Association Rule Mining

- **Market-Basket Analysis**
- **Grocery Store: Large no. of ITEMS**
- **Customers fill their market baskets with subset of items**
- **98% of people who purchase diapers also buy beer**
- **Used for shelf management**
- **Used for deciding whether an item should be put on sale**
- **Other interesting applications**
  - Basket=documents, Items=words

    Words appearing frequently together in documents may represent phrases or linked concepts. Can be used for intelligence gathering.

Association Rules

- Purchasing of one product when another product is purchased represents an AR
- Used mainly in retail stores to
  - Assist in marketing
  - Shelf management
  - Inventory control
- Faults in Telecommunication Networks, traffic analysis, document analysis, bioinformatics, computational chemistry,
- Transaction Database
- Item-sets, Frequent or large item-sets
- Support & Confidence of AR

Types of Association Rules

- **Boolean/Quantitative ARs**

  Based on type of values handled

  Bread ☐ Butter (Presence or absence)

  age(X, “30….39”) & income(X, “42K…48K”) ☐ buys(X, Projection TV)

- **Single/Multi-Dimensional ARs**

  Based on dimensions of data involved

  buys(X,Bread) ☐ buys(X,Butter)

  Single/Multi-Level ARs

  Based on levels of Abstractions involved

  age(X, “30….39”) ☐ buys(X, laptop)

  age(X, “30….39”) ☐ buys(X, computer)

Support & Confidence

- **A rule must have some minimum user-specified** *confidence*

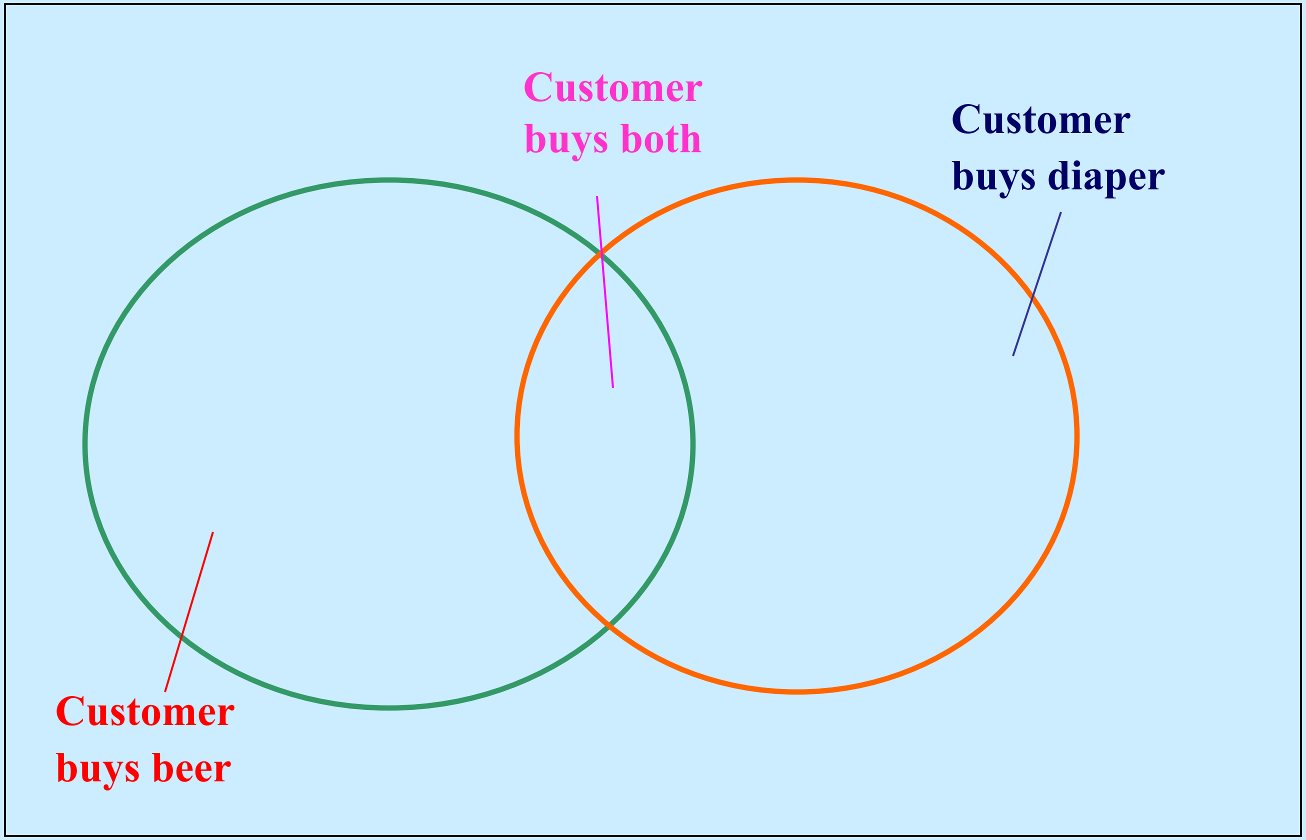
  1 & 2 => 3 has 90% confidence if when a customer bought 1 and 2, in 90% of cases, the customer also bought 3.

