Frequent Pattern Mining

Young-Rae Cho, Ph.D.
Associate Professor
Division of Software / Division of Digital Healthcare
Yonsei University – Mirae Campus

Overview

1. Market Basket Problem
2. Apriori Algorithm
3. CHARM Algorithm
4. Advanced Frequent Pattern Mining
5. Constraint-Based Mining

Market Basket Problem

- **Example**
  - "Customers who bought beer also bought diapers."
- **Motivation**
  - To promote sales in retail by cross-selling
- **Required Data**
  - Customers' purchase patterns
  - Items often purchased together by each customer
- **Applications**
  - Store arrangement
  - Catalog design
  - Discount plans

Basic Terms

- **Transaction**
  - A set of items which are bought by one person at one time
- **Frequent Itemset**
  - A set of items which occur frequently across transactions
  - A subset of a transaction
- **Association Rule**
  - A one-directional relationship between two sets of items
  - e.g., $A \rightarrow B$ where A and B are sets of items
- **Support**
  - Frequency of a set of items across transactions
- **Confidence**
  - For $A \rightarrow B$, percentage of transactions containing A that also contain B

Frequent Itemsets

- **Transaction Data**

T-ID	Items
1	bread, eggs, milk, diapers
2	coke, beer, nuts, diapers
3	eggs, juice, beer, nuts
4	milk, beer, nuts, diapers
5	milk, beer, diapers

- **Support**
  - What is support of {beer} ?   {beer, nuts} ?
  - Which itemsets have 80% support ? 60% support ?

- **Frequent Itemsets**
  - Itemsets having support greater than (or equal to) a user-specified minimum support

Association Rules

❑ Transaction Data

T-ID	Items
1	bread, eggs, milk, diapers
2	coke, beer, nuts, diapers
3	eggs, juice, beer, nuts
4	milk, beer, nuts, diapers
5	milk, beer, diapers

❑ Confidence

- What is the confidence of {beer} → {nuts}?
- Which associate rules have 100% confidence?

❑ Association Rules

- Rules having confidence greater than (or equal to) a user-specified minimum confidence

Solving Market Basket Problem

❑ **Process**

- Step 1:
  - Finding all frequent itemsets where size ≥ 2
  - based on the minimum support (min_sup)
  - e.g. { beer, nuts, diapers }

- Step 2:
  - Generating association rules from the frequent itemsets
  - based on the minimum confidence (min_conf)
  - e.g. { beer } → { nuts, diapers } → **Expected output knowledge**

Example of Finding Association Rules

- **College Course Registration Data**
  - Courses registered on Fall 2020

Student	Courses
John	Artificial Intelligence, Databases, Data Mining
Bob	Operating Systems, Data Comm., Bioinformatics, Data Mining
Mary	Graphics, Operating Systems, Data Comm., Data Mining
David	Artificial Intelligence, Databases, Operating Systems
Jack	Graphics, Artificial Intelligence, Databases
Lisa	Artificial Intelligence, Operating Systems, Data Comm., Data Mining

  - Find all association rules with 50% min_sup and 80% min_conf

Generalized Formulas

❑ Association Rules

- $I = \{ I_1, I_2, \ldots, I_m \}$, $T = \{ T_1, T_2, \ldots, T_n \}$, $T_k \subseteq I$ for $\forall\ k$
- $A \rightarrow B$ where

  $A \subseteq I$ ($A \neq \varnothing$), $B \subseteq I$ ($B \neq \varnothing$), $A \subseteq T_i$ for $\exists\ i$, $B \subseteq T_j$ for $\exists\ j$, and $A \cap B = \varnothing$

❑ Computation of Support

$$\boxed{\text{support } (A \rightarrow B) = \text{P } (A \cup B)}$$

where $P(X) = \dfrac{|\{ T_i | X \subseteq T_i \}|}{n}$

❑ Computation of Confidence

$$\boxed{\text{confidence } (A \rightarrow B) = \frac{\text{P } (A \cup B)}{\text{P } (A)}}$$

Problem of Support & Confidence

- **Support Data**

	Tea	Not Tea	SUM
Coffee	20	50	70
Not Coffee	10	20	30
SUM	30	70	100

- **Association Rule, {Tea} → {Coffee}**
  - Support ({Tea} → {Coffee}) ?
  - Confidence ({Tea} → {Coffee}) ?

- **Problems in this Dataset ?**

Alternative Measures

❑ Coverage

$$\text{coverage}(A \rightarrow B) = P(A)$$

❑ Lift

$$\text{lift}(A \rightarrow B) = \frac{\text{confidence}(A \rightarrow B)}{P(B)} = \frac{P(A \cup B)}{P(A) \times P(B)} = \text{correlation}(A, B)$$

- The association rule A → B is interesting if lift(A → B) > 1
- However, it is the same to correlation between A and B
- Positive correlation if lift(A,B) > 1
- Negative correlation if lift(A,B) < 1
- No relationship if lift(A,B) = 1

Alternative Measures – cont'd

❑ χ^2 Test (Chi-Square Test)

- Evaluates whether an observed distribution in a sample differs from
  a theoretical distribution (i.e., hypothesis).
  where E_i is an expected frequency and O_i is an observed frequency,

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

- The larger χ^2, the more likely the variables are related (positively or negatively).
- Example?

Overview

1. **Market Basket Problem**
2. **Apriori Algorithm**
3. **CHARM Algorithm**
4. **Advanced Frequent Pattern Mining**
5. **Constraint-Based Mining**

Frequent Itemset Mining

❑ **Process to solve Market Basket Problem**

1) Find frequent itemsets → computational problem
2) Find association rules

❑ **Finding All Frequent Itemsets**

- Brute Force Algorithm (Exhaustive Algorithm)
  - Enumerate all possible subsets of the total itemset, I
  - Count frequency of each subset
  - Select frequent itemsets

❑ **Problem ?**

- Enumerating all candidates is not computationally acceptable

  → Efficient & scalable algorithm is required.

