UNIT-5 Mining Association Rules in Large Databases

Lecture	Topic
**	
Lecture-27	Association rule mining
Lecture-28	Mining single-dimensional Boolean association rules from transactional databases
Lecture-29	Mining multilevel association rules from transactional databases
Lecture-30	Mining multidimensional association rules from transactional databases and data warehouse
Lecture-31	From association mining to correlation analysis
Lecture-32	Constraint-based association mining

Lecture-27

Association rule mining

What Is Association Mining?

- Association rule mining
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Applications
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

Association Mining

- Rule form

  prediction (Boolean variables) => prediction (Boolean variables) [support, confidence]

  – Computer => antivirus_software [support =2%, confidence = 60%]

  – buys (x, “computer”) $\rightarrow$ buys (x, “antivirus_software”) [0.5%, 60%]

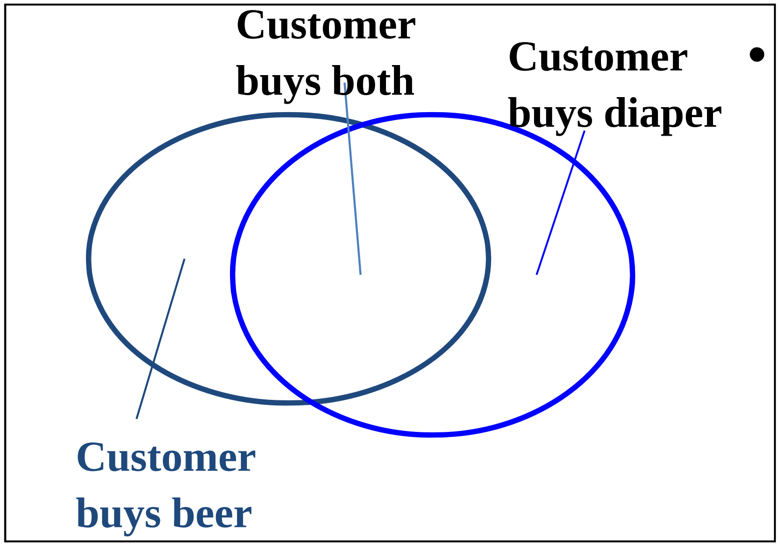
Association Rule: Basic Concepts

- Given a database of transactions each transaction is a list of items (purchased by a customer in a visit)
- Find all rules that correlate the presence of one set of items with that of another set of items
- Find frequent patterns
- Example for frequent itemset mining is market basket analysis.

Association rule performance measures

- Confidence
- Support
- Minimum support threshold
- Minimum confidence threshold

Rule Measures: Support and Confidence



- Find all the rules X & $Y \Rightarrow Z$ with minimum confidence and support
  - support, s, probability that a transaction contains $\{X \cup Y \cup Z\}$
  - confidence, c, conditional probability that a transaction having $\{X \cup Y\}$ also contains Z

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Let minimum support 50%, and minimum confidence 50%, we have

- $A \Rightarrow C$ (50%, 66.6%)
- $C \Rightarrow A$ (50%, 100%)

Martket Basket Analysis

- Shopping baskets
- Each item has a Boolean variable representing the presence or absence of that item.
- Each basket can be represented by a Boolean vector of values assigned to these variables.
- Identify patterns from Boolean vector
- Patterns can be represented by association rules.

Association Rule Mining: A Road Map

- Boolean vs. quantitative associations

  - Based on the types of values handled

  – buys(x, “SQLServer”) ^ buys(x, “DMBook”) => buys(x, “DBMiner”) [0.2%, 60%]

  – age(x, “30..39”) ^ income(x, “42..48K”) => buys(x, “PC”) [1%, 75%]

- Single dimension vs. multiple dimensional associations
- Single level vs. multiple-level analysis

Lecture-28

Mining single-dimensional Boolean association rules from transactional databases

Apriori Algorithm

- Single dimensional, single-level, Boolean frequent item sets
- Finding frequent item sets using candidate generation
- Generating association rules from frequent item sets

Mining Association Rules—An Example

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \Rightarrow C$:

support = support($\{A \cup C\}$) = 50%

confidence = support($\{A \cup C\}$)/support($\{A\}$) = 66.6%

The Apriori principle:

Any subset of a frequent itemset must be frequent

Mining Frequent Itemsets: the Key Step

- Find the *frequent itemsets*: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*} should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to *k (k*-itemset*)*
- Use the frequent itemsets to generate association rules.

