Data Mining II / Adv. Topics in Data Science

Association Rules

Rita P. Ribeiro

2023/2024

Summary

Mining Association Rules

Too many rules ...

- The association rule algorithms tend to generate an excessive number of rules (for some problems, there can be thousands).
- Too many rules leads to model's interpretability lack.
- How can we reduce this number?
  - Changing the parameters: *minsup*, *minconf*
  - Restrictions on items: which items are relevant?
  - Summarization techniques: can we represent subsets of rules by a single representative rule?
  - Filter rules: improvement, measures of interest, ...

How to measure the improvement of a rule?

Improvement [Bayardo and Ag, 2000]

- **Improvement** of a rule is the minimum difference between its confidence and the confidence of any of its immediate simplifications.

  $improv(A \rightarrow C) = min(\{conf(A \rightarrow C) - conf(As \rightarrow C) \mid As \subseteq A\})$

- Example:
  - $R_1 : \{eggs, flower, bread\} \rightarrow \{sugar\} (conf = 0.505)$
  - $R_2 : \{eggs, flower\} \rightarrow \{sugar\} (conf = 0.5)$
  - $improv(R_1)$ is at most 0.005
  - with a *minmprov* of 0.01, R_1 is excluded.

Are all the rules interesting?

- Are all the discovered patterns interesting?
- In recent years, several measures have been proposed to extract interesting patterns.
- The idea is to select a subset of rules, that somehow are more relevant.
- Interesting rule (Silberschatz & Tuzhilin,95)
  - Unexpected, surprising to the user
    - Measure of interest: deviation from the expected or from the initial belief
  - Useful, actionable
    - Measure of interest: estimated benefit

How to measure the interest of a rule?

- Subjective measures: based on user's belief in the data (ex: unexpectedness, novelty, actionability, confirm hypothesis user wishes to validate)
  - These measures are hard to incorporate in the pattern discovery task.
- Objective measures: based on facts, statistics and structures of patterns (ex: support and confidence), independent of the domain considered.
  - For instance, patterns that involve mutually independent items or cover very few transactions are considered uninteresting.

How to measure the interest of a rule? (cont.)

Typically

- $A \rightarrow B$ is interesting if A and B are not statistically independent
- if A and B are statistically independent, the occurrence of A does not affect the probability of occurrence of B

$$sup(A \cup B) \approx sup(A) * sup(B)$$

$$conf(A \rightarrow B) \approx conf(\emptyset \rightarrow B)$$

- $A \rightarrow B$ may have high support and confidence and still not be interesting.
  - $\{butter\} \rightarrow \{bread\}(sup = 5\%, conf = 95\%)$
  - it is not unexpected
  - it is not useful

How to measure the interest of a rule? (cont.)

- A measure of interest should evaluate the deviation from independence.
- A rule is unexpected as it deviates from independence.
- There are different approaches to measure this deviation:
  - *lift*
  - *conviction*
  - χ^2
  - *correlation*
  - ...

Measures of Interest: limitations of support and confidence

- Assume we are interested in studying the relationship between people who drink tea and coffee.
- We summarize the preferences of 1000 people

	Coffee	$\neg$*Coffee*	
Tea	150	50	200
$\neg$*Tea*	650	150	800
	800	200	1000

- How interesting is the rule *Tea* $\rightarrow$ *Coffee*?
- *sup* $= 150/1000 = 15\%$ and *conf* $= 150/200 = 75\%$
- The confidence of the rule is high, however the likelihood of a person drinking coffee regardless of drinking tea is 80%.
- Knowing that a person drinks tea actually decreases the probability of drinking coffee (from 80% to 75%).
- Thus, the rule is indeed deceitful.
- High confidence rules can be misleading.

Measures of Interest: LIFT

- **lift** is the ratio between confidence of the rule and the support of the itemset appearing in the consequent:

$$lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{sup(B)} = \frac{sup(A \cup B)}{sup(A)sup(B)}$$

- Measures the influence of A in the presence of B.
- $lift = 1$: A and B are independent ($sup(A \cup B) = sup(A)sup(B)$).
- $lift < 1$: A and B are negatively correlated.
- $lift > 1$: A and B are positively correlated.

- $lift(Tea \rightarrow Coffee) = 0.15/(0.2 * 0.8) = 0.9375$
- negative correlation between tea and coffee drinkers.

Measures of Interest: LIFT (cont.)

- The **lift** is a measure of the deviation from a rule $A \rightarrow B$ regarding the statistical independence between the antecedent A and consequent B.