- **A rule must have some minimum user-specified** *support*

  1 & 2 => 3 should hold in some minimum percentage of transactions to have business value

- ***AR X => Y holds with support T, if T% of transactions in DB that support X also support Y***

Support & Confidence



Support & Confidence

I=Set of all items

D=Transaction Database

AR A=>B has support s if s is the %age of transactions in D that contain AUB (both A & B)

s(A=>B)=P(AUB)

AR A=>B has confidence c in D if c is the %age of transactions in D containing A that also contain B

c(A=>B)=P (B/A)=P(AUB)/P(A)

Example

- Transaction Database

Transaction Id	Purchased Items
1	{1, 2, 3}
2	{1, 4}
3	{1, 3}
4	{2, 5, 6}

- For minimum support = 50%, minimum confidence = 50%, we have the following rules

  1 => 3 with 50% support and 66% confidence

  3 => 1 with 50% support and 100% confidence

Mining Associations Rules

2 Step Process

- **Find all frequent Itemsets**

i.e. all itemsets satisfying *min_sup*

- **Generate strong ARs from frequent itemsets**

i.e. ARs satisfying *min_sup* & *min_conf*

Frequent Itemsets (FIs)

Algorithms for finding FIs

1. *Apriori*
2. *Sampling*
3. *Partitioning*
4. *Hash based Technique*
5. *Transaction Reduction*
6. *etc*

Apriori Algorithm (Boolean ARs)

Candidate Generation

- **Level-wise search**
- **Frequent 1-itemset (L_1) is found**
- **Frequent 2-itemset (L_2) is found & so on…**
- **Until no more Frequent k-itemsets (L_k) can be found**
- **Finding each L_k requires one pass**

Apriori **Algorithm**

- **Apriority Property**

  All nonempty subsets of a FI must also be frequent”

  i.e., if {AB} is a frequent itemset, both {A} and {B} should be a frequent itemset

- **Anti-Monotone Property**

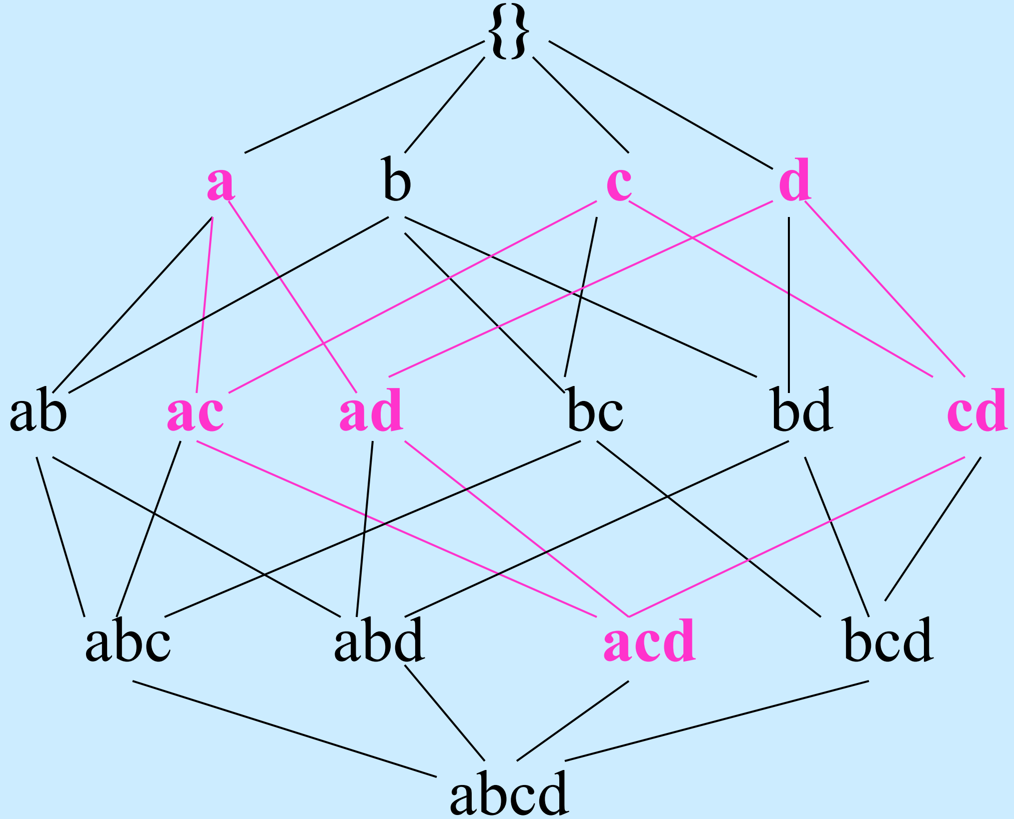
  “If a set cannot pass a test, all its supersets will fail the test as well”

