Association Rule Mining

- Kiran Bhowmick

Disclaimer: Slides taken from Han & Kamber, Dunham and Internet

Association rules

- Basic concepts and a road map
- Market Basket Analysis: A motivating example
- Apriori Algorithm
- FP Tree

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- E.g. bread and eggs, milk and bread
- A subsequence – PC, software, additional h/w (ext. hard disk)
- Why frequent patterns? – mining association rules, correlations and interesting relationships among data
- The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes such as catalog design, cross marketing and customer shopping behavior analysis.
- E.g., market basket analysis

Association rules

- Basic concepts and a road map
- Market Basket Analysis: A motivating example
- Apriori Algorithm
- FP Tree

Market Basket Analysis

- Analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets.
- Help to develop marketing strategies – how?
- Which groups or sets of items are likely to be purchased in this trip? - Plan marketing or advertising strategies, design new catalogs
- Design different store layouts
- Association rules
  - Computer $\Longrightarrow$ antivirus_s/w [support = 2%, confidence = 60%]

Association rule problem

- Given Itemset $X = \{x_1, \ldots, x_k\}$ and database of transactions $D = \{t_1, \ldots, t_n\}$
- To identify all association rules of the form $X \Rightarrow Y$ with a minimum support (s) & confidence (α). These (s, α) are given as input.
- support, *s*,
  - is percentage of transactions in the database that contain $X \cup Y$
  - probability that a transaction contains $P(X \cup Y)$

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

For example: Support of Item A = 3/5

Support of Item {B, E} = 2/5

Association rule problem

- For an association rule $X \Rightarrow Y$, confidence, α, is the ratio of number of transactions that contains $(X \cup Y)$ to the number of transactions that contain X
- conditional probability P(Y/X): The conditional probability is expressed in terms of itemset support count, where support count($A \cup B$) is the number of transactions containing the itemsets $A \cup B$, and support count(A) is the number of transactions containing the itemset A.

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

For example confidence:

$A \Rightarrow D = P(A \cup D)/P(A)$

$= 3/3 = 100\%$

$D \Rightarrow A = P(D \cup A)/P(D) = 3/4$
$= 0.75 = 75\%$

Association rule problem

- Approach
  - Find large itemsets/ frequent itemsets
  - Generate rules from frequent itemsets
- Large itemsets/frequent itemsets: an item set whose number of occurrences is above threshold s.
- Apriori algorithm: finding frequent itemsets using candidate generation

Association rules

- Basic concepts and a road map
- Market Basket Analysis: A motivating example
- Apriori Algorithm
- FP Tree

Apriori algorithm

- Uses prior knowledge of frequent itemset property
- Iterative approach
  - During scan i, candidates of size i, C_i are counted.
  - Only those items that are large L_i , are used to generate C_{i+1} next.
  - The iteration continues until no more frequent itemsets could be found.
  - From the set of frequent itemsets, generate strong association rules (satisfies min confidence).
- Apriori property: all nonempty subsets of frequent(large) itemsets should be frequent(large).

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan →

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan ←

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan →

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm—An Example

- Generating Association rules

  $l = \{B, C, E\}$ and $conf_{min} = 75\%$

Non-empty subsets : {B}, {C}, {E}, {B, C}, {B, E}, {C, E}

$B \Rightarrow C \wedge E = 2/3 = 66\%$

$C \Rightarrow B \wedge E = 2/3 = 66\%$

$E \Rightarrow B \wedge C = 2/3 = 66\%$

$C \wedge E \Rightarrow B = 2/2 = 100\%$

$B \wedge E \Rightarrow C = 2/3 = 66\%$

$B \wedge C \Rightarrow E = 2/2 = 100\%$

ONLY 4 AND 6 ARE OUTPUT AS THESE ARE ONLY THE STRONG RULES.

Find frequent patterns using Apriori

S = 20% and confidence = 75%

Transaction	Items
t_1	Bread, Jelly, PeanutButter
t_2	Bread, PeanutButter
t_3	Bread, Milk, PeanutButter
t_4	Beer, Bread
t_5	Beer, Milk

Scan	Candidate	Large Itemsets
1	Br, J, PB, M, B	Br, J, PB, M, B
2	{Br, J}, {Br, PB}, {Br, M}, {Br, B}, {J, PB}, {J, M}, {J, B}, {PB, M}, {PB, B}, {M, B}	{Br, J}, {Br, PB}, {Br, M}, {Br, B}, {J, PB}, {PB, M}, {M, B}
3	{Br, J, PB}, {Br, J, M}, {Br, J, B}, {Br, PB, M}, {Br, PB, B}, {Br, M, B}, {J, PB, M},	{Br, J, PB}, {Br, PB, M}
4	{Br, J, PB, M}	ϕ

