11-03-19

* Data Reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set. Data reduction techniques can be applied to obtain reduced representation of data set i.e. much smaller in volume yet closely maintains the integrity of the reduced data i.e. mining of it on the reduced data set should be more efficient yet produce the same analytical results.

Techniques are of 3 types:-

① Dimensionality reduction :- It is the process of reducing the no. of attributes under consideration. (dimension → attribute, size → rows)

② Numerosity reduction :- These techniques replace the original data volume by alternative smaller forms of data representation.

③ Data compression techniques :- These are applied so as to obtain a reduced or compressed representation of the original data. If the original data can be reconstructed from the compressed data without any information loss the data reduction is called lossless.

If instead we can reconstruct only an approximation of the original data, the data reduction is called lossy.

Some of techniques are described below:-

From category 2, a technique is used.

→ Data Cube Aggregation :- Rollup technique used (done previously give example)

→ Concept Hierarchy :- The concept hierarchy allows the analysis of data at multiple abstraction levels

e.g.: Street → City → State → Country → Continent

Days → Weeks → Months → Quarters → Years

→ Clustering :- Also a data reduction technique. In data reduction, the cluster representation of the data are used to replace the actual data. The effectiveness of this data technique depends upon thisse data's nature.

→ Sapton Sampling :- It can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random data sample.

gmp CTM

a) ** SRS WOR - Simple random sample without replacement.

This is created by drawing 's' of N's tuples from database where the probability of drawing any tuple is 1/N

b) ** SRS WR :- Simple random sample with replacement.

This is similar to the earlier method except that each time a couple is drawn from 'D' database, it is recorded & then replaced. After a tuple is drawn, it is placed back in database 'D' so that it may be drawn again.

c) ** Stratified Sample - If D is divided into mutually disjoint parts called stratas (divided part)

A stratified sample is generated by obtaining an SRS at each stratum. (strata) (simple random sample)

e.g. A stratified sample may be obtained from customer data where a stratum is created for each customer age group.

d) Cluster sample :- If the tuples in 'D' are grouped into 'm' mutually disjoint clusters, then an SRS of 's' clusters can be obtained.

e.g. Tuples in a database are usually retrieved a page at a time so that each page can be considered as a cluster. A reduced data representation can be obtained by applying SRSWOR to the pages resulting in a cluster sample of the tuples.

→ Attribute subset Selection :- Attribute subset selection reduces the data set size by removing irrelevant or redundant attributes (dimensions). The goal of the attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of data classes is as close as possible to the original distribution obtained using all attributes.

The best (& worst) attributes are typically determined using tests of statistical significance which assumes that attributes are independent of one another.

15-03-19

Data Transformation

In DT, the data are transformed or consolidated into forms appropriate for mining.

Strategies for data transformation include the following :-

① Smoothing :- It works to remove noise from the data - The techniques include binning, regression & clustering

② Attribute Construction (Feature Construction) :- In this method, new attributes are constructed & added from the given set of attributes to help the mining process.

③ Aggregation :- In this step, summary/aggregation operations are applied to the data. This step is typically used in constructing a data queue for data analysis at multiple abstraction levels.

grp:

④ Normalisation :- In this method the attribute data are scaled so as to fall within a smaller range, such as 0 to 1 or -1 to 1.

⑤ Discretization :- In this, the raw values of a numeric attribute are replaced by interval labels or conceptual labels.

e.g. Make continuous value to discrete

Label (Name)	20, 30, 25, 21, ..	Label → Conceptual name
Youth	20 - 30	frequency
Middleage	31 - 40	

(6) Concept Hierarchy generation for nominal data :-

In this method the attributes such as street can be generalized to higher level concepts like city / country.

e.g. Country ↑ State ↑ City ↑ Street

Map (Google) Scroll Down

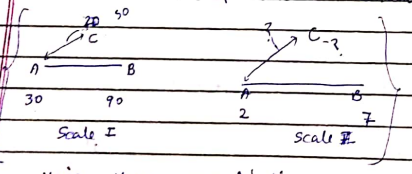
Concept Hierarchy ↑

Concept Hierarchy

→ Data Transformation by Normalisation

The measurement unit used can affect the data analysis e.g. Changing measurement units from metres to inches or from kg to pounds may lead to very different results. To help avoid dependance on the choice of measurement units, the data should be normalised or standardised. This involves transforming the data to fall within a smaller or common range such as -1 to 1 or 0 to 1.

