DWDM UNIT - III : (i)

Mining frequent Patrens, Associations and co-relations

Basics :-

Frequent Pattren :- a pattren (set of items, subsequences, substructures) that occur frequently in a dataset. [An intrinsic & important property of dataset]

motivation → finding inherent regularities in data
- what products are often purchased together
- what are subsequent purchases after buying a pc
- what kinds of DNA are sensitive to this new drug ?
- can we automatically classify web documents

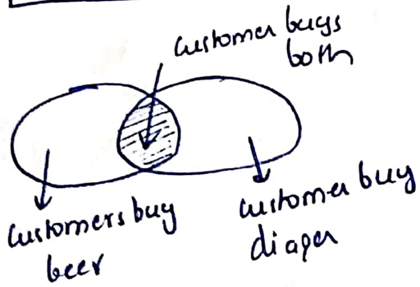
Applications
- Basket-data analysis
- cross-marketing
- Cataloguedesign
- DNA sequence analysis

"Frequent Pattrens lay foundation for many essential data mining tasks"
- classification : discriminative, frequent Pattren analysis
- clustering : Frequent Pattren-based clustering
- Pattren analysis & semantic data compression & other broad applications

TID	Items Bought
10	Beer, nuts, diaper
20	Beer, coffe, diaper
30	Beer, Diaper eggs
40	Nuts, eggs, Milk
50	Nuts, coffe, Diaper, eggs, Milk

Itemset :- A set of one or more items
eg: k-itemset $X = \{x_1, x_2, x_3 \dots x_k\}$

(absolute) Support / Support count of X : Frequency of / occurrancy of an itemset X



(relative) support :- s, is the fraction of transactions that contain X (Probability of transaction contains X) P(A∪B)

"An itemset X is frequent if X's support is no less than minimum support Threshold"

- If a subsequence appears frequently in dataset then it is a frequential sequence Pattren
- If a substructure appears frequently in dataset then it is a frequential structured Pattren :)

Structure → ~~lattice~~, subgraphs, subtrees, sub-lattices.

Confidence :- c, conditional Probability that a transaction having X also has Y P(B|A) $= \frac{P(A \cap B)}{P(B)}$

Association rules : finding all rules $X \to Y$ with min support & confidence

Market-Basket Analysis [ARM is used]
: customer analysis, buying habits shopping trends
& above all concepts come here

Long Patterns problem

: A long pattern contains combinatorial number of sub patterns

eg: $(a_1, \ldots a_{100})$

$\downarrow$

$\binom{100}{1} + \binom{100}{2} + \ldots$

$= 2^{100} - 1$

$\approx 1.27 \times 10^{30}$ subpatterns!

: Solution:
mine closed patterns & max patterns instead

worst case in generating itemsets

: M^N

- M : distinct items
- N : Max length.

Closed Pattern

: An itemset X is closed, if X is frequent and there exist no supper-pattern Y such that

$Y \supset X$

and with same support as X

Max Pattern

: An itemset X is Max, if X is frequent and there exist no super-pattern Y such that

$Y \supset X$

" closed Pattern is a lossless compression of frequent Patterns i.e reducing number of patterns & rules "

Mining Association Rules

: Process of finding frequent Patterns or associations within dataset or from set of data sets.

why? → mine Information, extract knowledge

How it is done / scalable Frequent itemset Mining methods

- Apriori (candidate generation & test)
- FP growth (frequent Pattern growth)
- Vertical data format (ECLAT)

Downward closure property of frequent patterns

" Any subset of a frequent itemset must be frequent "

Apriori Algorithm

:- [R. Agarwal, R. srikant in 1994] [iterative]

- Apriori Pruning Principle :- " If there is any itemset which is infrequent, its superset should not be generated/tested! "
- Method :-
  1. Initially scan DB for getting 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.

" Association Rule mining is of two types single level & multi-level "

TID	Itemset
T1	F, A, D, B
T2	D, A, C, E, B
T3	C, A, B, E
T4	B, A, D

An example of Apriori Algorithm

- Support-count : 60%.
- min-confidence : 80%.

