UNIVERSITY OF WARSAW
Faculty of Economic Sciences

Unsupervised Learning

Winter Semester, 2025/2026

Unsupervised learning: association rules

Literature

Association rules: literature

- Nisbet, R.; Elder, J.; Miner, G. (2009). Handbook of Statistical Analysis and Data Mining Applications. Academic Press.
- Gentle, J.E.; Härdle, W.A.; Mori, Y. (2012). Handbook of Computational Statistics: Concepts and Methods. Springer.
- Maimon, O; Rokach, L. (2005). Data Mining and Knowledge Discovery Handbook. Springer.

Association rules: introduction

Introduction to association rules

- Imagine that you deal with a database, where each row is made of some binary attributes
- Frequent patters of occurrence may be discovered
- Used for unsupervised knowledge discovery in large databases
- We may perform a basket analysis of the set of product in a transaction
  - Range of applications of association rules mining is broad

Introduction to association rules

- Association rules represent patterns in data without a specified target variable
- There is no need for the algorithm to be trained
- Data does not have to be labeled ahead of time
- Quite tough to evaluate the performance of a rule learner
- Types of undirect/unsupervised data mining
  - Actionable rules – high-quality and actionable information
  - Trivial rules – well-known information by those familiar with the business
  - Inexplicable rules – do not suggest any action

Introduction to association rules

	C1	C2	C3	C4	…
R1	1	0	0	0	…
R2	0	1	1	0	…
R3	1	0	1	0	…
R4	1	0	1	1	…

Introduction to association rules

Basket	Items
1	Bread, Butter, Eggs, Chips
2	Flour, Eggs, Meat, Wine
3	Vodka, Cucumber, Chocolate, Fish
4	Beer, Chips, Eggs, Bacon

Introduction to association rules

- Databases are common for transactional data & market basket analysis
- Association rule may represent an occurence of events in the database
- Population: set of all transactions
- One record: one transaction

- {Bread} -> {Cheese}
- Bread is the rule antecedent
- Cheese is the rule consequent

Introduction to association rules

- We may deal with more complex antecedents or consequents
- {peanut butter, jelly} -> {bread}
- If peanut butter and jelly are purchased, then bread is also likely to be purchased
- Or peanut butter and jelly imply bread

Introduction to association rules

- From simple combinatorics
  - Given k items that appear in a set, there are 2^k possible itemsets that must be searched for rules
- Impossible task to be done manually in most practical cases
- Many of the potential combinations of items are rarely found in practice
- Even if a store sells both automotive items and some cosmetics, products like this in one basket are rather uncommon

Introduction to association rules

- market basket analysis
- interesting patterns of DNA and protein sequences in an analysis of cancer data
- patterns of purchases in combination with fraudulent credit card use
- patterns of behavior that proceed customers dropping their cell phone service
- patterns in economics simulations

Measuring rule interest

Measuring rule interest

- Support:
  - How frequently an itemset or a rule occurs in the data
  - E.g. the support of {bread} would be 0.4 if bread appears in 40% of purchases
  - X indicates the number of transactions the itemset X appers in; N is the number of transactions in the database

$$\text{support}(X) = \frac{\text{count}(X)}{N}$$

- Confidence:
  - Confidence is the percentage in which the consequent is also satisfied upon particular antecedent
  - The proportion of transactions where the presence of item or itemset X results in the presence of item or itemset Y

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X, Y)}{\text{support}(X)}$$

Measuring rule interest

- Lift
  - controls for the support (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}
  - the rise in probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about presence of {X}
  - a value of lift greater than 1 vouches for positive association between {Y} and {X}
  - lift around 1 implies independent itemsets
  - a value of lift smaller than 1 implies negative association between itemsets

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

$$\text{lift}(X \rightarrow Y) = \frac{\text{confidence}(X \rightarrow Y)}{\text{support}(Y)}$$

Measuring rule interest

- Strong rules have both high support and confidence – rules that satisfy both a minimum support threshold and a minimum confidence threshold
  - In association rule mining we look for all frequent itemsets
  - Then we generate strong association rules from the frequent itemsets
- Support of an association pattern: percentage of task-relevant data transactions for which the pattern is true
- Confidence: measure of certainty or trustworthiness associated with each discovered pattern
- min_sup means the minimum support threshold
- Itemset satisfies minimum support when the occurrence frequency of the itemset is greater or equat to min_sup
- If the itemset satisfies minimum support, it is a frequent itemset

Measuring rule interest

- The logic
  - An itemset is a set of items
  - A $k{-}itemset$ is an itemset with k items
  - Given a dataset D, an itemset X has a (frequency) count in D
  - An association rule is about relationships between two disjoint itemsets X and Y

$$X \Longrightarrow Y$$

  - It presents the pattern when X occurs, Y also occurs
  - Association rules do not represent any sort of causality or correlation between the two itemsets

Measuring rule interest

- The logic
  - Define R as the set of attributes of the items in the database
  - Define X as a subset of attributes from R
  - Assume that X is a pattern from the database, if there is any row where all the attributes of X are 1
  - Define the support (frequency) of a pattern X

$$fr(X)=|M(X,r)|/|r|$$

  - $|M(X,r)|$ is the number of times that X appears in the database and the size of the database is represented by $|r|$

