Ludwig-Maximilians-Universität München
Lehrstuhl für Datenbanksysteme und Data Mining
Prof. Dr. Thomas Seidl

Knowledge Discovery and Data Mining I

Winter Semester 2018/19

Agenda

What is Frequent Pattern Mining?

Setting: Transaction Databases

A database of transactions, where each transaction comprises a set of items, e.g. one transaction is the basket of one customer in a grocery store.

Frequent Pattern Mining

Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

Applications

Basket data analysis, cross-marketing, catalogue design, loss-leader analysis, clustering, classification, recommendation systems, etc.

What is Frequent Pattern Mining?

Task 1: Frequent Itemset Mining

Find all subsets of items that occur together in many transactions.

Example

Which items are bought together frequently?

$$\begin{aligned} D = \{ \quad & \{ \quad \mathit{butter}, \mathit{bread}, \mathit{milk}, \mathit{sugar}\}, \\ & \{ \quad \mathit{butter}, \mathit{flour}, \mathit{milk}, \mathit{sugar}\}, \\ & \{ \quad \mathit{butter}, \mathit{eggs}, \mathit{milk}, \mathit{salt}\}, \\ & \{ \quad \mathit{eggs}\}, \\ & \{ \quad \mathit{butter}, \mathit{flour}, \mathit{milk}, \mathit{salt}, \mathit{sugar}\}\} \end{aligned}$$

$\rightsquigarrow$ 80% of transactions contain the itemset {milk, butter}

What is Frequent Pattern Mining?

Task 2: Association Rule Mining

Find all rules that correlate the presence of one set of items with that of another set of items in the transaction database.

Example

98% of people buying tires and auto accessories also get automotive service done

Agenda

Mining Frequent Itemsets: Basic Notions

- **Items** $I = \{i_1, \ldots, i_m\}$: a set of literals (denoting items)
- **Itemset** X: Set of items $X \subseteq I$
- **Database** D: Set of *transactions* T, each transaction is a set of items $T \subseteq I$
- Transaction T contains an itemset X: $X \subseteq T$
- **Length** of an itemset X equals its cardinality $|X|$
- k-**itemset**: itemset of length k
- (Relative) **Support** of an itemset: $supp(X) = |\{T \in D \mid X \subseteq T\}|/|D|$
- X is **frequent** if $supp(X) \geq minSup$ for threshold $minSup$.

Goal

Given a database D and a threshold $minSup$, find all frequent itemsets $X \in Pot(I)$.

Mining Frequent Itemsets: Basic Idea

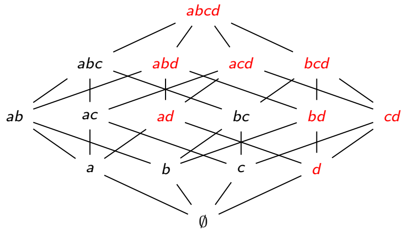
Naïve Algorithm

Count the frequency of all possible subsets of I in the database D.

Problem

Too expensive since there are 2^m such itemsets for m items (for $|I| = m$, $2^m =$ cardinality of the powerset of I).

Mining Frequent Patterns: Apriori Principle



- frequent
- non-frequent

Apriori Principle (anti-monotonicity)

- Any non-empty subset of a frequent itemset is frequent, too!

$$A \subseteq I : supp(A) \geq minSup \implies \forall \emptyset \neq A' \subset A : supp(A') \geq minSup$$

- Any superset of a non-frequent itemset is non-frequent, too!

$$A \subseteq I : supp(A) < minSup \implies \forall A' \supset A : supp(A') < minSup$$

Apriori Algorithm

Idea

- First count the 1-itemsets, then the 2-itemsets, then the 3-itemsets, and so on
- When counting $(k+1)$-itemsets, only consider those $(k+1)$-itemsets where all subsets of length k have been determined as frequent in the previous step

Apriori Algorithm

variable C_k: candidate itemsets of size k
variable L_k: frequent itemsets of size k
$L_1 = \{\text{frequent items}\}$
for $(k = 1;\ L_k \neq \emptyset;\ k{+}{+})$ **do**

Produce candidates.

join L_k with itself to produce C_{k+1} ▷ JOIN STEP
discard $(k+1)$-itemsets from C_{k+1} that ... ▷ PRUNE STEP
...contain non-frequent k-itemsets as subsets

C_{k+1} = candidates generated from L_k

Prove candidates.

for each transaction $T \in D$ **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 *minSupp*

return $\bigcup_k L_k$

Apriori Algorithm: Generating Candidates – Join Step

Requirements for Candidate $(k+1)$-itemsets

- *Completeness*: Must contain all frequent $(k+1)$-itemsets (superset property $C_{k+1} \supseteq L_{k+1}$)
- *Selectiveness*: Significantly smaller than the set of all $(k+1)$-subsets

Suppose the itemsets are sorted by any order (e.g. lexicographic)

Step 1: Joining ($C_{k+1} = L_k \bowtie L_k$)

- Consider frequent k-itemsets p and q
- p and q are joined if they share the same first $(k-1)$ items.