Transaction	Items
t_1	Bread, Jelly, PeanutButter
t_2	Bread, PeanutButter
t_3	Bread, Milk, PeanutButter
t_4	Beer, Bread
t_5	Beer, Milk

L1 = {Br, J, PB}, L2 = {Br, PB, M}

L1: Non-empty subsets: {Br}, {J}, {PB}, {Br, J}, {Br, PB}, {J, PB}

Br => {J,PB} ---- P(Br U J U PB)/P(Br) = 1/4 = 25%
J => {Br, PB} = 100%
PB => {J, Br} = 33%
{J,PB} => Br = 100%
{Br, PB} => J = 33%
{J, Br} => PB = 100%

L2: Non-empty subsets: {Br}, {M}, {PB}, {Br, M}, {Br, PB}, {M, PB}

Br => {M,PB} ---- P(Br U M U PB)/P(Br) = 1/4 = 25%
M => {Br, PB} = 50%
PB => {M, Br} = 33%
{M,PB} => Br = 100%
{Br, PB} => M = 33%
{M, Br} => PB = 100%

Improving Efficiency of the Apriori algorithm

- Hash-based technique
- Transaction reduction
- Partitioning
- Sampling

Direct Hashing with Efficient Pruning

- A hash-based technique can be used to reduce the size of the candidate k-itemsets, C_k, $k > 1$.
- When scanning each transaction to generate the L1 from C1, we can generate all of the 2-itemsets for each transaction, hash them into different buckets of hash table and increase the corresponding bucket counts.
- A 2-itemset whose corresponding bucket count is below support threshold cannot be frequent and should be removed from candidate set.

Database D

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

$Sup_{min} = 2$

Itemset	sup	L1
{A}	2	{A}
{B}	3	{B}
{C}	3	{C}
{D}	1	-
{E}	3	{E}

TID	Subsets
100	{A, C}, {A, D}, {C, D}
200	{B, C}, {B, E}, {C, E}
300	{A, B}, {A, C}, {A, E}, {B, C}, {B, E}, {C, E}
400	{B, E}

h(x, y) = (order of x*10 + order of y) mod 7

{C, E} {C, E} {A, D}	{A, E}	{B, C} {B, C}		{B, E} {B, E} {B, E}	{A, B}	{A, C} {C, D} {A, C}
3	1	2	0	3	1	3
0	1	2	3	4	5	6

Generating C2 from hash bucket and L1

L1 x L1	# in a bucket
{A, B}	1
{A, C}	3
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	3

C2
{A, C}
{B, C}
{B, E}
{C, E}

C3 = {B, C, E}

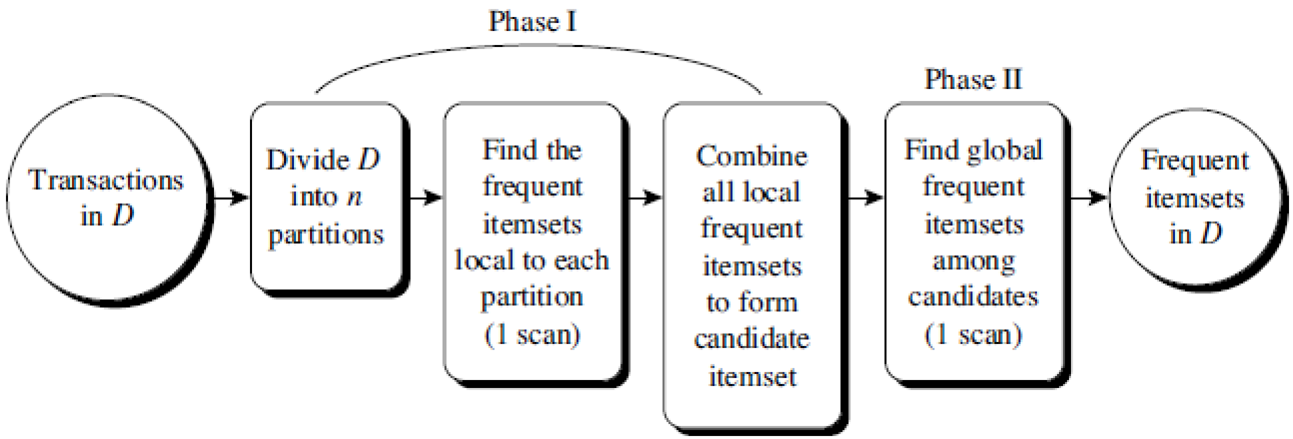
Transaction reduction

- Reducing the number of transactions scanned in future iterations
- A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k+1)-itemsets.
- Such transactions can be marked or removed from further consideration.