The Apriori Algorithm

- Join Step
  - C_k is generated by joining $L_{k-1}$with itself
- Prune Step
  - Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

The Apriori Algorithm

- <u>Pseudo-code</u>:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k

  L_1 = {frequent items};
  for (k = 1; L_k $!=\varnothing$; k++) **do begin**
  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 $\cup_k L_k$;

The Apriori Algorithm — Example

Database D

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

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$

  from L_{k-1} p, L_{k-1} q

  where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$

- Step 2: pruning

  forall *itemsets c in C_k* do

  forall *(k-1)-subsets s of c* do

  if *(s is not in L_{k-1})* then delete *c* from *C_k*

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method
  - Candidate itemsets are stored in a hash-tree
  - Leaf node of hash-tree contains a list of itemsets and counts
  - Interior node contains a hash table
  - Subset function: finds all the candidates contained in a transaction

Example of Generating Candidates

- L_3={*abc, abd, acd, ace, bcd*}
- Self-joining: $L_3 * L_3$
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *acde* is removed because *ade* is not in L_3
- C_4={*abcd*}

Methods to Improve Apriori's Efficiency

- Hash-based itemset counting
  - A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction
  - A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning
  - Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB

Methods to Improve Apriori's Efficiency

- Sampling
  - mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting
  - add new candidate itemsets only when all of their subsets are estimated to be frequent

Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only

Lecture-29

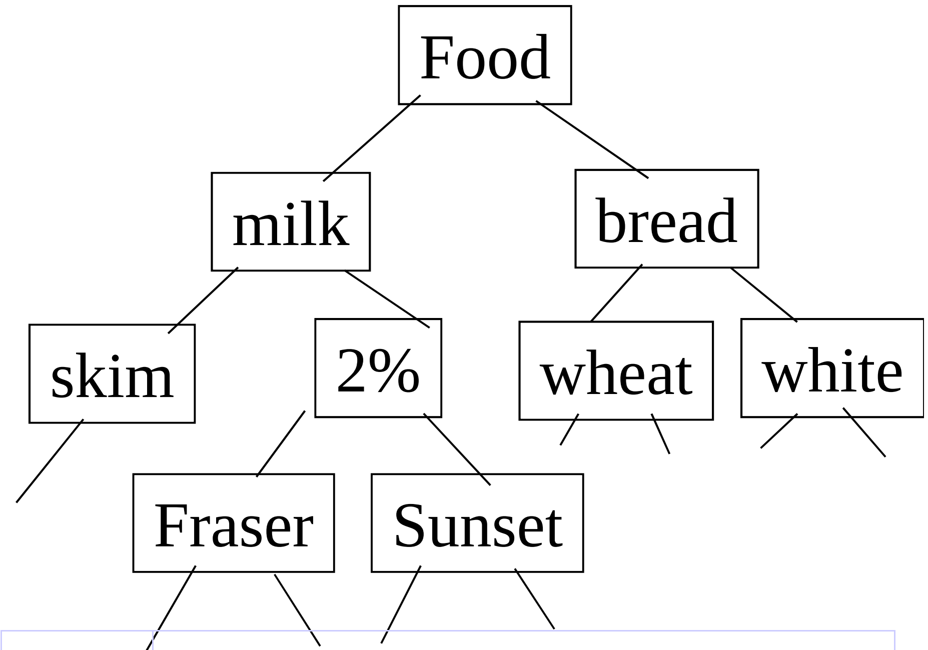
Mining multilevel association rules from transactional databases

Mining various kinds of association rules

- Mining Multilevel association rules
  - Concepts at different levels
- Mining Multidimensional association rules
  - More than one dimensional
- Mining Quantitative association rules
  - Numeric attributes

Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Multi-level Association

- Uniform Support- the same minimum support for all levels
  - **+** One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support.
  - – Lower level items do not occur as frequently. If support threshold
    - too high $\Rightarrow$ miss low level associations
    - too low $\Rightarrow$ generate too many high level associations

