Data Mining and Machine Learning

**Bioinspired computational methods**

**Biological data mining**

# Classification and Prediction

***Francesco Marcelloni***

Department of Information Engineering
University of Pisa
ITALY

Some slides belong to the collection

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign
Simon Fraser University








# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval: if a loan applicant is safe or risky
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is









# Classification—A Two-Step Process

**First Step**

- Model construction (learning step or training phase): a classifier is built describing a set of predetermined classes or concepts by exploiting a set of tuples/samples.
  - A tuple is represented by an n-dimensional attribute vector (or feature vector) $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is denoted as training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

# Classification—A Two-Step Process

Second Step

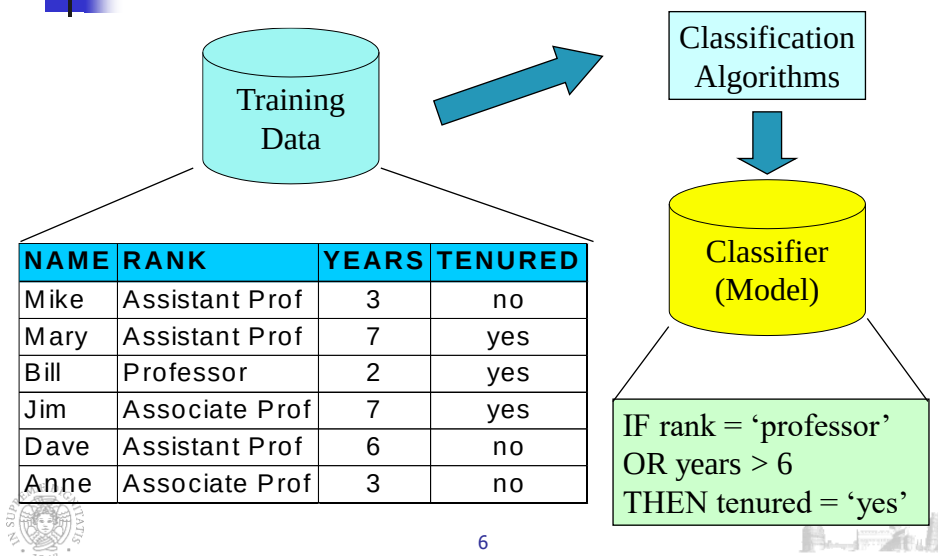
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model (valid if the dataset is not imbalanced)
    - Test set is independent of the training set
    - Why do you use a test set (overtraining)?
- If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known





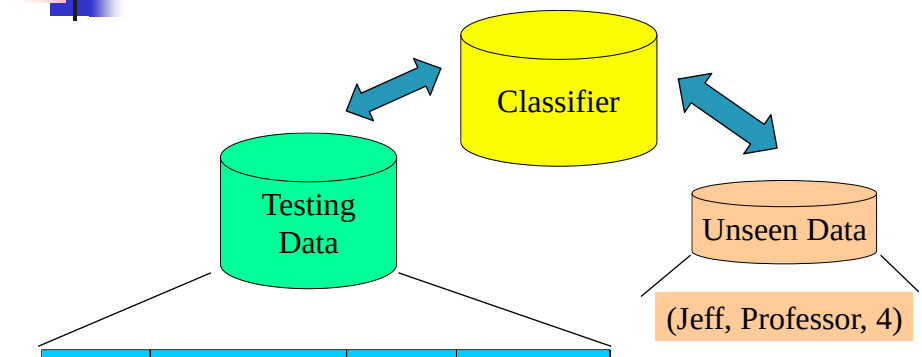


# Step 1: Model Construction




| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Mike | Assistant Prof | 3 | no |
| Mary | Assistant Prof | 7 | yes |
| Bill | Professor | 2 | yes |
| Jim | Associate Prof | 7 | yes |
| Dave | Assistant Prof | 6 | no |
| Anne | Associate Prof | 3 | no |

# Step 2: Model Usage




| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Tom | Assistant Prof | 2 | no |
| Merlisa | Associate Prof | 7 | no |
| George | Professor | 5 | yes |
| Joseph | Assistant Prof | 7 | yes |





# Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. aims to establish the existence of classes or clusters in the data

# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary









# Classification: Preparing Data

- Data cleaning: preprocesing of data for removing or reducing noise and treating missing values
- Relevance Analysis: many of the attributes may be irrelevant for the classification problem
  - Correlation Analysis
  - Attribute subset selection (or feature subset selection)
    - Ideally, the time spent on relevance analysis should be less than the time that would have been spent on learning from the original set of attributes

# Classification: Preparing Data

- Data Transformation and Reduction
  - Normalization: scaling the values for a given attribute so that they fall within a small specified range
  - Reduction
    - Generalization to higher level concepts. For instance, numeric values for attribute income can be generalized as low, medium, and high
    - several methods (wavelet transform, principle component analysis, discretization techniques such as binning, histogram analysis and clustering)







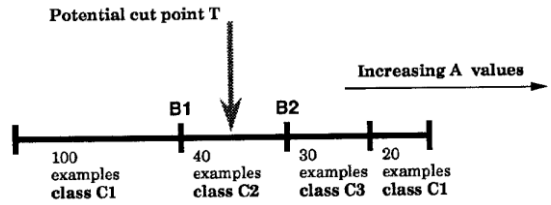


# Classification: Preparing Data

- Data Transformation and Reduction
  - Discretization: exploit ground truth for determining an optimal discretization of continuous-valued attributes
    - Some learning algorithms proposed in the literature are applicable only to categorical attributed
    - Especially when dealing with big data, performing discretization concurrently with learning can be very computationally expensive
    - Typically use of heuristic approaches based on spefixc metrics

# Classification: Data Discretization

- Discretization: The Fayyad and Irani approach (1992)
  - **Objective**: to find an optimal partition for each continuous-valued attribute A. Partitions are determined by a set of cut points.
  - **Observation**: in the original paper the authors prove that the optimal cut points for the metrics used by them lie always between two examples of different classes in the sequence of sorted values of attribute A








# Classification: Data Discretization

- Discretization: The Fayyad and Irani approach (1992)
  - **Definition**: the potential cut point T is a boundary point *b* iff in the sequence of examples sorted by the value A, there exist two examples, $e_1$ and $e_2$, having different classes, such that $A(e_1) < T < A(e_2)$ and there exists no other example e' such that $A(e_1) < A(e') < A(e_2)$. Let us assume that *b* is the midpoint value between $A(e_1)$ and $A(e_2)$
  - Let $B_A$ be the set of all candidate boundary points for attribute A.
  - The algorithm exploits the entropy as quality measure

# Classification: Data Discretization

- Discretization: The Fayyad and Irani approach (1992)
  - Let S be the set of examples. Let there be k classes $C_1$, …, $C_k$. Let $P(C_i,S)$ be the proportion of examples in S that have the class $C_i$. The class entropy of a subset S is defined as

$$\text{Ent(S)} = -\sum_{i=1}^{k} P(C_i,S)\cdot\log(P(C_i,S))$$

  - Let T be a cut point in $B_A$. Set S is partitioned in the subsets $S_1$ and $S_2$, where $S_1$ contains the subset of examples in S with A-values not exceeding T and $S_2$ = S- $S_1$.







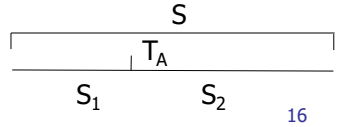


# Classification: Data Discretization

- Discretization: The Fayyad and Irani approach (1992)
  - The class information entropy of the partition induced by T, denoted as EP(A, T; S), is defined as:

$$\text{EP(A, T; S)} = \frac{|S_1|}{N}Ent(S_1) + \frac{|S_2|}{N}Ent(S_2)$$

  - The cut point $T_A$ for which EP(A, $T_A$; S) is minimal amongst all the candidate cut points is taken as the best cut point




# Classification: Data Discretization

- Discretization: The Fayyad and Irani approach (1992)
  - The algorithm first partition S, and then S=$S_1$ and S=$S_2$ until a stopping condition is met.
  - The stopping condition is defined as:

$$G(A, T_A; S) > \frac{log_2(N-1)}{N} + \frac{\Delta(A,TA;S)}{N}$$

where $\Delta(A, TA; S) = \log_2(3^c) - c_1 E(S_1) - c_2 E(S_2))$, c, $c_1$ and $c_2$ are, respectively, the numbers of classes in S, $S_1$ and $S_2$, and $G(A, T_A; S)=E(S) - EP(A, T_A, S)$









# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Comparing Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules









# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Classification by Using Frequent Patterns
- Rule-Based Classification
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Decision Tree Induction

- Decision Tree Induction: learning of decision trees from class-labeled training tuples
- Decision Tree: flowchart-like tree structure, where
  - each internal node (nonleaf node) denotes a test on an attribute,
  - each branch represents an outcome of the test
  - each leaf node (terminal node) holds a class label.









# Decision Tree Induction

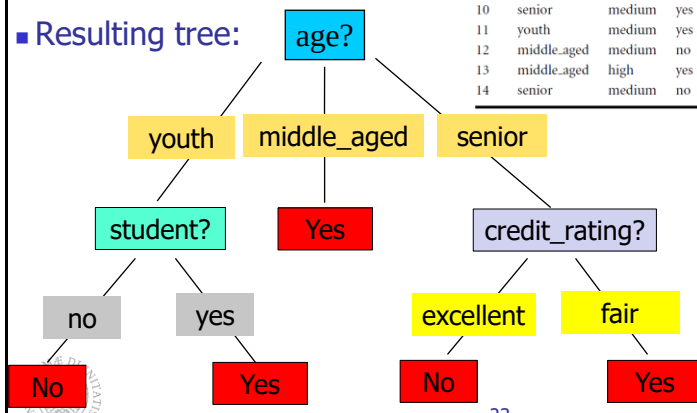
- How decision tree are used for classification?
  - Given a tuple, the attribute values of the tuple are tested against the decision tree.
- Why are decision tree classifiers so popular?
  - No domain knowledge
  - No parameter setting
  - Can manage high-dimensional data
  - Representation intuitive and easy to assimilate

# Decision Tree Induction: An example

- Training data set:

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

- Resulting tree:










# Algorithm for Decision Tree Induction

- Basic algorithm is a greedy algorithm - follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding the global optimum.
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - **Attributes are categorical** (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

# Algorithm for Decision Tree Induction

- Conditions for stopping partitioning
  - All samples for a given node belong to the **same class**
  - There are **no remaining attributes** for further partitioning – majority voting is employed for classifying the leaf
  - There are **no samples left**
- Strategy for selecting the attributes: to create partitions at each branch **as pure as possible**.
  - A partition is pure whether all the tuples in it belong to the same class









# Attribute Selection Measure: Information Gain (ID3/C4.5)

Let D be a training set of class-labeled tuples

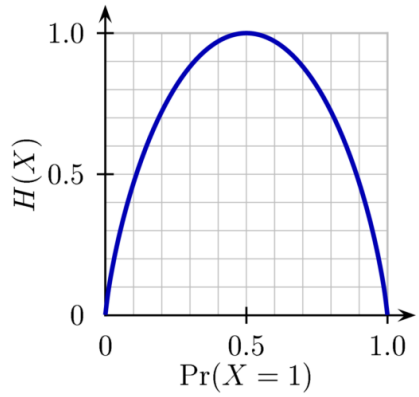
Suppose that the class label attribute has *m* distinct values defining *m* distinct classes, $C_i$ (for i=1,..,*m*).

