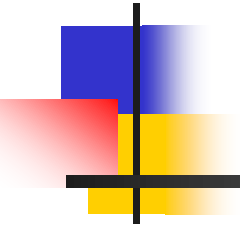


Classification & Prediction





Classification and Prediction

- **What is classification? What is prediction?**
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by back propagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary



Classification vs. Prediction

- **Classification:**

- Predicts categorical class labels
- 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

- **Prediction:**

- models continuous-valued functions, i.e., predicts unknown or missing values

- **Typical Applications**

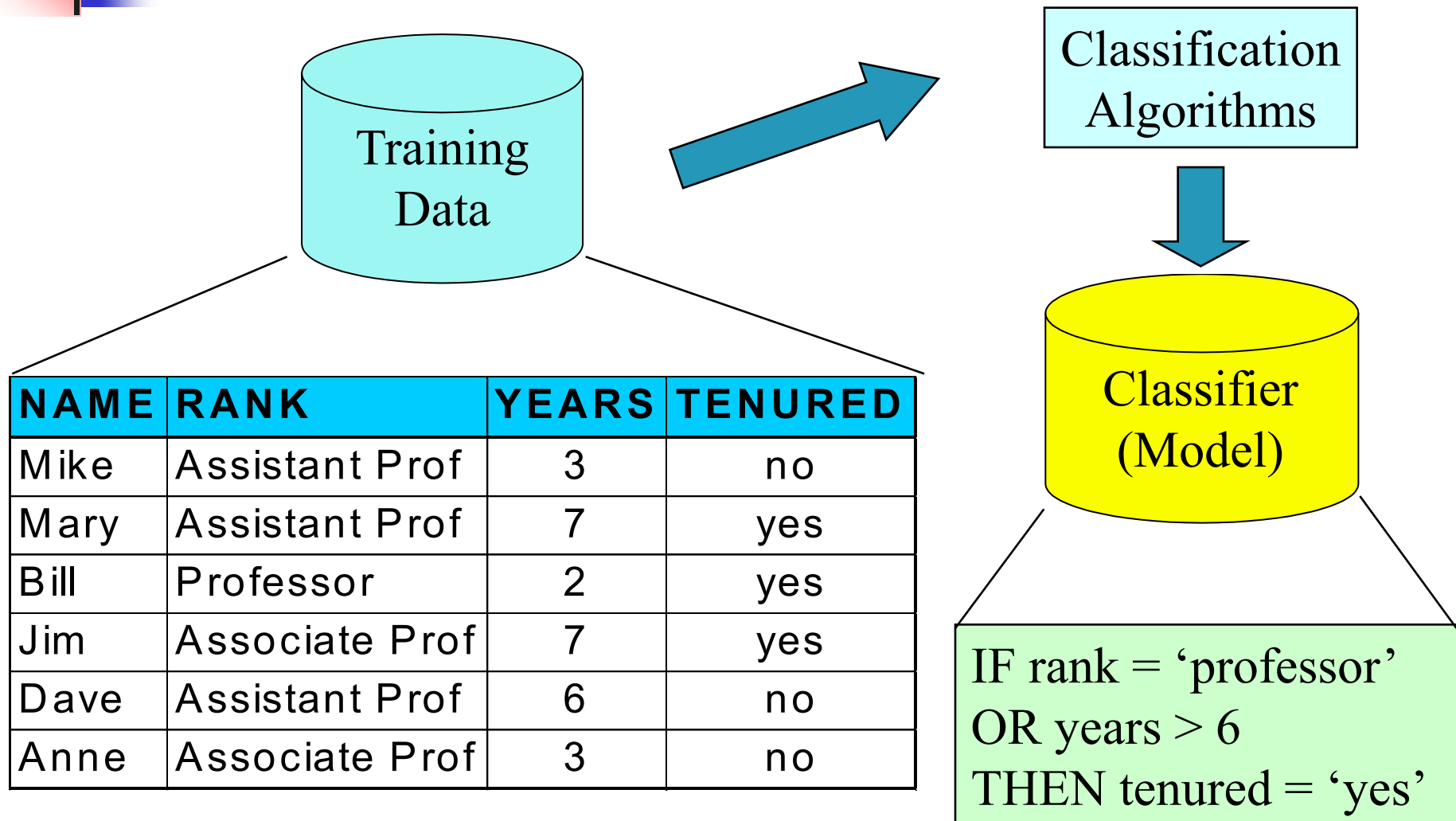
- Credit approval
- Target marketing
- Medical diagnosis
- Treatment effectiveness analysis



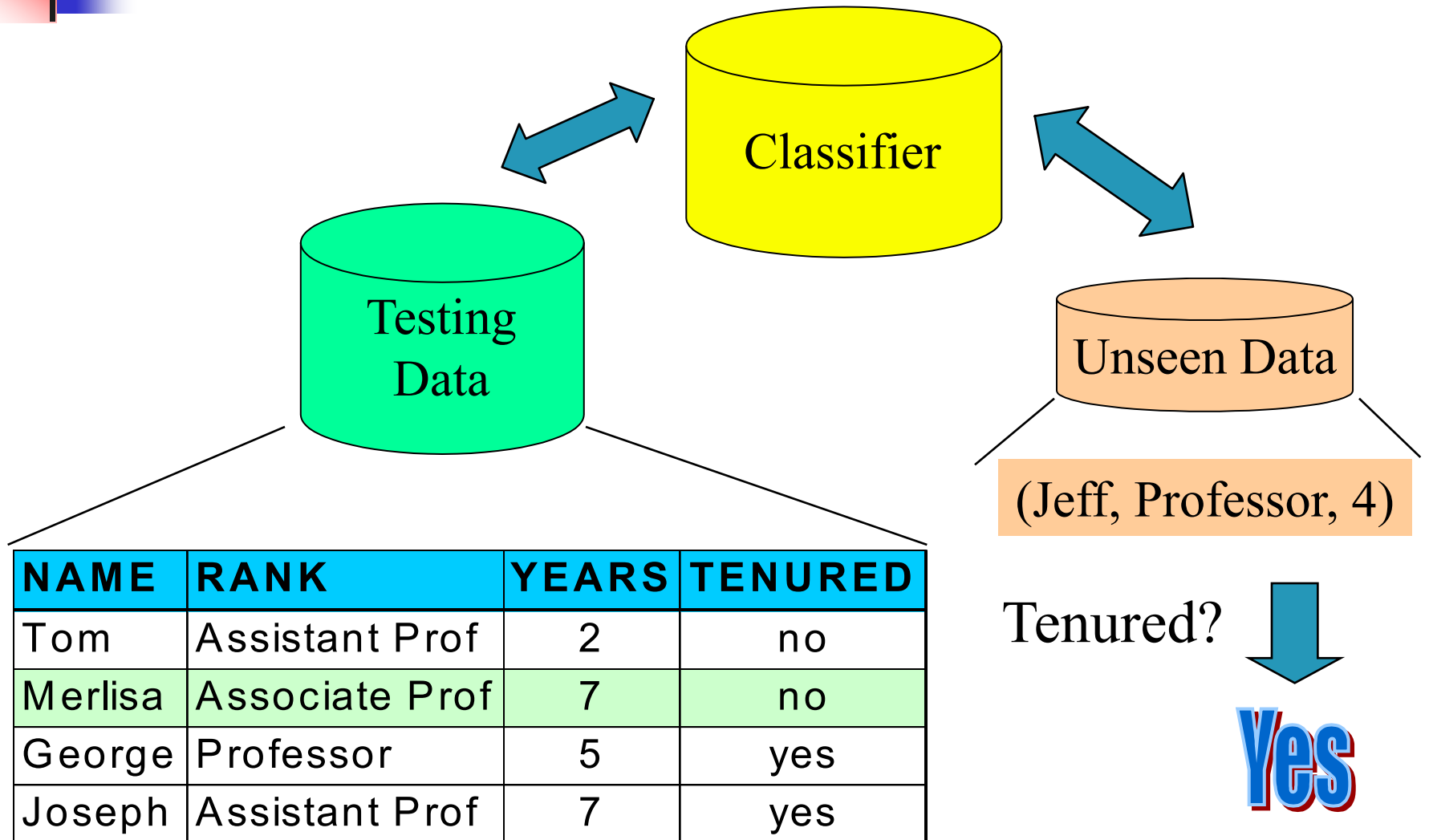
Classification—A Two-Step Process

- **Model construction**: describing a set of predetermined classes
 - 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: **training set**
 - The model is represented as classification rules, decision trees, or mathematical formulae
- **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
 - Test set is independent of training set, otherwise over-fitting will occur

Classification Process (1): Model Construction



Classification Process (2): Use the Model in Prediction



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. with the aim of establishing the existence of classes or clusters in the data



Classification and Prediction

- What is classification? What is prediction?
- **Issues regarding classification and prediction**
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary



Issues regarding classification and prediction (1): Data Preparation

- Data cleaning
 - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
 - Remove the irrelevant or redundant attributes
- Data transformation
 - Generalize and/or normalize data



Issues regarding classification and prediction (2): Evaluating Classification Methods

- Predictive accuracy
- Speed and scalability
 - time to construct the model
 - time to use the model
- Robustness
 - handling noise and missing values
- Scalability
 - efficiency in disk-resident databases
- Interpretability:
 - understanding and insight provided by the model
- Goodness of rules
 - decision tree size
 - compactness of classification rules



Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- **Classification by decision tree induction**
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary

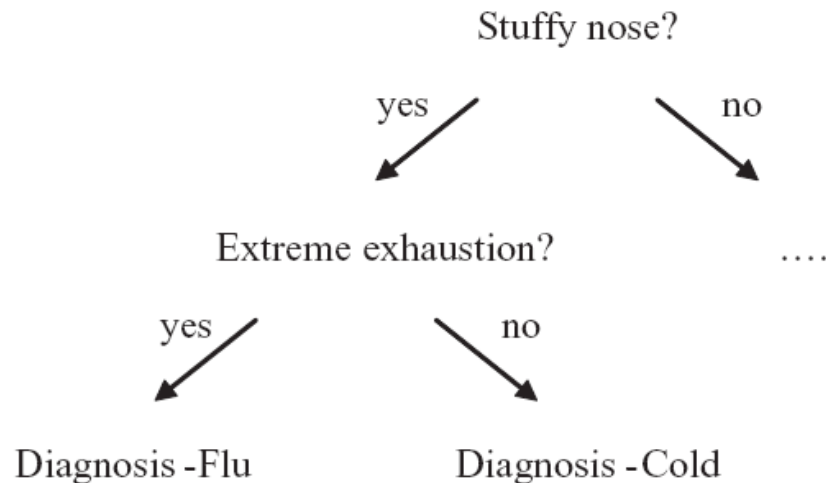


Classification by Decision Tree Induction

- Decision tree
 - A flow-chart-like tree structure
 - Internal node denotes a test on an attribute
 - Branch represents an outcome of the test
 - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
 - Tree construction
 - At start, all the training examples are at the root
 - Partition examples recursively based on selected attributes
 - Tree pruning
 - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
 - Test the attribute values of the sample against the decision tree

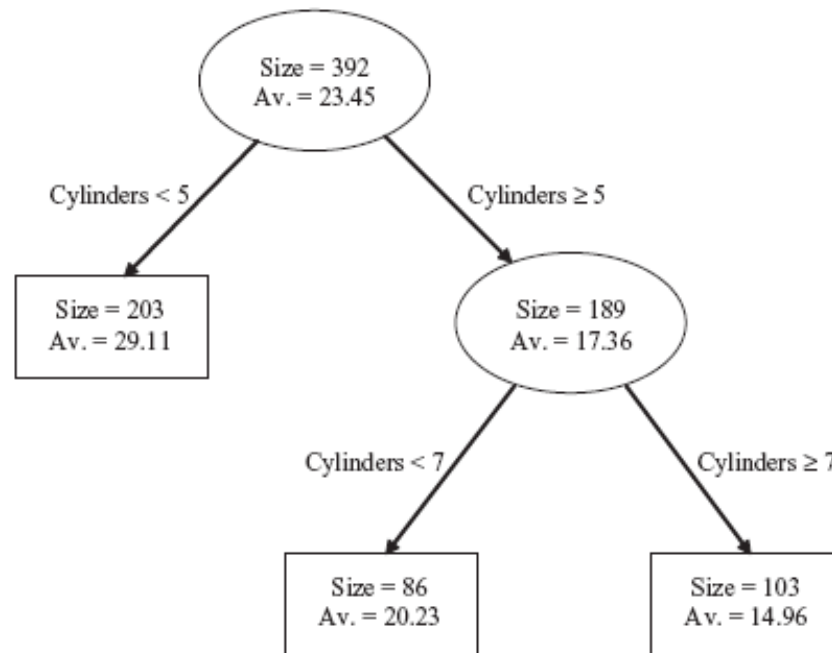
Decision Tree Example

For example, you may visit a doctor and your doctor may ask you to describe your symptoms. You respond by saying you have a stuffy nose. In trying to diagnose your condition the doctor may ask you further questions such as whether you are suffering from extreme exhaustion. Answering yes would suggest you have the flu, where as answering no would suggest you have a cold. This line of questioning is common to many decision making processes and can be shown visually as a decision tree,



Decision Tree for the diagnosis of cold and Flu

Decision Tree Example



Decision tree generated from a data set of cars

Decision Tree Example

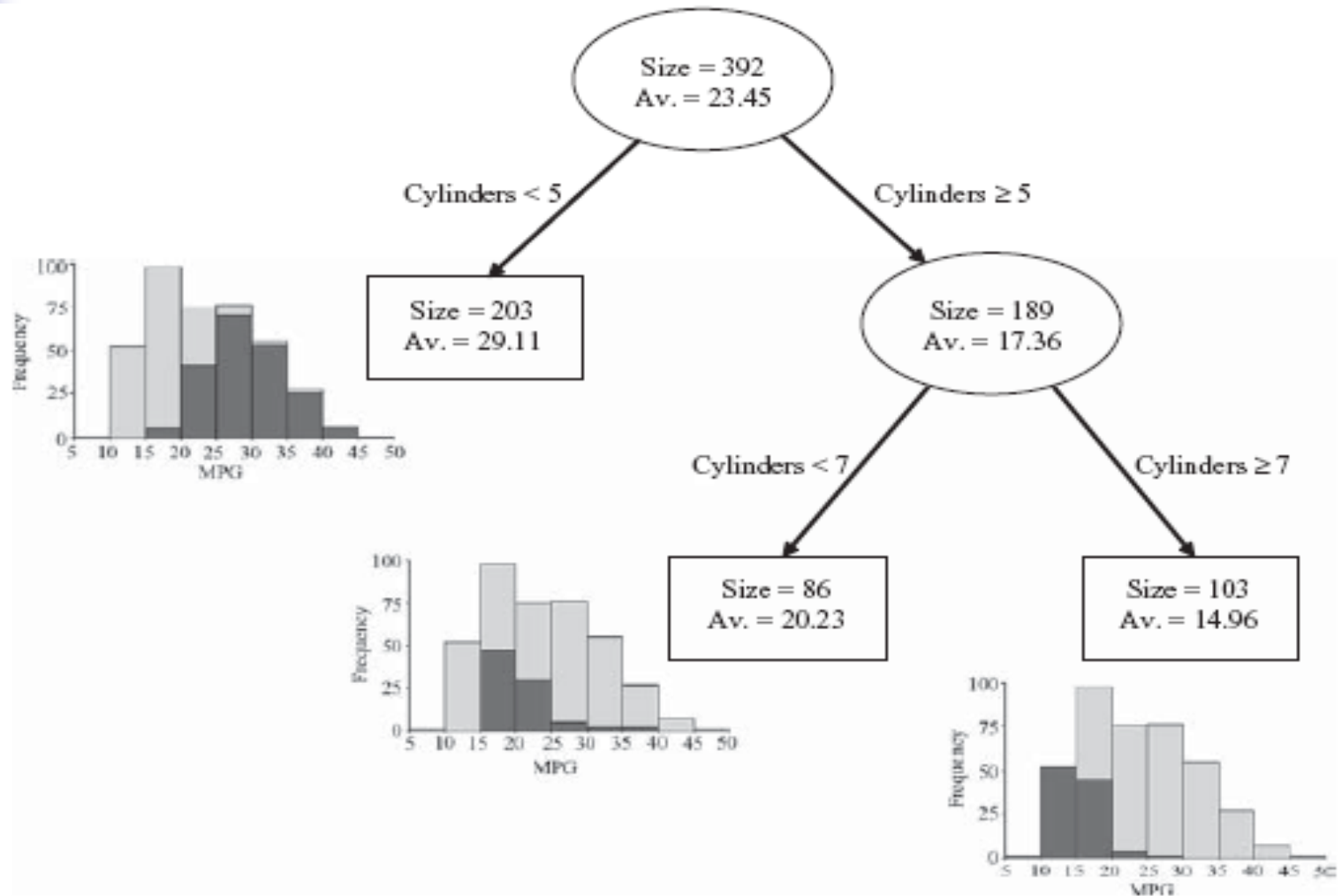


Figure 6.33. Decision tree illustrating the use of a response variable (MPG) to guide tree generation



Reasons for using Decision Tree

There are many reasons to use decision trees:

- **Easy to understand:** Decision trees are widely used to explain how decisions are reached based on multiple criteria.
- **Categorical and continuous variables:** Decision trees can be generated using either categorical data or continuous data.
- **Complex relationships:** A decision tree can partition a data set into distinct regions based on ranges or specific values.