Partitioning: Scan Database Only Twice

- In one scan it generates all potentially large itemsets by scanning d/b once. This set is the superset of actual large set. It may contain false positives but no false negatives will be found.
- During the second scan counters for these itemsets are set and their actual support is measured.
- The algorithm is executed in 2 phases.
- 1st phase: the algorithm logically divides the d/b into non-overlapping partition. The partitions are considered one at a time and large itemsets in each are generated. At the end of this phase, all large itemsets are then merged to generate a set of potentially large itemsets.
- 2nd phase: the actual support for these itemsets are generated and large itemsets are identified.

Partitioning: Scan Database Only Twice



- The algorithm is executed in 2 phases.
- 1st phase: the algorithm logically divides the d/b into non-overlapping partition. The partitions are considered one at a time and large itemsets in each are generated. At the end of this phase, all large itemsets are then merged to generate a set of potentially large itemsets.
- 2nd phase: the actual support for these itemsets are generated and large itemsets are identified.

Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Samples are small and can fit in main memory.
- Sampling is trade-off between accuracy and efficiency since only large sets within samples are found and those that are not in the sample may not be found.
- Pick a random sample S of the given data D, and search for frequent itemsets in S. Lower the support to ensure that all itemsets that are large in S are found. The rest of the d/b is used to find the actual frequencies of each itemsets in S. If S contains all the frequent itemsets, then only scan of d/b is reqd or else the d/b is scanned one more time to find the missing frequent itemset.

Association rules

- Basic concepts and a road map
- Market Basket Analysis: A motivating example
- Apriori Algorithm
- FP Tree

Mining Frequent Patterns Without Candidate Generation

- Advantages of Apriori
  - Significantly reduces size of candidate sets
  - Good performance gain
- Problems of Apriori
  - Need to generate a huge number of candidate sets
  - Need to repeatedly scan the database and check a large set of candidates by pattern matching
- Frequent-pattern growth method – adopts divide and conquer strategy
  - It compresses the d/b into frequent-pattern tree or FP-tree which retains the itemsets association information
  - Divides the compressed d/b into a set of conditional databases, each associated with one frequent item or pattern fragment and mines such d/b separately

Mining frequent patterns with FP-tree

- Start from each frequent length-1 pattern as an initial suffix pattern
- Construct its conditional pattern base which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern
- Construct its FP-tree and perform mining recursively on such a tree
- The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from the conditional FP-tree

Min support = 2

TID	Itemsets
100	I1, I2, I5
200	I2, I4
300	I2, I3
400	I1, I2, I4
500	I1, I3
600	I2, I3
700	I1, I3
800	I1, I2, I3, I5
900	I1, I2, I3

I1	6
I2	7
I3	6
I4	2
I5	2

L = {{I2:7}, {I1:6}, {I3:6}, {I4:2}, {I5:2}}

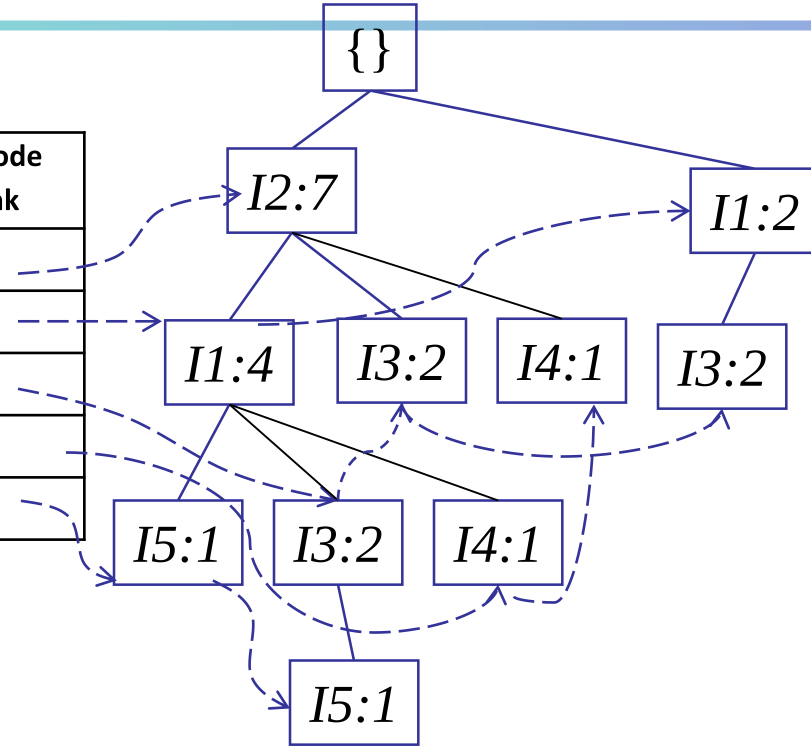
Arrange transactions in order of L

TID	Itemsets
100	I2, I1, I5
200	I2, I4
300	I2, I3
400	I2, I1, I4
500	I1, I3
600	I2, I3
700	I1, I3
800	I2, I1, I3, I5
900	I2, I1, I3