- Takes values between 0 and infinity:
  - a value close to 1 indicates that the occurrence of A has no effect on the occurrence of B.
  - a value smaller than 1 indicates that the occurrence of A has a negative effect on the occurrence of B, i.e. the occurrence of A is likely to lead to the absence of B.
  - a value greater than 1 indicates that the occurrence of A has a positive effect on the occurrence of B, i.e. the occurrence of A increases the likelihood of occurrence of B.

Measures of Interest: Conviction

- **lift** measures co-occurrence only (not implication) and is symmetric with respect to antecedent and consequent, i.e. $lift(A \rightarrow B) = lift(B \rightarrow A)$
- **conviction** is a measure proposed to tackle some of the weaknesses of *confidence* and **lift**.
- Unlike lift, **conviction** is sensitive to rule direction. It indicates the departure from independence of A and B taking into account the implication direction.
- Is inspired in the logical definition of implication and attempts to measure the degree of implication of a rule.

Measures of Interest: Conviction (cont.)

- **conviction** of a rule $A \rightarrow B$ is the ratio between
  - the expected frequency that A occurs without B, if A and B were independent
  - the observed frequency that the rule makes of incorrect predictions.
- Is the inverse **lift** of the rule $R' = A \rightarrow \neg B$.

$$conviction(A \rightarrow B) = \frac{1 - sup(B)}{1 - conf(A \rightarrow B)} = \frac{sup(A)sup(\neg B)}{sup(A \cup \neg B)}$$

Measures of Interest: Conviction (cont.)

- $conviction(A \rightarrow B) = 1$ indicates independence between A and B.
- A high value of **conviction** means that the consequent depends strongly on the antecedent.
- **conviction** increases a lot when *confidence* gets closer to 1.
- Example:
  - $sup(female) = 0.5$, $sup(mother) = 0.2$
  - $conf(mother \rightarrow female) = 1$
  - $lift(mother \rightarrow female) = 0.2/(0.2 * 0.5) = 2$
  - $conviction(mother \rightarrow female) = (1 - 0.5)/(1 - 1) = \infty$

Improving Apriori

- Challenges of Frequent Pattern Mining
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce number of transaction database scans
  - Shrink number of candidates (*bottleneck* of Apriori)
  - Facilitate support counting of candidates
- Some methods that improve Apriori's efficiency
  - Partitioning [Savasere et al., 1995]
  - Sampling [Toivonen, 1996]
  - Dynamic Itemset Counting [Brin et al., 1997]
  - **Frequent Pattern Projection and Growth (FP-Growth)** [Han et al., 2004]

FP-Growth

FP-Growth [Han et al., 2004] takes a different approach to discover frequent itemsets.

It proceeds in two phases:

- encodes the transaction data base into a compact structure called FP-Tree;
  - complete for frequent pattern mining and avoids costly database scans
- extracts frequent itemsets directly from this structure.
  - using a divide-and-conquer strategy and avoiding candidate generation

FP-Growth: FP-Tree representation

- FP-Tree is a compressed representation of the input data.
- It is constructed by reading one transaction at a time and mapping it onto to a path in the FP-Tree.
- As different transactions can have several items in common, paths may overlap.
- The more paths overlap, the more compression can be achieved with FP-Tree.

Ultimately, If the size of FP-Tree is small enough to fit into main memory, it will be possible to extract frequent itemsets directly from memory, without making repeated passes over the data stored in disk.

FP-Growth: FP-Tree representation (cont.)

The FP-Tree consists of:

- a root node
- a set of item prefix subtrees
- each node has
  - item name
  - counter for the transactions mapped into the given path (prefix)
  - node-link (pointer to next node with same item name)
- frequent item header table with
  - item name
  - head of node-link (pointer to first node with that name)

FP-Growth: Example 1 - FP-Tree

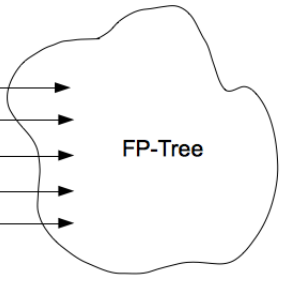
- The data is scanned once to determine the support of each item.
- Infrequent items are discarded.
- Frequent items are sorted in decreasing order of support.

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

Header Table

Item	Support (minsup=2)
a	8
b	7
c	6
d	5
e	3



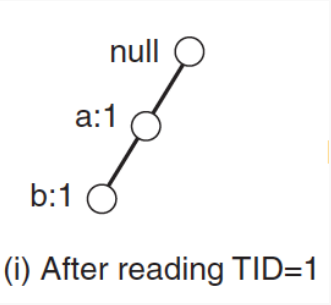
- A second pass over the data is done to construct the FP-Tree.

