Chaitanya Bharathi Institute of Technology

B.E. IT 3/4 II Sem Examination May 2017

Key for Data warehousing and Data Mining (Code: IT17322)

1. Differentiate between classification and prediction. (3)

   Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts. The model are derived based on the analysis of a set of **training data** (i.e., data objects for which the class labels are known). The model is used to predict the class label of objects for which the class label is unknown.

   Whereas classification predicts categorical (discrete, unordered) labels, **regression** models continuous-valued functions. That is, regression is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels. The term *prediction* refers to both numeric prediction and class label prediction. **Regression analysis** is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Regression also encompasses the identification of distribution *trends* based on the available data.

2. What are the various types of OLAP operations? (3)

   Roll-up: The roll-up operation (also called the *drill-up* operation by some vendors) performs aggregation on a data cube, either by *climbing up a concept hierarchy* for a dimension or by *dimension reduction*.

   Drill-down: Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either *stepping down a concept hierarchy* for a dimension or *introducing additional dimensions*.

   Slice and dice: The *slice* operation performs a selection on one dimension of the given cube, resulting in a subcube.

   Pivot (rotate): *Pivot* (also called *rotate*) is a visualization operation that rotates the data axes in view to provide an alternative data presentation

   Other OLAP operations: Some OLAP systems offer additional drilling operations. For example, **drill-across** executes queries involving (i.e., across) more than one fact table. The **drill-through** operation uses relational SQL facilities to drill through the bottom level of a data cube down to its back-end relational tables.

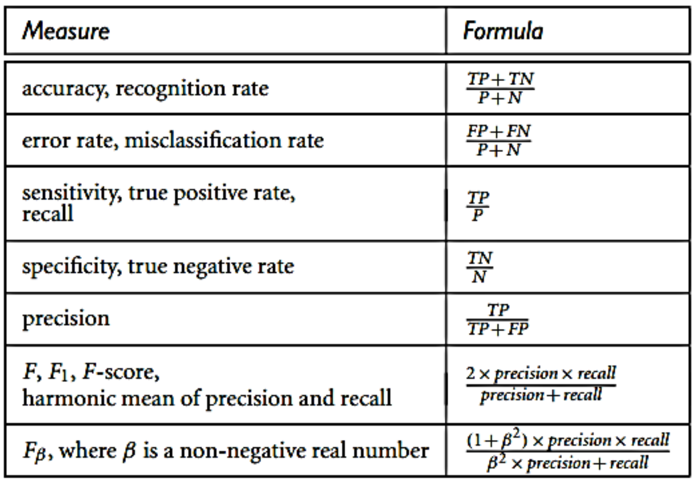
3. What is a data mart? (2)

   A data warehouse collects information about subjects that span the *entire organization*, such as *customers*, *items*, *sales*, *assets*, and *personnel*, and thus its scope is *enterprise-wide*. For data warehouses, the fact constellation schema is commonly used, since it can model multiple, interrelated subjects. A **data mart**, on the other hand, is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is *department wide*. For data marts, the *star* or *snowflake* schema is commonly used, since both are geared toward modeling single subjects, although the star schema is more popular and efficient.

4. Distinguish between agglomerative and divisive clustering (3)
Hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*, based on how the hierarchical decomposition is formed. The *agglomerative approach*, also called the *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups close to one another, until all the groups are merged into one (the topmost level of the hierarchy), or a termination condition holds. The *divisive approach*, also called the *top-down* approach, starts with all the objects in the same cluster. In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or a termination condition holds.

5. What is meant by accuracy? How do you test the classification accuracy? (3)

The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. The associated class label of each test tuple is compared with the learned classifier's class prediction for that tuple



Measure	*Formula*
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

Evaluation measures. Note that some measures are known by more than one name. TP, TN, FP, P, N refer to the number of true positive, true negative, false positive, positive, and negative samples, respectively (see text).

True positives (TP): These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives.
True negatives(TN): These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives.
False positives (FP): These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys computer = no for which the classifier predicted buys computer = yes). Let FP be the number of false positives.
False negatives (FN): These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys computer = yes for which the classifier predicted buys computer = no). Let FN be the number of false negatives.

6. Define an outlier (2)
An outlier is a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism.

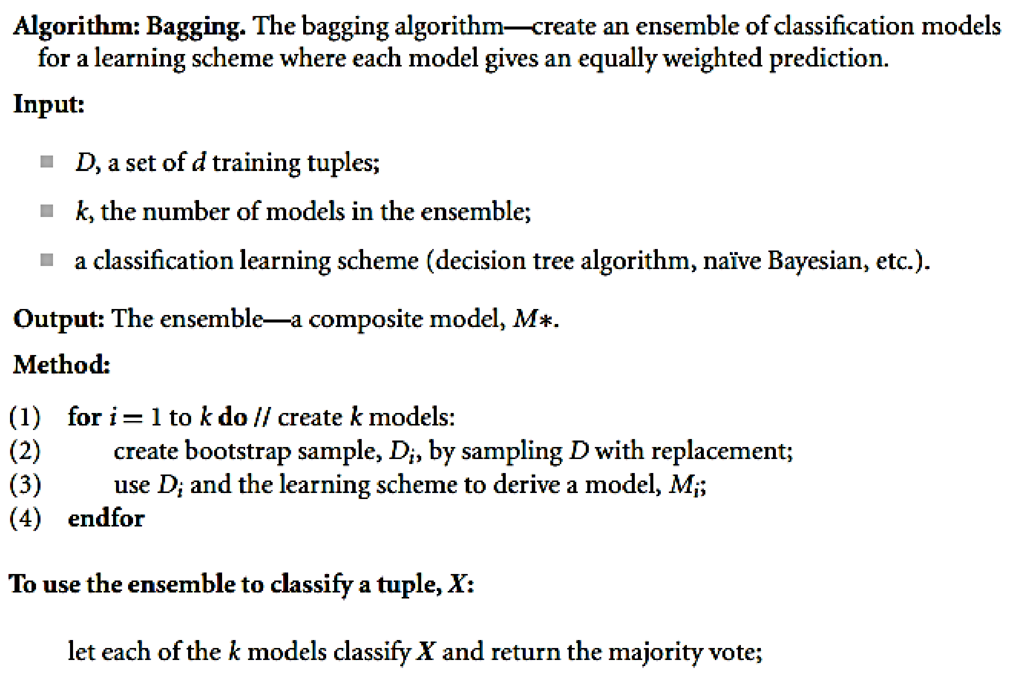
7. States Bayes theorem (2)

Let X be a data tuple. In Bayesian terms, X is considered “evidence.” As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis such as that the data tuple X belongs to a specified class C. For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the “evidence” or observed data tuple X. In other words, we are looking for the probability that tuple X belongs to class C, given that we know the attribute description of X.

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) = P(X|H)P(H) / P(X)

8. Write about bagging (2)

Given a set, D, of d tuples, bagging works as follows. For iteration i(i = 1, 2,..., k), a training set, Di , of d tuples is sampled with replacement from the original set of tuples, D. Note that the term bagging stands for bootstrap aggregation. Each training set is a bootstrap sample. Because sampling with replacement is used, some of the original tuples of D may not be included in Di, whereas others may occur more than once. A classifier model, Mi , is learned for each training set, Di. To classify an unknown tuple, X, each classifier, Mi , returns its class prediction, which counts as one vote. The bagged classifier, M∗, counts the votes and assigns the class with the most votes to X. Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.



Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, $M*$.

