Association Rules

Instructor: Junghye Lee

Department of Industrial Engineering
junghyelee@unist.ac.kr

Contents

Introduction: Point of Sale Transactions

- Transaction and item

Ex) Grocery Point-Of-Sale Transactions

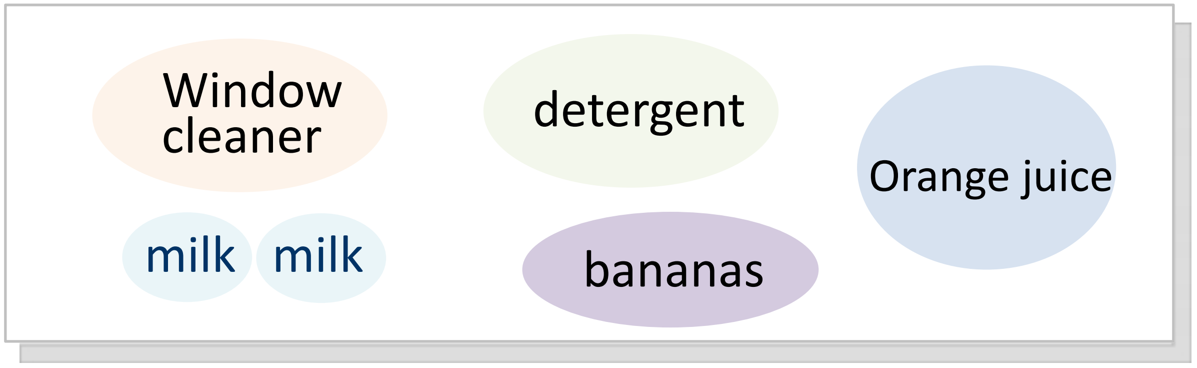
customer	**items**
1	orange juice, banana
2	orange juice, milk
3	detergent, window cleaner

transaction

Introduction: What is Market Basket Analysis?

- Finding useful information in ‘Market Basket’
- Useful information
  - Who customers are
  - **Which products tend to be purchased together**
  - Why some products tend to be purchased together
- Useful information like “If Item A then Item B” is called **‘Association rule’**

Example: shopping cart



Introduction: Transactions and Co-occurrence

Customer	Items
1	Orange juice, Soda
2	Milk, Orange juice, Window Cleaner
3	Orange juice, Detergent
4	Orange juice, Detergent ,Soda
5	Window Cleaner, Soda

- OJ and soda are more likely to be purchased together.
- Detergent is never purchased with window cleaner or milk.
- Milk is never purchased with soda or detergent.

- Confidence of the rule -“if soda, then orange juice”: 2/3 (67%)

- “if orange juice then soda” : 2/4 (50%)

	OJ	Window Cleaner	Milk	Soda	Detergent
OJ	4	1	1	2	1
Window Cleaner	1	2	1	1	0
Milk	1	1	1	0	0
Soda	2	1	0	3	1
Detergent	1	0	0	1	2

Association Rules

- The clear and useful result of market basket analysis.
- Only one result in association rules are strongly recommended.
  - “If diapers and Thursday, then beer” is more useful than “If Thursday, then diapers and beer”.
- Three types of rules
  - the useful
  - the trivial
  - the inexplicable

Association Rules - The Useful Rule

- Contains high quality, actionable information
- Once the pattern (rule) is found, it is often not hard to justify.

  Ex) Rules like “ On Thursday, customer who purchase diapers are likely to purchase beer”

  → Young couples prepare for the weekend by stocking up on diapers for the infants and beer for dad.

- Can be applied to Market layout

  Ex) Placing other baby products within sight of the beer

Association Rules - The Trivial Rule

- Trivial results are already known by anyone familiar with the business.

  Ex) "Customers who purchase maintenance agreements are very likely to purchase large appliances" (they purchase both at the same time.)

  → "customers purchasing paint buy paint brushes"

- Results from market basket analysis may simply be measuring the success of previous marketing campaigns.