Apriori Algorithm

- **Motivations**
  - Efficient frequent itemset analysis
  - Scalable approach

- **Process of Apriori**
  - Iterative increment of the itemset size
    1) Candidate itemset generation → computational problem
    2) Frequent itemset selection

- **Downward Closure Property**
  - Any superset of an itemset X cannot have higher support than X.

    → If an itemset X is frequent (support of X is higher than min_sup),

    then any subset of X should be frequent.

Candidate Itemset Generation

❑ Process

- Two steps: (1) <u>selective joining</u> and (2) <u>a priori pruning</u>

❑ Selective Joining

- Each candidate itemset with size k is generated by joining two frequent itemsets with size (k-1)
- The frequent itemsets with size (k-1) which share a frequent sub-itemset with size (k-2) are joined

❑ A priori Pruning

- A frequent itemset with size k which has any infrequent sub-itemsets with size (k-1) is pruned

Detail of Apriori Algorithm

❑ Notations

- C_k : Candidate itemsets of size k
- L_k : Frequent itemsets of size k
- sup_{min}: minimum support

❑ Pseudo Code

1. k $\leftarrow$ 1
2. $L_k \leftarrow$ frequent itemsets with size 1
3. while $L_k \neq \varnothing$ do
4.   k $\leftarrow$ k + 1
5. $C_k \leftarrow$ candidate itemsets by selective joining & a priori pruning from $L_{(k-1)}$
6. $L_k \leftarrow$ frequent itemsets using sup_{min}
7. end while
8. return U_k L_k

Example of Apriori Algorithm

- $sup_{min} = 2$

T-ID	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

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	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	Sup.
{B,C,E}	2

L_3

Itemset	Sup.
{B,C,E}	2

Summary of Apriori Algorithm

- **Features**
  - An iterative approach of a level-wise search
  - Reducing search space by downward closure property

- **Challenges**
  - Multiple scan of transaction database
  - Huge number of candidates
  - Tedious workload of support counting

- **Solutions**
  - Reducing transaction database scans
  - Shrinking number of candidates
  - Facilitating support counting

Overview

1. **Market Basket Problem**
2. **Apriori Algorithm**
3. **CHARM Algorithm**
4. **Advanced Frequent Pattern Mining**
5. **Constraint-Based Mining**

Association Rule Mining

❑ Process of Market Basket Problem

1) Find frequent itemsets → computational problem
2) Find association rules → redundant rule generation

❑ Example 1

- ~~{ beer } → { nuts } (40% support, 75% confidence)~~
- { beer } → { nuts, diapers } (40% support, 75% confidence)
- The first rule is not meaningful.

❑ Example 2

- { beer } → { nuts } (60% support, 75% confidence)
- { beer, diapers } → { nuts } (40% support, 75% confidence)
- Both rules are meaningful.

Frequent Closed Itemsets

General Definition of Closure

- A frequent itemset X is **closed** if there exists no superset of X with the same support as X.
- Different from frequent maximal itemsets

Frequent Closed Itemsets with Min. Support of 40%

- { milk, diapers } 60%
- ~~{ milk, beer }~~ 40%
- { beer, nuts } 60%
- { beer, diapers } 60%
- ~~{ nuts, diapers }~~ 40%
- { milk, beer, diapers } 40%
- { beer, nuts, diapers } 40%

T-ID	Items
1	bread, eggs, milk, diapers
2	coke, beer, nuts, diapers
3	eggs, juice, beer, nuts
4	milk, beer, nuts, diapers
5	milk, beer, diapers

Mapping between Items and Transactions

Mapping Functions

- $I = \{ I_1, I_2, \ldots, I_m \}$, $T = \{ T_1, T_2, \ldots, T_n \}$, $X \subseteq I$, $Y \subseteq T$
- $i : T \rightarrow I$, $i(Y)$: itemset that is contained in all transactions in Y
- $t : I \rightarrow T$, $t(X)$: set of transactions (tidset) that contain all items in X