- Select the attribute with the highest information gain
- Let $p_i$ be the probability that an arbitrary tuple in D belongs to class $C_i$, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Observe: $p_i$ is the probability of class i in D

# Attribute Selection Measure: Information Gain (ID3/C4.5)










# Attribute Selection Measure: Information Gain (ID3/C4.5)

Suppose that we split D into $v$ partitions $\{D_1, \ldots, D_v\}$ on some attribute A having $v$ distinct values $\{a_1,\ldots,a_v\}$

- Information needed (after using A to split D into v partitions) to classify D:

Cardinality of the set of objects in D with value for A equal to $a_j$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

where $D_j$ contains all the tuples in D with value $a_j$ for A

$$Gain(A) = Info(D) - Info_A(D)$$

- Information gained by branching on attribute A

# Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

| age | $p_i$ | $n_i$ | $I(p_i,n_i)$ |
|---|---|---|---|
| young | 2 | 3 | 0.971 |
| middle_aged | 4 | 0 | 0 |
| senior | 3 | 2 | 0.971 |

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

$\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

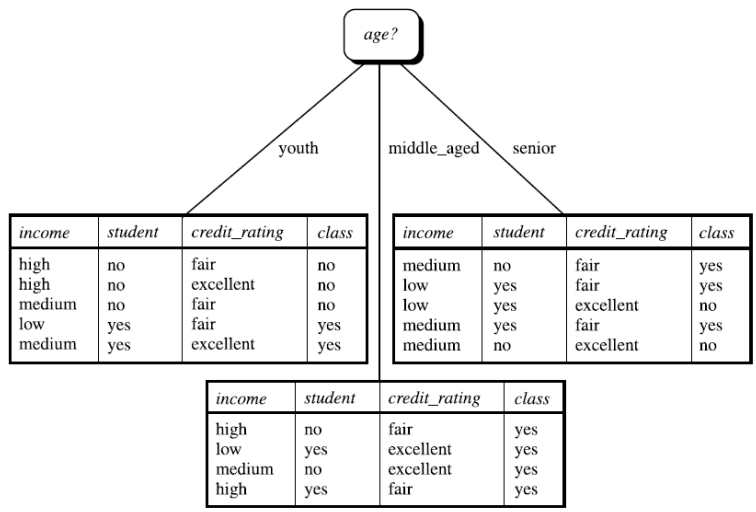
$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$







# First-level Decision Tree

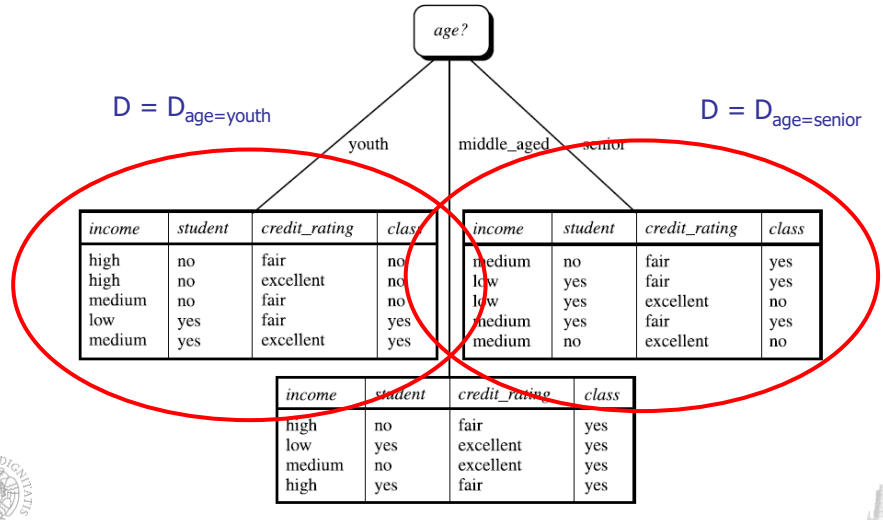



| income | student | credit_rating | class |
|---|---|---|---|
| high | no | fair | no |
| high | no | excellent | no |
| medium | no | fair | no |
| low | yes | fair | yes |
| medium | yes | excellent | yes |

| income | student | credit_rating | class |
|---|---|---|---|
| medium | no | fair | yes |
| low | yes | fair | yes |
| low | yes | excellent | no |
| medium | yes | fair | yes |
| medium | no | excellent | no |

| income | student | credit_rating | class |
|---|---|---|---|
| high | no | fair | yes |
| low | yes | excellent | yes |
| medium | no | excellent | yes |
| high | yes | fair | yes |

# First-level Decision Tree








# Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the best split point for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible split point

    $(a_i+a_{i+1})/2$ is the midpoint between the values of $a_i$ and $a_{i+1}$
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - $D_1$ is the set of tuples in D satisfying A ≤ split-point, and $D_2$ is the set of tuples in D satisfying A > split-point
- Alternative: Discretization before applying the decision tree learning

# Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values (for instance, attributes that act as unique identifier -> pure partition -> $Info_{product\_ID}(D)=0$)
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem
- Adopts a normalization to information gain using a "split information"

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- The value represents the potential information generated by splitting the training data set, D, into v partitions, corresponding to the v outcomes of a test on attribute A.









# Gain Ratio for Attribute Selection (C4.5)

- GainRatio(A) = Gain(A)/SplitInfo(A)

  Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557.$$

  GainRatio(income) = 0.029/1.557 = 0.019

- The attribute with the maximum gain ratio is selected as the splitting attribute

# Gini Index (CART, IBM IntelligentMiner)

- If a data set D contains examples from *m* classes, the Gini index, gini(D), is defined as

$$gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

where $p_i$ is the relative frequency of class i in D

- The Gini index considers a binary split for each attribute.
- If **A is a discrete variable** with *v* distinct values $\{a_1,\ldots,a_v\}$, all possible subsets that can be formed from A are examined to determine the optimal binary split
- For example, considering three possible values {low, medium, high}, the possible subsets are: {low, medium}, {low, high}, {medium, high}, {low}, {medium} {high}



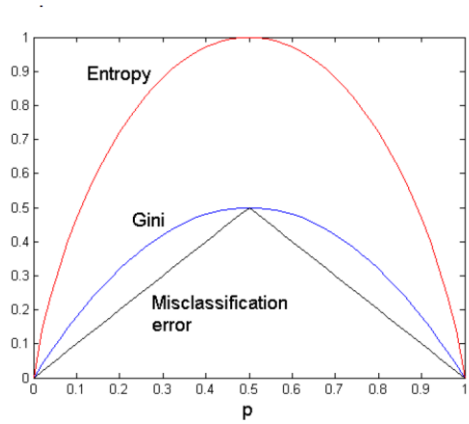






# Gini Index (CART, IBM IntelligentMiner)

- For a 2-class problem




# Gini Index (CART, IBM IntelligentMiner)

- If a data set D is split on A into two subsets $D_1$ and $D_2$, the gini index $gini_A(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

  - For a discrete attribute, each of the possible binary splits is considered
  - For a continuous attribute, each possible split-point must be considered

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute that maximizes the reduction of impurity (or equivalently has the minimum Gini index) is selected as the splitting attribute.







# An example of computationof the Gini index

Example

- D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Let's start with the attribute income and consider each of the possible splitting subsets. Consider the subset {low, medium}. This would result in 10 tuples in $D_1$ and 4 in $D_2$

$$\begin{aligned} &Gini_{income \in \{low,medium\}}(D) \\ &= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\ &= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\ &= 0.443 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

# An example of computationof the Gini index

- The Gini values for splits on the remaining subsets are: Gini{low, high} = Gini{medium} 0.458; Gini{medium, high} = Gini{low} = 0.450.
- Thus, split on the {low, medium} (and {high}) since it has the lowest Gini index
- The Gini index may need other tools, e.g., clustering, to get the possible split values









# Comparing Attribute Selection Measures

The three measures, in general, return good results but

- Information gain:
  - biased towards multivalued attributes
- Gain ratio:
  - tends to prefer unbalanced splits in which one partition is much smaller than the others
- Gini index:
  - biased to multivalued attributes
  - has difficulty when the number of classes is large
  - Tends to favor tests that result in equal-sized partitions and purity in both partitions

# Decision Tree Scheme

**Algorithm: Generate_decision_tree.** Generate a decision tree from the training tuples of data partition, *D*.

**Input:**

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

**Output:** A decision tree.









# Decision Tree Scheme

**Method:**

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
```

# Decision Tree Scheme

```
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)    let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)    if Dj is empty then
(13)       attach a leaf labeled with the majority class in D to node N;
(14)    else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```







# Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ2 test for independence
- C-SEP: performs better than information gain and the Gini index in certain cases
- G-statistic: has a close approximation to χ2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear combinations of attributes.

# Other Attribute Selection Measures

- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others
  - All measures have some bias
  - However, time complexity of decision tree induction increases exponentially with the tree height
    - Thus, measures which tend to produce shallower trees may be preferred
  - On the other hand, shallow trees tend to have a large number of leaves and higher error rates





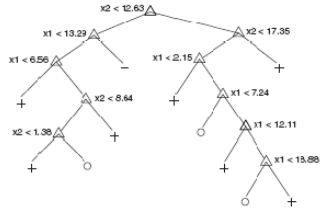




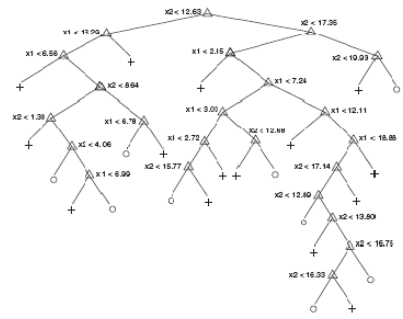
# Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples

# Overfitting and Tree Pruning




(a) Decision tree with 11 leaf nodes.



(b) Decision tree with 24 leaf nodes.

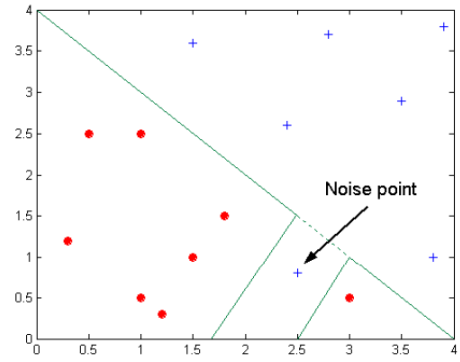


47



# Overfitting and Tree Pruning

- Decision boundary is distorted by noise points



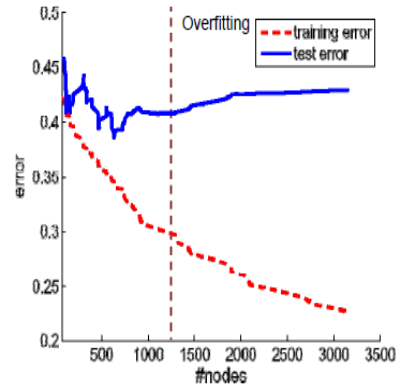

Decision boundary is distorted by noise point



48

# Overfitting and Tree Pruning

- How can we realize that the tree is affected by overfitting?
  1. Learn the decision model by using the training set
  2. Compute the accuracy ($A_{Training}$) of the model by classifying the training set
  3. Compute the accuracy ($A_{Test}$) of the model by classifying the test set
  - If $A_{Test} << A_{Training}$, then overtraining







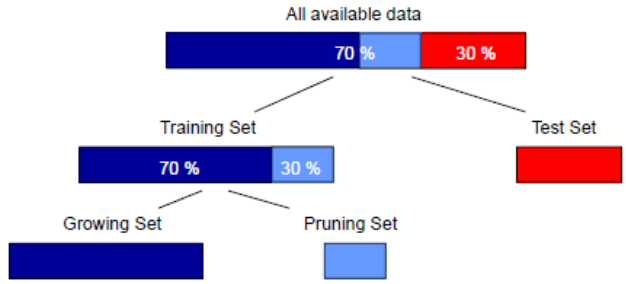



# Estimating Generilization Errors

- Re-substitution errors: error on training set ($e_{TR}$)
- Generalization errors: error on test set ($e_{TS}$)
- Methods for estimating generalization errors:
  - Optimistic approach: $e_{TS} = e_{TR}$
  - Pessimistic approach:
    - For each leaf node $e_{TS} = e_{TR} + 0.5$
    - Total errors: $e_{TS} = e_{TR} + 0.5 \times N$ ($N$: Number of leaf nodes)
    - For a tree with 30 leaf nodes and 10 errors on training
      - $e_{TR} = 10/1000 = 1\%$
      - $e_{TS} = (10 + 30 \times 0.5)/1000 = 2.5\%$
  - Reduced error pruning (REP)
    - Use a pruning data set to estimate generalization error

# Estimating Generilization Errors

- Pruning or validation data set










# Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex.
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

# Overfitting and Tree Pruning

- Two approaches to reduce overfitting
  - Prepruning: Halt tree construction early
    - Typical stopping conditions for a node:
      - Stop is all the instances belong to the same class
      - Stop if all the attribute values are the same
    - More restrictive conditions
      - Stop if the number of instances is less than some user-specified threshold
      - Stop if class distribution of instances are independent of the available features (e.g., using Chi-square test)
      - Stop if expanding the current node does not improve impurity measures (e.g., information gain or Gini index)
    - Difficult to choose appropriate thresholds