Method:

```
(1)  for $i = 1$ to $k$ do // create $k$ models:
(2)      create bootstrap sample, $D_i$, by sampling $D$ with replacement;
(3)      use $D_i$ and the learning scheme to derive a model, $M_i$;
(4)  endfor
```

To use the ensemble to classify a tuple, X:

let each of the k models classify X and return the majority vote;

9. Define IQR and five number summary (3)

The quartiles give an indication of a distribution’s center, spread, and shape. The first quartile, denoted by Q1, is the 25th percentile. It cuts off the lowest 25% of the data. The third quartile, denoted by Q3, is the 75th percentile—it cuts off the lowest 75% (or highest 25%) of the data. The second quartile is the 50th percentile. As the median, it gives the center of the data distribution. The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the interquartile range (IQR) and is defined as IQR = Q3 − Q1.

The five-number summary of a distribution consists of the median (Q2), the quartiles Q1 and Q3, and the smallest and largest individual observations, written in the order of Minimum, Q1, Median, Q3, Maximum

10. Define web mining and multimedia data mining (2)
Web mining can help us learn about the distribution of information on the WWW in general, characterize and classify web pages, and uncover web dynamics and the association and other relationships among different web pages, users, communities, and web-based activities

Multimedia data mining is the discovery of interesting patterns from multimedia databases that store and manage large collections of multimedia objects, including image data, video data, audio data, as well as sequence data and hypertext data containing text, text markups, and linkages. Multimedia data mining is an interdisciplinary field that integrates image processing and understanding, computer vision, data mining, and pattern recognition. Issues in multimedia data mining include content-based retrieval and similarity search, and generalization and multidimensional analysis.

11. a. With a neat diagram explain the process of KDD

The knowledge discovery process is an iterative sequence of the following steps:

i. Data cleaning (to remove noise and inconsistent data)
ii. Data integration (where multiple data sources may be combined)
iii. Data selection (where data relevant to the analysis task are retrieved from the database)
iv. Data transformation (where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations)
v. Data mining (an essential process where intelligent methods are applied to extract data patterns)
vi. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on interestingness measures
vii. Knowledge presentation (where visualization and knowledge representation techniques are used to present mined knowledge to users)

Steps i through iv are different forms of data preprocessing, where data are prepared for mining. The data mining step may interact with the user or a knowledge base. The interesting patterns are presented to the user and may be stored as new knowledge in the knowledge base. The preceding view shows data mining as one step in the knowledge discovery process, albeit an essential one because it uncovers hidden patterns for evaluation. However, in industry, in media, and in the research milieu, the term data mining is often used to refer to the entire knowledge discovery process (perhaps because the term is shorter than knowledge discovery from data). Therefore, we adopt a broad view of data mining functionality: Data mining is the process of discovering interesting patterns and knowledge from large amounts of data. The data sources can include databases, data warehouses, the Web, other information repositories, or data that are streamed into the system dynamically.

b. Explain data mining functionalities

Data mining functionalities are used to specify the kinds of patterns to be found in data mining tasks. In general, such tasks can be classified into two categories: **descriptive** and **predictive**. Descriptive mining tasks characterize properties of the data in a target data set. Predictive mining tasks perform induction on the current data in order to make predictions.

There are a number of ***data mining functionalities***. These include the following:

i. characterization and discrimination
Data characterization is a summarization of the general characteristics or features of a target class of data.
The output of data characterization can be presented in various forms. Examples include **pie charts**, **bar charts**, **curves**, **multidimensional data cubes**, and **multidimensional tables**, including crosstabs. The resulting descriptions can also be presented as **generalized relations** or in rule form (called **characteristic rules**).
Data discrimination is a comparison of the general features of the target class data objects against the general features of objects from one or multiple contrasting classes. The target and contrasting classes can be specified by a user, and the corresponding data objects can be retrieved through database queries.
The forms of output presentation are similar to those for characteristic descriptions, although discrimination descriptions should include comparative measures that help to distinguish between the target and contrasting classes. Discrimination descriptions expressed in the form of rules are referred to as **discriminant rules**.

ii. the mining of frequent patterns, associations, and correlations
Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including frequent itemsets, frequent subsequences (also known as sequential patterns), and frequent substructures. A *frequent itemset* typically refers to a set of items that often appear together in a transactional data set—for example, milk and bread, which are frequently bought together in grocery stores by many customers. A frequently occurring subsequence, such as the pattern that customers, tend to purchase first a laptop, followed by a digital camera, and then a memory card, is a (*frequent*) *sequential pattern*. A substructure can refer to different structural forms (e.g., graphs, trees, or lattices) that may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (*frequent*) *structured pattern*. Mining frequent patterns leads to the discovery of interesting associations and correlations within data.

iii. classification and regression
Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts. The model are derived based on the analysis of a set of **training data** (i.e., data objects for which the class labels are known). The model is used to predict the class label of objects for which the the class label is unknown. *"How is the derived model presented?"* The derived model may be represented in various forms, such as *classification rules* (i.e., *IF-THEN rules*), *decision trees*, *mathematical formulae*, or *neural networks*

Whereas classification predicts categorical (discrete, unordered) labels, **regression** models continuous-valued functions. That is, regression is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels. The term *prediction* refers to both numeric prediction and class label prediction. **Regression analysis** is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Regression also encompasses the identification of distribution *trends* based on the available data.

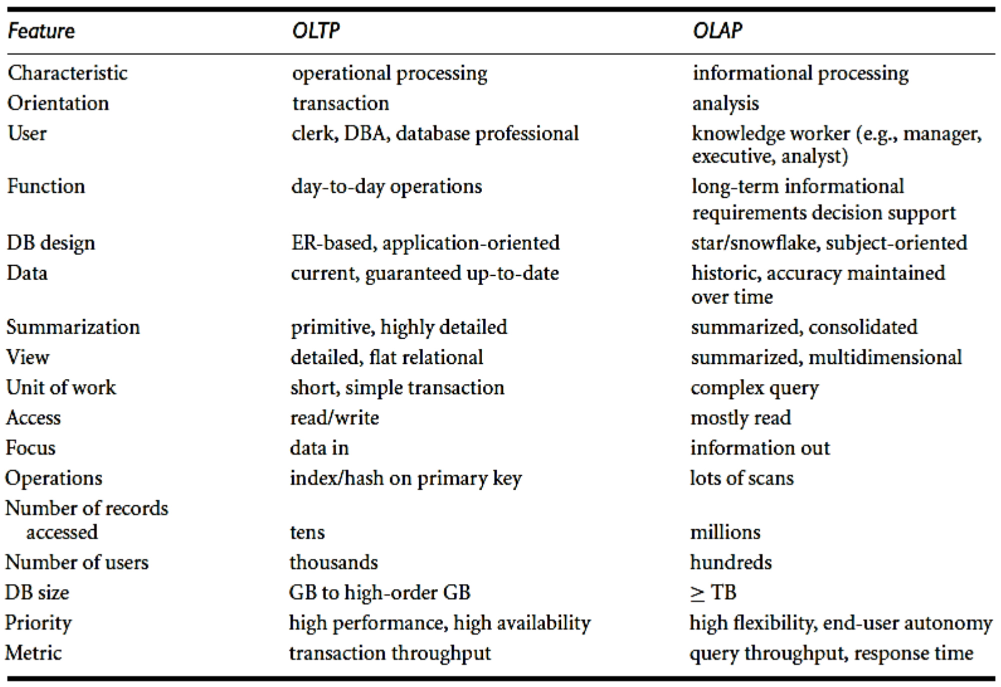
iv. clustering analysis

Clustering analyzes data objects without consulting class labels. In many cases, class labeled data may simply not exist at the beginning. Clustering can be used to generate class labels for a group of data. The objects are clustered or grouped based on the principle of *maximizing the intraclass similarity and minimizing the interclass similarity*. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are rather dissimilar to objects in other clusters. Each cluster so formed can be viewed as a class of objects, from which rules can be derived. Clustering can also facilitate **taxonomy formation**, that is, the organization of observations into a hierarchy of classes that group similar events together

v. outlier analysis

A data set may contain objects that do not comply with the general behavior or model of the data. These data objects are **outliers**. Many data mining methods discard outliers as noise or exceptions. However, in some applications (e.g., fraud detection) the rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as **outlier analysis** or **anomaly mining**. Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are remote from any other cluster are considered outliers. Rather than using statistical or distance measures, density-based methods may identify outliers in a local region, although they look normal from a global statistical distribution view