Association Rules - The inexplicable rules

- Inexplicable rules give a new fact but no explanation about consumer behavior and future actions.

  Ex) "When a new hardware store opens, one of the most commonly sold items is toilet rings"

- Inexplicable rules can be flukes in data.
- More investigation might give explanation.

The Basic Process in Market Basket Analysis

Choosing the right set of item and right level
- using taxonomy and virtual items

↓

Generating rules by co-occurrence matrix
(measures: support, confidence, improvement)

↓

Overcoming the practical limits
(pruning)

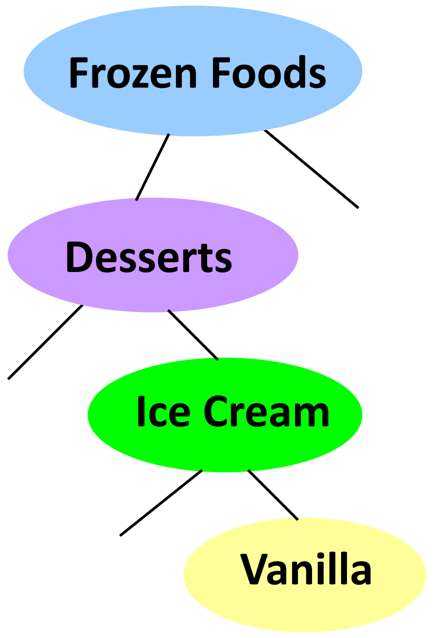
Basic Process: Choosing the right set of items - Taxonomy

- Why?

  - To reduce too many item combinations

- How to find the appropriate level?

  - Consider frequency (roll up rare items to higher levels to help to generalize items)
  - Consider importance (roll down expensive item to lower levels)



When the items occur in about the same number of times in the data, the analysis produce the best results

Basic Process: Choosing the right set of items - Virtual Items

- Items that cross product boundaries
  - They do not appear in the product taxonomy (e.g. “Calvin Klein”, “cash”, “month”)

- Cautions
  - Prime cause of redundant rules (e.g., If Coke products, then Coke)
  - When a virtual item and a generalized item appearing together are a proxy for individual items

    (e.g., If “coke product” and “diet soda” then “pretzels”

    $\longrightarrow$ If “diet coke” then pretzels”)

Basic Process: Generating rules

1. Gathering transactions for selected items
   (including virtual items)
2. Making co-occurrence matrix
3. Finding the most frequent combination from the matrix
4. Making a distinction between ‘condition’ and ‘result’
   from that combination

If “condition”, then “result”.

– **support, confidence, improvement**

Performance Measures - Support

Rule: If “condition” then “result”.

- Support

  - How many transactions that contain “condition” and “result” at the same time ?

$$S = P(\text{"}condition\text{"} \cap \text{"}result\text{"}) = \frac{\text{\# of transactions that include "}condition\text{" and "}result\text{"}}{\text{\# of total transactions}}$$

- Support can be used to eliminate uninteresting rules.

Performance Measures - Confidence

- Confidence

  - How many transactions that contain “condition” and “result” among the transactions including “condition” ?

$$C = P(\text{" }result\text{"}|\text{" }condition\text{" }) = \frac{P(condition \cap result)}{P(condition)}$$

$$= \frac{\text{\# of transactions that include condition and result}}{\text{\# of transactions that include condition}}$$

  - Conditional probability

  - Degree of association – may not imply causality

  - not symmetric

Performance Measures - Improvement

- Improvement or lift

  - Lift (improvement) tells us how much better a rule is at predicting the result than just guessing the result at random

$$I = \frac{P(\mathit{result} \mid \mathit{condition})}{P(\mathit{result})} = \frac{P(\mathit{condition} \cap \mathit{result})}{P(\mathit{condition})\, P(\mathit{result})}$$

Improvement	Relationship	Example
1	Independent	Pepper and cookie
> 1	Complementary	Bread and butter
< 1	Substitutional	Butter and margarine