Properties

- $\boldsymbol{X_1 \subseteq X_2 \rightarrow t(X_1) \supseteq t(X_2)}$

  (e.g.) {ACW} $\subseteq$ {ACTW} $\rightarrow$ {1345} $\supseteq$ {135}

- $\boldsymbol{Y_1 \subseteq Y_2 \rightarrow i(Y_1) \supseteq i(Y_2)}$

  (e.g.) {245} $\subseteq$ {2456} $\rightarrow$ {CDW} $\supseteq$ {CD}

- $\boldsymbol{X \subseteq i(t(X)),\ Y \subseteq t(i(Y))}$

  (e.g.) t({AC}) = {1345}, i({1345}) = {ACW}

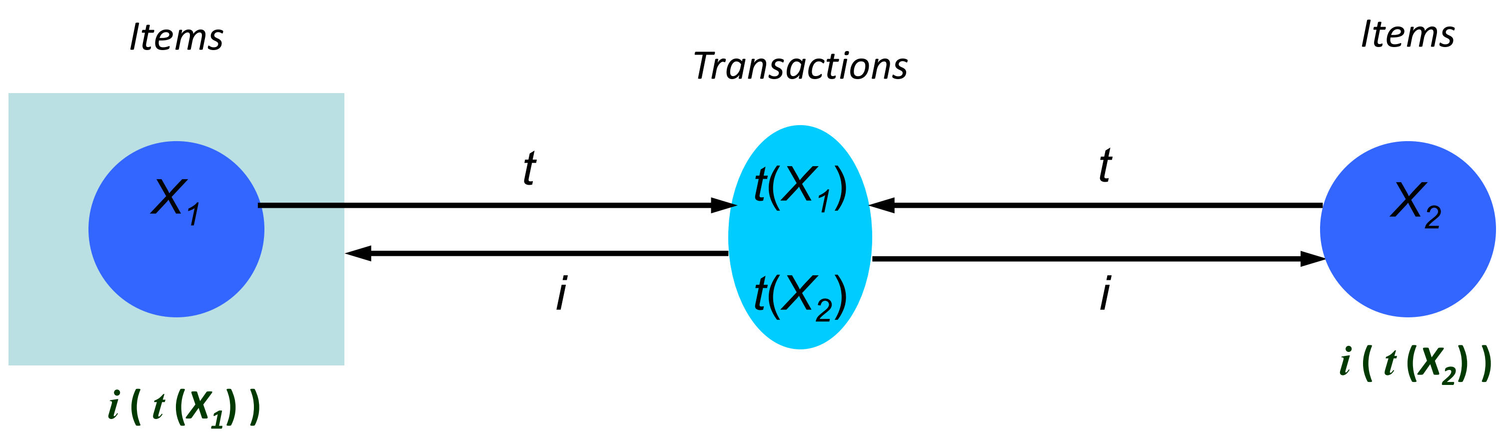
  (e.g.) i({134}) = {ACW}, t({ACW}) = {1345}

T-ID	Items
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

Definition of Closure

- **Closure Operator**
  - $c_{it}(X) = i(t(X))$
  - $c_{ti}(Y) = t(i(Y))$

- **Formal Definition of Closure**
  - An itemset X is **closed** if $X = c_{it}(X)$
  - A tid-set Y is **closed** if $Y = c_{ti}(Y)$



X_1 is not closed.

X_2 is closed.

Examples of Closed Itemsets

- **Examples**
  - $X = \{ACW\}$ $\boxed{\text{support}(X) = 67\%}$

    $t(X) = \{1345\}$, $i(t(X)) = \{ACW\}$

    → X is closed.
  - $X = \{AC\}$ $\boxed{\text{support}(X) = 67\%}$

    $t(X) = \{1345\}$, $i(t(X)) = \{ACW\}$

    → X is not closed.
  - $X = \{ACT\}$ $\boxed{\text{support}(X) = 50\%}$

    $t(X) = \{135\}$, $i(t(X)) = \{ACTW\}$

    → X is not closed.
  - $X = \{CT\}$ $\boxed{\text{support}(X) = 67\%}$

    $t(X) = \{1356\}$, $i(t(X)) = \{CT\}$

    → X is closed.

T-ID	Items
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

CHARM Algorithm

- **Motivations**
  - Efficient frequent closed itemset analysis
  - Non-redundant rule generation

- **Property**
  - Simultaneous exploration of itemset space and tid-set space
  - Not enumerating all possible subsets of a closed itemset
  - Early pruning strategy for infrequent and non-closed itemsets

- **Process of CHARM**
  - for each itemset pair
    1) Computing the frequency of their union set
    2) Pruning all infrequent and non-closed branches

Frequency Computation

Operation

- Tid-set of the union of two itemsets, X_1 and X_2
- Intersection of two tid-sets, $t(X_1)$ and $t(X_2)$

$$\boxed{t(X_1 \cup X_2) = t(X_1) \cap t(X_2)}$$

Example

- $X_1 = \{AC\}$, $X_2 = \{D\}$
- $t(X_1 \cup X_2) = t(\{ACD\}) = \{45\}$
- $t(X_1) \cap t(X_2) = \{1345\} \cap \{2456\} = \{45\}$