12. a. Distinguish between OLAP and OLTP



Feature	OLTP	OLAP
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements decision support
DB design	ER-based, application-oriented	star/snowflake, subject-oriented
Data	current, guaranteed up-to-date	historic, accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	GB to high-order GB	≥ TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

b. Discuss about pattern mining in multilevel and multidimensional space with examples

Association rules generated from mining data at multiple abstraction levels are called **multiple-level** or **multilevel association rules**. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework. In general, a top-down strategy is employed, where counts are accumulated for the calculation of frequent itemsets at each concept level, starting at concept level 1 and working downward in the hierarchy toward the more specific concept levels, until no more frequent itemsets can be found. For each level, any algorithm for discovering frequent itemsets may be used, such as Apriori or its variations. A number of variations to this approach are described next, where each variation involves "playing" with the support threshold in a slightly different way.

Using uniform minimum support for all levels (referred to as uniform support):
The same minimum support threshold is used when mining at each abstraction level. For example, in Figure 7.3, a minimum support threshold of 5% is used throughout (e.g., for mining from"computer" downward to "laptop computer"). Both "computer" and "laptop computer" are found to be frequent, whereas "desktop computer" is not. When a uniform minimum support threshold is used, the search procedure is simplified. The method is also simple in that users are required to specify only one minimum support threshold.

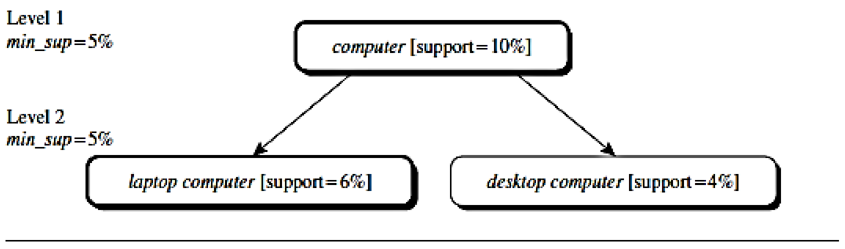


Figure 7.3 Multilevel mining with uniform support.

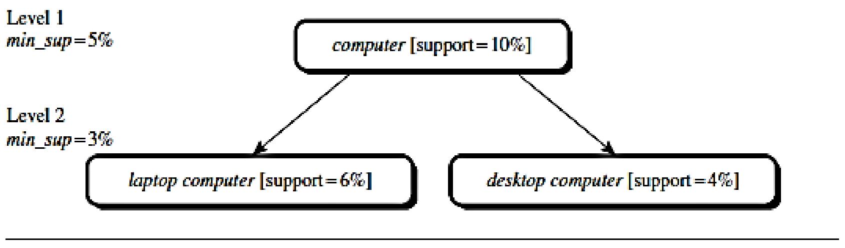


Figure 7.4 Multilevel mining with reduced support.

An Apriori-like optimization technique can be adopted, based on the knowledge that an ancestor is a superset of its descendants: The search avoids examining itemsets containing any item of which the ancestors do not have minimum support. The uniform support approach, however, has some drawbacks. It is unlikely that items at lower abstraction levels will occur as frequently as those at higher abstraction levels. If the minimum support threshold is set too high, it could miss some meaningful associations occurring at low abstraction levels. If the threshold is set too low, it may generate many uninteresting associations occurring at high abstraction levels. This provides the motivation for the next approach.
Using reduced minimum support at lower levels (referred to as reduced support): Each abstraction level has its own minimum support threshold. The deeper the abstraction level, the smaller the corresponding threshold. For example, in Figure 7.4, the minimum support thresholds for levels 1 and 2 are 5% and 3%, respectively. In this way, "computer," "laptop computer," and "desktop computer" are all considered frequent.
Using item or group-based minimum support (referred to as group-based support):
Because users or experts often have insight as to which groups are more important than others, it is sometimes more desirable to set up user-specific, item, or group-based minimal support thresholds when mining multilevel rules. For example, a user could set up the minimum support thresholds based on product price or on items of interest, such as by setting particularly low support thresholds for "camera with price over $1000" or "Tablet PC," to pay particular attention to the association patterns containing items in these categories.

we can therefore mine association rules containing *multiple* predicates such as

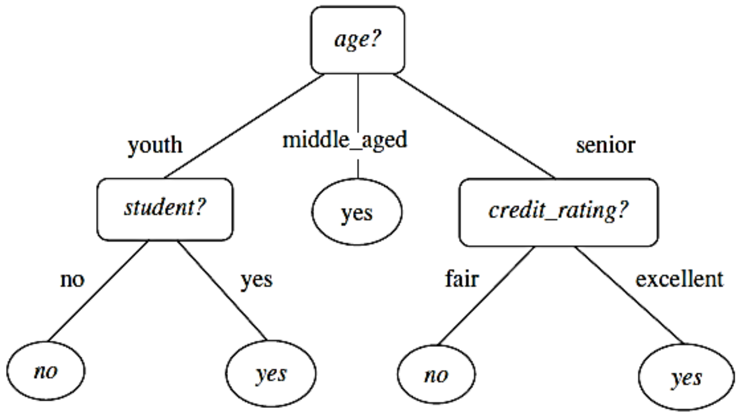
$$age(X, \text{"20...29"}) \wedge occupation(X, \text{"student"}) \Rightarrow buys(X, \text{"laptop"}). \qquad (7.7)$$

Association rules that involve two or more dimensions or predicates can be referred to as **multidimensional association rules**. Rule (7.7) contains three predicates (*age*, *occupation*, and *buys*), each of which occurs *only once* in the rule. Hence, we say that it has **no repeated predicates.** Multidimensional association rules with no repeated predicates are called **interdimensional association rules.** We can also mine multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called **hybrid-dimensional association rules.** An example of such a rule is the following, where the predicate *buys* is repeated:

$$age(X, \text{"20...29"}) \wedge buys(X, \text{"laptop"}) \Rightarrow buys(X, \text{"HP printer"}). \qquad (7.8)$$

13. a. Explain classification by decision tree induction

Decision tree induction is the learning of decision trees from class-labeled training tuples. A **decision tree** is a flowchart-like tree structure, where each **internal node** (nonleaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node** (or *terminal node*) holds a class label. The topmost node in a tree is the **root** node. A typical decision tree is



A decision tree for the concept *buys_computer*, indicating whether an *AllElectronics* customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

"How are decision trees used for classification?" Given a tuple, ***X***, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.
"Why are decision tree classifiers so popular?" The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle multidimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast. In general, decision tree classifiers have good accuracy

During tree construction, attribute selection measures are used to select the attribute that best partitions the tuples into distinct classes. Popular measures of attribute selection are Information gain, Gini Index and Gain ratio

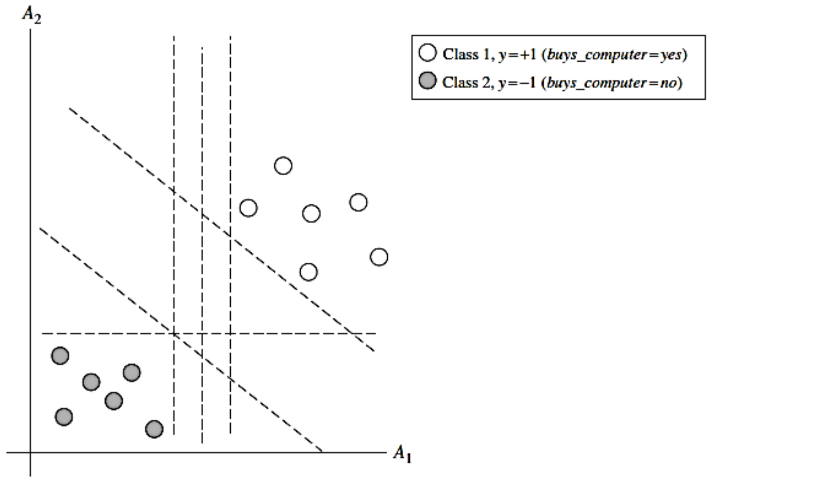
b. Write about Support vector machine in detail

It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e., a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors).