Apriori Algorithm: Generating Candidates – Join Step

Example

- $k = 3 \ (\implies k + 1 = 4)$
- $p = (a, c, f) \in L_k$
- $q = (a, c, g) \in L_k$
- $r = (a, c, f, g) \in C_{k+1}$

SQL example

insert into C_{k+1}

select $p.i_1, p.i_2, \ldots, p.i_k, q.i_k$

from $L_k : p, L_k : q$

where $p.i_1 = q.i_1, \ldots, p.i_{k-1} = q.i_{k-1}, p.i_k < q.i_k$

Apriori Algorithm: Generating Candidates – Prune Step

Step 2: Pruning ($L_{k+1} = \{X \in C_{k+1} \mid supp(X) \geq minSup\}$)

- *Naïve*: Check support of every itemset in C_{k+1} $\rightsquigarrow$ inefficient for huge C_{k+1}
- *Better*: Apply Apriori principle first: Remove candidate $(k+1)$-itemsets which contain a non-frequent k-subset s, i.e., $s \notin L_k$

Pseudocode

for all $c \in C_{k+1}$ **do**
    for all k-subsets s of c **do**
        if $s \notin L_k$ **then**
            Delete c from C_{k+1}

Apriori Algorithm: Generating Candidates – Prune Step

Example

- $L_3 = \{acf, acg, afg, afh, cfg\}$
- Candidates after join step: $\{acfg, afgh\}$
- In the pruning step: delete *afgh* because $fgh \notin L_3$, i.e. *fgh* is not a frequent 3-itemset (also $agh \notin L_3$)
- $C_4 = \{acfg\} \rightsquigarrow$ check the support to generate L_4

Apriori Algorithm: Full example

Database

TID	items
0	acdf
1	bce
2	abce
3	aef

minSup = 0.5

Alphabetic Ordering

k	candidate	prune	count	threshold
1	a		3	a
	b		2	b
	c		3	c
	d		1	
	e		3	e
	f		2	f
2	ab		1	
	ac		2	ac
	ae		2	ae
	af		2	af
	bc		2	bc
	be		2	be
	bf		0	
	ce		2	ce
	cf		1	
	ef		1	
3	ace		1	
	acf	with cf		
	aef	with ef		
	bce		2	bce

Frequency-Ascending Ordering

k	candidate	prune	count	threshold
1	d		1	
	b		2	b
	f		2	f
	a		3	a
	c		3	c
	e		3	e
2	bf		0	
	ba		1	
	bc		2	bc
	be		2	be
	fa		2	fa
	fc		1	
	fe		1	
	ac		2	ac
	ae		2	ae
	ce		2	ce
3	bce		2	bce
	ace		1	

Counting Candidate Support

Motivation

Why is counting supports of candidates a problem?

- Huge number of candidates
- One transaction may contain many candidates

Solution

Store candidate itemsets in hash-tree

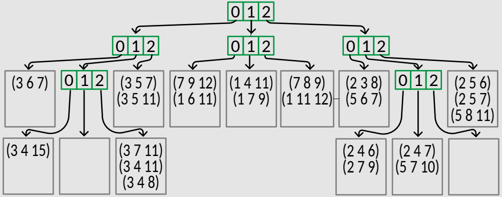
Counting Candidate Support: Hash Tree

Hash-Tree

- Leaves contain itemset lists with their support (e.g. counts)
- Interior nodes comprise hash tables
- *subset* function to find all candidates contained transaction

Example

3-itemsets; $h(i) = i \mod 3$



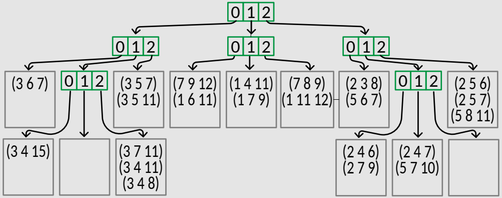
Hash-Tree: Construction

Search

- Start at the root (level 1)
- At level d: Apply hash function h to d-th item in the itemset

Example

3-itemsets; $h(i) = i \mod 3$



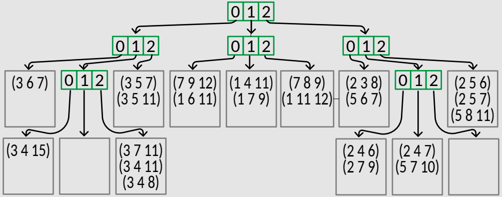
Hash-Tree: Construction

Insertion

- Search for the corresponding leaf node
- Insert the itemset into leaf; if an overflow occurs:
  - Transform the leaf node into an internal node
  - Distribute the entries to the new leaf nodes according to the hash function h

Example

3-itemsets; $h(i) = i \mod 3$



Hash-Tree: Counting

Search all candidates of length k in transaction $T = (t_1, \ldots, t_n)$

- At root:
  - Compute hash values for all items $t_1, \ldots, t_{n-k+1}$
  - Continue search in all resulting child nodes
- At internal node at level d (reached after hashing of item t_i):
  - Determine the hash values and continue the search for each item t_j with $i < j \leq n - k + d$
- At leaf node:
  - Check whether the itemsets in the leaf node are contained in transaction T

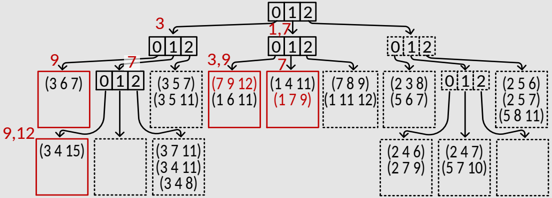
Example

3-itemsets;

$h(i) = i \mod 3$

Transaction:

$\{1, 3, 7, 9, 12\}$



Apriori – Performance Bottlenecks

Huge Candidate Sets

- 10^4 frequent 1-itemsets will generate 10^7 candidate 2-itemsets
- To discover a frequent pattern of size 100, one needs to generate $2^{100} \approx 10^{30}$ candidates.

Multiple Database Scans

- Needs n or $n + 1$ scans, where n is the length of the longest pattern

Is it possible to mine the complete set of frequent itemsets without candidate generation?

