DATA MINING

[MODULE 3]

ASSOCIATION

* Frequent Pattern Analysis - Frequent Pattern is a set of items, subsequences, substructures, etc that occurs frequently in a data set. It is done to find inherent regularities in data. It is used in Basket data analysis, cross-marketing, catalog-design, sale-campaign analysis, and DNA sequence analysis.

• Association Rules -

$X \longrightarrow Y$; X item's occurrence is strongly related with Y.

if support & confidence is more than min. threshold support & confidence.

→ Support : probability of $X \cup Y$ i.e. $S = \frac{X \cup Y}{N}$

→ Confidence : conditional probability that a transaction having X also has Y ie. $C = \frac{X \cup Y}{X}$

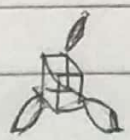
(13-2)

An item X is frequent in dataset if its support is greater than min support threshold.

↳ Closed Frequent Itemset - it is one for which none of its immediate supersets have the same support count as itself.

↳ Maximal Frequent Itemset - it is one for which none of its immediate supersets are frequent.

Downward closure property of frequent pattern says that any subset of a frequent itemset must be frequent.
ie. if {Beer, Diaper, Nuts} is frequent then {Beer, Diaper} or {Diaper, Nuts} is also frequent.



APRIORI –

Apriori pruning principle: if there is any itemset which is infrequent, its superset should not be generated or tested.

→ Method:

1) Scan DB (Database) once to get frequent 1-itemset
2) Generate length (k+1) candidate itemsets from length k frequent itemsets
3) Test the candidates against DB.
4) Terminate when no frequent or candidate set can be generated

→ Pseudo code/ Algorithm–

L_1 = {frequent items}
for ($k=1$; $L_k \neq \emptyset$; $k++$) do begin
  C_{k+1} = candidate generated from L_k
  for each transaction t in DB do
    increment the count of all candidates in C_{k+1} that are contained in t.
  L_{k+1} = candidates in C_{k+1} with min support
end
Return L_k

where C_k = Candidate itemset for size k.
L_k = frequent itemset of size k.

eg

Transactions	itemset
I_1	{A, B, C}
I_2	{A, C}
I_3	{A, D}
I_4	{B, E, F}

min support = 50%
min confidence = 50%

Support count = min support × no. of transactions
$= 0.5 \times 4 = 2$

C_1 =

item	support
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

$\Rightarrow$

L_1 =

item	support
{A}	3
{B}	2
{C}	2

C_2 =

items	support
{A B}	1
{B, C}	1
{A, C}	2

$\Rightarrow$

L_2 =

items	support
{A, C}	2
~~{B, C}~~	~~2~~

association rules	support	confidence	confidence %
A → C	2	2/3 = 0.67	67%
C → A	2	2/2 = 1.0	100%

∴ Rules are ⇒ A → C and C → A Ans

→ Challenges in Apriori algorithm -

+ Multiple scans of transaction database
+ Huge number of candidates
+ Tedious Workload of support counting for candidates
+ ~~Time~~ Breadth First Search.

→ General ideas for improving Apriori algo -

Partition; Dic, DHP
- Reduce passes of transactions database scans.
- Shrink no. of candidates
- Facilitate support counting of candidates.

FP Growth Approach:

alternative to apriori algo, uses depth first search and avoid explicit candidate generating. It is based on the philosophy: Grow long patterns from short ones using local frequent items only.

