relational forms.

30/11/2024

UNIT-3

* Frequent Pattern Analysis:

→ Frequent pattern: a pattern that occurs frequently in a data set.

*→ Applications:

↳ Basket Data analysis, cross marketing, catalog design, sale campaign analysis, DNA sequence analysis.

→ Importance of frequent patterns:

① Association, correlation & casuality analysis.
② sequential, structural patterns.
③ Pattern analysis in spatiotemporal
④ classification
⑤ cluster analysis
⑥ data warehousing
⑦ semantic data compression
⑧

* Market Basket Analysis:

Process of analysing customer b/w diff items that a customer will place in their baskets.

→ Basic Concepts:

→ Itemsets: a set of 1/more items.
→ k-itemset: $X = \{x_1, \ldots, x_k\}$
→ support / support count of X: Frequency of X
→ relative support: fraction of transactions that contains X.
(or) probability that a transaction contains X.

Ex:

ID	ITEMS
1	Bread, milk
2	Bread, diaper, beer, eggs.
3	milk, diaper, beer, coke
4	Bread, milk, diaper, beer.
5	bread, milk, diaper, coke.

$\ast\rightarrow$ Support $(A \rightarrow B) = \frac{\text{SPP}(A \rightarrow B)}{|T|} = \frac{s(A \cup B)}{|T|}$

① {milk, diaper}

$= \frac{3}{5} \longrightarrow$ no. of times the above combination is seen in the item sets.

$= 0.6 = 60\%$

② {milk, bread} $= \frac{3}{5} = 0.6 = 60\%$.

$\ast\rightarrow$ Confidence $(A \rightarrow B) = \frac{\text{support}(A \rightarrow B)}{\text{support}(A)}$

① {milk, bread}

Support $(A \rightarrow B) = 0.6$.

support (A) = support (milk)

$= \frac{4}{5} = 0.8$

Conf $(A \rightarrow B) = \frac{0.6}{0.8} = 0.75$.

Ex: {milk, bread, beer}

Support $= \frac{1}{5} = 0.2$

Conf $= \frac{0.2}{0.8} = 0.25$

* Closed and Max Patterns:

→ Closed: An itemset X is closed if X is frequent and there exists no super pattern $Y \supset X$ with the same support as X.

→ Max: An itemset X is a max-pattern if X is frequent and there exists no frequent super pattern $Y \supset X$

→ An itemset is said to be frequent if the corresponding support count is greater than minimum support count.

Ex:

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

Support $(E) = \frac{3}{5} = 0.6$.

Support $(CE) = \frac{2}{5} = 0.4$

Supp $(DE) = \frac{2}{5} = 0.4$

→ min_support = 2

∵ supp $(E) \neq$ support (CE) and $\neq$ supp (DE)

→ CE, DE, E are closed itemset

∵ support $(E) \geq$ min_support ⇒ frequent

" $<$ " ⇒ not frequent

∵ Supp $(E) <$ min_support ⇒ E is not maximal.

$Supp(CDE) = \frac{1}{5} = 0.2$

*→ Download closure property of frequent patterns:

All subsets of any frequent itemset must also be frequent.

Ex: If {milk, bread, butter} is a frequent itemset then {milk}, {Bread}, {butter}, {milk, bread}, {milk, butter}, {bread, butter} are frequent itemsets.

Ex:

TID	List of items
T1	A, B, C, D
T2	A, B, C, D
T3	A, B, C
T4	B, C, D
T5	C, D.

Item	Count
A	3
B	4
C	5
D.	4

min_supp = 3

∴ all the support_counts of items are ≥ min_supp

→ all items are frequent.

supersets:

Item	Count
AB	3
AC	3
AD	2
BC	4
BD	3
CD	~~4~~

A(3) → AB(3), AC(3), AD(2)

→ A (count) is not greater than its immediate superset.

A is not closed.

→ A' supersets are having count of 3 which is equal/ greater than min_supp; hence A is not maximal.