Mining Frequent Patterns *Without Candidate Generation*

Idea

- Compress large database into compact tree structure; complete for frequent pattern mining, but avoiding several costly database scans (called *FP-tree*)
- Divide compressed database into *conditional databases* associated with one frequent item

FP-Tree Construction

minSup=0.4

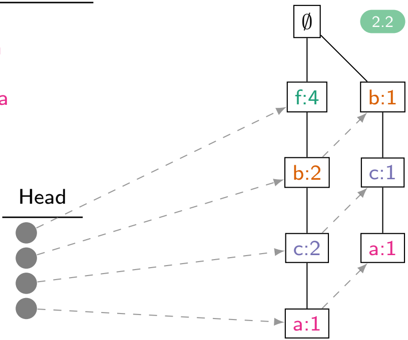
Database

TID	Items	Freq. Item
1	bcfi	fbc
2	abch	bca
3	df	f
4	abcef	fbca
5	fg	f

2.1

Header Table 1

Item	Frequency
f	4
b	3
c	3
a	2



1. Scan DB once, find frequent 1-itemsets (single items); Order frequent items in frequency descending order
2. Scan DB again:
   - 2.1 Keep only freq. items; sort by descending freq.
   - 2.2 Does path with common prefix exist?
     Yes: Increment counter; append suffix;
     No: Create new branch

Benefits of the FP-Tree Structure

Completeness

- never breaks a long pattern of any transaction
- preserves complete information for frequent pattern mining

Compactness

- reduce irrelevant information – infrequent items are gone
- frequency descending ordering: more frequent items are more likely to be shared
- never be larger than the original database (if not count node-links and counts)
- Experiments demonstrate compression ratios over 100

Mining Frequent Patterns Using FP-Tree

General Idea: (Divide-and-Conquer)

Recursively grow frequent pattern path using the FP-tree

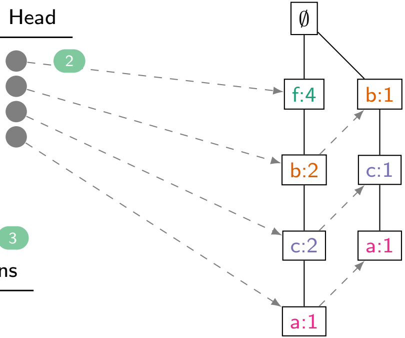
Method

1. Construct conditional pattern base for each node in the FP-tree
2. Construct conditional FP-tree from each conditional pattern-base
3. Recursively mine conditional FP-trees and grow frequent patterns obtained so far; If the conditional FP-tree contains a single path, simply enumerate all the patterns

Major Steps to Mine FP-Tree: Conditional Pattern Base

Header Table 1

Item	Frequency	Head
f	4	
b	3	
c	3	
a	2	



Conditional Pattern 3

Item	Cond. Patterns
f	∅
b	f:2, ∅
c	fb:2, b:1
a	fbc:1, bc:1

1. Start from header table
2. Visit all nodes for this item (following links)
3. Accumulate all transformed prefix paths to form conditional pattern base (the frequency can be read from the node).

Properties of FP-Tree for Conditional Pattern Bases

Node-Link Property

For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.

Prefix Path Property

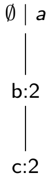
To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P needs to be accumulated, and its frequency count should carry the same count as node a_i.

Major Steps to Mine FP-Tree: Conditional FP-Tree

Item	Cond. Patterns
f	∅
b	f:2, ∅
c	fb:2, b:1
a	fbc:1, bc:1

Example: *a*-conditional FP-Tree

Item	Frequency
b	2
c	2
f	1



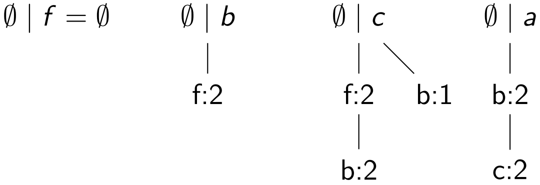
Construct conditional FP-tree from each conditional pattern-base

- The prefix paths of a suffix represent the conditional basis $\rightsquigarrow$ can be regarded as transactions of a database.
- For each pattern-base:
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

Major Steps to Mine FP-Tree: Conditional FP-Tree

- Build conditional FP-Trees for each item

Item	Cond. Patterns
f	∅
b	f:2, ∅
c	fb:2, b:1
a	fbc:1, bc:1



Major Steps to Mine FP-Tree: Recursion

Base Case: Single Path

If the conditional FP-tree contains a single path, simply enumerate all the patterns (enumerate all combinations of sub-paths)

Example

$\emptyset \mid a$

|

b:2 $\rightsquigarrow$

|

c:2

All frequent patterns concerning a:

a,

ab, ac

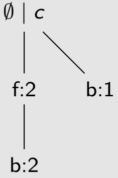
abc

Major Steps to Mine FP-Tree: Recursion

Recursive Case: Non-degenerated Tree

If the conditional FP-tree is not just a single path, create conditional pattern base for this smaller tree, and recurse.

Example



Conditional Pattern Base

Item	Cond. Patterns
f	∅
b	f:2, ∅

$\emptyset \mid fc = \emptyset$

Principles of Frequent Pattern Growth

Pattern Growth Property

Let X be a frequent itemset in D, B be X's conditional pattern base, and Y be an itemset in B. Then $X \cup Y$ is a frequent itemset in D if and only if Y is frequent in B.

Example

"abcdef" is a frequent pattern, if and only if

- "abcde" is a frequent pattern, and
- "f" is frequent in the set of transactions containing "abcde"

Why Is Frequent Pattern Growth Fast?

Performance study[1] shows: FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection

Reasoning:

- No candidate generation, no candidate test (Apriori algorithm has to proceed breadth-first)
- Use compact data structure
- Eliminate repeated database scan
- Basic operation is counting and FP-tree building

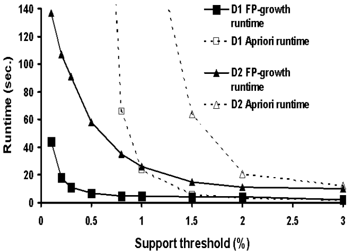


Image Source: [1]

[1] Han, Pei & Yin, *Mining frequent patterns without candidate generation*, SIGMOD'00

Maximal or Closed Frequent Itemsets

Challenge

Often, there is a huge number of frequent itemsets (especially if minSup is set too low), e.g. a frequent itemset of length 100 contains $2^{100} - 1$ many frequent subsets

Closed Frequent Itemset

Itemset X is *closed* in dataset D if for all $Y \supset X : supp(Y) < supp(X)$.

