Data Mining:

Concepts and Techniques

(3rd ed.)

Slides slightly adapted.

— Chapter 6 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University
©2013 - 2018 Han, Kamber & Pei. All rights reserved.

Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?

  — Pattern Evaluation Methods (next week)
- Summary

What Is Pattern Discovery?

- What are patterns?
  - A set of items, subsequences, or substructures, that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets
- Pattern discovery
  - Uncovering patterns from massive data sets

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a notebook?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

Beer and Diapers

A 26-year-old legend:
beer and diaper sales spike
between the hours of 5 p.m. and 7 p.m.

Original NCR (now TeraData) study in 1992 by Thomas Blischok (MindMeld Inc.) for American retail chain Osco Drugs.

- examined 1.2 million market baskets in 25 stores
- NCR identified 30 different shopping experiences, such as a correlation between fruit juice and cough medication sales.
- Osco removed approximately 5,000 slow-moving items from its inventory
- by re-arranging merchandise, consumers actually thought that Osco's selection had increased.
- putting the right merchandise in the right quantities at the right time

Why Is Pattern Discovery Important?

- Finding inherent regularities in a data set
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative pattern-based analysis
  - Cluster analysis: pattern-subspace clustering
- Many Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log analysis, and biological sequence analysis

5

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

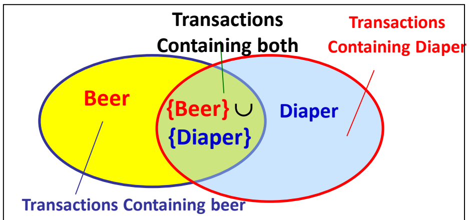
- Let *minsup = 50%*
- Freq. 1-itemsets:
  - Beer: 3 (60%); Nuts: 3 (60%)
  - Diaper: 4 (80%); Eggs: 3 (60%)
- Freq. 2-itemsets:
  - {Beer, Diaper}: 3 (60%)

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

6

From Frequent Itemsets to 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



{Beer} ∪ {Diaper} = {Beer, Diaper}

Note: Itemset: X ∪ Y, a subtle notation!

- Association rules: $X \rightarrow Y$ *with* (s, c)
  - Support, *s*: The probability that a transaction contains X ∪ Y
  - Confidence, *c: The* conditional probability that a transaction containing X also contains *Y*
  - c = sup(X ∪ Y) / sup(X)
- **Association rule mining**: Find all of the rules, $X \rightarrow Y$, with minimum support and confidence
- Frequent itemsets: Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3
- Association rules: Let *minconf = 50%*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain?
  - $TDB_{1:}$ T_1: {a_1, …, a_{50}}; T_2: {a_1, …, a_{100}}
  - Assuming (absolute) *minsup* = 1
  - Let's have a try

  1-itemsets: {a_1}: 2, {a_2}: 2, …, {a_{50}}: 2, {a_{51}}: 1, …, {a_{100}}: 1,

  2-itemsets: {a_1, a_2}: 2, …, {a_1, a_{50}}: 2, {a_1, a_{51}}: 1 …, …, {a_{99}, a_{100}}: 1,

  …, …, …, …

  99-itemsets: {a_1, a_2, …, a_{99}}: 1, …, {a_2, a_3, …, a_{100}}: 1

  100-itemset: {a_1, a_2, …, a_{100}}: 1

  - In total: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1$ sub-patterns!

Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- **Solution 1: Closed patterns**: A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ɔ X, *with the same support* as X
  - Let Transaction DB TDB_1: T_1: $\{a_1, …, a_{50}\}$; T_2: $\{a_1, …, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, …, a_{50}\}$: 2"; P_2: "$\{a_1, …, a_{100}\}$: 1"
- Closed pattern is a lossless compression of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say: "$\{a_2, …, a_{40}\}$: 2", "$\{a_5, a_{51}\}$: 1"

Expressing Patterns in Compressed Form: Max-Patterns

- **Solution 2: Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X
- Difference from close-patterns?
  - Do not capture the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: $\{a_1, …, a_{50}\}$; T_2: $\{a_1, …, a_{100}\}$
  - Suppose *minsup* = 1. How many max-patterns does TDB_1 contain?
    - One: P: "$\{a_1, …, a_{100}\}$: 1"
- Max-pattern is a lossy compression!
  - We only know $\{a_1, …, a_{40}\}$ is frequent
  - But we do not know the real support of $\{a_1, …, a_{40}\}$, …, any more!
- Thus in many applications, mining close-patterns is more desirable than mining max-patterns

Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?