TID	Itemsets
100	I2, I1, I5
200	I2, I4
300	I2, I3
400	I2, I1, I4
500	I1, I3
600	I2, I3
700	I1, I3
800	I2, I1, I3, I5
900	I2, I1, I3

Sample

Item ID	Support Count	Node link
I2	7	
I1	6	
I3	6	
I4	2	
I5	2	



Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I5	{{I2, I1:1}, {I2, I1, I3:1}}	<I2:2, I1:2>	{I2, I5}, {I1, I5}, {I2, I1, I5}
I4	{{I2, I1:1}, {I2:1}}	<I2:2>	{I2, I4}
I3	{{I2, I1:2}, {I2:2}, {I1:2}}	<I2:4, I1:2>, <I1:2>	{I2, I3}, {I1, I3}, {I2, I1, I3}
I1	{I2:4}	<I2:4>	{I2, I1}

Construct FP-tree from a Transaction Database

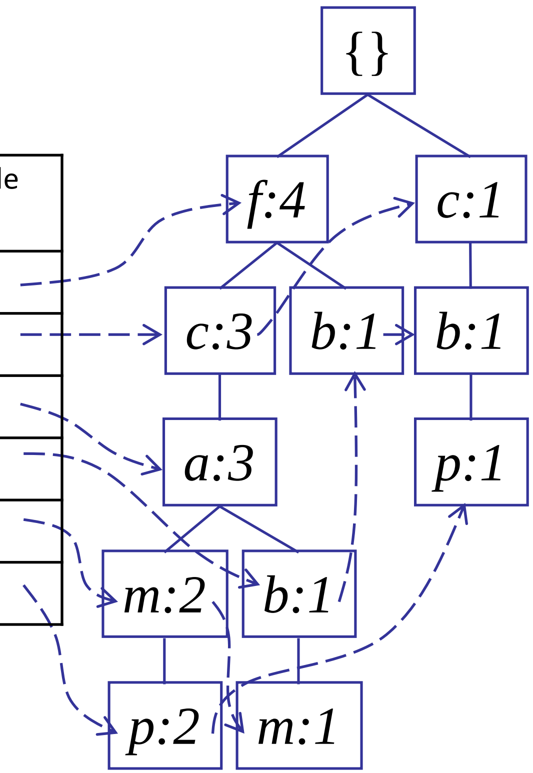
TID	*Items bought*	*(ordered) frequent items*
100	{*f*, *a*, *c*, *d*, *g*, *i*, *m*, *p*}	{*f*, *c*, *a*, *m*, *p*}
200	{*a*, *b*, *c*, *f*, *l*, *m*, *o*}	{*f*, *c*, *a*, *b*, *m*}
300	{*b*, *f*, *h*, *j*, *o*, *w*}	{*f*, *b*}
400	{*b*, *c*, *k*, *s*, *p*}	{*c*, *b*, *p*}
500	{*a*, *f*, *c*, *e*, *l*, *p*, *m*, *n*}	{*f*, *c*, *a*, *m*, *p*}

***min_support* = 3**

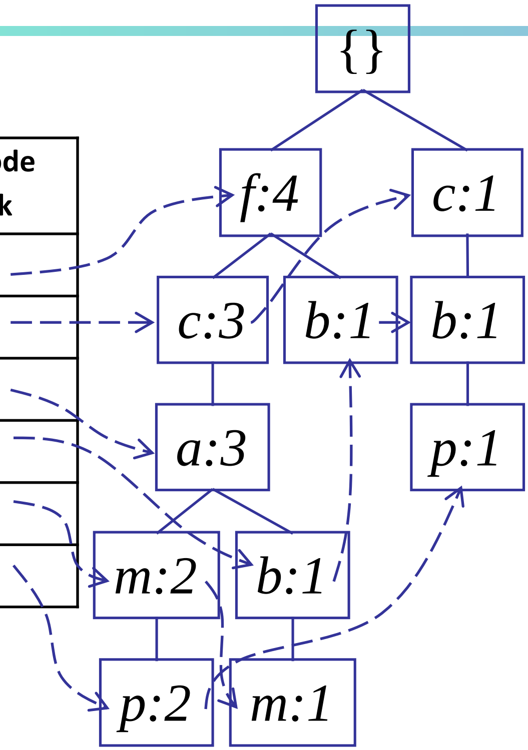
L={{f:4}, {c:4}, {a:3}, {b:3}, {m:3}, {p:3}}