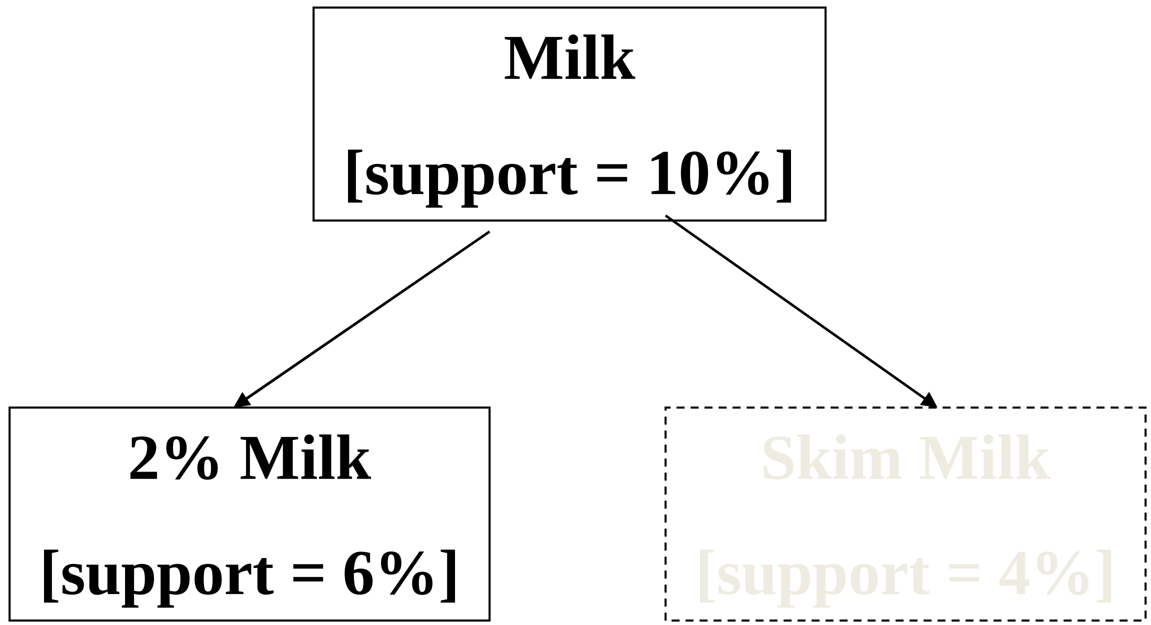
Multi-level Association

- Reduced Support- reduced minimum support at lower levels
  - There are 4 search strategies:
    - Level-by-level independent
    - Level-cross filtering by k-itemset
    - Level-cross filtering by single item
    - Controlled level-cross filtering by single item

Uniform Support

Multi-level mining with uniform support

Level 1
min_sup = 5%



Level 2
min_sup = 5%

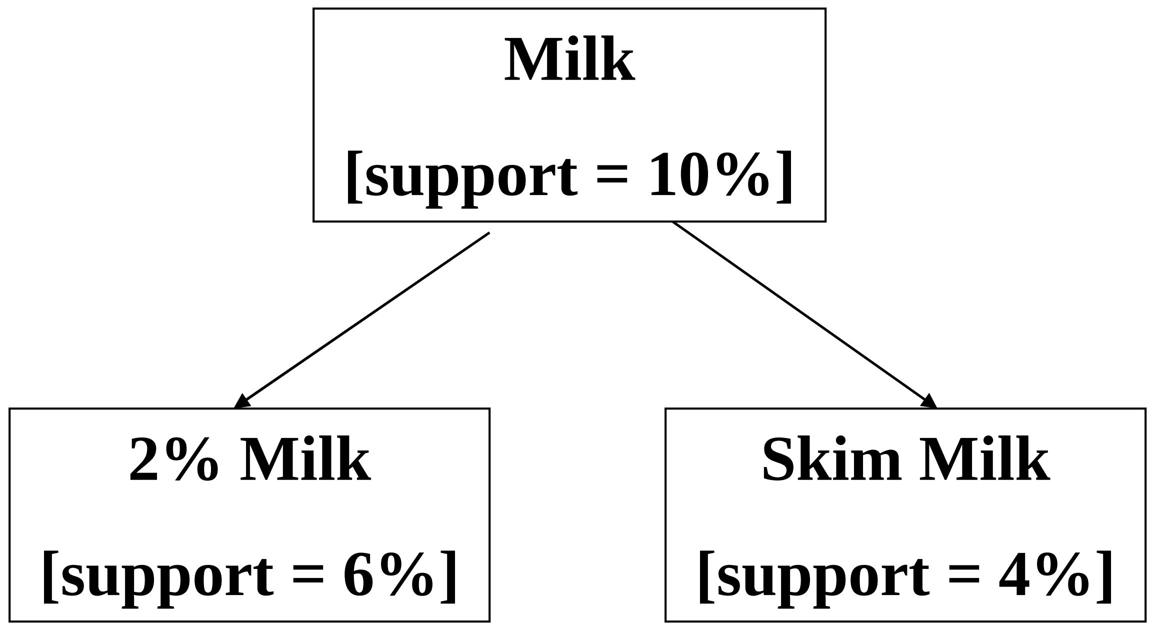
Back

Reduced Support

Multi-level mining with reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 3%



Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

Lecture-30

Mining multidimensional association rules from transactional databases and data warehouse

Multi-Dimensional Association

- Single-dimensional rules

  $buys(X, \text{"milk"}) \Rightarrow buys(X, \text{"bread"})$

- Multi-dimensional rules
  - Inter-dimension association rules -no repeated predicates

    $age(X,\text{"19-25"}) \wedge occupation(X,\text{"student"}) \Rightarrow buys(X,\text{"coke"})$
  - hybrid-dimension association rules -repeated predicates

    $age(X,\text{"19-25"}) \wedge buys(X, \text{"popcorn"}) \Rightarrow buys(X, \text{"coke"})$

Multi-Dimensional Association

- Categorical Attributes
  - finite number of possible values, no ordering among values
- Quantitative Attributes
  - numeric, implicit ordering among values

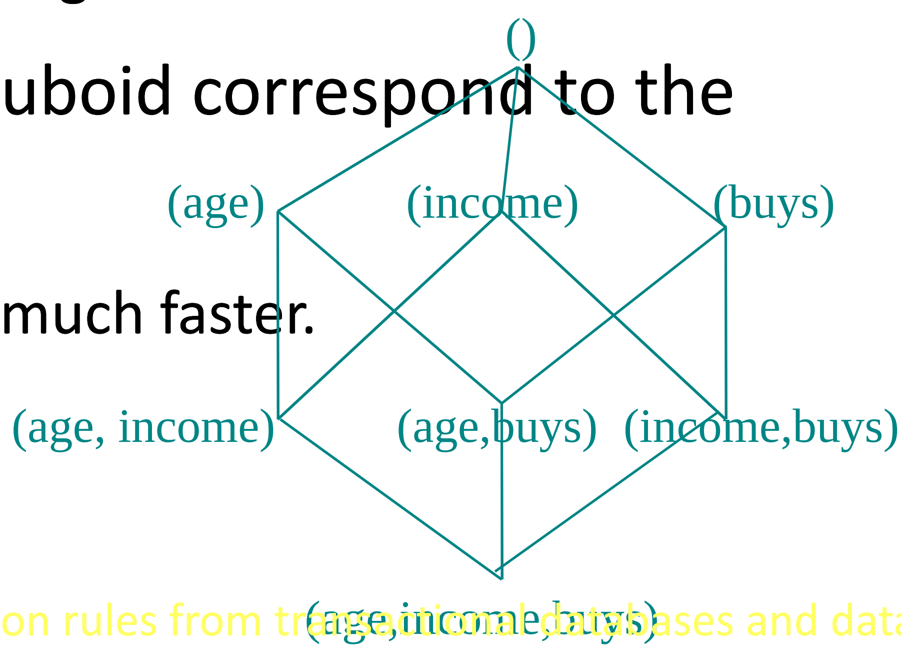
Techniques for Mining MD Associations

- Search for frequent *k*-predicate set:
  - Example: {age, occupation, buys} is a 3-predicate set.
  - Techniques can be categorized by how age are treated.

1. Using static discretization of quantitative attributes
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.
2. Quantitative association rules
   - Quantitative attributes are dynamically discretized into “bins”based on the distribution of the data.
3. Distance-based association rules
   - This is a dynamic discretization process that considers the distance between data points.