Sol:- Finding the 1-itemset

Items	Count	Support Count	
A	4	$4/4 \times 100 = 100\%$	✓
B	4	$4/4 \times 100 = 100\%$	✓
D	3	$3/4 \times 100 = 75\%$	✓
C	2	$2/4 \times 100 = 50\%$	
E	2	$2/4 \times 100 = 50\%$	
F	1	$1/4 \times 100 = 25\%$	

Finding the 2-itemset

Items	Count	Support Count %	
A, D	3	$3/4 \times 100 = 75\%$	✓
A, B	4	$4/4 \times 100 = 100\%$	✓

finding the 3-itemset

Items	Count	Support Count %	
A, D, B	3	$3/4 \times 100\% = 75\%$	✓

Association rules :-

$A, D \rightarrow B$	$3/3 \times 100\% = 100\%$	✓
$B \rightarrow A, D$	$3/4 \times 100\% = 75\%$	
$A, B \rightarrow D$	$3/4 \times 100\% = 75\%$	
$D \rightarrow A, B$	$3/3 \times 100\% = 100\%$	✓
$B, D \rightarrow A$	INVALID	
$A \rightarrow B, D$	INVALID	

The Apriori Algorithm psuedocode

C_k: candidate itemset of size k

L_k: frequent itemset of size k

L_1: {frequent itemsets};

for (k=1; L_k != ∅ ; k++) do began

C_{k+1} = candidates generated from L_k ;

for each transaction t in database 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 $\cup_k L_k$

How to generate candidate keys

step1 : Self-joining L_k

step2 : Pruning

eg: L_3 = {abc, abd, acd, ace, bcd}

self joining $L_3 * L_3$

- abcd from abc & abd
- acde from acd & ace

→ Pruning :

acde is removed, because ade is not in L_3

→ C_4 = {abcd}

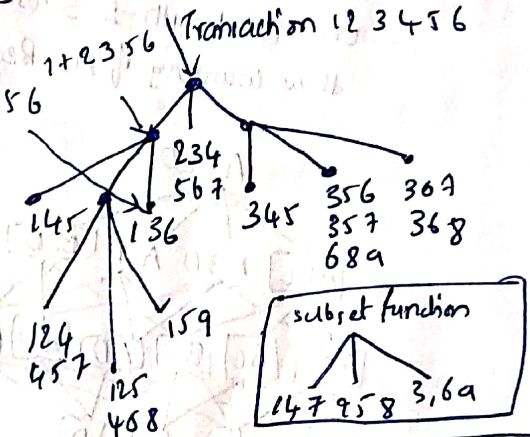
why counting supports of a candidate a problem

→ Total number of candidates can be very huge

→ One transactions may contain many candidates

solution / Hashing itemset count

- Candidate itemsets are stored in hash tree
- leaf node of hash tree contains a list of itemsets & counts
- Interior node contains a hash table
- subset function, finds all candidates contained in a transaction



Anti monotonicity

: If a set cannot pass a test, all of its supersets will fail same test as well

Improving the efficiency of Apriori Algorithm

1. Partition : Scan database only twice
   - Any itemset that is potentially frequent in DB, must be frequent in atleast one of partitions of DB

   Scan1 : Partition database & find local frequent patterns

   Scan2 : consolidate global frequent patterns

2. DHP : Reduce number of candidates :-
   - A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.

An effective hash-based Algorithm for Mining association rules
↓
SIGMOD'95

Count	itemsets
35	{ab, ad, ae}
88	{bd, be, de}
⋮	
102	{yz, qs, wt}

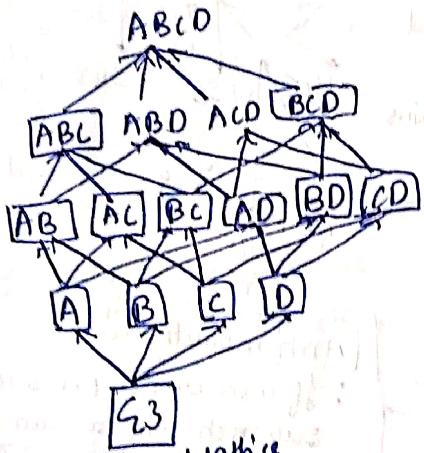
- Frequent 1-itemset: a, b, d, e
- ab is not a candidate 2-itemset if sum of count (ab, ad, ae) is below support threshold.

