Reg. No.										

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL

A Constituent unit of MAHE, Manipal

VII SEMESTER B.TECH. (COMPUTER SCIENCE & ENGINEERING)

END SEMESTER EXAMINATIONS, MAY 2021

SUBJECT: DATA WAREHOUSE AND DATA MINING [CSE 4060]

REVISED CREDIT SYSTEM
(--/05/2022)

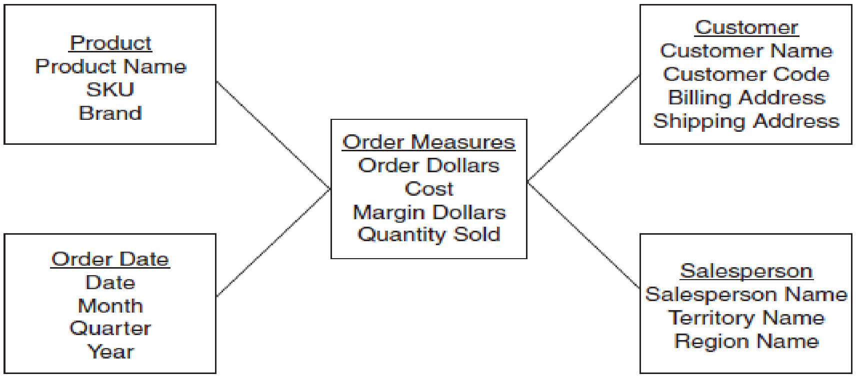
Time: 3 Hours MAX. MARKS: 50

Instructions to Candidates:

- Answer **ALL FIVE** questions.
- Missing data may be suitably assumed.

			CO	BL
1A.	Describe how User Interaction is an issue in Data Mining.	**1**	**1**	**1**
	1. ***Interactive mining:*** The data mining process should be highly *interactive*. Thus, it is important to build flexible user interfaces and an exploratory mining environment, facilitating the user's interaction with the system. A user may like to first sample a set of data, explore general characteristics of the data, and estimate potential mining results. Interactive mining should allow users to dynamically change the focus of a search, to refine mining requests based on returned results, and to drill, dice, and pivot through the data and knowledge space interactively, dynamically exploring "cube space" while mining. 2. ***Incorporation of background knowledge:*** Background knowledge, constraints, rules, and other information regarding the domain under study should be incorporated into the knowledge discovery process. Such knowledge can be used for pattern evaluation as well as to guide the search toward interesting patterns. 3. ***Ad hoc data mining and data mining query languages:*** Query languages (e.g., SQL) have played an important role in flexible searching because they allow users to pose ad hoc queries. Similarly, high-level data mining query languages or other high-level flexible user interfaces will give users the freedom to define ad hoc data mining tasks. This should facilitate specification of the relevant sets of data for analysis, the domain knowledge, the kinds of knowledge to be mined, and the conditions and constraints to be enforced on the discovered patterns. Optimization of the processing of such flexible mining requests is another promising area of study. 4. ***Presentation and visualization of data mining results:*** How can a data mining system present data mining results, vividly and flexibly, so that the discovered knowledge can be easily understood and directly usable by humans? This is especially crucial if the data mining process is interactive. It requires the system to adopt expressive knowledge representations, user-friendly interfaces, and visualization techniques. **(each point carries 1M)**			
1B.	Explain any four architectural types of Data Warehouses.	**4**	**1**	**2**