T−ID	Items
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

Pruning Strategy

Pruning non-closed itemsets

- Suppose two itemsets $X_1 \leq X_2$

(1) $t(X_1) = t(X_2) \rightarrow t(X_1) \cap t(X_2) = t(X_1) = t(X_2)$

$\rightarrow$ Replace X_1 with ($X_1 \cup X_2$), and prune X_2

(2) $t(X_1) \subset t(X_2) \rightarrow t(X_1) \cap t(X_2) = t(X_1) \neq t(X_2)$

$\rightarrow$ Replace X_1 with ($X_1 \cup X_2$), and keep X_2

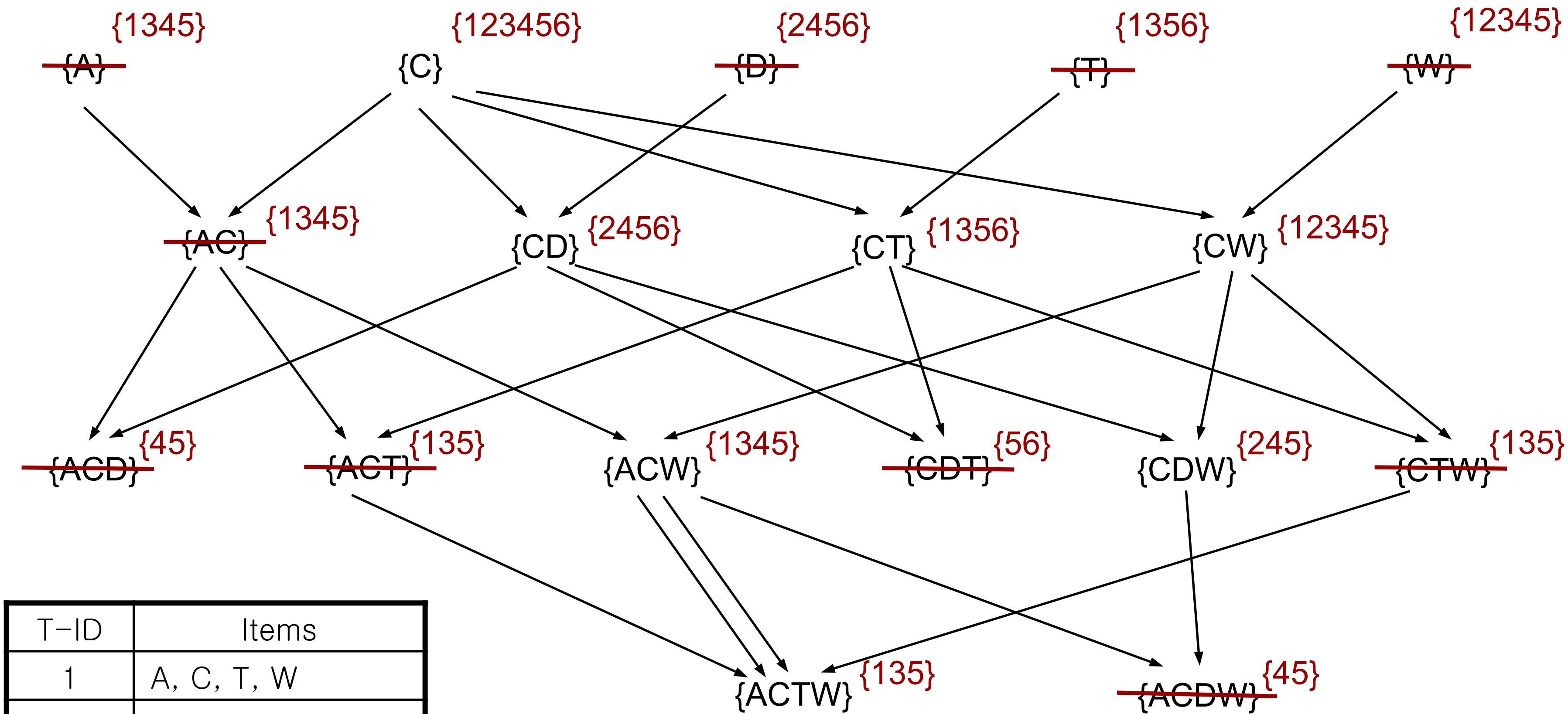
(3) $t(X_1) \supset t(X_2) \rightarrow t(X_1) \cap t(X_2) = t(X_2) \neq t(X_1)$

$\rightarrow$ Replace X_2 with ($X_1 \cup X_2$), and keep X_1

(4) $t(X_1) \neq t(X_2) \rightarrow t(X_1) \cap t(X_2) \neq t(X_1) \neq t(X_2)$

$\rightarrow$ Keep X_1 and X_2

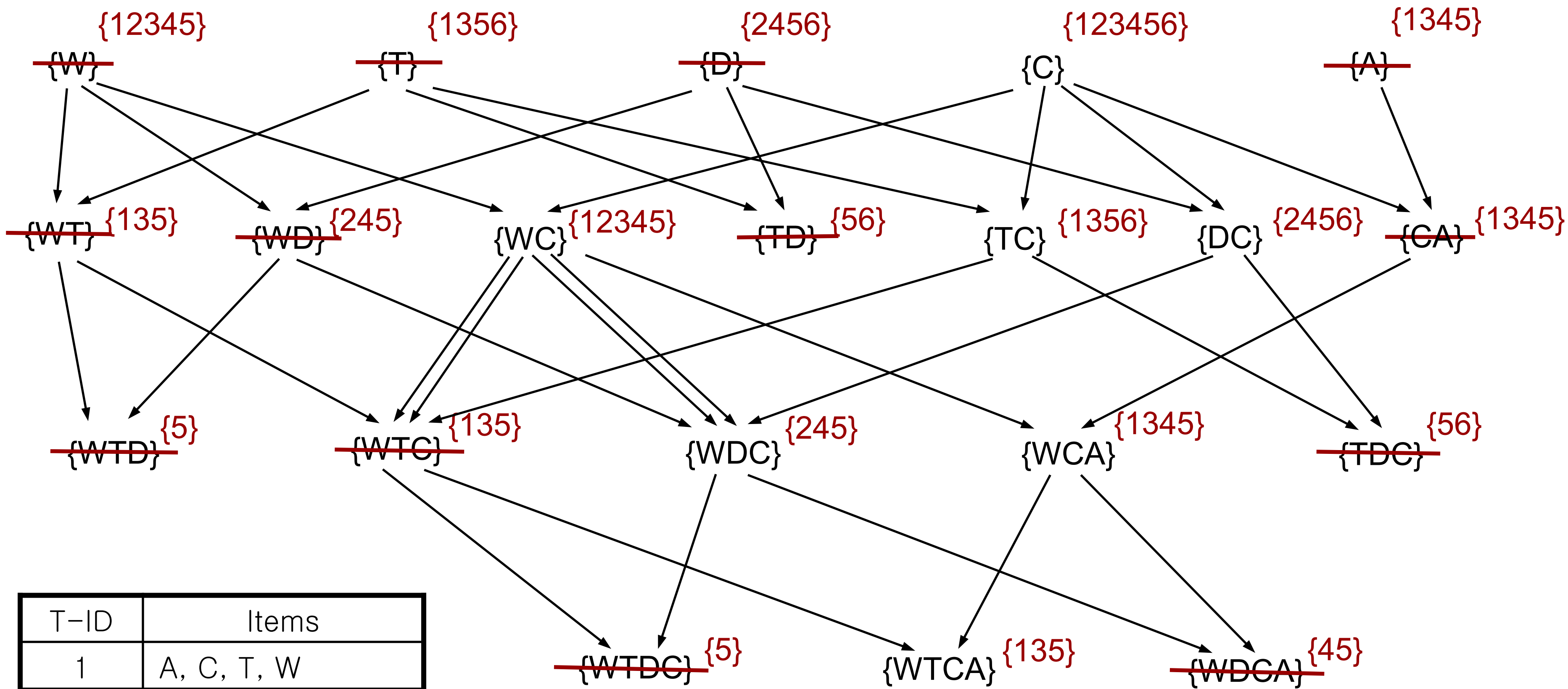
Example of CHARM Algorithm



T−ID	Items
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

50% minimum support

Example of CHARM Algorithm – cont'