There are 3 prominent methods for data normalisation



M-I Min Max normalisation

$$v_i' = \frac{v_i - \text{Min}_A}{\text{Max}_A - \text{Min}_A} (\text{newMax}_A - \text{NewMin}_A) + \text{NewMin}_A$$

This method performs a linear transformation on the original data. Suppose Min_A & Max_A are the minimum & maximum values of an attribute A. MinMax Normalisation maps a value v_i to v_i' in the range [NewMin_A, NewMax_A] by using the above mentioned formula.

$\text{Min}_A = 30$
$\text{Max}_A = 90$
$\text{newMin}_A = 2$
$\text{newMax}_A = 7$

$$v_i' = \frac{50-30}{90-30}(7-2)+2 = 3.6$$

$$\frac{20}{60}(5)+2$$

⇒ A is 1.6 away from C

M-II Z-Score Normalisation

In z-score normalisation, the values of an attribute A are normalised based on the mean & standard deviation of A.

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}$$

This method of transformation is used when actual min & max are unknown.

M-III Normalisation by decimal scaling

This method normalises by moving the decimal pt. of values of attribute A. The no. of decimal pt. moved depends on the maximum absolute value of A.

The formula is given below:-

$$v_i' = \frac{v_i}{10^j}$$

j is smallest integer such that $\text{Max}(|v_i'|) < 1$

e.g. Min value = -0.985
Max value = .975

If j=3 → $\frac{-.985}{3}$ * $\frac{.975}{3}$

If j=2 ✗ Not valid

If j=4 ⇒ Valid ✓ Max value of j

18.03.19

Chapter: Mining Frequent Patterns & Associations

1. Frequent Patterns → These are patterns that appear frequently in a data set. e.g.: Frequent item sets or frequent subsequences fall in category of Frequent patterns.

2. Frequent item Set → A set of items that appear frequently together in a transaction data set is called a frequent itemset. e.g. Milk & Bread. {Milk, Bread, eggs}

3. Frequent Subsequence Pattern → A subsequence such as buying first a PC, then a digital camera & then a memory card. If it occurs frequently in a shopping history database is a frequent sequential pattern.

Frequent item set mining leads to the discovery of associations & correlations among items in large transactional or relational data sets. With massive amounts of data continuously being collected & stored, many industries are becoming interested in mining such patterns from their databases. The discovery of interesting patterns can help in many business decision making processes such as catalog design, cross marketing & customer shopping behaviour analysis.

Antecedent

- Association Rule :- Head {A, B} ⇒ {C, D} Body consequent
  e.g.: [support = 2 %, confidence = 60 %]

It defines how strongly one itemset (or one set of items) is associated with / to other set of items.

{computer, antivirus} ⇒ {Hard disk, printer}
positive

It is represented with an arrow symbol
Left set of items: Head / Antecedent
Right : Body / Consequent

Antecedent ~~Antedent~~ is associated with consequent

Transactions dataset

1.	A	B	C	D	
2.	A	C	D		
3.	A	B	E		
4.	A	B	C	D	E
5.	C	D	B		
6.	B	D			
7.	A	B	E		

- Support 2 % → It means that 2% of all the transactions under analysis shows that ~~computer & ABCD~~ ~~antivirus~~ are purchased together.

ABCD occurs together in (2) transaction ABCD with support is 28.57%

$\Rightarrow \frac{2}{7} \times 100 = 28.57\%$ (7 = total transaction) support count = (2)

In 28.57% of all transactions, ABCD appear together.