Basic Process: Generating rules - Example

- Grocery shopping cart

Customer	Items
1	Orange juice, Soda
2	Milk, Orange juice, Window Cleaner
3	Orange juice, Detergent
4	Orange juice, Detergent ,Soda
5	Window Cleaner, Soda

Basic Process: Generating rules - Example

- Co-occurrence Matrix

	OJ	Window Cleaner	Milk	Soda	Detergent
OJ	4 (0.8)	1 (0.2)	1 (0.2)	2 (0.4)	1 (0.2)
Window Cleaner	1 (0.2)	2 (0.4)	1 (0.2)	1 (0.2)	0
Milk	1 (0.2)	1 (0.2)	1 (0.2)	0	0
Soda	2 (0.4)	1 (0.2)	0	3 (0.6)	1 (0.2)
Detergent	1 (0.2)	0	0	1 (0.2)	2 (0.4)

Basic Process: Generating rules - Example

- Assume most common combination ‘A,B,C’

Combination	Probability	Combination	Probability
A	45%	A and B	25%
B	42.5%	A and C	20%
C	40%	B and C	15%
		A and B and C	5%

Basic Process: Generating rules - Example

- Which is Result between A,B,C?
- Setting a result on the basis of ‘confidence’
- Confidence of rule “ If condition then result”
  - P(Result|Condition) = P(Result and Condition)/P(Condition)
- Confidence of rule “If AB then C” = P(ABC|AB)

support

Association Rule	**P(condition)**	**P(condition and result)**	**confidence**
If AB then C	25%	5%	0.20
If AC then B	20%	5%	0.25
If BC then A	15%	5%	0.33

Basic Process: Generating rules - Example

- What if P(R) > P(R|C) (= Confidence) ?
- Improvement' tells how much better a rule is than just guessing randomly the result
- Improvement = P(R|C) / P(R) = P(RC)/P(R)P(C)
  - If Improvement > 1 → the rule is better
  - If Improvement < 1 → "If C, then NOT R" is better

(Negative rule)

Rule	confidence	Improvement
If BC then A	0.33	**0.73** ← 0.33/0.45
If BC then Not A	0.67	**1.22** ← 0.67/0.55
If A then B	0.56	**1.31**

0.25/0.45

Basic Process: Overcoming the practical limits

- Exponential growth as problem size increases
  - On menu with 100 items, how many combinations are there?
    → 161,700 (3 items)

- How to solve the problem of Big Data?
  - **To use the taxonomy** ; generalize items that can meet criterion
  - **To use Pruning** ; throw out item or combination of items that do not meet criterion. "**Minimum support pruning"** is the most common pruning mechanism

Strengths of Market Basket Analysis

- It produces clear and understandable results
  - because the results are association rules
- It supports both directed and undirected data mining
- It can handle transactions themselves
- The computations it uses are simple to understand
  - The calculation of confidence and improvement is simple

Weaknesses of Market Basket Analysis

- It requires exponentially more computational effort as the problem size grows
  - number of items, complexity of the rules
- It has limited support for data attribute
  - Virtual items make rules more expressive
- It is difficult to determine the right number of items
- It discounts rare items

Application of Market Basket Analysis

- It can suggest store layouts.
- It can tell which products are amenable to promotion.
- It is used to compare stores.
- It can be applied to time-series problems.

Example

Transactions	Items
1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

Rule: If **b** and **c**, then **g**

$$X = \{b, c\}, Y = \{g\}$$

$$conf(R) = \frac{supp(X \cup Y)}{supp(X)} = \frac{0.6}{0.8} = 0.75$$

$$lift(R) = \frac{conf(R)}{supp(Y)} = \frac{0.75}{0.6} = 1.25$$

X "affects" Y **25%**

How Many Possible Rules?