T-ID	Items
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

50% minimum support

Summary of CHARM Algorithm

- **Advantages**
  - No need multiple scan of transaction database
    → Revision and enhancement of Apriori algorithm
  - No loss of information

- **References**

  - Zaki, M.J., “Generating Non-Redundant Rule Generation”, In Proceedings of ACM SIGKDD (2000)
  - Zaki, M.J. and Hsiao, C.-J., “CHARM: An Efficient Algorithm for Closed Itemset Mining”, In Proceedings of SDM (2002)

Overview

1. Market Basket Problem
2. Apriori Algorithm
3. CHARM Algorithm
4. <u>Advanced Frequent Pattern Mining</u>
5. Constraint-Based Mining

Frequent Pattern Mining

- **Definition**
  - Discovering frequent patterns
    - patterns (sets of items, sub-sequences, sub-structures, etc.) that occur frequently in a data set

- **Motivation**
  - Finding inherent regularities in data

    e.g., What products were often purchased together?

    e.g., What are the subsequent purchases after buying a PC?

    e.g., What kinds of DNA sequences are sensitive to this new drug?

    e.g., Can we find web documents similar to my research?

- **Applications**
  - Market basket analysis, DNA sequence analysis, Web log analysis

Why Frequent Pattern Mining?

- **Importance**
  - A frequent pattern is an intrinsic and important property of data sets
  - Foundation for many essential data mining tasks
    - Association, correlation, and causality analysis
    - Sequential, structural (sub-graph) pattern analysis
    - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
    - Classification: discriminative, frequent pattern analysis
    - Cluster analysis: frequent pattern-based clustering
    - Data pre-processing: data reduction and compression
    - Data warehousing: iceberg cube computation

Sampling Approach

- **Motivation**
  - Problem: Typically huge data size
  - Mining a subset of the data to reduce candidate search space
  - Trade-off some degree of accuracy against efficiency

- **Process**
  1) Selecting a set of random samples from the original database
  2) Mining frequent itemsets with the set of samples using Apriori
  3) Verifying the frequent itemsets on the border of closure of frequent itemsets

- **Reference**
  - Toivonen, H., “Sampling large databases for association rules.” In Proceedings of VLDB (1996)

Partitioning Approach

- **Motivation**
  - Problem: Typically huge data size
  - Partitioning data to reduce candidate search space

- **Process**
  1) Partitioning database and find local frequent patterns
  2) Consolidating global frequent patterns

- **Reference**
  - Savasere, A., Omiecinski, E. and Navathe, S., “An efficient algorithm for mining association in large databases.” In Proceeding of VLDB (1995)

Hashing Approach

- **Motivation**
  - Problem: A very large number of candidates generated
  - The process in the initial iteration (e.g., size-2 candidate generation) dominates the total execution cost
  - Hashing itemsets to reduce the size of candidates

- **Process**
  1) Hashing itemsets into several buckets in a hash table
  2) If a k-itemset whose corresponding hashing bucket count is below the min support, then it cannot be frequent, thus should be removed

- **Reference**
  - Park, J.S., Chen, M.S. and Yu, P., “An efficient hash-based algorithm for mining association rules.” In Proceedings of SIGMOD (1995)

Pattern Growth Approach

- **Motivation**
  - Problem: A very large number of candidates generated
  - Finding frequent itemsets without candidate generation
  - Grows short patterns to long ones using local frequent items only
  - Depth-first search (Apriori: Breadth-first search, Level-wise search)

- **Example**
  - "abc" is a frequent pattern
  - "d" is a frequent item → "abcd" is a frequent pattern ?

- **Reference**
  - Han, J., Pei, J. and Yin, Y. "Mining frequent patterns without candidate generation." In Proceedings of SIGMOD (2000)

FP(Frequent Pattern)-Tree

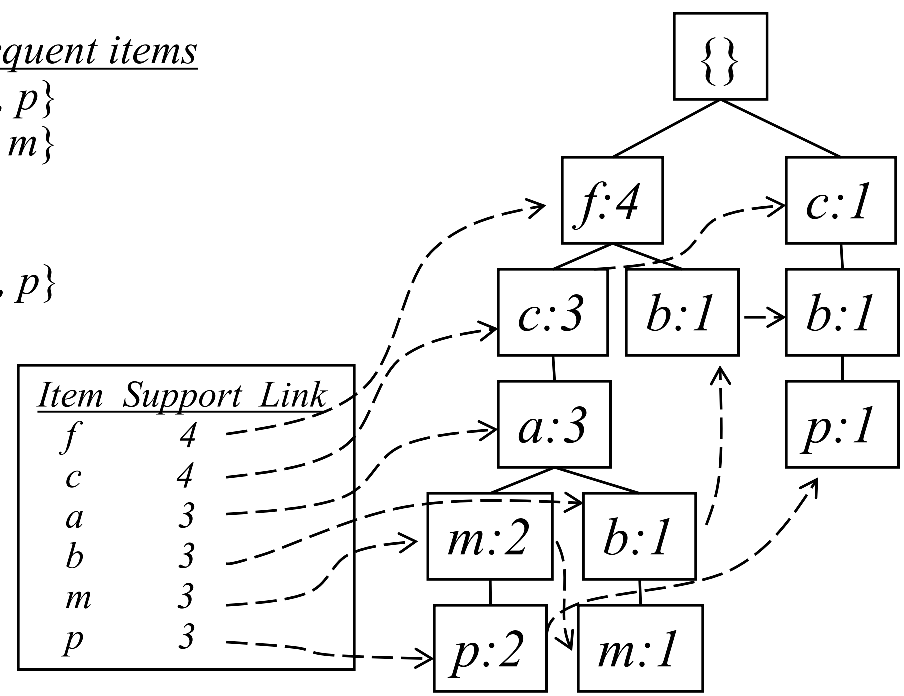
❑ FP-Tree Construction Process

1) Scan DB once to find all frequent 1-itemsets
2) Sort frequent items in a descending order of support, called f-list
3) Scan DB again to construct FP-tree