- Confidence 60%: It means that 60% of the customers who purchase a computer & antivirus also bought harddisk & printer. (A) (B) (C) (D)

Those persons who bought A, B ⇒ 60% of those persons also bought C, D.

A, B appearing in 4 transactions. ⇒ $\frac{2}{4} \times 100 = 50\%$.

C, D ——— 2 ———

⇒ confidence is 50%

(Imp: I/P of support & confidence is given by user)

Association rules are considered interesting if they satisfy both a minimum support threshhold & a min. confidence threshhold.

$A \Rightarrow B$ A, B : itemset

Confidence of association rule = $\frac{\text{Support count } (A \cup B)}{\text{Support count } (A)}$

In general, association rule mining can be viewed as a 2 step process.

Step 1. Find all frequent itemsets
Each of these itemsets will occur atleast as frequently as predetermined minimum support count (Min_Sup)

Step 2. Generate strong association rules from the frequent itemset. These rules must satisfy minimum support & minimum confidence.

→ K-itemset : A set of items is referred to as an itemset. An itemset that contains K items is an K-itemset.

e.g. 2-itemset {harddisk, printer}

28-03-19

Apriori Algorithm

Apriori algo. was proposed by Aggarwal & Shreekanth. in 1994 for mining frequent item-set. The name of the algo. is based on the fact that the algo. uses prior knowledge of frequent item set properties.
The algorithm uses an iterative approach k/a a level wise search where K-item sets are used to explore K+1 item sets.

Apriori starts with item sets of length 1 also k/a 1 item sets as candidiates & determines their support by making a scan of the dataset 'D'. Candidates that are infrequent are discarded while the frequent ones are given as outputs. The frequent itemsets are then used to form candidate itemsets of length 2. This process repeats for longer itemsets until there are no more candidates count. The pseudo-code of this algo. is written below:- (Program + English)

```
Apriori (D, min sup):        (support)
C = { all 1_itemsets)
while (|C| > 0)
Scan D to find counts of C
F = Sets of in C with count ≥ minsup
output (F)
```

5 C = Apriori Gen (F)

→ Apriori Gen (F) :

6 for each pair of elements X, Y in F

7 if X and Y share all items except last

Z = X ∪ Y

if any immediate subset of Z not in F

Prune Z

Prune → remove

Transaction Id	List of items
T_1	I_1, I_2, I_5
T_2	I_2, I_4
T_3	I_2, I_3
T_4	I_1, I_2, I_4
T_5	I_1, I_3
T_6	I_2, I_3
T_7	I_1, I_3
T_8	I_1, I_2, I_3, I_5
T_9	I_1, I_2, I_3

e.g

⇒ database

Support count / Frequency of items in D

Let minsup. = 2 (I/P by user)

Here items = I_1, I_2, I_3, I_4, I_5

Step1. Make itemsets & put in C

C

$\{I_1\}$

$\{I_2\}$

$\{I_3\}$

$\{I_4\}$

$\{I_5\}$

Step2. $|C| > 0$ means there are no. of items in C

Step3.

C	Support Count
$\{I_1\}$	6
$\{I_2\}$	7
$\{I_3\}$	6
$\{I_4\}$	2
$\{I_5\}$	2

C → set of candidate / candidate item sets

Step4. Here Min Support = 2 (given)

⇒ Put in F i.e. List of Frequent 1 itemsets

F	Support Count
$\{I_1\}$	6
$\{I_2\}$	7
$\{I_3\}$	6
$\{I_4\}$	2
$\{I_5\}$	2

And Part a)

Step5 C = Apriori Gen (F)

pass F to C as arguments

Step6 Make Pairs

(In this step we convert K item sets → K+1 itemsets & C → candidates)

C	Count	
$\{I_1, I_2\}$	4	✓
$\{I_1, I_3\}$	4	✓
$\{I_1, I_4\}$	1	✗
$\{I_1, I_5\}$	2	✓
$\{I_2, I_3\}$	4	✓
$\{I_2, I_4\}$	2	✓
$\{I_2, I_5\}$	2	✓
$\{I_3, I_4\}$	0	✗
$\{I_3, I_5\}$	1	✗
$\{I_4, I_5\}$	0	✗