FP-Growth: Example 1 - FP-Tree (cont.)

- Reads TID 1: $\{a, b\}$
  - forms the path $null \rightarrow a \rightarrow b$ to encode the transaction.
  - both nodes have frequency count of 1

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



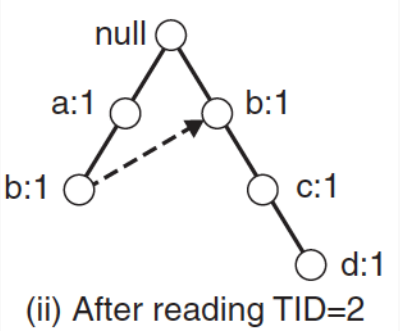
(i) After reading TID=1

FP-Growth: Example 1 - FP-Tree (cont.)

- Reads TID 2: $\{b, c, d\}$
  - forms the path $null \rightarrow b \rightarrow c \rightarrow d$ to encode the transaction.
  - all nodes have frequency count of 1
  - although the first two transactions have an item in common, their paths are disjoint because they don't share a common prefix.

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



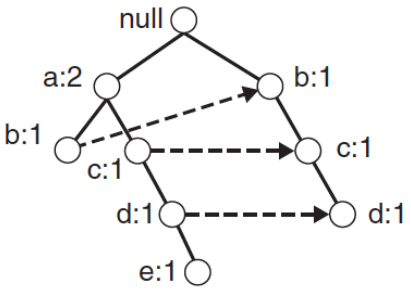
(ii) After reading TID=2

FP-Growth: Example 1 - FP-Tree (cont.)

- Reads TID 3: $\{a, c, d, e\}$
  - forms the path $null \rightarrow a \rightarrow c \rightarrow d \rightarrow e$ to encode the transaction.
  - shares a common prefix with TID 1, thus this path overlaps the path for TID 1
  - frequency count for node a is incremented to 2.
  - frequent counts for nodes c, d and e are equal to 1.

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



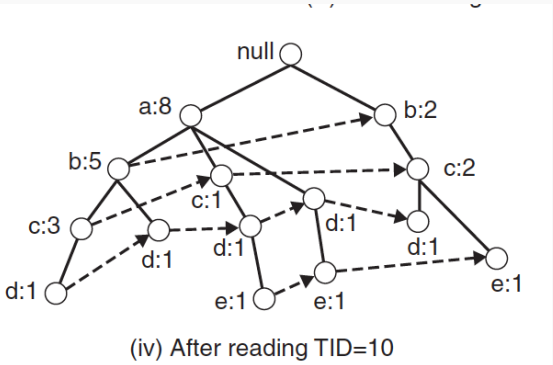
(iii) After reading TID=3

FP-Growth: Example 1 - FP-Tree (cont.)

- The process continues until every transaction is mapped onto one of the paths in the FP-Tree.

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



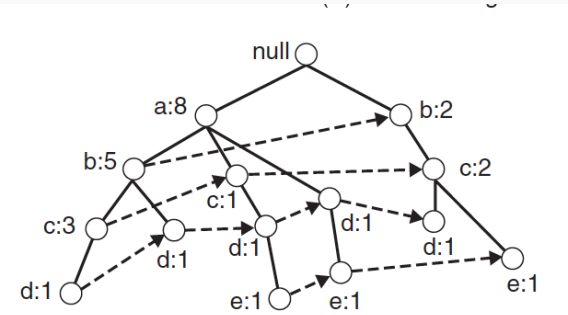
(iv) After reading TID=10

FP-Growth: Example 1 - Frequent Itemset Generation

- To have the frequent patterns ending with y we only need to consider the prefix paths of nodes with y.
- An itemset $X \cup \{y\}$ is frequent iff X is frequent in the conditional pattern base of y.
- These prefix paths can be easily accessed using the pointers associated with node of that item.

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- FP-Growth explores the built FP-Tree in a bottom-up way to generate frequent itemsets.