Static Discretization of Quantitative Attributes

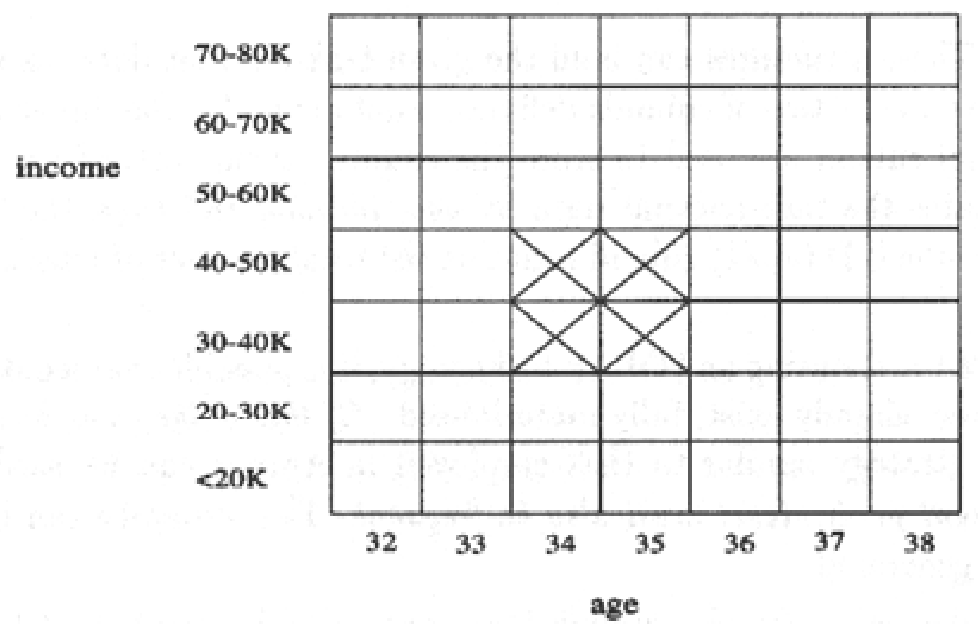
- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubescan be much faster.



Quantitative Association Rules

- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized.
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster "adjacent"

  association rules
  to form general
  rules using a 2-D
  grid.



- Example:

age(X,"30-34") $\wedge$ income(X,"24K - 48K")

$\Rightarrow$ buys(X,"high resolution TV")

Lecture-31

From association mining to correlation analysis

Interestingness Measurements

- Objective measures
  - Two popular measurements

    support

    confidence

- Subjective measures

  A rule (pattern) is interesting if

  *it is *unexpected* (surprising to the user); and/or

  **actionable* (the user can do something with it)

Criticism to Support and Confidence

- Example
  - Among 5000 students
    - 3000 play basketball
    - 3750 eat cereal
    - 2000 both play basket ball and eat cereal
  - *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
  - *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is far more accurate, although with lower support and confidence

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(col.)	3000	2000	5000

Criticism to Support and Confidence

- Example
  - X and Y: positively correlated,
  - X and Z, negatively related
  - support and confidence of X=>Z dominates
- We need a measure of dependent or correlated events

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

- P(B|A)/P(B) is also called the lift of rule A => B

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

Rule	Support	Confidence
X=>Y	25%	50%
X=>Z	37.50%	75%

Other Interestingness Measures: Interest

- Interest (correlation, lift) $$\frac{P(A \wedge B)}{P(A)P(B)}$$
  - taking both P(A) and P(B) in consideration
  - P(A^B)=P(B)*P(A), if A and B are independent events
  - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

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

Itemset	Support	Interest
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

Lecture-32

Constraint-based association mining

Constraint-Based Mining

- Interactive, exploratory mining
- kinds of constraints
  - Knowledge type constraint- classification, association, etc.
  - Data constraint: SQL-like queries
  - Dimension/level constraints
  - Rule constraint
  - Interestingness constraints

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - P(x, y) ^ Q(x, w) $\rightarrow$ takes(x, “database systems”).
  - Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD’98).
    - sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000
- 1-variable vs. 2-variable constraints
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Constrain-Based Association Query

- Database: (1) trans (TID, Itemset), (2) itemInfo (Item, Type, Price)
- A constrained asso. query (CAQ) is in the form of $\{(S_1, S_2)|C\}$,
  - where C is a set of constraints on S_1, S_2 including frequency constraint
- A classification of (single-variable) constraints:
  - Class constraint: $S \subset A$. *e.g.* $S \subset Item$
  - Domain constraint:
    - $S \theta v$, $\theta \in \{=, \neq, <, \leq, >, \geq\}$. *e.g.* $S.Price < 100$
    - $v \theta S$, θ is $\in$ or $\notin$. *e.g.* $snacks \notin S.Type$
    - $V \theta S$, or $S \theta V$, $\theta \in \{\subseteq, \subset, \not\subset, =, \neq\}$
      - *e.g.* $\{snacks, sodas\} \subseteq S.Type$
  - Aggregation constraint: $agg(S)\ \theta\ v$, where *agg* is in {*min, max, sum, count, avg*}, and $\theta \in \{=, \neq, <, \leq, >, \geq\}$.
    - *e.g.* $count(S_1.Type) = 1$, $avg(S_2.Price) < 100$