	Centralized Data Warehouse This architectural type takes into account the enterprise-level information requirements. An overall infrastructure is established. Atomic level normalized data at the lowest level of granularity is stored in the third normal form. **Independent Data Marts** This architectural type evolves in companies where the organizational units develop their own data marts for their own specific purposes The data marts are independent of one another. As a result, these different data marts are likely to have inconsistent data definitions and standards. Such variances hinder analysis of data across data marts **Federated** Some companies get into data warehousing with an existing legacy of an assortment of decision-support structures in the form of operational systems, extracted datasets, primitive data marts, and so on. For such companies, it may not be prudent to discard all that huge investment and start from scratch. The practical solution is a federated architectural type where data may be physically or logically integrated through shared key fields, overall global metadata, distributed queries, and such other methods. **Hub-and-Spoke** Similar to the centralized data warehouse architecture, here too is an overall enterprise-wide data warehouse. Atomic data in the third normal form is stored in the centralized data warehouse. The major and useful difference is the presence of dependent data marts in this architectural type. Dependent data marts obtain data from the centralized data warehouse. The centralized data warehouse forms the hub to feed data to the data marts on the spokes. **Data-Mart Bus** This is the Kimbal conformed supermarts approach. You begin with analyzing requirements for a specific business subject such as orders, shipments, billings, insurance claims, car rentals, and so on. You build the first data mart (supermart) using business dimensions and metrics. These business dimensions will be shared in the future data marts. The principal notion is that by conforming dimensions among the various data marts, the result would be logically integrated supermarts that will provide an enterprise view of the data. (any 4 may be explained. Each point carries 1M)			
1C.	With the help of a diagram explain the Star Schema Data Model	**2**	**2**	**2**
	It consists of the orders fact table shown in the middle of the schema diagram. Surrounding the fact table are the four dimension tables of customer, salesperson, order date, and product. The users in this department will analyze the orders using dollar amounts, cost, profit margin, and sold quantity. This information is found in the fact table of the structure. The users will analyze these measurements by breaking down the numbers in combinations by customer, salesperson, date, and product. All these dimensions along which the users will analyze are found in the structure. The STAR schema structure is a structure that can be easily understood by the users and with which they can comfortably work. The structure mirrors how the users normally view their critical measures along their business dimensions. 1M			



(Figure 1M)

2A.	With the help of an example describe the general principles and method of application of Type 2 changes to Data Warehouses	5	2	1

Here are the general principles for this type of change:

† They usually relate to true changes in source systems.
† There is a need to preserve history in the data warehouse.
† This type of change partitions the history in the data warehouse.
† Every change for the same attribute must be preserved.
(each point carries 1/2M. 1/2X4=2M)

Applying Type 2 Changes to the Data Warehouse

Figure 11-3 shows the application of type 2 changes to the customer dimension table. The method for applying type 2 changes is:

† Add a new dimension table row with the new value of the changed attribute.
† An effective date field may be included in the dimension table.
† There are no changes to the original row in the dimension table.
† The key of the original row is not affected.
† The new row is inserted with a new surrogate key.
(each point carries 1/2M. 1/2X5=2.5M)