Although the training time of even the fastest SVMs can be extremely slow, they are highly accurate, owing to their ability to model complex nonlinear decision boundaries. They are much less prone to overfitting than other methods. The support vectors found also provide a compact description of the learned model. SVMs can be used for numeric prediction as well as classification. They have been applied to a number of areas, including handwritten digit recognition, object recognition, and speaker identification, as well as benchmark time-series prediction tests.

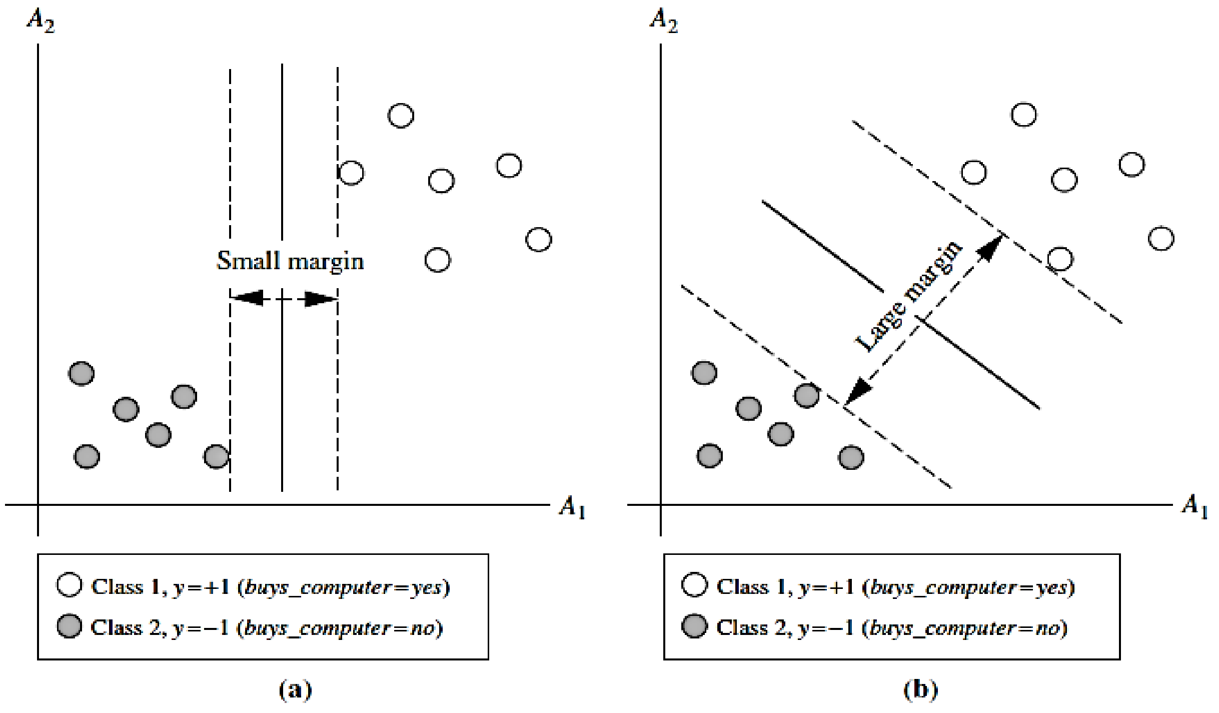
The Case When the Data Are Linearly Separable

To explain the mystery of SVMs, let's first look at the simplest case—a two-class problem where the classes are linearly separable. Let the data set D be given as (X1, y1), (X2, y2), … , (XjDj, yjDj), where Xi is the set of training tuples with associated class labels, yi . Each yi can take one of two values, either +1 or -1 corresponding to the classes buys computer D yes and buys computer D no, respectively



The 2-D training data are linearly separable. There are an infinite number of possible separating hyperplanes or "decision boundaries," some of which are shown here as dashed lines. Which one is best?

To aid in visualization, let's consider an example based on two input attributes, *A*1 and *A*2, as shown in above Figure. From the graph, we see that the 2-D data are **linearly separable** (or "linear," for short), because a straight line can be drawn to separate all the tuples of class +1 from all the tuples of class -1 There are an infinite number of separating lines that could be drawn. We want to find the "best" one, that is, one that (we hope) will have the minimum classification error on previously unseen tuples. How can we find this best line? Note that if our data were 3-D (i.e., with three attributes), we would want to find the best separating *plane*. Generalizing to *n* dimensions, we want to find the best *hyperplane*. We will use "hyperplane" to refer to the decision boundary that we are seeking, regardless of the number of input attributes. So, in other words, how can we find the best hyperplane? An SVM approaches this problem by searching for the maximum marginal hyperplane. Consider below Figure, which shows two possible separating hyperplanes and their associated margins. Before we get into the definition of margins, let's take an intuitive look at this figure. Both hyperplanes can correctly classify all the given data tuples. Intuitively, however, we expect the hyperplane with the larger margin to be more accurate at classifying future data tuples than the hyperplane with the smaller margin. This is why (during the learning or training phase) the SVM searches for the hyperplane with the largest margin, that is, the maximum marginal hyperplane (MMH). The associated margin gives the largest separation between classes.


A_2
Small margin
A_1
○ Class 1, y=+1 (buys_computer=yes)
● Class 2, y=−1 (buys_computer=no)
(a)
A_2
Large margin
A_1
○ Class 1, y=+1 (buys_computer=yes)
● Class 2, y=−1 (buys_computer=no)
(b)

Here we see just two possible separating hyperplanes and their associated margins. Which one is better? The one with the larger margin (b) should have greater generalization accuracy.

Getting to an informal definition of **margin**, we can say that the shortest distance from a hyperplane to one side of its margin is equal to the shortest distance from the hyperplane to the other side of its margin, where the "sides" of the margin are parallel to the hyperplane. When dealing with the MMH, this distance is, in fact, the shortest distance from the MMH to the closest training tuple of either class.