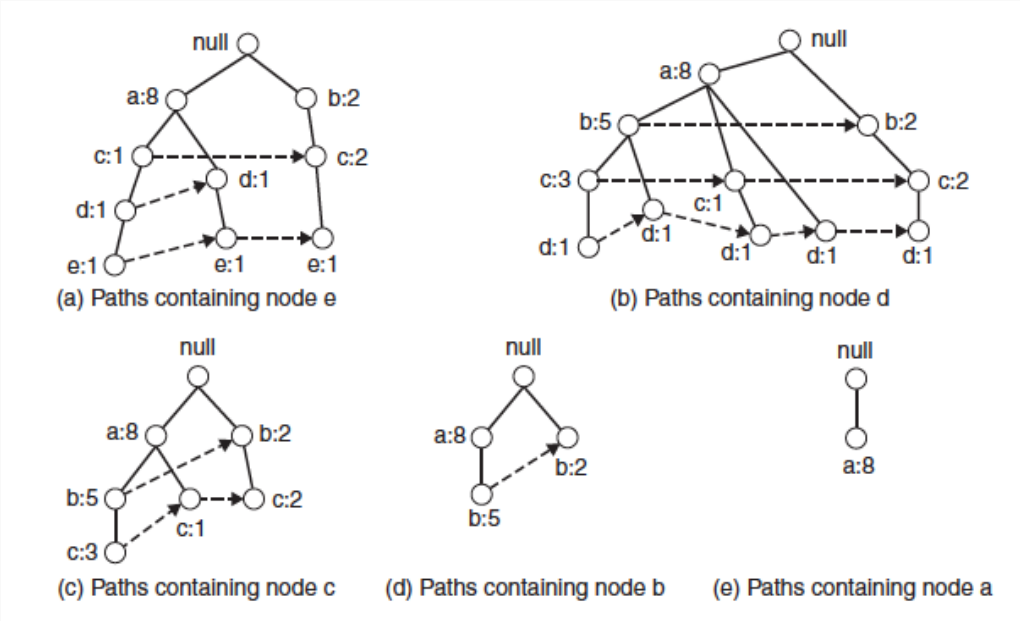


(iv) After reading TID=10

- Looks for frequent itemsets ending with e first, followed by d, c, b and a.

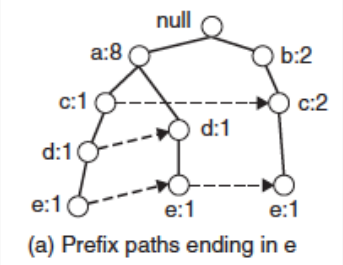
FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- Divides the frequent itemset generation problem into multiple subproblems.



FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- Find all frequent itemsets ending in *e*.

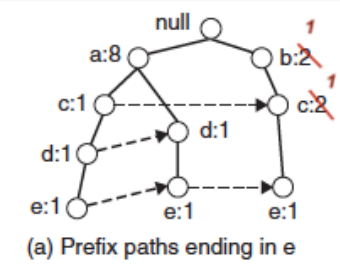


- The support count for *e* is the sum of support counts associated with node *e*.
- $\{e\}$ is considered a frequent itemset because its support is 3.
- As $\{e\}$ is frequent, now it has to find frequent itemsets ending in *de*, *ce*, *be* and *ae*.

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

It builds a **conditional FP-tree** to find frequent itemsets ending with a particular suffix (e), as follows:

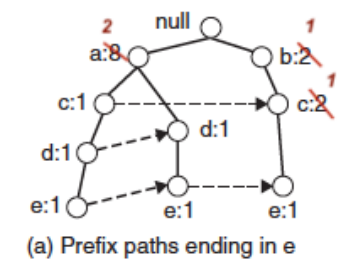
- **conditional FP-Tree - step 1**
  - updates the support counts of the prefix paths (some of them include transactions that do not contain item e)
  - $null \rightarrow b:2 \rightarrow c:2 \rightarrow e:1$ includes the transaction $\{b, c\}$ that does not contain e.
  - is updated to $null \rightarrow b:1 \rightarrow c:1 \rightarrow e:1$ to reflect the actual number of transactions that contain $\{b, c, e\}$



(a) Prefix paths ending in e

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

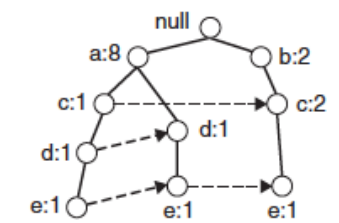
- **conditional FP-Tree - step 2**
  - removes nodes for that suffix (e).
  - the support counts along the prefix paths have been updated
  - finding frequent itemsets ending in de, ce, be and ae no longer need information about e.



