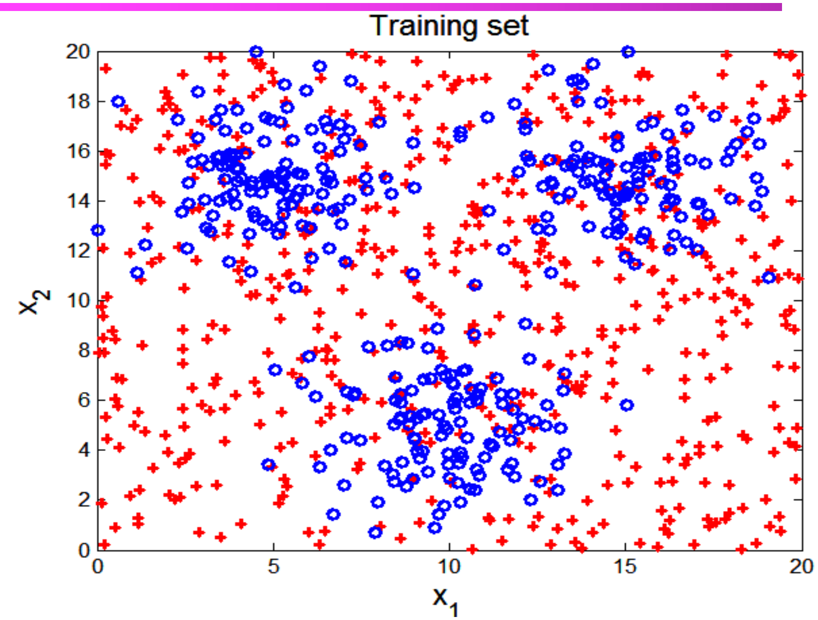


Continuous-valued outputs



ROC Curve Example

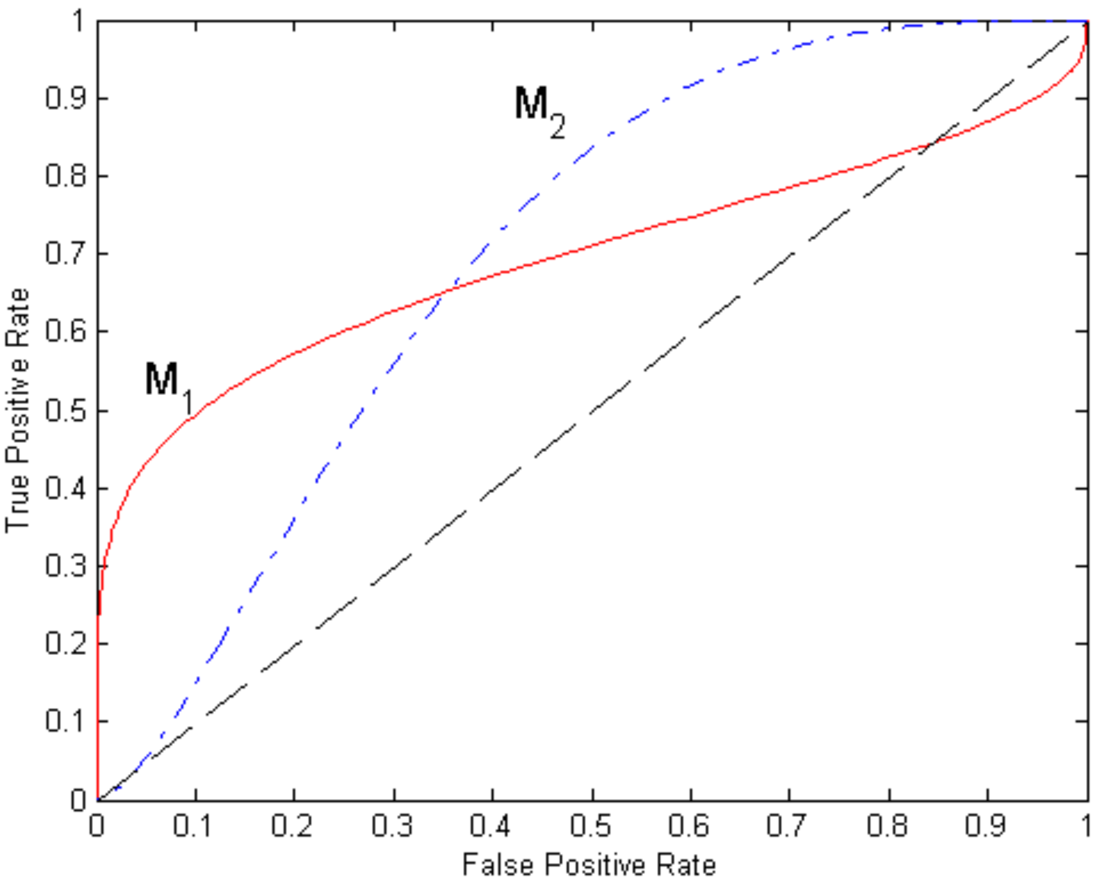




$\alpha = 0.3$		Predicted Class	
		Class o	Class +
Actual Class	Class o	645	209
	Class +	298	948

$\alpha = 0.7$		Predicted Class	
		Class o	Class +
Actual Class	Class o	181	673
	Class +	78	1168

Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

How to Construct an ROC curve

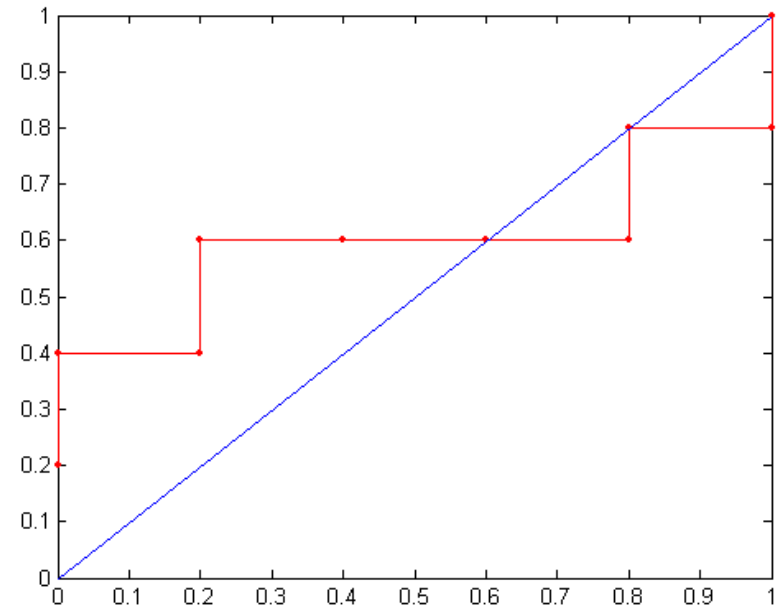
Instance	score(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces continuous-valued output for each test instance score(+|A)
- Sort the instances according to score(+|A) in decreasing order
- Apply threshold at each unique value of score(+|A)
- Count the number of TP, FP, TN, FN at each threshold
  - TPR = TP/(TP+FN)
  - FPR = FP/(FP + TN)

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:



Handling Class Imbalanced Problem

- Class-based ordering (e.g. RIPPER)
  - Rules for rare class have higher priority
- Cost-sensitive classification
  - Misclassifying rare class as majority class is more expensive than misclassifying majority as rare class
- Sampling-based approaches

Cost Matrix

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	f(Yes, Yes)	f(Yes,No)
Class=No	f(No, Yes)	f(No, No)

C(i,j): Cost of misclassifying class i example as class j

Cost Matrix	PREDICTED CLASS	
ACTUAL CLASS C(i, j)	Class=Yes	Class=No
Class=Yes	C(Yes, Yes)	C(Yes, No)
Class=No	C(No, Yes)	C(No, No)

$$\text{Cost} = \sum C(i,j) \times f(i,j)$$

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i,j)	**+**	**-**
	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		**+**	**-**
	+	150	40
	-	60	250