A separating hyperplane can be written as

$$W \cdot X + b = 0, \tag{9.12}$$

where W is a weight vector, namely, $W = \{w_1, w_2, \ldots, w_n\}$; n is the number of attributes; and b is a scalar, often referred to as a bias. To aid in visualization, let's consider two input attributes, A_1 and A_2, as in Figure 9.8(b). Training tuples are 2-D (e.g., $X = (x_1, x_2)$), where x_1 and x_2 are the values of attributes A_1 and A_2, respectively, for X. If we think of b as an additional weight, w_0, we can rewrite Eq. (9.12) as

$$w_0 + w_1x_1 + w_2x_2 = 0. \tag{9.13}$$

Thus, any point that lies above the separating hyperplane satisfies

$$w_0 + w_1x_1 + w_2x_2 > 0. \tag{9.14}$$

Similarly, any point that lies below the separating hyperplane satisfies

$$w_0 + w_1x_1 + w_2x_2 < 0. \tag{9.15}$$

The weights can be adjusted so that the hyperplanes defining the "sides" of the margin can be written as

$$H_1 : w_0 + w_1x_1 + w_2x_2 \geq 1 \quad \text{for } y_i = +1, \tag{9.16}$$

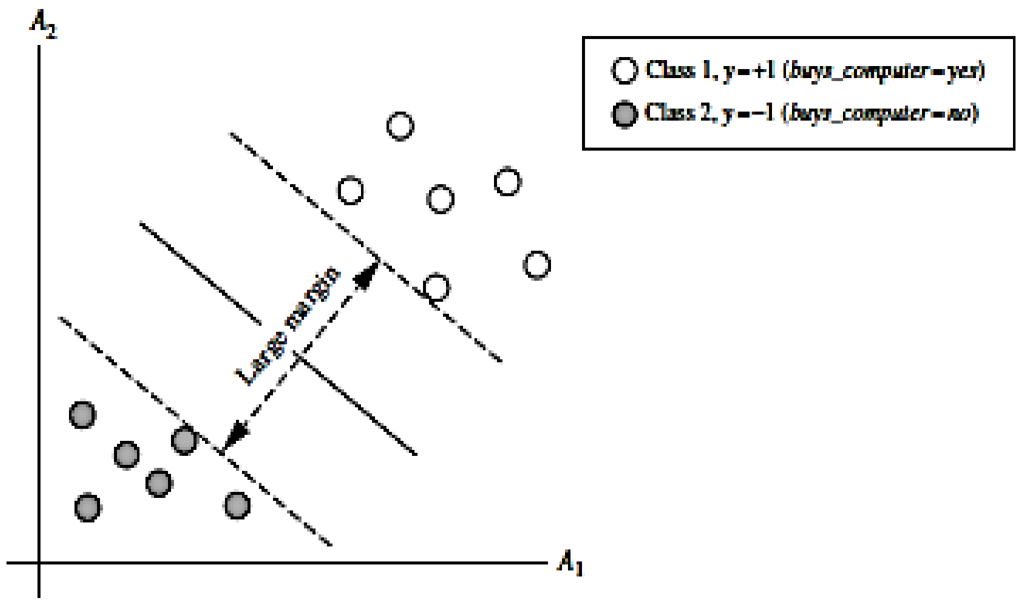
$$H_2 : w_0 + w_1x_1 + w_2x_2 \leq -1 \quad \text{for } y_i = -1. \tag{9.17}$$

That is, any tuple that falls on or above H_1 belongs to class +1, and any tuple that falls on or below H_2 belongs to class −1. Combining the two inequalities of Eqs. (9.16) and (9.17), we get

$$y_i(w_0 + w_1x_1 + w_2x_2) \geq 1, \ \forall i. \tag{9.18}$$

Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the "sides" defining the margin) satisfy Eq. (9.18) and are called **support vectors**. That is, they are equally close to the (separating) MMH. In Figure 9.9, the support vectors are shown encircled with a thicker border. Essentially, the support vectors are the most difficult tuples to classify and give the most information regarding classification.

From this, we can obtain a formula for the size of the maximal margin. The distance from the separating hyperplane to any point on H_1 is $\frac{1}{||W||}$, where $||W||$ is the Euclidean norm of W, that is, $\sqrt{W \cdot W}$.[2] By definition, this is equal to the distance from any point on H_2 to the separating hyperplane. Therefore, the maximal margin is $\frac{2}{||W||}$.



A_2
A_1
Large margin
○ Class 1, y=+1 (buys_computer=yes)
● Class 2, y=−1 (buys_computer=no)

Support vectors. The SVM finds the maximum separating hyperplane, that is, the one with maximum distance between the nearest training tuples. The support vectors are shown with a thicker border.

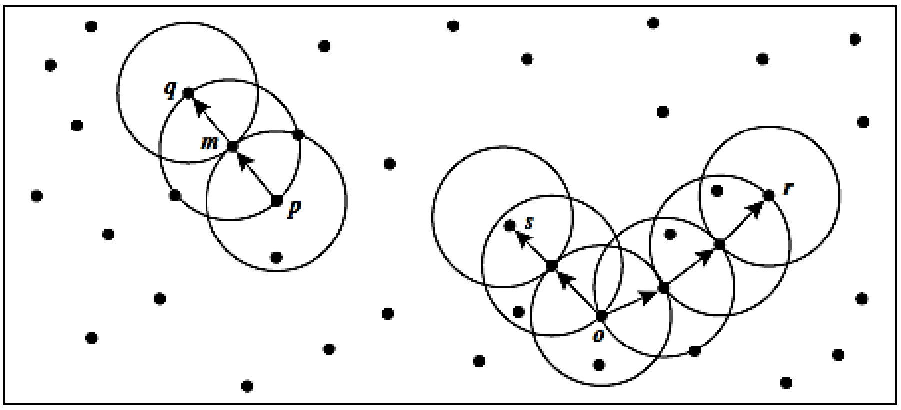
14. a. Explain about DBSCAN clustering algorithm

DBSCAN: Density-Based Clustering Based on Connected Regions with High Density

"How can we find dense regions in density-based clustering?" The *density* of an object o can be measured by the number of objects close to o. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) finds *core objects*, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters.

"How does DBSCAN *quantify the neighborhood of an object?"* A user-specified parameter $\epsilon > 0$ is used to specify the radius of a neighborhood we consider for every object. The **ϵ-neighborhood** of an object o is the space within a radius ϵ centered at o.

Due to the fixed neighborhood size parameterized by ϵ, the density of a neighborhood can be measured simply by the number of objects in the neighborhood. To determine whether a neighborhood is dense or not, DBSCAN uses another user-specified



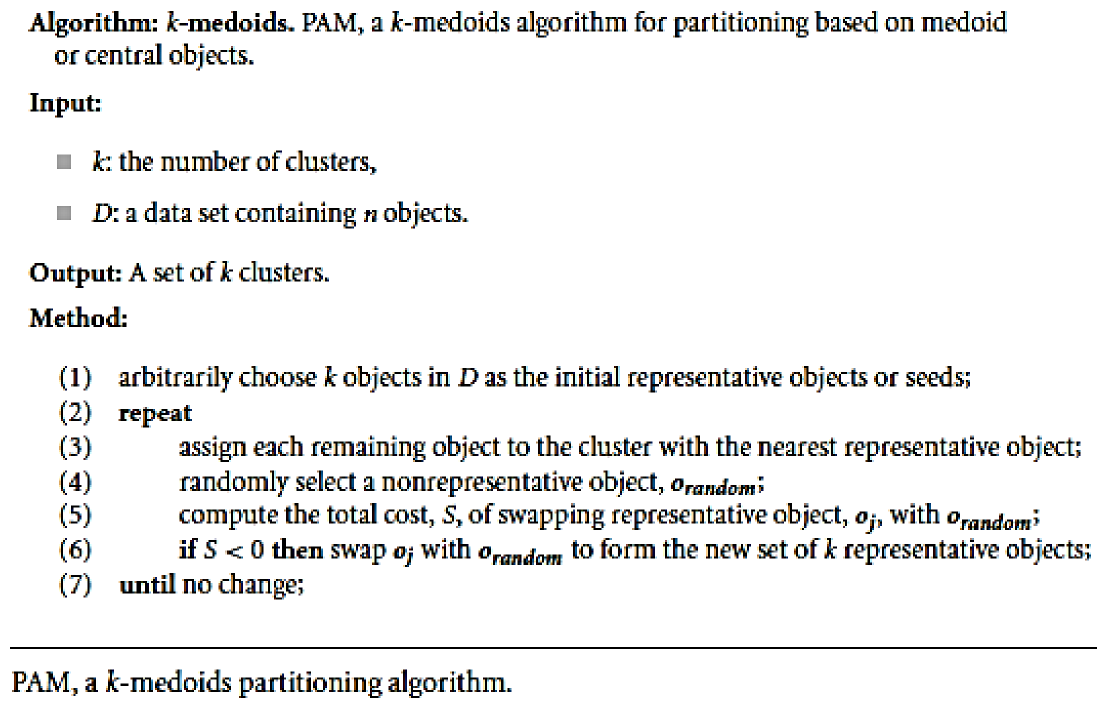
Density-reachability and density-connectivity in density-based clustering. *Source:* Based on Ester, Kriegel, Sander, and Xu [EKSX96].