(a) Prefix paths ending in e

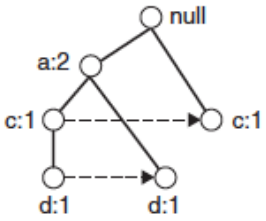
FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- **conditional FP-Tree - step 3**
  - removes items that are no longer frequent
  - node *b* appears only once and has a count of 1, i.e. there is only one transaction that contains both *b* and *e*
  - all itemsets ending with *be* must be infrequent
  - *b* can be ignored



(a) Prefix paths ending in e

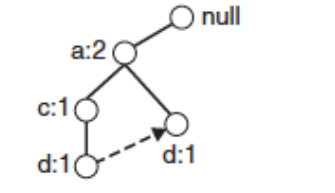
→



(b) Conditional FP-tree for e

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- FP-Growth uses the conditional FP-Tree for *e* to solve the subproblems of finding frequent itemsets ending in *de*, *ce* and *ae*
- frequent itemsets ending in *de*

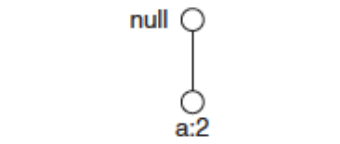


(c) Prefix paths ending in de

- adding frequency counts of node *d*, we obtain that the support count for $\{d, e\}$ is 2, so it is frequent.

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- builds the conditional FP-tree for *de*
- updates counts and removes infrequent item *c*

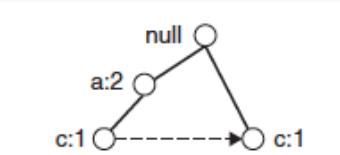


(d) Conditional FP-tree for de

- as it contains only one item (a) and its support is 2, it extracts the frequent itemset $\{a, d, e\}$
- moves on to the next problem

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- finding frequent itemsets ending with *ce*
- process the prefix paths for *c*

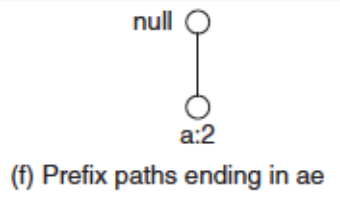


(e) Prefix paths ending in ce

- only $\{c, e\}$ is found to be frequent
- moves on to the next problem

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- finding frequent itemsets ending with *ae*
- process the prefix paths for *a*

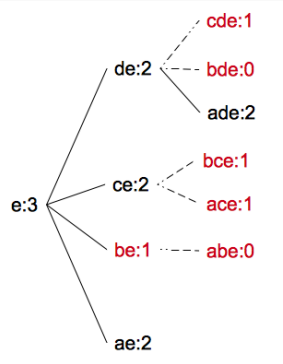


(f) Prefix paths ending in ae

- only $\{a, e\}$ is found to be frequent

FP-Growth: Example 1 - Frequent Itemset Generation (cont.)

- frequent itemsets with suffix *e*



- at the end, the following frequent items are obtained

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

FP-Growth: Example 2 - FP-Tree

Consider the following set of transactions with $minsup = 3$.

For each transaction we obtain the ordered set of frequent items.

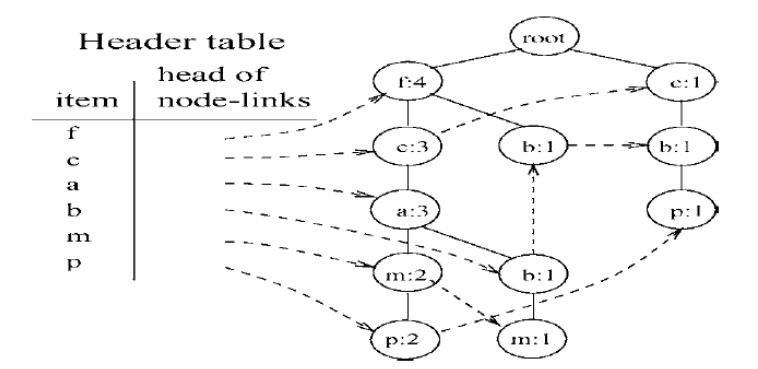
TID	Items bought	(Ordered) frequent items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

For each frequent item, we have the following support

$f : 4, c : 4, a : 3, b : 3, m : 3, p : 3$

FP-Growth: Example 2 - FP-Tree (cont.)