3. Sampling for frequent Patterns
   - select a sample of original database, mining frequent Patterns within sample using Apriori
   - Scan database once to verify frequent itemsets found in sample, only borders of closure of frequent patterns are checked

VLDB'96
↓
sampling large databases for Association rules

eg: check abcd, instead of ab, ac, - - etc
   - Scan database again to find missing frequent patterns

4. DIC: Reduce number of scans [dynamic itemset counting]
   - once both A & D are determined frequent, the counting of AD begins
   - once all length-2 subsets of BCD are determined frequent then counting of BCD Begins



Itemset lattice

SIGMOD'97

Pattern growth approach

- Bottlenecks of Apriori Approach:
   - Breadth first (level-wise) search
   - Candidate generation & test → often generates a huge number of candidates
- FPGrowth Approach
   - Depth first Search
   - Avoid explicit candidate generation

«« Growing long Patterns from short ones using local frequent items only

eg: If d is a local frequent pattern, abc → abcd is frequent pattern.

- compress a large database into a compact, frequent-Pattern tree (FP-tree) structure.
  - avoids costly database scans.
  - highly condensed, but complete.

Construction of FP-Tree

1. First create the root of tree, labelled with Null
2. scan database D a second time.
   The items in each transaction are processed in L order (Sorted according to descending support count) and a branch is created for each transaction.
3. when considered the branch to be added for a transaction, the count of each node among a common prefix is incremented by '1'.

Steps to create FP Tree

1. scan DB once, find frequent -1 itemset (single item pattern)
2. Order frequent items in frequency descending order
3. scan DB again, construct FP-Tree

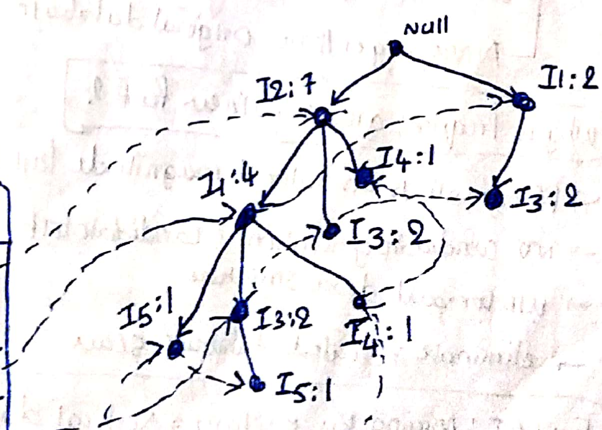
eg:-

T100 : I1, I2, I5
T200 : I2, I4.
T300 : I2, I3
T400 : I1, I2, I4
T500 : I1, I3
T600 : I2, I3
T700 : I3, I1
T800 : I1, I2, I3, I5
T900 : I1, I2, I3

Item	Supp Count
I1	6
I2	7
I3	6
I4	2
I5	2

In descending order:

Item	Support count	Node-link
I2	7	
I1	6	
I3	6	
I4	2	
I5	2	



Mining Frequent Patterns using FP-Tree

- divide & conquer
  - recursively grow frequent pattern path using FP Tree
- Method
  - For each item, construct its conditional Pattern-Base and then its conditional FP-Tree
  - Repeat the process on each newly created conditional FP-Tree
  - until resulting FP-tree is empty (or) it contains only one path

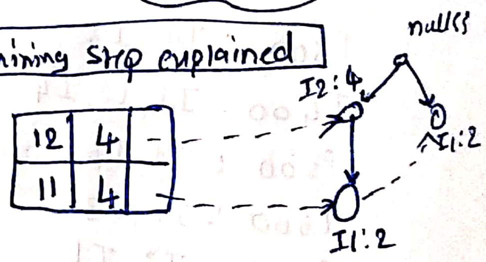
Item	conditional Pattern Base	Conditional FP Tree	Frequent Patterns
I_5	$\{(I_2, I1:1), (I_2, I1, I_3:1)\}$	$\{I_2:2, I1:2\}$	$(I_2, I_5:2)$ $(I1, I_5:2)$ $(I_2, I_1, I_5:2)$
I_4	$\{(I_2, I_1:1)(I_2:1)\}$	$\{I_2:2\}$	$\{I_2\ I_4:2\}$
I_3	$\{(I_2, I_1:2), (I2:2)\ (I1:2)\}$	$\{I_2:4, I1:2\}\ \{I1:2\}$	$(I_2\ I_3:4)$, $(I_1\ I3:2)$ $(I_2\ I_1\ I_3:2)$
I_1	$\{(I_2:4)\}$	$\{I2:4\}$	$(I_2\ I_1:4)$