"How does DBSCAN find clusters?" Initially, all objects in a given data set D are marked as "unvisited." DBSCAN randomly selects an unvisited object p, marks p as "visited," and checks whether the ϵ-neighborhood of p contains at least *MinPts* objects. If not, p is marked as a noise point. Otherwise, a new cluster C is created for p, and all the objects in the ϵ-neighborhood of p are added to a candidate set, N. DBSCAN iteratively adds to C those objects in N that do not belong to any cluster. In this process, for an object p' in N that carries the label "unvisited," DBSCAN marks it as "visited" and checks its ϵ-neighborhood. If the ϵ-neighborhood of p' has at least *MinPts* objects, those objects in the ϵ-neighborhood of p' are added to N. DBSCAN continues adding objects to C until C can no longer be expanded, that is, N is empty. At this time, cluster C is completed, and thus is output.

To find the next cluster, DBSCAN randomly selects an unvisited object from the remaining ones. The clustering process continues until all objects are visited. The pseudocode of the DBSCAN algorithm is given in Figure 10.15.

If a spatial index is used, the computational complexity of DBSCAN is $O(n \log n)$, where n is the number of database objects. Otherwise, the complexity is $O(n^2)$. With appropriate settings of the user-defined parameters, ϵ and *MinPts*, the algorithm is effective in finding arbitrary-shaped clusters.

b. Briefly explain about K Mediods clustering algorithm



Algorithm: ***k*-medoids.** PAM, a k-medoids algorithm for partitioning based on medoid or central objects.

Input:

- k: the number of clusters,
- D: a data set containing n objects.

Output: A set of k clusters.

Method:

```
(1) arbitrarily choose $k$ objects in $D$ as the initial representative objects or seeds;
(2) repeat
(3)     assign each remaining object to the cluster with the nearest representative object;
(4)     randomly select a nonrepresentative object, $o_{random}$;
(5)     compute the total cost, $S$, of swapping representative object, $o_j$, with $o_{random}$;
(6)     if $S < 0$ then swap $o_j$ with $o_{random}$ to form the new set of $k$ representative objects;
(7) until no change;
```

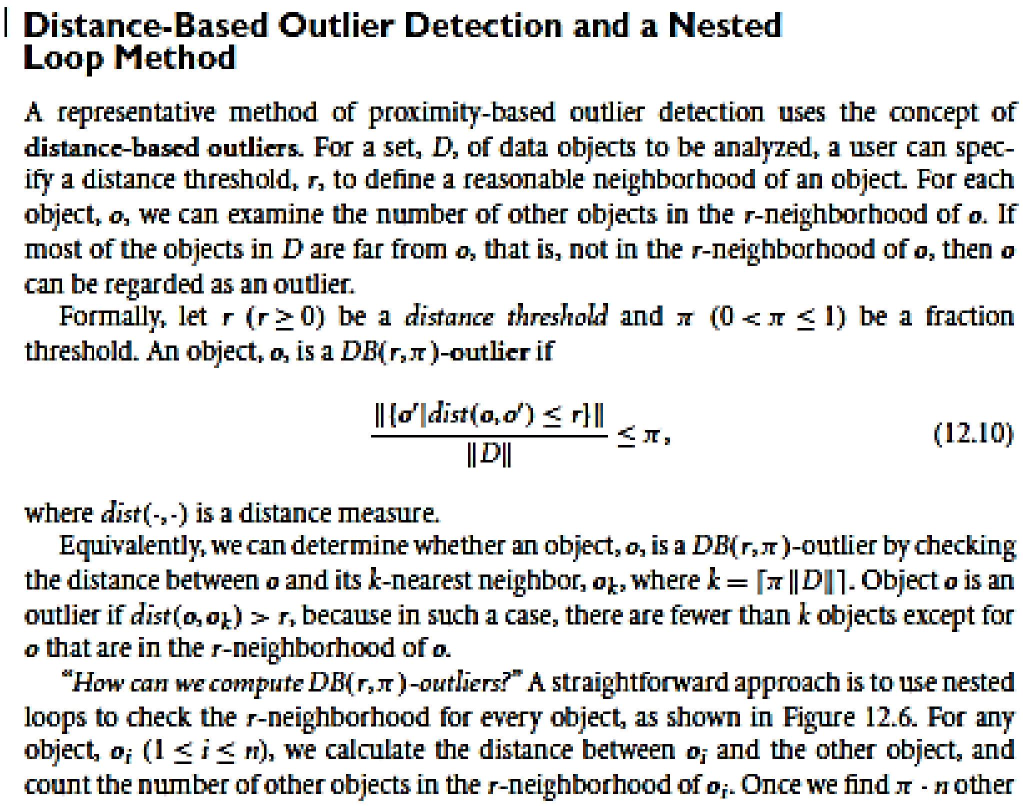
PAM, a k-medoids partitioning algorithm.

15. a. Write about the various data mining application

Data Mining for Financial Data Analysis:
Design and construction of data warehouses for multidimensional data analysis and data mining
Loan payment prediction and customer credit policy analysis
Classification and clustering of customers for targeted marketing:
Detection of money laundering and other financial crimes
Data Mining for Retail and Telecommunication Industries:
Design and construction of data warehouses
Multidimensional analysis of sales, customers, products, time, and region
Analysis of the effectiveness of sales campaigns
Customer retention—analysis of customer loyalty:
Product recommendation and cross-referencing of items:
Fraudulent analysis and the identification of unusual patterns:
Data Mining in Science and Engineering
Data warehouses and data preprocessing:
Mining complex data types
Graph-based and network-based mining: It
Visualization tools and domain-specific knowledge:
Data mining in engineering
data mining in social science and social studies
Data mining in computer science
Data Mining for Intrusion Detection and Prevention
Signature-based detection:
Anomaly-based detection

New data mining algorithms for intrusion detection
Association, correlation, and discriminative pattern analyses help select and build discriminative classifiers
Analysis of stream data
Distributed data mining
Visualization and querying tools:
Data Mining and Recommender Systems

b. Explain distance based outlier detection



Distance-Based Outlier Detection and a Nested Loop Method

A representative method of proximity-based outlier detection uses the concept of **distance-based outliers**. For a set, D, of data objects to be analyzed, a user can specify a distance threshold, r, to define a reasonable neighborhood of an object. For each object, o, we can examine the number of other objects in the r-neighborhood of o. If most of the objects in D are far from o, that is, not in the r-neighborhood of o, then o can be regarded as an outlier.

