Chapter 6: Mining Frequent Patterns, Association and Correlations

- **Basic concepts**
- Frequent itemset mining methods
- Constraint-based frequent pattern mining (ch7)
- Association rules

What Is Frequent Pattern Analysis?

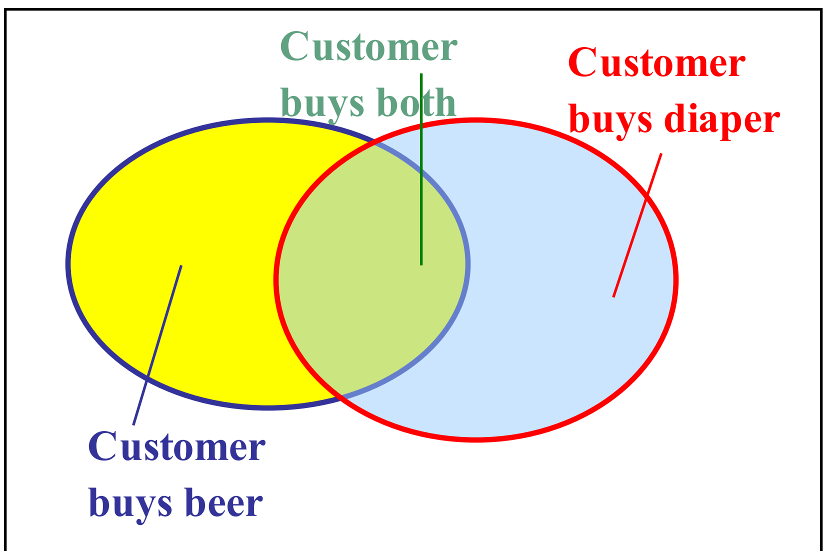
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

- Freq. pattern: intrinsic and important property of data sets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- itemset: A set of items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is a closed pattern if X is *frequent* and there exist *no super-patterns with the same support*
  - all super-patterns must have smaller support
- An itemset X is a max-pattern if X is frequent and there exist no super-patterns that are frequent
- Relationship between the two?
- Closed patterns are a lossless compression of freq. patterns, whereas max-patterns are a lossy compression
  - Lossless: can derive all frequent patterns as well as their support
  - Lossy: can derive all frequent patterns

Closed Patterns and Max-Patterns

- DB = {$<a_1, \ldots, a_{100}>$, $< a_1, \ldots, a_{50}>$}
  - min_sup = 1
- What is the set of closed patterns?
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
  - How to derive frequent patterns and their support values?
- What is the set of max-patterns?
  - $<a_1, \ldots, a_{100}>$: 1
  - How to derive frequent patterns?
- What is the set of all patterns?
  - $\{a_1\}$: 2, …, $\{a_1, a_2\}$: 2, …, $\{a_1, a_{51}\}$: 1, …, $\{a_1, a_2, \ldots, a_{100}\}$: 1
  - A big number: $2^{100} - 1$

Closed Patterns and Max-Patterns

For a given dataset with itemset I = {a,b,c,d} and min_sup = 8, the closed patterns are {a,b,c,d} with support of 10, {a,b,c} with support of 12, and {a, b,d} with support of 14. Derive the frequent 2-itemsets together with their support values

{a,b}: 14	{a,c}: 12	{a,d}: 14
{b,c}: 12	{b,d}: 14	{c,d}: 10

Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic concepts
- **Frequent itemset mining methods**
- Constraint-based frequent pattern mining (ch7)
- Association rules

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach

- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format

Scalable Methods for Mining Frequent Patterns

- The downward closure (anti-monotonic) property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB' 94)
  - Freq. pattern growth (Fpgrowth: Han, Pei & Yin @SIGMOD'00)
  - Vertical data format (Charm—Zaki & Hsiao @SDM' 02)

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is any itemset that is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

min_sup= 2

DB

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 (Pseudo-code)

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
C_{k+1} = candidates generated from L_k;
for each transaction t in database do
increment the count of all candidates in C_{k+1} that are contained in t
L_{k+1} = candidates in C_{k+1} with min_support
end
return $\cup_k L_k$;