Similarly B & D are also not closed and not maximal.

for C:

C(5) → AC(3), BC(4), CD(4)

∴ C (count) is greater than and not equal to ~~AC~~, its supersets; hence C is closed; whereas AC, BC, CD being frequent; makes C a non-maximal

further supersets:

Item	Count
A, B, C	3
A, B, D	2
A, C, D	2
B, C, D	3.

Count (AB) →

AB(3) → ABC(3), ABD(2)

AB is not closed.

∵ ABC & ABD are frequent

→ AB is not maximal.

Why AC is not closed & maximal.
BC is closed & not maximal.
CD "
BD is not closed & maximal.

ABCD → (2) ⇒ not frequent.

ABC is closed & maximal.
BCD is closed & maximal

* Computational Complexity of frequent itemset mining:

→ The no. of frequent itemsets to be generated is sensitive to minsup threshold.

→ worst case: M^N

M → distinct items
N → max length of transactions.

* Frequent Item set Mining Methods:

(1) Apriori: A candidate generation & test approach
Improve efficiency of apriori

(2) FP Growth: A frequent pattern growth approach.

(3) ECLAT: Frequent Pattern Mining with vertical data format.

* Apriori:

→ Principle: If there is any itemset which is infrequent, its superset shd not be generated/tested.

→ Steps:
- Initially; scan database once to get frequent 1-itemset.
- Generate length (k+1) candidate itemsets from length k frequent itemsets
- Test the candidates against DB.
- Terminate when no frequent/candidate set can be generated.

→ Algorithm: C_k = candidate itemset of size k
L_k = frequent itemset of "

L_1 = {frequent items};
for ($k = 1$; $L_k != \emptyset$; k++) do begin
C_{k+1} = candidates generated from L_k;
for each transaction t in database do
↑ment the count of all candidates in C_{k+1} that are contained in t.
L_{k+1} = candidates in C_{k+1} with min_supp
end

→ Objective is to generate an association.

Ex: min supp = 50%

Threshold confidence = 70%

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

①

ItemSet	Support	Min Support
1	2	2/4 = 50%
2	3	75%
3	3	75%
4	1	25% → Not possible
5	3	75%

Item Set = (1, 2, 3, 5)

Pairs: (1,2); (1,3), (1,5), (2,3), (2,5) (3,5)

②

Item Set	Support	Min-Support
(1,2)	1	1/4 = 25% → X
(1,3)	2	50%
(1,5)	1	25% → X
(2,3)	2	50%
(2,5)	3	75%
(3,5)	2	50%

Item Sets: (1,3), (2,3), (2,5), (3,5)

③ (1,2,3), (1,3,5), (2,3,5), (1,2,5)

Itemset	Support	Min-Support
(1,2,3)	1	25% → X
(1,3,5)	1	25% → X
(2,3,5)	2	50%
(1,2,5)	1	25% → X

Item Set = (2,3,5)

→ Generate Association Rules for (2,3,5)

A → B

Rules	Support	Confidence
(2^3) → 5	2	2/2 = 100%
(3^5) → 2	2	2/2 = 100%
(2^5) → 3	2	2/3 = 66.6%
2 → (3^5)	2	2/3 = 66.6%
3 → (2^5)	2	2/3 = 66.6%
5 → (2^5)	2	2/3 = 66.66%

X

$$Conf = \frac{Supp(A \cup B)}{Supp(A)}$$

Conf = 70% (threshold).

⇒ Item sets = (2,3,5) and (3,5,2).

* Computational Challenges:

→ Multiple scans of transaction database.

→ Huge number of candidates.

→ Tedious workload of support counting for candidates.

* Ways to improve:

→ Reduce passes of transaction database scans

→ Shrink no. of candidates

→ facilitate support counting.

→ Partition: Scan database only twice

* Any itemset that is potentially frequent in DB must be frequent in atleast one of the partitions:

→ Scan 1: local frequent sets

→ Scan 2: global frequent sets.

