SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Subject Name : Data mining and Warehousing   Unit :IV   Subject Code : SIT1301

CHAPTER NAME : CLASSIFICATION AND PREDICTION

4.1 Classification and Prediction

- predicts categorical classlabels
- classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying newdata
- models continuous-valued functions, i.e., predicts unknown or missingvalues

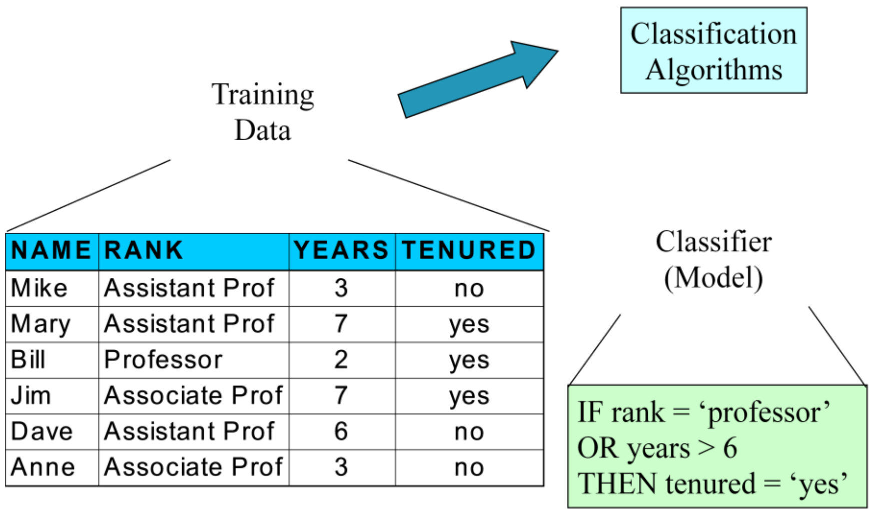
- **Typical applications**

  - Creditapproval
  - Targetmarketing
  - Medicaldiagnosis
  - Frauddetection
  -

Classification—A Two-Step Process

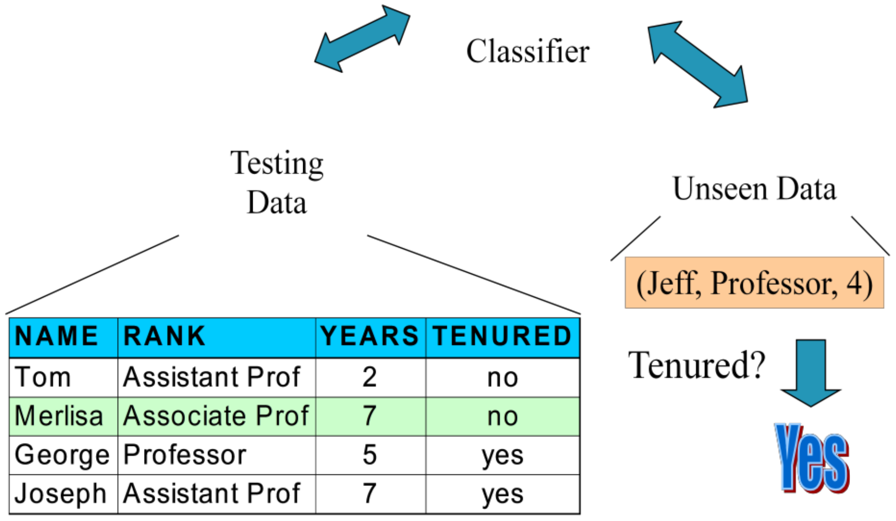
- Model construction: describing a set of predeterminedclasses
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class labelattribute
  - The set of tuples used for model construction: trainingset
  - The model is represented as classification rules, decision trees, or mathematicalformulae
- Model usage: for classifying future or unknownobjects
  - Estimate accuracy of themodel
    - The known label of test sample is compared with the classified result from themodel
    - Accuracy rate is the percentage of test set samples that are correctly classified by themodel
    - Test set is independent of training set, otherwise over-fitting willoccur

Process (1): Model Construction


Training Data
Classification Algorithms
Classifier (Model)
IF rank = 'professor' OR years > 6 THEN tenured = 'yes'

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

Process (2): Using the Model in Prediction



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 theobservations
  - New data is classified based on the trainingset
- Unsupervised learning(clustering)
  - The class labels of training data isunknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in thedata

4.2 Issues Regarding Classification and Prediction

This describes issues regarding preprocessing the data of classification and prediction. Criteria for the comparison and evaluation of classification methods are also described.

Preparing the Data for Classification and Prediction

The following preprocessing steps may be applied to the data in order to help improve the accuracy, efficiency, and scalability of the classification or prediction process.

Data Cleaning: This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques) and the treatment of missing values (e.g., by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics.) Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.

Relevance Analysis: Many of the attributes in the data may be irrelevant to the classification or prediction task. For example, data recording the day of the week on which a bank loan application was filed is unlikely to be relevant to the success of the application. Furthermore, other attributes may be redundant. Hence, relevance analysis may be performed on the data with the aim of removing any irrelevant or redundant attributes from the learning process. In machine learning, this step is known as feature selection. Including such attributes may otherwise slow down, and possibly mislead, the learning step.