→ Method -

1) For each frequent item, construct its conditional pattern base, and then its conditional FP tree.
2) Repeat the process on each newly created conditional FP tree
3) Until the resulting FP tree is empty, or it contains only one path - single path will generate all the combinations of its sub-paths, each of which is a frequent pattern.

eg: min support = 3

Tid	items	Ordered
1	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
2	{a, b, c, f, l, m, o}	{f, c, a, b, m}
3	{b, f, h, j, o, w}	{f, b}
4	{b, c, k, s, p}	{c, b, p}
5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

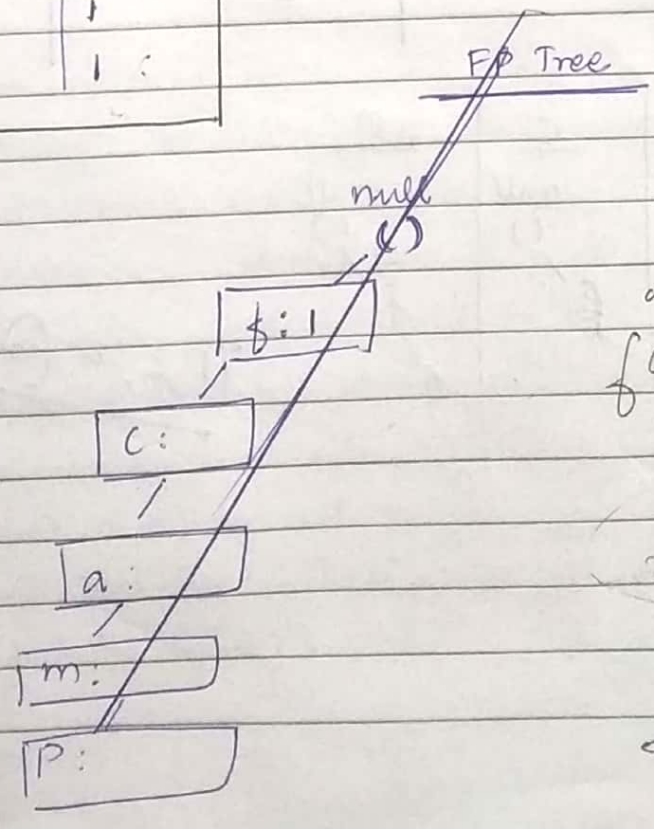
items	support
a	3
b	3
c	4
d	1
e	1
f	4
g	1
h	1
i	1
j	1
k	1
l	2
m	3
n	1
o	2
p	3
s	1
w	1

[Ordered]

items	support
f	4
c	4
a	3
b	3
m	3
p	3

f-list : f-c-a-b-m-p

FP Tree (from ordered table)