F

Frequent item sets as Min Sup. = 2

Here not applicable

** Apriori Property :- This property says that all known comply subsets of a frequent itemset must all be frequent.

Ans Part b)

F	Count
I_1, I_2	4
I_1, I_3	4
I_1, I_4	1
...	...
I_2, I_5	2

Here I_2, I_3 are last & except last other are same ⇒ combine

Step 3: $Z = X \cup Y$

C

$\{I_1, I_2\}$ ⇒ I_1, I_2, I_3
$\{I_1, I_3\}$

I_1, I_2, I_5
$\{I_1, I_2\}$
$\{I_1, I_5\}$

I_1, I_3, I_5
$\{I_1, I_3\}$
$\{I_1, I_5\}$

$\{I_2, I_3\}$ ⇒ I_2, I_3, I_4
$\{I_2, I_4\}$

$\{I_2, I_3\}$ ⇒ I_2, I_3, I_5
$\{I_2, I_5\}$

$\{I_2, I_4\}$
$\{I_2, I_5\}$ ⇒ $\{I_2, I_4, I_5\}$

Before counting
Check
Apriori property

⇒ Check for C

I_1, I_2, I_3

Possible subsets: I_1, I_2 I_1, I_3 I_2, I_3

All are frequent i.e. count ≥ Min sup. similarly for all

Step :-

C	Count
I_1, I_2, I_3	2 ✓
I_1, I_2, I_5	2 ✓

Step :-

F
I_1, I_2, I_3
I_1, I_2, I_5

Step: Combine
I_1, I_2, I_3, I_5

Step: Check Apriori property
Many subsets that are not valid

C	
I_1, I_2, I_3, I_5	✗ Not valid

⇒ candidate list becomes empty.
⇒ Done with loop

19-03-19

Generating Association rules from frequent itemsets

Once the freq. item sets have been found, it is straight forward to generate strong association rules from them. A strong association rule satisfies both minimum support & minimum confidence.
This can be done using following equations :-

$$\text{Confidence}(A \Rightarrow B) = \frac{\text{Support\_count}(A \cup B)}{\text{Support\_count}(A)}$$

A & B are itemsets
Here Support count $(A \cup B)$ is the no. of transactions containing the itemsets $A \cup B$ & Support count (A) is the no. of transactions containing the itemset A.
Based on the above equation, the association rules can be generated as follows :-

Step 1. For each freq. itemset l, generate all non-empty subsets of l (l_{non})

Step 2. For all non-empty subset s of l, output the rule

$s \Rightarrow l - s$

associated with

Rule → if $\frac{\text{Support count } (l)}{\text{support count } (s)} \geq \text{min conf}$ (confidence)

where "min conf" is the minimum confidence threshold.

e.g. $\{I_1, I_2, I_5\} \rightarrow l$

Step1: Non-empty subsets are:
- $\{I_1\}$
- $\{I_2\}$
- $\{I_5\}$
- $\{I_1, I_2\}$
- $\{I_2, I_5\}$
- $\{I_1, I_5\}$

Step2: $l \rightarrow$ freq. item set

$s \Rightarrow l - s$

a) $I_1 \Rightarrow \{I_1 I_2 I_5\} - \{I_1\} = I_2, I_5$

$\therefore I_1 \Rightarrow \{I_2, I_5\}$

$\frac{2}{6} \times 100 = 33\%$

(6 from previous pages example)

Let Min conf. = 50%.

Here Min conf. is 33% $\Rightarrow$ Rule not satisfied

$\therefore$ cancel this value.

b) Consider $\{I_2, I_5\} \rightarrow = s$

$\{I_2, I_5\} \Rightarrow \{I_1\}$

$\frac{2}{2} \times 100\% = 100\% \geq$ Min conf (50%) ✓ $\Rightarrow$ Strong association rule

↳ Given as O/P

It is CPU-Killer algorithm for all itemsets if they are large in no.