Example

Transactions	Items
1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

Rule: If **b** and **c**, then **g**

$$X = \{b, c\}, Y = \{g\}$$

$$conf(R) = \frac{supp(X \cup Y)}{supp(X)} = \frac{0.6}{0.8} = 0.75$$

$$lift(R) = \frac{conf(R)}{supp(Y)} = \frac{0.75}{0.6} = 1.25$$

X "affects" Y **25%**

How Many Possible Rules?

$$I = \{a, b, c, d, e, f, g\}$$

When $|X| = 1, 7$, $|X| = 2, 21$, $|X| = 3, 35$ $\cdots$

Apriori Algorithm

- Agrawal and Srikant, 1994

(Phase 1) Find all frequent itemsets having the minimum support smin.

(Phase 2) Consider a subset A of a frequent itemset L.

For a specified confidence cmin,

if $\frac{supp(L)}{supp(A)} \geq C_{min}$,

then, generate a rule R: A => (L-A).

So, the support of this rule will be supp(R)=supp(L), and the confidence will be conf(R)= supp(L)/supp(A).

This is exactly same as Page 20!

Apriori Algorithm – Phase 1

Step 0. Specify the minimum support s_{min}.

$k = 1 \qquad C_1 = [\{i_1\}, \{i_2\}, \dots, \{i_m\}] \qquad L_1 = \{c \in C_1 | supp(c) \geq s_{min}\}$

Step 1. $k = k + 1$

Generate new candidate itemsets C_k from L_{k-1}

(apriori-gen function)

Step 1-1. (join)

Generate k-itemsets by joining like $C_k = L_{k-1} * L_{k-1}$

Step 1-2. (prune)

Delete any itemset in C_k having the itemset not belonging to L_{k-1}

Step 2. Generate L_k such that $L_k = \{c \in C_k | supp(c) \geq s_{min}\}$

Repeat Step 1.

Apriori Algorithm – Example

transaction	items
1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

$S_{min} = 0.4$

C1=[{a}, {b}, {c}, {d}, {e}, {f}, {g}]

L1=[{a}, {b}, {c}, {e}, {f}, {g}]

C2=[{a,b}, {a,c}, {a,e}, {a,f}, {a,g}, {b,c}, {b,e}, {b,f}, {b,g}, {c,e}, {c,f}, {c,g}, {e,f}, {e,g}, {f,g}]

L2=[{a,b}, {b,c}, {b,e}, {b,f}, {b,g}, {c,e}, {c,f}, {c,g}, {e,f}]

C3=[{b,c,e}, {b,c,f}, {b,c,g}, {b,e,f}, {c,e,f}]

L3=[{b,c,e}, {b,c,f}, {b,c,g}, {b,e,f}, {c,e,f}]

C4=[{b,c,e,f}]

L4=[{b,c,e,f}]