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree
4. To facilitate tree traversal, an item header table is built and each item points to its occurrences via a chain of node-links

Item ID	Support Count	Node link
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Item ID	Support Count	Node link
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



TID	*Items bought*	*(ordered) frequent item*
100	{*f*, *a*, *c*, *d*, *g*, *i*, *m*, *p*}	{*f*, *c*, *a*, *m*, *p*}
200	{*a*, *b*, *c*, *f*, *l*, *m*, *o*}	{*f*, *c*, *a*, *b*, *m*}
300	{*b*, *f*, *h*, *j*, *o*, *w*}	{*f*, *b*}
400	{*b*, *c*, *k*, *s*, *p*}	{*c*, *b*, *p*}
500	{*a*, *f*, *c*, *e*, *l*, *p*, *m*, *n*}	{*f*, *c*, *a*, *m*, *p*}

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
p	{{f, c, a, m:2}, {c, b:1}}	<c:3>	{c, p}
m	{{f, c, a:2}, {f, c, a, b:1}}	<f:3, c:3, a:3>	{f, m}, {c, m}, {a, m}, {f, c, m}, {f, a, m}, {c, a, m}, {f, c, a, m}
b	{{f, c, a:1}, {f:1}, {c:1}}	--	--
a	{{f, c:3}}	<f:3, c:3>	{f, a}, {c, a}, {f, c, a}
c	{{f:3}}	<f:3>	{f, c}

A grocery store chain keeps a record of weekly transactions where each transaction represents the items bought during one cash register transaction.

Transaction	Items	Transaction	Items
t_1	Blouse	t_{11}	TShirt
t_2	Shoes, Skirt, TShirt	t_{12}	Blouse, Jeans, Shoes, Skirt, TShirt
t_3	Jeans, TShirt	t_{13}	Jeans, Shoes, Shorts, TShirt
t_4	Jeans, Shoes, TShirt	t_{14}	Shoes, Skirt, TShirt
t_5	Jeans, Shorts	t_{15}	Jeans, TShirt
t_6	Shoes, TShirt	t_{16}	Skirt, TShirt
t_7	Jeans, Skirt	t_{17}	Blouse, Jeans, Skirt
t_8	Jeans, Shoes, Shorts, TShirt	t_{18}	Jeans, Shoes, Shorts, TShirt
t_9	Jeans	t_{19}	Jeans
t_{10}	Jeans, Shoes, TShirt	t_{20}	Jeans, Shoes, Shorts, TShirt

Using minsup = 20%, minconf = 75%, generate frequent itemsets as well as association rules

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - For Connect-4 DB, compression ratio could be over 100

Mining frequent itemsets using vertical data format

- Horizontal format – {TID:itemset}

Table 6.1 Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Vertical format – {itemset:TID}

Table 6.3 The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

Mining frequent itemsets using vertical data format

- Transform into vertical format by scanning the dataset once
- Support count of an itemset is the length of the TID_set
- Start with k=1, the frequent k-itemsets are used to construct (k+1) itemsets based on Apriori property
- Computation is done by intersection of the TID_sets of frequent k-itemsets to compute TID_sets of frequent (k+1)-itemsets

Table 6.3 The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

Table 6.4 2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

Table 6.5 3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Mining Multiple-Level Association Rules

- Developing effective methods for mining patterns at multiple abstraction levels
- Concept hierarchy defines sequence of mappings from a set of low-level concepts to a higher-level more general concept set
- Items often form hierarchies
- Association rules generated from multiple levels of abstraction are called multiple-level or multilevel association rules.