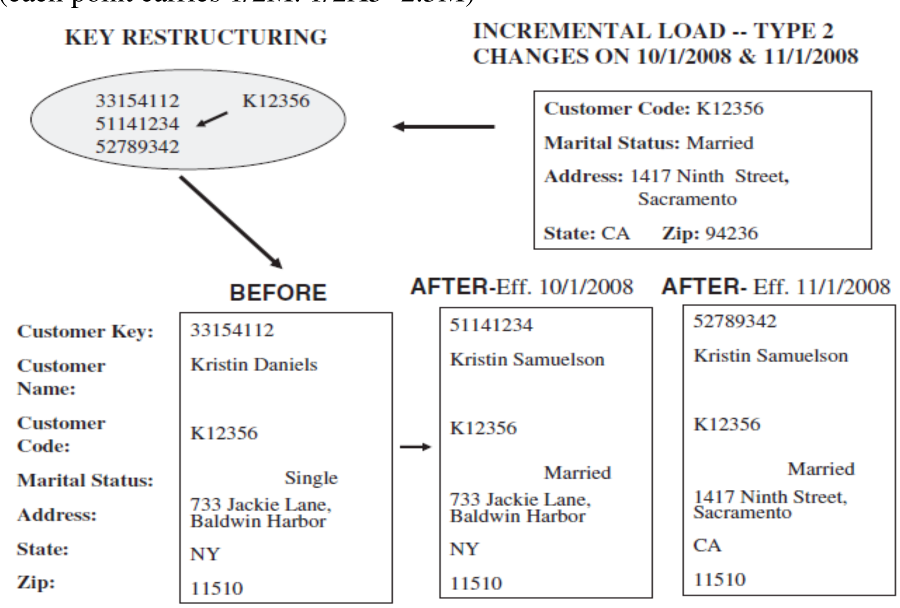
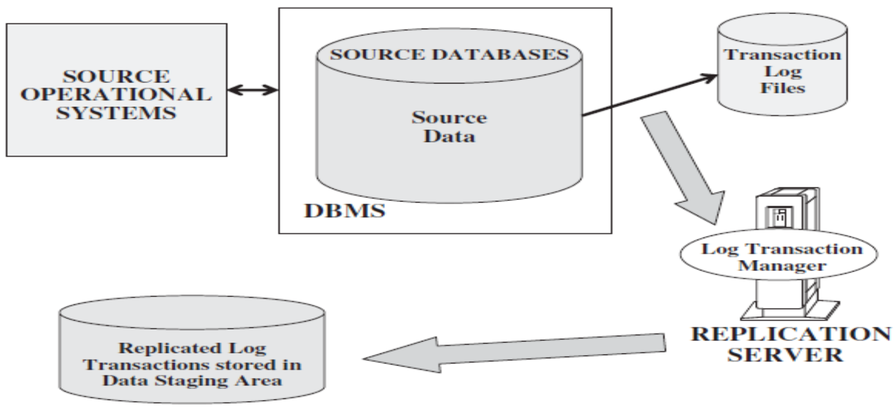


Figure 11-3 The method for applying type 2 changes.

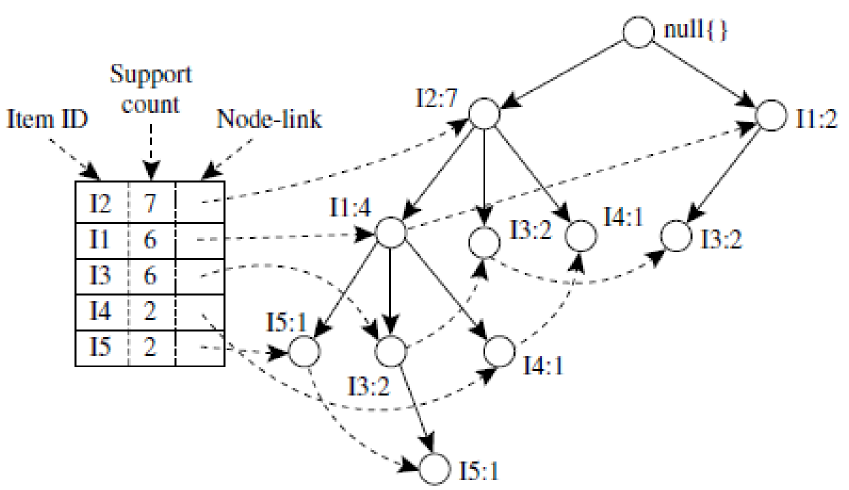
Figure 0.5M

2B.	With the help of a diagram describe the steps involved while using replication to capture changes to source data	3	2	1

	Identify the source system database table † Identify and define target files in the staging area † Create mapping between the source table and target files † Define the replication mode † Schedule the replication process † Capture the changes from the transaction logs † Transfer captured data from logs to target files † Verify transfer of data changes † Confirm success or failure of replication † In metadata, document the outcome of replication † Maintain definitions of sources, targets, and mapping  **Figure 12-5** Data extraction using replication technology. (Steps 2M. Figure 1M)			
2C.	Describe any 4 strategies for Data Transformation	**2**	**2**	**1**
	1. **Smoothing**, which works to remove noise from the data. Techniques include binning, regression, and clustering. 2. **Attribute construction** (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process. 3. **Aggregation**, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for data analysis at multiple abstraction levels. 4. **Normalization**, where the attribute data are scaled so as to fall within a smaller range, such as −1.0 to 1.0, or 0.0 to 1.0. 5. **Discretization**, where the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth, adult, senior*). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute. Figure 3.12 shows a concept hierarchy for the attribute *price*. More than one concept hierarchy can be defined for the same attribute to accommodate the needs of various users. 6. **Concept hierarchy generation for nominal data**, where attributes such as *street* can be generalized to higher-level concepts, like *city* or *country*. Many hierarchies for nominal attributes are implicit within the database schema and can be automatically defined at the schema definition level. (any 4. Each point carries 1/2M. 4x ½=2M)			
3A.	For the given dataset, assuming minimum support is set to a value of 2, find all frequent itemsets using the FP-Growth algorithm. Show the detailed steps by constructing the	**5**	**3**	**6**