Disadvantages of Decision Tree

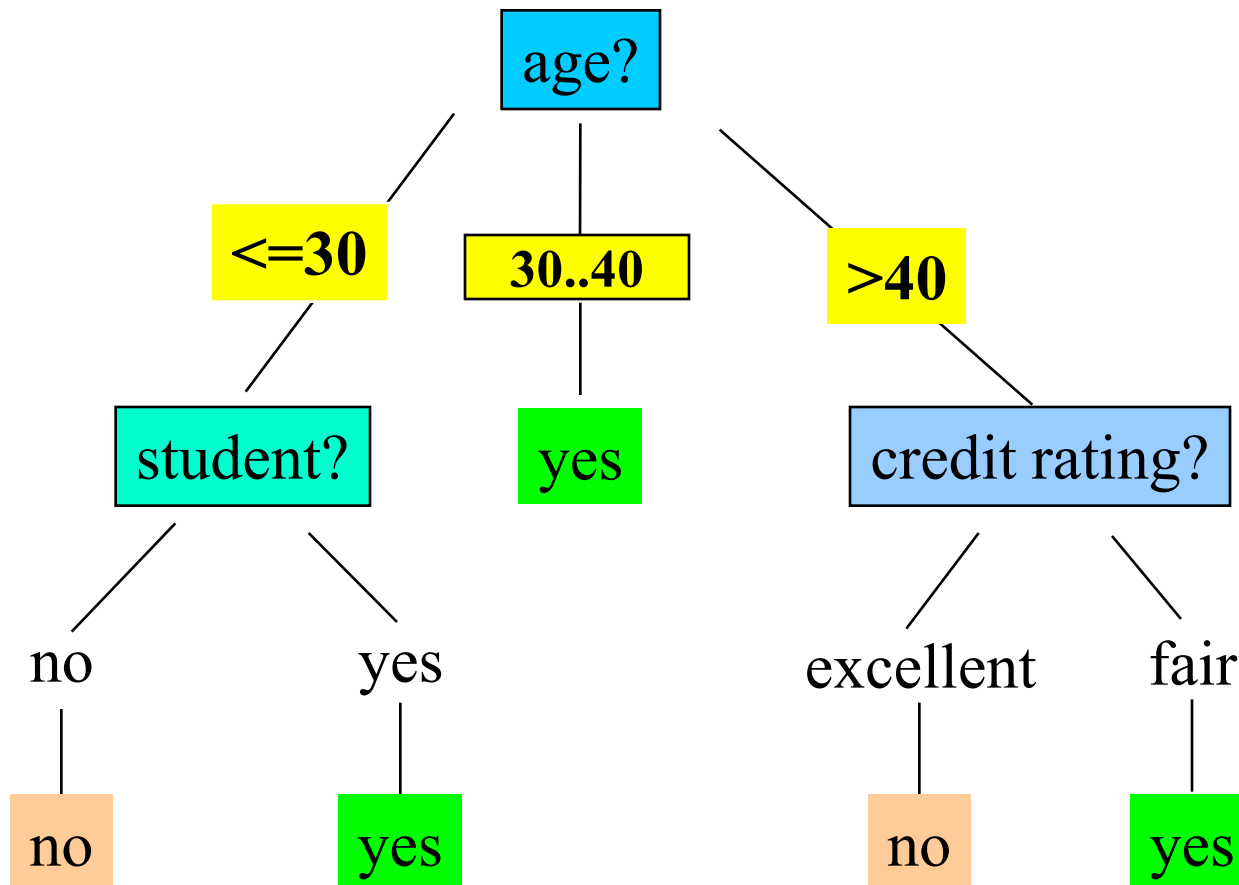
- **Computationally expensive:** Building decision trees can be computationally expensive, particularly when analyzing a large data set with many continuous variables.
- **Difficult to optimize:** Generating a useful decision tree automatically can be challenging, since large and complex trees can be easily generated. Trees that are too small may not capture enough information. Generating the 'best' tree through optimization is difficult.

Training Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating
<=30	high	no	fair
<=30	high	no	excellent
31...40	high	no	fair
>40	medium	no	fair
>40	low	yes	fair
>40	low	yes	excellent
31...40	low	yes	excellent
<=30	medium	no	fair
<=30	low	yes	fair
>40	medium	yes	fair
<=30	medium	yes	excellent
31...40	medium	no	excellent
31...40	high	yes	fair
>40	medium	no	excellent

Output: A Decision Tree for “*buys_computer*”





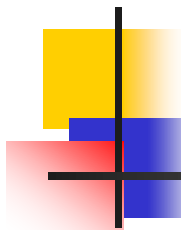
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
 - 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**)
- 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

Example

	Attributes			Class
#	Outlook	Company	Sailboat	Sail?
1	Sunny	Big	Small	Yes
2	Sunny	Med	Small	Yes
3	Sunny	Med	Big	Yes
4	Sunny	No	Small	Yes
5	Sunny	Big	Big	Yes
6	Rainy	No	Small	No
7	Rainy	Med	Small	No
8	Rainy	Big	Big	Yes
9	Rainy	No	Big	Yes
10	Rainy	Med	Big	No
11	Sunny	No	Big	?
12	Rainy	Big	Small	?

Another Example



#	Attribute				Class Play
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	no	N
2	sunny	hot	high	yes	N
3	overcast	hot	high	no	P
4	rainy	moderate	high	no	P
5	rainy	cold	normal	no	P
6	rainy	cold	normal	yes	N
7	overcast	cold	normal	yes	P
8	sunny	moderate	high	no	N
9	sunny	cold	normal	no	P
10	rainy	moderate	normal	no	P
11	sunny	moderate	normal	yes	P
12	overcast	moderate	high	yes	P
13	overcast	hot	normal	no	P
14	rainy	moderate	high	yes	N



Example

Triangles and Squares

#	Attribute			Shape
	Color	Outline	Dot	
1	green	dashed	no	triange
2	green	dashed	yes	triange
3	yellow	dashed	no	square
4	red	dashed	no	square
5	red	solid	no	square
6	red	solid	yes	triange
7	green	solid	no	square
8	green	dashed	no	triange
9	yellow	solid	yes	square
10	red	solid	no	square
11	green	solid	yes	square
12	yellow	dashed	yes	square
13	yellow	solid	no	square
14	red	dashed	yes	triange



Attribute Selection Measure

- **Information gain** (ID3/C4.5)
 - All attributes are assumed to be categorical
 - Can be modified for continuous-valued attributes
- **Gini index** (IBM IntelligentMiner)
 - All attributes are assumed continuous-valued
 - Assume there exist several possible split values for each attribute
 - May need other tools, such as clustering, to get the possible split values
 - Can be modified for categorical attributes

Attribute Selection Measures

Is a heuristic measure for **selecting the splitting criterion** that **best separates** a given data partition D , of class-labeled training tuples **into individual classes**.

- If we split the D into smaller partitions according to the outcomes of the splitting criterion, then ideally, each partition will be **PURE**.
- Therefore, **best splitting criterion** is needed.
- It is also known as **splitting rules**, as they determine how the tuples at a given node are to be split.
- It provides a ranking for each attribute, describing the given tuples.
- The attributes having **best score** is chosen as **splitting attribute** for the given tuples.

Example

1 Class-labeled training tuples from the *AllElectronics* customer database.

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
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

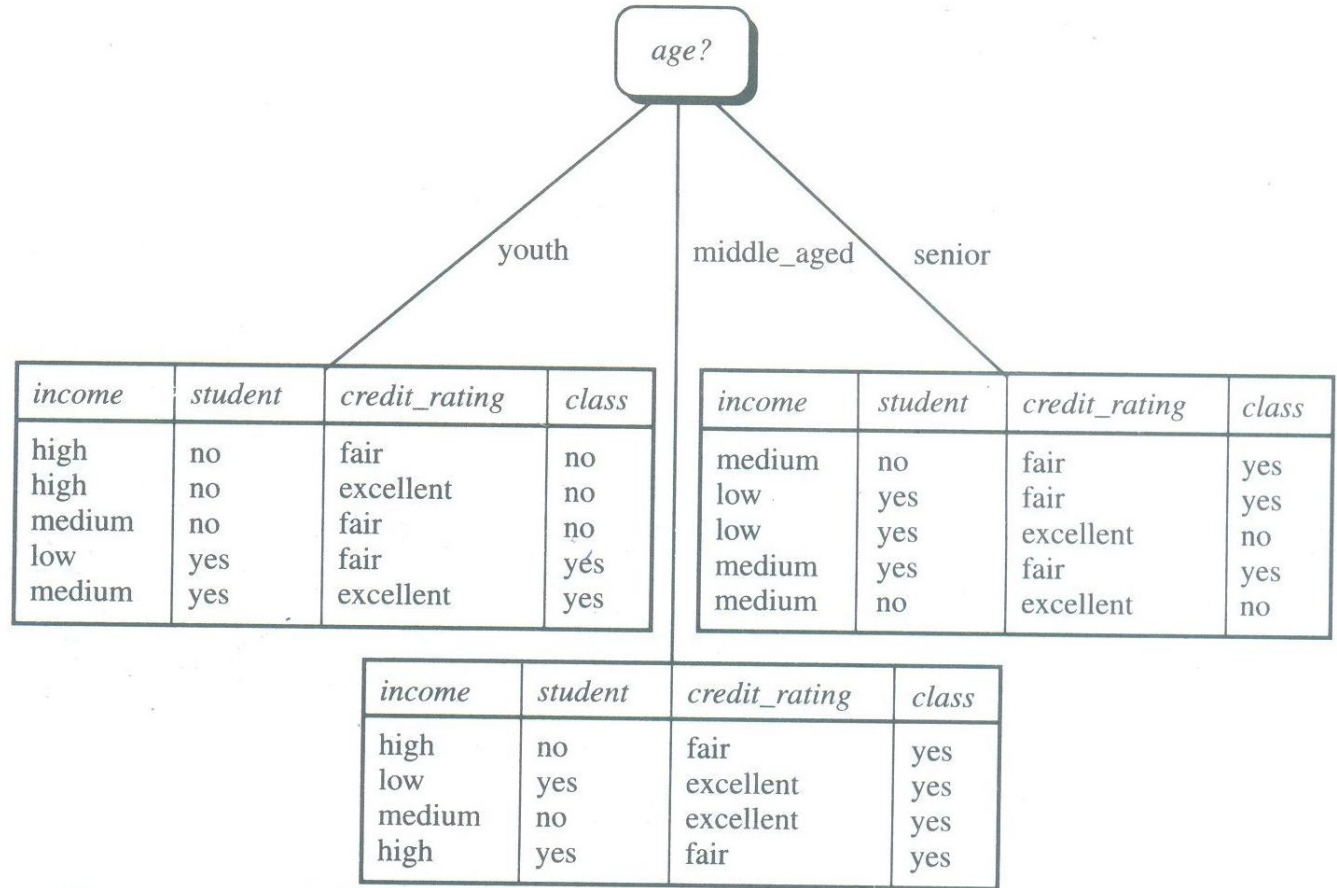
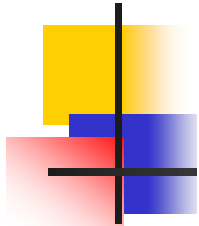


Figure 6.5 The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of *age*. The tuples are shown partitioned accordingly.

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, P and N
 - Let the set of examples S contain p elements of class P and n elements of class N
 - The amount of information, needed to decide if an arbitrary example in S belongs to P or N is defined as

$$I(p, n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that using attribute A a set S will be partitioned into sets $\{S_1, S_2, \dots, S_v\}$
 - If S_i contains p_i examples of P and n_i examples of N , the **entropy**, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on A

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- $I(p, n) = I(9, 5) = 0.940$
- Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
≤ 30	2	3	0.971
30...40	4	0	0
> 40	3	2	0.971

$$E(\text{age}) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

$$\text{Gain}(\text{age}) = I(p, n) - E(\text{age})$$

Similarly

$$\text{Gain}(\text{income}) = 0.029$$

$$\text{Gain}(\text{student}) = 0.151$$

$$\text{Gain}(\text{credit_rating}) = 0.048$$

Gini Index (IBM IntelligentMiner)

- If a data set T contains examples from n classes, gini index, $gini(T)$ is defined as

$$gini(T) = 1 - \sum_{j=1}^n p_j^2$$

where p_j is the relative frequency of class j in T .