Association rules can be drawn out same as apriori

Benefits of FP-Tree structure

- Completeness
  - Preserve complete information for frequent pattern mining
- Compactness
  - Reduce irrelevant information
    - infrequent items are gone
  - Items in frequency descending order
  - Never larger than original database

Mining step explained



Why is frequent growth pattern fast ?

- → FP-growth is an order of magnitude faster than Apriori & also tree projection
- → No candidate generation & candidate test
- → use compact data structure
- → eliminate repeated database scan

ECLAT : Mining By exploring vertical data format

[mining closed patterns]

- Vertical format : $t(AB) = \{T_{11}, T_{25} \ldots\}$ ← Trans-id
- Deriving frequent patterns based on vertical intersections

$t(x) = t(y)$: x & y always happen together

$t(x) \subset t(y)$: transaction having x always have Y

Which patterns are interesting? - Pattern evaluation methods

- Most Association rule mining algorithms employ a support-confidence framework.
- many of rules generated are still not interesting to user.
- The above statement is true when mining for long patterns or at low support thresholds.

Strong Rules are not necessarily Interesting

Based on subjective (or) objective perspectives.

- The Interestingness measures are objective statistics.

Then which strong association rules are interesting?

- There are several Co-relation measures help us to choose Association Rules.

Lift: The occurrence of itemset A is independent of occurrence of itemset B if $P(A \cup B) = P(A) / P(B)$;

Otherwise itemsets A & B are dependent & correlated as events

$$\therefore \text{Lift}(A,B) = \frac{P(A \cup B)}{P(A)(P(B))} \quad \left(\text{or } P(B/A)/P(B)\right)$$

- If $\text{Lift}(A,B) < 1$, then occurrence of A is negatively correlated with occurrence of B (occurrence of one likely would be absence of other)
- If $\text{Lift}(A,B) > 1$, then A & B are positively correlated. (one occurrence implies another occurrence)
- If $\text{Lift}(A,B) = 1$ then A & B are independent & there is no correlation between them.

eg:- Calculating the chi-square value (χ^2) for the given data

	game	$\overline{\text{game}}$	Σrow
video	4000	3500	7500
$\overline{\text{video}}$	2000	500	2500
Σcol	6000	4000	10000

→ P(Game) = 60%   P(Game, video) / P(game) * P(video) = 0.? $\boxed{<1}$

P(video) = 75%

∴ negative Co-relation

customer Purchasing ~~store~~

two independent purchases.

$$\chi^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

$$= \frac{(4000-4500)^2}{4500} + \frac{(3500-3000)^2}{3000} + \frac{(2000-1500)^2}{1500}$$

$$+ \frac{(500-100)^2}{100} = 555.6 \qquad \chi^2 \; (>1)$$

Lift & χ^2 are not null invariant

A comparision of pattern evaluation measures

→ If two itemsets are given then

· <u>All confidence</u> : $\text{all\_conf}(A,B) = \dfrac{\sup(A \cup B)}{\max[\sup(A), \sup(B)]}$

$$= \min[P(A|B), P(B|A)]$$

<u>MAX confidence</u> : $\text{max\_conf}(A,B) = \max[P(A|B), P(B|A)]$

<u>KULCZYNSKI</u> : $\text{Kulc}(A,B) = \frac{1}{2}(P(A|B) + P(B|A))$

<u>Cosine measure</u> : $\text{cosine}(A,B) = \dfrac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \dfrac{\sup(A \cup B)}{\sqrt{\sup(A) \times \sup(B)}}$

$$= \sqrt{P(A|B) \times P(B|A)}$$

Multiple-level Association Rules

- It is not always easy to find the strong association
- So we must use multi levels of abstraction

Imbalance ratio

3 (ii)