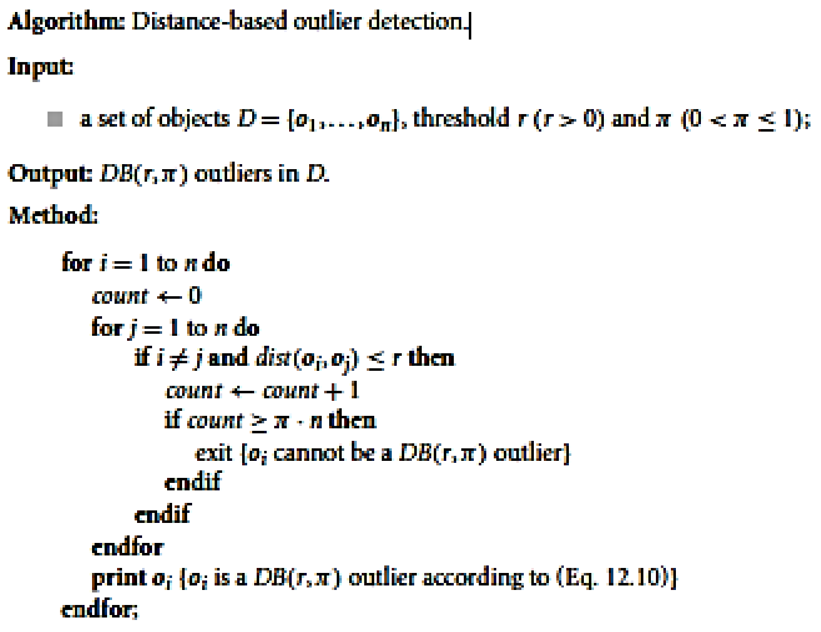
Formally, let r ($r \geq 0$) be a *distance threshold* and π ($0 < \pi \leq 1$) be a fraction threshold. An object, o, is a $DB(r, \pi)$**-outlier** if

$$\frac{\|\{o' | dist(o, o') \leq r\}\|}{\|D\|} \leq \pi, \tag{12.10}$$

where $dist(\cdot, \cdot)$ is a distance measure.

Equivalently, we can determine whether an object, o, is a $DB(r, \pi)$-outlier by checking the distance between o and its k-nearest neighbor, o_k, where $k = \lceil \pi \|D\| \rceil$. Object o is an outlier if $dist(o, o_k) > r$, because in such a case, there are fewer than k objects except for o that are in the r-neighborhood of o.

"How can we compute $DB(r, \pi)$-outliers?" A straightforward approach is to use nested loops to check the r-neighborhood for every object, as shown in Figure 12.6. For any object, o_i ($1 \leq i \leq n$), we calculate the distance between o_i and the other object, and count the number of other objects in the r-neighborhood of o_i. Once we find $\pi \cdot n$ other



Algorithm: Distance-based outlier detection.

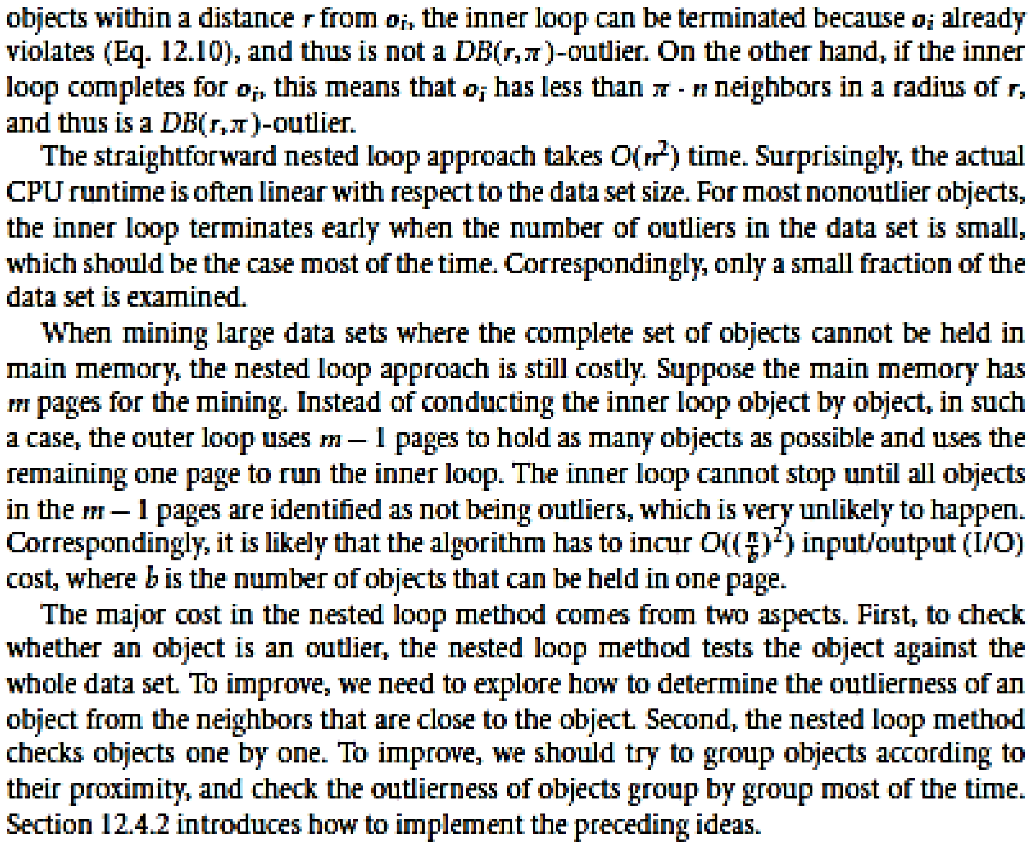
Input:

- a set of objects $D = \{o_1, \ldots, o_n\}$, threshold r ($r > 0$) and π ($0 < \pi \leq 1$);

Output: $DB(r, \pi)$ outliers in D.

Method:

```
for $i = 1$ to $n$ do
   $count \leftarrow 0$
   for $j = 1$ to $n$ do
      if $i \neq j$ and $dist(o_i, o_j) \leq r$ then
         $count \leftarrow count + 1$
         if $count \geq \pi \cdot n$ then
            exit {$o_i$ cannot be a $DB(r, \pi)$ outlier}
         endif
      endif
   endfor
   print $o_i$ {$o_i$ is a $DB(r, \pi)$ outlier according to (Eq. 12.10)}
endfor;
```



objects within a distance r from o_i, the inner loop can be terminated because o_i already violates (Eq. 12.10), and thus is not a $DB(r,\pi)$-outlier. On the other hand, if the inner loop completes for o_i, this means that o_i has less than $\pi \cdot n$ neighbors in a radius of r, and thus is a $DB(r,\pi)$-outlier.

The straightforward nested loop approach takes $O(n^2)$ time. Surprisingly, the actual CPU runtime is often linear with respect to the data set size. For most nonoutlier objects, the inner loop terminates early when the number of outliers in the data set is small, which should be the case most of the time. Correspondingly, only a small fraction of the data set is examined.

When mining large data sets where the complete set of objects cannot be held in main memory, the nested loop approach is still costly. Suppose the main memory has m pages for the mining. Instead of conducting the inner loop object by object, in such a case, the outer loop uses $m-1$ pages to hold as many objects as possible and uses the remaining one page to run the inner loop. The inner loop cannot stop until all objects in the $m-1$ pages are identified as not being outliers, which is very unlikely to happen. Correspondingly, it is likely that the algorithm has to incur $O((\frac{n}{b})^2)$ input/output (I/O) cost, where b is the number of objects that can be held in one page.

The major cost in the nested loop method comes from two aspects. First, to check whether an object is an outlier, the nested loop method tests the object against the whole data set. To improve, we need to explore how to determine the outlierness of an object from the neighbors that are close to the object. Second, the nested loop method checks objects one by one. To improve, we should try to group objects according to their proximity, and check the outlierness of objects group by group most of the time. Section 12.4.2 introduces how to implement the preceding ideas.