FP Growth Algorithm → generate frequent itemsets (Frequent pattern)

The Apriori algo. deals with 2 problems:
a) It may lead to generate a huge no. of candidate sets
b) It may need to repeatedly scan the whole database to check a large set of candidates by pattern matching.

→ The FP growth algorithm adopts a divide & conquer strategy. This algorithm creates an FP tree that contains the itemset association information.

The first scan of database is same as apriori which derives the set of frequent itemsets (1-itemsets) & their support counts. Then FP tree is then constructed as follows:

The set of frequent items is sorted in the order of descending support count.

Step1.

	count
I_2	7
I_1	6
I_3	6
I_4	2
I_5	2

(descending order) Table1

T_1	$I_2\ I_1\ I_5$
T_2	$I_2\ I_4$
T_3	$I_2\ I_3$
T_4	$I_2\ I_1\ I_4$
T_5	$I_1\ I_3$
T_6	$I_2\ I_3$
T_7	$I_1\ I_3$
T_8	$I_2\ I_1\ I_3\ I_5$
T_9	$I_2\ I_1\ I_3$

(each transaction in descending order of Table1)

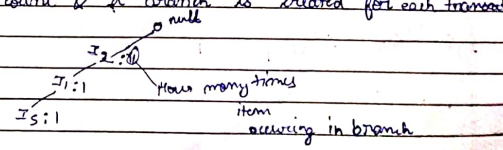
Step2. Create FP tree

First create the root of tree labelled with null

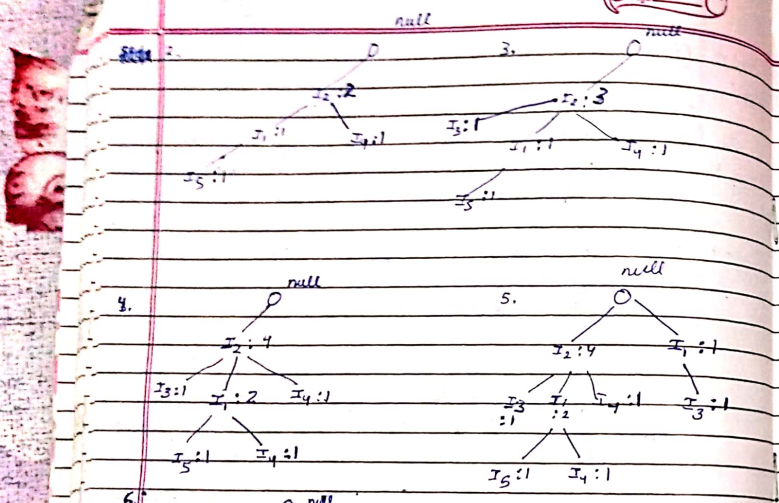
Scan database 'D' a second time

The items in each transaction are sorted acc. to descending support count & a branch is created for each transaction

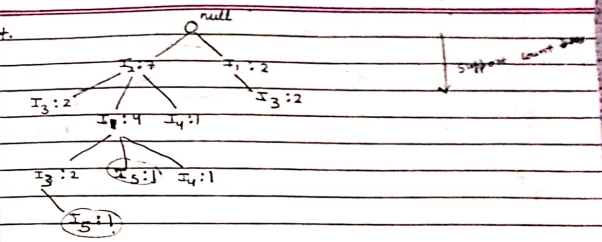
1.



FP tree



30-09-19



★ (Advantage) Search time is reduced ⇒ Preferred everytime.

Avg. length of branch is $\log n$

e.g. $\log_2 1000 \approx 10$ (avg. 1000 nodes ≤ 1024, $\log_2 1024 = 10$)

$\log_{10} 1000 = \log_{10}(10)^3 = 3 \log_{10} 10 = 3$

$\log_a b$

Generating frequent patterns from FP tree

An FP tree is mined as follows

- start from each frequent length 1 pattern as an initial suffix pattern, construct its conditional pattern base which consists of the set of prefix paths in the FP tree co-occurring with the suffix patterns. Then construct its conditional FP tree & perform mining recursively on the tree. The pattern growth is achieved by the concatination of a suffix pattern with the frequent patterns generated from conditional FP tree.