- If a data set T is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the gini index of the split data contains examples from n classes, the gini index $gini(T)$ is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Extracting Classification Rules from Trees

- Represent the knowledge in the form of **IF-THEN** rules
- 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 node holds the class prediction
- Rules are easier for humans to understand
- Example

IF *age* = " ≤ 30 " AND *student* = "*no*" THEN *buys_computer* = "*no*"
IF *age* = " ≤ 30 " AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
IF *age* = " $31 \dots 40$ " THEN *buys_computer* = "*yes*"
IF *age* = " > 40 " AND *credit_rating* = "*excellent*" THEN
buys_computer = "*yes*"
IF *age* = " > 40 " AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"



Avoid Overfitting in Classification

- The generated tree may overfit the training data
 - Too many branches, some may reflect anomalies due to noise or outliers
 - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
 - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
 - Difficult to choose an appropriate threshold
 - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
 - Use a set of data different from the training data to decide which is the “best pruned tree”



Approaches to Determine the Final Tree Size

- Separate training (2/3) and testing (1/3) sets
- Use cross validation, e.g., 10-fold cross validation
- Use all the data for training
 - but apply a **statistical test** (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle:
 - halting growth of the tree when the encoding is minimized



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 decision tree induction in data mining?
 - 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



Scalable Decision Tree Induction Methods in Data Mining Studies

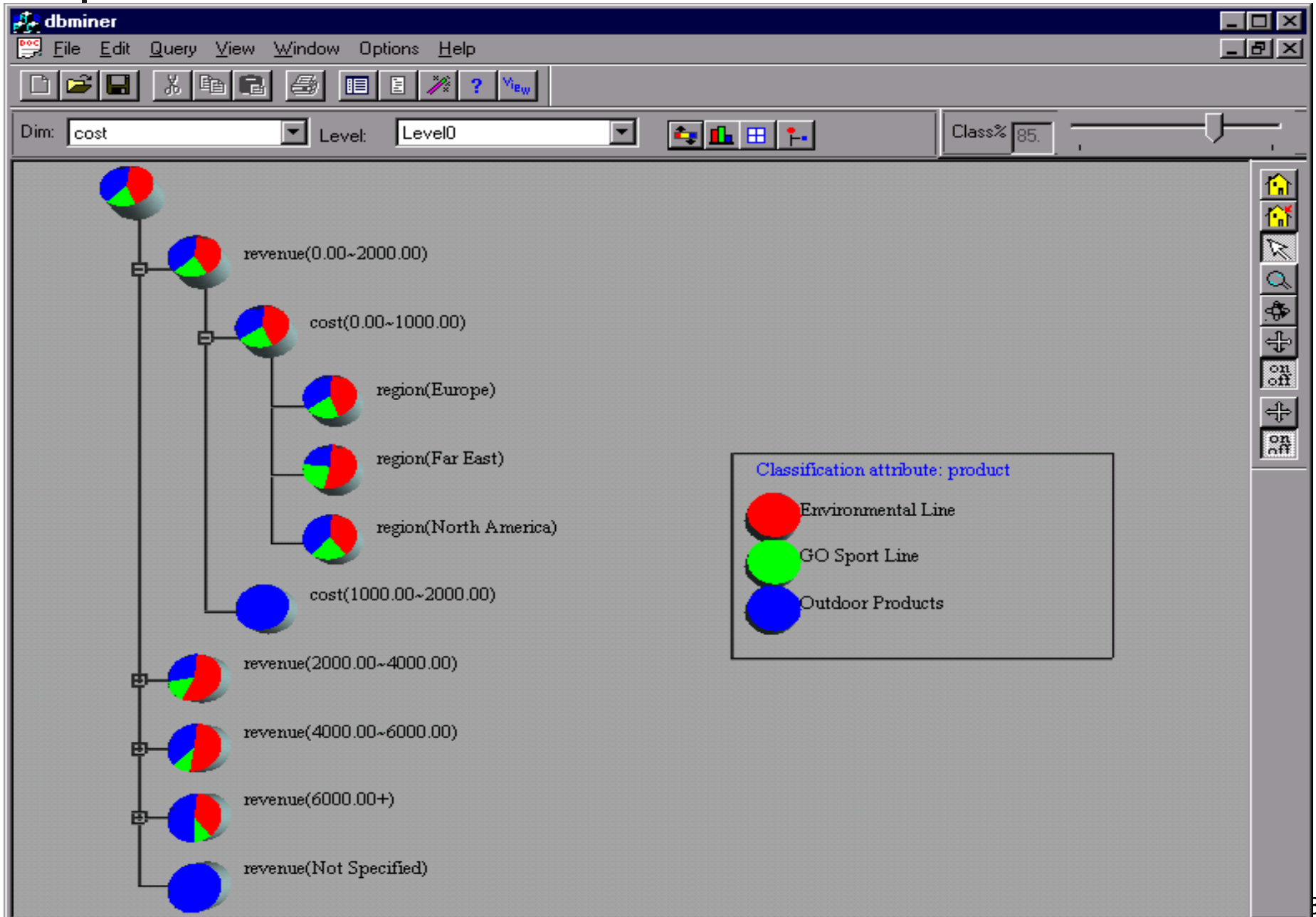
- **SLIQ** (EDBT'96 — Mehta et al.)
 - builds an index for each attribute and only class list and the current attribute list reside in memory
- **SPRINT** (VLDB'96 — J. Shafer et al.)
 - constructs an attribute list data structure
- **PUBLIC** (VLDB'98 — Rastogi & Shim)
 - integrates tree splitting and tree pruning: stop growing the tree earlier
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
 - separates the scalability aspects from the criteria that determine the quality of the tree
 - builds an AVC-list (attribute, value, class label)



Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al'97).
- Classification at primitive concept levels
 - E.g., precise temperature, humidity, outlook, etc.
 - Low-level concepts, scattered classes, bushy classification-trees
 - Semantic interpretation problems.
- Cube-based multi-level classification
 - Relevance analysis at multi-levels.
 - Information-gain analysis with dimension + level.

Presentation of Classification Results





Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- **Bayesian Classification**
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary



Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- 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 Classification: Example

- The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high.
- Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either **GREEN** or **RED**. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently existing objects.



Bayesian Classification: Example

- Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.



Thus, we can write:

$$\text{Prior probability for GREEN} \propto \frac{\text{Number of GREEN objects}}{\text{Total number of objects}}$$

$$\text{Prior probability for RED} \propto \frac{\text{Number of RED objects}}{\text{Total number of objects}}$$

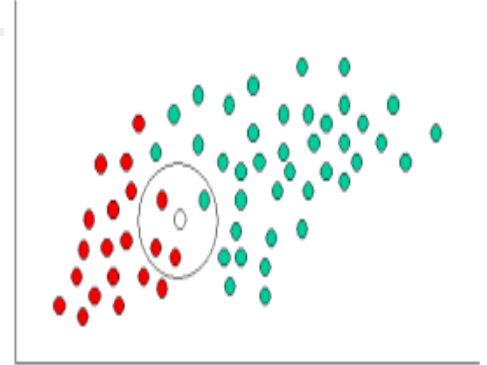
Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:

$$\text{Prior probability for GREEN} \propto \frac{40}{60}$$

$$\text{Prior probability for RED} \propto \frac{20}{60}$$

Bayesian Classification: Example

Having formulated our prior probability, we are now ready to classify a new object (WHITE circle). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:



$$\text{Likelihood of } X \text{ given GREEN} \propto \frac{\text{Number of GREEN in the vicinity of } X}{\text{Total number of GREEN cases}}$$

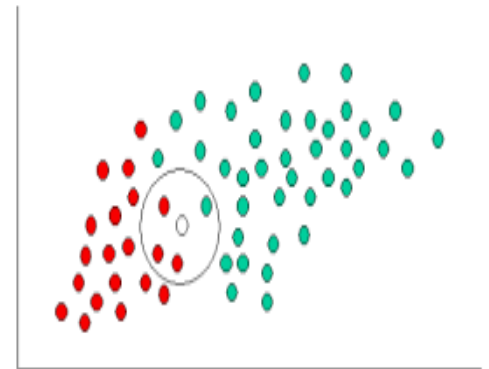
$$\text{Likelihood of } X \text{ given RED} \propto \frac{\text{Number of RED in the vicinity of } X}{\text{Total number of RED cases}}$$

Bayesian Classification: Example

- From the illustration above, it is clear that Likelihood of X given GREEN is smaller than Likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones. Thus:

$$\text{Probability of } X \text{ given GREEN} \propto \frac{1}{40}$$

$$\text{Probability of } X \text{ given RED} \propto \frac{3}{20}$$



Bayesian Classification: Example

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).

Posterior probability of X being GREEN \propto

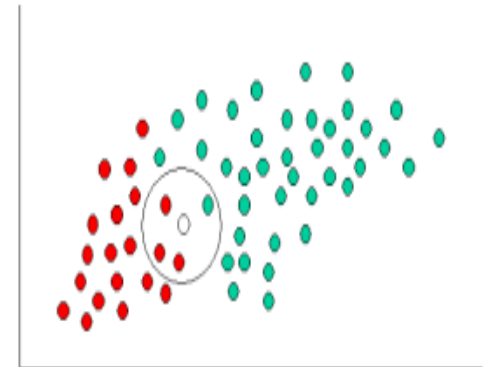
Prior probability of GREEN \times Likelihood of X given GREEN

$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

Posterior probability of X being RED \propto

Prior probability of RED \times Likelihood of X given RED

$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$



Finally, we classify X as RED since its class membership achieves the largest posterior probability.



Bayesian Theorem

- Given training data D , *posteriori probability of a hypothesis h* , $P(h|D)$ follows the Bayes theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost



Bayesian classification

- The classification problem may be formalized using **a-posteriori probabilities**:
- $P(C|X)$ = prob. that the sample tuple $X = \langle x_1, \dots, x_k \rangle$ is of class C .
- E.g. $P(\text{class}=\text{N} \mid \text{outlook}=\text{sunny}, \text{windy}=\text{true}, \dots)$
- Idea: assign to sample X the class label C such that $P(C|X)$ is maximal



Estimating a-posteriori probabilities

- **Bayes theorem:**

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative freq of class C samples
- C such that $P(C|X)$ is maximum =
C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C)$ is unfeasible!



Naïve Bayesian Classification

- Naïve assumption: **attribute independence**

$$P(x_1, \dots, x_k | C) = P(x_1 | C) \cdot \dots \cdot P(x_k | C)$$

- If i-th attribute is **categorical**:
P(x_i|C) is estimated as the relative freq of samples having value x_i as i-th attribute in class C
- If i-th attribute is **continuous**:
P(x_i|C) is estimated thru a Gaussian density function
- Computationally easy in both cases

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$$P(p) = 9/14$$

$$P(n) = 5/14$$

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{true} p) = 3/9$	$P(\text{true} n) = 3/5$
$P(\text{false} p) = 6/9$	$P(\text{false} n) = 2/5$



Play-tennis example: classifying X

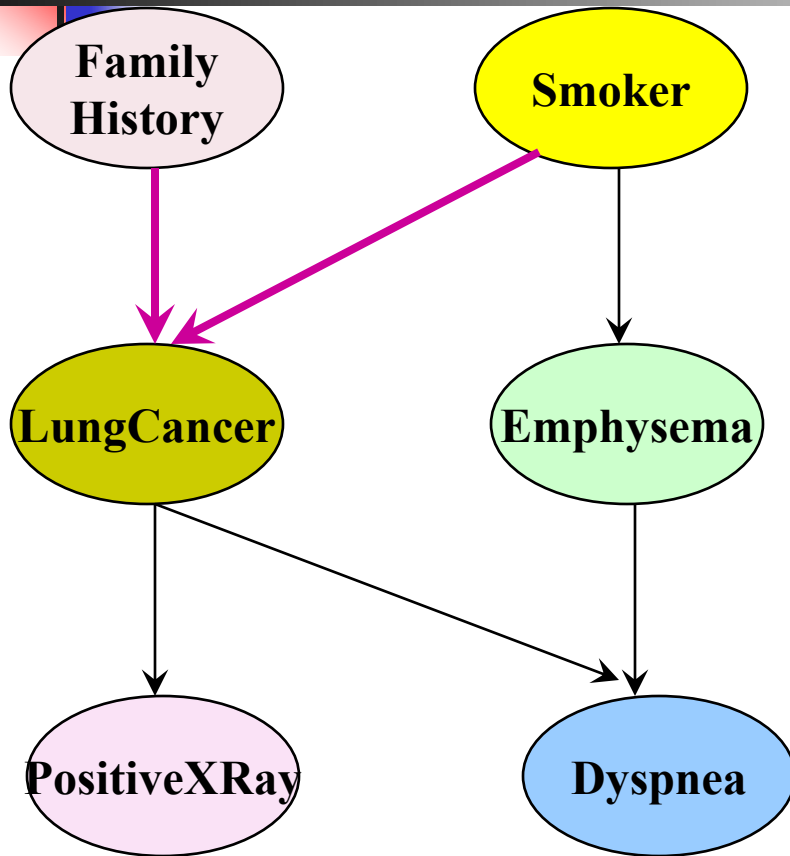
- An unseen sample $X = \langle \text{rain}, \text{hot}, \text{high}, \text{false} \rangle$
- $P(X|p) \cdot P(p) =$
 $P(\text{rain}|p) \cdot P(\text{hot}|p) \cdot P(\text{high}|p) \cdot P(\text{false}|p) \cdot P(p) =$
 $3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$
- $P(X|n) \cdot P(n) =$
 $P(\text{rain}|n) \cdot P(\text{hot}|n) \cdot P(\text{high}|n) \cdot P(\text{false}|n) \cdot P(n) =$
 $2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286$
- Sample **X** is classified in class **n** (don't play)



The independence hypothesis...

- ... makes computation possible
- ... yields optimal classifiers when satisfied
- ... but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
 - **Bayesian networks**, that combine Bayesian reasoning with causal relationships between attributes
 - **Decision trees**, that reason on one attribute at the time, considering most important attributes first

Bayesian Belief Networks (I)



(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

The conditional probability table for the variable LungCancer

Bayesian Belief Networks



Bayesian Belief Networks (II)

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
- Several cases of learning Bayesian belief networks
 - Given both network structure and all the variables: easy
 - Given network structure but only some variables
 - When the network structure is not known in advance



Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- **Classification by backpropagation**
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary



Neural Networks

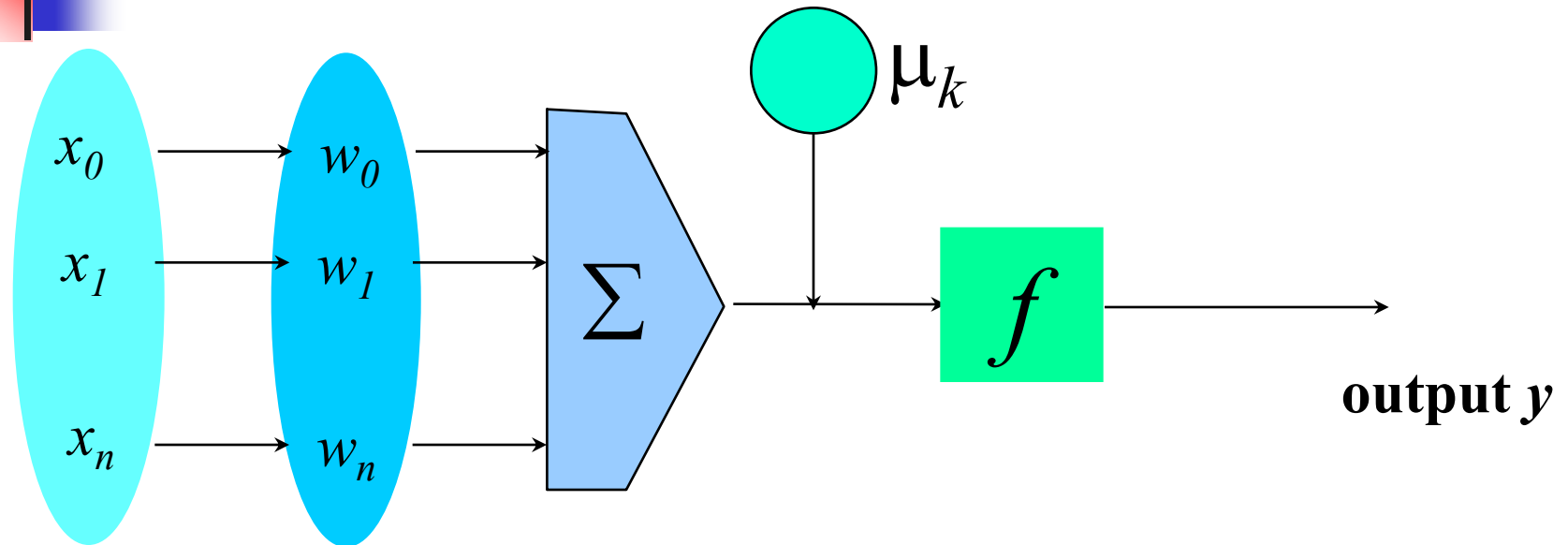
- Advantages

- prediction accuracy is generally high
- robust, works when training examples contain errors
- output may be discrete, real-valued, or a vector of several discrete or real-valued attributes
- fast evaluation of the learned target function

- Criticism

- long training time
- difficult to understand the learned function (weights)
- not easy to incorporate domain knowledge

A Neuron

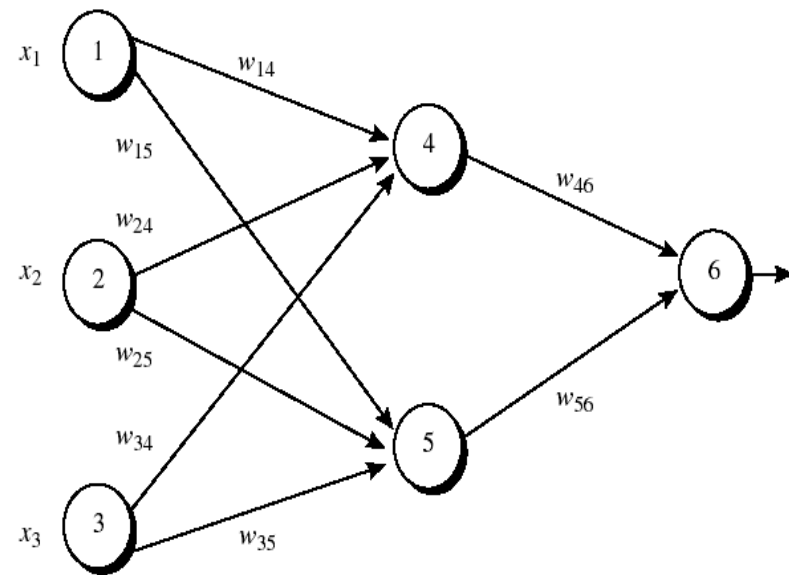


Input **weight** **weighted** **Activation**
vector x **vector w** **sum** **function**