Ideally, the time spent on relevance analysis, when added to the time spent on learning from the resulting “reduced” feature subset should be less than the time that would have been spent on learning from the original set of features. Hence, such analysis can help improve classification efficiency and scalability.

Data Transformation: The data can be generalized to higher – level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous – valued attributes. For example, numeric values for the attribute income may be generalized to discrete ranges such as low, medium, and high. Similarly, nominal – valued attributes like street, can be generalized to higher – level concepts, like city. Since generalization compresses the original training data, fewer input / output operations may be involved during learning.

The data may also be normalized, particularly when neural networks or methods involving distance measurements are used in the learning step. Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as – 1.0 to 1.0, or 0.0 to 1.0. In methods that use distance measurements, for example, this would prevent attributes with initially large ranges (like, say, income) from outweighing attributes with initially smaller ranges (such as binary attributes).

Comparing Classification Methods

Classification and prediction methods can be compared and evaluated according to the following criteria:

Predictive Accuracy: This refers to the ability of the model to correctly predict the class label of new or previously unseen data.

Speed: This refers to the computation costs involved in generating and using the model.

Robustness: This is the ability of the model to make correct predictions given noisy data or data with missing values.

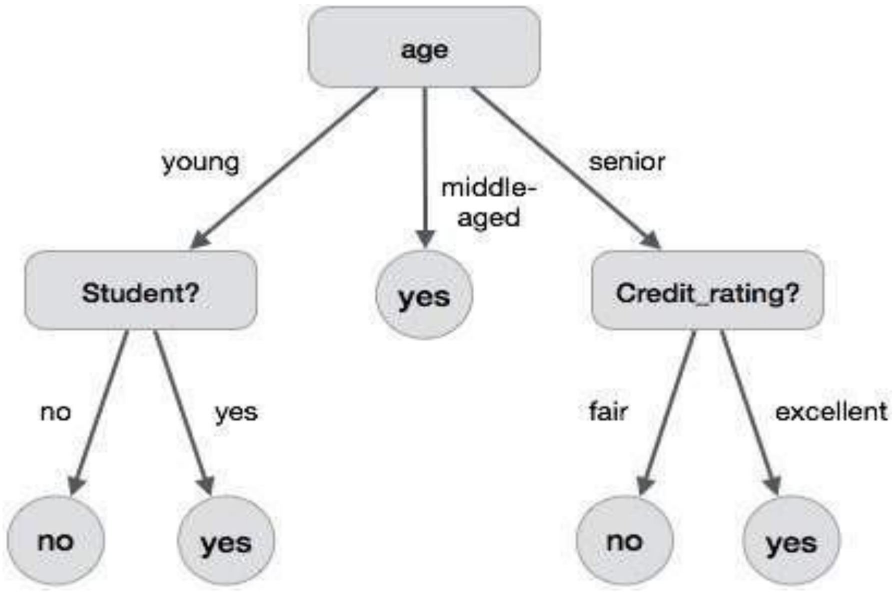
Scalability: This refers to the ability to construct the model efficiently given large amount of data.

Interpretability: This refers to the level of understanding and insight that is provided by the model.

4.3 Classification by Decision Tree Induction

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept buys_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents aclass.



The benefits of having a decision tree are as follows :

- It does not require any domainknowledge.
- It is easy tocomprehend.
- The learning and classification steps of a decision tree are simple andfast.

The expected information needed to classify a tuple in D is given by

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

Then, for each attribute A,

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

where Dj / D is the weight of the jth partition.

Info A (D) is the expected information required to classify a tuple from D based on the partitioning by A. The smaller the expected information, the greater the purity of thepartitions.

Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,

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

//Generating a decision tree form training tuples of data partition D

<u>Algorithm</u>: Generate_decision_tree 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 that the data tuples into individual classes. This criterion includesa
splitting_attribute and either a splitting point or splitting subset.

Output:
A Decision Tree

Method

```
create a node N;

if tuples in D are all of the same class, C then return N
    as leaf node labeled with class C;


if attribute_list is empty then return N as
    leaf node withlabeled
    with majority class in D;|| majority voting
```

```
apply attribute_selection_method(D, attribute_list) to find
the best splitting_criterion;
label node N with splitting_criterion;

if splitting_attribute is discrete-valued and
    multiway splitsallowedthen           // no restricted to binarytrees

attribute_list = splitting attribute; // remove splittingattribute for each
outcome j of splitting criterion

    // partition the tuples and grow subtrees for each partition
    let Dj be the set of data tuples in D satisfying outcome j; // a partition

    if Dj is empty then
        attach a leaf labeled with the majority class in
        D to node N;
    else
        attach the node returned by Generate decision
        tree(Dj, attribute list) to nodeN;
    end for
return N;
```

Tree Pruning

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

Here is the Tree Pruning Approaches listed below −

- **Pre-pruning** − The tree is pruned by halting its constructionearly.
- **Post-pruning -** This approach removes a sub-tree from a fully growntree.