- Kinds of Patterns & rules
  - Basic Patterns
    - frequent Pattern
    - association rule
    - closed/max pattern
    - generation
  - Multilevel & multidimensional Patterns
    - Multilevel {uniform, varied, itemset support}
    - multidimensional Pattern {high dimensional Pattern}
    - continuous data {discretization based or Statistical}
  - Extended Patterns
    - approximate pattern
    - uncertain pattern
    - compressed pattern
    - rare Pattern/negative pattern
    - high dimensional & colossal pattern

- Mining Methods
  - Basic Mining methods
    - candidate generation (Apriori, Partition, sampling)
    - Pattern growth (FPGrowth, H-mine, FPmax, closet +)
    - vertical format (Eclat, CHARM)
  - Mining Interesting Patterns
    - interestingness (subjective vs objective)
    - constraint based mining
    - correlation rules
    - exception rules
  - Distributed, Parallel & Incremental
    - distributed/parallel mining
    - incremental mining
    - stream Pattern

- Extensions & Applications
  - extended data types
    - sequential & time series Patterns
    - structural (tree, lattice)
    - spatial
    - temporal (Periodic, evolutionary)
    - image, video etc
    - network Patterns
  - Applications
    - pattern-based classification
    - Pattern-based clustering
    - Pattern-based semantic annotation
    - Collaborative filtering
    - Privacy preserving

Pattern mining in Multi-level & multi dimensional

- Sometimes we also want interesting or rare patterns (occurs rarely but of critical importance) & negative patterns (Patterns with negative correlation between them).

"Based on the abstraction levels involved in a pattern: patterns or association rules may have items that are residing at high, low, multiple abstraction levels."

eg: buys (x, "computer") → buys (x, "Printer") ↕ high level abstraction

buys(x, "laptop") → buys(x "laser printer") ↕ low level abstraction

→ These are multi-level association rules.

"If the items or attributes in an association rule or pattern reference only one dimension then it is a single dimensional association rule/pattern."

"Otherwise multi-dimensional association rule/pattern"

eg: age(X, "20..29") ∧ income (X, "52k..58k") → buys(x, "ipad")

Boolean association rule: If a rule involve the associations between presence or absence of items.

quantitative association rule: If a rule describes association between quantitative attributes or items.

→ Based on constraints or criteria used to select patterns, patterns or rules can be discovered as:

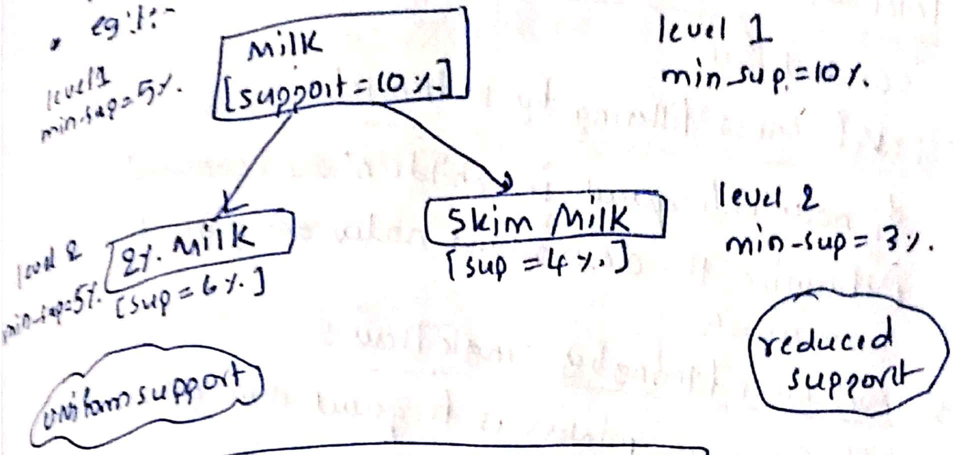
- constraint Based
- approximate
- compressed
- top-k
- redundancy-aware top k

→ Based on kinds of data & features to be mined

- Sequential Pattern
- Structural Pattern

Mining multiple level Association rules

- Items often form hierarchies
- Flexible support settings (Items at lower level are expected to have low support)
- eg:-



It is easy to find generic items to find interesting pattern

Flexible support & Redundancy filtering

- Flexible support : some items are rare but more valuable (less frequent)

→ use non-uniform, group based min support

eg: (diamond, watch, camera) : 0.05%.