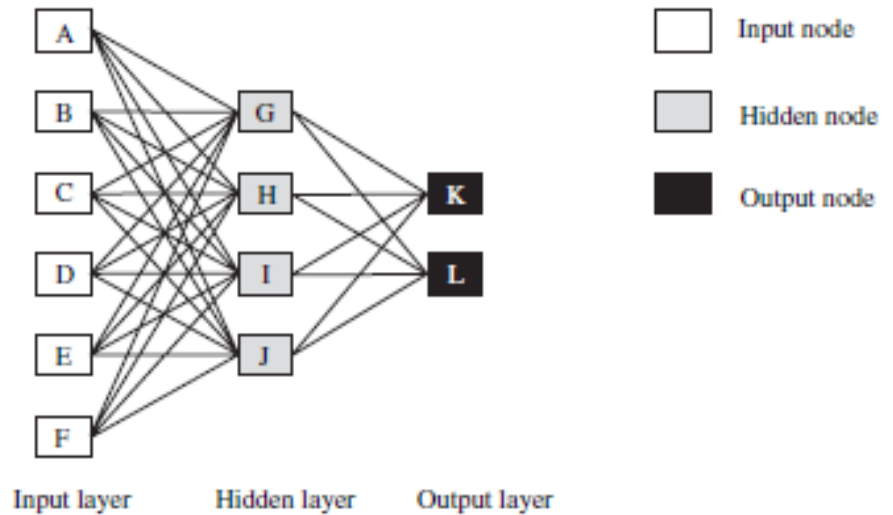
- The n -dimensional input vector x is mapped into variable y by means of the scalar product and a nonlinear function mapping

Network Training

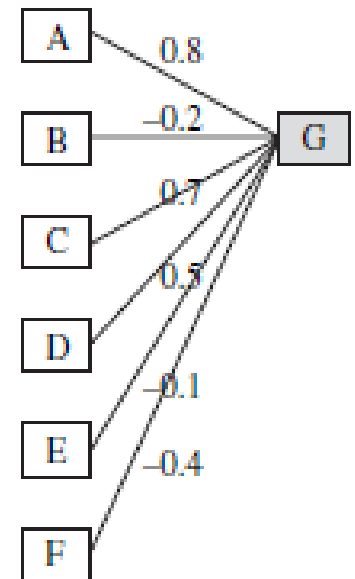
- The ultimate objective of training
 - obtain a set of weights that makes almost all the tuples in the training data classified correctly
- Steps
 - Initialize weights with random values
 - Feed the input tuples into the network one by one
 - For each unit
 - Compute the net input to the unit as a linear combination of all the inputs to the unit
 - Compute the output value using the activation function
 - Compute the error
 - Update the weights and the bias



Neural Network Layers



- ✓ Input layer contains a set of nodes A, B, C, D, E, and F (six descriptor variables/ one instance)
- ✓ Two output layers K and L
- ✓ Four hidden layers G, H, I and J
- ✓ Each node is connected to all nodes in the layers adjacent to the node.
- ✓ Every connection has a number/weight associated with it.
- ✓ Prior to learning, the weights are assigned random values usually in the range +1 to -1.

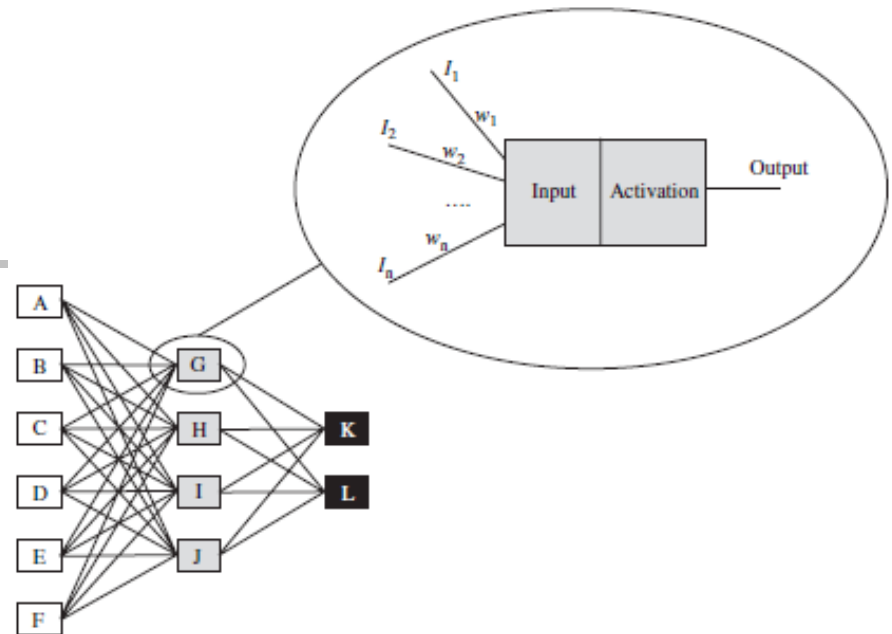


Node Calculation

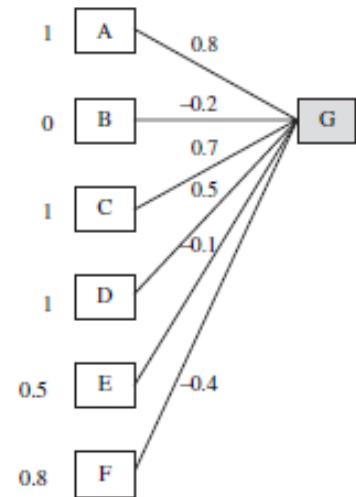
- Each node in the neural network calculates a single output value based on a set of input values (I_1 to I_n).
- Each input connection has a weight and is assigned a value. The total input of the node is calculated using these weights and values.
- For example, the following formula for calculating the combined input is often used:

$$Input = \sum_{j=1}^n I_j w_j$$

where I_j are the individual input values and w_j are the individual weights.



Descriptor variables						Response variables	
V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈
1	0	1	1	0.5	0.8	0.4	1



Node Calculation and Learning

The combined input value for node G is calculated using the input values and the weights:

$$Input_G = \sum_{j=1}^n I_j w_j$$

$$Input_G = (1 \times 0.8) + (0 \times -0.2) + (1 \times 0.7) + (1 \times 0.5) + (0.5 \times -0.1) + (0.8 \times -0.4)$$

$$Input_G = 1.63$$

This combined input value is now processed further using an activation function. This function will generate the output for the node. Common activation functions include:

Sigmoid : $Output = \frac{1}{1 + e^{-Input}}$

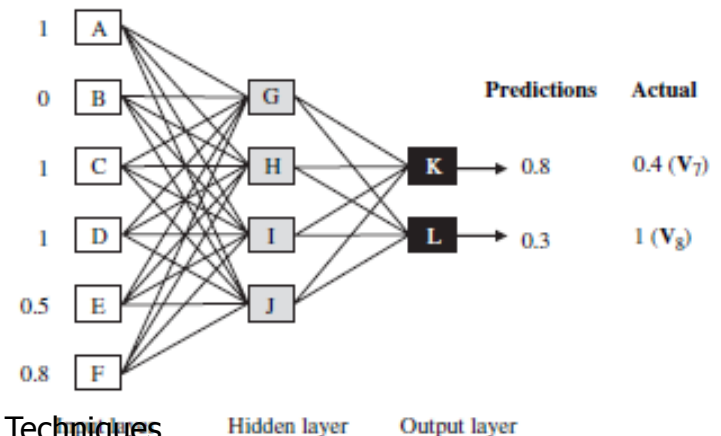
Tanh : $Output = \frac{e^{Input} - e^{-Input}}{e^{Input} + e^{-Input}}$

The following output from the neural network node G would be generated:

$$Output_G = \frac{1}{1 + e^{-Input_G}}$$

$$Output_G = \frac{1}{1 + e^{-1.63}}$$

$$Output_G = 0.84$$



Back propagation

- One of the most commonly used techniques for learning in neural networks is called back propagation.
- In order for the weights of the neural network connections to be adjusted, an error first needs to be calculated between the predicted response and the actual response.
- The following formula is commonly used for the output layer:

$$Error_i = Output_i(1 - Output_i)(Actual_i - Output_i)$$

where $Error_i$ is the error resulting from node i , $Output_i$ is the predicted response value and $Actual_i$ is the actual response value.

Error Calculation

- For example, the errors calculated for nodes K and L are:

Node K :

$$Error_K = Output_K(1 - Output_K)(Actual_K - Output_K)$$

$$Error_K = 0.8 \times (1 - 0.8) \times (0.4 - 0.8)$$

$$Error_K = -0.064$$

Node L :

$$Error_L = Output_L(1 - Output_L)(Actual_L - Output_L)$$

$$Error_L = 0.3 \times (1 - 0.3) \times (1 - 0.3)$$

$$Error_L = 0.147$$

- Once the error has been calculated for the output layer, it can now be backpropagated, that is, the error can be passed back through the neural network. To calculate an error value for the hidden layers, the following calculation is commonly used:

$$Error_i = Output_i(1 - Output_i) \sum_{j=1}^n Error_j w_{ij}$$

where $Error_i$ is the error resulting from the hidden node, $Output_i$ is the value of the output from the hidden node, $Error_j$ is the error already calculated for the j^{th} node connected to the output and w_{ij} is the weight on this connection.

Error Calculation

After calculating the error for nodes K and L , the error of the hidden layer can be calculated. Node G is used as an example for a hidden layer error calculation as shown below.

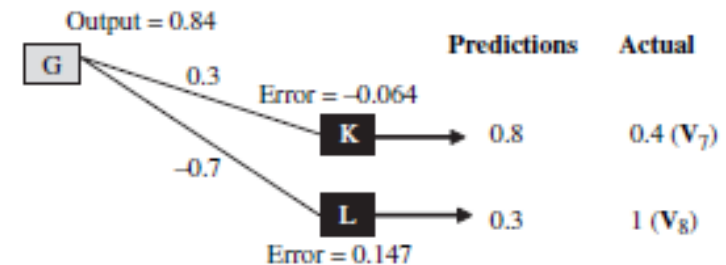
Node G :

$$Error_G = Output_G(1 - Output_G)((Error_K \times w_{GK}) + (Error_L \times w_{GL}))$$

$$Error_G = 0.84 \times (1 - 0.84) \times ((-0.064 \times 0.3) + (0.147 \times -0.7))$$

$$Error_G = 0.0112$$

An error should be calculated for all output and hidden layer nodes. Errors for hidden layer nodes use errors from the nodes their output is attached to, which have already been calculated.



Error Calculation

Once the error has been propagated throughout the neural network, the error values can be used to adjust the weights of the connections using the formula:

$$w_{ij} = w_{ij} + l \times Error_j \times Output_i$$

- Where w_{ij} is the weight of the connection between nodes i and j , $Error_j$ is the calculated error for node j , $Output_i$ is the computed output from node i , and l is the predefined learning rate. This takes a value between 0 and 1.
- To calculate the new weight for the connection between G and K where the learning rate (l) has been set to 0.2, the following formula is used:

$$w_{GK} = w_{GK} + l \times Error_K \times Output_G$$

$$w_{GK} = 0.3 + 0.2 \times -0.064 \times 0.84$$

$$w_{GK} = 0.276$$

- In this example, the weight has been adjusted lower. The remaining weights in the network are adjusted and the process continues with another example presented to the neural network causing the weights of all connections in the network to be adjusted based on the calculated error values.

Multi-Layer Perceptron

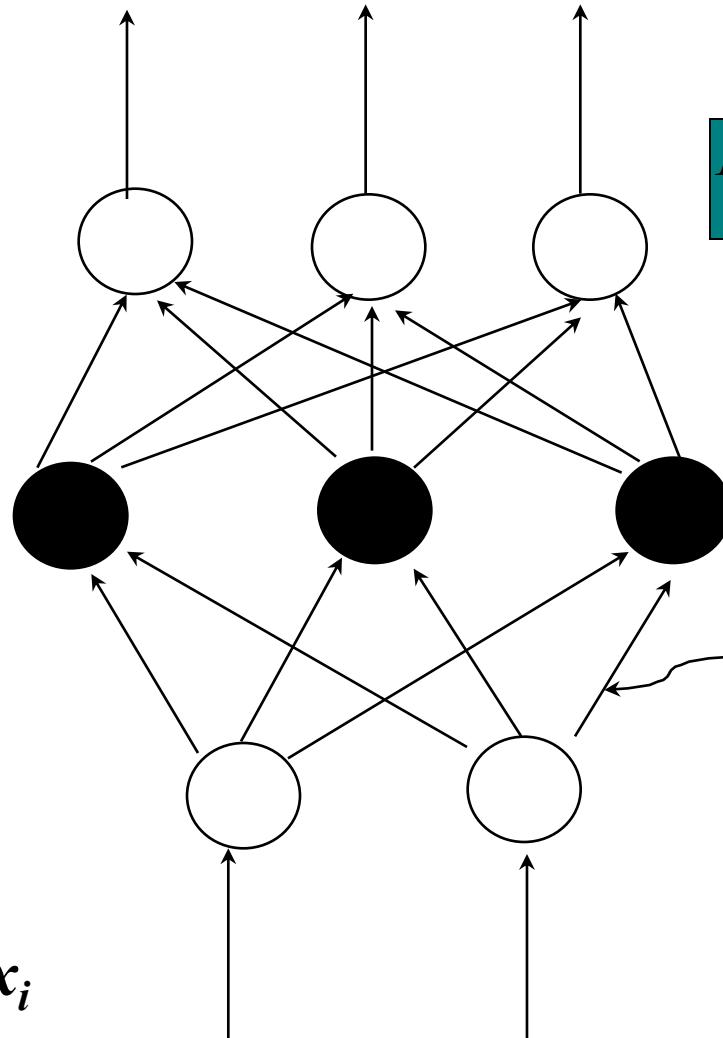
Output vector

Output nodes

Hidden nodes

Input nodes

Input vector: x_i



$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

$$\theta_j = \theta_j + (l)Err_j$$

$$w_{ij} = w_{ij} + (l)Err_j O_i$$

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

$$O_j = \frac{1}{1 + e^{-I_j}}$$

$$I_j = \sum_i w_{ij} O_i + \theta_j$$



Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary



Association-Based Classification

- Several methods for association-based classification
 - ARCS: Quantitative association mining and clustering of association rules (Lent et al'97)
 - It beats C4.5 in (mainly) scalability and also accuracy
 - Associative classification: (Liu et al'98)
 - It mines high support and high confidence rules in the form of "cond_set => y", where y is a class label
 - CAEP (Classification by aggregating emerging patterns) (Dong et al'99)
 - Emerging patterns (EPs): the itemsets whose support increases significantly from one class to another
 - Mine Eps based on minimum support and growth rate



Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- **Other Classification Methods**
- Prediction
- Classification accuracy
- Summary



Other Classification Methods

- k-nearest neighbor classifier
- case-based reasoning
- Genetic algorithm
- Rough set approach
- Fuzzy set approaches



Classification: Eager & Lazy Learners

- Decision Tree classifier is an example of an “eager learner”
- Because they are designed to learn a model that maps the input attributes to the class label as soon as the training data becomes available
- An opposite strategy would be to delay the process of modeling the training data until it is needed to classify the test examples
- LAZY Learners