  — Pattern Evaluation Methods (next week)
- Summary

11

Scalable Frequent Itemset Mining Methods

- The Downward Closure Property of Frequent Patterns
- The Apriori Algorithm
- Extensions or Improvements of Apriori
- Mining Frequent Patterns by Exploring Vertical Data Format
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Closed Patterns

12

The Downward Closure Property of Frequent Patterns

- Observation: From $TDB_{1:}$ T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - We get a frequent itemset: $\{a_1, \ldots, a_{50}\}$
  - Also, its subsets are all frequent: $\{a_1\}, \{a_2\}, \ldots, \{a_{50}\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_{49}\}, \ldots$
  - There must be some hidden relationships among frequent patterns!
- The downward closure (also called "Apriori") property of frequent patterns
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction containing {beer, diaper, nuts} also contains {beer, diaper}
  - Apriori: Any subset of a frequent itemset must be frequent
- Efficient mining methodology
  - If any subset of an itemset S is infrequent, then there is no chance for S to be frequent—why do we even have to consider S!?

A sharp knife for pruning!

Apriori Pruning and Scalable Mining Methods

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not even be generated! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Scalable mining Methods: Three major approaches
  - Level-wise, join-based approach: Apriori (Agrawal & Srikant@VLDB'94)
  - Vertical data format approach: Eclat (Zaki, Parthasarathy, Ogihara, Li @KDD'97)
  - Frequent pattern projection and growth: FPgrowth (Han, Pei, Yin @SIGMOD'00)

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Outline of Apriori (level-wise, candidate generation and test)
  - Initially, scan DB once to get frequent 1-itemset
  - Repeat
    - Generate length-(k+1) candidate itemsets from length-k frequent itemsets
    - Test the candidates against DB to find frequent (k+1)-itemsets
    - Set k := k +1
  - Until no frequent or candidate set can be generated
  - Return all the frequent itemsets derived

15

The Apriori Algorithm—An Example

minsup = 2

Database TDB

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

C_1

1st scan

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

16

The Apriori Algorithm (Pseudo-Code)

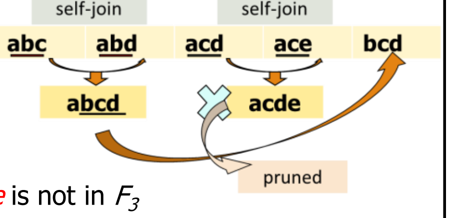
C_k: Candidate itemset of size k
F_k : frequent itemset of size k

```
k := 1;
F_1 = {frequent items};
while ( F_k !=∅ ) do
    C_{k+1} = candidates generated from F_k ;
    for each transaction t in database do
        increment the count of all candidates in C_{k+1} that are contained
         in t ;
    F_{k+1}  = candidates in C_{k+1} with min_support
    k := k + 1;
    od
return ∪_k F_k ;
```

17

Implementation of Apriori

- How to generate candidates?
  - Step 1: self-joining F_k
  - Step 2: pruning
- Example of Candidate-generation
  - F_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: F_3*F_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in F_3
  - C_4 = {*abcd*}



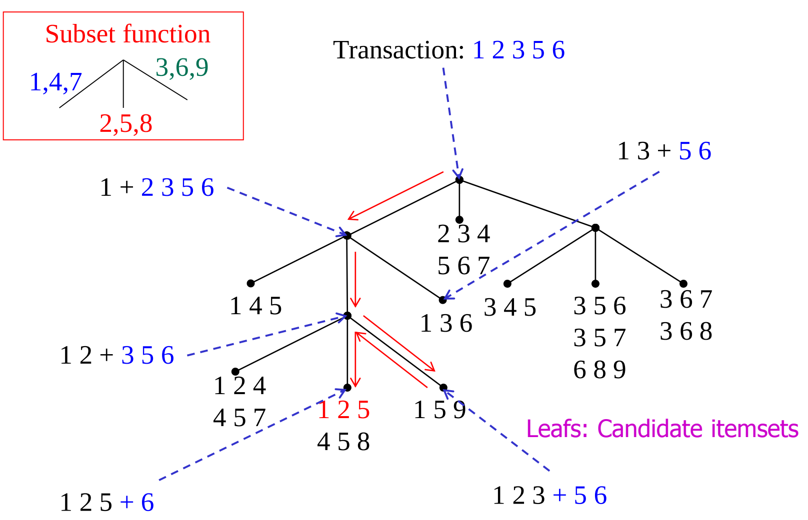
18

How to Count Supports of Candidates?

- Why is 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

Counting Supports of Candidates Using Hash Tree

Items: 1, 2, 3, 4, 5, 6, 7, 8, 9



Candidate Generation: An SQL Implementation

- SQL Implementation of candidate generation
  - Suppose the items in F_{k-1} are listed in an order
  - Step 1: **self-joining** F_{k-1}

    insert into C_k

    select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

    from $F_{k-1}\ p, F_{k-1}\ q$

    where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
  - Step 2: **pruning**

    forall *itemsets c in* C_k do

        forall *(k-1)-subsets s of c* do

            if *(s is not in* F_{k-1}*)* then delete *c* from C_k

- Use object-relational extensions like UDFs, BLOBs, and Table functions for efficient implementation [S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98]