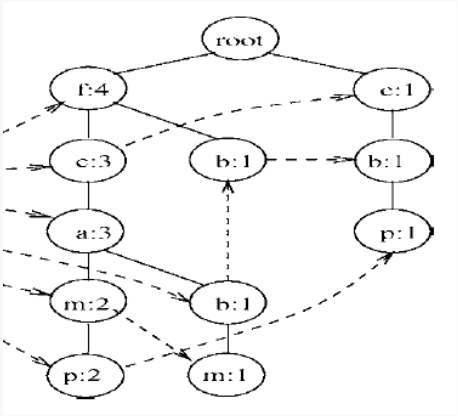
The obtained FP-Tree is



The list of ordered frequent items is $f - c - a - b - m - p$

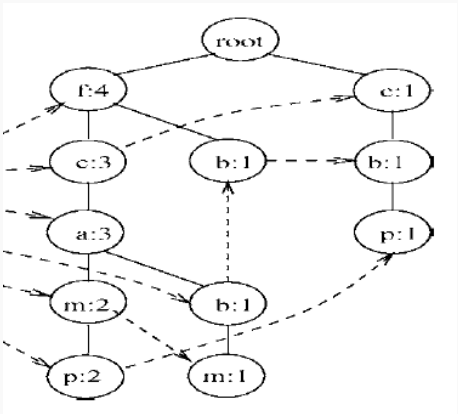
FP-Growth: Example 2 - Frequent Itemset Generation

- patterns containing p
- p's conditional patterns
  - $fcam : 2$, $cb : 1$
- conditional FP-Tree
  - only c is frequent
  - $null \rightarrow c : 3$
- size 2 patterns
  - $cp : 3$
- patterns with p
  - $p : 3$, $cp : 3$



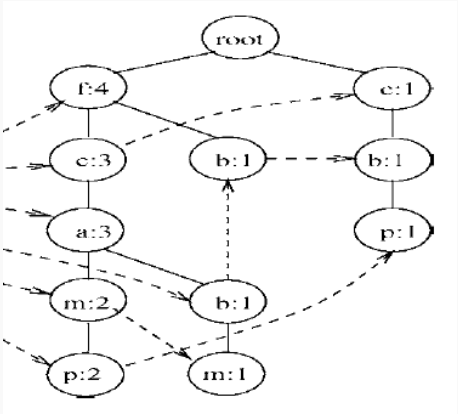
FP-Growth: Example 2 - Frequent Itemset Generation (cont.)

- patterns containing m (but no p)
- m's conditional patterns
  - $fca : 2$, $fcab : 1$
- conditional FP-Tree
  - b is not frequent
  - $null \rightarrow f : 3 \rightarrow c : 3 \rightarrow a : 3$
- size 2 patterns
  - $am : 3$, $cm : 3$, $fm : 3$
- extend am
  - ...



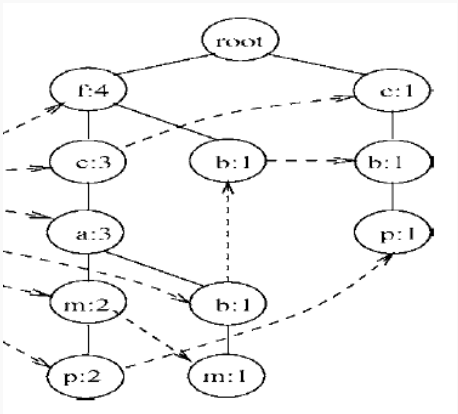
FP-Growth: Example 2 - Frequent Itemset Generation (cont.)

- patterns containing *am*
- FP-Tree is *null* $\rightarrow$ *f* : 3 $\rightarrow$ *c* : 3
- *am*'s conditional patterns
  - *fc* : 3
- conditional FP-Tree
  - *null* $\rightarrow$ *f* : 3 $\rightarrow$ *c* : 3
- size 3 patterns
  - *cam* : 3, *fam* : 3
- extend *cam*
  - *fam* is not extendable



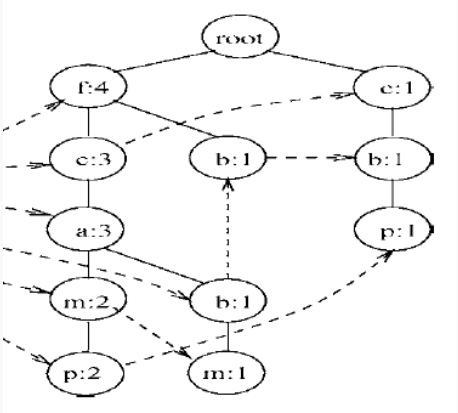
FP-Growth: Example 2 - Frequent Itemset Generation (cont.)

- patterns containing *cam*
- FP-Tree is *null* $\rightarrow$ *f* : 3
- *cam*'s conditional patterns
  - *f* : 3
- conditional FP-Tree
  - *null* $\rightarrow$ *f* : 3
- size 4 patterns
  - *fcam* : 3
- no longer patterns available



FP-Growth: Example 2 - Frequent Itemset Generation (cont.)