Classification: Eager & Lazy Learners

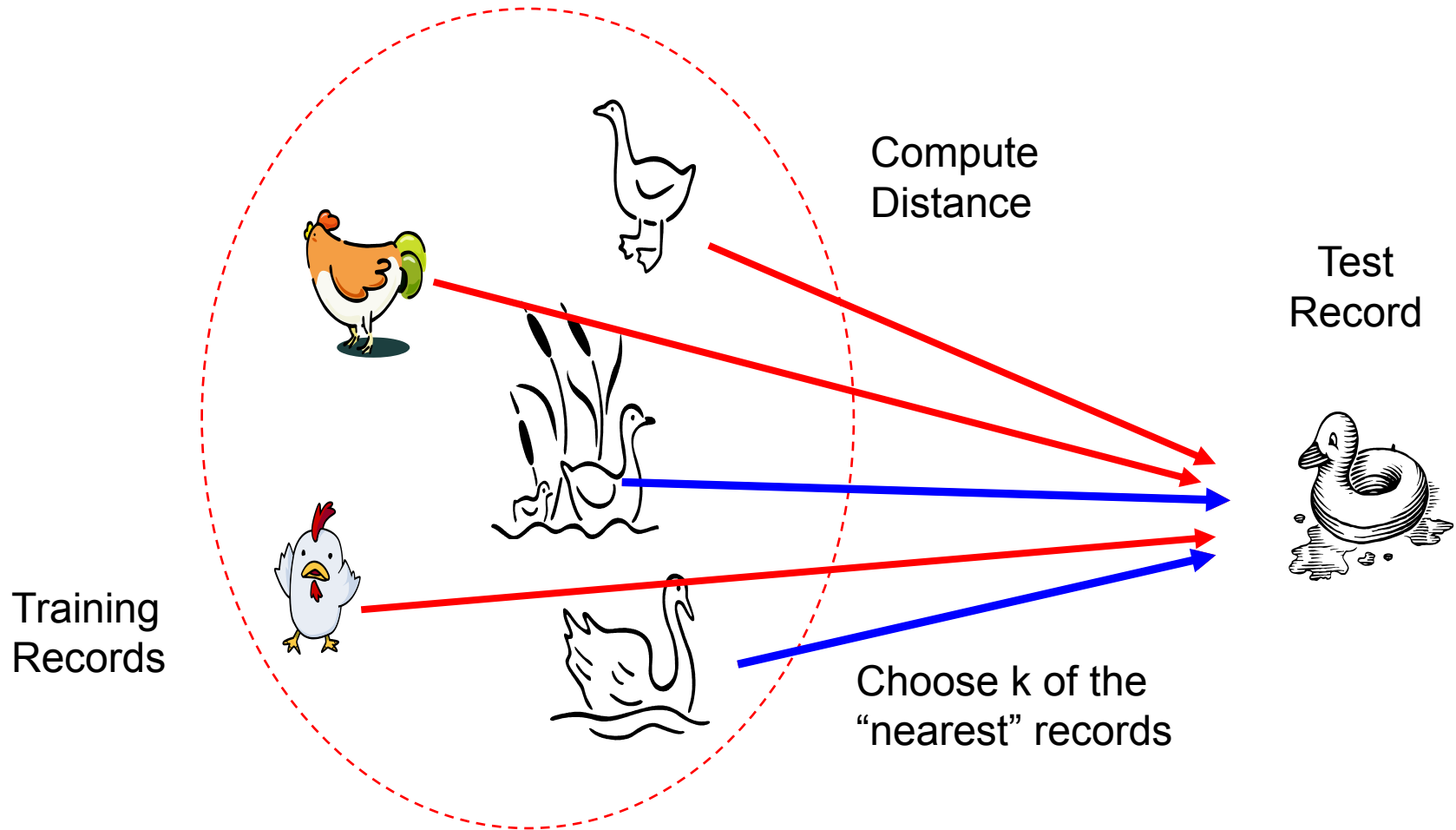
- KNN classifier is an example of lazy learner, which memorizes the entire training data & performs the classification only if the attributes of a test instance matches exactly with one of the training examples
- Drawback: Cannot classify a new instance if it does not match any training example
- To overcome this drawback, we find all training examples that are relatively 'similar' to the test example
- Examples, which are known as 'Nearest Neighbors' can be used to determine the class label of the test example
- *If it walks like a duck, quacks like a duck, and looks like a duck, then it is probably a duck*



Remarks on Lazy vs. Eager Learning

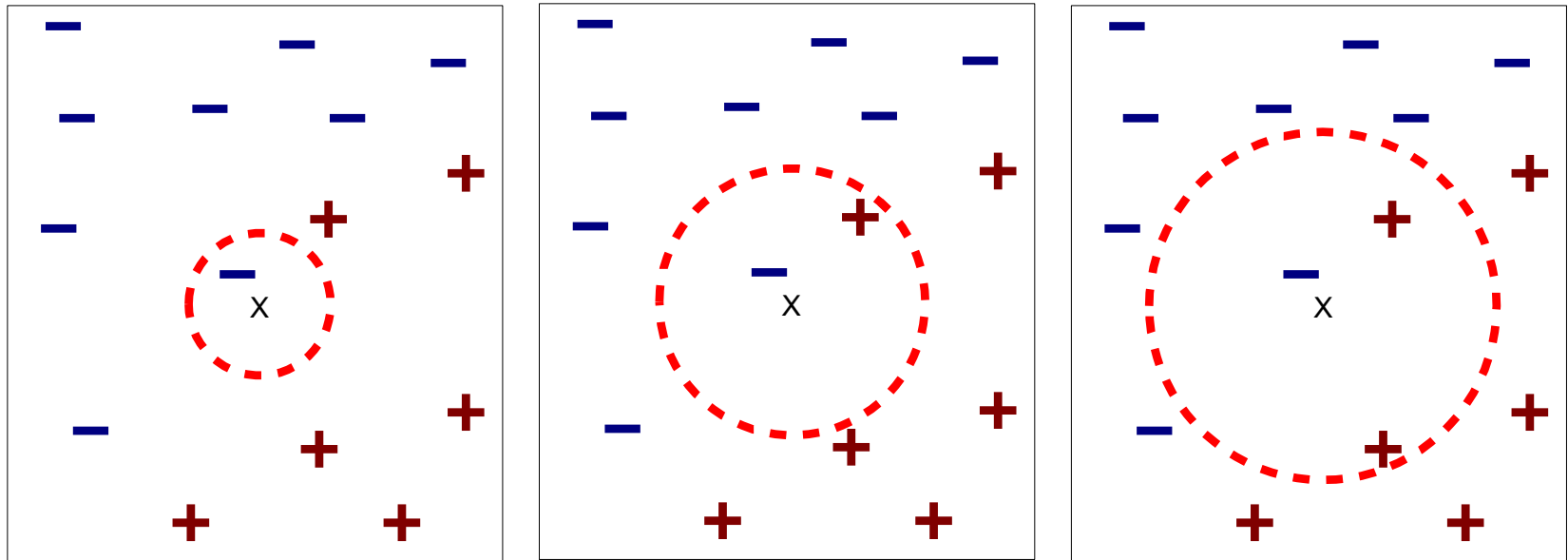
- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences
 - Lazy method may consider query instance xq when deciding how to generalize beyond the training data D
 - Eager method cannot since they have already chosen global approximation when seeing the query
- Efficiency: Lazy - less time training but more time predicting
- Accuracy
 - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
 - Eager: must commit to a single hypothesis that covers the entire instance space

Nearest Neighbor Classifiers



KNN - Definition

KNN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor
K-nearest neighbors of a record x are data points that have the k smallest distance to x



KNN – different names

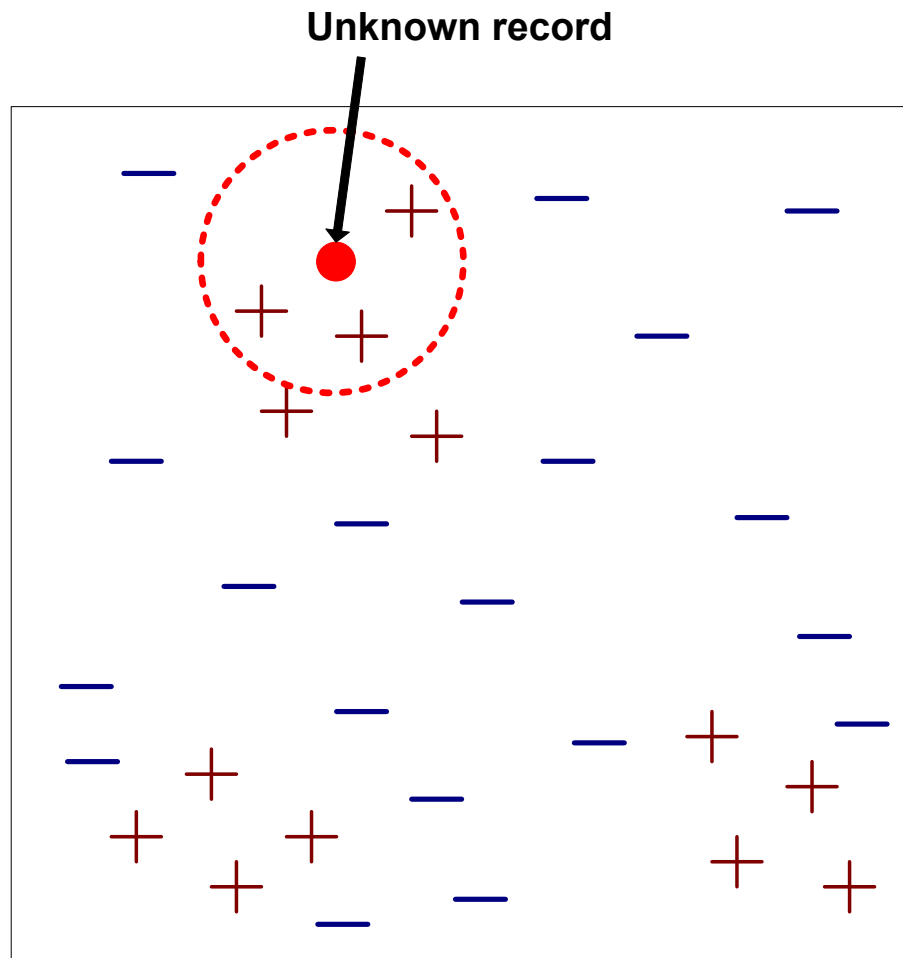
- K-Nearest Neighbors
- Memory-Based Reasoning
- Example-Based Reasoning
- Instance-Based Learning
- Case-Based Reasoning
- Lazy Learning



Classification: Nearest Neighbors

- Each test example is represented as a point in a d -dimensional space
- For each test example we use a proximity measure
- k -nearest neighbors of a given example z refer to the k points that are closest to z
- Place items in class to which they are “closest”.
- Must determine distance between an item and a class.
- Classes represented by
 - ***Centroid***: Central value.
 - ***Medoid***: Representative point.
 - Individual points
- Algorithm: KNN

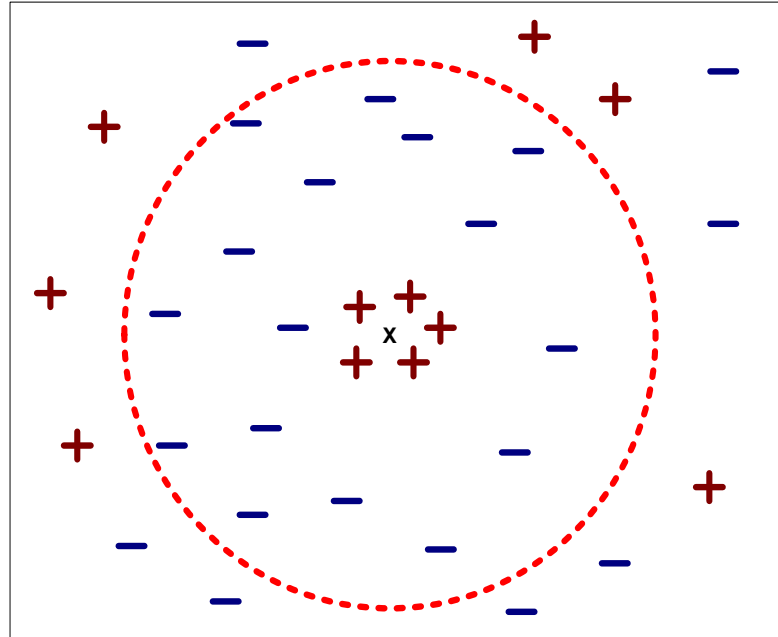
Nearest-Neighbor Classifiers



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

Nearest Neighbor Classification...

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





K Nearest Neighbors

- K Nearest Neighbors
 - Advantage
 - Nonparametric architecture
 - Simple
 - Powerful
 - Requires no training time
 - Disadvantage
 - Memory intensive
 - Classification/estimation is slow



Case-Based Reasoning

- Also uses: lazy evaluation + analyze similar instances
- Difference: Instances are not “points in a Euclidean space”
- Example: Water faucet problem in CADET (Sycara et al'92)
- Methodology
 - Instances represented by rich symbolic descriptions (e.g., function graphs)
 - Multiple retrieved cases may be combined
 - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Research issues
 - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

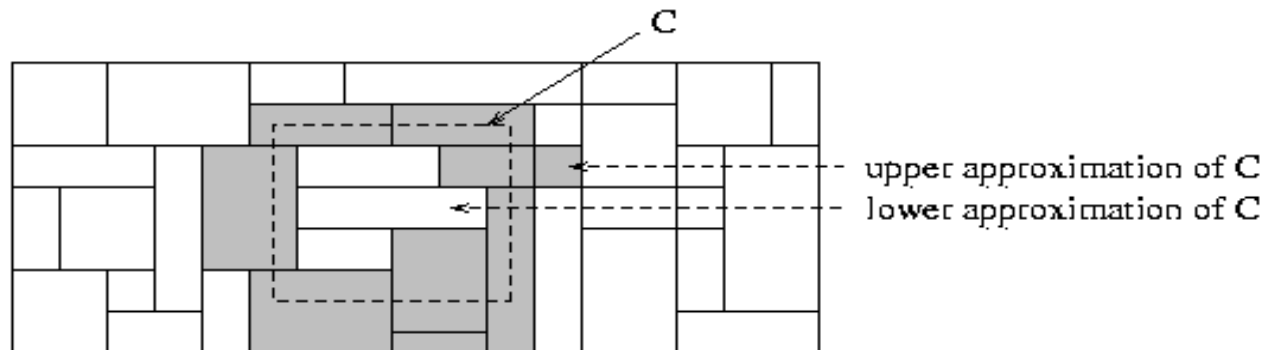


Genetic Algorithms

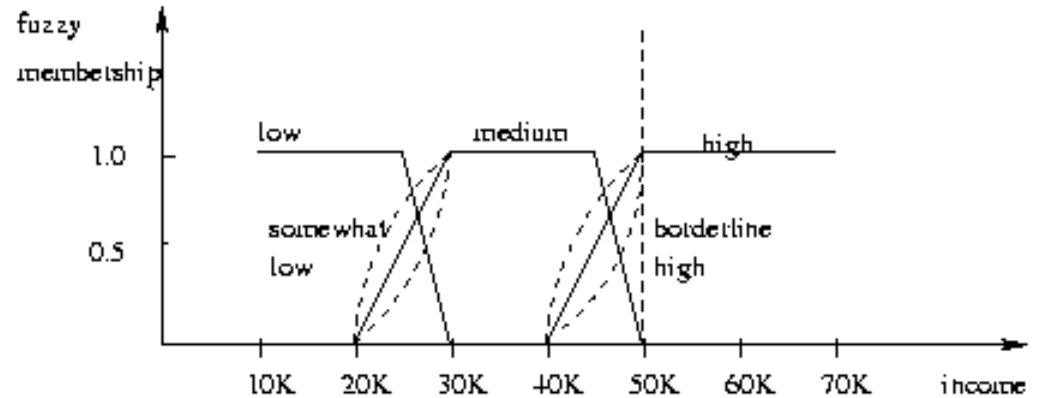
- GA: based on an analogy to biological evolution
- Each rule is represented by a string of bits
- An initial population is created consisting of randomly generated rules
 - e.g., IF A_1 and Not A_2 then C_2 can be encoded as 100
- Based on the notion of survival of the fittest, a new population is formed to consists of the fittest rules and their offsprings
- The fitness of a rule is represented by its classification accuracy on a set of training examples
- Offsprings are generated by crossover and mutation

Rough Set Approach

- Rough sets are used to approximately or “roughly” define equivalent classes
- A rough set for a given class C is approximated by two sets: a **lower approximation** (certain to be in C) and an **upper approximation** (cannot be described as not belonging to C)
- Finding the minimal subsets (reducts) of attributes (for feature reduction) is NP-hard but a discernibility matrix is used to reduce the computation intensity



Fuzzy Set Approaches



- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using **fuzzy membership graph**)
- Attribute values are converted to fuzzy values
 - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed



Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- **Prediction**
- Classification accuracy
- Summary



What Is Prediction?