Conditional (Sub-)Pattern Bases and also show the conditional FP-tree associated with the conditional node I3 using pictorial representation.

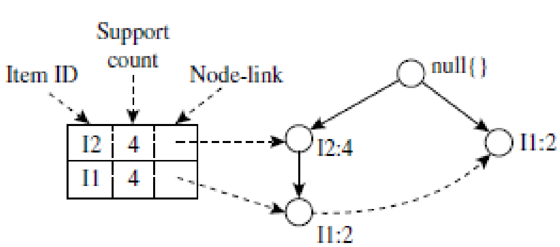
TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



An FP-tree registers compressed, frequent pattern information.

Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}



The conditional FP-tree associated with the conditional node I3.

(Generating Conditional Pattern Base -2M, FP-Tree-2M, pictorial representation of conditional FP tree for I3-1M)

3B.	For the dataset shown in question 3A find all frequent itemsets using Vertical Data Format	**3**	**3**	**6**

The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

(each table carries 1M. Total 3M)

3C.	With the help of an example describe any 2 pruning strategies involved in mining Closed Itemsets	**2**	**3**	**1**

Item merging: *If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y, then X ∪ Y forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y.*

For example, in Table 6.2 of Example 6.5, the projected conditional database for prefix itemset {I5:2} is {{I2, I1}, {I2, I1, I3}}, from which we can see that each of its transactions contains itemset {I2, I1} but no proper superset of {I2, I1}. Itemset {I2, I1} can be merged with {I5} to form the closed itemset, {I5, I2, I1: 2}, and we do not need to mine for closed itemsets that contain I5 but not {I2, I1}.

Sub-itemset pruning: *If a frequent itemset X is a proper subset of an already found frequent closed itemset Y and* support_count(X)=support_count(Y), *then X and all of X's descendants in the set enumeration tree cannot be frequent closed itemsets and thus can be pruned.*

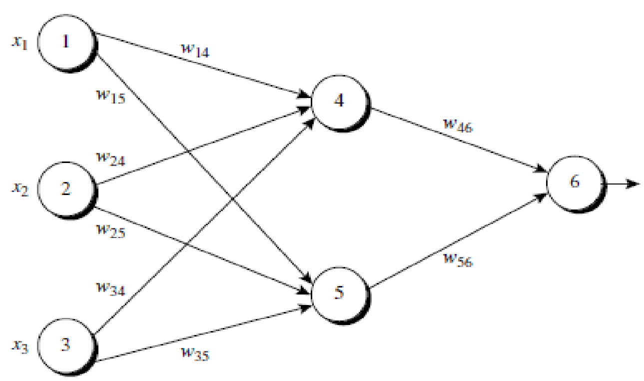
Similar to Example 6.2, suppose a transaction database has only two transactions: $\{\langle a_1, a_2, \ldots, a_{100}\rangle, \langle a_1, a_2, \ldots, a_{50}\rangle\}$, and the minimum support count is $min\_sup = 2$. The projection on the first item, a_1, derives the frequent itemset, $\{a_1, a_2, \ldots, a_{50} : 2\}$, based on the *itemset merging* optimization. Because $support(\{a_2\}) = support(\{a_1, a_2, \ldots, a_{50}\}) = 2$, and $\{a_2\}$ is a proper subset of $\{a_1, a_2, \ldots, a_{50}\}$, there is no need to examine a_2 and its projected database. Similar pruning can be done for $a_3, \ldots, a_{50}$ as well. Thus, the mining of closed frequent itemsets in this data set terminates after mining a_1's projected database.