$\Rightarrow$ The set of closed frequent itemsets contains complete information regarding its corresponding frequent itemsets.

Maximal Frequent Itemset

Itemset X is *maximal* in dataset D if for all $Y \supset X : supp(Y) < minSup$.

$\Rightarrow$ The set of maximal itemsets does not contain the complete support information

$\Rightarrow$ More compact representation

Agenda

Simple Association Rules: Introduction

Example

Transaction database:

$$D = \{ \; \{ \; butter, bread, milk, sugar\},$$
$$\{ \; butter, flour, milk, sugar\},$$
$$\{ \; butter, eggs, milk, salt\},$$
$$\{ \; eggs\},$$
$$\{ \; butter, flour, milk, salt, sugar\}\}$$

Frequent itemsets:

items	support
{butter}	4
{milk}	4
{butter, milk}	4
{sugar}	3
{butter, sugar}	3
{milk, sugar}	3
{butter, milk, sugar}	3

Question of interest

- If milk and sugar are bought, will the customer always buy butter as well? milk, sugar $\Rightarrow$ butter?
- In this case, what would be the probability of buying butter?

Simple Association Rules: Basic Notions

Let *Items, Itemset, Database, Transaction, Transaction Length, k-itemset, (relative) Support, Frequent Itemset* be defined as before. Additionally:

- The items in transactions and itemsets are **sorted** lexicographically: itemset $X = (x_1, \ldots, x_k)$, where $x_1 \leq, \ldots, \leq x_k$
- **Association rule**: An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ are two itemsets with $X \cap Y = \emptyset$
- Note: simply enumerating all possible association rules is not reasonable!

What are the interesting association rules w.r.t. D?

Interestingness of Association Rules

Goal

Quantify the interestingness of an association rule with respect to a transaction database D.

Support

- Frequency (probability) of the entire rule with respect to D:

$$supp(X \Rightarrow Y) = P(X \cup Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|D|} = supp(X \cup Y)$$

- "Probability that a transaction in D contains the itemset."

Interestingness of Association Rules

Confidence

- Indicates the strength of implication in the rule:

$$
\begin{aligned}
conf(X \Rightarrow Y) = P(Y \mid X) &= \frac{|\{T \in D \mid X \subseteq T\} \cap \{T \in D \mid Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|} \\
&= \frac{|\{T \in D \mid X \subseteq T \wedge Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|} \\
&= \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|} = \frac{supp(X \cup Y)}{supp(X)}
\end{aligned}
$$

- "Conditional probability that a transaction in D containing the itemset X also contains itemset Y."

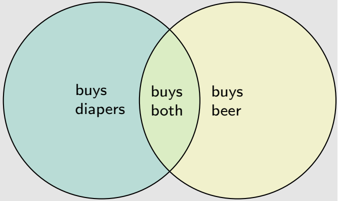
Interestingness of Association Rules

Rule form

"Body $\Rightarrow$ Head [support, confidence]"

Association rule examples

- buys diapers $\Rightarrow$ buys beer [0.5 %, 60%]
- major in CS $\wedge$ takes DB $\Rightarrow$ avg. grade A [1%, 75%]



Mining of Association Rules

Task of mining association rules

Given a database D, determine all association rules having a $supp \geq minSup$ and a $conf \geq minConf$ (so-called *strong association rules*).

Key steps of mining association rules

1. Find frequent itemsets, i.e., itemsets that have $supp \geq minSup$ (e.g. Apriori, FP-growth)
2. Use the frequent itemsets to generate association rules
   - For each itemset X and every nonempty subset $Y \subset X$ generate rule $Y \Rightarrow (X \setminus Y)$ if *minSup* and *minConf* are fulfilled
   - We have $2^{|X|} - 2$ many association rule candidates for each itemset X

Mining of Association Rules

Example

- Frequent itemsets:

1-itemset	count	2-itemset	count	3-itemset	count
{ a }	3	{ a,b }	3	{ a,b,c }	2
{ b }	4	{ a,c }	2		
{ c }	5	{ b,c }	4		

- Rule candidates
  - From 1-itemsets: None
  - From 2-itemsets: $a \Rightarrow b$; $b \Rightarrow a$; $a \Rightarrow c$; $c \Rightarrow a$; $b \Rightarrow c$; $c \Rightarrow b$
  - From 3-itemsets: $a, b \Rightarrow c$; $a, c \Rightarrow b$; $c, b \Rightarrow a$; $a \Rightarrow b, c$; $b \Rightarrow a, c$; $c \Rightarrow a, b$

Generating Rules from Frequent Itemsets

Rule generation

- For each frequent itemset X:
  - For each nonempty subset Y of X, form a rule $Y \Rightarrow (X \setminus Y)$
  - Delete those rules that do not have minimum confidence
- Note:
  - Support always exceeds *minSup*
  - The support values of the frequent itemsets suffice to calculate the confidence
- Exploit anti-monotonicity for generating candidates for strong association rules!
  - $Y \Rightarrow Z$ not strong $\implies$ for all $A \subseteq D : Y \Rightarrow Z \cup A$ not strong
  - $Y \Rightarrow Z$ not strong $\implies$ for all $Y' \subseteq Y$: $(Y \setminus Y') \Rightarrow (Z \cup Y')$ not strong