  P(I) < min_sup □ P(I U A) < min_sup, where A is any item

  Property is monotonic in the context of failing a test

Frequent itemset /Apriori Property: example

If {*a*, *c*, *d*} is a large itemset then {*a*, *c*}, {*a*, *d*}, {*c*, *d*}, {*a*}, {*c*},{*d*}, {} are large itemsets too.



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

Apriori Algorithm

2-Step Process

Join Step (candidate generation)

Guarantees that no candidate of length > k are generated using Lk-□

Prune Step

Prunes those candidate itemsets all of whose subsets are not frequent

Candidate Generation

Given L_{k-1}

$C_k = \varphi$

For all itemsets $l_1 \in L_{k-1}$ do

For all itemsets $l_2 \in L_{k-1}$ do

If $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots \wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k-1]$

Then c= $l_1[1], l_1[2], l_1[3] \ldots. l_1[k-1], l_2[k-1]$

$C_k = C_k \cup \{c\}$

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*}

ARs from FIs

- For each FI l, generate all non-empty subsets of l
- For each non-empty subset s of l, output the rule $s \Rightarrow (l-s)$ if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$

Example

- Suppose l = {2,3,5}
- {2,3}, {2.5}, {3,5}, {2}, {3}, & {5}
- Association Rules are

2,3 ⇒ 5 confidence 100%

2,5 ⇒ 3 confidence 66%

3,5 ⇒ 2 confidence 100%

2 ⇒ 3,5 confidence 100%

3 ⇒ 2,5 confidence 66%

5 ⇒ 2,3 confidence 100%

Apriori: Some Observations

- C_2= L1*L1
- No. of Candidates in C_2 = ${}^{L1}C_2$
- The larger the C_2 / C_k the more processing cost required to discover FIs

Variations of the Apriori

Many variations of the Apriori has been proposed that focus on improving the efficiency of the original algorithm

- Hash-based technique- hashing itemset counts
- Transaction reduction-reducing the number of transactions scanned in future iterations
- Partitioning-partitioning the data to find candidate itemsets
- Sampling-mining on a subset of the given data
- Dynamic itemset counting-adding candidate itemsets at different points during a scan

Sampling Algorithm

- Random transactions of the original database are selected (sampled) and placed in a much smaller sampled database.
- The size of sampled database is small enough so that it can reside in main memory.
- This reduces the number of (original) database scans to at most two.
- Any standard algorithm, such as Apriori, can be used to create a set of large itemsets in sampled database.