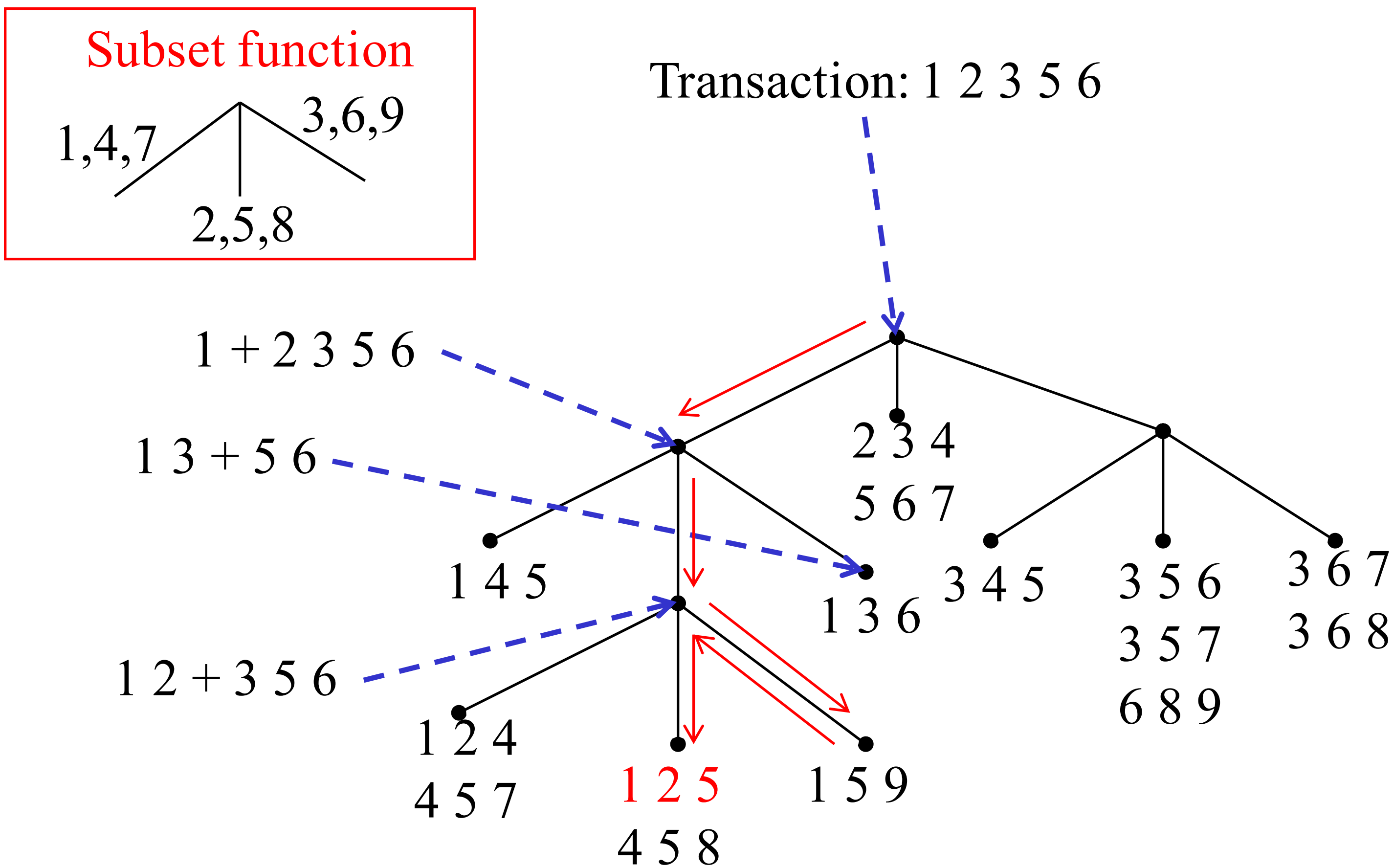
Implementation of Apriori

- Generate candidates, then count support for the generated candidates
- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example:
  - 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*}
- The above procedures do not miss any legitimate candidates. Thus Apriori mines a complete set of frequent patterns.

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Example: Counting Supports of Candidates



Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Apriori applications beyond freq. pattern mining

- Given a set S of students, we want to find each subset of S such that the age range of the subset is less than 5.
  - Apriori algorithm, level-wise search using the downward closure property for pruning to gain efficiency

- Can be used to search for any subsets with the downward closure property (i.e., anti-monotone constraint)

- CLIQUE for subspace clustering used the same Apriori principle, where the one-dimensional cells are the items

Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Frequent itemset mining methods
- **Constraint-based frequent pattern mining (ch7)**
- Association rules

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - Optimization: explores such constraints for efficient mining — constraint-based mining: constraint-pushing, similar to push selection first in DB query processing
  - Note: still find all the answers satisfying constraints, not finding some answers in "heuristic search"