Generating Rules from Frequent Itemsets

Example: $minConf = 60\%$

$conf(a \Rightarrow b) = 3/3$ ✓
$conf(b \Rightarrow a) = 3/4$ ✓
$conf(a \Rightarrow c) = 2/3$ ✓
$conf(c \Rightarrow a) = 2/5$ ✗
$conf(b \Rightarrow c) = 4/4$ ✓
$conf(c \Rightarrow b) = 4/5$ ✓
$conf(b, c \Rightarrow a) = 1/2$ ✗
$conf(a, c \Rightarrow b) = 1$ ✓
$conf(a, b \Rightarrow c) = 2/3$ ✓
$conf(a \Rightarrow b, c) = 2/3$ ✓
$conf(b \Rightarrow a, c) = 2/4$ ✗ (pruned with $b, c \Rightarrow a$)
$conf(c \Rightarrow a, b) = 2/5$ ✗ (pruned with $b, c \Rightarrow a$)

itemset	count
{ a }	3
{ b }	4
{ c }	5
{ a,b }	3
{ a,c }	2
{ b,c }	4
{ a,b,c }	2

Interestingness Measurements

Objective measures

Two popular measures:

- Support
- Confidence

Subjective measures [Silberschatz & Tuzhilin, KDD95]

A rule (pattern) is interesting if it is

- *unexpected* (surprising to the user) and/or
- *actionable* (the user can do something with it)

Criticism to Support and Confidence

Example 1 [Aggarwal & Yu, PODS98]

- Among 5000 students
  - 3000 play basketball (=60%)
  - 3750 eat cereal (=75%)
  - 2000 both play basket ball and eat cereal (=40%)
- Rule "play basketball $\Rightarrow$ eat cereal [40%, 66.7%]" is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%
- Rule "play basketball $\Rightarrow$ not eat cereal [20%, 33.3%]" is far more accurate, although with lower support and confidence
- Observation: "play basketball" and "eat cereal" are negatively correlated

Not all strong association rules are interesting and some can be misleading.

- Augment the support and confidence values with interestingness measures such as the correlation: "A $\Rightarrow$ B [*supp*, *conf*, *corr*]"

Other Interestingness Measures: Correlation

Correlation

Correlation (sometimes called *Lift*) is a simple measure between two items A and B:

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B \mid A)}{P(B)} = \frac{conf(A \Rightarrow B)}{supp(B)}$$

- The two rules $A \Rightarrow B$ and $B \Rightarrow A$ have the same correlation coefficient
- Takes both $P(A)$ and $P(B)$ in consideration
- $corr_{A,B} > 1$: The two items A and B are positively correlated
- $corr_{A,B} = 1$: There is no correlation between the two items A and B
- $corr_{A,B} < 1$: The two items A and B are negatively correlated

Other Interestingness Measures: Correlation

Example 2

item	transactions							
X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

rule	support	confidence	correlation
$X \Rightarrow Y$	25%	50%	2
$X \Rightarrow Z$	37.5%	75%	0.86
$Y \Rightarrow Z$	12.5%	50%	0.57

- X and Y: positively correlated
- X and Z: negatively related
- Support and confidence of $X \Rightarrow Z$ dominates
- But: items X and Z are negatively correlated
- Items X and Y are positively correlated

Hierarchical Association Rules: Motivation

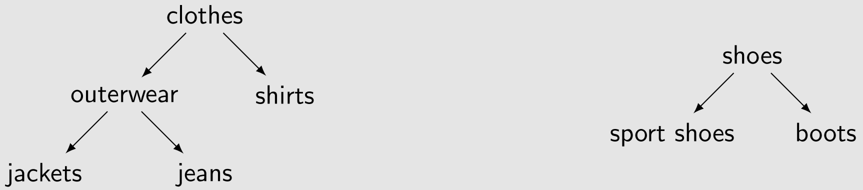
Problem

- High minSup: apriori finds only few rules
- Low minSup: apriori finds unmanagably many rules

Solution

Exploit item taxonomies (generalizations, is-a hierarchies) which exist in many applications

Example



Hierarchical Association Rules

New Task

Find all generalized association rules between generalized items, i.e. Body and Head of a rule may have items of any level of the hierarchy

Generalized Association Rule

$X \Rightarrow Y$ with $X, Y \subset I, X \cap Y = \emptyset$ and no item in Y is an ancestor of any item in X

Example

- Jeans $\Rightarrow$ Boots; supp $<$ minSup
- Jackets $\Rightarrow$ Boots; supp $<$ minSup
- Outerwear $\Rightarrow$ Boots; supp $>$ minSup

Hierarchical Association Rules: Characteristics

$$\begin{tikzcd}
 & & Y \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
X_1 & \cdots & X_i & \cdots & X_k
\end{tikzcd}$$

Characteristics

Let $Y = \biguplus_{i=1}^{k} X_i$ be a generalisation.

- For all $1 \leq i \leq k$ it holds $supp(Y \Rightarrow Z) \geq supp(X_i \Rightarrow Z)$
- In general, $supp(Y \Rightarrow Z) = \sum_{i=1}^{k} supp(X_i \Rightarrow Z)$ does not hold (a transaction might contain elements from multiple low-level concepts, e.g. boots *and* sport shoes).

Mining Multi-Level Associations

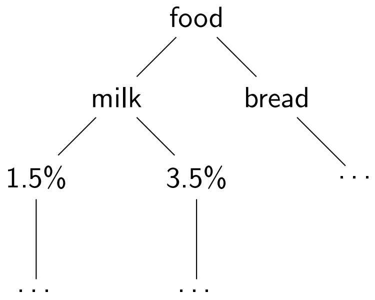
Top-Down, Progressive-Deepening Approach

1. First find high-level strong rules, e.g. milk $\Rightarrow$ bread [20%, 60%]
2. Then find their lower-level "weaker" rules, e.g. low-fat milk $\Rightarrow$ wheat bread [6%, 50%].