Accuracy = 80%

Cost = 3910

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		**+**	**-**
	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

Cost Sensitive Classification

- Example: Bayesian classifer
  - Given a test record x:
    - Compute p(i|x) for each class *i*
    - Decision rule: classify node as class k if

$$k = \arg\max_i p(i \mid x)$$

  - For 2-class, classify x as + if p(+|x) > p(-|x)
    - This decision rule implicitly assumes that C(+|+) = C(-|-) = 0 and C(+|-) = C(-|+)

Cost Sensitive Classification

- General decision rule:
  – Classify test record x as class k if

$$k = \arg\min_{j} \sum_{i} p(i \mid x) \times C(i, j)$$

- 2-class:
  – Cost(+) = p(+|x) C(+,+) + p(-|x) C(-,+)
  – Cost(-) = p(+|x) C(+,-) + p(-|x) C(-,-)
  – Decision rule: classify x as + if Cost(+) < Cost(-)
    - if C(+,+) = C(-,-) = 0:

$$p(+ \mid x) > \frac{C(-,+)}{C(-,+) + C(+,-)}$$

Sampling-based Approaches

- Modify the distribution of training data so that rare class is well-represented in training set
  - Undersample the majority class
  - Oversample the rare class

- Advantages and disadvantages

DNN vs. GMM

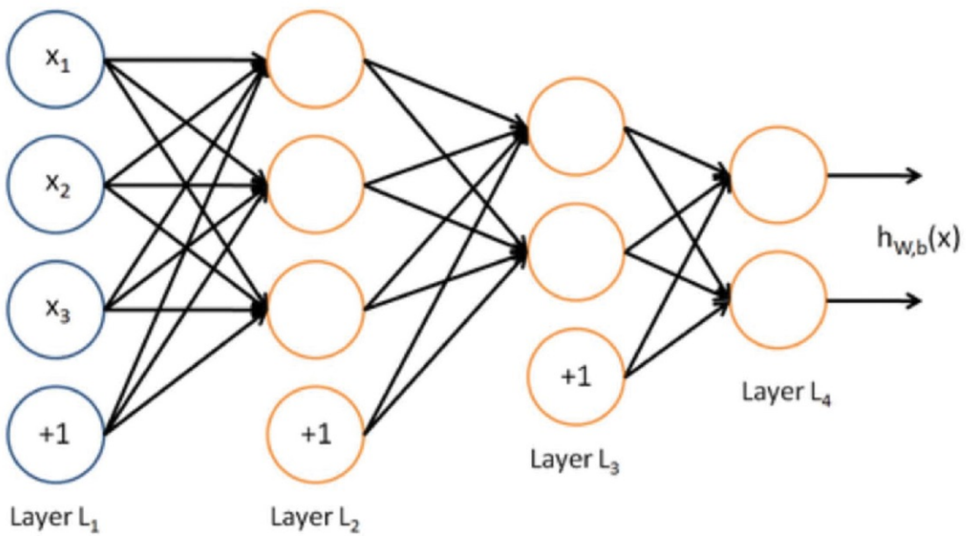
- Gaussian mixture models (GMM) for speech recognition

Task	GMM	DNN	Research Group
	WER (%)		
Switchboard	27.4	18.5	Microsoft
YouTube	52.3	47.6	Google
Broadcast News	17.2	14.9	IBM

Table 1. *Word error rate (WER) for three speech recognition tasks. GMM, Gaussian mixture models. DNN, deep neural nets. Data provided by Brian Kingsbury from IBM.*

Big Data Learning

- Deep Neural Networks
  - 5 to 7 hidden layers for a typical DNN for speech recognition, with about 1000 units per layer
  - Tens of millions of parameters to be optimized
  - Training requires up to 1000 hours, on the order of 100 million training examples, two weeks on a super machine.



Big Data Learning – Future Development

- Large Scale
- Stochasticity
- Nonlinearity
- Parallelism
- Good Generalization
- Good Interpretation if possible