Sampling Algorithm cont…

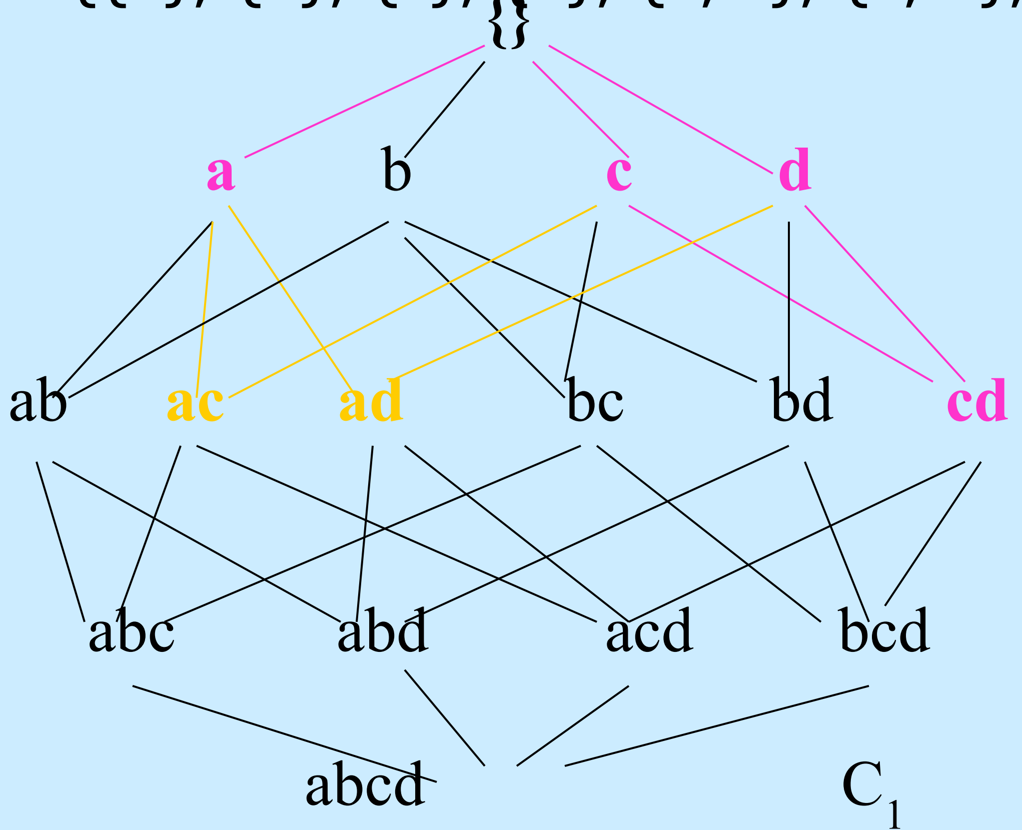
- Since these large itemsets is applied to sampled database, some may not be the actual large itemsets of the original database. These itemsets are called *potentially large itemsets*, and *PL* denotes the set of potentially large itemsets.
- Some actual large itemsets may not be in *PL*. Additional candidates for large itemsets are determined by applying *negative border function*, *NB*(), against *PL*.
- Negative border returns the itemsets that are not in *PL* but has all of their subsets in *PL*.
- Usually, the minimum support threshold is lowered when finding the *PL* from sampled database.

Sampling Algorithm : Algorithm

1. Sample transactions from Database *D*.
2. Using Apriori (or something else) algorithm to find *PL* from sampled database.
3. The candidate set C_1 contain itemsets from $PL \cup \mathrm{NB}(PL)$.
4. Scan the original database, check the support of each candidate in C_1. Those that meet the minimum support requirement will be added into *L*.
5. If some itemsets from NB(*PL*) were added into *L* in step 4.
   Initially candidate set C_2 is equal to *L*.
   Repeatedly add $\mathrm{NB}(C_2)$ into C_2 until no growth in C_2.
   Scan the original database, check the support for each candidate in C_2 . Adding large itemsets into *L*.