- Prediction is similar to classification
 - First, construct a model
 - Second, use model to predict unknown value
 - Major method for prediction is regression
 - Linear and multiple regression
 - Non-linear regression
- Prediction is different from classification
 - Classification refers to predict categorical class label
 - Prediction models continuous-valued functions



Predictive Modeling in Databases

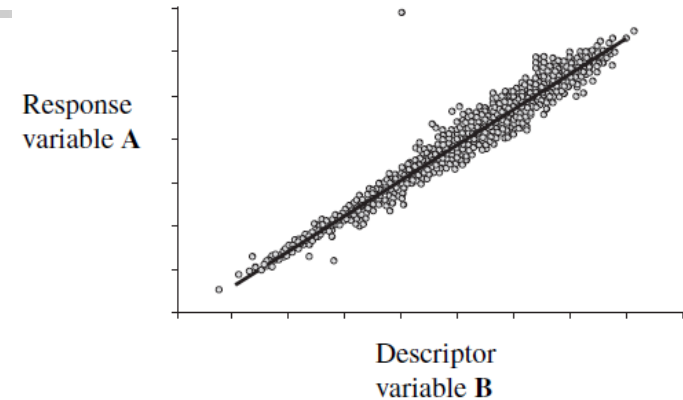
- Predictive modeling: Predict data values or construct generalized linear models based on the database data.
- One can only predict value ranges or category distributions
- Method outline:
 - Minimal generalization
 - Attribute relevance analysis
 - Generalized linear model construction
 - Prediction
- Determine the major factors which influence the prediction
 - Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis

Regress Analysis and Log-Linear Models in Prediction

- Linear regression: $Y = \alpha + \beta X$
 - Two parameters , α and β specify the line and are to be estimated by using the data at hand.
 - using the least squares criterion to the known values of $Y_1, Y_2, \dots, X_1, X_2, \dots$
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$.
 - Many nonlinear functions can be transformed into the above.
- Log-linear models:
 - The multi-way table of joint probabilities is approximated by a product of lower-order tables.
 - Probability: $p(a, b, c, d) = \alpha_{ab} \beta_{ac} \chi_{ad} \delta_{bcd}$

Simple Regression Model

- A simple regression model is a formula describing the relationship between one descriptor and one response variable.
- The analysis is sensitive to any outliers in data.
- Fig. shows the relation ship between a descriptor variable B and response variable A.
- Fig. shows high correlation between the two variables.
- As the descriptor variable B increases the response variable A increases at the same rate.



A straight line representing a model can be drawn through the center of the points.

A model that would predict values along this line would provide a good model.

A straight line can be represented as:

$y = a + bx$, where a is the point of intersection with the y-axis and b is the slope of the line.

Example

Table 7.8. Table of the customer's Income and Monthly Sales

Income (x)	Monthly Sales (y)
\$15,000.00	\$54.00
\$16,000.00	\$61.00
\$17,000.00	\$70.00
\$18,000.00	\$65.00
\$19,000.00	\$68.00
\$20,000.00	\$84.00
\$23,000.00	\$85.00
\$26,000.00	\$90.00
\$29,000.00	\$87.00
\$33,000.00	\$112.00
\$35,000.00	\$115.00
\$36,000.00	\$118.00
\$38,000.00	\$120.00
\$39,000.00	\$118.00
\$41,000.00	\$131.00
\$43,000.00	\$150.00
\$44,000.00	\$148.00
\$46,000.00	\$151.00
\$49,000.00	\$157.00
\$52,000.00	\$168.00
\$54,000.00	\$156.00
\$52,000.00	\$158.00
\$55,000.00	\$161.00
\$59,000.00	\$183.00
\$62,000.00	\$167.00
\$65,000.00	\$186.00
\$66,000.00	\$191.00

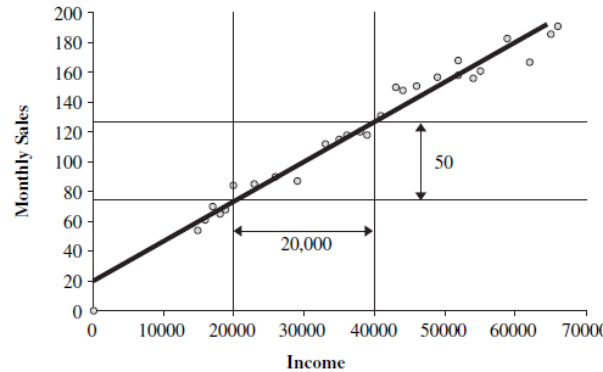
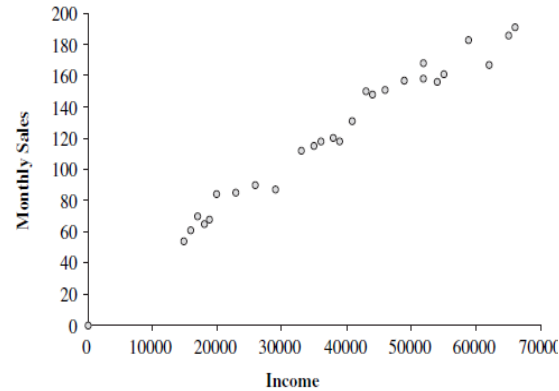


Figure 7.11. Scatterplot of the customer's Income vs Monthly Sales

Figure 7.12. Calculating the slope of the line

The point at which the line intercepts with the y-axis is noted (approximately 20) and the slope of the line is calculated (approximately 50/20,000 or 0.0025). For this data set an approximate formula for the relationship between Income and Monthly sales is:

$$\text{Monthly sales} = 20 + 0.0025 * \text{Income}$$

Once a formula for the straight line has been established, predicting values for the **y** response variable based on the **x** descriptor variable can be easily calculated.

Income refers to the yearly income of the customer and **Monthly sale** represents the amount of particular customer purchases for a month.

Example



- In this example, Monthly sales should only be predicted based on Income between \$15,000 and \$66,000. A prediction for a customer's Monthly sales based on their Income can be calculated. For a customer with an Income of \$31,000, the Monthly sales would be predicted as:
 - Monthly sales (predicted) = $20 + 0.0025 * \$31,000$
 - Monthly sales (predicted) = \$ 97.50

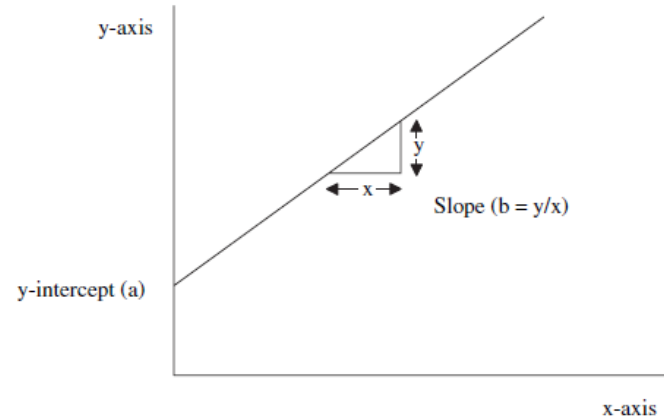
Least Square Method

- Parameters **a** and **b** can be derived manually by drawing a best guess line through the points in the scatterplot and then visually inspecting where the line crosses the y-axis (a) and measuring the slope (b) as previously described.

- The least squares method is able to calculate these parameters automatically. The formula for calculating a slope is:

$$b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

- Where x_i and y_i are the individual values for the descriptor variable (x_i) and the response (y_i). \bar{x} is the mean of the descriptor variable x and \bar{y} is the mean of the response variable y .
- The formula for calculating $a = \bar{y} - b\bar{x}$ except with the y-axis is:



Example

- Using the data from Table 7.8 , the slope and intercept are calculated using Table 7.9.
- The average Income is \$38,963 and the average Monthly sales is \$124.22.

$$\text{Slope } (b) = 17,435,222/6,724,962,963$$

$$\text{Slope } (b) = 0.00259$$

$$\text{Intercept } (a) = 124.22 - (0.00259 \times 38,963)$$

$$\text{Intercept } (a) = 23.31$$

Hence the formula is:

$$\text{Monthly sales} = 23.31 + 0.00259 \times \text{Income}$$

Table 7.9. Calculation of linear regression with least squares method

Income (x)	Monthly Sales (y)	$(x_i - \bar{x})$	$(y_i - \bar{y})$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
\$15,000.00	\$54.00	-23,963	-70.22	1,682,733	574,223,594
\$16,000.00	\$61.00	-22,963	-63.22	1,451,770	527,297,668
\$17,000.00	\$70.00	-21,963	-54.22	11,908,801	482,371,742
\$18,000.00	\$65.00	-20,963	-59.22	1,241,473	439,445,816
\$19,000.00	\$68.00	-19,963	-56.22	1,122,362	398,519,890
\$20,000.00	\$84.00	-18,963	-40.22	762,733	359,593,964
\$23,000.00	\$85.00	-15,963	-39.22	626,103	254,816,187
\$26,000.00	\$90.00	-12,963	-34.22	443,621	168,038,409
\$29,000.00	\$87.00	-9,963	-37.22	370,844	99,260,631
\$33,000.00	\$112.00	-5,963	-12.22	72,881	35,556,927
\$35,000.00	\$115.00	-3,963	-9.22	36,547	15,705,075
\$36,000.00	\$118.00	-2,963	-6.22	18,436	8,779,150
\$38,000.00	\$120.00	-963	-4.22	4,066	927,298
\$39,000.00	\$118.00	37	-6.22	-230	1,372
\$41,000.00	\$131.00	2,037	6.78	13,807	4,149,520
\$43,000.00	\$150.00	4,037	25.78	104,066	16,297,668
\$44,000.00	\$148.00	5,037	23.78	119,770	25,371,742
\$46,000.00	\$151.00	7,037	26.78	188,436	49,519,890
\$49,000.00	\$157.00	10,037	32.78	328,992	100,742,112
\$52,000.00	\$168.00	13,037	43.78	570,733	169,964,335
\$54,000.00	\$156.00	15,037	31.78	477,844	226,112,483
\$52,000.00	\$158.00	13,037	33.78	440,362	169,964,335
\$55,000.00	\$161.00	16,037	36.78	589,807	257,186,557
\$59,000.00	\$183.00	20,037	58.78	1,177,733	401,482,853
\$62,000.00	\$167.00	23,037	42.78	985,473	530,705,075
\$65,000.00	\$186.00	26,037	61.78	1,608,510	677,927,298
\$66,000.00	\$191.00	27,037	66.78	1,805,473	731,001,372
Totals				17,435,222	6,724,962,963

Simple Nonlinear Regression

- In situations where the relationship between two variables is nonlinear, a simple way of generating a regression equation is to transform the nonlinear relationship to a linear relationship using a mathematical transformation. A linear model (as described above) can then be generated. Once a prediction has been made, the predicted value is transformed back to the original scale.
- For example, in Table 7.10 two columns show a nonlinear relationship. Plotting these values results in the scatterplot in Figure 7.13.
- There is no linear relationship between these two variables and hence we cannot calculate a linear model directly from the two variables. To generate a model, we transform x or y or both to create a linear relationship. In this example, we transform the y variable using the following formula:

$$y' = \frac{-1}{y}$$

Table 7.10. Table of observations for variables x and y

x	y
3	4
6	5
9	7
8	6
10	8
11	10
12	12
13	14
13.5	16
14	18
14.5	22
15	28
15.2	35
15.3	42

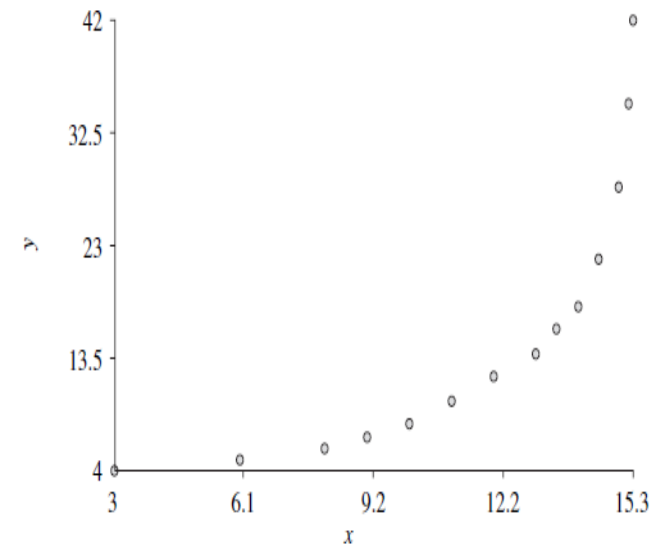


Figure 7.13. Scatterplot showing the nonlinear relationship between x and y

Example

- We now generate a new column, y' (Table 7.11). If we now plot x against y' , we can see that we now have an approximate linear relationship (see Figure 7.14).
- Using the least squares method described previously, an equation for the linear relationship between x and y' can be calculated. The equation is:

$$y' = -0.307 + 0.018 \times x$$

- Using x we can now calculate a predicted value for the transformed value of y (y').

Table 7.11. Transformation of y to create a linear relationship

X	y	$y' = -1/y$
3	4	-0.25
6	5	-0.2
9	7	-0.14286
8	6	-0.16667
10	8	-0.125
11	10	-0.1
12	12	-0.08333
13	14	-0.07143
13.5	16	-0.0625
14	18	-0.05556
14.5	22	-0.04545
15	28	-0.03571
15.2	35	-0.02857
15.3	42	-0.02381

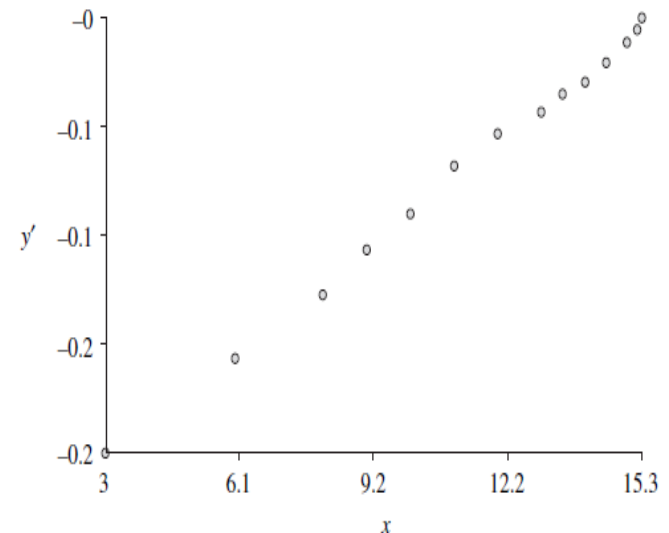


Figure 7.14. Scatterplot illustrating the new linear relationship



Example

Table 7.12. Prediction of y using a nonlinear model

x	y	$y' = -1/y$	Predicted y'	Predicted y
3	4	-0.25	-0.252	3.96
6	5	-0.2	-0.198	5.06
9	7	-0.143	-0.143	6.99
8	6	-0.167	-0.161	6.20
10	8	-0.125	-0.125	8.02
11	10	-0.1	-0.107	9.39
12	12	-0.083	-0.088	11.33
13	14	-0.071	-0.070	14.28
13.5	16	-0.062	-0.061	16.42
14	18	-0.056	-0.052	19.31
14.5	22	-0.045	-0.043	23.44
15	28	-0.036	-0.033	29.81
15.2	35	-0.029	-0.023	33.45
15.3	42	-0.024	-0.028	35.63

To map this new prediction of y' we must now perform an inverse transformation, that is, $-1/y'$. In Table 7.12, we have calculated the predicted value for y' and transformed the number to Predicted y . The **Predicted y values** are close to the **actual y values**.