- and continues to extend patterns with
  - *cm*
  - *fm*
- and then find patterns with
  - *b*
  - *a*
  - *c*
  - *f*

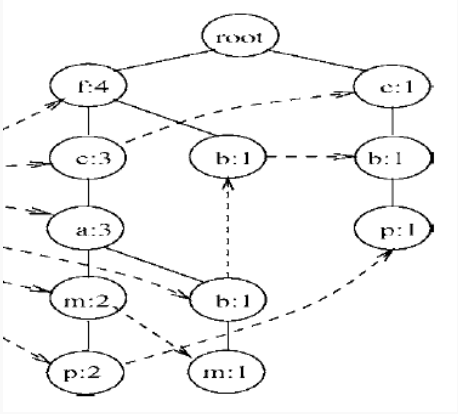


FP-Growth: Example 2 - Frequent Itemset Generation (cont.)

Item	Conditional pattern-base	Conditional FP-Tree
p	*fcam* : 2, *cb* : 1	*null* $\rightarrow$ *c* : 3
m	*fca* : 2, *fcab* : 1	*null* $\rightarrow$ *f* : 3 $\rightarrow$ *c* : 3 $\rightarrow$ *a* : 3
b	*fca* : 1, *f* : 1, *c* : 1	$\emptyset$
a	*fc* : 3	*null* $\rightarrow$ *f* : 3 $\rightarrow$ *c* : 3
c	*f* : 3	*null* $\rightarrow$ *f* : 3
f	$\emptyset$	$\emptyset$

Frequent Itemsets (*minsup* $= 3$)

$$\{p:3, cp:3, m:3, am:3, cm:3, fm:3, cam:3, fam:3, fcam:3, fcm:3, b:3, a:3, fa:3, ca:3, fca:3, c:4, fc:3, f:4\}$$



FP-Growth: Example 2 - Frequent Itemset Generation (cont.)

Single Path FP-Tree special case:

- A FP-Tree with a single path can be mined by enumerating all the combinations of the subpaths.

  Example:
  conditional FP-Tree for m:
  $$null \rightarrow\rightarrow f:3 \rightarrow c:3 \rightarrow a:3$$
  leads to the frequent patterns
  $\{m:3, fm:3, cm:3, am:3, fcm:3, fam:3, cam:3, fcam:3\}$

- the count of each subpath is the minimum of the nodes in it.

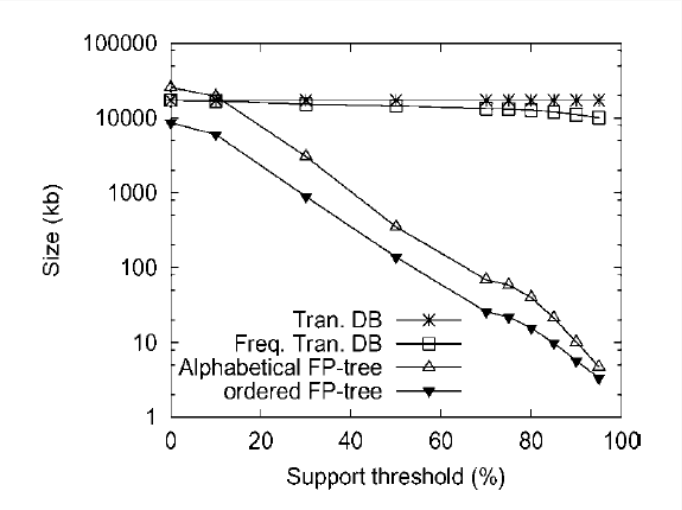
Analysis

It can be shown that FP-Growth:

- finds the complete set of frequent patterns in the given transaction DB;
- the FP-Tree is usually much smaller than DB.
- it scans the FP-Tree of DB once foreach frequent item A
  - generates a small pattern involving A
  - pattern mining is done recursively on that small pattern
- mining operation consists of:
  - prefix counts adjustment
  - counting local frequent items
  - pattern fragment concatenation

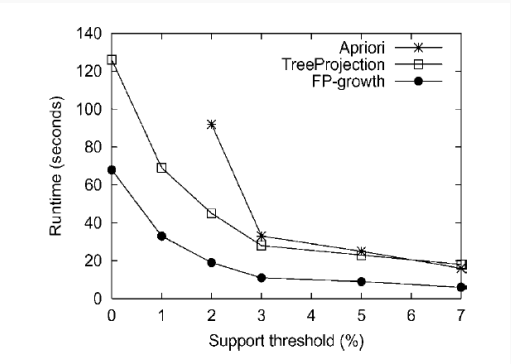
Analysis (cont.)