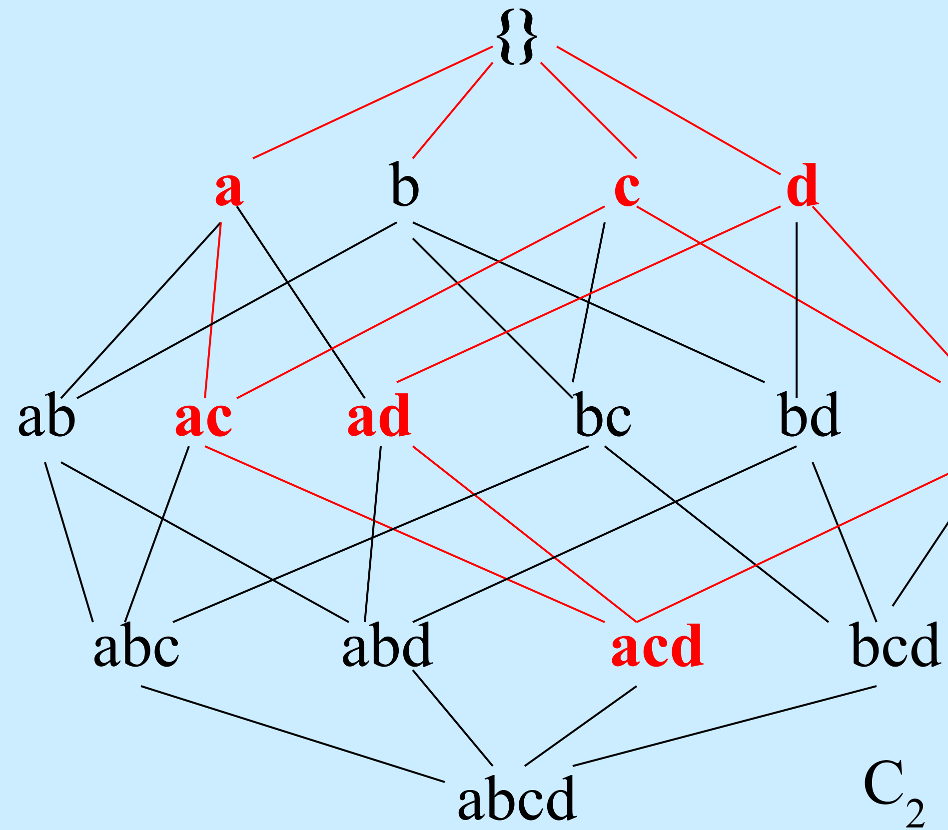
Sampling Algorithm :Example

- Let $I = \{a, b, c, d\}$,
  - After step 2, let $PL = \{\{a\}, \{c\}, \{d\}, \{c, d\}\}$.
  - After step 3, $C_1 = \{\{a\}, \{b\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}, \{c, d\},$



Sampling Algorithm :Example

Assume that L = {{a},{c}, {d}, {a, c}, {a, d}, {c, d}} after the database scan in step 4. Since {a, c} and {a, d} are in NB(PL), we need to execute step 5. C_2 will be L U {{a, c, d}}.



Partitioning

- Instead of sampling transactions in database, the database D is subdivided into n partitions D_1, D_2, …, D_n.
- Partitioning may improve the performance by:
  - A large itemset must be large in at least one of the partitions.
  - We can adjust the size of each partition so that it is small enough to fit in main memory.

Partitioning

Algorithm

1. Split database D into n partitions
2. Using apriori algorithm to find set of large itemset of each partition, Let L^i denote set of large itemsets of partition i.
3. Candidate set $C = \cup n\ L^i$
4. Scan the original database, check the minimum support of each candidate c in C. If the criteria is met, add c into L.

Partitioning: Example

σ = 20%

A1	A2	A3	A4	A5	A6	A7	A8	A9
1	0	0	0	1	1	0	1	0
0	1	0	1	0	0	0	1	0
0	0	0	1	1	0	1	0	0
0	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0
0	1	1	1	0	0	0	0	0
0	1	0	0	0	1	1	0	1
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	0	0	0	1	1	0	1	0
0	1	0	1	0	1	1	0	0
1	0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	0	1

Partitioning: Example

Apriori:

L_1 =

{2}	6
{3}	6
{4}	4
{5}	8
{6}	5
{7}	7
{8}	4

L_2 =

{2,3}	3
{2,4}	3
{3,5}	3
{3,7}	3
{5,6}	3
{5,7}	5
{6,7}	3

L_3 =

{3,5,7}	3

The Frequent set L=$L_1 \cup L_2 \cup L_3$

Partitioning: Example

Dividing database in 3 equal partitions. Local support=20%=$\sigma_1=\sigma_2=\sigma_3=\sigma$

L^1= **{{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {1,5}, {1,6}, {1,8}, {2,3}, {2,4}, {2,8}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {5,7}, {6,7}, {6,8}, {1,6,8}, {1,5,6}, {1,5,8}, {2,4,8}, {4,5,7}, {5,6,8}, {5,6,7}, {1,5,6,8}**

L^2=**{……}** L^3=**{…….}**

The candidate set $C=L^1 \cup L^2 \cup L^3$

Read database once to compute the global support of the sets in C and get the final set of frequent itemsets L

Hash-Based Algorithm

- The larger the C_k the more processing cost required to discover FIs
- Reduces the size of C_k for $k>1$
- DHP or PCY has 2 major features
  - Efficient generation for FIs (2-itemsets)
  - Reduction of Tr. DB size (right after the generation of large 2-itemsets)

Hash-Based Algorithm

- Efficient counting
- For each Tr. After 1-itemsets are counted, 2-itemsets of the Tr. are generated and hashed into a hash table H_2
- *Subset function*: finds all the candidates contained in a transaction
- When a 2-itemset is hashed to a bucket, the count of the bucket is incremented

Hash-Based Algorithm: Example

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

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

L1*L1=({1,2},{1,3},{1,5},{2,3} {2,5},{3,5})

Hash-Based Algorithm: Example (generating C_2)

C_2

{1,3}

{2,3}

{2,5}

{3,5}

100	(1,3) (1,4), (3,4)
200	(2,3) (2,5), (3,5)
300	(1,2) (1,3), (1,5), (2,3), (2,5), (3,5)
400	(2,5)