(milk, bread) → 5%.

- Redundancy filtering : some rules may be redundant due to "ancestor" relationships between items.

eg:- milk → bread [s = 8%, c = 70%] (ancestor)

2% milk → bread [s = 2%, c = 72%]

"A rule is redundant if it's support is close to the expected value, based on ancestor rule."

Uniform support vs Reduced support

- uniform support : same minimum support for all levels

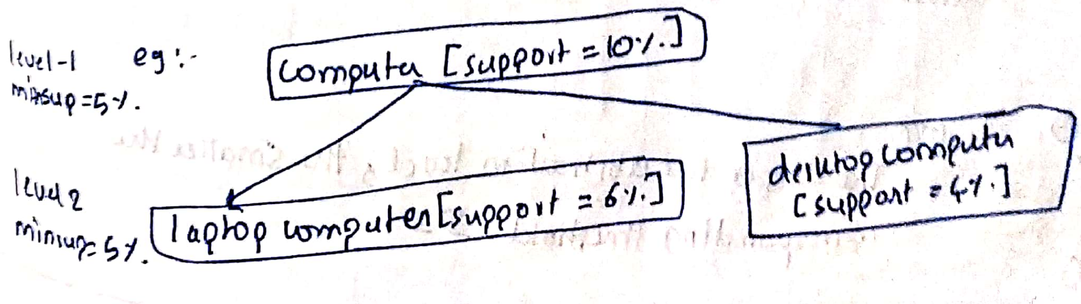
because if (sup-threshold)

↓

too high ⇒ miss low level association

too low ⇒ generate too many high level associations

eg:-



• Reduced support : reduced minimum support at lower levels

• 4 Search strategies

→ level-by-level independent (full breadth search)

Searched full

→ level-cross filtering by k-itemset

If node is frequent, its children are examined
Otherwise the descendent nodes are pruned
from search

→ level-cross filtering by single item :-

If {computer, printer} is frequent then other
nodes are examined

→ Controlled level-cross filtering by
Single item :- (same as above except)

A threshold called level passage threshold
can be setup for passing down relatively
frequent items to lower levels

Mining Multi dimensional Association

• single dimensional rules :-

buys(x, "milk") → buys (x, "bread")

• multidimensional rules (2 ≥ predicates)

→ Inter-dimension assoc. rules (no repeated predicates)

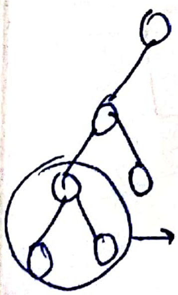
Eg:- age (x, "19-25") ∧ occupation (x, "student") → buys(X, "coke")

→ hybrid-dimension assoc. rules (repeated predicates)

age(x, "19-25") ∧ buys(x, "Popcorn") → buys(X, "coke")

• Categorical attributes : finite no. of possible values & no ordering
[data cube]

• quantitative Attributes : Numeric, implicit ordering among
values – discretization, clustering, gradient approaches



" The deeper the abstraction level, the smaller the
Corresponding threshold "

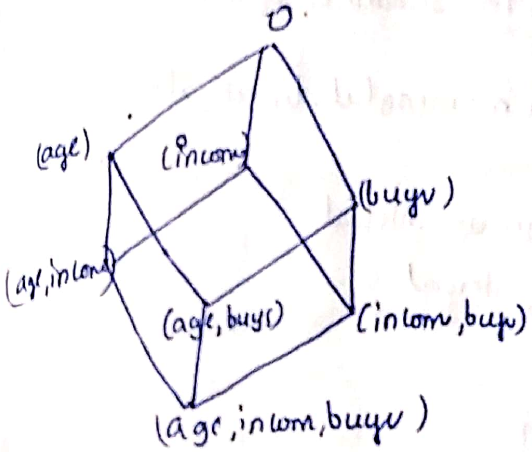
→ for each group different support, threshold