Cost Complexity

The cost complexity is measured by the following two parameters −

- Number of leaves in the tree,and
- Error rate of thetree.

Example:

Class-Labeled Training Tuples from the *AllElectronics* Customer Database

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

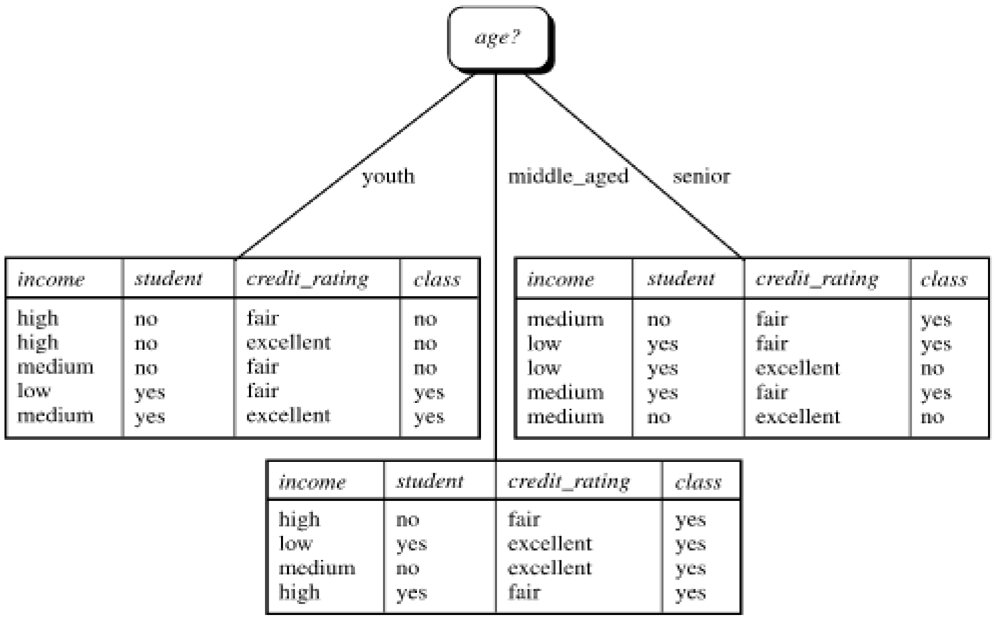
$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$\begin{aligned} Info_{age}(D) = {} & \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ & + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) \\ & + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ & = 0.694 \text{ bits.} \end{aligned}$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Similarly,
Gain(income)=0.029
Gain(Credit_rating)=0.1
51 Gain(Student)=0.048

Therefore out of all Gain values obtained, the attribute Age has gained a higher value, and hence it proves itself to be the best splliting attribute. Hence, the decision tree would look like the one given below:



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

The branch middle-aged is classified as pure class (YES). Repeat the same process for the youth and senior branch until all the branches of decision tree turned to pure class.

4.4 Bayesian Classification

Bayesian framework assumes that we always have a prior distribution for everything.

- The prior may be very vague.
- When we see some data, we combine our prior distribution with a likelihood term to get a posterior distribution.
- The likelihood term takes into account how probable the observed data is given the parameters of the model.
    - It favors parameter settings that make the data likely.
    - It fights the prior
    - With enough data the likelihood terms always win.

4.5 Baye's THeorem

$P(H|X)$ is the **posterior probability**, or *a posteriori probability*, of H conditioned on X. For example, suppose our world of data tuples is confined to customers described by the attributes *age* and *income*, respectively, and that X is a 35-year-old customer with an income of \$40,000. Suppose that H is the hypothesis that our customer will buy a computer. Then $P(H|X)$ reflects the probability that customer X will buy a computer given that we know the customer's age and income.

In contrast, $P(H)$ is the **prior probability**, or *a priori probability*, of H. For our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter.

Bayes' theorem is useful in that it provides a way of calculating the posterior probability $P(H|X)$, from $P(H)$, $P(X|H)$, and $P(X)$.
Bayes' theorem is

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

Naïve Bayesian Classification

The **naïve Bayesian** classifier, or **simple Bayesian** classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, $X = (x_1, x_2, \ldots, x_n)$, depicting n measurements made on the tuple from n attributes, respectively, $A_1, A_2, \ldots, A_n$.
2. Suppose that there are m classes, $C_1, C_2, \ldots, C_m$. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class C_i if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \le j \le m, j \ne i.$$

Thus, we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called the *maximum posteriori hypothesis*. By Bayes' theorem (Eq. 8.10),

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

3. As $P(X)$ is constant for all classes, only $P(X|C_i)P(C_i)$ needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \cdots = P(C_m)$, and we would therefore maximize $P(X|C_i)$. Otherwise, we maximize $P(X|C_i)P(C_i)$. Note that the class prior probabilities may be estimated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D.

4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(X|C_i)$. To reduce computation in evaluating $P(X|C_i)$, the naïve assumption of **class-conditional independence** is made. This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

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

We can easily estimate the probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ from the training tuples. Recall that here x_k refers to the value of attribute A_k for tuple X. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute $P(X|C_i)$, we consider the following:

(a) If A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k, divided by $|C_{i,D}|$, the number of tuples of class C_i in D.