H(x,y)= {(order of x)*10+ (order of y)} mod 7

Hash Table H_2

3,5				2,5		1,3
3,5		2,3		2,5		3,4
1,4	1,5	2,3		2,5	1,2	1,3

count	3	1	2	0	3	1	3
Bucket no	0	1	2	3	4	5	6

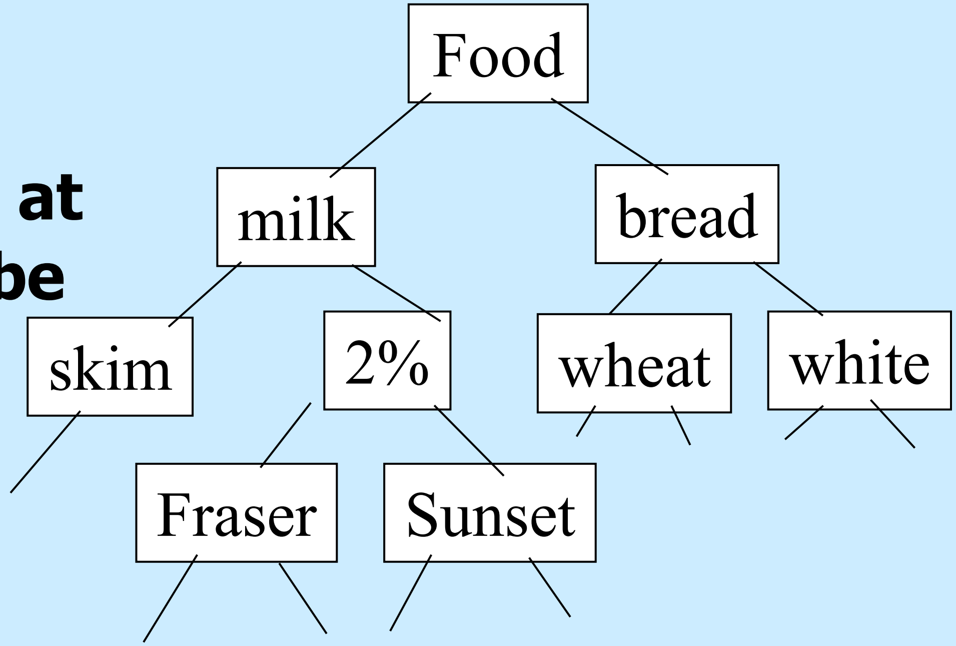
1 0 1 0 1 0 1 Bit Vector

L1*L1=({1,2},{1,3},{1,5},{2,3} {2,5},{3,5})

1 3 1 2 3 3 No. in the bucket with itemset

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.



milk $\Rightarrow$ bread [20%, 60%]

2% milk $\Rightarrow$ wheat bread [6%, 50%].

Multiple-Level Association Rules

mining multilevel association rules.

2% milk $\Rightarrow$ wheat bread

2% milk $\Rightarrow$ bread

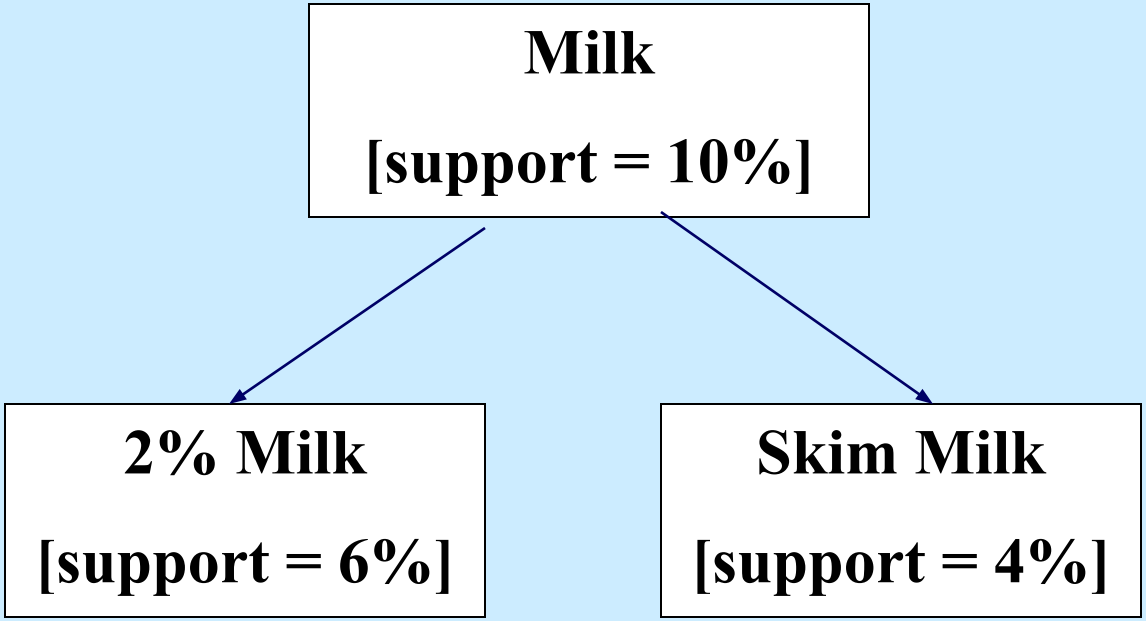
Multi-level Association: Uniform Support vs. Reduced Support