Locally Weighted Regression

- Construct an explicit approximation to f over a local region surrounding query instance x_q .
- Locally weighted linear regression:

- The target function f is approximated near x_q using the linear function:
$$\hat{f}(x) = w_0 + w_1 a_1(x) + \dots + w_n a_n(x)$$

- minimize the squared error: distance-decreasing weight

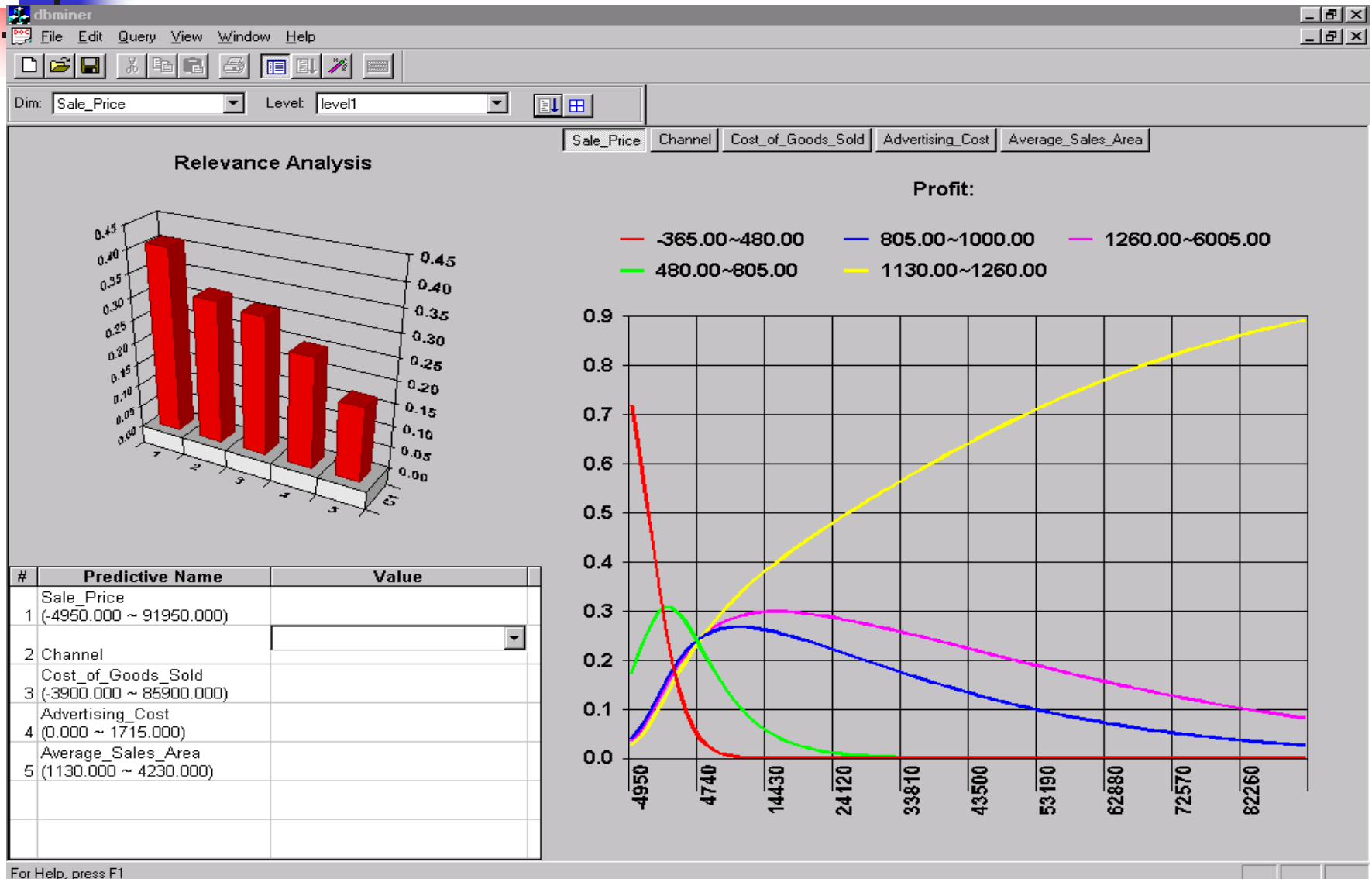
$$K \quad E(x_q) \equiv \frac{1}{2} \sum_{x \in k_nearest_neighbors_of_x_q} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

- the gradient descent training rule:

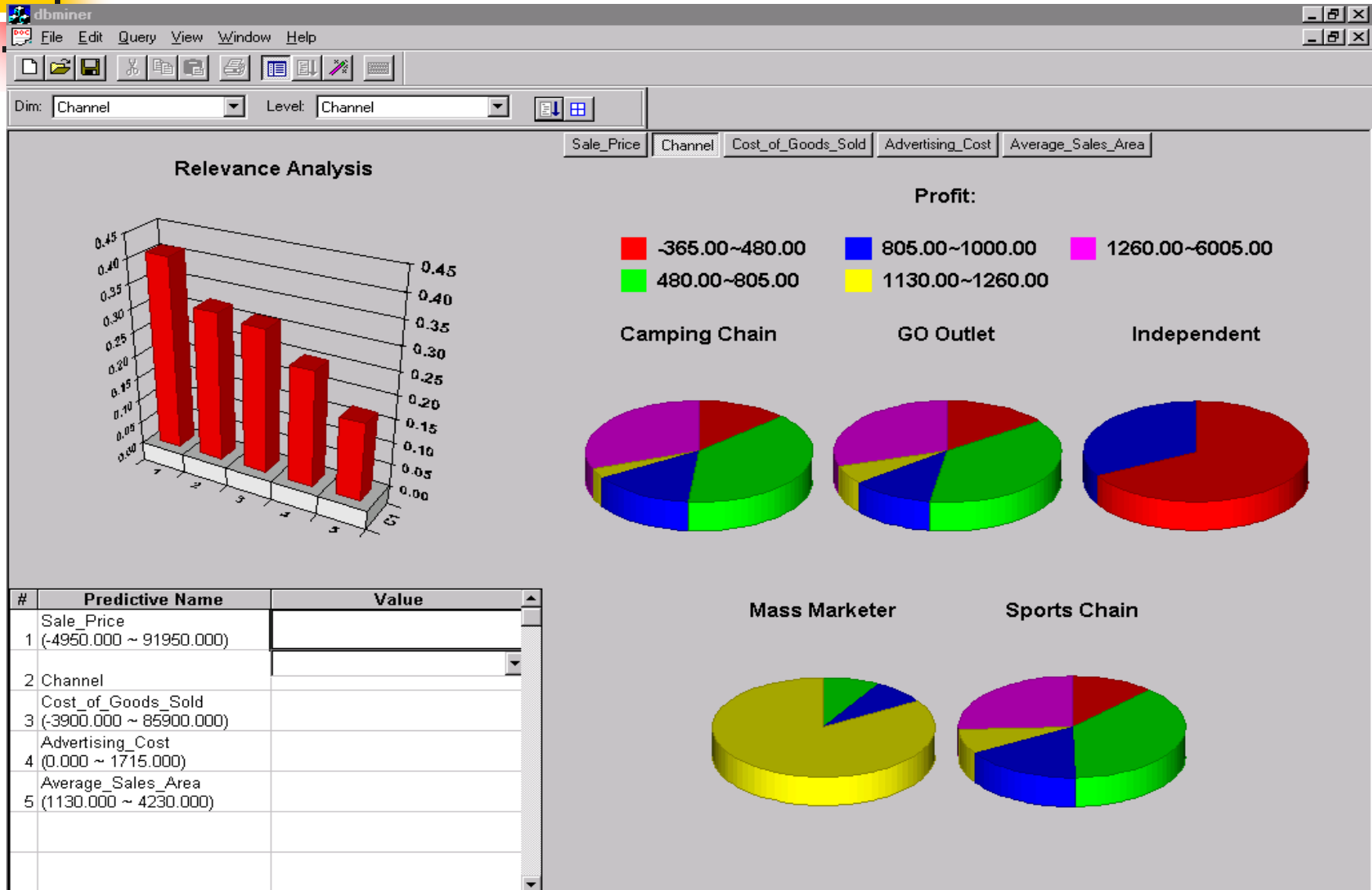
$$\Delta w_j \equiv \eta \sum_{x \in k_nearest_neighbors_of_x_q} K(d(x_q, x)) ((f(x) - \hat{f}(x)) a_j(x))$$

- In most cases, the target function is approximated by a constant, linear, or quadratic function.

Prediction: Numerical Data



Prediction: Categorical Data





Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- **Classification accuracy**
- Summary

Evaluation of a Classifier



- How predictive is the model we learned?
 - Which performance measure to use?
- Natural performance measure for classification problems: *error rate* on a test set
 - Success: instance's class is predicted correctly
 - Error: instance's class is predicted incorrectly
 - **Accuracy**: proportion of correctly classified instances over the whole set of instances
 - **Accuracy=1- error rate**

Issues: Evaluating Classification Methods

- Accuracy classifier accuracy: this refers to the ability of the model to correctly predict the class label of new or previously unseen data:

accuracy = % of testing set examples correctly classified by the classifier

- Predictor accuracy: guessing value of predicted attributes
- 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

Classifier Accuracy Measures

	C_1	C_2
C_1	True positive	False negative
C_2	False positive	True negative

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy of a classifier M , $\text{acc}(M)$: percentage of test set tuples that are correctly classified by the model M
 - Error rate (misclassification rate) of $M = 1 - \text{acc}(M)$
 - Given m classes, $CM_{i,j}$ an entry in a **confusion matrix**, indicates # of tuples in class i that are labeled by the classifier as class j
- Alternative accuracy measures (e.g., for cancer diagnosis)
 - sensitivity = $t\text{-pos}/\text{pos}$ /* true positive recognition rate */
 - specificity = $t\text{-neg}/\text{neg}$ /* true negative recognition rate */
 - precision = $t\text{-pos}/(t\text{-pos} + f\text{-pos})$
 - accuracy = $\text{sensitivity} * \text{pos}/(\text{pos} + \text{neg}) + \text{specificity} * \text{neg}/(\text{pos} + \text{neg})$
 - This model can also be used for cost-benefit analysis

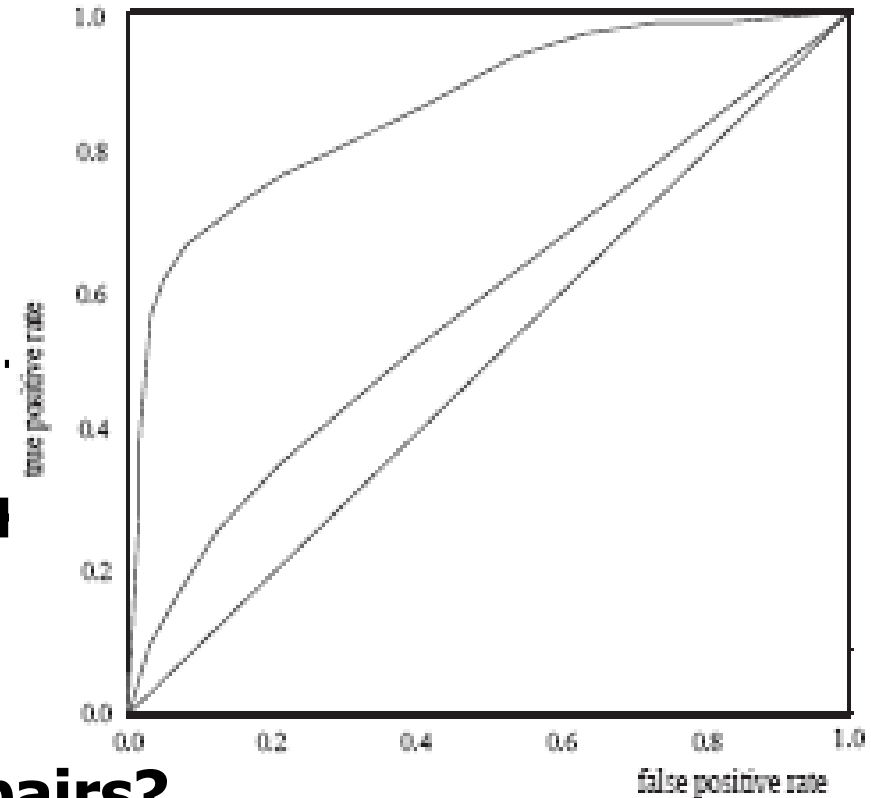
ROC Curve: Receiver Operating Characteristics

	$C_{1(\text{Pre})}$	$C_{2(\text{pre})}$
C_1	True positive	False negative
C_2	False positive	True negative

True positive rate: $TP / (TP + FN)$

False positive rate: $FP / (FP + TN)$

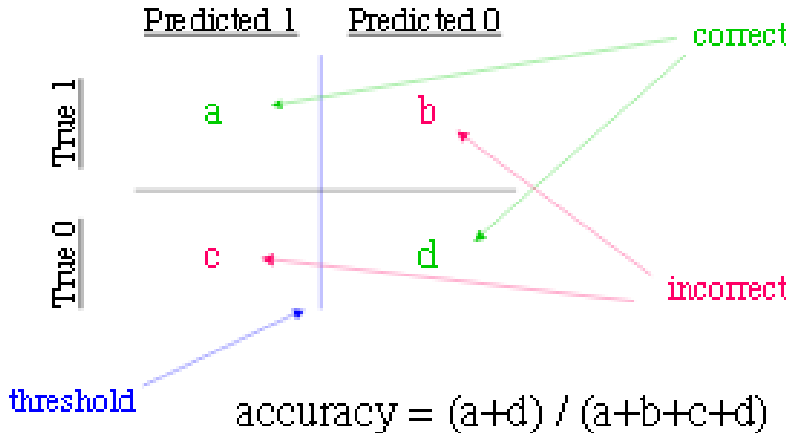
How to get multiple TP, FP pairs?