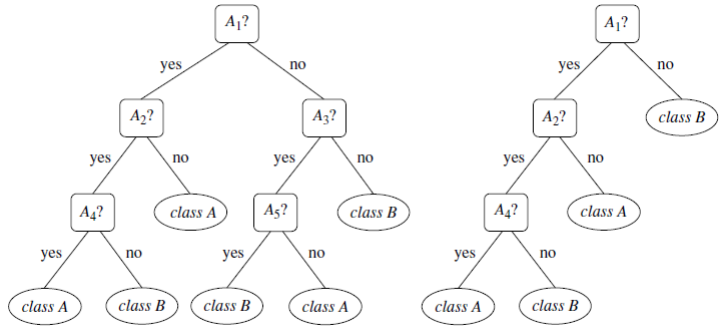


# Overfitting and Tree Pruning

- Postpruning: Remove branches from a "fully grown" tree — get a sequence of progressively pruned trees
  - **Removes subtrees from a "fully grown" tree**. A subtree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the subtree being replaced.
  - Use a set of data different from the training data to decide which is the "best pruned tree"

# Overfitting and Tree Pruning

- Postpruning: Example



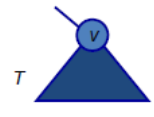







# Reduced Error Pruning (REP)

- Use pruning set to estimate accuracy of sub-trees and accuracy of individual nodes
- Let T be a sub-tree rooted at node v



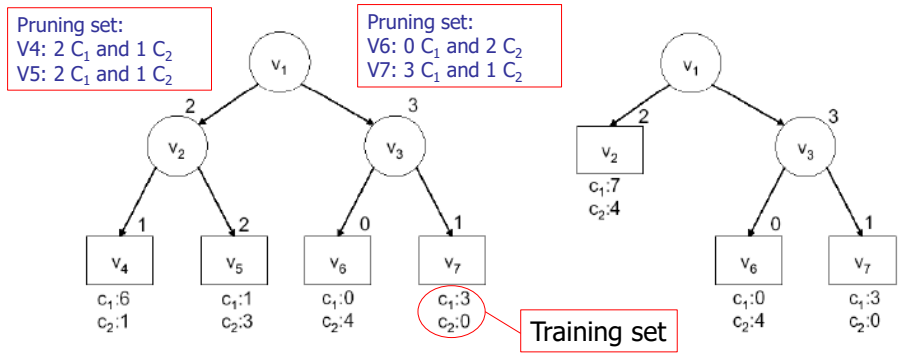

- The gain from pruning at v can be defined as

  Gain from pruning =

  #misclassifications in T - #misclassifications at v
- Repeat: prune at node with the largest gain until only negative gain nodes remain
- "Bottom/up restriction": T can only be pruned if it does not contain a sub-tree with lower error than T

# Reduced Error Pruning (REP)

- An example




$$E(T_{v_2}) = 3, E(v_2) = 2, E(T_{v_3}) = 1, E(v_3) = 3.$$







# Postpruning: cost complexity

- Cost complexity pruning algorithm in CART
  - Cost complexity: function of the number of leaves in the tree and the resubstitution error of the tree

Cost complexity =

Resubstitution Error + ß · Number of leaf nodes

where the Resubstitution error is the misclassification rate computed on the training set and ß is a penalty per additional terminal nodes

# Postpruning: cost complexity

- Search for the right-sized tree :
  - prune or collapse some of the branches of the largest tree from the bottom up, using the cost complexity parameter, and cross-validation or an independent test sample to measure the predictive accuracy of the pruned tree.
  - use the resubstitution cost for ranking the subtrees and generating a tree sequence table ordered from the most complex tree at the top to a less complex tree at the bottom
  - Identify the minimum-cost tree and pick an optimal tree as the tree within one standard error of the minimum cost tree
    - An optimal tree should be the one with the smallest terminal nodes among those that lie within one standard error of the minimum-cost tree.







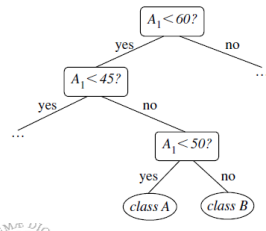


# Postpruning in C4.5

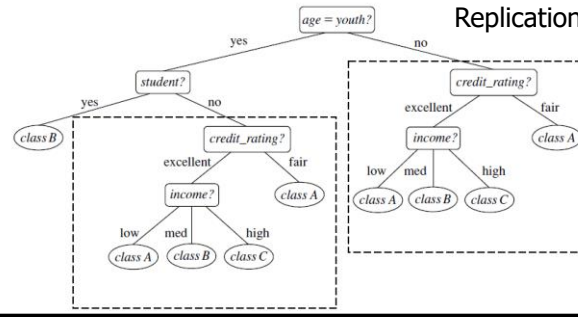
- Pessimistic pruning
  - **Similar to the cost complexity pruning** but does not use a pruning set.
  - It adjusts the error rates obtained by the training set **by adding a penalty**, computed by adopting a heuristic approach based on statistical theory. If the error rate in the node is lower than the error rate in the subtree originated from the node, then the subtree is pruned.

# Postpruning

- Pruned trees tend to be more compact than their unpruned counterparts, but can still suffer from
  - Repetition – an attribute is repeatedly tested along a given branch
  - Replication – duplicate subtrees exist within the tree




Repetition




Replication





# Postpruning

- Repetition and Replication can impede the accuracy and comprehensibility of a decision tree.
- The use of multivariate splits (splits based on a combination of attributes) can prevent these problems.
- Another approach is to use a different form of knowledge representation, such as rules, instead of decision trees.
- Indeed rule-based classifier can be constructed by extracting IF-THEN rules from a decision tree.

# Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication









# Classification in Large Databases

- Classification - a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
- Note: if the training set does not fit in memory, decision tree construction becomes inefficient due to swapping of the training tuples in and out

# Scalability Framework for RainForest

- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Use of special data structures: AVC-set
- Adapts to the amount of main memory available and applies to any decision tree induction algorithm.
- AVC-set (of an attribute *X*)
  - AVC (Attribute, Value, Class_label)
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- The method maintains an AVC-set for each attribute at each tree node, describing the training tuples at node.
- AVC-group (of a node *n*)
  - Set of AVC-sets of all attributes at the node *n*
- Dimension of an AVC-set depends on number of distinct values of A and the number of classes







# Rainforest: Training Set and Its AVC Sets

## Training Examples

| RID | age | income | student | credit_rating | Class |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

AVC-set on *Age*

| Age | Buy_Computer | |
|---|---|---|
| | yes | no |
| youth | 2 | 3 |
| middle | 4 | 0 |
| senior | 3 | 2 |

AVC-set on *income*

| income | Buy_Computer | |
|---|---|---|
| | yes | no |
| high | 2 | 2 |
| medium | 4 | 2 |
| low | 3 | 1 |

AVC-set on *Student*

| student | Buy_Computer | |
|---|---|---|
| | yes | no |
| yes | 6 | 1 |
| no | 3 | 4 |

AVC-set on *credit_rating*

| Credit rating | Buy_Computer | |
|---|---|---|
| | yes | no |
| fair | 6 | 2 |
| excellent | 3 | 3 |

# BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called bootstrapping to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree T'
- It turns out that T' is very close to the tree that would be generated using the whole data set together
- Advantages
  - Requires only two scans of DB
    - Makes one scan over the training database while collecting a small subset of the training database in-memory
    - Compute the final splitting criteria in only one scan over the training database
  - An incremental algorithm: can be used for incremental updates









# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Bayesian Classification: Why?

- A statistical classifier: performs probabilistic prediction, i.e., predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured









# Bayesian Theorem: Basics

- Let X be a data tuple ("evidence"): class label is unknown
- Let H be a hypothesis that X belongs to class C
- Classification consists of determining
- P(H|**X**) (posteriori probability): the probability that the hypothesis holds given the observed data sample X
  - Let us suppose that H is the hypothesis that a customer will buy a computer and a customer is characterised by age and income.
  - Then, P(H|**X**) is the probability that X will buy a computer given his age and income. For instance, X is a 35-year-old customer with an income of $40,000
- P(H) (prior probability of H): the initial probability
  - P(H) is the probability that a generic customer will buy a computer, regardless of age, income, …

# Bayesian Theorem: Basics

- P(X) (prior probability of X): probability that sample data is observed
  - Probability that a customer is 35 years old and earns $40,000.
- P(**X**|H) (posteriori probability of **X** conditioned on H - likelyhood): the probability of observing the sample X, given that the hypothesis holds
  - Given that X will buy computer, the prob. that X is 31..40, medium income
- Note: P(H), P(X) and P(**X**|H) can be estimated from the given data.
- P(H|**X**) can be computed by the Bayes' theorem









# Bayes' theorem

- Given training data X,

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts X belongs to $C_i$ iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the k classes
- Practical difficulty: requires initial knowledge of many probabilities, significant computational cost

# Towards Naïve Bayesian Classifier

- Let $D$ be a training set of tuples and their associated class labels, and each tuple is represented by an n-dimensional attribute vector $X = (x_1, x_2, \ldots, x_n)$
- Suppose there are $m$ classes $C_1, C_2, \ldots, C_m$.
- Classification objective: to determine the class having the highest posterior probability, i.e., the maximal $P(C_i|X)$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since $P(\mathbf{X})$ is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Can be estimated by $|C_{i,D}|/|D|$







# Towards Naïve Bayesian Classifier

- Assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost:
  - Only counts the class distribution
- If $A_k$ is categorical, $P(x_k|C_i)$ is the number of tuples in $C_i$ having value $x_k$ for $A_k$ divided by $|C_i, D|$ (number of tuples of $C_i$ in D)

# Towards Naïve Bayesian Classifier

- If $A_k$ is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $$P(x_k \mid C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

where $\mu_{Ci}$ and $\sigma_{Ci}$ are the mean value and the standard deviation, respectively, of the values of attribute $A_k$ for training tuples of class $C_i$.







# Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age =youth,
Income = medium,
Student = yes
Credit_rating = Fair)

| RID | age | income | student | credit_rating | Class |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Naïve Bayesian Classifier: An Example

P(Ci): P(buys_computer = "yes") = 9/14 = 0.643
P(buys_computer = "no") = 5/14= 0.357

Compute P(X|Ci) for each class

P(age = "youth" | buys_computer = "yes") = 2/9 = 0.222
P(age = "youth" | buys_computer = "no") = 3/5 = 0.6
P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

**X = (age = youth, income = medium, student = yes, credit_rating = fair)**

P(X|Ci) : P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

P(X|Ci)*P(Ci) : P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")







# Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case
    Prob(income = low) = 1/1003=0.001
    Prob(income = medium) = 991/1003=0.988
    Prob(income = high) = 11/1003=0.011
- The "corrected" prob. estimates are close to their "uncorrected" counterparts

# Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks





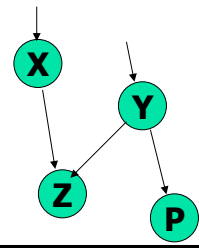


# Bayesian Belief Networks

- Bayesian belief networks (also known as Bayesian networks, probabilistic networks):
  - allow the representation of dependencies among subsets of attributes (both discrete- and continuous-valued)

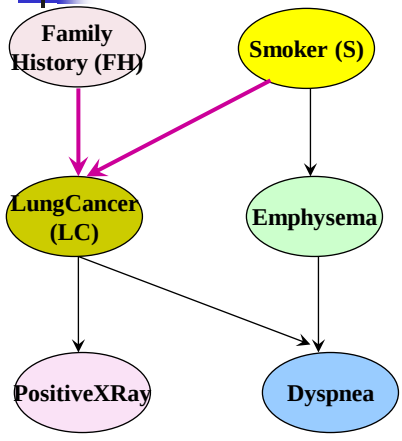
Nodes may correspond to actual attributes given in the data or to "hidden variables" believed to form a relationship (e.g., in the case of medical data, a hidden variable may indicate a syndrome, representing a number of symptoms that, together, characterize a specific disease).