Apriori Algorithm – Example

- Rules generated

  L={b,c,g}

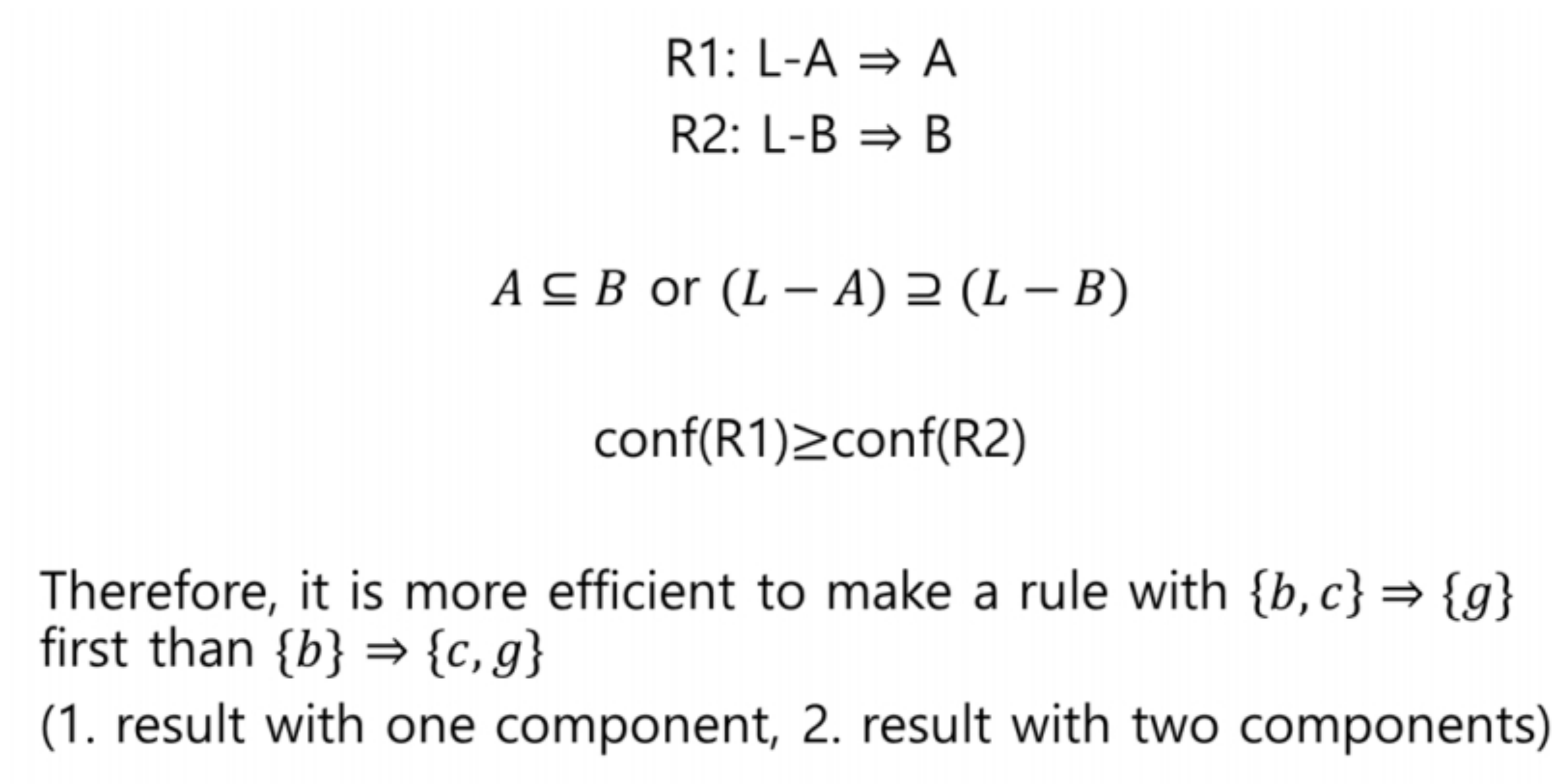
  Candidates of rules having 1-item in ‘result’:

  R1: {b,c} {g} $\Rightarrow$ conf(R1)=0.6/0.8 = 0.75

  R2: {b,g} {c} $\Rightarrow$ conf(R2)=0.6/0.6 = 1

  R3: {c,g} {b} $\Rightarrow$ conf(R3)=0.6/0.6 = 1

Apriori Algorithm – Theorem



R1: L-A ⇒ A

R2: L-B ⇒ B

$$A \subseteq B \text{ or } (L - A) \supseteq (L - B)$$

$$\mathrm{conf}(R1) \geq \mathrm{conf}(R2)$$

Therefore, it is more efficient to make a rule with $\{b, c\} \Rightarrow \{g\}$ first than $\{b\} \Rightarrow \{c, g\}$

(1. result with one component, 2. result with two components)

Quiz

$L = \{a, b, c, d, e\}$

When only two rules are left ,

$Rule\ 1\colon \{a, c, d, e\} \Rightarrow \{b\}$

$Rule\ 2\colon \{a, b, c, e\} \Rightarrow \{d\}$

What about the following rules?