Confusion Matrix

	$C_{1(Pre)}$	$C_{2(pre)}$
C_1	True positive	False negative
C_2	False positive	True negative

Confusion Matrix



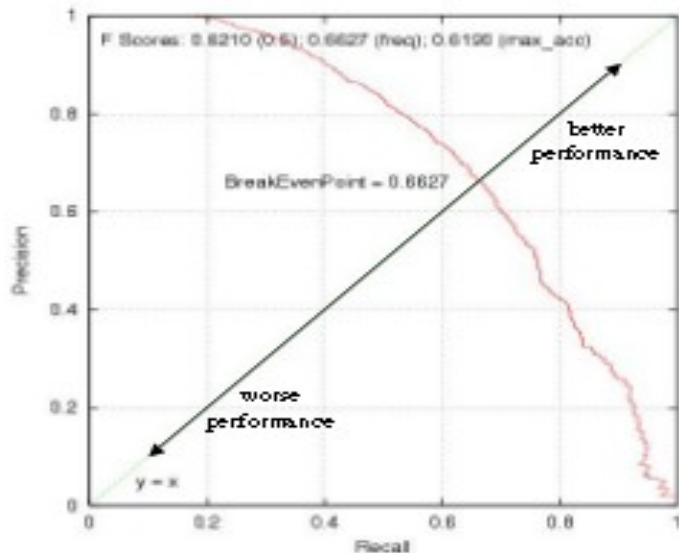
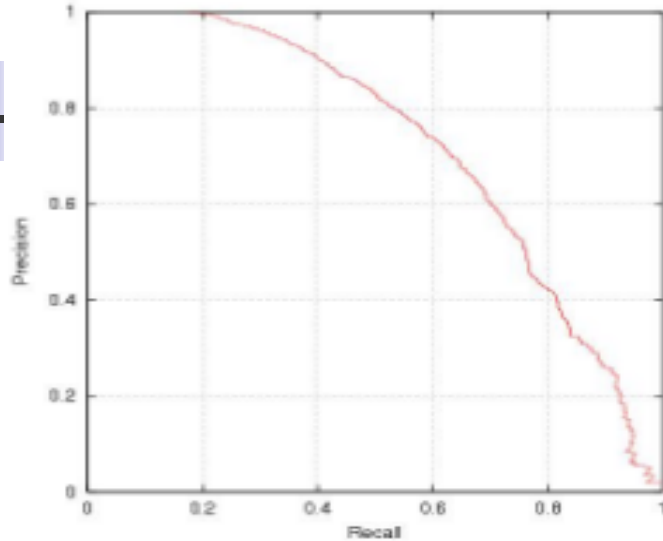
	<u>Predicted 1</u>	<u>Predicted 0</u>
<u>True 1</u>	true positive	false negative
<u>True 0</u>	false positive	true negative

	<u>Predicted 1</u>	<u>Predicted 0</u>
<u>True 1</u>	TP	FN
<u>True 0</u>	FP	TN

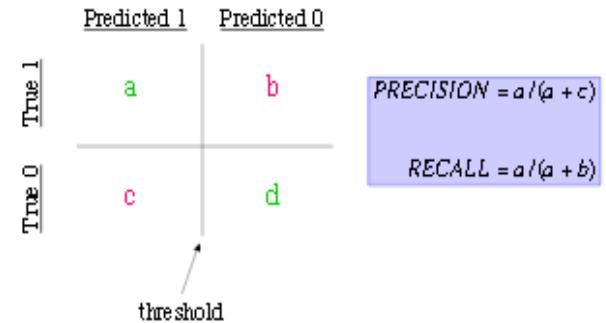
	<u>Predicted 1</u>	<u>Predicted 0</u>
<u>True 1</u>	hits	misses
<u>True 0</u>	false alarms	correct rejections

	<u>Predicted 1</u>	<u>Predicted 0</u>
<u>True 1</u>	$P(pr1 tr1)$	$P(pr0 tr1)$
<u>True 0</u>	$P(pr1 tr0)$	$P(pr0 tr0)$

Precision and Recall



Precision/Recall



32

Summary Stats: F & BreakEvenPt

$$PRECISION = a / (a + c)$$

$$RECALL = a / (a + b)$$

harmonic average of precision and recall

$$F = \frac{2 * (PRECISION * RECALL)}{(PRECISION + RECALL)}$$

$$BreakEvenPoint = PRECISION = RECALL$$

34

Predictor Error Measure

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function:** measures the error between y_i and the predicted value y_i'
 - Absolute error: $|y_i - y_i'|$
 - Squared error: $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set

- Mean absolute error: $\frac{\sum_{i=1}^d |y_i - y_i'|}{d}$ Mean squared error: $\frac{\sum_{i=1}^d (y_i - y_i')^2}{d}$
- Relative absolute error: $\frac{\sum_{i=1}^d |y_i - y_i'|}{\sum_{i=1}^d |y_i - \bar{y}|}$ Relative squared error: $\frac{\sum_{i=1}^d (y_i - y_i')^2}{\sum_{i=1}^d (y_i - \bar{y})^2}$

The mean squared-error exaggerates the presence of outliers. Popularly use (square) root mean-square error, similarly, root relative squared error.

Evaluating the Accuracy of a Classifier or Predictor (I)

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 from each iteration.
- 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, 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

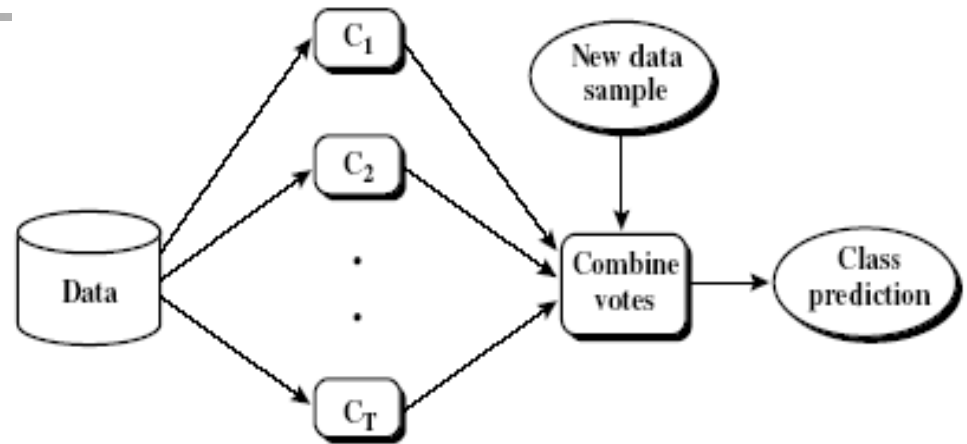
Evaluating the Accuracy of a Classifier or Predictor (II)

■ Bootstrap

- Works well with small data sets
- 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
- Several bootstrap methods, and a common one is **.632 bootstrap**
 - Suppose we are given a data set of d tuples. The data set 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 will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
 - Repeat the sampling procedure k times, overall accuracy of the model:

$$acc(M) = \sum_{i=1}^k (0.632 \times acc(M_i)_{test_set} + 0.368 \times acc(M_i)_{train_set})$$

Ensemble Methods: Increasing the Accuracy



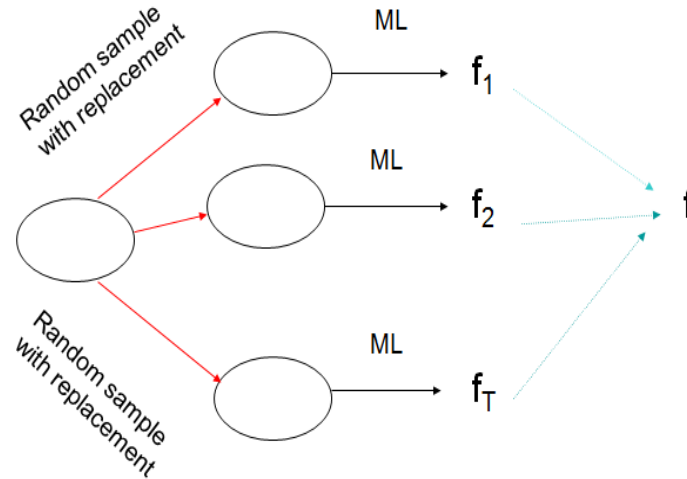
■ Ensemble methods

- Use a combination of models to increase accuracy
- Combine a series of k learned models, M_1, M_2, \dots, 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: Example



- Breiman, 1996
- Derived from bootstrap (Efron, 1993)
- Create classifiers using training sets that are bootstrapped (drawn with replacement)
- Average results for each case



Bagging

- Sampling (with replacement) according to a uniform probability distribution
 - Each bootstrap sample D has the same size as the original data.
 - Some instances could appear several times in the same training set, while others may be omitted.
- Build classifier on each bootstrap sample D
- D will contain approximately 63% of the original data.
- Each data object has probability $1 - (1 - 1/n)^n$ of being selected in D



Bagging: Bootstrap 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 \mathbf{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 \mathbf{X}
- 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 significant better than a single classifier derived from D
 - For noise data: not considerably worse, more robust
 - Proved improved accuracy in prediction

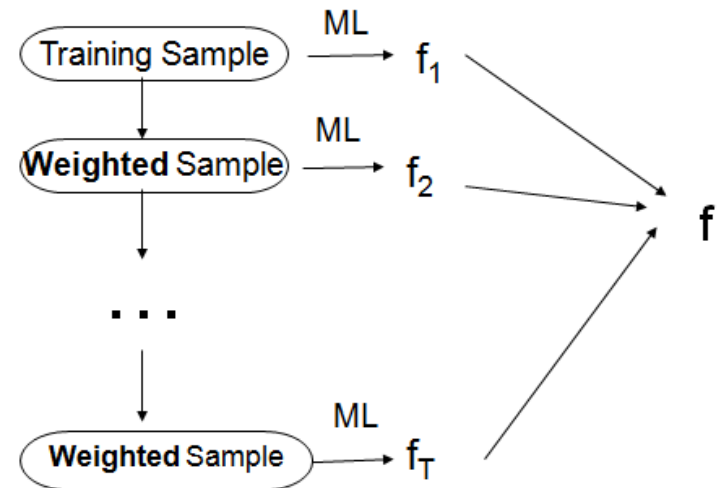


Bagging: Advantages and Limitations

- Bagging improves generalization performance by reducing variance of the base classifiers. The performance of bagging depends on the stability of the base classifier.
 - If a base classifier is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data.
 - If a base classifier is stable, bagging may not be able to improve, rather it could degrade the performance.
- Bagging is less susceptible to model overfitting when applied to noisy data.

Boosting

- Sequential production of classifiers
- Each classifier is dependent on the previous one, and focuses on the previous one's errors
- Examples that are incorrectly predicted in previous classifiers are chosen more often or weighted more heavily





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
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

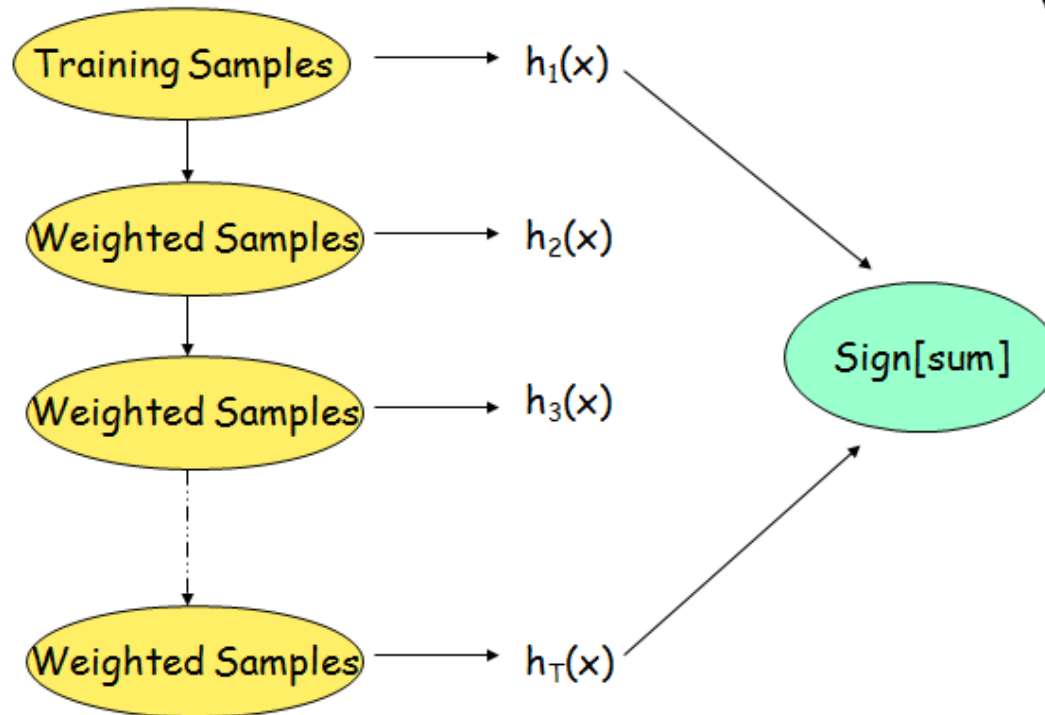
Adaboost (Freund and Schapire, 1997)

- Given a set of d class-labeled tuples, $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- Initially, all the weights of tuples are set 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
 - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: $err(\mathbf{X}_j)$ is the misclassification error of tuple \mathbf{X}_j . Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X}_j)$$

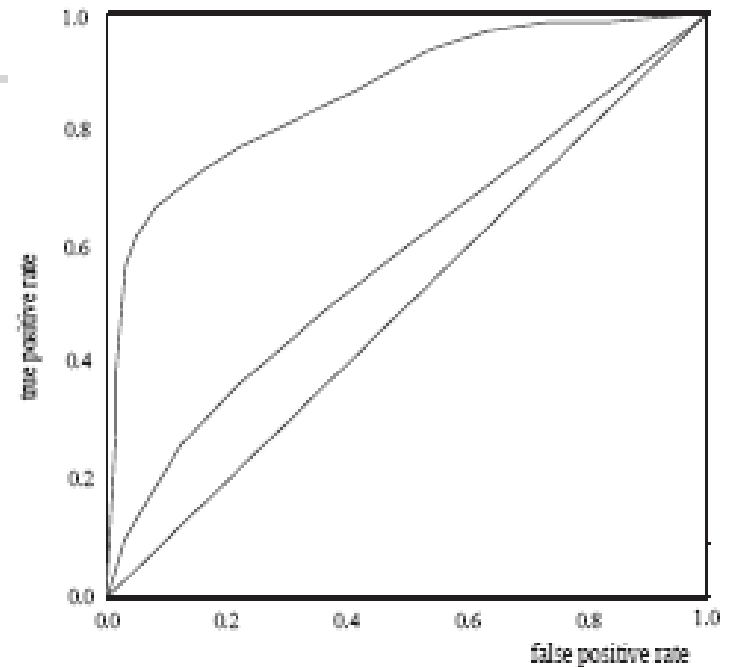
- The weight of classifier M_i 's vote is $\log \frac{1 - error(M_i)}{error(M_i)}$

Schematic of Adaboost



Model Selection: ROC Curves

- 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 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: true positive rate
- Horizontal axis: false positive rate
- Diagonal line?
- A model with perfect accuracy: area of 1.0



Classification Accuracy: Estimating Error Rates

- Partition: Training-and-testing
 - use two independent data sets, e.g., training set (2/3), test set(1/3)
 - used for data set with large number of samples
- Cross-validation
 - divide the data set into k subsamples
 - use $k-1$ subsamples as training data and one subsample as test data --- k -fold cross-validation
 - for data set with moderate size
- Bootstrapping (leave-one-out)
 - for small size data



Boosting and Bagging

- Boosting increases classification accuracy
 - Applicable to decision trees or Bayesian classifier
- Learn a series of classifiers, where each classifier in the series pays more attention to the examples misclassified by its predecessor
- Boosting requires only linear time and constant space



Boosting Technique (II) — Algorithm

- Assign every example an equal weight $1/N$
- *For $t = 1, 2, \dots, T$ Do*
 - Obtain a hypothesis (classifier) $h^{(t)}$ under $w^{(t)}$
 - Calculate the error of $h^{(t)}$ and re-weight the examples based on the error
 - Normalize $w^{(t+1)}$ to sum to 1
- Output a weighted sum of all the hypothesis, with each hypothesis weighted according to its accuracy on the training set



Chapter 7. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification by backpropagation
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- **Summary**



Summary

- Classification is an **extensively studied** problem (mainly in statistics, machine learning & neural networks)
- Classification is probably one of the most **widely used** data mining techniques with a lot of extensions
- **Scalability** is still an important issue for database applications: thus combining classification **with database techniques** should be a promising topic
- Research directions: classification of **non-relational data**, e.g., text, spatial, multimedia, etc..



References (I)

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13, 1997.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proc. 1st Int. Conf. Knowledge Discovery and Data Mining (KDD'95)*, pages 39-44, Montreal, Canada, August 1995.
- U. M. Fayyad. Branching on attribute values in decision tree generation. In *Proc. 1994 AAAI Conf.*, pages 601-606, AAAI Press, 1994.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In *Proc. 1998 Int. Conf. Very Large Data Bases*, pages 416-427, New York, NY, August 1998.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In *Proc. 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97)*, pages 111-120, Birmingham, England, April 1997.



References (II)

- J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, *Advanced Methods of Marketing Research*, pages 118-159. Blackwell Business, Cambridge Massachusetts, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In *Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96)*, Avignon, France, March 1996.
- S. K. Murthy, *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey*, *Data Mining and Knowledge Discovery* 2(4): 345-389, 1998
- J. R. Quinlan. Bagging, boosting, and c4.5. In *Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96)*, 725-730, Portland, OR, Aug. 1996.
- R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and pruning. In *Proc. 1998 Int. Conf. Very Large Data Bases*, 404-415, New York, NY, August 1998.
- J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. In *Proc. 1996 Int. Conf. Very Large Data Bases*, 544-555, Bombay, India, Sept. 1996.
- S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufman, 1991.