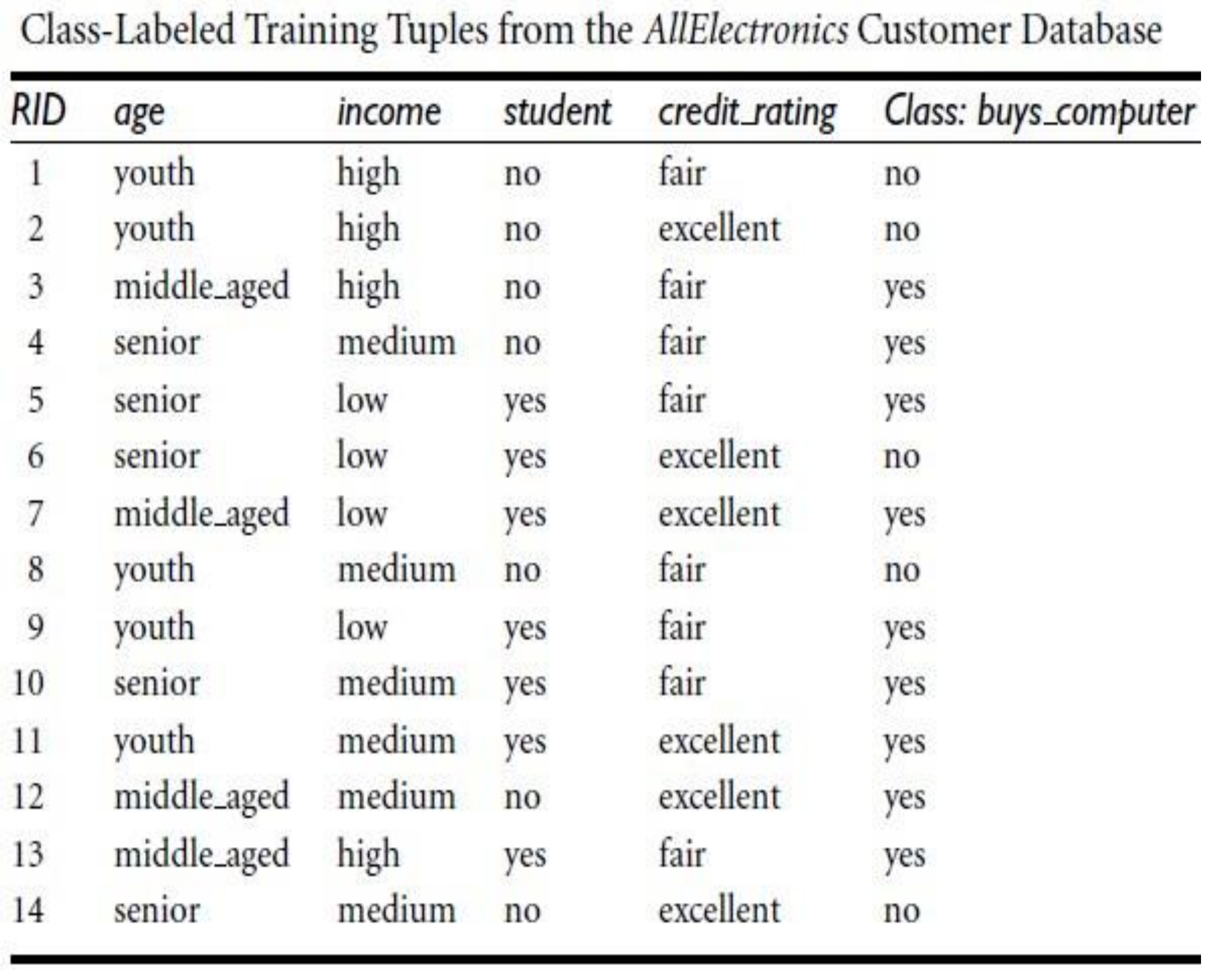
(b) If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