Constrained Association Query Optimization Problem

- Given a CAQ = { (S_1, S_2) / C }, the algorithm should be :
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found
- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.
- Our approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Anti-monotone and Monotone Constraints

- A constraint C_a is anti-monotone iff. for any pattern S not satisfying C_a, none of the super-patterns of S can satisfy C_a
- A constraint C_m is monotone iff. for any pattern S satisfying C_m, every super-pattern of S also satisfies it

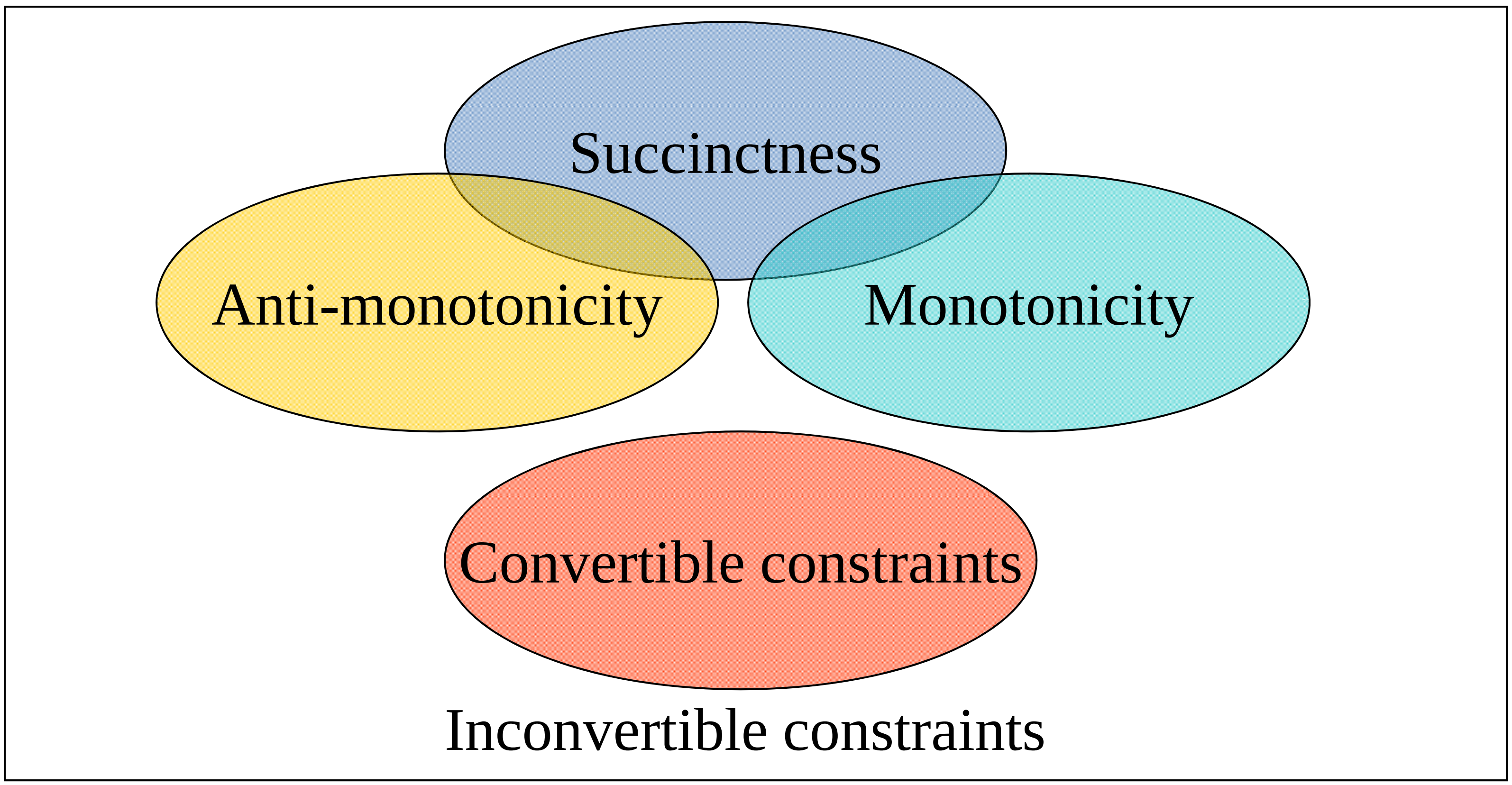
Succinct Constraint

- A subset of item I_s is a succinct set, if it can be expressed as $\sigma_p(I)$ for some selection predicate p, where σ is a selection operator
- $SP \subseteq 2^I$ is a succinct power set, if there is a fixed number of succinct set $I_1, \ldots, I_k \subseteq I$, s.t. SP can be expressed in terms of the strict power sets of $I_1, \ldots, I_k$ using union and minus
- A constraint C_s is succinct provided $SAT_{Cs}(I)$ is a succinct power set

Convertible Constraint

- Suppose all items in patterns are listed in a total order R
- A constraint C is convertible anti-monotone iff a pattern S satisfying the constraint implies that each suffix of S w.r.t. R also satisfies C
- A constraint C is convertible monotone iff a pattern S satisfying the constraint implies that each pattern of which S is a suffix w.r.t. R also satisfies C

Relationships Among Categories of Constraints



Property of Constraints: Anti-Monotone

- Anti-monotonicity: *If a set S violates the constraint, any superset of S violates the constraint.*
- Examples:
  - *sum(S.Price)* $\leq$ *v* is anti-monotone
  - *sum(S.Price)* $\geq$ *v* is not anti-monotone
  - *sum(S.Price)* = *v* is partly anti-monotone
- Application:
  - Push "*sum(S.price)* $\leq$ 1000" deeply into iterative frequent set computation.

