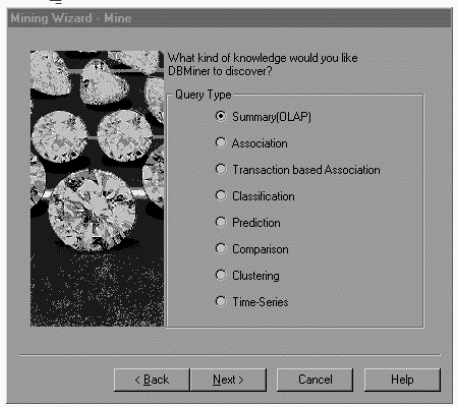
Mining Association Rules

Data Mining

CSE 6331 / CSE 6362

Fall 1999

Selecting a Data Mining Task

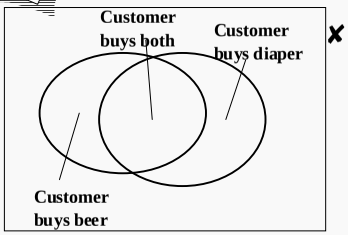


- **Major data mining functions:**
  - **Summary (Characterization)**
  - **Association**
  - **Classification**
  - **Prediction**
  - **Clustering**
  - **Time-Series Analysis**

Mining Association Rules

- Assocation rule mining:
  - Finding associations or correlations among a set of items or objects in transaction databases, relational databases, and data warehouses.
- Applications:
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, etc.
- Examples:
  - Rule form: "Body → Head [support, confidence]".
  - Buys=Diapers → Buys=Beer [0.5%, 60%]
  - Major=CS ^ Class=DataMining → Grade=A [1%, 75%]

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, Y, Z}
  - confidence, *c*, conditional probability that a transaction having {X, Y} also contains *Z*.

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

For minimum support 50%, minimum confidence 50%:

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

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

Association Rule

- Given
  - Set of items $I = \{i_1, i_2, ..., i_m\}$
  - Set of transactions D
  - Each transaction T in D is a set of items s.t. $T \subseteq I$
- An association rule is an implication $X \Rightarrow Y$
  - X and Y are itemsets, $X \subset I, Y \subset I, X \cap Y = \Phi$
  - Rule meets minimum confidence c
    (c% of transactions in D which contain X contain Y)
  - A minimum support s is also met

    $c \leq |X \cup Y| / |X|$

    $s \leq |X \cup Y| / |D|$

Mining Strong Association Rules in Transaction DBs

- Measurement of rule strength in a transaction DB.

  $A \rightarrow B$ *[support, confidence]*

  $support = Prob(A \cup B) = \frac{\#\_of\_trans\_containing\_all\_the\_items\_in\ A \cup B}{total\_\#\_of\_trans}$

  $confidence = Prob(B|A) = \frac{\#\_of\_trans\_that\_contain\_both\ A\ and\ B}{\#\_of\_trans\_containing\ A}$

- We are often interested in only strong associations, i.e.

  $support \geq min\_sup$ and $confidence \geq min\_conf$.
- Examples.

  milk $\rightarrow$ bread [5%, 60%].

  tire $\wedge$ auto_accessories $\rightarrow$ auto_services [2%, 80%].

Rule Evaluation

- Confidence
- Support
- Significance
- Simplicity
- Novelty
- Application dependent
  - What is useful is often application dependent
  - Need domain expert

Methods for Mining Associations

- Apriori (AprioriTid, AprioriHybrid):
  (Agrawal & Srikant, Mannila et al)
- Partition Technique:
  (Savasere, Omiecinski, Navathe)
- Sampling technique
  (Toivonen)
- Anti-Skew
  (Dunham)
- Multi-level or generalized association
  (Agrawal & Srikant, Han & Fu)
- Constraint-based or query-based association
  (Ng et al, Tsur et al)

Apriori (Levelwise)

- Scan database multiple times
- For the ith scan, find all large itemsets of size i with min support
- Use the large itemsets from scan i as input to scan i+1
- Generate candidates by creating subsets of size i+1 which contain only large itemsets as subsets
- Notation: Large k-itemset, L_k
  Set of candidate large itemsets of size k, C_k
- Note: If $\{A,B\}$ is not a large itemset, then no superset of it can be either.
- Hash tree used to store candidate itemsets (optimizes subset testing)

Mining Association Rules -- 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($\{A, C\}$) = 50%

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

Apriori principle:

Any subset of a frequent itemset must be frequent.