Scalable Frequent Itemset Mining Methods

- The Downward Closure Property of Frequent Patterns
- The Apriori Algorithm
- Extensions or Improvements of Apriori
- Mining Frequent Patterns by Exploring Vertical Data Format
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Closed Patterns

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

23

Apriori: Improvements and Alternatives

- Reduce passes of transaction database scans
  - Partitioning (e.g., Savasere, et al., 1995)
  - Dynamic itemset counting (Brin, et al., 1997)

To be discussed in subsequent slides

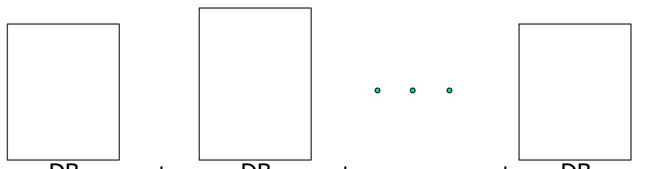
- Shrink the number of candidates
  - Hashing (e.g., DHP: Park, et al., 1995)
  - Pruning by support lower bounding (e.g., Bayardo 1998)
  - Sampling (e.g., Toivonen, 1996)

To be discussed in subsequent slides

- Exploring special data structures
  - Tree projection (Agarwal, et al., 2001)
  - H-miner (Pei, et al., 2001)
  - Hypercube decomposition (e.g., LCM: Uno, et al., 2004)

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*



DHP: Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the support threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries
    - {ab, ad, ae}
    - {bd, be, de}
    - …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below the support threshold
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95* (Direct Hashing and Pruning (DHP))

Itemsets	Count
{ab, ad, ae}	35
{bd, be, de}	298
……	…
{yz, qs, wt}	58

Hash Table

26

Exploring Vertical Data Format: ECLAT

- ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- Tid-List: List of transaction-ids containing the itemset(s)
- Vertical format: $t(e) = \{T_{10}, T_{20}, T_{30}\}$; $t(a) = \{T_{10}, T_{20}\}$; $t(ae) = \{T_{10}, T_{20}\}$
- Properties of Tid-Lists
  - $t(X) = t(Y)$: X and Y always happen together (e.g., $t(ac) = t(d)$)
  - $t(X) \subset t(Y)$: transaction having X always has Y (e.g., $t(ac) \subset t(ce)$)
- Deriving frequent patterns based on vertical intersections
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(e) = \{T_{10}, T_{20}, T_{30}\}$, $t(ce) = \{T_{10}, T_{30}\}$ → Diffset $(ce, e) = \{T_{20}\}$

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

Item	Tid-List
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

Sampling for Frequent Patterns

- Select a sample of the original database, mine frequent patterns within the sample using Apriori
- Scan database once to verify frequent itemsets found in sample. Here only *borders* of closure of frequent patterns are checked:
  - Example: check *abcd* instead of *ab, ac, …, etc. (why?)*
- Scan database again to find missed frequent patterns.
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

Frequent Itemset Mining

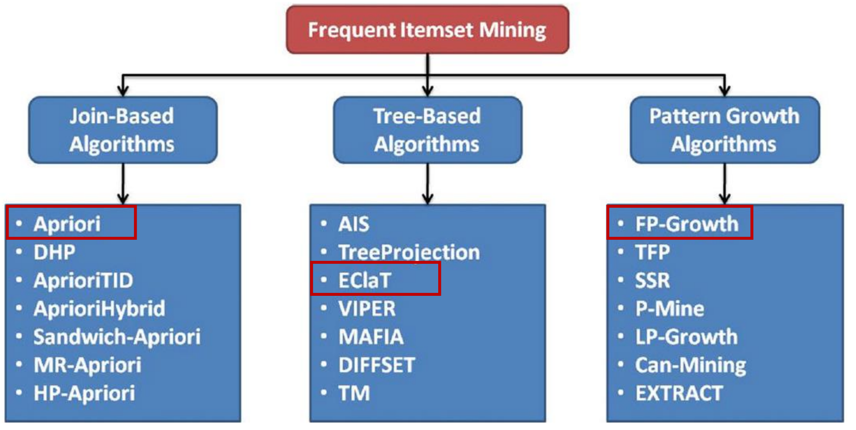


Fig. 12 Classification of Frequent Pattern Mining algorithms

Figure from: C. Chin-Hoong et al., Algorithms for frequent itemset mining: a literature review", Artificial Intelligence Review, Springer, 2018

Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

References (II) Efficient Pattern Mining Methods

- R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases", VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules", SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating association rule mining with relational database systems: Alternatives and implications", SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "Parallel algorithm for discovery of association rules", Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", SIGMOD'00
- M. J. Zaki and Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", SDM'02
- J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, "Frequent Pattern Mining Algorithms: A Survey", in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014
- C. Chin-Hoong, J. Jafreezal, A.Izzatdin Abdul, H. Mohd Hilmi, Y. William, Algorithms for frequent itemset mining: a literature review", Artificial Intelligence Review, Springer, 2018

References (III) Pattern Evaluation

- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010