16. a. Explain about spatial data mining

Spatial data mining discovers patterns and knowledge from spatial data. Spatial data, in many cases, refer to geospace-related data stored in geospatial data repositories. The data can be in "vector" or "raster" formats, or in the form of imagery and geo-referenced multimedia. Recently, large *geographic data warehouses* have been constructed by integrating thematic and geographically referenced data from multiple sources. From these, we can construct *spatial data cubes* that contain spatial dimensions and measures, and support *spatial OLAP* for *multidimensional spatial data analysis*. Spatial data mining can be performed on spatial data warehouses, spatial databases, and other geospatial data repositories. Popular topics on geographic knowledge discovery and spatial data mining include *mining spatial associations and co-location patterns*, *spatial clustering*, *spatial classification*, *spatial modeling*, and *spatial trend and outlier analysis*.

b. Write about the various types of data used in cluster analysis

Data sets are made up of data objects. A data object represents an entity—in a sales database, the objects may be customers, store items, and sales; in a medical database, the objects may be patients; in a university database, the objects may be students, professors, and courses. Data objects are typically described by attributes. Data objects can also be referred to as samples, examples, instances, data points, or objects. If the data objects are stored in a database, they are data tuples. That is, the rows of a database correspond to the data objects, and the columns correspond to the attributes.

An attribute is a data field, representing a characteristic or feature of a data object. The nouns attribute, dimension, feature, and variable are often used interchangeably in the literature. The term dimension is commonly used in data warehousing

Nominal Attributes

Nominal means "relating to names." The values of a nominal attribute are symbols or names of things. Each value represents some kind of category, code, or state, and so nominal attributes are also referred to as categorical. The values do not have any meaningful order. In computer science, the values are also known as enumerations.
Example of Nominal attributes. Suppose that hair color and marital status are two attributes describing person objects. In our application, possible values for hair color are black, brown, blond, red, auburn, gray, and white. The attribute marital status can take on the values single, married, divorced, and widowed. Both hair color and marital status are nominal attributes. Another example of a nominal attribute is occupation, with the values teacher, dentist, programmer, farmer, and so on.

Binary Attributes: A binary attribute is a nominal attribute with only two categories or states: 0 or 1, where 0 typically means that the attribute is absent, and 1 means that it is present. Binary attributes are referred to as Boolean if the two states correspond to true and false.
Example of Binary attributes. Given the attribute smoker describing a patient object, 1 indicates that the patient smokes, while 0 indicates that the patient does not.

A binary attribute is symmetric if both of its states are equally valuable and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. One such example could be the attribute gender having the states male and female.
A binary attribute is asymmetric if the outcomes of the states are not equally important, such as the positive and negative outcomes of a medical test for HIV. By convention, we code the most important outcome, which is usually the rarest one, by 1 (e.g., HIV positive) and the other by 0 (e.g., HIV negative).

Ordinal Attributes: An ordinal attribute is an attribute with possible values that have a meaningful order or ranking among them, but the magnitude between successive values is not known.
Example of Ordinal attributes. Suppose that drink size corresponds to the size of drinks available at a fast-food restaurant. This nominal attribute has three possible values: small, medium, and large. The values have a meaningful sequence (which corresponds to increasing drink size); however, we cannot tell from the values how much bigger, say, a medium is than a large.

Numeric Attributes: A numeric attribute is quantitative; that is, it is a measurable quantity, represented in integer or real values. Numeric attributes can be interval-scaled or ratio-scaled.
Interval-Scaled Attributes: Interval-scaled attributes are measured on a scale of equal-size units. The values of interval-scaled attributes have order and can be positive, 0, or negative. Thus, in addition to providing a ranking of values, such attributes allow us to compare and quantify the difference between values.
Interval-scaled attributes. A temperature attribute is interval-scaled. Suppose that we have the outdoor temperature value for a number of different days, where each day is an object. By ordering the values, we obtain a ranking of the objects with respect to temperature. In addition, we can quantify the difference between values.
Ratio-Scaled Attributes: A ratio-scaled attribute is a numeric attribute with an inherent zero-point. That is, if a measurement is ratio-scaled, we can speak of a value as being a multiple (or ratio) of another value. In addition, the values are ordered, and we can also compute the difference between values, as well as the mean, median, and mode.

16. a. Write about web usage mining

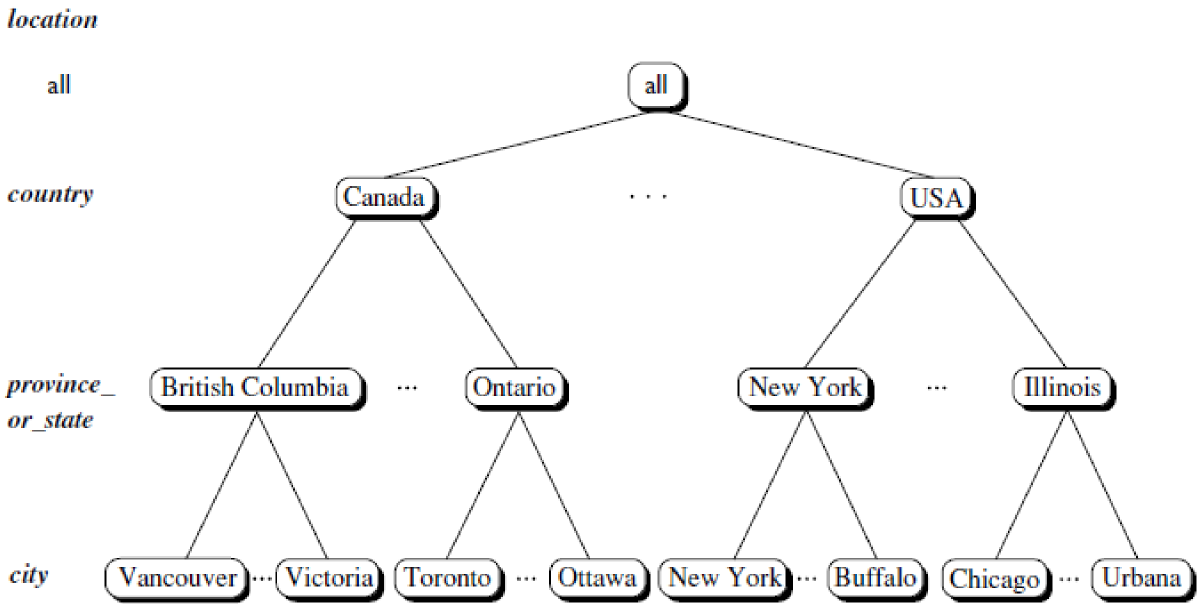
Web usage mining is the process of extracting useful information (e.g., user click streams) from server logs. It finds patterns related to general or particular groups of users; understands users' search patterns, trends, and associations; and predicts what users are looking for on the Internet. It helps improve search efficiency and effectiveness, as well as promotes products or related information to different groups of users at the right time. Web search companies routinely conduct web usage mining to improve their quality of service.

b. Write short notes on concept hierarchy

A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. Consider a concept hierarchy for the dimension location. City values for location include Vancouver, Toronto, New York, and Chicago. Each city, however, can be mapped to the province or state to which it belongs. For example, Vancouver can be mapped to British Columbia, and Chicago to Illinois. The provinces and states can in turn be mapped to the country (e.g., Canada or the United States) to which they belong. These mappings form a concept hierarchy for the

dimension *location*, mapping a set of low-level concepts (i.e., cities) to higher-level, more general concepts (i.e., countries). This concept hierarchy is illustrated in Figure 4.9.

Many concept hierarchies are implicit within the database schema. For example, suppose that the dimension *location* is described by the attributes *number, street, city, province_or_state, zip_code*, and *country*. These attributes are related by a total order, forming a concept hierarchy such as "*street* < *city* < *province_or_state* < *country*." This hierarchy is shown in Figure 4.10(a). Alternatively, the attributes of a dimension may



A concept hierarchy for *location*. Due to space limitations, not all of the hierarchy nodes are shown, indicated by ellipses between nodes.