Characterization of Anti-Monotonicity Constraints

$S\ \theta\ v,\ \theta \in \{=, \leq, \geq\}$	yes
$v \in S$	no
$S \supseteq V$ $S \subseteq V$ $S = V$	no yes partly
$min(S) \leq v$ $min(S) \geq v$ $min(S) = v$	no yes partly
$max(S) \leq v$ $max(S) \geq v$ $max(S) = v$	yes no partly
$count(S) \leq v$ $count(S) \geq v$ $count(S) = v$	yes no partly
$sum(S) \leq v$ $sum(S) \geq v$ $sum(S) = v$	yes no partly
$avg(S)\ \theta\ v,\ \theta \in \{=, \leq, \geq\}$	convertible
(frequent constraint)	(yes)

Example of Convertible Constraints: Avg(S) θ V

- Let R be the value descending order over the set of items
  - E.g. I={9, 8, 6, 4, 3, 1}
- Avg(S) $\geq$ v is convertible monotone w.r.t. R
  - If S is a suffix of S_1, avg(S_1) $\geq$ avg(S)
    - {8, 4, 3} is a suffix of {9, 8, 4, 3}
    - avg({9, 8, 4, 3})=6 $\geq$ avg({8, 4, 3})=5
  - If S satisfies avg(S) $\geq$v, so does S_1
    - {8, 4, 3} satisfies constraint avg(S) $\geq$ 4, so does {9, 8, 4, 3}

Property of Constraints: Succinctness

- Succinctness:
  - For any set S_1 and S_2 satisfying C, $S_1 \cup S_2$ satisfies C
  - Given A_1 is the sets of size 1 satisfying C, then any set S satisfying C are based on A_1 , i.e., it contains a subset belongs to A_1 ,
- Example :
  - $sum(S.Price) \geq v$ is not succinct
  - $min(S.Price) \leq v$ is succinct
- Optimization:
  - If C is succinct, then C is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.

Characterization of Constraints by Succinctness

$S \theta v, \theta \in \{=, \leq, \geq\}$	Yes
$v \in S$	yes
$S \supseteq V$	yes
$S \subseteq V$	yes
$S = V$	yes
$\min(S) \leq v$	yes
$\min(S) \geq v$	yes
$\min(S) = v$	yes
$\max(S) \leq v$	yes
$\max(S) \geq v$	yes
$\max(S) = v$	yes
$\text{count}(S) \leq v$	weakly
$\text{count}(S) \geq v$	weakly
$\text{count}(S) = v$	weakly
$\text{sum}(S) \leq v$	no
$\text{sum}(S) \geq v$	no
$\text{sum}(S) = v$	no
$\text{avg}(S) \theta v, \theta \in \{=, \leq, \geq\}$	no
(frequent constraint)	(no)