❑ Example

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



Benefits of FP-Tree Structure

- **Compactness**
  - Remove irrelevant (infrequent) items
  - Reduce common prefix items of patterns
  - Order items in the descending order of support

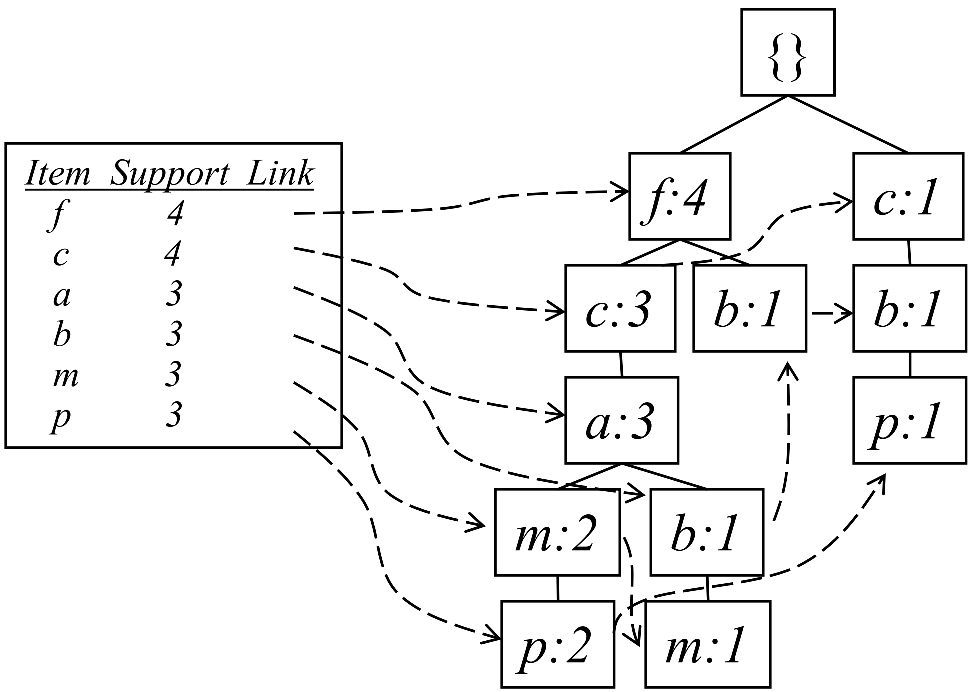
    → The more frequently occurring, the more likely to be shared.
  - Never be larger than the original database

- **Completeness**
  - Preserve complete information of frequent patterns
  - Never break any long patterns

Conditional Pattern Bases

- **Conditional Pattern Base Construction Process**
  1) Traverse the FP-tree by following the link of each frequent item p
  2) Accumulate all prefix paths of p to form p's conditional pattern base

- **Example**



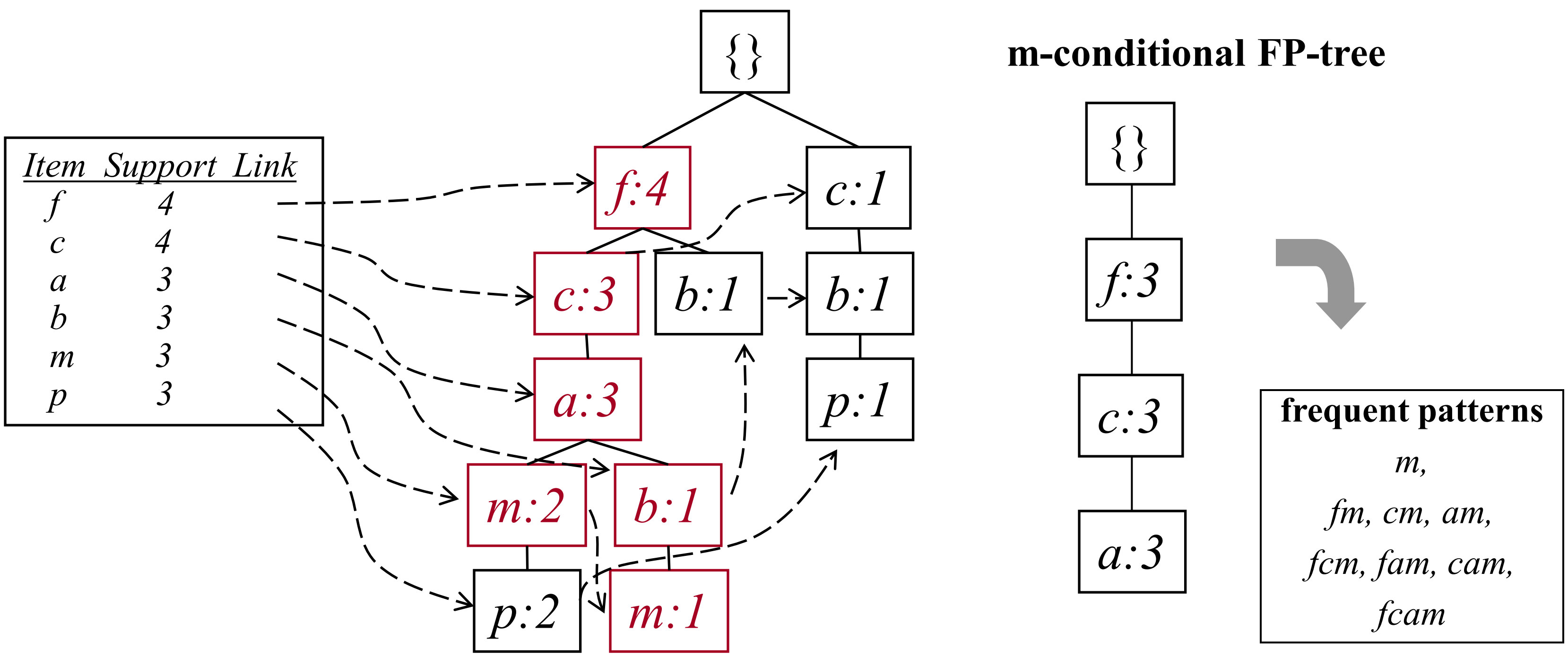
item	*conditional pattern base*
f	-
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