Pseudocode

- First pass: count item occurrences for large 1-itemsets L_1

```
Apriori()
  for (k=2; L_{k-1} != NULL; k++)
    C_k = AprioriGen(L_{k-1})
    forall transactions t ∈ D
      C_t = subset(C_k, t)
      forall candidates c ∈ C_t
        c.count++
    L_k = {c ∈ Ck | c.count >= minsup}
```

```
AprioriGen(L)                    /* Assume transactions in lexicographic order */
  insert into C_k all p.item_1, p.item_2, …, p.item_{k-1}, q.item_{k-1} from p, q ∈ L
  where p.item1=q.item1, p.item2=q.item2, …, p.itemk-1 < q.itemk-1
                                 /* prune itemsets s.t. some (k-1)-subset of c is ∉ L */
  forall itemsets c ∈ C_k
    forall (k-1)-subsets s of c do
      if (s ∉ L) then
        delete c from C_k
```

Pseudocode (continued)

```
AssocRules()
  forall large itemsets l_k, k>=2
    GenRules(l_k, l_k)
```

```
GenRules(lk, am)                 /* Generate all valid rules a => (l_k - a), for all a ⊂ am */
  A = {(m-1)-itemsets a_{m-1} | a_{m-1} ⊂ a_m}
  forall a_{m-1} ∈ A
    conf = support(l_k) / support(a_{m-1})
    if (conf >= minconf) then
      output the rule a_{m-1} => (l_k - a_{m-1}) with confidence=conf and support=support(l_k)
      if (m-1 > 1) then
        GenRules(l_k, a_{m-1})   /* Generate rules with subsets of a_{m-1} as antecedents */
```

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

Minsup = 0.25, Minconf = 0.5

L1 = {(A, 3), (B, 2), (C, 2), (D, 1), (E, 1), (F, 1)}

C2 = {(A,B), (A,C), (A,D), (A,E), (A,F), (B,C), …, (E,F)}

Pruned C2 = C2

L2 = {(A,B, 1), (A,C, 2), (A,D, 1), (B,C, 1), (B,E, 1), (B,F, 1), (E,F, 1)}

C3 = {(A,B,C), (A,B,D), (A,C,D), (B,C,E), (B,C,F), (B,E,F)}, Pruned C3 = {(A,B,C), (B,E,F)}

L3 = {(A,B,C, 1), (B,E,F, 1)}

C4 = {}, L4 = {}, End of program

Possible Rules

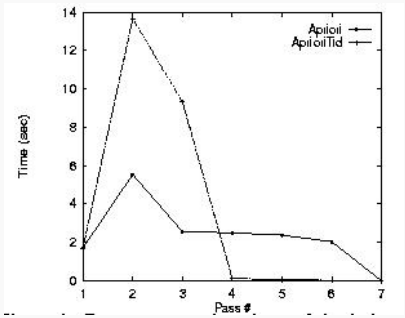
A=>B (c=.33,s=1), B=>A (c=.5,s=1), A=>C (c=.67,s=2), C=>A (c=1.0,s=2)
A=>D (c=.33,s=1), D=>A (c=1.0,s=1), B=>C (c=.5,s=1), C=>B (.5,s=1),
B=>E (c=.5,s=1), E=>B(c=1,s=1), B=>F (c=.5,s=1), F=>B(c=1,s=1)

A=>B&C (c=.33,s=1), B=>A&C (c=.5,s=1), C=>A&B (c=.5,s=1), A&B=>C(c=1,s=1), A&C=>B (c=.5,s=1), B&C=>A (c=1,s=1), B=>E&F (c=.5,s=1), E=>B&F(c=1,s=1), F=>B&E (c=1,s=1), B&E=>F (c=1,s=1), B&F=>E(c=1,s=1), E&F=>B (c=1,s=1)

AprioriTid & AprioriHybrid

- AprioriTid
  - Like Apriori, but doesn't have to scan entire database after first scan
  - Encode *instances* of itemsets with Tid, check if itemsets are contained in transaction coding
  - If transaction does not contain itemset, remove transaction from consideration for later passes
- AprioriHybrid
  - In initial passes encoding can be larger than database
  - Use Apriori in initial passes until encodings will fit in memory