- Represents dependency among the variables
- Gives a specification of joint probability distribution




- ❑ Nodes: random variables
- ❑ Links: dependency
- ❑ X and Y are the parents of Z, and Y is the parent of P
- ❑ No dependency between Z and P
- ❑ Has no loops/cycles

# Bayesian Belief Network: An Example




CPT: Conditional Probability Table for variable LungCancer:

| | (FH, S) | (FH, ~S) | (~FH, S) | (~FH, ~S) |
|---|---|---|---|---|
| LC | 0.8 | 0.5 | 0.7 | 0.1 |
| ~LC | 0.2 | 0.5 | 0.3 | 0.9 |

shows the conditional probability for each possible combination of the values of its parents





# Training Bayesian Networks: Several Scenarios

- Let $X=(x_1,..,x_n)$ be a data tuple described by the variables or attributes $X_1$, …, $X_n$, respectively.
  - Note that each variable is conditionally independent of its nondescendants in the network graph, given its parents.
  - A complete representation of the existing joint probability distribution can be obtained by the following equation

$$P(x_1,\ldots,x_n)=\prod_{i=1}^{n}P(x_i|Parents(Y_i))$$

# Training Bayesian Networks: Several Scenarios

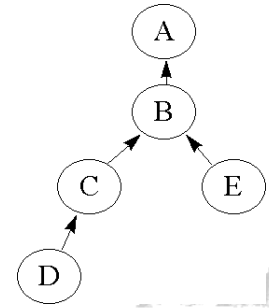
Just an example.

- Suppose that we have five variables A, B, C, D and E.
- If we do not specify the dependencies explicitly, then all the variables are assumed to be dependent on each other.

  p(A,B,C,D,E) =
  p(A|B,C,D,E)*p(B|C,D,E)*p(C|D,E)*p(D|E)*P(E)

- If the dependencies are explicitly modeled

- p(A,B,C,D,E) =
  p(A|B)*p(B|C,E)*p(C|D)*p(D)*p(E)




84



# Training Bayesian Networks: Several Scenarios

- A node within the network can be selected as an "output" node, representing a class label attribute.
- There may be more than one output node.
- Rather than returning a single class label, the classification process can return a probability distribution that gives the probability of each class.
- Belief networks can be used to answer probability of evidence queries (e.g., what is the probability that an individual will have LungCancer, given that they have both PositiveXRay and Dyspnea) and most probable explanation queries (e.g., which group of the population is most likely to have both PositiveXRay and Dyspnea).

85

# Training Bayesian Networks: Several Scenarios

- Scenario 1: Given both the network structure and all variables observable: compute only the CPT entries
- Scenario 2: Network structure known, some variables hidden: gradient descent (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights (CPT entries) are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, without backtracking
  - Weights are updated at each iteration and converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to reconstruct network topology
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. A Tutorial on Learning with Bayesian Networks. In Learning in Graphical Models, M. Jordan, ed.. MIT Press, 1999.



86



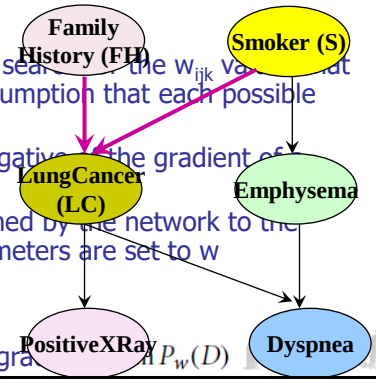
# Training Bayesian Networks: Several Scenarios

Scenario 2:

- Let $w_{ijk}$ be a CPT entry for the variable $Y_i=y_{ij}$ having the parents $U_i=u_{ik}$, where $w_{ijk}=P(Y_i=y_{ij}|U_i=u_{ik})$.
- If $w_{ijk}$ is the upper leftmost CPT entry in the previous example, then $Y_i$ is LungCancer, $y_{ij}$ is its value "yes", $U_i$ lists the parent nodes of $Y_i$, namely {FamilyHistory, Smoker} and $u_{ik}$ list the values of the parent nodes, namely {"yes", "yes"}
- A gradient descent strategy is used to search among the $w_{ijk}$ values that best model the data, based on the assumption that each possible setting of $w_{ijk}$ is equally likely.
- It searches for a solution along the negative of the gradient of a criterion function
  - We maximize the probability assigned by the network to the observed data when the CPT parameters are set to w

$$P_w(D) = \prod_{d=1}^{|D|} P_w(X_d)$$

  - This can be done by following the gradient of $P_w(D)$




87

# Training Bayesian Networks: Several Scenarios

The algorithm proceeds as follows:

- Compute the gradients (for each training tuple $\boldsymbol{X}_d$)

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}} = \sum_{d=1}^{|D|} \frac{P(Y_i = y_{ij}, U_i = u_{ik} | \boldsymbol{X}_d)}{w_{ijk}}.$$

- Take a small step in the direction of the gradient (where $l$ is the learning rate)

$$w_{ijk} \leftarrow w_{ijk} + (l) \frac{\partial \ln P_w(D)}{\partial w_{ijk}}.$$

- Renormalize the weights (all the weights have to be between 0 and 1 being probabilities and $\sum_j w_{ijk}$ must equal 1 for all $i,k$.)







# Classification: Basic Concepts


- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary


# Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules
  *R: IF age = youth AND student = yes THEN buys_computer = yes*
  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: coverage and accuracy
  - $n_{covers}$ = number of tuples covered by rule R
  - $n_{correct}$ = number of tuples correctly classified by rule R
  - coverage(R) = $n_{covers}$ /|D| /* D: training data set */
  - accuracy(R) = $n_{correct}$ / $n_{covers}$
- If more than one rule are triggered, need conflict resolution







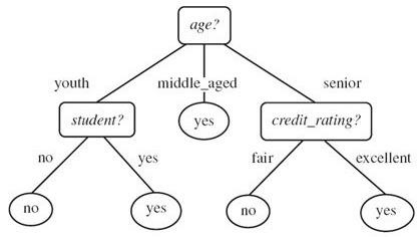
# Using IF-THEN Rules for Classification: conflict resolution

Possible conflict resolution strategies

- Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the most attribute tests (rule antecedent size)
- Class-based ordering: decreasing order of prevalence (rules for the most frequent class come first) or misclassification cost per class (rules for the class with the highest cost come first)
- Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality (accuracy, coverage or size) or based on advice from domain experts.
  - Each rule in a decision list implies the negation of the rules that come before it in the list (difficult to interpret)

# Rule Extraction from a Decision Tree

- How can we mine rules?
  1. Rule extraction from a decision tree




IF age is youth AND student is no THEN buy_computer is no

IF age is youth AND student is yes THEN buys_computer is no

⋮

  2. Heuristic approaches



92



# Rule Extraction from a Decision Tree

Rule extraction from a decision tree

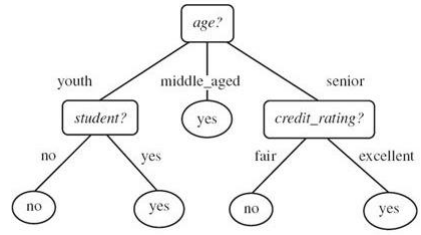
- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive (no rule conflict) and exhaustive (one rule for each possible attribute-value)

- Note: one rule per leaf! With decision trees which suffer from repetition and replication, the extracted rule base can be difficult to interpret



93

# Rule Extraction from a Decision Tree




- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = youth AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = youth AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = middle-aged THEN *buys_computer* = *yes*
IF *age* = senior AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = senior AND *credit_rating* = *fair* THEN *buys_computer* = *yes*







# Rule Extraction from a Decision Tree How can we prune the rule set

- Each condition which does not improve the estimated accuracy of the rule can be pruned.
  - C4.5 uses a pessimistic approach for counteracting the bias generated by using the training set.
- Further, any rule which does not contribute to the overall accuracy is pruned
- After pruning, the rules will no longer be mutually exclusive and exhaustive.
  - C4.5 adopts a class-based ordering scheme: groups the rules per class and then determines a ranking of these class rule sets so as to minimize the number of false-positive errors (the rule predicts a class C, but the actual class is not C).
  - The class rule set with the least number of false positives is examined first.
  - **Default class**: class which contains the highest number of tuples not covered by any rule (the majority class will likely have many rules for its tuples).

# Rule Induction: Sequential Covering Method

Heuristic approaches

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned sequentially, each for a given class $C_i$ will cover many tuples of $C_i$ but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, **the tuples covered by the rules are removed**
  - The process repeats on the remaining tuples unless termination condition, e.g., when **no more training examples or when the quality of a rule returned is below a user-specified threshold**
- Comparison with decision-tree induction: decision-tree induction learns a set of rules simultaneously

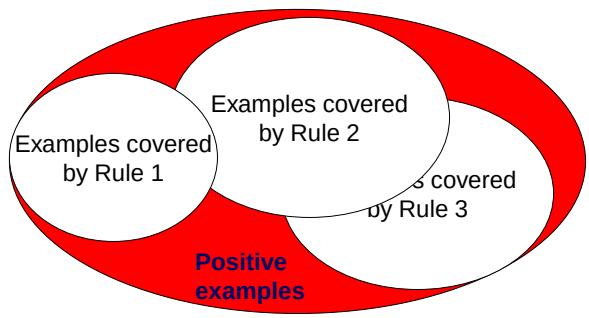






# Sequential Covering Algorithm

**while** (enough target tuples left)
  generate a rule
  remove positive target tuples satisfying this rule



# Sequential Covering Algorithm

## Formally

**Algorithm: Sequential covering.** Learn a set of IF-THEN rules for classification.

**Input:**

- *D*, a data set of class-labeled tuples;
- *Att_vals*, the set of all attributes and their possible values.

**Output:** A set of IF-THEN rules.

**Method:**

```
(1)  Rule_set = {}; // initial set of rules learned is empty
(2)  for each class c do
(3)      repeat
(4)          Rule = Learn_One_Rule(D, Att_vals, c);
(5)          remove tuples covered by Rule from D;
(6)          Rule_set = Rule_set + Rule; // add new rule to rule set
(7)      until terminating condition;
(8)  endfor
(9)  return Rule_Set;
```





# Rule Generation

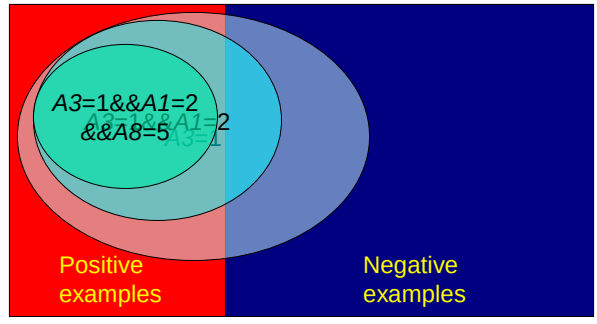
To generate a rule

**while**(true)

find the best predicate p

**if** foil-gain(p) > threshold **then** add p to current rule

**else** break



# How to Learn-One-Rule?

Greedy depth-first strategy

- Start with the most general rule possible: condition = empty
- Suppose our training set, D, consists of loan application data. Attributes regarding each applicant include their age, income, education level, residence, credit rating, and the term of the loan.
- Start with the rule

IF THEN *loan_decision* = *accept*.

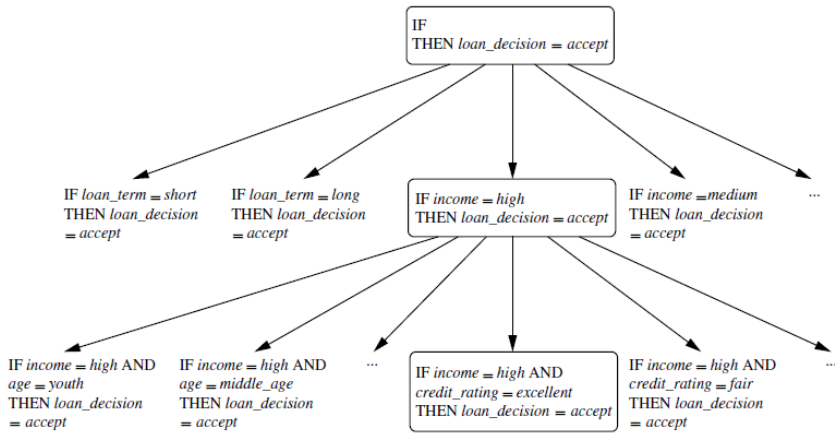
- Then, we consider each possible attribute that may be added to the rule



100



# How to Learn-One-Rule?



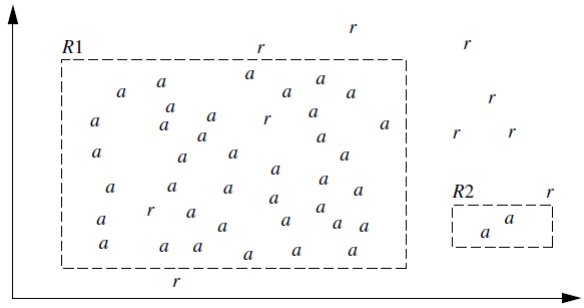



101

# How to Learn-One-Rule?

- Rule-Quality measures: Accuracy?