Item skipping: *In the depth-first mining of closed itemsets, at each level, there will be a prefix itemset X associated with a header table and a projected database. If a local frequent item p has the same support in several header tables at different levels, we can safely prune p from the header tables at higher levels.*

Consider, for example, the previous transaction database having only two transactions: $\{\langle a_1, a_2, \ldots, a_{100}\rangle, \langle a_1, a_2, \ldots, a_{50}\rangle\}$, where $min\_sup = 2$. Because a_2 in a_1's projected database has the same support as a_2 in the global header table, a_2 can be pruned from the global header table. Similar pruning can be done for $a_3, \ldots, a_{50}$. There is no need to mine anything more after mining a_1's projected database.

(any 2 may be explained. Each method carries 1M. Total 2M)

4A. Consider the following figure showing a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in Table 1, Classify the tuple, ***X*** =(1, 0, 1) with a class label of 1 using Backpropagation algorithm. Show all steps in detail for the first iteration. **5** **3** **4**



Initial Input, Weight, and Bias Values

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

Net Input and Output Calculations

Unit, j	Net Input, I_j	Output, O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

Calculation of the Error at Each Node

Unit, j	Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

Calculations for Weight and Bias Updating

Weight or Bias	New Value
w_{46}	$-0.3+(0.9)(0.1311)(0.332)=-0.261$
w_{56}	$-0.2+(0.9)(0.1311)(0.525)=-0.138$
w_{14}	$0.2+(0.9)(-0.0087)(1)=0.192$
w_{15}	$-0.3+(0.9)(-0.0065)(1)=-0.306$
w_{24}	$0.4+(0.9)(-0.0087)(0)=0.4$
w_{25}	$0.1+(0.9)(-0.0065)(0)=0.1$
w_{34}	$-0.5+(0.9)(-0.0087)(1)=-0.508$
w_{35}	$0.2+(0.9)(-0.0065)(1)=0.194$
θ_6	$0.1+(0.9)(0.1311)=0.218$
θ_5	$0.2+(0.9)(-0.0065)=0.194$
θ_4	$-0.4+(0.9)(-0.0087)=-0.408$

(calculating input and output-2M, calculating error at each node 1M, calcualting weight and bias updation 2M. Total 5M)

4B.	Describe how Sampling and Dynamic Itemset Counting techniques can help in improving the efficiency of Apriori Algorithm	3	3	1

	Sampling (mining on a subset of the given data): The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. In this way, we trade off some degree of accuracy against efficiency. The S sample size is such that the search for frequent itemsets in S can be done in main memory, and so only one scan of the transactions in S is required overall. Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets. To reduce this possibility, we use a lower support threshold than minimum support to find the frequent itemsets local to S (denoted L^S). The rest of the database is then used to compute the actual frequencies of each itemset in L^S. A mechanism is used to determine whether all the global frequent itemsets are included in L^S. If L^S actually contains all the frequent itemsets in D, then only one scan of D is required. Otherwise, a second pass can be done to find the frequent itemsets that were missed in the first pass. The sampling approach is especially beneficial when efficiency is of utmost importance such as in computationally intensive applications that must be run frequently. **Dynamic itemset counting** (adding candidate itemsets at different points during a scan): A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan. The technique uses the count-so-far as the lower bound of the actual count. If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collection and can be used to generate longer candidates. This leads to fewer database scans than with Apriori for finding all the frequent itemsets. (sampling 2M, Dynamic Itemset Counting 1M, Total 3M)			
4C.	Write an algorithm for classifying tuples using Backpropagation algorithm	**2**	**4**	**4**

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.