Comparison



Example

Knowledge Stream Partners

Insurance:
Data Mining Example

Using data mining tools, one insurance company found a previously unknown pattern:

people with good credit ratings have fewer accidents

Source: Washington Technology Online, wtonline.com January 97

(C) 1998 Knowledge Stream Partners 38

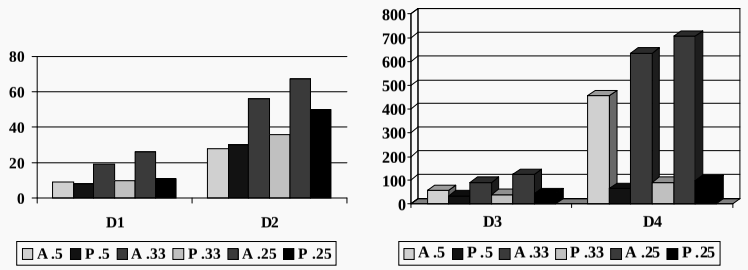
Partitioning

- Requires only two passes through external database
- Divide database into *n* partitions, each fits in main memory
- Scan 1: Process one partition in memory at a time, finding local large itemsets
- Candidate large itemsets are the union of all local large itemsets (superset of actual large itemsets, contains false +)
- Scan 2: Calculate support, determine actual large itemsets
- If data is skewed, partitioning may not work well. The chance that a local large itemset is a global large itemset may be small.

Partitioning

- Will any large itemsets be missed?
- If $l \notin Li$, then
  $t1(l)/t1 < MS \ \& \ t2(l)/t2 < MS \ \& \ \ldots \ \& \ tn(l)/tn < MS$, and
  $t1(l) < MS*t1 \ \& \ t2(l) < MS*t2 \ \& \ \ldots \ \& \ tn(l) < MS*tn$, thus
  $t1(l) + t2(l) + \ldots + tn(l) < MS * (t1 + t2 + \ldots + tn)$
- $t1 + t2 + \ldots + tn$ is total #transactions, $t1(l) + t2(l) + \ldots + tn(l)$ is #transactions containing l, so support(l) < MS and $l \notin L$

How do run times compare?



Sampling

- Use a sampling process to generate candidate sets
- Randomly select subset of transactions to consider
- Negative Border, Bd-(S) : Set of minimal itemset sets X where X is not in S
- Scan D using S union Bd-(S) as candidate set
- Expand by adding negative border recursively until negative border empty
- Worse case - 2 passes, Best case - 1 pass
- First scan may be very inefficient
- Small non-zero probability of missing itemset, for complete accuracy use entire database
- With Partitioning, may view 1st partition as sample

Anti-Skew Counting Partition

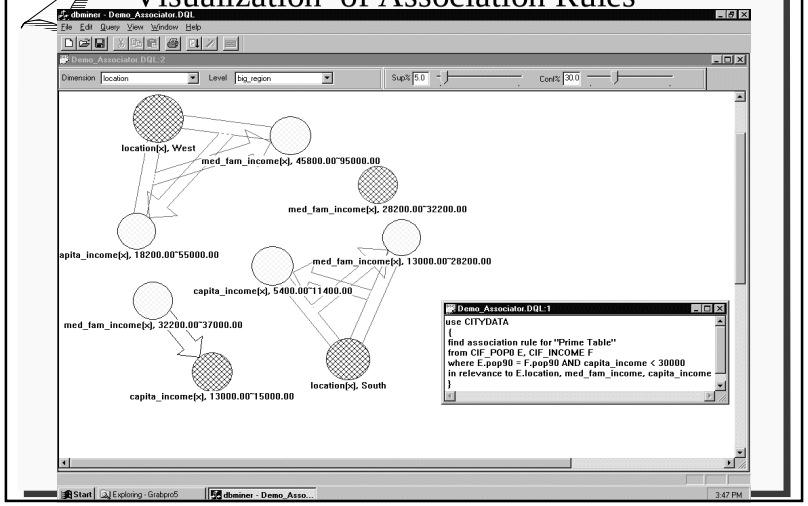
- ✘ AS-CPA - Use techniques to improve partitioning by reducing number of false candidate
- ✘ Early local pruning: Candidates must be not only local large 1-itemsets, but must also be large 1-itemsets in the union of all partitions examined so far (Note: any large itemset must be a large itemset in ANY partition)
- ✘ First Global Anti-Skew: Remove possibly false global candidate itemsets during first scan
- ✘ Second Global Anti-Skew: Removes false global candidates during second pass

Mining Association Rules (Table Form)