* Hash Based Techniques:

Ex:

TID	Items
T1	I_1, I_2, I_5
T2	I_2, I_4
T3	I_2, I_3
T4	I_1, I_2, I_4
T5	I_1, I_3
T6	I_2, I_3
T7	I_1, I_3
T8	I_1, I_2, I_3, I_5
T9	I_1, I_2, I_3

Hash

Order of $I_1 = 1$, $I_2 = 2$, $I_3 = 3$, $I_4 = 4$, $I_5 = 5$

Itemset	Support count
I_1	6
I_2	7
I_3	6
I_4	2
I_5	2

$H(x,y) = (or(1) * 10 + or(2)) \bmod 7$

Item set	Count	Hash Function
I_1, I_2	4	5
I_1, I_3	4	6
I_1, I_4	1	0
I_1, I_5	2	1
I_2, I_3	4	2
I_2, I_4	2	3
I_2, I_5	2	4
I_3, I_4	0	—
I_3, I_5	1	0
I_4, I_5	0	—

min_supp = 3

(BC)

Bucket Addr	Bucket count	Bucket contents	L_2
0	2	I_1, I_4 – 1 I_3, I_5 – 1	No
1	2	I_1, I_5 – 2	No
2	4	I_2, I_3 – 4	Yes
3	2	I_2, I_4 – 2	No
4	2	I_2, I_5 – 2	No
5	4	I_1, I_2 – 4	Yes
6	4	I_1, I_3 – 4	Yes

L_2 is Y/N by B.C > min-sup

- Transaction Reduction.
- Partitioning

→ Dynamic Itemset Counting:

* algorithm which reduces no. of passes made over the data while keeping the no. of itemsets which are counted in any pass relatively low.

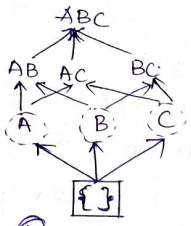
Trans	~~Ita~~ Items
T_1	A, B
T_2	A
T_3	B, C
T_4	—

	A	B	C
T_1	1	1	0
T_2	1	0	0
T_3	0	1	1
T_4	0	0	0

Min support = 25%. M = 2 (partition)

* solid box → □ itemset we have finished counting and exceeds threshold min_support.
* dashed box: → ⬚ itemset we are still counting that exceeds min_supp.
* solid circle → O we have finished counting and it is below min_supp.
* dashed circle → ◌ we are still counting that is below min_supp.

* Before transactions.

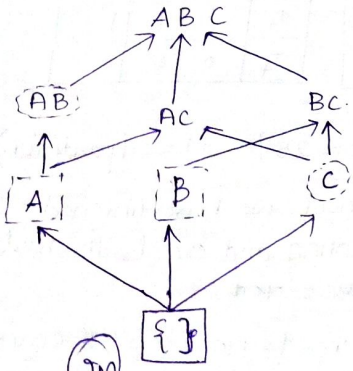


A = 0
B = 0
C = 0

(1M)

→ After 2 transactions are read: (T_1, T_2)

A = 2 ; B = 1 ; C = 0. AB = 0



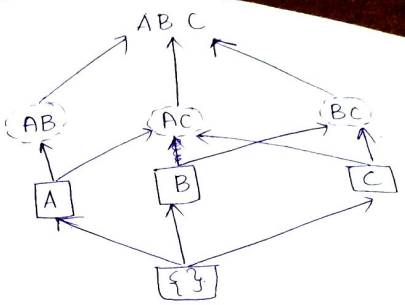
min_supp = 25%.
⇒ (1)

(2M)

→ After 4 transactions. (T_1, T_2, T_3, T_4)

A = 2 B = 2 C = 1

AB = 0 AC = 0 BC = 0.