$Rule\ 3\colon \{a, c, d\} \Rightarrow \{b, e\}$

$Rule\ 4\colon \{a, c, e\} \Rightarrow \{b, d\}$

$Rule\ 5\colon \{a, d, e\} \Rightarrow \{b, c\}$

$Rule\ 6\colon \{c, d, e\} \Rightarrow \{a, b\}$

Sequential Patterns

(optional)

Sequential Patterns

- Sequence: List of items in the order of time, etc.

  - Ex: $s_1 =< A_1 , A_2 , \dots , A_n >, s_2 =< B_1 , B_2 , \dots , B_n >$
- Length of sequence: number of items in a sequence, (k-sequence)
- Subsequence s_1 of s_2: sequence s_1 is a subsequence of s_2 if

  $A_1 \subseteq B_{i_1}, A_2 \subseteq B_{i_2}, \dots , A_n \subseteq B_{i_n} \quad i_1 < i_2 < \cdots < i_n$

  - Ex: s1=<{a}, {b,c}, {e}>, s2=<{f}, {a,g}, {h}, {b,c,d},{e,j}>, s3=<{a}, {b}, {c}, {e}>

    → s1 is a subsequence of s2, but s3 is not a subsequence of s2

Sequential Patterns

- Sequence *s* is called as *maximal* if it is not a subsequence of others.
- Support of a sequence *s*:

  supp(s) = proportion of customers that contain sequence *s*

<example>

#	Customer sequences
1	<{a}, {b}>
2	<{c,d}, {a}, {e,f,g}>
3	<{a,h,g}>
4	<{a}, {e,g}, {b}>
5	<{b}>

Maximal sequences having min. support 0.4:

s1=<{a}, {b}>, s2=<{a}, {e, g}>
s3=<{a}, {e}> is not maximal

Algorithm for Finding Sequences

- Agrawal and Srikant (1995)

  1. Sort Phase: convert the transaction database into customer sequences
  2. *L*-itemset Phase: Find the set of all *l*-itemsets *L* by considering the minimum support.
  3. Transformation Phase: Transform each customer sequence into the set of all *l*-itemsets contained in that transaction.
  4. Sequence Phase: Generate large sequences
  5. Maximal Phase: Find the maximal sequences among large sequences

Algorithm for Finding Sequences

- Example (cont'd)

Min. support: 0.4

2) L-itemset Phase

itemset	support	#
{a}	4	1
{b}	3	2
{e}	2	3
{e, g}	2	4
{g}	3	5

3)Transformation Phase

cust #	Cust seq	Transformed seq
1	<{a}, {b}>	<{1}, {2}>
2	<{c,d}, {a}, {e,f,g}>	<{1}, {3, 4, 5}>
3	<{a,h,g}>	<{1, 5}>
4	<{a}, {e,g}, {b}>	<{1}, {3, 4, 5}, {2}>
5	<{b}>	<{2}>

Algorithm for Finding Sequences

4)Sequence Phase

- AprioriAll: does not guarantee max seq, so requires Maximal Phase
- AprioriSome, ‘DynamicSome’: guarantee max seq

AprioriAll

```
Step 0. Set all large 1-seq to L1.
        k = 1
Step 1. k = k+1
        $C_k = L_{k-1} * L_{k-1}$
Step 2. Obtain $L_k$ from $C_k$
        Stop if $L_k = \phi$   Repeat Step 1, otherwise.
```

Example (cont’d)

L1=[<1>, <2>, <3>, <4>, <5>]
L2=[<1, 2>, <1, 3>, <1, 4>, <1, 5>]
L3=[] $\rightarrow$ stop
Max seq : <1, 2> and <1,4>

Algorithm for Finding Sequences

- Example

cust	Transformed seq
1	<{1,5}, {2}, {3}, {4}>
2	<{1}, {3}, {4}, {3,5}>
3	<{1}, {2}, {3}, {4}>
4	<{1}, {3}, {5}>
5	<{4}, {5}>