We first consider I_5 which is the last item in the initial list rather than the first item. The item I_5 occurs in 2 FP tree branches. The occurrences can be found by following chain of node links.

Write item having least support count

Item	Conditional Pattern Base	Conditional FP tree	Frequent Pattern Generated
I_5	$\{I_2\ I_1 : 1\}$ $\{I_2\ I_1\ I_3 : 1\}$	$\langle I_2 : 2, I_1 : 2 \rangle$	$\langle I_2\ I_5 : 2 \rangle$ $\langle I_1\ I_5 : 2 \rangle$ $\langle I_2\ I_1\ I_5 : 2 \rangle$
I_4	$\{I_2\ I_1 : 1\}$ $\{I_2 : 1\}$	$\langle I_2 : 2 \rangle$	$\langle I_2\ I_4 : 2 \rangle$

I_5 acts as suffix

Considering I_5 as a suffix, its corresponding two prefix paths are :-

$I_2\ I_1 : 1$

$I_2\ I_1\ I_3 : 1$ — occurring 1 time in tree

For conditional FP tree, I_2 has support count 2 in conditional Pattern Base

I_1 also

Reject I_3 as Min Sup. Count should be 2 (as considered in prev. example)

I_3	$\{I_2 : 2\}$ $\{I_1 : 2\}$ $\{I_2\ I_1 : 2\}$	$\langle I_2 : 4, I_1 : 4 \rangle$	$\langle I_2\ I_3 : 4 \rangle$ $\langle I_1\ I_3 : 4 \rangle$ $\langle I_1\ I_2\ I_3 : 4 \rangle$
I_1	$\{I_2 : 4\}$	$\langle I_2 : 4 \rangle$	$\langle I_2\ I_1 : 4 \rangle$

Don't consider I_2 (least item)

Table I from bottom up approach

The FP growth method transforms the problem of finding long freq. patterns into searching for shorter ones in much smaller conditional databases recursively.

A study of the FP growth method performance shows that it is efficient & scalable for mining both long & short frequent patterns and it is about an order of magnitude faster than the apriori algorithm.

01-04-19 Chapter - Classification

- Classification is a form of data analysis that extracts models describing important data classes. Such models called classifiers predict categorical class labels (discrete & unordered). e.g.:- We can build a classification model to categorise bank loan applications as either safe or risky.

nominal

finite

Many classification methods have been proposed by this method. In Machine learning, pattern recognition & statistics.

Classification has numerous applications including fraud detection, performance prediction, manufacturing & medical diagnosis. Classification is a 2 step process consisting of :-

① Learning step — a classification model is constructed

② Classification Step — model is used to predict class labels for given data

① → In the first step, a classifier is build describing a pre determined set of data classes or concepts.

② → This is the learning step or training phase where a classification algorithm builds the classifier by analysing or learning from a training set made up of database tuples

& their associated class labels. A tuple X is represented by an 'n' dimensional attribute vector X

$$X = (x_1, x_2, \ldots, x_n)$$

Each tuple is assumed to belong to a predefined class as determined by another database attribute called class label Attribute.
It is discrete & unordered.

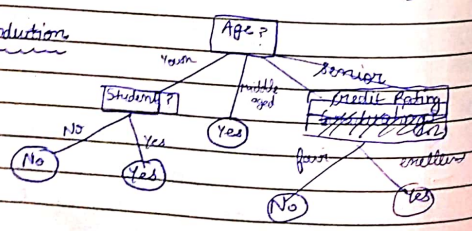
Name	Age	Income	Loan decision	
ABC	25	50,000	Safe	Training set
DEF	30	20,000	Risky	
GHI	32	25,000	Risky	
JKL	22	15000	Risky	
MNO	28	35,000	?	

Three tuples whose class labels are unknown fall in the category of Testing Set.