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

so that

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

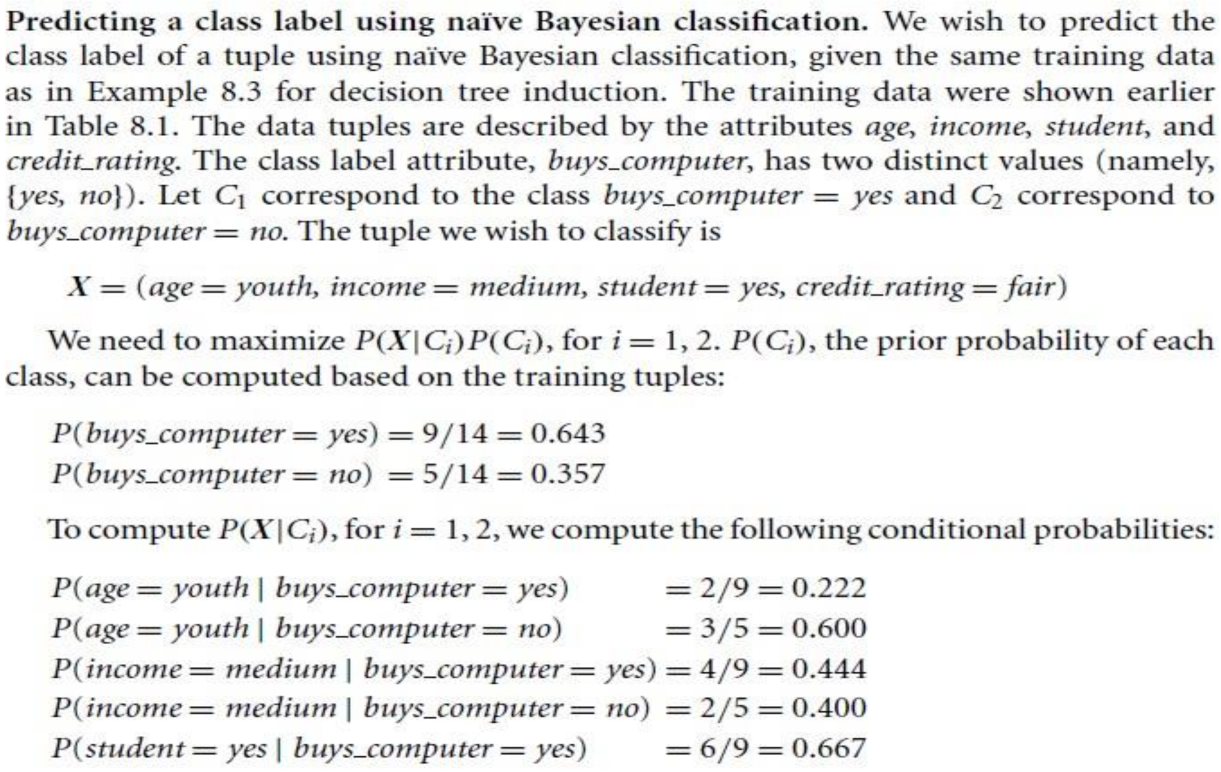
Given database:



Class-Labeled Training Tuples from the *AllElectronics* Customer Database

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

Example:



Predicting a class label using naïve Bayesian classification. We wish to predict the class label of a tuple using naïve Bayesian classification, given the same training data as in Example 8.3 for decision tree induction. The training data were shown earlier in Table 8.1. The data tuples are described by the attributes *age*, *income*, *student*, and *credit_rating*. The class label attribute, *buys_computer*, has two distinct values (namely, {*yes*, *no*}). Let C_1 correspond to the class *buys_computer* = *yes* and C_2 correspond to *buys_computer* = *no*. The tuple we wish to classify is

$$X = (age = youth,\ income = medium,\ student = yes,\ credit\_rating = fair)$$

We need to maximize $P(X|C_i)P(C_i)$, for $i = 1, 2$. $P(C_i)$, the prior probability of each class, can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

To compute $P(X|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$
$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$
$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$
$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$
$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$

$$\begin{aligned}
P(student = yes \mid buys\_computer = no) &= 1/5 = 0.200\\
P(credit\_rating = fair \mid buys\_computer = yes) &= 6/9 = 0.667\\
P(credit\_rating = fair \mid buys\_computer = no) &= 2/5 = 0.400
\end{aligned}$$

Using these probabilities, we obtain

$$\begin{aligned}
P(X \mid buys\_computer = yes) = {} & P(age = youth \mid buys\_computer = yes)\\
& \times P(income = medium \mid buys\_computer = yes)\\
& \times P(student = yes \mid buys\_computer = yes)\\
& \times P(credit\_rating = fair \mid buys\_computer = yes)\\
= {} & 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.
\end{aligned}$$

Similarly,

$$P(X \mid buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, C_i, that maximizes $P(X|C_i)P(C_i)$, we compute

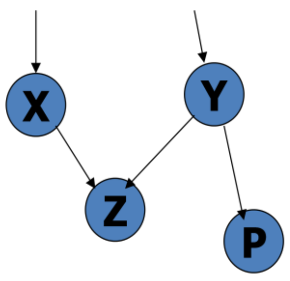
$$\begin{aligned}
P(X \mid buys\_computer = yes)P(buys\_computer = yes) &= 0.044 \times 0.643 = 0.028\\
P(X \mid buys\_computer = no)P(buys\_computer = no) &= 0.019 \times 0.357 = 0.007
\end{aligned}$$

Therefore, the naïve Bayesian classifier predicts $buys\_computer = yes$ for tuple X.