Support Threshold Variants

Different minSup threshold across multi-levels lead to different algorithms:

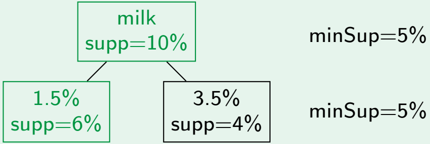
- adopting the same minSup across multi-levels
- adopting reduced minSup at lower levels



Minimum Support for Multiple Levels

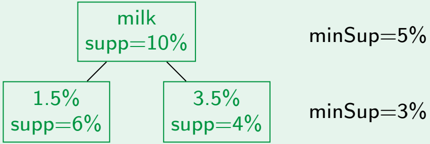
Uniform Support

- Search procedure is simplified (monotonicity)
- User only specifies one threshold



Reduced Support (Variable Support)

- Takes into account lower frequency of items in lower levels



Multilevel Association Mining using Reduced Support

Level-by-level independent method

Examine each node in the hierarchy, regardless of the frequency of its parent node.

Level-cross-filtering by single item

Examine a node only if its parent node at the preceding level is frequent.

Level-cross-filtering by k-itemset

Examine a k-itemset at a given level only if its parent k-itemset at the preceding level is frequent.

Multi-level Association: Redundancy Filtering

Some rules may be redundant due to "ancestor" relationships between items.

Example

- R_1: milk $\Rightarrow$ wheat bread [8%, 70%]
- R_2: 1.5% milk $\Rightarrow$ wheat bread [2%, 72%]

We say that rule 1 is an ancestor of rule 2.

Redundancy

A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

,

Interestingness of Hierarchical Association Rules: Notions

Let $X, X', Y, Y' \subseteq I$ be itemsets.

- X' is ancestor of X iff there exists ancestors $x'_1, \ldots, x'_k$ of $x_1, \ldots, x_k \in X$ and $x_{k+1}, \ldots, x_n$ with $n = |X|$ such that $X' = \{x'_1, \ldots, x'_k, x_{k+1}, \ldots, x_n\}$
- Let X' and Y' be ancestors of X and Y. Then we call the rules $X' \Rightarrow Y'$, $X \Rightarrow Y'$, and $X' \Rightarrow Y$ ancestors of the rule $X \Rightarrow Y$.
- The rule $X' \Rightarrow Y'$ is a direct ancestor of rule $X \Rightarrow Y$ in a set of rules if:
  1. Rule $X' \Rightarrow Y'$ is an ancestor of rule $X \Rightarrow Y$, and
  2. There is no rule $X'' \Rightarrow Y''$ being ancestor of $X \Rightarrow Y$ and $X' \Rightarrow Y'$ is an ancestor of $X'' \Rightarrow Y''$

R-Interestingness

R-Interestingness

A hierarchical association rule $X \Rightarrow Y$ is called R-interesting if:

- There are no direct ancestors of $X \Rightarrow Y$ or
- The actual support is larger than R times the expected support or
- The actual confidence is larger than R times the expected confidence

R-Interestingness: Expected Support

Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$ the expected support of $X \Rightarrow Y$ is defined as:

$$\mathbb{E}_{Z'}[P(Z)] = P(Z') \cdot \prod_{i=1}^{j} \frac{P(y_i)}{P(y_i)'}$$

where $Z = X \cup Y = \{z_1, \ldots, z_n\}$, $Z' = X' \cup Y' = \{z'_1, \ldots, z'_j, z_{j+1}, \ldots, z_n\}$ and each $z'_i \in Z'$ is an ancestor of $z_i \in Z$.

R. Srikant, R. Agrawal: Mining Generalized Association Rules. In VLDB, 1995.

R-Interestingness: Expected Confidence

Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$, then the expected confidence of $X \Rightarrow Y$ is defined as:

$$\mathbb{E}_{X' \Rightarrow Y'}[P(Y|X)] = P(Y' \mid X') \cdot \prod_{i=1}^{j} \frac{P(y_i)}{P(y_i)'}$$

where $Y = \{y_1, \ldots, y_n\}$ and $Y' = \{y_1', \ldots, y_j', y_{j+1}, \ldots, y_n\}$ and each $y_i' \in Y'$ is an ancestor of $y_i \in Y$.

R. Srikant, R. Agrawal: Mining Generalized Association Rules. In VLDB, 1995.

R-Interestingness: Example

Item	Support
clothes	20
outerwear	10
jackets	4

Let $R = 1.6$

No	Rule	Support	R-Interesting?
1	clothes $\Rightarrow$ shoes	10	yes: no ancestors
2	outerwear $\Rightarrow$ shoes	9	yes (wrt. rule 1): $supp(X \Rightarrow Y) = 9 > 1.6 \cdot \frac{10}{20} \cdot 10 = 8 = 1.6 \cdot \mathbb{E}(P(Z))$
3	jackets $\Rightarrow$ shoes	4	Not wrt. support: $\mathbb{E}(P(jackets \cup shoes)) = 3.2$ (wrt rule 1) $\mathbb{E}(P(jackets \cup shoes)) = 5.75$ (wrt rule 2)

Still need to check the confidence!

Summary Frequent Itemset & Association Rule Mining

- Frequent Itemsets
  - Mining: Apriori algorithm, hash trees, FP-tree
  - support, confidence
- Simple Association Rules
  - Mining: (Apriori)
  - Interestingness measures: support, confidence, correlation
- Hierarchical Association Rules
  - Mining: Top-Down Progressive Deepening
  - Multilevel support thresholds, redundancy, R-interestingness
- Further Topics (not covered)
  - Quantitative Association Rules (for numerical attributes)
  - Multi-dimensional association rule mining

Agenda

Motivation

Motivation

- In many applications the order matters, e.g. because the ordering encodes spatial or temporal aspects.
- In an ordered sequence, items are allowed to occur more than one time

Applications

Bioinformatics (DNA/protein sequences), Web mining, text mining, sensor data mining, process mining, ...