In learning step { model (formula) is created / Algorithm of classification is generated i.e. used in next step }

(The attribute to be predicted, is class label Attribute)

Decision Tree Induction



Age: continuous → convert → discrete
man 60 → 0-10 child, 10-20 Teenager, ..., 50-60 elder

- Decision tree induction is learning of decision trees from class label training tuple.
- Decision tree is a flowlike decision structure where each internal node denotes a test on an attribute. Each branch represents an outcome of the test & each leaf node holds a class label.

The topmost node in the tree is root node. A typical decision tree is shown above. It represents the concept buys computer. It predicts whether a customer is likely to purchase a computer or not. Internal nodes are denoted by rectangles & leaf nodes are denoted by ovals. Given the tuple for which class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to the leaf node which holds the class prediction for that tuple.

Adv.
- The construction of decision trees doesn't require any domain knowledge.
- The learning & classification of decision tree induction are simple & fast.
- Decision tree classifiers have good accuracy.
- Decision tree induction algos have been used for classification in many application areas such as medicine, financial analysis, astronomy, manufacturing & production.

03-04-19

K- Nearest Neighbour Algorithm.

Classification → predicting value of an attribute.

A K-nearest neighbour method is a simple technique which considers a training data set itself as the classification model. In this approach, the records of the training data set are considered as points in a d-dimensional space (d- attributes). Several possible distance functions can be used to find the similarity b/w the 2 objects.

Given a new point, 'p' to classify the k-nearest points of p in the training data set are found & class i.e. assigned to most of these k-points is selected as the a class for 'p'.

Although simple, this approach could be very slow during classification because it needs to search for the k-nearest points among the entire training data set which could be huge.

Student ID (S.ID)	Height	Weight	Class (to be predicted)
1	140 cm	50 Kg	7
2	147 "	60 "	7
3	148 "	48 "	6
4	150 "	60 "	8
5	152 "	55 "	7
6	155 "	65 "	8
7	153 "	62 "	?

This is numerical attribute data ⇒ We use Euclidean dist.

For object 2,3 $= \sqrt{(142-140)^2 + (60-48)^2}$ (Euclidean dist.)

k should be odd to predict majority.

→ Find distance b/w object 7 with all others.

1. $(1,7) = \sqrt{(140-153)^2 + (50-62)^2} = \sqrt{169+144} = \sqrt{313} = 17.6$
2. $(2,7) = \sqrt{(147-153)^2 + (60-62)^2} =$ 6.324
3. $(3,7) = \sqrt{(148-153)^2 + (48-62)^2} =$ 14.86
4. $(4,7) = \sqrt{(150-153)^2 + (60-62)^2} = \sqrt{3^2+2^2} = \sqrt{13}$ 3.605
5. $(5,7) = \sqrt{(152-153)^2 + (55-62)^2} = \sqrt{1^2+7^2} = \sqrt{48} = 6.928$
6. $(6,7) = \sqrt{(155-153)^2 + (65-62)^2} = \sqrt{(2)^2+(3)^2} = \sqrt{4+9} = \sqrt{13} = 3.605$

(4, 5, 6) in ascending order having least dist. value.

class: 8 7 8 ⇒ Predict value = 8

Perform with k other values if not majority

values: 3.6, 10.04, 3.6

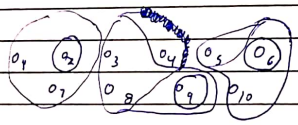
06-04-19 Partitioning Based algorithm

03

K-Means Algorithm

Input = K, no. of cluster, D dataset
output : A set of K clusters

1) arbitrarily choose K objects from D as initial clusters
2) Repeat
3) Reassign each object to the cluster to which the object is the most similar based on the mean value of the objects in the cluster.
4) Update cluster means; that is calculate the mean value of the objects for each cluster until no change.

e.g. 