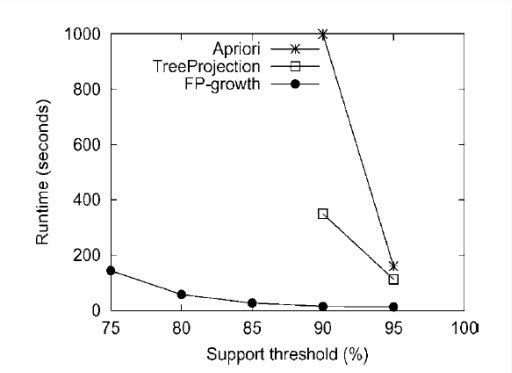
- Assess compactness of FP-Tree
  - by measuring their size on Connect-4 dataset



Analysis (cont.)

- Scalability study
  - by measuring time of competing algorithms on a sparse artificial dataset and (dense) Connect-4 dataset





Exercises

1. Consider the following set of transactions:

$$\{\{a, b, c, d\}, \{a, b, c, d\}, \{a, e\}, \{a, b\}\}$$

Assuming that $minsup = 50\%$, how much space is needed for

- Apriori (number of candidate sets) ?
- FP-growth (size of tree and size of header table) ?

Exercises (cont.)

2. Consider the following set of transactions:

$$\{\{a, d, e, f\}, \{b, d, e, f\}, \{c, d, e, f\},$$
$$\{a\}, \{a\}, \{a\}, \{b\}, \{b\}, \{b\}, \{c\}, \{c\}, \{c\}\}$$

Assuming a minimum support of 3 transactions, build the corresponding FP-Tree.

Exercises (cont.)

3. Consider the following set of transactions:

TID	Itemset
1	A B E
2	B D
3	B C
4	A B D
5	A C
6	B C
7	A C
8	A B C E
9	A B C

Using the FP-growth algorithm with $minsup = 20\%$

- find the frequent itemsets

Association Rules: Conclusions

- GOAL: Finding associations
- Association rule mining:
  - Frequent itemsets (requires min support)
  - Association rules (requires min confidence)
    - Probabilistic implications
- One of the most used data mining tools
  - Problem: generates too much rules
  - Pattern compression and pattern selection
- Several algorithms:
  - Apriori is the most known algorithm
  - There are variants of Apriori that return exactly the same patterns!
  - Completeness: find all rules.

References

References


Aggarwal, C. C. (2015).
***Data Mining, The Texbook*.**
Springer.


Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. (1996).
Fast discovery of association rules.
In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. American Association for Artificial Intelligence.


Agrawal, R. and Srikant, R. (1994).
Fast algorithms for mining association rules in large databases.
In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499. Morgan Kaufmann Publishers Inc.


Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997).
Dynamic itemset counting and implication rules for market basket data.
In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, volume 26, pages 255–264. ACM.


Domingo, C., Gavalda, R., and Watanabe, O. (1998).
On-line sampling methods for discovering association rules.

References (cont.)

Gama, J. (2016).
Association rules.
Slides.

Gama, J., Oliveira, M., Lorena, A. C., Faceli, K., and de Leon Carvalho, A. P. (2015).
***Extração de Conhecimento de Dados - Data Mining*.**
Edições Sílabo, 2nd edition.

Han, J., Kamber, M., and Pei, J. (2011).
***Data Mining: Concepts and Techniques*.**
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

Han, J., Pei, J., Yin, Y., and Mao, R. (2004).
Mining frequent patterns without candidate generation: A frequent-pattern tree approach.
Data Mining and Knowledge Discovery, 8(1):53–87.

Jorge, A. (2016).
Association rules.
Slides.

Liu, B. (2011).
***Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data*.**
Springer, 2nd edition.

References (cont.)

Savasere, A., Omiecinski, E., and Navathe, S. B. (1995).
An efficient algorithm for mining association rules in large databases.
In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, pages 432–444. Morgan Kaufmann Publishers Inc.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005).
***Introduction to Data Mining*.**
Addison Wesley.

Toivonen, H. (1996).
Sampling large databases for association rules.
In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 134–145. Morgan Kaufmann Publishers Inc.

Torgo, L. (2017).
***Data Mining with R: Learning with Case Studies*.**
Chapman and Hall/CRC, 2nd edition.