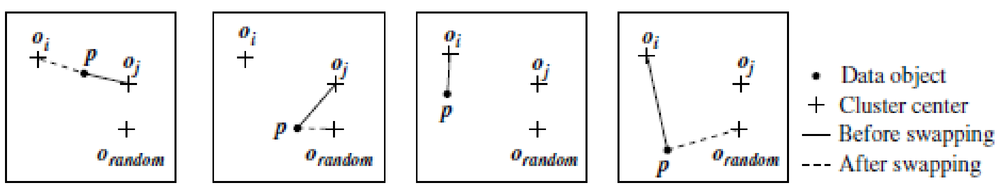
Output: A trained neural network.

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 {
            O_j = I_j; // output of an input unit is its actual input value
        for each hidden or output layer unit j {
            I_j = Σ_i w_ij O_i + θ_j; //compute the net input of unit j with respect to
                the previous layer, i
            O_j = 1/(1+e^(-I_j)); } // compute the output of each unit j
        // Backpropagate the errors:
        for each unit j in the output layer
            Err_j = O_j(1 − O_j)(T_j − O_j); // compute the error
        for each unit j in the hidden layers, from the last to the first hidden layer
            Err_j = O_j(1 − O_j) Σ_k Err_k w_jk; // compute the error with respect to
                the next higher layer, k
        for each weight w_ij in network {
            Δw_ij = (l)Err_j O_i; // weight increment
            w_ij = w_ij + Δw_ij; } // weight update
        for each bias θ_j in network {
            Δθ_j = (l)Err_j; // bias increment
            θ_j = θ_j + Δθ_j; } // bias update
        } }
```

Backpropagation algorithm.

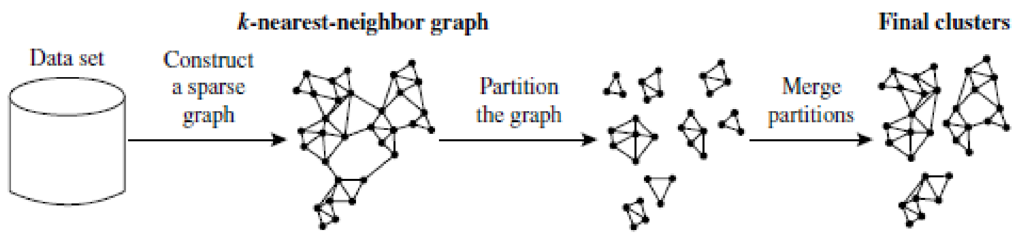
5A.	With the help of a diagram explain how clustering can be carried out using k-medoids clustering	**4**	**5**	**2**

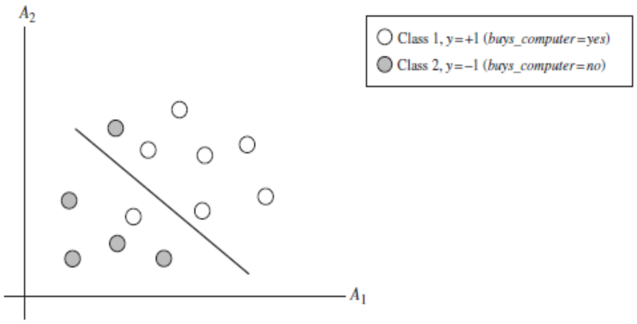


(a) Reassigned to o_i (b) Reassigned to o_{random} (c) No change (d) Reassigned to o_{random}

Four cases of the cost function for k-medoids clustering.

Specifically, let $o_1, \ldots, o_k$ be the current set of representative objects (i.e., medoids). To determine whether a nonrepresentative object, denoted by o_{random}, is a good replacement for a current medoid o_j ($1 \le j \le k$), we calculate the distance from every object p to the closest object in the set $\{o_1, \ldots, o_{j-1}, o_{random}, o_{j+1}, \ldots, o_k\}$, and use the distance to update the cost function. The reassignments of objects to $\{o_1, \ldots, o_{j-1}, o_{random}, o_{j+1}, \ldots, o_k\}$ are simple. Suppose object p is currently assigned to a cluster represented by medoid o_j (Figure 10.4a or b). Do we need to reassign p to a different cluster if o_j is being replaced by o_{random}? Object p needs to be reassigned to either o_{random} or some other cluster represented by o_i ($i \neq j$), whichever is the closest. For example, in Figure 10.4(a), p is closest to o_i and therefore is reassigned to o_i. In Figure 10.4(b), however, p is closest to o_{random} and so is reassigned to o_{random}. What if, instead, p is currently assigned to a cluster represented by some other object o_i, $i \neq j$?

	Object o remains assigned to the cluster represented by o_i as long as o is still closer to o_i than to o_{random} (Figure 10.4c). Otherwise, o is reassigned to o_{random} (Figure 10.4d). Each time a reassignment occurs, a difference in absolute error, E, is contributed to the cost function. Therefore, the cost function calculates the *difference* in absolute-error value if a current representative object is replaced by a nonrepresentative object. The total cost of swapping is the sum of costs incurred by all nonrepresentative objects. If the total cost is negative, then o_j is replaced or swapped with o_{random} because the actual absolute-error E is reduced. If the total cost is positive, the current representative object, o_j, is considered acceptable, and nothing is changed in the iteration. (explanation 3M, Figure 1M)			