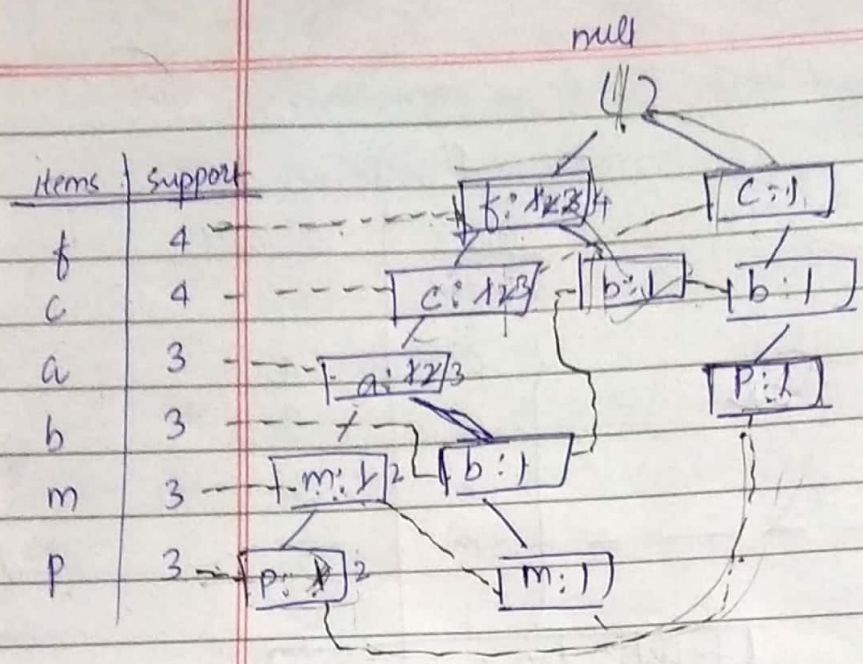


f a c f m p

f a c m p

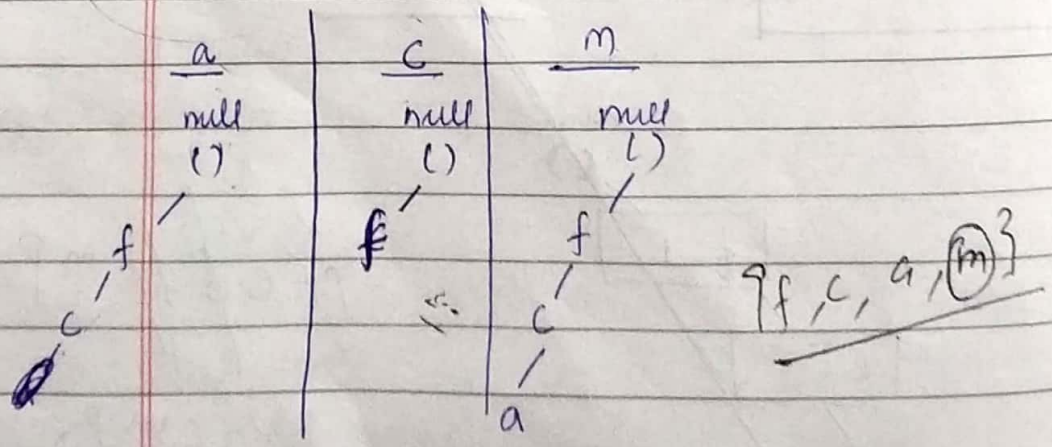
f c a m p

FP tree



Items	Support
f	4
c	4
a	3
b	3
m	3
p	3

Items	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Generator
p	(fcam:2) (cb:1)	f:2, c:2, a:2, m:2, b:1	=
m	(fca:2) (fcab:1)	f:3 c:3 a:3 b:1	(fm:3)(cm:3)(am:3) (fcm:3)(acm:3) (fam:3)
b	(fca:1)(f:1)(c:1)	f:1, c:1, a:1	ca:3
a	(fc:3)	f:3, c:3	fa:3, fca:3
c	(f:3)	f:3	cf:3
f	–	–	



Benefits of FP Tree –

1. Completeness –
   - preserve complete info for frequent pattern generation
   - Never break a long pattern of any transaction

2. Compactness –
   - Reduce irrelevant info i.e. infrequent items are gone.
   - Items in frequency descending order : the more frequently occuring, the more likely to be shared.
   - Never be larger than original database.

CLUSTERING

Clustering is finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters (a collection)

It has application in –
- Biology : defining kingdom, class, order, family.
- Marketing : distinct customer groups
- Land use : identification of earth's land use
- Climate : understand climate & find pattern.
- Earthquake study : observe epicentres.

- a good clustering will produce high quality clusters with high intra-class similarity (Cohesive within clusters) and low inter-class similarity (distinctive between clusters).

→ The quality of clustering method depends on -

- Similarity ~~method~~ measure used by method
- its implementation
- its ability to discover some or all hidden patterns.

Clustering Approaches -

1. Partitioning Approach: construct various partitions and then evaluate them by some criteria. eg: k-means.

2. Hierarchial Approach: Create a hierarchical decomposition of set of data (or objects) using some criteria eg: Diana, Agnes

3. Density-based Approach: based on density & connectivity functions. eg: ~~DBSCAN. DBSACN~~ DBSCAN

4. Grid-based Approach: based on multiple-level granularity structure. eg: STING, WaveCluster.

Partitioning Algorithms - [k-means Clustering]

Step 1: Begin with a decision on value ~~of~~ k = no. of clusters

Step 2: Find any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as follows:

1) Take the ~~first~~ k ~~trains~~ training ~~element~~ samples as single-element cluster.

2) Assign each of remaining (n-k) training sample to the

+8 BT June

Date
Page

cluster with nearest centroid. After each assignment, recompute the centroid of gaining cluster.