a = accept
r = reject




- R1 = 95% accuracy R2 = 100% accuracy
  but R2 very small coverage







# How to Learn-One-Rule?

- Rule-Quality measures:
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition (FOIL = First Order Inductive Learner)

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

where pos (pos') and neg (neg') are the numbers of positive and negative tuples covered by current rule *R (new rule R')*

  - Foil-gain favors rules that have high accuracy and cover many positive tuples

# How to Learn-One-Rule?

- Learn_One_Rule does not employ a test set when evaluating rules. Thus, the evaluation is optimistic.
- Rule pruning based on an independent set of test tuples
- Pruning is carried out by removing a conjunct
- FOIL uses a simple yet effective method:

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

If FOIL_Prune is higher for the pruned version of R, prune R

- By convention, RIPPER starts with the most recently added conjunt when considering pruning.
- Conjunts are pruned one at a time as long as this results in an improvement









# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Associative Classification

- Associative classification: Major steps
  - Mine data to find strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Association rules are generated in the form of
    $p_1 \wedge p_2 \ldots \wedge p_l \Rightarrow$ "$A_{class} = C$" (conf, sup)
  - Organize the rules to form a rule-based classifier
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - Associative classification has been found to be often more accurate than some traditional classification methods, such as C4.5









# Typical Associative Classification Methods

- CBA (Classification Based on Associations: Liu, Hsu & Ma, KDD'98)
  - Mine possible association rules in the form of
    - Cond-set (a set of attribute-value pairs) => class label
  - Build the classifier: heuristic method, where the rules are organized according to decreasing precedence based on confidence and then support
    - If a set of rules has the same antecedent, then the rule with the highest confidence is selected to represent the set.
  - Classify a new tuple:
    - the first rule satisfying the tuple is used to classify it
    - The classifier also contains a default rule, having the lowest precedence

# Typical Associative Classification Methods

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Build the classifier: adopts a variant of the FP-growth algorithm to find the complete set of rules satisfying the minimum confidence and minimum support thresholds.
  - Rule pruning whenever a rule is inserted into the rule base:
    - Given two rules, $R_1$ and $R_2$, if the antecedent of $R_1$ is more general than that of $R_2$ and $conf(R_1) >= conf(R_2)$, then $R_2$ is pruned
    - Prunes rules for which the rule antecedent and class are not positively correlated, based on a $\chi^2$ test of statistical significance









# Typical Associative Classification Methods

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classify a new tuple:
  - If the rules matching the new object are not consistent in class label
    - Statistical analysis on multiple rules
      - Rules are divided into groups according to class labels
      - Uses a weighted $\chi^2$ measure to find the strongest group of rules, based on statistical correlation of rules.
      - The new tuple is assigned to the class label of the strongest group.
- CMAR has slightly higher average accuracy and more efficient use of memory than CBA

# Typical Associative Classification Methods

- CPAR (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Build the classifier: Generation of predictive rules based on a rule generation algorithm (FOIL-like analysis) rather than frequent itemset mining
    - In FOIL, each time a rule is generated, the positive samples it satisfies are removed. In CPAR, the covered tuples remain under consideration, but reducing their weight.
  - Classify a new tuple:
    - Rules are divided into groups according to class labels
    - The best k rules, based on expected accuracy, of each group are used to predict the class label
  - High efficiency, accuracy similar to CMAR









# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Lazy vs. Eager Learning

- Lazy vs. eager learning
  - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it **uses many local linear functions to form** an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space







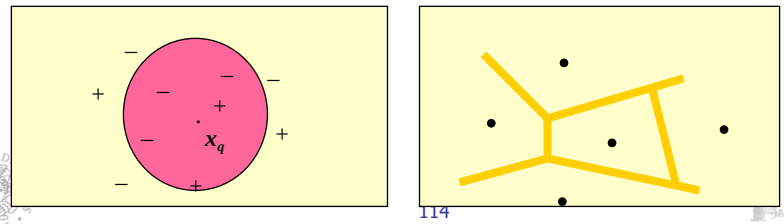


# Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - k-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
- Case-based reasoning
  - Uses symbolic representations and knowledge-based inference

# The k-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X}_1$, $\mathbf{X}_2$)
- Target function could be discrete- or real- valued
- For discrete-valued, k-NN returns the most common value among the k training examples nearest to $x_q$
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples










# Discussion on the k-NN Algorithm

- How can the distance be computed for non numeric attributes (categorical attributes)?
  - Compare the corresponding values of the attributes: if they are identical, the difference is 0; otherwise is 1
- What about missing values?
  - If a value of a given attribute is missing in one of the two tuples, then assume the maximum possible difference. This difference is 1 for nominal attributes and for numeric attributes when the value is missing in both the tuples.
- How is possible to determine a good value for k?
  - Experimentally: uses increasing values of k and chooses k with the maximum accuracy
    - Generally, the larger the number training instances, the larger the value of k

# Discussion on the k-NN Algorithm

- Choice of the distance: different distances can be used for incorporating attribute weighting and the pruning of noisy data tuples.
- Computational efficiency: 1-NN requires O(|D|) comparisons
  - By sorting and arranging the tuples into search trees, the number of comparisons can be reduced to O(log(|D|))
  - Parallel implementations
  - Partial distance: use only a subset of the n attributes; if the distance is higher than a threshold, then computation of distance is stopped
  - Editing methods: remove training tuples that prove to be useless.





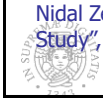


# Editing methods

- Wilson editing
  - Wilson editing cleans interclass overlap regions, thereby leading to smoother boundaries between classes.

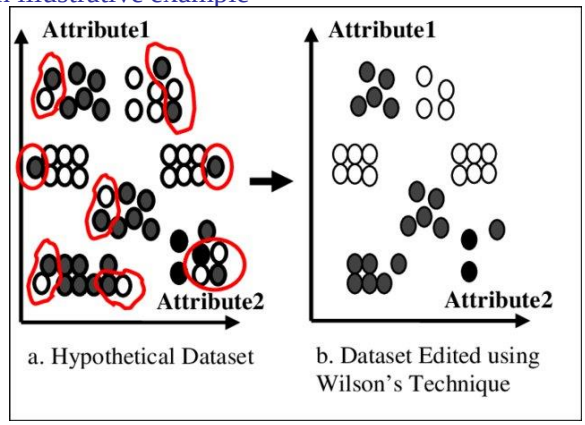
```
PREPROCESSING
A: For each example o_i in the dataset O,
  1: Find the K-Nearest Neighbors of o_i in O(excluding o_i)
  2: Label o_i with the class associated with the largest number of
     examples among the K nearest neighbors (breaking ties randomly)
B: Edit Dataset O by deleting all examples that were misclassified
  in step A.2

CLASSIFICATION RULE:
Classify new example q using K-NN rule with the edited subset O_r
```

Nidal Zeidat, Sujing Wang, and Christoph F. Eick, "Dataset Editing Techniques: A Comparative Study", Research Gate



# Editing methods

- Wilson editing
  - An illustrative example




Nidal Zeidat, Sujing Wang, and Christoph F. Eick, "Dataset Editing Techniques: A Comparative Study", Research Gate





# Editing methods

- Multi-edit
  - repeatedly applies Wilson editing to N random subsets of the original dataset until no more examples are removed.

A: DIFFUSION: Divide the dataset O into $N$>=3 random subsets $S_1$ ,…$S_N$
B: CLASSIFICATION: Classify $S_i$ using the K-NN rule with $S_{(i+1)Mod\ N}$ as the training set ($i$=1,…,$N$)
C: EDITING: Discard all incorrectly classified examples
D: CONFUSION: Replace $O$ by subset $O_r$ consisting of the union of all remaining examples in $S_1$,…$S_N$
E: TERMINATION: if the last iteration produced no editing then terminate; otherwise go to step A.

# Editing methods

- Citation Editing
  - Analogy: if a paper cites another article, the paper is related to that article. Similarly, if a paper is cited by an article, the paper is also related to that article. Thus both the citers and references are considered to be related to a given paper.

A: For each example $o_i$ in dataset $O$ do:
1: Find the $K$ nearest neighbors of $o_i$ *in* $O$ (excluding $o_i$)
2: Find the $C$ nearest citers *in* $O$ which count $o_i$ among their K nearest neighbors
3: Classify $o_i$ with the class of the majority of examples in a group consisting of $K$ nearest neighbors and $C$ nearest citers for example $o_i$.
B: Discard examples $o_i$ from $O$ that were misclassified in step A.3, obtaining $O_r$.



120



# Editing methods

- Supervised Clustering
  - O is replaced by subset Or which consists of cluster representatives that have been selected by the supervised clustering algorithm

PREPROCESSING
A: Apply a supervised clustering algorithm to dataset $O$ to produce a set clusters (each having a single representative).
*B: Edit* dataset $O$ by deleting all non-representative examples to produce subset $O_r$.



121

# Editing methods

- Experimental evaluation
  - Comparison on a number of datasets

| | 1-NN | Wilson | Multi-edit | Citation | SC Editing |
|---|---|---|---|---|---|
| a. ***UCI Datasets*** | | | | | |
| Classification accuracy | 81.81 | 82.66 | 72.80 | 82.73 | 82.67 |
| TS Compression Rate | 0 | 18.40 | 40.10 | 8.10 | 89.50 |
| b. ***2D Datasets*** | | | | | |
| Classification accuracy | 90.61 | 93.33 | 93.40 | 92.21 | 88.93 |
| TS Compression rate | 0 | 9.30 | 10.20 | 5.80 | 99.10 |

Training set compression rate = $\left(1 - \frac{r}{n}\right) \cdot 100$

$r$: Number of instances after editing in the training set

$n$: Number of instances in the original training ser

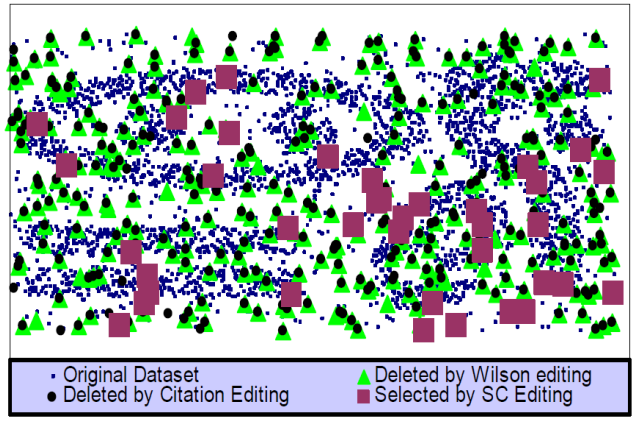
Nidal Zeidat, Sujing Wang, and Christoph F. Eick, "Dataset Editing Techniques: A Comparative Study", Research Gate

122



# Editing methods

- Visualization of the application of editing methods on Complex9_RN323 datase




Nidal Zeidat, Sujing Wang, and Christoph F. Eick, "Dataset Editing Techniques: A Comparative Study", Research Gate

123

# Discussion on the k-NN Algorithm for predicition

- k-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the k nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the k neighbors according to their distance to the query $x_q$
    - Give greater weight to closer neighbors $w \equiv \frac{1}{d(x_q, x_i)^2}$
- Robust to noisy data by averaging k-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, elimination of the least relevant attributes







# Case-Based Reasoning (CBR)

- Case-based reasoning (CBR) is the process of solving new problems based on the solutions of equal or similar past problems
- Core assumption: **similar problems have similar solutions**
- Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Everyday examples of CBR:
  - A lawyer who advocates a particular outcome in a trial based on legal precedents or a judge who creates case law.
  - An engineer copying working elements of nature (practicing biomimicry) is treating nature as a database of solutions to problems.