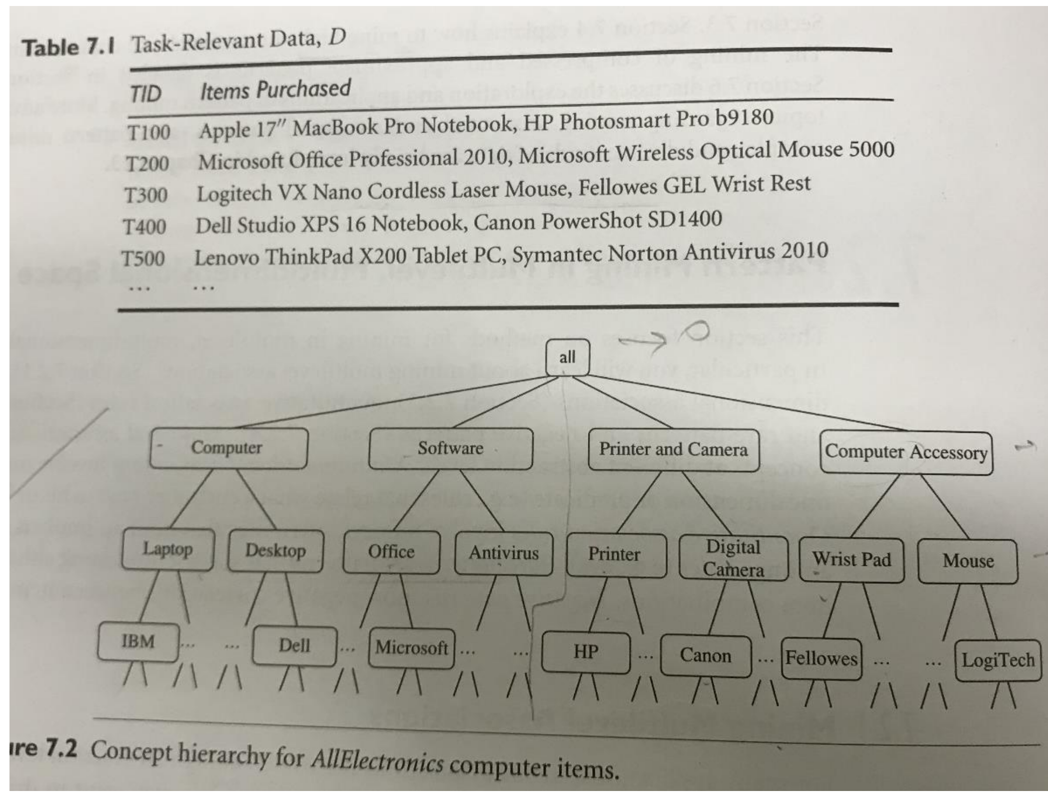


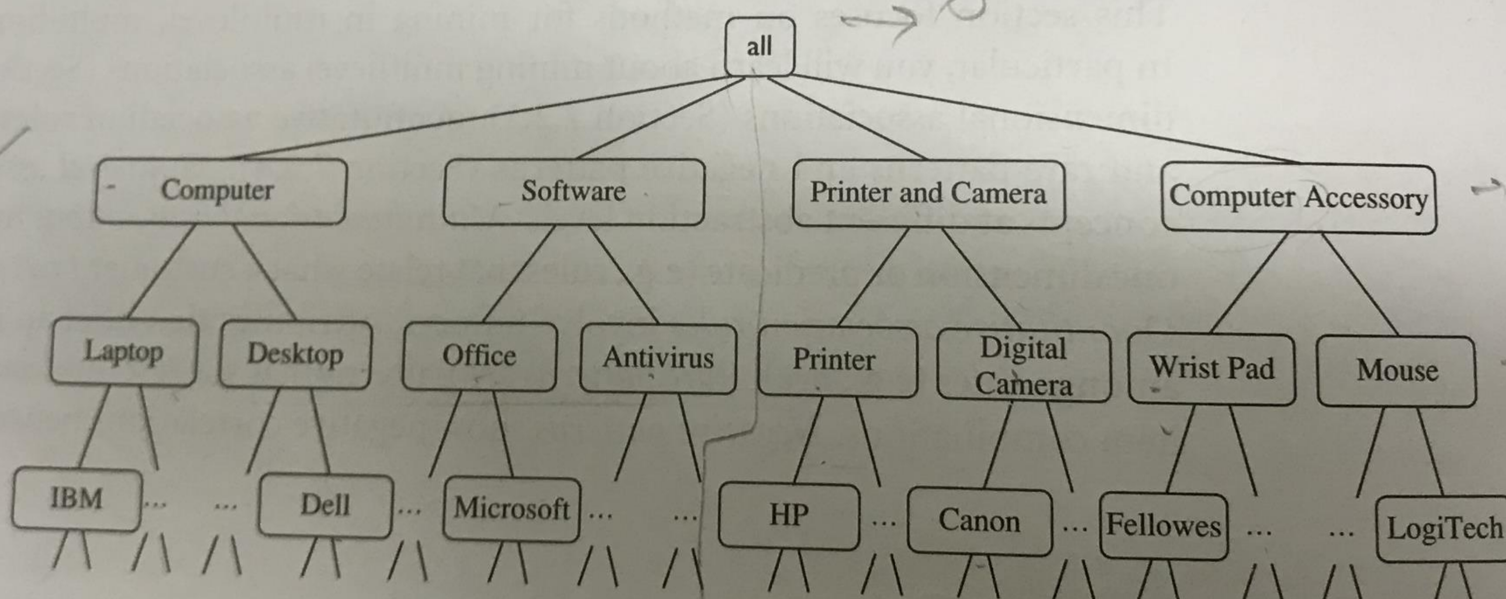
Table 7.1 Task-Relevant Data, *D*

TID	*Items Purchased*
T100	Apple 17″ MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
…	…

ure 7.2 Concept hierarchy for *AllElectronics* computer items.