Conditional FP-Trees

❑ Conditional FP-Tree Construction Process

- For each pattern base,
  1) Accumulate the count for each item
  2) Construct the conditional FP-tree with frequent items of the pattern base

❑ Example



Pattern Growth Mining Algorithm

❑ Algorithm

1) Construct FP tree
2) For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
3) Repeat (2) recursively on each newly created conditional FP-tree until the resulting FP-tree is empty, or it contains only a single path
4) The single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

❑ Advanced Techniques

- To fit an FP-tree in memory, partitioning a database into a set of projected databases
- Efficient mining of the FP-tree for each projected database

Overview

1. **Market Basket Problem**

2. **Apriori Algorithm**

3. **CHARM Algorithm**

4. **Advanced Frequent Pattern Mining**

5. **Constraint-Based Mining**

Constraint-based Mining

- **Motivation**
  - Finding all the patterns (association rules) in a database?
    → Too many, diverse patterns
  - Users can give directions (constraints) for mining patterns

- **Features**
  - User flexibility
    - Users can provide any constraints on what to be mined
  - System optimization
    - It reduces the search space for efficient mining

Constraint Types

- **Knowledge Type Constraints**
  - Association, Classification, etc.
- **Data Constraints**
  - Selecting data having specific values using SQL-like queries
- **Dimension/Level Constraints**
  - Selecting specific dimensions or levels of the concept hierarchies
- **Interestingness Constraints**
  - Using interestingness measures, ex, support, confidence, coverage, lift, correlation
- **Rule Constraints**
  - Specifying rules to be mined

Rule Constraint Types

- **Anti-monotonic Constraints**
  - If a constraint c is violated, then its further mining is terminated
- **Monotonic Constraints**
  - If a constraint c is satisfied, then its further mining is redundant
- **Succinct Constraints**
  - The itemsets satisfying a constraint c can be directly generated
- **Convertible Constraints**
  - A constraint c is not monotonic nor anti-monotonic, but it can be converted if items are properly ordered

Anti-Monotonicity in Constraints

- **Definition**
  - For an anti-monotonic constraint c,
    if an itemset S violates c, so does any of its supersets.
    if a pattern satisfies c, all of its sub-patterns satisfy c too.

- **Examples**
  - count(S) < 3 → Anti-monotonic
  - count(S) ≥ 4 → **Not** anti-monotonic
  - sum(S.price) ≤ 100 → Anti-monotonic
  - sum(S.price) ≥ 150 → **Not** anti-monotonic
  - sum(S.profit) ≤ 80 → **Not** anti-monotonic
  - support(S) ≥ 2 → Anti-monotonic

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Price	Profit
a	100	40
b	50	0
c	60	-20
d	80	10
e	100	-30
f	70	30
g	95	20
h	100	-10

Monotonicity in Constraints

Definition

- For a monotonic constraint c,

  if an itemset S satisfies c, so does any of its supersets.

  if a pattern satisfies c, all of its super-patterns satisfy c too.

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Examples

- count(S) $\geq$ 2 → Monotonic
- sum(S.price) $\leq$ 100 → **Not** monotonic
- sum(S.price) $\geq$ 150 → Monotonic
- sum(S.profit) $\geq$ 100 → **Not** monotonic
- min(S.price) $\leq$ 80 → Monotonic
- min(S.price) $>$ 70 → **Not** monotonic

Item	Price	Profit
a	100	40
b	50	0
c	60	-20
d	80	10
e	100	-30
f	70	30
g	95	20
h	100	-10

Converting Constraints

- **Definition**
  - A constraint c is convertible,
    if c is not anti-monotonic nor monotonic,
    but c becomes anti-monotonic or monotonic
    when items are properly ordered.
  - Anti-monotonic convertible if …
  - Monotonic convertible if …
  - Strongly convertible if …

- **Examples**
  - avg(S.price) > 80
    → Neither anti-monotonic nor monotonic
    → if items are in a value-descending order
    <a, e, h, g, d, f, c, b>, then anti-monotonic
    → if items are in a value-ascending order
    <b, c, f, d, g, a, e, h>, then monotonic
    → strongly convertible

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Price	Profit
a	100	40
b	50	0
c	60	-20
d	80	10
e	100	-30
f	70	30
g	95	20
h	100	-10

Anti-Monotonic Constraints in Apriori

❑ Handling Anti-monotonic Constraints

- Can apply apriori pruning
- Example: sum(S.price) < 5

T-ID	Items
10	1, 3, 4
20	2, 3, 5
30	1, 2, 3, 5
40	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~~