Constrained Mining vs. Constraint-Based Search

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - It is an interesting research problem on how to integrate them
- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing

Constraint-Based Frequent Pattern Mining

- Pattern space pruning constraints
  - Anti-monotonic: If constraint c is violated, its further mining can be terminated
  - Monotonic: If c is satisfied, no need to check c again
  - Succinct: c must be satisfied, so one can start with the data sets satisfying c
  - Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered
- Data space pruning constraint
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining

Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity
  - *When an itemset S* ***violates*** *the constraint, so does any of its superset*
  - *sum(S.Price)* ≤ *v* is anti-monotonic
  - *sum(S.Price)* ≥ *v* is not anti-monotonic
- C: range(S.profit) ≤ 15 is anti-monotonic
  - Itemset *ab* violates C
  - So does every superset of *ab*
- support count >= min_sup is anti-monotonic
  - core property used in Apriori

TDB (min_sup=2)

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	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Monotonicity for Constraint Pushing

- Monotonicity
  - *When an itemset S* ***satisfies*** *the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq$ *v* is monotonic
  - *min(S.Price)* $\leq$ *v* is monotonic
- C: range(S.profit) $\geq$ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Succinctness

- Given A_1, the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1
- Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items
- If a constraint is *succinct*, we can directly generate precisely the sets that satisfy it, even before support counting begins.
- Avoids substantial overhead of generate-and-test,
  - i.e., such constraint is pre-counting pushable

- $min(S.Price) \le v$ is succinct
- $sum(S.Price) \ge v$ is not succinct

Constraint-Based Mining—A General Picture

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

Chapter 6: Mining Frequent Patterns, Association and Correlations

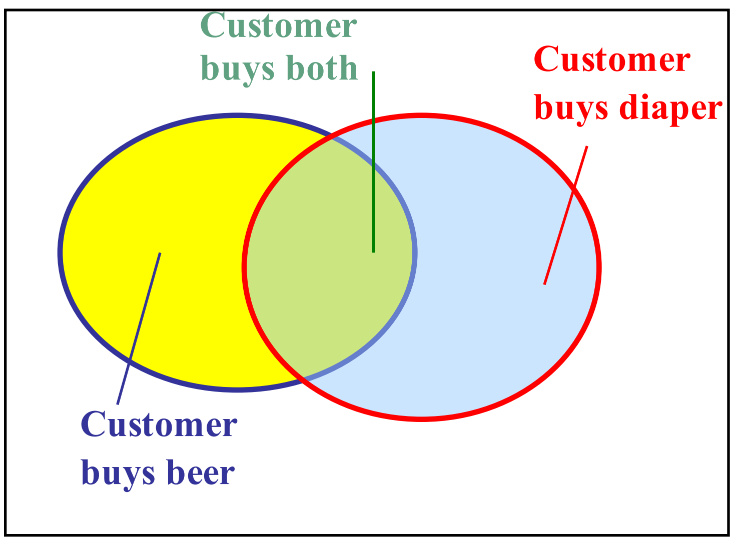
- Basic concepts
- Frequent itemset mining methods
- Constraint-based frequent pattern mining (ch7)
- **Association rules**

Basic Concepts: Association Rules

- An association rule is of the form $X \rightarrow Y$, where $X, Y \subset I$, $X \cap Y = \phi$
  - A rule is **strong** if it satisfies both support and confidence thresholds.
- support(X->Y): probability that a transaction contains $X \cup Y$, i.e., support(X->Y) = P(X U Y)
  - Can be estimated by the percentage of transactions in DB that contain $X \cup Y$. Not to be confused with P(X or Y)
- confidence(X->Y): conditional probability that a transaction having X also contains Y, i.e. confidence(X->Y) = P(Y|X)
  - confidence(X->Y) = P(Y|X) = support($X \cup Y$) / support (X) = support_count($X \cup Y$) / support_count(X)
- confidence(X->Y) can be easily derived from the support count of X and the support count of $X \cup Y$. Thus association rule mining can be reduced to frequent pattern mining

Basic Concepts: Association rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

If {a} => {b} is an association rule, then {b} => {a} is also an association rule?

- Same support, different confidence

If {a,b} => {c} is an association rule, then {b} => {c} is also an association rule?

If {b} => {c} is an association rule then {a,b} => {c} is also an association rule?

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
  - *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
  - Support and confidence are not good to indicate correlations
- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000*3750/5000} = 0.89$$

$$lift(B,\neg C) = \frac{1000/5000}{3000/5000*1250/5000} = 1.33$$