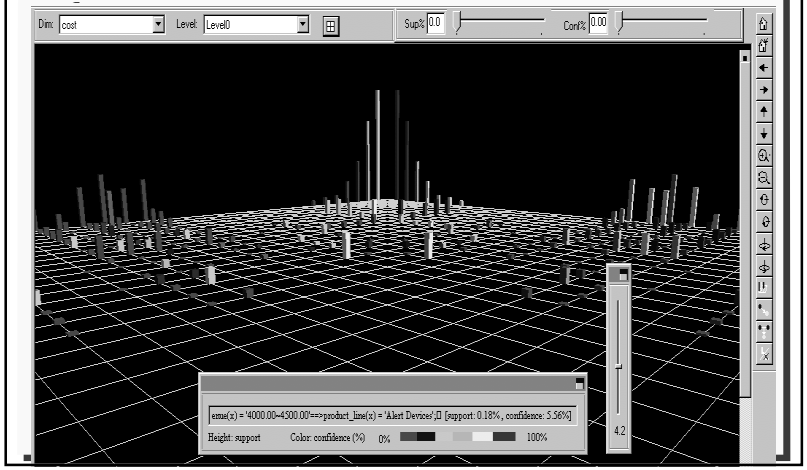
	Body	Implies	Head	Supp (%)	Conf (%)	F	G	H	I
1	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00'	28.45	40.4				
2	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00'	20.46	29.05				
3	cost(x) = '0.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	59.17	84.04				
4	cost(x) = '0.00~1000.00'	==>	revenue(x) = '1000.00~1500.00'	10.45	14.84				
5	cost(x) = '0.00~1000.00'	==>	region(x) = 'United States'	22.56	32.04				
6	cost(x) = '1000.00~2000.00'	==>	order_qty(x) = '0.00~100.00'	12.91	69.34				
7	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '0.00~500.00'	28.45	34.54				
8	order_qty(x) = '0.00~100.00'	==>	cost(x) = '1000.00~2000.00'	12.91	15.67				
9	order_qty(x) = '0.00~100.00'	==>	region(x) = 'United States'	25.9	31.45				
10	order_qty(x) = '0.00~100.00'	==>	cost(x) = '0.00~1000.00'	59.17	71.86				
11	order_qty(x) = '0.00~100.00'	==>	product_line(x) = 'Tents'	13.52	16.42				
12	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	23.88				
13	product_line(x) = 'Tents'	==>	order_qty(x) = '0.00~100.00'	13.52	98.72				
14	region(x) = 'United States'	==>	order_qty(x) = '0.00~100.00'	25.9	81.94				
15	region(x) = 'United States'	==>	cost(x) = '0.00~1000.00'	22.56	71.39				
16	revenue(x) = '0.00~500.00'	==>	cost(x) = '0.00~1000.00'	28.45	100				
17	revenue(x) = '0.00~500.00'	==>	order_qty(x) = '0.00~100.00'	28.45	100				
18	revenue(x) = '1000.00~1500.00'	==>	cost(x) = '0.00~1000.00'	10.45	96.75				
19	revenue(x) = '500.00~1000.00'	==>	cost(x) = '0.00~1000.00'	20.46	100				
20	revenue(x) = '500.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	19.67	96.14				
21									
22									
23	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4				
24	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4				
25	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93				
26	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93				
27	cost(x) = '0.00~1000.00' AND order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	33.23				