C_2

Itemset	Sup.
{1,2}	1
{1,3}	2
{2,3}	2

L_2

Itemset	Sup.
{1,3}	2
~~{2,3}~~	~~2~~

Monotonic Constraints in Apriori

❑ Handling Monotonic Constraints

- Cannot apply apriori pruning
- Example: sum(S.price) ≥ 3

T-ID	Items
10	1, 3, 4
20	1, 2, 3
30	1, 2, 3, 5
40	2, 5

C_1

Itemset	Sup.
{1}	3
{2}	3
{3}	3
{4}	1
{5}	2

L_1

Itemset	Sup.
~~{1}~~	~~3~~
~~{2}~~	~~3~~
{3}	3
{5}	2

C_2

Itemset	Sup.
{1,2}	2
{1,3}	3
{1,5}	1
{2,3}	2
{2,5}	2
{3,5}	1

L_2

Itemset	Sup.
{1,2}	2
{1,3}	2
{2,3}	3
{2,5}	2

C_3

Itemset	Sup.
{1,2,3}	2

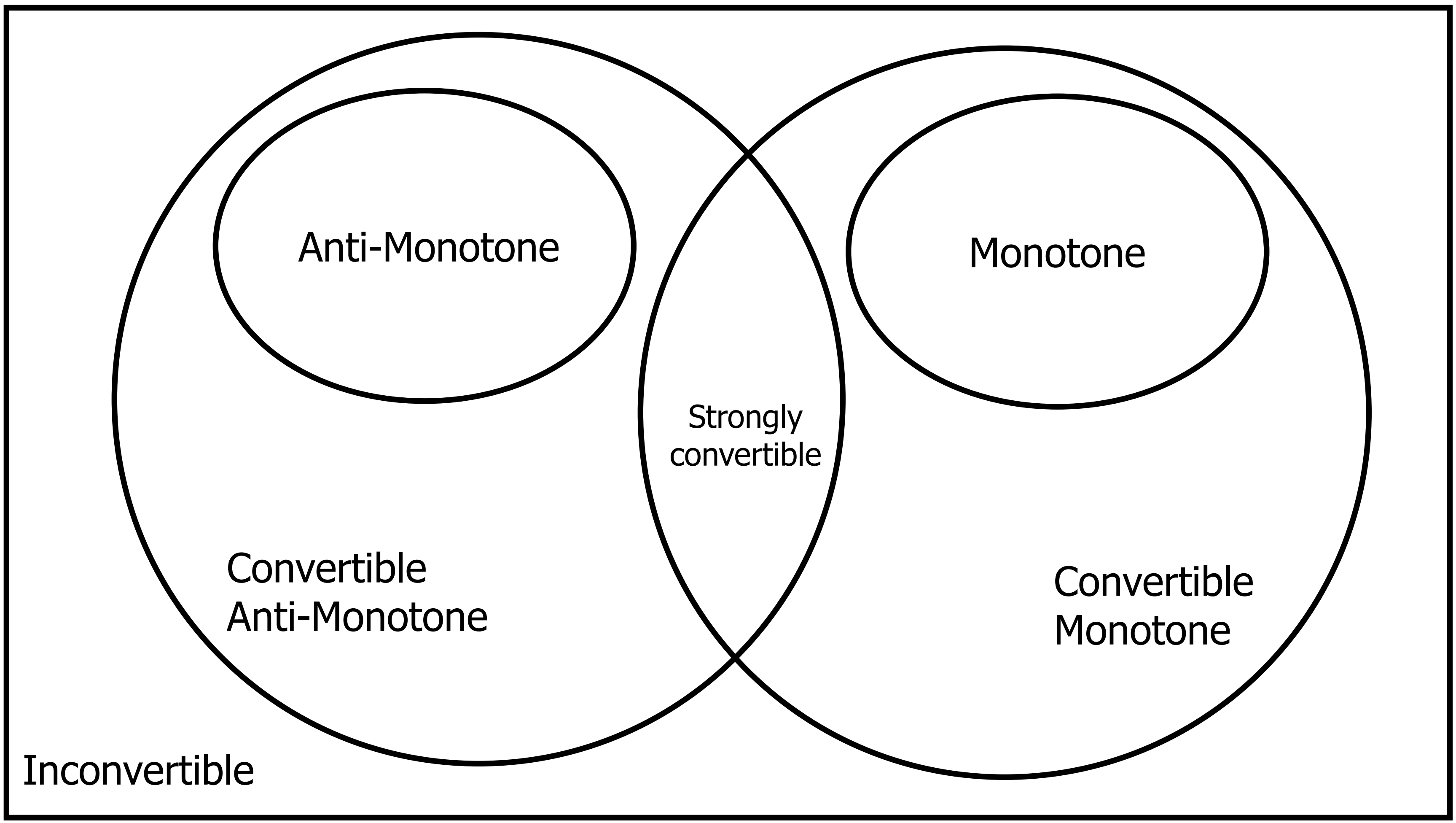
L_3

Itemset	Sup.
{1,2,3}	2

Examples of Constraints

Constraint	Anti-Monotone	Monotone
$v \in S$	no	yes
$S \supseteq V$	no	yes
$S \subseteq V$	yes	no
$\min(S) \le v$	no	yes
$\min(S) \ge v$	yes	no
$\max(S) \le v$	yes	no
$\max(S) \ge v$	no	yes
$\text{count}(S) \le v$	yes	no
$\text{count}(S) \ge v$	no	yes
$\text{sum}(S) \le v$ ($a \in S, a \ge 0$)	yes	no
$\text{sum}(S) \ge v$ ($a \in S, a \ge 0$)	no	yes
$\text{range}(S) \le v$	yes	no
$\text{range}(S) \ge v$	no	yes
$\text{avg}(S)\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	convertible	convertible
$\text{support}(S) \ge \xi$	yes	no
$\text{support}(S) \le \xi$	no	yes

Scope of Constraints



Questions?

- Lecture Slides on the Course Website, "https://it.yonsei.ac.kr/adslab/faculty/data_mining"