- 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 ⇒ miss low level associations**
    - **too low ⇒ generate too many high level associations**
- Reduced Support: reduced minimum support at lower levels

Uniform Support

Level 1
min_sup = 5%

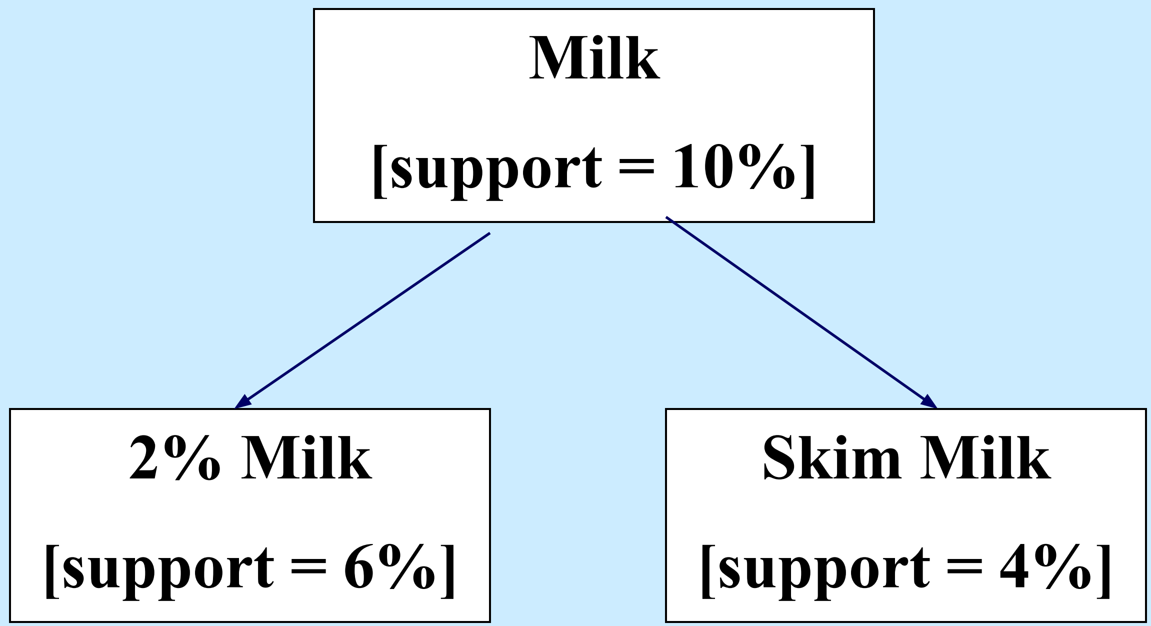
Level 2
min_sup = 5%



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.

Multi-Dimensional Association: Concepts

- **Single-dimensional rules:**

  buys(X, "milk") $\Rightarrow$ buys(X, "bread")

- **Multi-dimensional rules: ☐ 2 dimensions or predicates**
  - Inter-dimension association rules (no repeated predicates)

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

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

- **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: **{<u>age</u>, occupation, buys}** is a 3-predicate set.
  - Techniques can be categorized by how **age** are treated.

