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

Knowledge Discovery and Data Mining 1

(Data Mining Algorithms 1)

Winter Semester 2019/20

Counting Candidate Support

Motivation

Why is counting supports of candidates a problem?

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

Solution

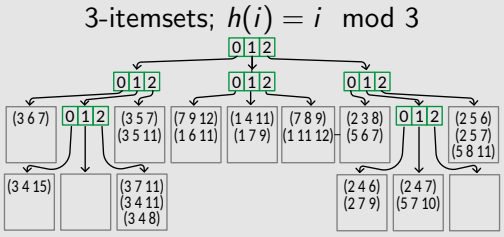
Store candidate itemsets in hash-tree

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

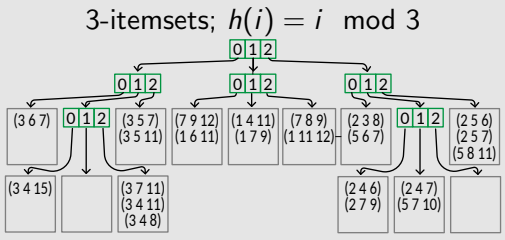


Hash-Tree: Construction

Search

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

Example

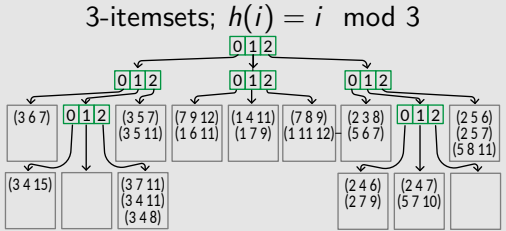


Hash-Tree: Construction

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



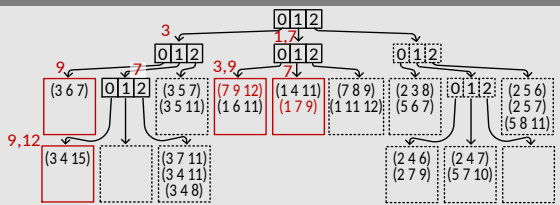
Hash-Tree: Counting

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=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

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

FP-Tree Construction

minSup=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

1

Header Table

Item	Frequency
b	8
c	7
d	6
e	3
f	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

FP-Tree Construction

minSup=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

Freq. Item (2.1)

c
cd
cef
cef
bcd
bcd
bcd
bde
bd
b
b
b

(1)

Header Table

Item	Frequency
b	8
c	7
d	6
e	3
f	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

FP-Tree Construction

minSup=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

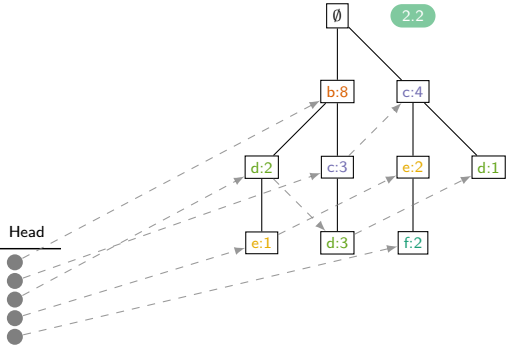
Freq. Item — 2.1

c
cd
cef
cef
bcd
bcd
bcd
bde
bd
b
b
b

Header Table — 1

Item	Frequency
b	8
c	7
d	6
e	3
f	2

Head



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

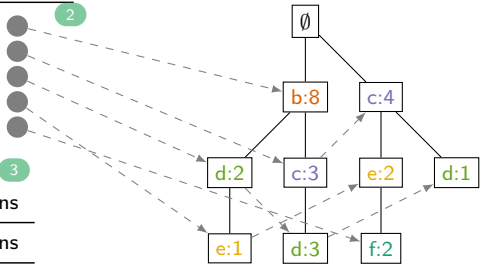
1. Construct conditional pattern base for each node in the FP-tree
2. Construct conditional FP-tree from each conditional pattern-base
3. Recursively mine conditional FP-trees and grow frequent patterns obtained so far;
   If the conditional FP-tree contains a single path, simply enumerate all the patterns

Major Steps to Mine FP-Tree: Conditional Pattern Base

Header Table 1

Item	Frequency	Head
b	8	
c	7	
d	6	
e	3	
f	2	

2



Conditional Pattern 3

Item	Cond. Patterns
b	∅
c	b:3, ∅
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2

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

Conditional Pattern

Item	Cond. Patterns
b	$\emptyset$
c	b:3, $\emptyset$
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2

Example: *e*-conditional FP-Tree

Item	Frequency
c	2
b	**1**
d	**1**

$\emptyset \mid e$

c:2

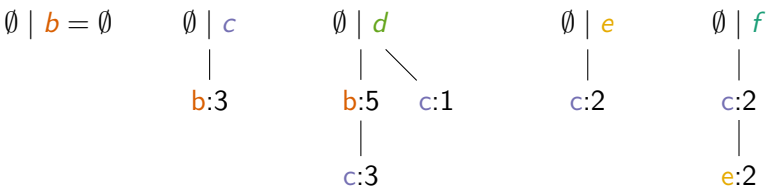
Construct conditional FP-tree from each conditional pattern-base

- 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
  - Re-sort items within sets by frequency
  - 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
b	$\emptyset$
c	b:3, $\emptyset$
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2



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

$\emptyset \mid f$

c:2

e:2

$\rightsquigarrow$

All frequent patterns concerning f:

f,

fc, fe

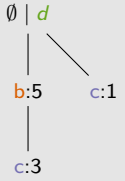
fce

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
b	∅
c	b:3, ∅

$\emptyset \mid db = \emptyset$

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

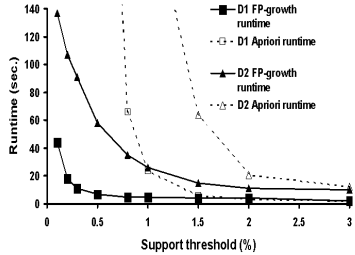


Image Source: [1]

[5] 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:

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

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."

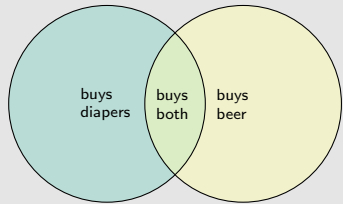
Interestingness of Association Rules

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*]"