4.7 Bayesian networks

A Bayesian network, Bayes network, belief network, Bayes(ian) model or probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

- Bayesian belief network allows a *subset* of the variables conditionallyindependent
- A graphical model of causalrelationships
  - Represents dependency among thevariables
  - Gives a specification of joint probabilitydistribution



❑ Nodes: random variables

❑ Links: dependency

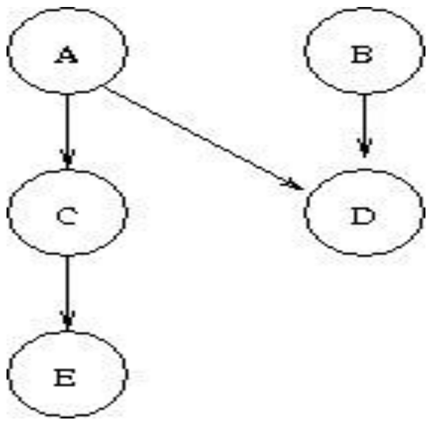
❑ X and Y are the parents of Z, and Y is the parent of P

❑ No dependency between Z and P

❑ Has no loops or cycles

Bayesian Net Example:

Consider the following Bayesian network:



Thus, the independence expressed in this Bayesian net are that A and B are (absolutely) independent.
C is independent of B given A.
D is independent of C given A and B.
E is independent of A, B, and D given C.

Suppose that the net further records the following probabilities:
Prob(A=T) = 0.3
Prob(B=T) = 0.6
Prob(C=T|A=T) = 0.8
Prob(C=T|A=F) = 0.4
Prob(D=T|A=T,B=T) = 0.7
Prob(D=T|A=T,B=F) = 0.8
Prob(D=T|A=F,B=T) = 0.1
Prob(D=T|A=F,B=F) = 0.2
Prob(E=T|C=T) = 0.7
Prob(E=T|C=F) = 0.2

Some sample computations:

<u>Prob(D=T):</u>

P(D=T) =

P(D=T,A=T,B=T) + P(D=T,A=T,B=F) + P(D=T,A=F,B=T) + P(D=T,A=F,B=F) = P(D=T|A=T,B=T) P(A=T,B=T) + P(D=T|A=T,B=F) P(A=T,B=F) + P(D=T|A=F,B=T) P(A=F,B=T) + P(D=T|A=F,B=F) P(A=F,B=F) =
(since A and B are independent absolutely)

P(D=T|A=T,B=T) P(A=T) P(B=T) + P(D=T|A=T,B=F) P(A=T) P(B=F) + P(D=T|A=F,B=T) P(A=F) P(B=T) + P(D=T|A=F,B=F) P(A=F) P(B=F) =

0.7*0.3*0.6 + 0.8*0.3*0.4 + 0.1*0.7*0.6 + 0.2*0.7*0.4 = 0.32

<u>Prob(A=T|C=T):</u>

P(A=T|C=T) = P(C=T|A=T)P(A=T) / P(C=T).

Now, P(C=T) = P(C=T,A=T) + P(C=T,A=F) =
P(C=T|A=T)P(A=T) + P(C=T|A=F)P(A=F) =
0.8*0.3+ 0.4*0.7 = 0.52

So P(C=T|A=T)P(A=T) / P(C=T) = 0.8*0.3/0.52= 0.46.

4.8 Rule Based Classification

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. ruleconsequent
  Assessment of a rule: *coverage* and*accuracy*
- n_{covers} = # of tuples covered byR
- $n_{correct}$ = # of tuples correctly classified byR

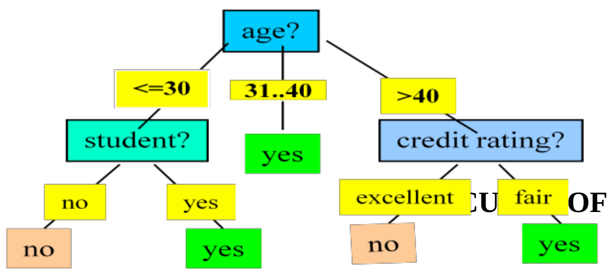
coverage(R) = n_{covers} /|D| /* D: training data set */

accuracy(R) = $n_{correct}$ /n_{covers}

- If more than one rule is triggered, need conflict resolution
  - Sizeordering:assignthehighestprioritytothetriggeringrulesthathasthe –toughest‖ requirement (i.e., with the *most attribute test*)
  - Class-based ordering: decreasing order of *prevalence or misclassification cost per class*
  - Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality or by experts

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 and exhaustive



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

IF *age* = young AND *student=no* THEN *buys_computer = no*
IF *age* = young AND *student=yes* THEN *buys_computer = yes*
IF *age*=mid-age THEN *buys_computer =yes*
IF *age* = old AND *credit_rating = excellent* THEN *buys_computer = yes*
IF *age* = young AND *credit_rating=fair* THEN *buys_computer = no*

Rule Extraction from the Training Data

- Sequential covering algorithm: Extracts rules directly from trainingdata
- 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-specifiedthreshold
- Comp. w. decision-tree induction: learning a set of rules*simultaneously*

4.9 Classification by Back propagation

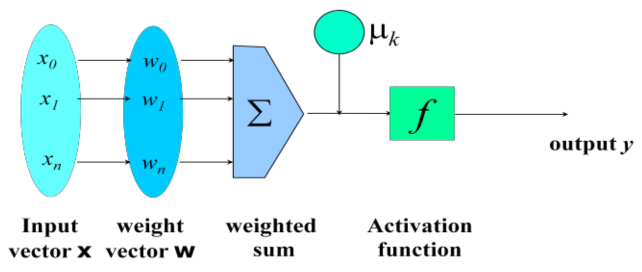
- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection hasa **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically,e.g., the network topology or``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behindthe learned weights and of ``hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel

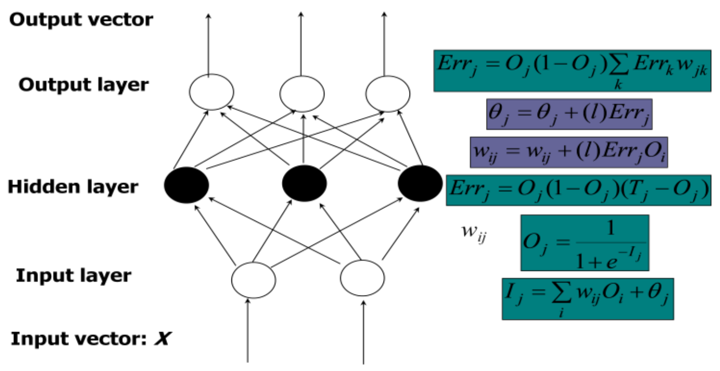
- Techniques have recently been developed for the extraction of rules from trained neural networks

A Neuron (= a perceptron)



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

A Multi-Layer Feed-Forward Neural Network


Output vector
Output layer
Hidden layer
Input layer
Input vector: *X*
$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$
$\theta_j = \theta_j + (l)Err_j$
$w_{ij} = w_{ij} + (l)Err_j O_i$
$Err_j = O_j(1-O_j)(T_j - O_j)$
w_{ij}
$O_j = \frac{1}{1+e^{-I_j}}$
$I_j = \sum_i w_{ij} O_i + \theta_j$

- The inputs to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the inputlayer
- They are then weighted and fed simultaneously to a hiddenlayer
- The number of hidden layers is arbitrary, although usually onlyone
- The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network's prediction
- The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previouslayer
- From a statistical point of view, networks perform nonlinear regression: Given enough hidden units and enough training samples, they can closely approximate any function

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known targetvalue
- For each training tuple, the weights are modified to minimize the mean squared error

between the network's prediction and the actual targetvalue

- Modificationsaremadeinthe–backwardsǁdirection:fromtheoutputlayer,througheach hidden layer down to the first hidden layer, hence–backpropagationǁ
- Steps
  - Initialize weights (to small random #s) and biases in thenetwork
  - Propagate the inputs forward (by applying activationfunction)
  - Backpropagate the error (by updating weights andbiases)
  - Terminating condition (when error is very small,etc.)
- Efficiency of backpropagation: Each epoch (one interation through the training set) takes O(|D| * *w*), with |D| tuples and *w* weights, but # of epochs can be exponential to n, the number of inputs, in the worstcase
- Rule extraction from networks: networkpruning
  - Simplify the network structure by removing weighted links that have the least effect on the trainednetwork
  - Then perform link, unit, or activation valueclustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unitlayers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented inrules

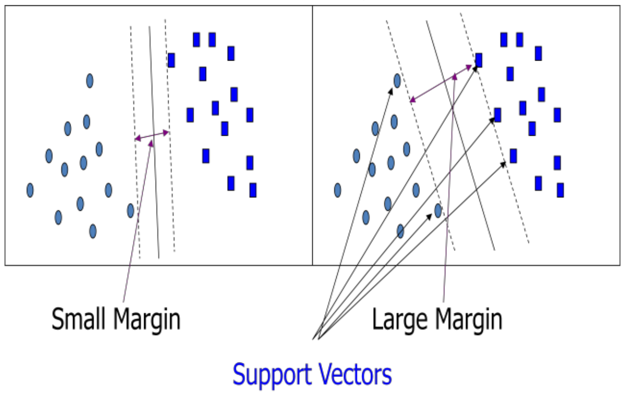
Algorithm: Backpropagation. Neural network learning for classification or numeric prediction, using the backpropagation algorithm.
Input:
D, a data set consisting of the training tuples and their associated target values;
l, the learning rate;
network, a multilayer feed-forward network.
Output: A trained neural network.

```
Method:
Initialize all weights and biases in network;
while terminating condition is not satisfied {
for each training tuple X in D {
// Propagate the inputs forward:
for each input layer unit j {
Oj = Ij ; // output of an input unit is its actual input value
for each hidden or output layer unit j {
(8)   $I_j = \sum_i w_{ij} O_i + \theta_j$ ; //compute the net input of unit j with respect to the previous layer, i
Oj = $\frac{1}{1+e^{-I_j}}$ ; } // compute the output of each unit j
// Backpropagate the errors:
for each unit j in the output layer
Errj = Oj (1 − Oj )(Tj − Oj ); // compute the error
for each unit j in the hidden layers, from the last to the first hidden layer
Errj = Oj (1 − Oj ) $\sum_k$ Errk wjk ; // compute the error with respect to the next higher layer, k
for each weight wij in network {
Owij = (l)Errj Oi ; // weight increment
wij = wij + Owij ; } // weight update
for each bias θj in network {
Oθj = (l)Errj ; // bias increment
θj = θj + Oθj ; } // bias update
} }
```