Sequential Pattern Mining: Basic Notions I

We now consider transactions having an order of the items. Define:

- *Alphabet* Σ is set symbols or characters (denoting items)
- *Sequence* $S = s_1 s_2 \ldots s_k$ is an ordered list of a length $|S| = k$ items where $s_i \in \Sigma$ is an item at position i also denoted as $S[i]$
- A *k-sequence* is a sequence of length k
- *Consecutive subsequence* $R = r_1 r_2 \ldots r_m$ of $S = s_1 s_2 \ldots s_n$ is also a sequence in Σ such that $r_1 r_2 \ldots r_m = s_j s_{j+1} \ldots s_{j+m-1}$ with $1 \leq j \leq n - m + 1$. We say S contains R and denote this by $R \subseteq S$
- In a more general *subsequence* R of S we allow for gaps between the items of R, i.e. the items of the subsequence $R \subseteq S$ must have the same order of the ones in S but there can be some other items between them
- A *prefix* of a sequence S is any consecutive subsequence of the form $S[1:i] = s_1 s_2 \ldots s_i$ with $0 \leq i \leq n$, $S[1:0]$ is the empty prefix

Sequential Pattern Mining: Basic Notions II

- A *suffix* of a sequence S is any consecutive subsequence of the form $S[i:n] = s_i s_{i+1} \ldots s_n$ with $1 \leq i \leq n+1$, $S[n+1:n]$ is the empty suffix.
- *(Relative) support* of a sequence R in D: $supp(R) = |\{S \in D \mid R \subseteq S\}|/|D|$
- S is frequent (or sequential) if $supp(S) \geq minSup$ for threshold $minSup$.
- A frequent sequence is *maximal* if it is not a subsequence of any other frequent sequence
- A frequent sequence is *closed* if it is not a subsequence of any other frequent sequence with the same support

Sequential Pattern Mining

Task

Find all frequent subsequences occuring in many transactions.

Difficulty

The number of possible patterns is even larger than for frequent itemset mining!

Example

There are $|\Sigma|^k$ different k-sequences, where $k > |\Sigma|$ is possible and often encountered, e.g. when dealing with DNA sequences where the alphabet only comprises four symbols.

Sequential Pattern Mining Algorithms

Breadth-First Search Based

- GSP (Generalized Sequential Pattern) algorithm[2]
- SPADE[3]
- ...

Depth-First Search Based

- PrefixSpan[4]
- SPAM[5]
- ...

---

[2] Sirkant & Aggarwal: *Mining sequential patterns: Generalizations and performance improvements*. EDBT 1996

[3] Zaki M J. *SPADE: An efficient algorithm for mining frequent sequences*. Machine learning, 2001, 42(1-2): 31-60.

[4] Pei at. al.: *Mining sequential patterns by pattern-growth: PrefixSpan approach*. TKDE 2004

[5] Ayres, Jay, et al: *Sequential pattern mining using a bitmap representation*. SIGKDD 2002.

GSP (Generalized Sequential Pattern) algorithm

- Breadth-first search: Generate frequent sequences ascending by length
- Given the set of frequent sequences at level k, generate *all* possible sequence extensions or candidates at level $k+1$
- Uses the Apriori principle (anti-monotonicity)
- Next compute the support of each candidate and prune the ones with $supp(c) < minSup$
- Stop the search when no more frequent extensions are possible

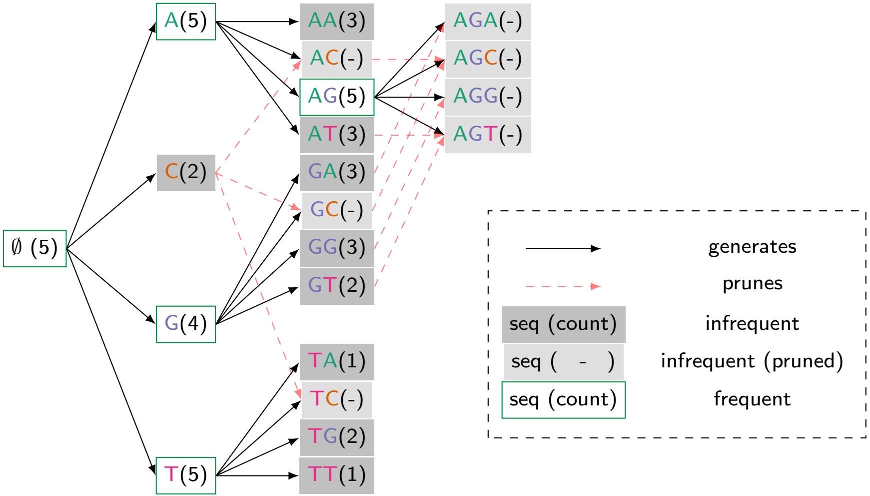
Projection-Based Sequence Mining: PrefixSpan: Representation

- The sequence search space can be organized in a prefix search tree
- The root (level 0) contains the empty sequence with each item $x \in \Sigma$ as one of its children
- A node labelled with sequence: $S = s_1 s_2 \ldots s_k$ at level k has children of the form $S' = s_1 s_2 \ldots s_k s_{k+1}$ at level $k+1$ (i.e. S is a prefix of S' or S' is an extension of S)

Prefix Search Tree: Example

ID	Sequence
S_1	CAGAAGT
S_2	TGACAG
S_3	GAG
S_4	AGTT
S_5	ATAG

$minSup = .8$



Projected Database

- For a database D and an item $s \in \Sigma$, the projected database w.r.t. s is denoted D_s and is found as follows: For each sequence $S_i \in D$ do
  - Find the first occurrence of s in S_i, say at position p
  - $suff_{S_i,s} \leftarrow suffix(S_i)$ starting at position $p+1$
  - Remove infrequent items from $suff_{S_i,s}$
  - $D_s = D_s \cup suff_{S_i,s}$

Example

$minSup = .8$ (i.e. 4 transactions)

ID	Sequence	D_A	D_G	D_T
S_1	CAGAAGT	GAAGT	AAGT	$\emptyset$
S_2	TGACAG	AG	AAG	GAAG
S_3	GAG	G	AG	-
S_4	AGTT	GTT	TT	T
S_5	ATAG	TAG	$\emptyset$	AG