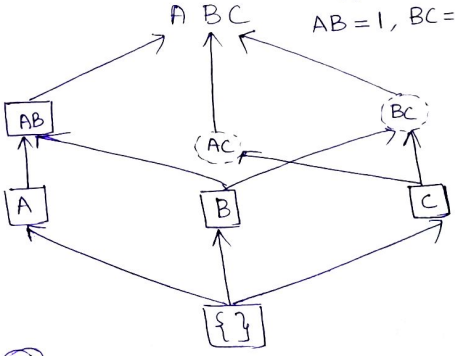


(3M)

→ After 6 transactions: $(T_1, T_2, T_3, T_4, T_1, T_2)$

A = 2, B = 2, C = 1

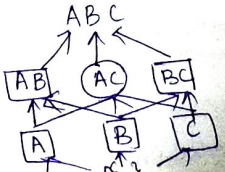
AB = 1, BC = 0, AC = 0



(4M)

→ After 8 transactions: $(T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8)$

A = 2, B = 2, C = 1 ; AB = 1 ; BC = ϕ, AC = 0



∵ There are no dashes left; we are done.

→ Sampling:

Pick up a random sample S of the given data D, then search for frequent itemsets in S instead of D.

* Apriori Approach:
  - → BFS
  - → generates a huge no. of candidates.

* FP growth approach:
  - → DFS
  - → Avoid explicit candidate generation.

* Benefits of FP tree:

→ Completeness
→ Compactness

*→ FP tree construction:

1. Scan DB once, find frequent 1-itemset
2. Sort frequent items in frequency descending order f-list.
3. Scan DB again; construct FP tree.

*→ Idea of FP growth method:

→ Recursively grow frequent patterns & database partition.

*→ Method:

→ Construct conditional pattern-base for each item set, and then its conditional FP-tree.

→ Repeat the process on each newly created conditional FP tree

→ 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

*→ Parallel projection:
  - → Project DB in parallel for each frequent item
  - → space costly
  - → partitions can be processed in parallel.

*→ Partition projection:
  - → partition the DB based on ordered frequent items.
  - → passing the unprocessed parts to the subsequent partitions.

* Advantages of FP growth:

→ Decompose both mining task & DB according to the frequent patterns obtained so far.
→ focused search of smaller databases.
→ Compressed database
→ No repeated scan.

Ex: min_supp = 3

Transaction ID	Items
T1	{E, K, M, N, O, Y}
T2	{D, E, K, N, O, Y}
T3	{A, E, K, M}
T4	{C, K, M, U, Y}
T5	{C, E, I, K, O, O}

Frequency

Item	Freq
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

→ A frequency pattern set (L) is built which will contain all the elements whose frequency is greater than (or) equal to min_supp