5B.	With the help of a figure explain the working CHAMELEON for clustering data.	**4**	**5**	**2**
	Figure 10.10 illustrates how Chameleon works. Chameleon uses a k-nearest-neighbor graph approach to construct a sparse graph, where each vertex of the graph represents a data object, and there exists an edge between two vertices (objects) if one object is among the k-most similar objects to the other. The edges are weighted to reflect the similarity between objects. Chameleon uses a graph partitioning algorithm to partition the k-nearest-neighbor graph into a large number of relatively small subclusters such that it minimizes the **edge cut**. That is, a cluster C is partitioned into subclusters C_i and C_j so as to minimize the *weight of the edges* that would be cut should C be bisected into C_i and C_j. It assesses the *absolute* interconnectivity between clusters C_i and C_j. Chameleon then uses an agglomerative hierarchical clustering algorithm that iteratively merges subclusters based on their similarity. To determine the pairs of most similar subclusters, it takes into account both the interconnectivity and the closeness of the clusters. Specifically, Chameleon determines the similarity between each pair of clusters C_i and C_j according to their *relative interconnectivity*, $RI(C_i, C_j)$, and their *relative closeness*, $RC(C_i, C_j)$.  Chameleon: hierarchical clustering based on k-nearest neighbors and dynamic modeling. *Source:* Based on Karypis, Han, and Kumar [KHK99]. Explanation 3M, Figure 1M			
5C.	With the help of a diagram describe how Support Vector Machines classify data when data are linearly separable	**2**	**4**	**4**



A simple 2-D case showing linearly inseparable data. Unlike the linear separable data of Figure 9.7, here it is not possible to draw a straight line to separate the classes. Instead, the decision boundary is nonlinear.

A separating hyperplane can be written as

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

where $\boldsymbol{W}$ is a weight vector, namely, $\boldsymbol{W} = \{w_1, w_2, \ldots, w_n\}$; 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., $\boldsymbol{X} = (x_1, x_2)$), where x_1 and x_2 are the values of attributes A_1 and A_2, respectively, for $\boldsymbol{X}$. 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

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

$$d(X^T) = \sum_{i=1}^{l} y_i\alpha_iX_iX^T + b_0, \tag{9.19}$$

where y_i is the class label of support vector X_i; X^T is a test tuple; α_i and b_0 are numeric parameters that were determined automatically by the optimization or SVM algorithm noted before; and l is the number of support vectors.

Given a test tuple, X^T, we plug it into Eq. (9.19), and then check to see the sign of the result. This tells us on which side of the hyperplane the test tuple falls. If the sign is positive, then X^T falls on or above the MMH, and so the SVM predicts that X^T belongs to class $+1$ (representing *buys_computer* = *yes*, in our case). If the sign is negative, then X^T falls on or below the MMH and the class prediction is -1 (representing *buys_computer* = *no*).

(Figure 0.5 M. Explanation 1M)