Projection-Based Sequence Mining: PrefixSpan Algorithm

- The *PrefixSpan* algorithm computes the support for only the individual items in the projected databased D_s
- Then performs recursive projections on the frequent items in a depth-first manner

Initialization: $D_R \leftarrow D, R \leftarrow \emptyset, \mathcal{F} \leftarrow \emptyset$

procedure $\textsc{PrefixSpan}(D_R, R, minSup, \mathcal{F})$

  for all $s \in \Sigma$ such that $supp(s, D_R) \geq minSup$ **do**

    $R_s \leftarrow R + s$ ▷ append s to the end of R

    $\mathcal{F} \leftarrow \mathcal{F} \cup \{(R_s, sup(s, D_R))\}$ ▷ calculate support of s for each R_s within D_R

    $D_s \leftarrow \emptyset$

    for all $S_i \in D_R$ **do**

      $S_i' \leftarrow$ projection of S_i w.r.t. item s

      Remove all infrequent symbols from S_i'

      if $S' \neq \emptyset$ **then**

        $D_s \leftarrow D_s \cup S_i'$

    if $D_s \neq \emptyset$ **then**

      PrefixSpan$(D_s, R_s, minSup, \mathcal{F})$

PrefixSpan: Example

minSup = 0.8 (i.e. 4 transactions)

$D_\emptyset$

ID	Sequence
S_1	CAGAAGT
S_2	TGACAG
S_3	GAG
S_4	AGTT
S_5	ATAG
	A(5) ~~C(2)~~ G(5) T(4)

D_G

ID	Sequence
S_1	AAGT
S_2	AAG
S_3	AG
S_4	TT
S_5	$\emptyset$
	~~A(3)~~ ~~G(3)~~ ~~T(2)~~

D_T

ID	Sequence
S_1	$\emptyset$
S_2	GAAG
-	-
S_4	T
S_5	AG
	~~A(2)~~ ~~G(2)~~ ~~T(1)~~

D_A

ID	Sequence
S_1	GAAGT
S_2	AG
S_3	G
S_4	GTT
S_5	TAG
	~~A(3)~~ G(5) ~~T(3)~~

D_{AG}

ID	Sequence
S_1	G
S_2	$\emptyset$
S_3	$\emptyset$
S_4	$\emptyset$
S_5	$\emptyset$
	~~G(1)~~

Hence, the frequent sequences are: $\emptyset$, A, G, T, AG

Interval-based Sequential Pattern Mining

Interval-Based Representation

- Deals with the more common interval-based items s (or events).
- Each event has a starting t_s^+ and an ending time point t_s^-, where $t_s^+ < t_s^-$

Application

Health data analysis, Stock market data analysis, etc.

Relationships

Predefined relationships between items are more complex.

- Point-based relationships: before, after, same time.
- Interval-based relationships: Allen's relations[6], End point representation[7], etc.

---

[6] Allen: Maintaining knowledge about temporal intervals. In Communications of the ACM 1983

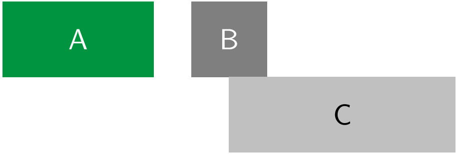
[7] Wu, Shin-Yi, and Yen-Liang Chen: Mining nonambiguous temporal patterns for interval-based events. TKDE 2007

Allen's Relations

Before	Overlaps	Contains	Starts	Finished-By	Meets	Equal
After	Overlapped-By	During	Started-By	Finishes	Met-by	Equal

Problem

- Allen's relationships only describe the relation between two intervals.
- Describing the relationship between k intervals unambiguously requires $\mathcal{O}(k^2)$ comparisons.



Interval-based Sequential Pattern Mining

- *TPrefixSpan*[8] converts interval-based sequences into point-based sequences:

$\{A^+\},\{A^-\},\{B^+\},\{B^-\}$

$\{A^+\},\{B^+\},\{A^-\},\{B^-\}$

$\{A^+\},\{A^-,B^+\},\{B^-\}$

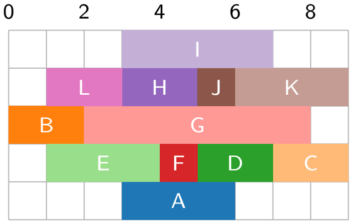
- Similar prefix projection mining approach as PrefixSpan algorithm.
- Validation checking is necessary in each expanding iteration to make sure that the appended time point can form an interval with a time point in the prefix.

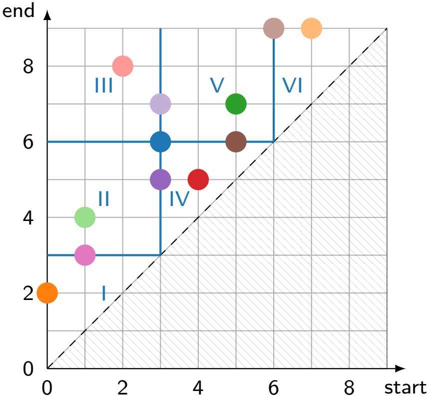
[8] Wu, Shin-Yi, and Yen-Liang Chen: Mining nonambiguous temporal patterns for interval-based events. TKDE 2007

An Open Issue: Considering Timing Information

Idea

Learn pattern from data by clustering, e.g. QTempIntMiner[9], Event Space Miner[10], PIVOTMiner[11]





[9] Guyet, T., & Quiniou, R.: *Mining temporal patterns with quantitative intervals*. ICDMW 2008

[10] Ruan, G., Zhang, H., & Plale, B.: *Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data*. IEEE Big Data 2014

[11] Hassani M., Lu Y. & Seidl T.: *A Geometric Approach for Mining Sequential Patterns in Interval-Based Data Streams*. FUZZ-IEEE 2016