∴ L = {K:5, E:4, M:3, O:3, Y:3}

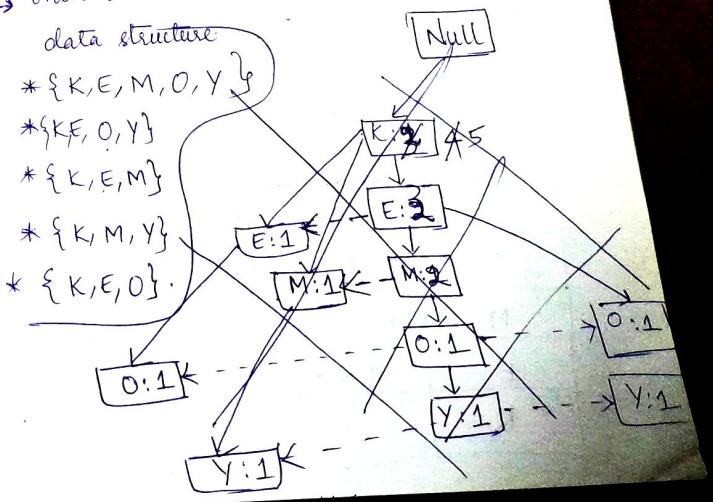
decreasing order

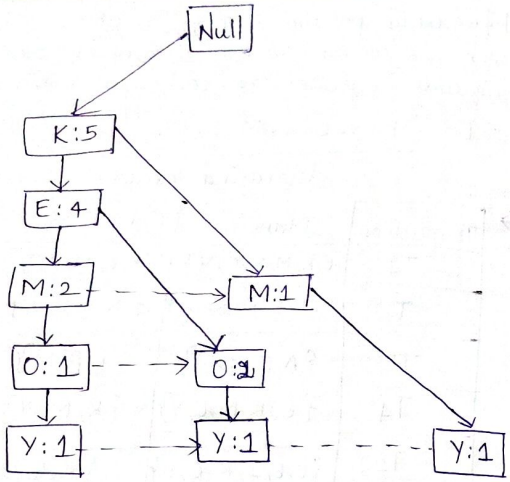
→

TransactionID	Items	Ordered-Item set
T1	{E, K, M, N, O, Y}	{K, E, M, O, Y}
T2	{D, E, K, N, O, Y}	{K, E, O, Y}
T3	{A, E, K, M}	{K, E, M}
T4	{C, K, M, U, Y}	{K, M, Y}
T5	{C, E, I, K, O, O}	{K, E, O}

→ Ordered item sets are inserted into a trie data structure

* {K, E, M, O, Y}
* {K, E, O, Y}
* {K, E, M}
* {K, M, Y}
* {K, E, O}.





* Conditional Pattern Base is computed which is path labels of all the paths which lead to any node of the given item in the frequent pattern tree.

Items	Conditional Pattern Base	Cond. Frequent Pattern Tree
Y	{K,E,M,O:1},{K,E,O:1},{K,M:1}	{K:3}
O	{K,E,M:1},{K,E:2}	{K,E:3}
M	{K,E:2},{K:1}	{K:3}
E	{K:4}	{K:4}
K		

* Frequent Pattern Rules

Items	Frequent Pattern Generated
Y	<K,Y:3>
O	<K,O:3>,<E,O:3>,<E,K,O:3>
M	<K,M:3>
E	<E,K:4>
K	

* ECLAT: Equivalence Class Clustering and Bottom-Up Lattice Traversal:

Ex:

TID	Bread	Butter	Milk	Coke	Jam
T1	1	1	0	0	1
T2	0	1	0	1	0
T3	0	1	1	0	0
T4	1	1	0	1	0
T5	1	0	1	0	0
T6	0	1	1	0	0
T7	1	0	1	0	0
T8	1	1	1	0	1
T9	1	1	1	0	0

1 → true 0 → false

min_supp = 2.

① k = 1 (1 item at a time)

Item	Tid set
Bread	$T_1, T_4, T_5, T_7, T_8, T_9$
Butter	$T_1, T_2, T_3, T_4, T_6, T_8, T_9$
Milk	$T_3, T_5, T_6, T_7, T_8, T_9$
Coke	T_2, T_4
Jam	T_1, T_8

② k = 2

Item	Tidset
{Bread, Butter}	T_1, T_4, T_8, T_9
{Bread, Milk}	T_5, T_7, T_8, T_9
~~{Bread, Coke}~~	~~T_4~~
{Bread, Jam}	T_1, T_8
{Butter, Milk}	T_3, T_6, T_8, T_9
{Butter, Coke}	T_2, T_4
{Butter, Jam}	T_1, T_8
{milk, Coke}	~~T_8~~
~~{milk, Jam}~~	~~T_8~~
{Coke, Jam}	

③ k = 3

Item	Tidset	
{Bread, Butter, Milk}	$\{T_8, T_9\}$	—
{Bread, Butter, Jam}	$\{T_1, T_8\}$	—
{Bread, Butter, Coke}	$\{T_4\}$	< 2
{Bread, Milk, Jam}	$\{T_8\}$	< 2

④ k = 4

~~Bread, Butter, Milk, Jam~~ ⇒ T_8 < (1)

⇒ Rules

Items Bought	Products
Bread	Butter
Bread	Milk
Bread	Jam
Butter	Milk
Butter	Coke
Butter	Jam
Bread & Butter	Milk
Bread & Butter	Jam

Advantages:

① Less memory
② faster speed
③ less no. of computations

* Pattern Evaluation Methods:

$$\text{lift} = \frac{P(A \cup B)}{P(A) * P(B)}$$

** Imbalance Ratio: (IR)

$$IR(A,B) = \frac{|\text{sup}(A) - \text{sup}(B)|}{\text{sup}(A) + \text{sup}(B) - \text{sup}(A \cup B)}$$

* Mining multiple-level association rules:

→ Hierarchy
→ Flexible support settings – lower level has lower support.