Min support: 0.4

L1=[<1>, <2>, <3>, <4>, <5>]

L2=[<1 2>, <1 3>, <1 4>, <1 5>, <2 3>, <2 4>, <3 4>, <3 5>, <4 5>]

C3=[<1 2 3>, <1 2 4>, <1 3 4>, <1 3 5>, <1 4 5>, <2 3 4>, <2 3 5>, <2 4 5>, <3 4 5>]

L3=[<1 2 3>, <1 2 4>, <1 3 4>, <1 3 5>, <2 3 4>]

C4=L4=[<1 2 3 4>]

Max seq: <1 2 3 4>, <1 3 5>, <4 5>

Algorithm for Finding Sequences

```
AprioriSome
  (Forward phase)
    Step 0. k=1; Obtain L1; C1=L1; last=1.
    Step 1. (Generate $C_k$)
        $k \leftarrow k + 1$
        1) $L_{k-1}$ known: $C_k = L_{k-1} * L_{k-1}$
        2) $L_{k-1}$ unknown: $C_k = C_{k-1} * C_{k-1}$
    Step 2. (select $k$ for $L_k$)
        Stop if $C_k = \phi$,      Proceed, otherwise.
        1) If $k$ = next(last), obtain $L_k$, last = $k$, Go to Step 1.
        2) If not $k$ = next(last), go to Step 1.
  (Backward phase)
    Step 0. $k = kmax$
    Step 1.
    1) $L_k$ known: delete all subsequences in $L_i$ $(i > k)$ from $L_k$
    2) $L_k$ unknown: delete all subsequences in $L_i$ $(i > k)$ from $C_k$
    Step 2. $k \leftarrow k - 1$; go to Step 1.
```

Algorithm for Finding Sequences

- Function ‘next’ determines the length of sequences.
  - Agrawal & Srikant (1995)

    hit(k) = $\frac{|L_k|}{|C_k|}$

    1) If hit(k) < 0.666 , next(k) = k + 1
    2) If 0.666 ≤ hit(k) < 0.75 , next(k) = k + 2
    3) If 0.75 ≤ hit(k) < 0.80, next(k) = k + 3
    4) If 0.80 ≤ hit(k) < 0.85, next(k) = k + 4
    5) If hit(k) ≥ 0.85, next(k) = k + 5

- Once the all large sequences are obtained, the union will be the maximal sequence.

Algorithm for Finding Sequences

Example(cont'd)- AprioriSome

next(i)=2i

(Forward phase)

<u>Iteration 0.</u>

L1=C1=[<1>, <2>, <3>, <4>, <5>],   last=1

<u>Iteration 1. (*k* = 2)</u>

C2=[<1 2>, <1 3>, <1 4>, <1 5>, <2 3>, <2 4>, <2 5>, <3 4>, <3 5>, <4 5>]

next(1)=2=k

L2=[<1 2>, <1 3>, <1 4>, <1 5>, <2 3>, <2 4>, <3 4>, <3 5>, <4 5>]

last=2

<u>Iteration 2. (*k* = 3)</u>

C3=[<1 2 3>, <1 2 4>, <1 3 4>, <1 3 5>, <1 4 5>, <2 3 4>, <2 3 5>, <2 4 5>,<3 4 5>]

next(2)=$4 \neq 3$

<u>Iteration 3. (*k* = 4)</u>

C4=[<1 2 3 4>, <1 2 3 5>, <1 2 4 5>, <1 3 4 5>, <2 3 4 5>]

next(2)=4=k

L4=[<1 2 3 4>]

Algorithm for Finding Sequences

Example(cont'd)

(Backward phase)

Iteration 0.

kmax = 4

Iteration 1.

L4=[<1 2 3 4>]; *k* = 3

Iteration 2.

L3=[<1 3 5>]; *k* = 2

Iteration 3.

L2=[<4 5>]

Max sequences: <1 2 3 4>, <1 3 5>, <4 5>