Sheet1

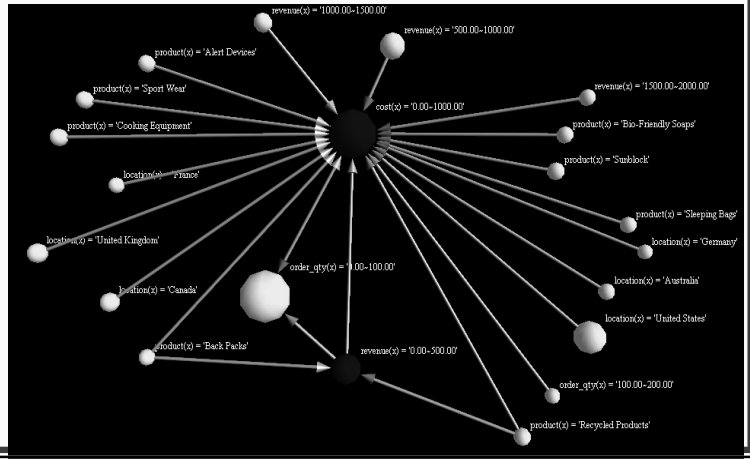
Visualization of Association Rules



Association Rule in Plane Form

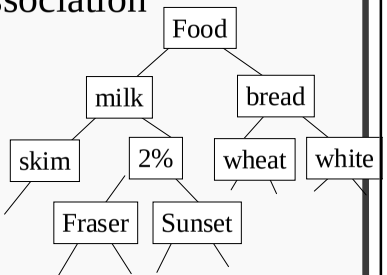


Association Rule Graph



Multi-Level Association

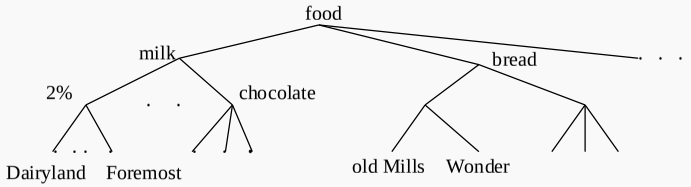
- A descendant of an infrequent itemset cannot be frequent
- A transaction database can be encoded by dimensions and levels



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}

Encoding Hierarchical Information in Transaction Database

- A taxonomy for the relevant data items



- Conversion of bar_code into generalized_item_id.

GID	bar_code_set	category	content_spec	brand
112	{17325, 34823, 91265}	milk	2%	Foremost
141	{29394, 77454, 89157}	milk	skim	Dairy_land
171	{73295, 99184, 79520}	milk	chocolate	Dairy_land
212	{88452, 35672, 98427, 31205}	bread	wheat	Wonder
...	{..., ...}	...	...	...
711	{32514, 78152}	fruit_juice	orange	Minute_maid

Variation of Apriori Algorithm

- Basic, Cumulate, EstMerge
- Add ancestors of each item in t to t, removing duplicates
- Increment count of candidates that are contained in t
- Delete candidates consisting solely of item and its ancestor
- Delete ancestors not present in any candidates
- At each level, delete descendants of items that do not have minimum support

Interestingness

Rule #	Rule	Support
1	"Clothes ⇒ Footwear"	10
2	"Outerwear ⇒ Footwear"	8
3	"Jackets ⇒ Footwear"	4

Item	Support
Clothes	5
Outerwear	2
Jackets	1

A rule is R-interesting with respect to closest ancestor
if support is R*support for closest ancestor

Progressive Deepening

- A top_down, progressive deepening approach:
  - First find high-level strong rules:

    milk → bread [20%, 60%].
  - Then find their lower-level "weaker" rules:

    2% milk → wheat bread [6%, 50%].
- A transaction item that supports x also supports all ancestors of x
- Support will be weaker for lower levels, use different thresholds
- Find the purchase patterns for fresh foods (SQL query)

```
find association rules
related to category, content_spec, brand
from sales_transactions T, sales_item I
where T.bar_code = I.bar_code and I.category = 'food'
and I.storage_period < 21
```

Constraint-Based Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? --- Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98.
  - Dimension/level constraints:
    - in relevance to region, price, brand, customer category.
  - Rule constraints:
    - small sales (price $< \$10$) triggers big sales (sum $> \$200$).
  - Interestingness constraints:
    - strong rules: min_support $\geq$ 3%, min_confidence $\geq$ 60%.

Incremental Update of Discovered Association Rules

- Partitioned derivation and incremental updating.
- A fast updating algorithm, FUP (Cheung et al.'96)
  - View a database: original *DB* $\cup$ incremental *db*
  - A *k*-itemset (for any *k*),
    - * large in *DB* $\cup$ *db* if large in both *DB* and *db*
    - * small in *DB* $\cup$ *db* if small in both *DB* and *db*
  - For those only large in *DB*, merge corresp. counts in *db*
  - For those only large in *db*, search *DB* to update their itemset counts
- Similar methods can be adopted for data removal and update.
- Principles are further developed for distributed/parallel mining of association rules.

Association Rules - Strengths & Weaknesses

- Strengths
  - Understandable and easy to use
  - Useful
- Weaknesses
  - Brute force methods can be expensive (memory and time)
    - Apriori is O(CD), where

      C = sum of sizes of candidates
      (2^n possible, n = #items)

      D = size of database
  - Association does not necessarily imply correlation
  - Validation?
  - Maintenance?
  - Classification?

Example Application