Step 3: Take each sample in sequence & compute distance from centroid from each cluster. If sample is not currently in the cluster with closest centroid, switch the cluster and update the centroid of that cluster.

Step 4: Repeat step 3 until convergence is achieved, ie. until a pass through the training sample causes no new assignments.

→ Advantages -

- Easy to implement
- Easy to assign new data to existing clusters.
- Concise output - co-ordinates of the K cluster centres.

→ Disadvantages -

- Dosent handle categorical variables
- Sensetive to initializations (first guess)
- Variables should all be ~~scaled~~ measured on similar scale.
- K must be known or decided a priori. wrong guess can possibly form poor results.

K=2

Ind.	x	y	d(C1)(2,3)	d(C2)(5,6)	Cluster assign
1	2	3	0	$\sqrt{18}$	C1
2	5	6	4.24	0	C2
3	8	7	7.21	3.162	C2
4	1	4	1.414	4.47	C1
5	2	2	1	5	C1
6	6	7	5.66	1.414	C2
7	3	4	1.414	2.82	C1
8	8	6	$\sqrt{45}$	$\sqrt{9}$	C2

$C_1 = \frac{1}{4}(2+1+2+3), \frac{1}{4}(3+4+2+4)$

$= (2, 3.25)$

$C_2 = \frac{1}{4}(5+8+6+8), \frac{1}{4}(6+7+7+6) = (6.75, 6.5)$

Ind.	x	y	dC1(2, 3.25)	dC2(6.75, 6.5)	Cluster
K1	2	3	$\sqrt{(0.25)^2} = 0.25$	$\sqrt{34.81}$	C1
	1	4	$\sqrt{1.56}$	$\sqrt{39.31}$	C1
	2	2	1.25	$\sqrt{42.81}$	C1
	3	4	$\sqrt{1.56}$	$\sqrt{20.31}$	C1
K2	5	6	$\sqrt{16.56}$	$\sqrt{3.3125}$	C2
	8	7	$\sqrt{50.06}$	$\sqrt{1.81}$	C2
	6	7	$\sqrt{30.06}$	$\sqrt{0.8125}$	C2
	8	6	$\sqrt{43.56}$	$\sqrt{1.8125}$	C2

$C_1' = \frac{1}{4}(2+1+2+3), \frac{1}{4}(3+4+2+4) = (2, 3.25)$

$C_2' = \frac{1}{4}(5+8+6+8), \frac{1}{4}(6+7+7+6) = (6.75, 6.5)$ Ans

Meetings | Things To Do | Important Calls

Hierarchial Clustering -

it uses distance matrix as clustering criteria. This method does not require the no. of clusters k as an input, but needs a termination condition. It decomposes data objects into several levels of nested partitioning (tree of clusters), called dendograms.

- Types:

1) Agglommerative /Agnes/ bottom-up/: HAC it start with each example as a cluster and iteratively combines them to form larger and larger clusters.

2) Divisive /Diana/ top down :- it divides one of the existing clusters into two clusters till the desired no. of clusters are obtained.

- Method -
  - Input : $D = \{d_1, d_2, d_3 \ldots d_n\}$
  - Steps : Calculate similarity matrix Sim [i, j]
  - Repeat :
    - Merge the two most similar clusters C_1 and C_2, to form a new cluster C_0.
    - Compute similarities between C_0 and each of remaining clusters and update Sim [i, j]
  - Until there remains(s) a single or specified no. of clusters (s)
  - Output : Dendogram of clusters.

SUNDAY 12

Meetings	✓ Things To Do	✓ Important Calls	✓
	☐	☐	☐
	☐	☐	☐
	☐	☐	☐

Q. Solve using hierarchial clustering. (AGNES) and draw dendogram.

individual	V_1	V_2
A_1	1	1
A_2	1.5	2
A_3	3	4
A_4	5	7
A_5	3.5	5
A_6	4.5	6
A_7	3.5	4.5

Q. Solve using partitioning clustering algorithm.

individual	x	y
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5