# Case-Based Reasoning (CBR)

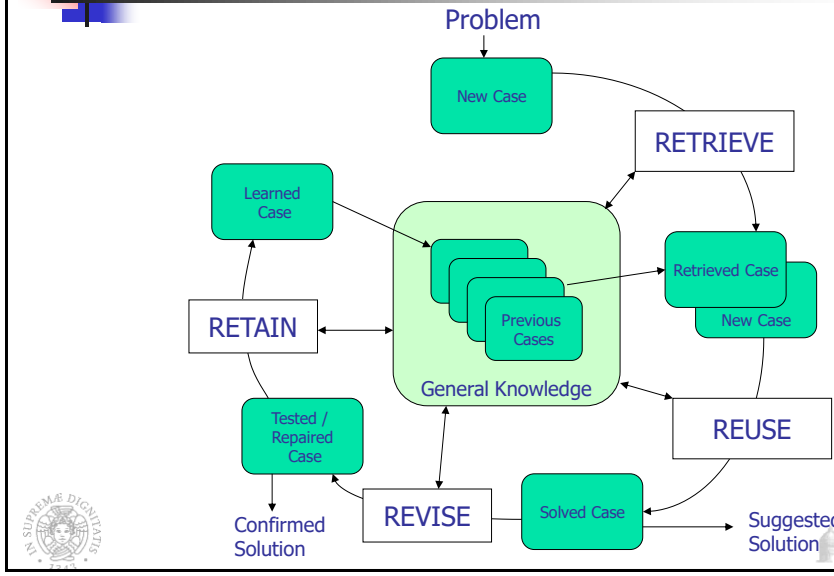
- Methodology
  - Cases represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - When a new (successful) solution to the new problem is found, a new experience is made, which can be stored in the case-base to increase its competence, thus implementing a learning behavior.
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- The CBR cycle was proposed by Aamodt and Plaza ("Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches") and consists of 4 sequential steps







# Case-Based Reasoning (CBR)






# Case-Based Reasoning (CBR)

- Retrieve
  - One or several cases from the case base are selected, based on the modeled similarity.
  - The retrieval task is defined as finding a small number of cases from the case-base with the highest similarity to the query.
  - This is a k-nearest-neighbor retrieval task considering a specific similarity function.
  - When the case base grows, the efficiency of retrieval decreases => methods that improve retrieval efficiency, e.g. specific index structures such as kd-trees, case-retrieval nets, or discrimination networks.







# Case-Based Reasoning (CBR)

- Reuse
  - Reusing a retrieved solution can be quite simple if the solution is returned unchanged as the proposed solution for the new problem.
  - Adaptation (if required, e.g. for synthetic tasks).
  - Several techniques for adaptation in CBR
    - Transformational adaptation
    - Generative adaptation
  - Most practical CBR applications today try to avoid extensive adaptation for pragmatic reasons.

# Case-Based Reasoning (CBR)

- Revise
  - In this phase, feedback related to the solution constructed so far is obtained.
  - This feedback can be given in the form of a correctness rating of the result or in the form of a manually corrected revised case.
  - The revised case or any other form of feedback enters the CBR system for its use in the subsequent retain phase.





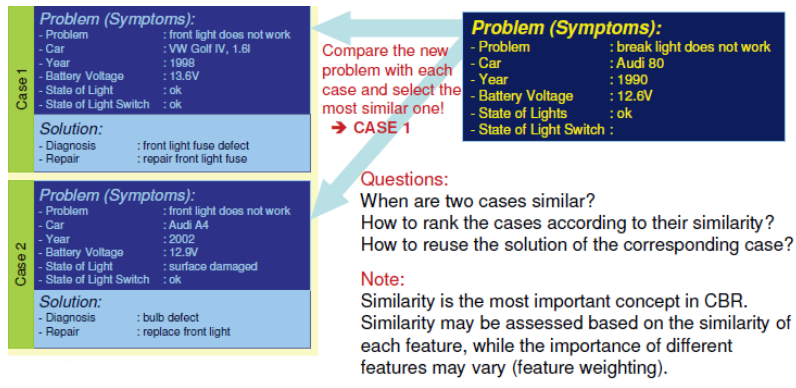


# Case-Based Reasoning (CBR)

- Retain
  - The retain phase is the learning phase of a CBR system (adding a revised case to the case base).
  - Explicit competence models have been developed that enable the selective retention of cases (because of the continuous increase of the case-base).

# Case-Based Reasoning (CBR)

- An example




Questions:
When are two cases similar?
How to rank the cases according to their similarity?
How to reuse the solution of the corresponding case?

Note:
Similarity is the most important concept in CBR. Similarity may be assessed based on the similarity of each feature, while the importance of different features may vary (feature weighting).





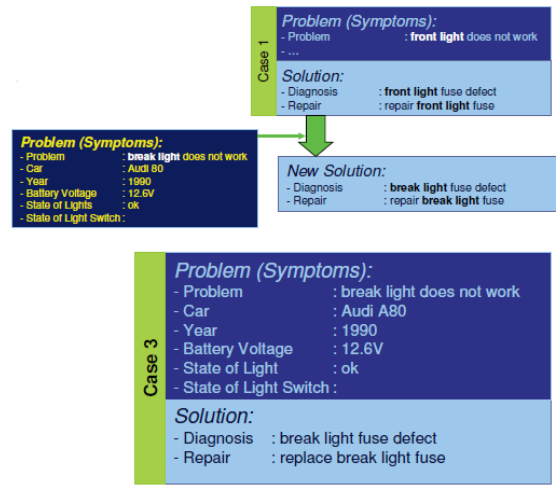


# Case-Based Reasoning (CBR)

- An example

Reuse

Retain



# Case-Based Reasoning (CBR)

- Application areas
  - help-desk and customer service
  - recommender systems in electronic commerce
  - knowledge and experience management
  - medical applications and applications in image processing
  - applications in law, technical diagnosis, design, planning
  - applications in the computer games and music domain.
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases







# Classification: Basic Concepts

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

# Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use test set of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves









# Classifier Evaluation Metrics: Confusion Matrix

**Confusion Matrix:**

| Actual class\Predicted class | $C_1$ | $\neg C_1$ |
|---|---|---|
| $C_1$ | **True Positives (TP)** | **False Negatives (FN)** |
| $\neg C_1$ | **False Positives (FP)** | **True Negatives (TN)** |

**Example of Confusion Matrix:**

| Actual class\Predicted class | buy_computer = yes | buy_computer = no | Total |
|---|---|---|---|
| buy_computer = yes | **6954** | **46** | 7000 |
| buy_computer = no | **412** | **2588** | 3000 |
| Total | 7366 | 2634 | 10000 |

- Given *m* classes, an entry, $CM_{i,j}$ in a confusion matrix indicates # of tuples in class i that were labeled by the classifier as class j
- May have extra rows/columns to provide totals

# Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

| A\P | C | ¬C | |
|---|---|---|---|
| C | **TP** | **FN** | **P** |
| ¬C | **FP** | **TN** | **N** |
| | **P'** | **N'** | **All** |

- **Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified**
  - Accuracy = (TP + TN)/All
- **Error rate: 1 – accuracy, or**
  - Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**







# Classifier Evaluation Metrics: Precision and Recall, and F-measures

- Precision: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- Recall: completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

- **Perfect precision score of 1.0**: every tuple that the classifier labeled as belonging to class C does indeed belong to class C. However, it does not tell us anything about the number of class C tuples that the classifier mislabeled.
- **Perfect recall score of 1.0**: every item from class C was labeled as such, but it does not tell us how many other tuples were incorrectly labeled as belonging to class C.

# Classifier Evaluation Metrics: Precision and Recall, and F-measures

- Inverse relationship between precision and recall
- A medical classifier may achieve high precision by labeling all cancer tuples that present a certain way as cancer, but may have low recall if it mislabels many other instances of cancer tuples.
- Precision and recall are also combined into a single measure
- F measure ($F_1$ or *F-score*): harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- $F_\beta$: weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

  - Commonly used $F_\beta$ measures are $F_2$ (weights recall more than precision) and $F_{0.5}$ (weights precision more than recall)







# Classifier Evaluation Metrics: Example

| Actual Class\Predicted class | cancer = yes | cancer = no | Total | Recognition(%) |
|---|---|---|---|---|
| cancer = yes | **90** | **210** | 300 | 30.00 (*sensitivity*) |
| cancer = no | **140** | **9560** | 9700 | 98.56 (*specificity*) |
| Total | 230 | 9770 | 10000 | 96.40 (*accuracy*) |

- *Precision* = 90/230 = 39.13%
- *Recall* = 90/300 = 30.00%

# Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- Evaluating classifier accuracy
  - A unique execution of different classifiers cannot allow concluding that one classifier is better than another
  - In one trial, by chance one classifier could have an accuracy higher than the other
  - We need to perform different trials. But how?
  - We have only a labelled dataset.



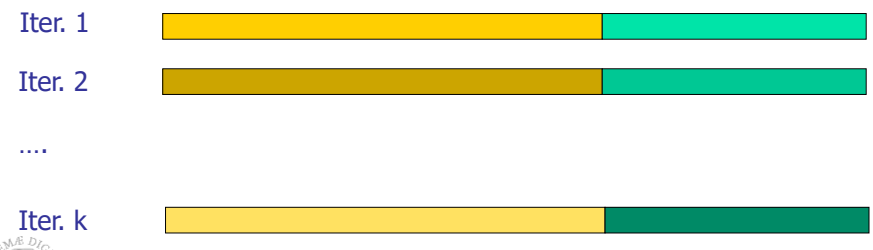






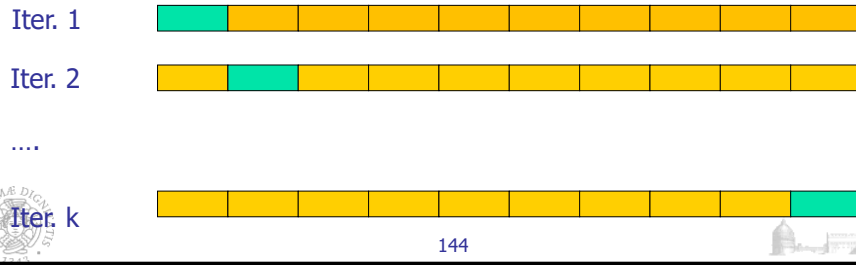
# Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained




# Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- Cross-validation (k-fold, where k = 10 is most popular)
  - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
  - At i-th iteration, use $D_i$ as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, that is, only a sample is left out at a time for the test set. Suitable for small sized data.
  - *Stratified cross-validation*: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data










# Evaluating Classifier Accuracy: Bootstrap

- Bootstrap
  - Samples the given training tuples uniformly with replacement
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
  - Works well with small data sets
- Several bootstrap methods, and a commonly used is .632 boostrap
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since the probability of not being chosen is $(1 - 1/d)^d$ , when *d* is large results to be $\approx e^{-1} = 0.368$)
  - Repeat the sampling procedure k times. The overall accuracy of the model is:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

# Estimating Confidence Intervals: Classifier Models $M_1$ vs. $M_2$

- Suppose we have 2 classifiers, $M_1$ and $M_2$, which one is better than the other?
  - Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
  - These mean error rates are just estimates of error on the true population of future data cases
- What if the difference between the 2 error rates is just attributed to chance?
  - Use a test of statistical significance
  - Obtain confidence limits for our error estimates. For instance "One model is better than the other by a margin of error of ± 4%."



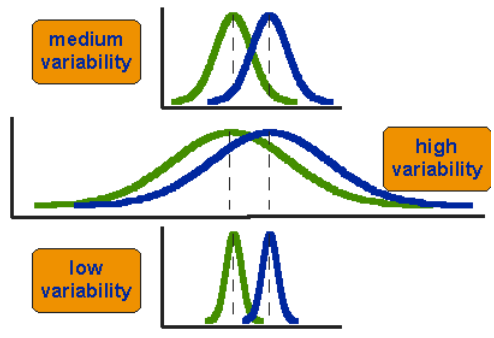






# Estimating Confidence Intervals: Classifier Models $M_1$ vs. $M_2$

- Why do we need a statistical test?




# Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assumptions: the samples (for instance, classification accuracy) are normally distributed within each group (output of each classifier) and the variances of the two populations are not reliably different
- Use t-test (or Student's t-test)
  - evaluate the differences in means between two groups
  - Null Hypothesis: the two distribuitions of accuracy for $M_1$ and $M_2$ are the same
- If we can reject null hypothesis, then
  - we conclude that the difference between $M_1$ and $M_2$ is statistically significant
- Choose the model with lower error rate







# Parametric statistical tests: t-test

- Same test set used for $M_1$ and $M_2$: pairwise comparison
  - For the $i^{th}$ round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
  - Average over 10 rounds to get $err(M_1)$ and $err(M_2)$
  - t-test computes t-statistic with k-1 degrees of freedom:

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}}$$

$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))]^2$$

- If two test sets available: use non-paired t-test

$$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}}$$

where $k_1$ & $k_2$ are # of cross-validation samples used for $M_1$ & $M_2$, resp.