Mining quantitative associations

- Techniques can be categorized by how numerical are attributes are treated

1. Static discretization, based on predefined concept hierarchies (data cubes method)
2. Dynamic discretization, based on data distribution (discreted into bins dynamically)
3. Distance-Based association (clustering)

Static discretization



- Mining from data cubes can be much faster
- data cubes are well suited for mining
- Numerical values are replaced by ranges.

dynamic discretization

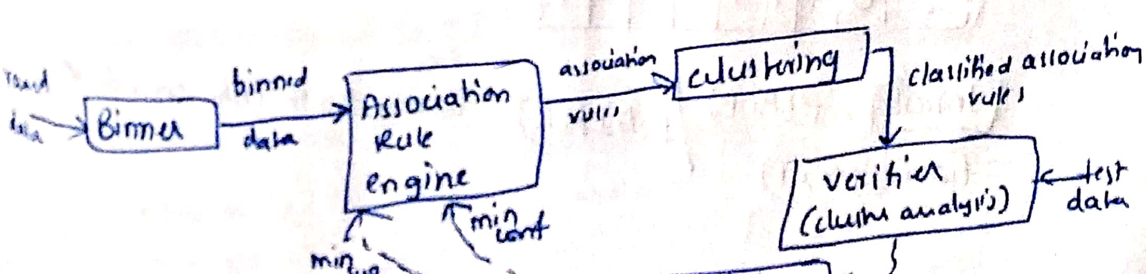
Ex (Sex = female) ⇒ wage : Mean = \$7/hr (overall mean = \$9)

- LHS : subset of population
- RHS : An extraordinary behaviour of this subset

- This rule is accepted only if a statistical test (z-test) confirms the interference with high confidence.

→ Numeric values are dynamically discretized. Such that confidence or compactness of rules is maximized

ARCS [Association Rule clustering system]



clustering the association rule

- Age (x, 34) ∧ Income (x, "31k .. 40k") → buys (x, "iPhone")

  Age (x, 35) ∧ Income (x, "31k .. 49k") → buys (x, "iPhone")

  Age (x, 34) ∧ Income (x, "31k .. 50k") → buys (x, "iPhone")

↓

They can be clustered & replaced by

Age (x, 34..35) ∧ income (x, "31k .. 50K") ⇒ buys (x, "iPhone")

Mining distance base association rules

- Binning methods donot capture the semantics of interval data
- distance-based partitioning more meaningful discretization considering
  - density, number of points in an interval
  - "closeness" of points in an interval

Interesting new measurements

Objective measures
- support
- confidence

Subjective measure

A rule pattern is interesting if

(unexpected) suprise to user &

& actionable (user can do something with it)

Correlation (lift)

$$corr(A,B) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)}$$

If corr < 1 -ve corr

If corr > 1 +ve corr

If corr = 1 independent

eg:-

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

$\boxed{\text{Interest}}$: $P(A \cap B) / P(A)P(B)$: Taking Both into consideration

(correlation, lift)

Constraint Based mining | mining by Constraints

Dimensional constraint

Knowledge Based mining constraint → specific type of knowledge (specification)

Data constraint → task-relevant data

Interestingness Constraint → specifying thresholds to support & confidence

Rule Based mining constraint → mining by specific forms of rules.

Such constraints may be expressed as metarules [Syntactic form of rules]

as max or min # of predicates that can occur in rule antecedent, consequent or relationship among attributes or in aggregates.

improve efficiency of mining process.

↳ template of metarule.

$P1 \wedge P2 \wedge P3 \wedge \ldots P_i \Rightarrow Q1 \wedge Q2 \wedge Q3 - \wedge Q_j$

eg: Data mining can search for rules that match with metarule

age (x, "30..39") ∧ income (x, "41k..60k") → buys (x, "SW")

Monotonicity: If a set S satisfies a Constraint then any superset of S satisfies a constraint.

eg: sum (I.Price) >=100

Anti montonicity:- If a set S ~~satisfies~~ Violates a constraint then any superset of S also violates constraint.

eg:1 sum (S.Price) ≤ V → is antimonotone

eg:2 Avg (I.Price) <=100 is not antimonotone

Succint Constraint:- only sets that satisfy constraint are enumerated

Convertible Constraint:- neither Monotonic nor Anti-monotonic But Convertible

Relationships among Category of Constraints