Table 7.1 Task-Relevant Data, *D*

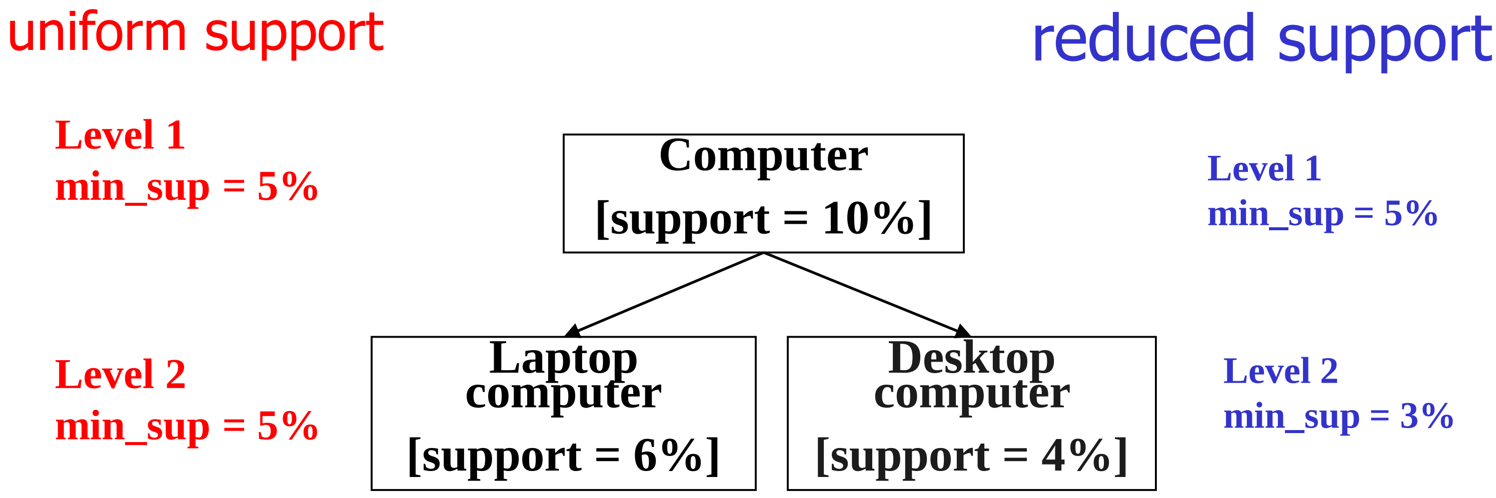
TID	*Items Purchased*
T100	Apple 17″ MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
…	…



ure 7.2 Concept hierarchy for *AllElectronics* computer items.

Mining Multiple-Level Association Rules

- Uses concept hierarchies under support-confidence framework.
- For each level Apriori or variations are used
  - Using uniform minimum support for all levels
  - Using reduced minimum support at lower levels
  - Using item or group-based minimum support



Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - buys(X, “laptop computer”) $\Rightarrow$ buys(X, “HP printer”) [support = 8%, confidence = 70%]
  - buys(X, “IBM laptop”) $\Rightarrow$ buys(X, “HP printer”) [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.

Mining Multi-Dimensional Association

- Single-dimensional rules or intra-dimensional association rules:

  buys(X, "milk") ⇒ buys(X, "bread")
- Multi-dimensional rules: ≥ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    age(X, "19-25") ∧ occupation(X, "student") ⇒ buys(X, "coke")
  - hybrid-dimension assoc. rules (*repeated predicates*)