# Parametric statistical tests: t-test

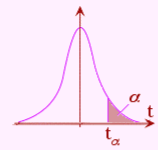
- Are $M_1$ and $M_2$ significantly different?
  - Compute t. Select significance level (e.g. sig = 5%)
  - Consult table for t-distribution: Find t value corresponding to k-1 degrees of freedom (here, 9)
  - t-distribution is symmetric: typically upper % points of distribution shown → look up value for confidence limit z=sig/2 (here, 0.025)
  - If t > z or t < -z, then t value lies in rejection region:
    - Reject null hypothesis that means error rates of $M_1$ and $M_2$ are same
    - Conclude: statistically significant difference between $M_1$ and $M_2$
  - Otherwise, conclude that any difference is chance
  - In non-paired version, the number of degrees of freedom used is taken as the minimum number of degrees of the two models







# Parametric statistical tests: t-test




**T-Distribution Table**

| df | $\alpha = 0.1$ | 0.05 | 0.025 | 0.01 | 0.005 | 0.001 | 0.0005 |
|---|---|---|---|---|---|---|---|
| ∞ | $t_\alpha$=1.282 | 1.645 | 1.960 | 2.326 | 2.576 | 3.091 | 3.291 |
| 1 | 3.078 | 6.314 | 12.706 | 31.821 | 63.656 | 318.289 | 636.578 |
| 2 | 1.886 | 2.920 | 4.303 | 6.965 | 9.925 | 22.328 | 31.600 |
| 3 | 1.638 | 2.353 | 3.182 | 4.541 | 5.841 | 10.214 | 12.924 |
| 4 | 1.533 | 2.132 | 2.776 | 3.747 | 4.604 | 7.173 | 8.610 |
| 5 | 1.476 | 2.015 | 2.571 | 3.365 | 4.032 | 5.894 | 6.869 |
| 6 | 1.440 | 1.943 | 2.447 | 3.143 | 3.707 | 5.208 | 5.959 |
| 7 | 1.415 | 1.895 | 2.365 | 2.998 | 3.499 | 4.785 | 5.408 |
| 8 | 1.397 | 1.860 | 2.306 | 2.896 | 3.355 | 4.501 | 5.041 |
| 9 | 1.383 | 1.833 | 2.262 | 2.821 | 3.250 | 4.297 | 4.781 |
| 10 | 1.372 | 1.812 | 2.228 | 2.764 | 3.169 | 4.144 | 4.587 |

# Non parametric statistical tests: The Wilcoxon signed rank sum test

- The **Wilcoxon signed rank sum test** is an example of non-parametric or distribution free test.
- **The null hypothesis for this test is that the medians of two samples are equal** (in other words, we test whether two populations have the same distribution with the same median). It is generally used:
  - As a non-parametric alternative to the one-sample t test or paired t test.
  - For ordered (ranked) categorical variables without a numerical scale.

  Assumptions (as in t-test, except for normal distribution):
  1. The two samples are independent of one another
  2. The two populations have equal variance or spread









# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Wilcoxon test for Paired data

1. **Calculate each paired difference**, $d_i = x_i - y_i$, where $x_i$, $y_i$ are the pairs of observations
2. **Rank difference $d_i$ ignoring the signs** (i.e. assign rank 1 to the smallest $|d_i|$, rank 2 to the next one, etc.
3. **Label each rank with its sign**, according to the sign of $d_i$
4. **Calculate $W^+$**, the sum of ranks of the positive differences $d_i$ and **$W^-$** the sum of the ranks of the negative differences $d_i$. (As a check the total $W^+ + W^-$ should be equal to $\frac{n(n+1)}{2}$, where n is the number of pairs of observations in the sample)

# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Under the null hypothesis, we would expect the distribution of the differences to be approximately symmetric around zero and the distribution of positives and negatives to be distributed at random among the ranks. Under this assumption, it is possible to work out the exact probability of every possible outcome for W as follows:

5. **Choose $W=\min(W^+, W^-)$**
6. **Use tables of critical values for the Wilcoxon signed rank sum test to find the probability of observing a value of W or more extreme**. Most tables give both one-sided and two-sided p-values. If not, double the one-sided p-value to obtain the two-sided p-value.









# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Normal approximation
  - If the number of observations/pairs is such that $\frac{n(n+1)}{2}$ is large enough (>20), a normal approximation can be used with

$$\mu_W = \frac{n(n+1)}{4} \qquad \sigma_W = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

Dealing with ties:

- Observations in the sample may be exactly equal to the median value M (i.e. 0 in the case of paired differences). Ignore such observations and adjust n accordingly.
- Two or more observations/differences may be equal. If so, average the ranks across the tied observations and reduce the variance by $\frac{t^3-t}{48}$ for each group of t tied ranks.

# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Example
  - Adapted to classification.
  - 12-fold cross-validation.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M1 | 2.0 | 3.6 | 2.6 | 2.6 | 7.3 | 3.4 | 14.9 | 6.6 | 2.3 | 2.0 | 6.8 | 8.5 |
| M2 | 3.5 | 5.7 | 2.9 | 2.4 | 9.9 | 3.3 | 16.7 | 6.0 | 3.8 | 4.0 | 9.1 | 20.9 |
| Diff | +1.5 | +2.1 | +0.3 | -0.2 | +2.6 | -0.1 | +1.8 | -0.6 | +1.5 | +2.0 | +2.3 | +12.4 |

  - Ranking the differences

| Diff | 0.1 | 0.2 | 0.3 | 0.6 | 1.5 | 1.5 | 1.8 | 2.0 | 2.1 | 2.3 | 2.6 | 12.4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rank | 1 | 2 | 3 | 4 | 5.5 | 5.5 | 7 | 8 | 9 | 10 | 11 | 12 |
| Sign | - | - | + | - | + | + | + | + | + | + | + | + |

The Wilcoxon signed rank sum test, Rosie Shier, 2004.







# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Example
  - Calculating $W^+$ and $W^-$ gives:

    $W^- = 1 + 2 + 4 = 7$

    $W^+ = 3 + 5.5 + 5.5 + 7 + 8 + 9 + 10 + 11 + 12 = 71$

  - Therefore

$$\frac{n(n+1)}{2} = \frac{12 \cdot 13}{2} = 78 > 20$$

$$W = min(W^-, W^+) = 7$$

# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Example
  - We can use a normal approximation. We have one group of 2 tied ranks, so we must reduce the variance by $\frac{8-2}{48} = 0.125$.
  - We can compute the z-score defined as:

$$z = \frac{x - \mu_W}{\sigma_W}$$

  - A z-score is a measure of how many standard deviations below or above the population mean a raw score is.









# Non parametric statistical tests: The Wilcoxon signed rank sum test

- Example
  - We get:

$$z = \frac{7 - \frac{12 \cdot 13}{4}}{\sqrt{\frac{12 \cdot 13 \cdot 25}{24} - 0.125}} = \frac{7 - 39}{\sqrt{162.5 - 0.125}} = 2.511$$

  - Looking up this score in the z-table*, we get an area of 0.9880, equal to a two-tailed p-value of 0.012. This is a tiny p-value, a strong indication that the medians are significantly different.
  - *The z-table here is to the right of the mean, so I had to double the actual result I found (.4940).

# Estimating Confidence Intervals: Statistical Significance

- We applied the statistical tests to the accuracy (just one value)
- How can we compare two classifiers whether the dataset is imbalanced using statistical tests?
  - You recall in this case we have the pairs (Recall-Precision) or (sensitivity-specificity).
  - Possible solutions
    - F-measure (combination between recall and precision)
    - AUC - Area Under the ROC curve (see next slides)



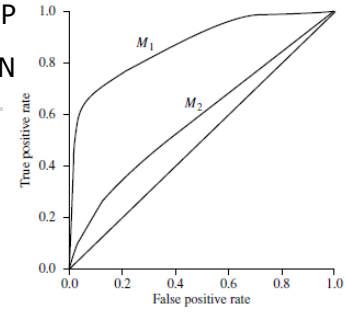




# Model Selection: ROC Curves

TPR= TP/P

FPR= FP/N



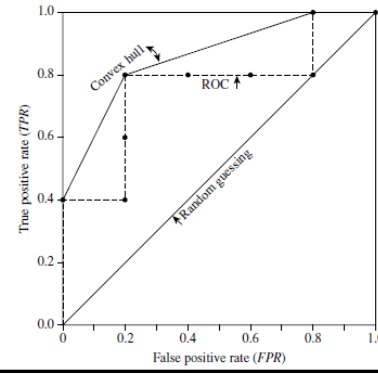

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (AUC) is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

# Model Selection: ROC Curves

- Example: plotting an ROC curve
- Convex hull: minimal convex set containing the points of the ROC curve

| Tuple # | Class | Prob. | TP | FP | TN | FN | TPR | FPR |
|---|---|---|---|---|---|---|---|---|
| 1 | P | 0.90 | 1 | 0 | 5 | 4 | 0.2 | 0 |
| 2 | P | 0.80 | 2 | 0 | 5 | 3 | 0.4 | 0 |
| 3 | N | 0.70 | 2 | 1 | 4 | 3 | 0.4 | 0.2 |
| 4 | P | 0.60 | 3 | 1 | 4 | 2 | 0.6 | 0.2 |
| 5 | P | 0.55 | 4 | 1 | 4 | 1 | 0.8 | 0.2 |
| 6 | N | 0.54 | 4 | 2 | 3 | 1 | 0.8 | 0.4 |
| 7 | N | 0.53 | 4 | 3 | 2 | 1 | 0.8 | 0.6 |
| 8 | N | 0.51 | 4 | 4 | 1 | 1 | 0.8 | 0.8 |
| 9 | P | 0.50 | 5 | 4 | 0 | 1 | 1.0 | 0.8 |
| 10 | N | 0.40 | 5 | 5 | 0 | 0 | 1.0 | 1.0 |










# Model Selection: Cost of a classifier

- Cost of a classifier represented by the point (FPR, TPR)

$$\text{cost} = FPR \cdot \mathrm{P}(n) \cdot c(Y,n) + FNR \cdot \mathrm{P}(p) \cdot c(N,p)$$

$$TPR = TP/P = TP/(TP + FN) \quad FNR = FN/(FN + TP) = 1 - TPR$$

  - P(n) and P(p): a-priori probabilities of a negative example and a positive example
  - C(Y,n) and C(N,p): false positive cost and false negative cost

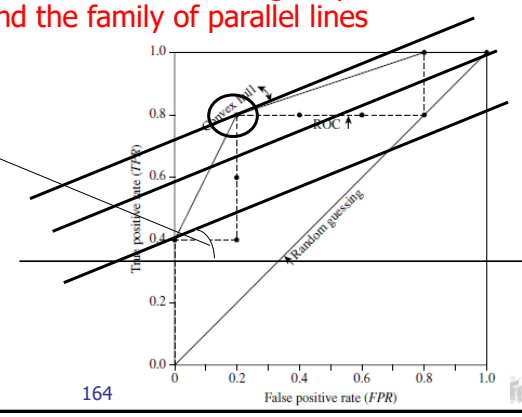
- Once fixed the values of P(n), c(Y,n), P(p) and C(N,p), we can obtain a family of parallel lines (called iso-cost lines) with slope

$$\frac{\mathrm{P}(n)c(Y,n)}{\mathrm{P}(p)c(N,p)}$$

# Model Selection: Cost of a classifier

- The points belonging to the same line have the same cost, and the cost decreases as we move to parallel lines closer to the point (0,1), i.e., more north-west
- Point that minimizes the classification cost: tangent point between the ROC curve and the family of parallel lines

$$\frac{P(n)c(Y,n)}{P(p)c(N,p)}$$




- Just as example

$$c(Y, n)/c(N, p) = 1/60$$

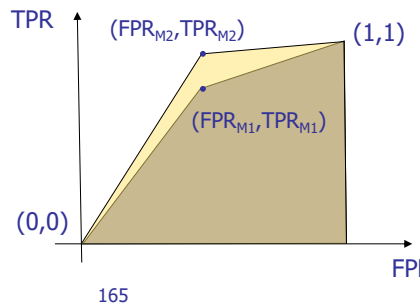






# How can we compare different classifiers with AUC (case imbalanced datasets)?

- Connect the point (FPR,TPR) corresponding to the classifier to point (0,1) and to point (1,1).
- Compute the AUC for each trial, thus generating a distribution of AUCs in place of a distribution of accuracies (for instance, in the cross-validation for each repetition with different folds)
- Perform a statistical test on the distributions of AUCs