* Some items are more valuable but less frequent.

↳ use non-uniform, group based support.

Eg: {diamond, watch, camera}: 0.05%
{bread, milk}: 5%

* Redundancy filtering: Some rules may be redundant due to "ancestor" relationships.

→ milk ⇒ wheat bread [support = 8%, conf = 70%]
→ 2% milk ⇒ wheat bread [supp = 2%, conf = 72%]

** A rule is said to be redundant if support is close to expected value.

→ using uniform support for all levels.
⇒ giving a min-supp. value which is same for all levels.
→ using reduced min. support at lower levels.
↳ min supp is given.
→ using item (or) group based min. support.

* Mining multi-dimensional association rules from relational database (or) data warehouse:

↳ Single dimensional rules:

Ex: buys(X, "milk") ⇒ buys(X, "bread")

↳ Multi-dimensional rules:

① Inter dimension:

age(X, "19-25") ∧ occup(X, "student") ⇒ buys(X, "coke")

② hybrid dimension: (repeated predicates)

age(x, "19-25") ∧ buys(x, "popcorn") ⇒ buys(x, "coke")

predicate (task)

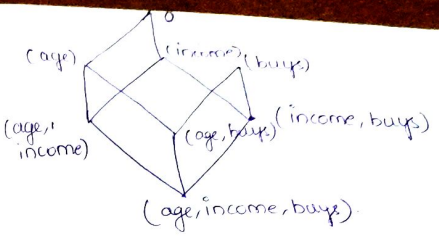
* Mining Quantitative Associations: (numerical)

→ Based on numerical attributes; methods are divided:

* static discretization (data cube methods) (manual)
* dynamic discretization (quantitative rules)
* Clustering
* Deviation

→ numerical values are replaced by ranges.

- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes is much faster.



* Negative and Rare patterns:

→ Rare patterns: Very low support but interesting

Mining: Setting individual-based/group-based support threshold for items

→ Negative patterns:

Ex: ∵ It is unlikely that one buys Ford Exp (SUV) & Toyota Prius (hybrid) together

→ Ford Exp & Toyota Prius are -vely correlated patterns

① If itemsets X and Y are both frequent but rarely occur together

∵ SUP(XUY) < SUP(X) * SUP(Y)

→ then X & Y are negatively correlated

Ex:

A store sold 2 needle 100 packages A & B, only one transaction containing both A and B.

① 200 transactions

1 tran → 100

200 → ?

* Constraint Based Frequent Pattern Mining:

→ Mining is done using data mining query language.

→ Advantages of Constraint-based mining:

* User flexibility
* Optimization.

→ Constraints

① Knowledge type constraint
   ↳ association, classification, regression

② Data constraint: using SQL like queries (task)

③ dimension/level constraint: hierarchy

④ Rule/pattern constraint → Metarules
   ↳ Constraint pushing

⑤ Interestingness constraint: supp/confidence

↓ decides the type of variables to be used together.

→ Pattern space pruning constraints:

① Anti-monotonic: if constraint c is violated, further mining is stopped.

② monotonic: if c is satisfied; no recheck.

③ succinct: c must be satisfied; so one can start with data sets satisfying c.

④ convertible: c is not monotic/anti-monotonic, it can be converted by proper ordering.

→ Data space pruning constraint:

① data succinct: space is pruned with initial pattern

② data anti-monotonic: if transaction t does not satisfy c; t can be pruned.

→ Meta-rules guided mining allows the user to give syntactical form of any rule in the form of constraint.