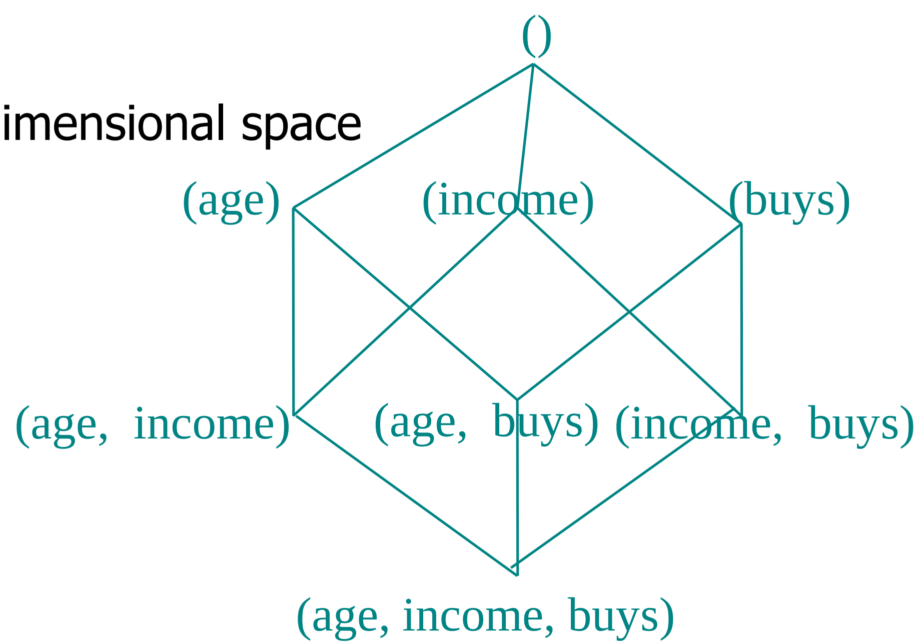
    age(X, "19-25") ∧ buys(X, "popcorn") ⇒ buys(X, "coke")
- Categorical Attributes:
  - finite number of possible values, no ordering among values
  - E.g. color, occupation
- Quantitative Attributes:
  - numeric, implicit ordering among values
  - E.g. income, age, price
- Techniques for mining multidimensional association rules

Techniques for mining multidimensional association rules

- Discretization before mining
  - Use predefined concept hierarchies
  - Static and predetermined
  - “0—20K”, ”21K…30K”, ……
  - Mining multidimensional association rules using static discretization of quantitative attributes
- Discretization based on distribution
  - Use bins based on distribution of data
  - Dynamic and established to satisfy a particular mining criteria
  - Mining quantitative association rules

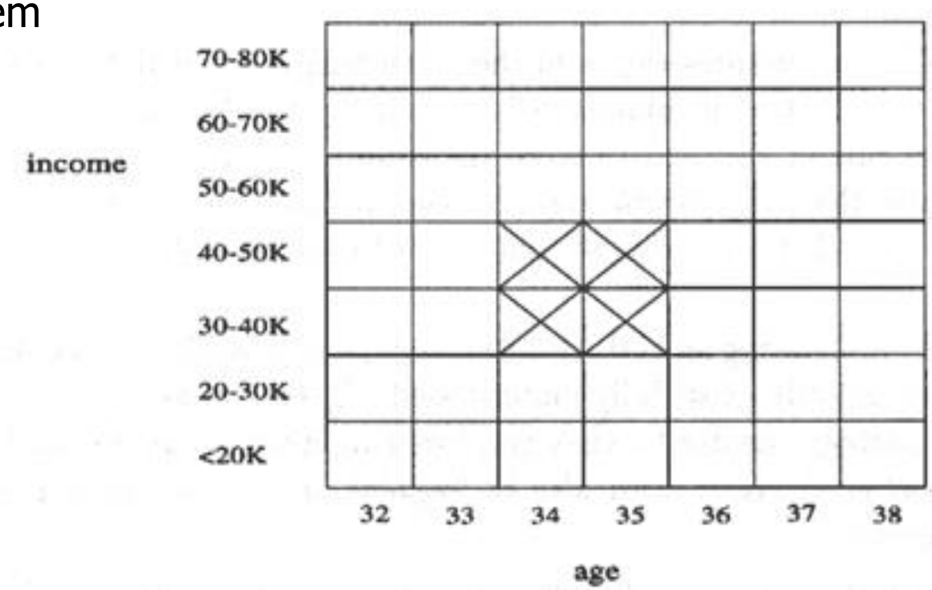
Mining Multidimensional Association rules using Static Discretization of Quantitative Attributes

- Quantitative attributes discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent predicate sets instead of frequent itemsets. E.g instead of searching one attribute *buys* one can search through all of the relevant attributes in the form of attribute-value pair
- Data cube is well suited for mining.
- The cube stores aggregates in multidimensional space
- The cells of an n-dimensional cuboid correspond to the support count of corresponding n-predicate sets.
- Mining from data cubes can be much faster.



Mining Quantitative Association Rules

- Quantitative Association Rules are Multidimensional association rules where numeric attributes are dynamically discretized during mining process to satisfy some mining criteria
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
  e.g. age(X, "30…39") Λ income(X, "42K…48K") $\Rightarrow$ buys(X, "HDTV") …. ?
- ARCS: Association Rules Clustering System
  - Binning
    - Equal width binning
    - Equal-frequency binning
    - Clustering-based binning
  - Finding frequent predicate sets
  - Clustering the association rules
- Cluster *adjacent* association rules to form general rules using a 2-D grid



- Example
- age(X, "34-35") $\wedge$ income(X, "30-50K") $\Rightarrow$ buys(X, "high resolution TV")