# Classification: Basic Concepts

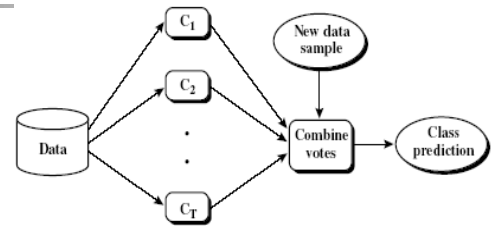
- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary







# Ensemble Methods: Increasing the Accuracy




- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, $M_1$, $M_2$, …, $M_k$, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

# Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of d tuples, at each iteration i, a training set $D_i$ of d tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model $M_i$ is learned for each training set $D_i$
- Classification: classify an unknown sample X
  - Each classifier $M_i$ returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to X

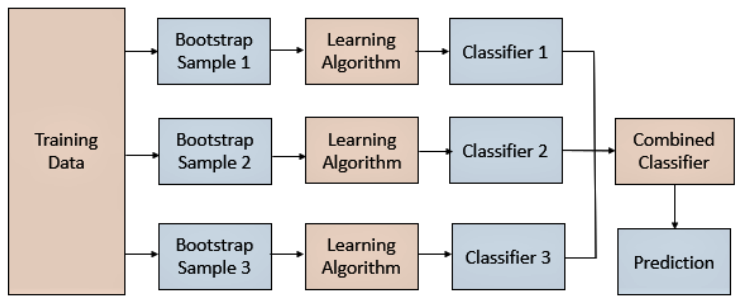








# Bagging: Boostrap Aggregation




# Bagging: Boostrap Aggregation

- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction



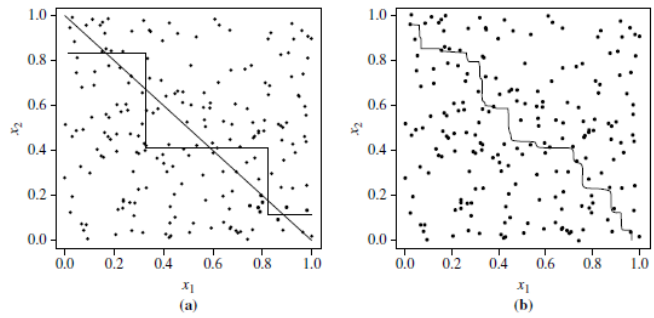






# Bagging: Boostrap Aggregation

- To help illustrate the power of an ensemble, consider a simple two-class problem described by two attributes, $x_1$ and $x_2$. The problem has a linear decision boundary.
- Figure (a) shows the decision boundary of a decision tree classifier on the problem. Figure (b) shows the decision boundary of an ensemble of decision tree classifiers on the same problem.




# Boosting

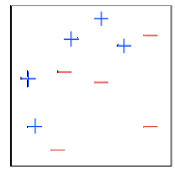
- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier $M_i$ is learned, the weights are updated to allow the subsequent classifier, $M_{i+1}$, to pay more attention to the training tuples that were misclassified by $M_i$
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data





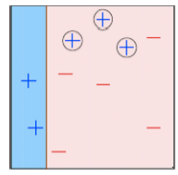


# Boosting



Nikolaos Nikolaou, "Introduction to Adaboost", University of Manchester

# Boosting

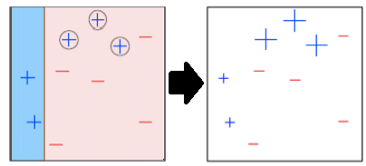




175



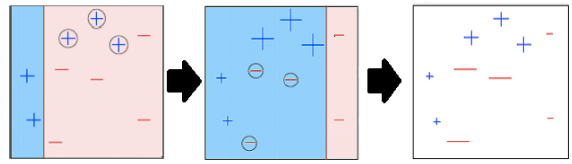
# Boosting





176

# Boosting

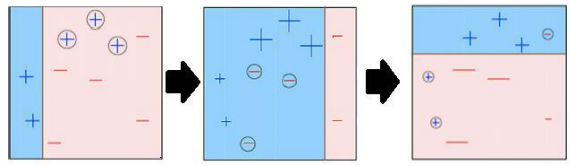




177



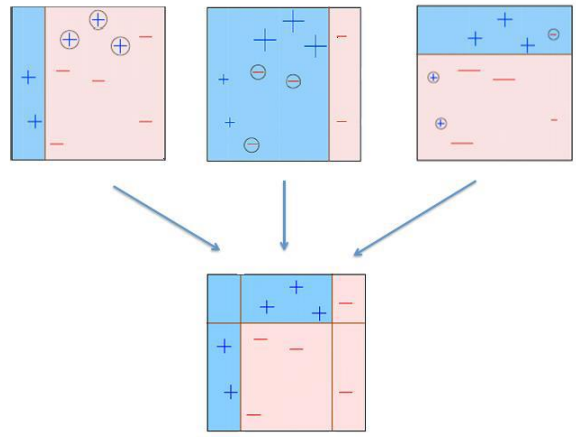
# Boosting





178

# Boosting



Nikolaos Nikolaou, "Introduction to Adaboost", University of Manchester







# Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, $y_1$), …, ($\mathbf{X_d}$, $y_d$)
- Initially, all the weights of tuples are set to the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set $D_i$ of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model $M_i$ is derived from $D_i$
  - Its error rate is calculated using $D_i$ as a test set
  - Classifier $M_i$ error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(\mathbf{X_j})$$

where err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$.

# Adaboost (Freund and Schapire, 1997)

- If a tuple ($\mathbf{X_j}$, $y_j$), is correctly classified, then the weight is updated as follows:

$$w_j = w_j \cdot error(M_i)/(1-error(M_i))$$

  - Once the weights are updated, the weights for all the tuples (both correctly and incorrectly classified) are normalised so that their sum remains the same it was before.
  - The normalization is performed by multiplying the weight by the sum of the old weights and dividing it by the sum of the new weights (the weights of misclassified tuples are increased!)
- Once the boosting is complete, how is the ensemble of classifiers used to predict the class label of a tuple?









# Adaboost (Freund and Schapire, 1997)

- **Boosting assigns a weight to each classifier's vote**, based on how well the classifier performed.
  - The lower a classifier's error rate, the more accurate it is and therefore the higher its weight for voting should be
  - **The weight of classifier $M_i$'s vote** is

$$log\frac{1-error(M_i)}{error(M_i)}$$

- For each class *c* we sum the weights of each classifier that assigned class *c* to X. The class with the highest sum is the "winner" and is returned as the class prediction for tuple X.

# Adaboost (Freund and Schapire, 1997)

**Algorithm: AdaBoost.** A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

**Input:**

- $D$, a set of $d$ class-labeled training tuples;
- $k$, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

**Output:** A composite model.









# Adaboost (Freund and Schapire, 1997)

**Method:**

(1) initialize the weight of each tuple in $D$ to $1/d$;
(2) **for** $i = 1$ to $k$ **do** // for each round:
(3) sample $D$ with replacement according to the tuple weights to obtain $D_i$;
(4) use training set $D_i$ to derive a model, $M_i$;
(5) compute $error(M_i)$, the error rate of $M_i$ (Eq. 8.34)
(6) **if** $error(M_i) > 0.5$ **then**
(7) go back to step 3 and try again;
(8) **endif**
(9) **for** each tuple in $D_i$ that was correctly classified **do**
(10) multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights
(11) normalize the weight of each tuple;
(12) **endfor**

# Adaboost (Freund and Schapire, 1997)

**To use the ensemble to classify tuple, $X$:**

(1) initialize weight of each class to 0;
(2) **for** $i = 1$ to $k$ **do** // for each classifier:
(3) $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4) $c = M_i(X)$; // get class prediction for $X$ from $M_i$
(5) add $w_i$ to weight for class $c$
(6) **endfor**
(7) return the class with the largest weight;









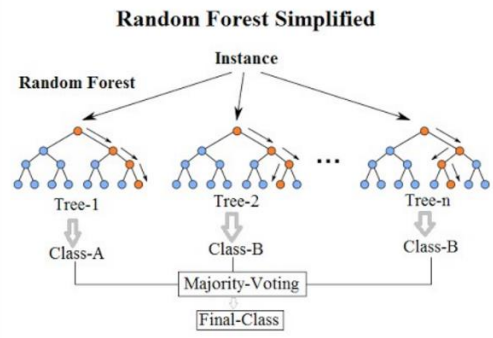
# Boosting vs Bagging

- Both train many models on different versions of initial dataset and then aggregate, but

| BAGGING | ADABOOST |
|---|---|
| Resample dataset | Resample or reweight dataset |
| Builds base models in parallel | Builds base models sequentially |
| Reduces variance (doesn't work well with e.g. decision stumps) | Also reduces bias (works well with stumps) |

# Random Forest (Breiman 2001)

- Random Forest (RF):
  - Each classifier is a **decision tree classifier** and is generated using a random selection of attributes at each node to determine the split
  - Each tree votes and the most popular class is returned






187



# Random Forest (Breiman 2001)

- Classical Algorithm:

  Let N and M be the number of training instances and attributes, respectively. Let m be the number of attributes to be used for choosing the decision attribute at any node

  1. **Choose a training set by randomly** extracting N samples with replacement from all available training instances (i.e. take a bootstrap sample). Use the rest of the instances to estimate the error of the tree.
  2. For each node of the tree**, randomly choose m attributes** on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
  3. Each tree is **fully grown and not pruned** (as may be done in constructing a normal tree classifier).



188

# Random Forest (Breiman 2001)

- Classification:

  Each tree assigns a class to the unlabeled instance. **The most popular class is returned.**

- Advantages:
  - RF is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
  - RF runs efficiently on large databases.
  - RF can handle thousands of input variables without variable deletion.
  - RF gives estimates of what variables are important in the classification.
  - RF generates an internal unbiased estimate of the generalization error as the forest building progresses.
  - RF has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.









# Random Forest (Breiman 2001)

- Disadvantages:
  - **RFs have been observed to overfit** for some datasets with noisy classification/regression tasks.
  - For data including categorical variables with different number of levels, RFs are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.

# Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - Oversampling: re-sampling of data from positive class
  - Under-sampling: randomly eliminate tuples from negative class
  - Threshold-moving: moves the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - Ensemble techniques: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks



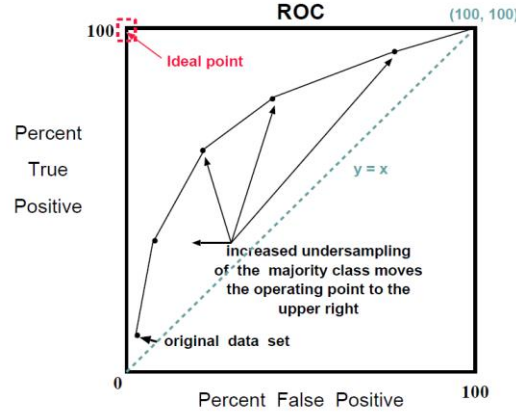






# Classification of Class-Imbalanced Data Sets

- SMOTE: Synthetic Minority Over-sampling Technique
  - Example of oversampling in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement
  - Observation




# Classification of Class-Imbalanced Data Sets

- SMOTE: Synthetic Minority Over-sampling Technique
  - The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors.
  - Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen.
    - For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each.





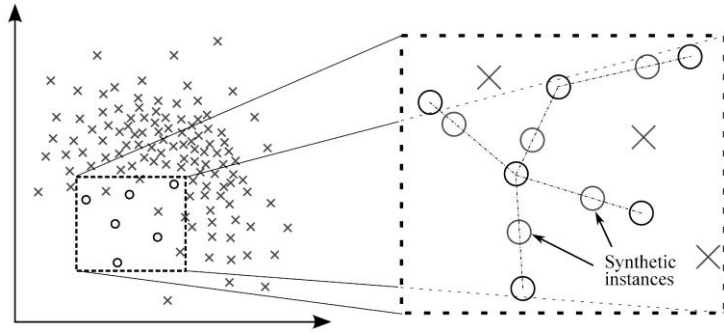




# Classification of Class-Imbalanced Data Sets

- SMOTE: Synthetic Minority Over-sampling Technique
  - Synthetic samples are generated in the following way:
    - **Take the difference between the feature vector** (sample) under consideration and its nearest neighbor.
    - **Multiply this difference by a random number between 0 and 1**, and add it to the feature vector under consideration.
    - This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

# Classification of Class-Imbalanced Data Sets






195



# Classification

- Classification: Basic Concepts
- Preparing Data for Classification
- Comparing Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Classification by Using Frequent Patterns
- Lazy Learners
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary



196

# Summary

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and $F_\beta$ measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.









# Summary

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method