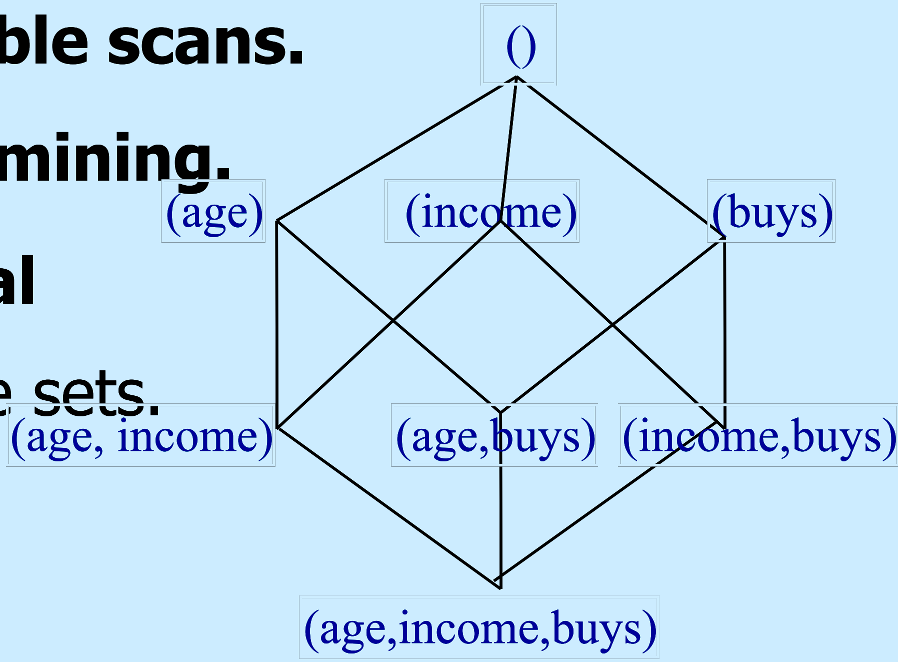
1. **Using static discretization of quantitative attributes**
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.
2. **Quantitative association rules**
   - Quantitative attributes are dynamically discretized into “bins”based on the distribution of the data.
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.

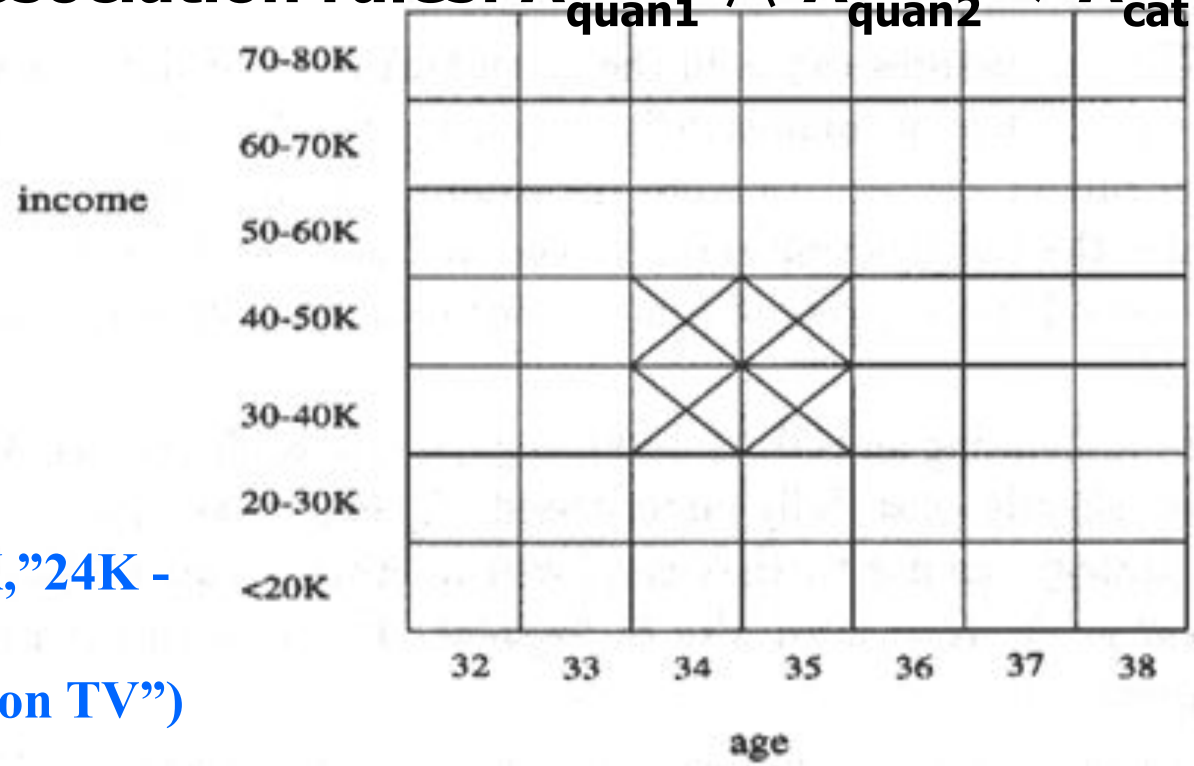
- Mining from data cubes can be much faster.



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,"34-35") $\wedge$ income(X,"24K - 48K")
$\Rightarrow$ buys(X,"high resolution TV")



Mining Distance-based Association Rules

- Binning methods do not capture the semantics of interval data

Price($)	Equi-width (width $10)	Equi-depth (depth 2)	Distance-based
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

- Distance-based partitioning, more meaningful discretization considering:
  - **density/number of points in an interval**
  - **“closeness” of points in an interval**