→ objects

Let K = 3

- check similarity of clusterheads with other objects
- Update cluster head → $O_1 + O_2 + O_7 = H_1$

	Age	Salary
e.g.:-	20	20000
	30	30000
Mean	25	25000

Similarly H_2, H_3

- check again similarity

until Stable clusters

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)$$

(capital) (small)

Here E is the sum of the squared error for all objects in the data set.
P → point in space representing a given object.
c_i → it is the centroid of the cluster

This objective fn. tries to make the resulting K-clusters as compact & as seperate as possible.

(This value should be min. for cluster stability)

12-04-19

Theoretical Methods

The cluster which are very similar to each other are grouped into larger clusters. These larger clusters may further be grouped into still larger clusters. In this way, a hierarchy or tree of clusters is produced.

Hierarchical approach types

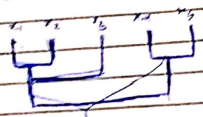
(1) Agglomerative approach

1. N = ∅
2. for each point p in D
3. Initiate cluster $C_i = \{P\}$
4. N = N ∪ C;
5. Repeat :
6. (c_i, c_j) = Closest pair of unmerged cluster
7. c_x = Merge (c_i, c_j)
8. N = N ∪ c_x
9. until No unmerged cluster exist in N

63

Agglomerative approach (cluster algorithms) - starts with all the data objects in separate clusters & then merge nearby clusters until a single cluster is formed that contains all the objects.

The algo. first initialises each data object to be in its own cluster then in each iteration, it merges the closest pair of clusters.



Dendrogram

② Divisive Approach

1) C = Cluster containing all points in D.
2) N = {C}
3) Repeat
4) C_i = leaf node in N with largest diameter
5) (C_j, C_k) = Split (C_i)
6) Insert (C_j, C_k) in N as children of C_i
7) Until all leaf nodes in N contain only a single point.

Split (C_i)

1) P = point in C_i that is the most distant from other points in C_i
2) C_j = set of points in C_i that are closer to P than C_i
3) $C_k = C_i - C_j$
4) return (C_j, C_k)

Divisive hierarchical clustering is the reverse of agglomerative algo. It starts with all the data objects in a single cluster & then divide the cluster into sub-clusters successively until each object is in its own cluster.

The divisive algorithm starts with a cluster containing all the data objects (points) in the input database, then in each iteration it splits the cluster with the largest diameter.

Diameter of cluster - It is the distance between its most distant points. The selected cluster is then split into 2 parts using a split function.

The split function finds a pt. P in the cluster which is furthest from all other points in the cluster. This distance is then calculated as the avg. distance of each point to all the other points in the cluster. The points in the cluster are then split into 2 groups based on whether they are closest to the most distant point 'P' or to the old cluster.

15-04-19

03

Balanced Iterative Reducing & Clustering using Hierarchies

This is the first hierarchical method i.e. designed to be scalable for very large datasets. It treats in 2 phases:-

1. During the first phase it forms a hierarchical clustering. This clustering is K/a clustering feature tree (CFT).
2. In the second phase, the leaf level clustering is obtained at the end of the first phase are reclustered ~~once tree clustered~~ using any other standard clustering strategy.

For each cluster in the CFT with during the first phase the algorithm maintains a vector of summary information K/a Clustering Feature.

If $x_1, x_2, \ldots, x_n$ are points in the cluster which themselves are vectors of attribute values. Its clustering feature is given as

$$\sum n, \sum x_i, \sum x_i^2$$

The clustering feature gives very useful information about clustering. It is possible to calculate size, mean & variance within a cluster.

The formula to calculate diameter of cluster is:

$$\sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i' - x_j)^2}$$

While forming the clustering feature tree, if the diameter of any leaf level cluster exceeds a user given threshold t', the cluster is split into 2. This is done in manner similar to splitting of a node in B+ trees.

In B+ tree would involve creating or updating the interior nodes that are ancestors of the leaf been split. The pseudo code of the algorithm is written as:-

1. N = { } // N is initial CF tree.
2. For each point p in D
3. m = leaf node in N i.e. closest to p.
4. Add p to m
5. Compute diameter M of m.
6. If m > t → threshhold (Overisize)
7. Split m
8. Apply another clustering algorithm to cluster the leaves of N'.