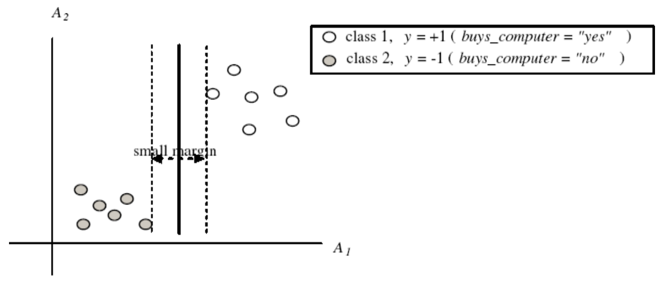
4.10 SVM—Support Vector Machines

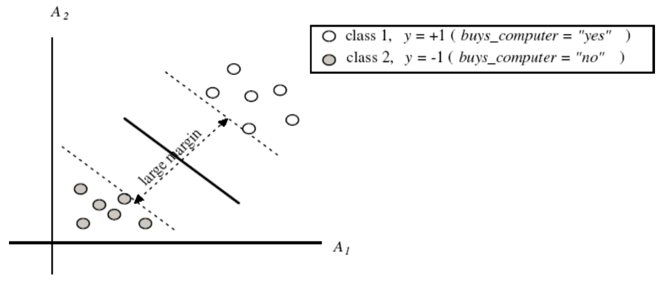
- A new classification method for both linear and nonlineardata
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane(i.e., –decision boundary‖)
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by ahyperplane
- SVM finds this hyperplane using support vectors (—essential‖ training tuples) and margins (defined by the supportvectors)
- **Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (marginmaximization)**
- **Used both for classification andprediction**
- **Applications:**
    - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series predictiontests

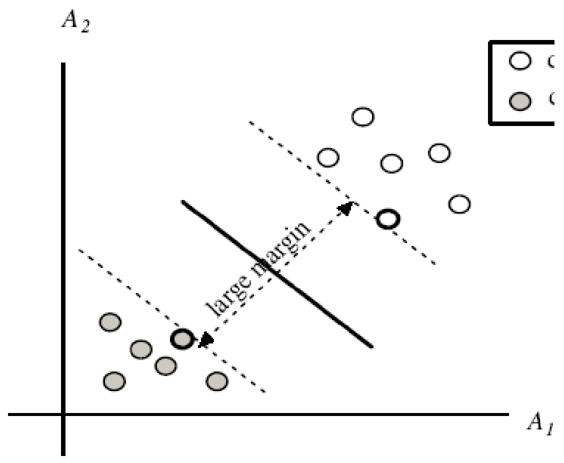
SVM—General Philosophy



SVM—Margins and Support Vectors


A_2
A_1
class 1, y = +1 (buys_computer = "yes")
class 2, y = -1 (buys_computer = "no")
small margin


A_2
A_1
class 1, y = +1 (buys_computer = "yes")
class 2, y = -1 (buys_computer = "no")
large margin



SVM—Linearly Separable

■ A separating hyperplane can be writtenas

W ● X + b =0

where W={w_1, w_2, …, w_n} is a weight vector and b a scalar (bias)

- ■ For 2-D it can be written as

$w_0 + w_1 x_1 + w_2 x_2 =0$

- ■ The hyperplane defining the sides of the margin:

H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for y_i= +1,and

H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- ■ Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are supportvectors
- ■ This becomes a constrained (convex) quadratic optimization problem: Quadratic objective function and linear constraints *Quadratic Programming (QP)* Lagrangianmultipliers

Why Is SVM Effective on High Dimensional Data?

- ■ The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data

- The support vectors are the essential or critical training examples —they lie closest to the decision boundary(MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would befound
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data ishigh

Associative Classification

- Associativeclassification
  - Association rules are generated and analyzed for use inclassification
  - Search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and classlabels
  - Classification: Based on evaluating a set of rules in the form of

$P_1 \wedge p_2 \ldots \wedge p_l$ —A_{class} = C‖(conf, sup)

- Whyeffective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at atime

In many studies, associative classification has been found to be more accurate than some traditional classification methods, such asC4.

4.11 Prediction

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or predictor variables and a *dependent* or response variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

4.12 Linear Regression

- Linear regression: involves a response variable y and a single predictor variable x

$y = w_0 + w_1x$
where w_0 (y-intercept) and w_1 (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line
    - Multiple linear regression: involves more than one predictor variable
    - Training data is of the form ($\mathbf{X_1}$, y_1), ($\mathbf{X_2}$, y_2),…, ($\mathbf{X_{|D|}}$,$y_{|D|}$)
    - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2x_2$
    - Solvable by extension of least square method or using SAS,S-Plus
    - Many nonlinear functions can be transformed into the above

4.13 Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

$y = w_0 + w_1 x + w_2 x^2 + w_3x^3$
convertible to linear with new variables: $x_2= x^2$, $x_3=x^3$
$y = w_0 + w_1 x + w_2 x_2 + w_3x_3$

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
    - possible to obtain least square estimates through extensive calculation on more complex formulae