Measuring rule interest

- The logic
  - Assume the minimum support min_sup∈[0,1]
  - State that the pattern X is frequent if

$$fr(x) \geq \text{min\_}sup$$

  - $\mathcal{F}$ (r,min_sup) is the set of patterns frequent in
  - $\mathcal{F}(r,\text{min\_}sup)=\{X \subseteq R \; / fr(X) \geq \text{min\_}sup\}$
  - We have R attributes and X and Y subsets of attributes with $X \cap Y = \emptyset$
  - Association rule

$$X \Longrightarrow Y$$

  - $conf(X \Longrightarrow Y)$ stands for the confidence of the association rule
  - $conf(X \Longrightarrow Y)=(fr(X \cup Y))/(fr(X))$
  - We shall consider the minimum value of confidence min_$conf$∈[0,1]

Apriori

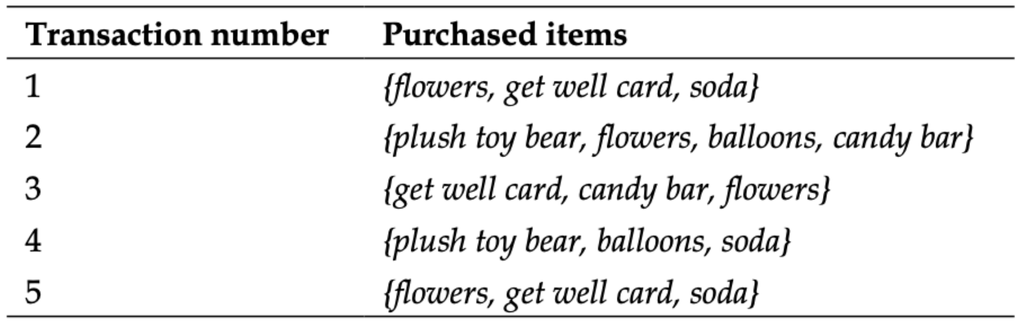
Apriori algorithm

- The algorithm uses prior knowledge of frequent items
- The apriori property is used to reduce the search space
- Apriori property: all nonempty subsets of frequent items must be also frequent
- If a set does not pass a test, its subsets will fail the same test as well
- Any subset of a frequent itemset must be frequent

Apriori algorithm

- By ignoring some rare combinations, it is possible to limit the scope of the search for rules to a more manageable size
- It uses statistical measures of an itemset’s interestingness to locate association rules in much larger databases
- The name of the algorithm
  - It utilizes a simple prior (a priori) belief about the properties of frequent itemsets
  - It allows to limit the number of rules to search
- The heuristic (Apriori property)
  - All subsets of a frequent itemset must be also frequent
  - If {A, B} is frequent, then {A} and {B} both must be frequent
  - if we know that {A} does not meet a desired support threshold, there is no reason to consider {A, B} or any itemset containing {A}
    - it cannot possibly be frequent

Apriori algorithm (Lantz, 2013)



Transaction number	Purchased items
1	*{flowers, get well card, soda}*
2	*{plush toy bear, flowers, balloons, candy bar}*
3	*{get well card, candy bar, flowers}*
4	*{plush toy bear, balloons, soda}*
5	*{flowers, get well card, soda}*

- Some patterns are notable as they appear frequently enough to catch our interest
- We may apply a bit of logic and subject-matter experience to explain the rule

Apriori algorithm

- Identify all itemsets that meet a minimum support threshold
  - Multiple iterations
  - Each of those involves evaluating the support of storing a set of increasingly large itemsets
  - All itemsets from iteration i are combined to generate candidate itemsets for evaluation in iteration i+1
  - The Apriori algorithm can eliminate some of them even before the next round begins
  - E.g. if {A}, {B}, and {C} are frequent in iteration 1 while {D} is not frequent, then iteration 2 will consider only {A, B}, {A, C}, and {B, C}

- Create rules from these itemsets that meet a minimum confidence threshold
  - Given the set of frequent itemsets, association rules are generated from all possible subsets
  - For instance, {A, B} would result in candidate rules for {A} -> {B} and {B} -> {A}
  - These are evaluated against a minimum confidence threshold, and any rules that do not meet the desired confidence level are eliminated

Apriori algorithm

- First, we want to find the association rule
- We assume minimum confidence and minimum support
- Association rule ($X \Longrightarrow Y$) exists if

$$(fr(X \cup Y) \geq \text{min\_}sup) \land (conf(X \Longrightarrow Y) \geq \text{min\_}conf)$$

- It should be fine to find all frequent subsets from R
- We shall explore $\forall X\ \forall Y\ ((X \subseteq R) \land (Y \subseteq X))$ the association rule $(X - Y) \Longrightarrow Y$
- There are 2^$|R|$ elements that may be treated as candidates

Apriori algorithm

- How to check the properties of frequent sets?
- Prune the search space
- Given X and Y with $Y \subseteq X$
  - When $fr(Y) \geq fr(X)$ if X is frequent, Y is also frequent
  - If any of the subset Y from X is not frequent, then X is not frequent at all
  - This phenomenon is called as anti-monotone property of support
- Feasible exploration
  - Find the frequent sets starting from minimal size and then increase the size
  - Prune the candidates (include infrequent itemsets)

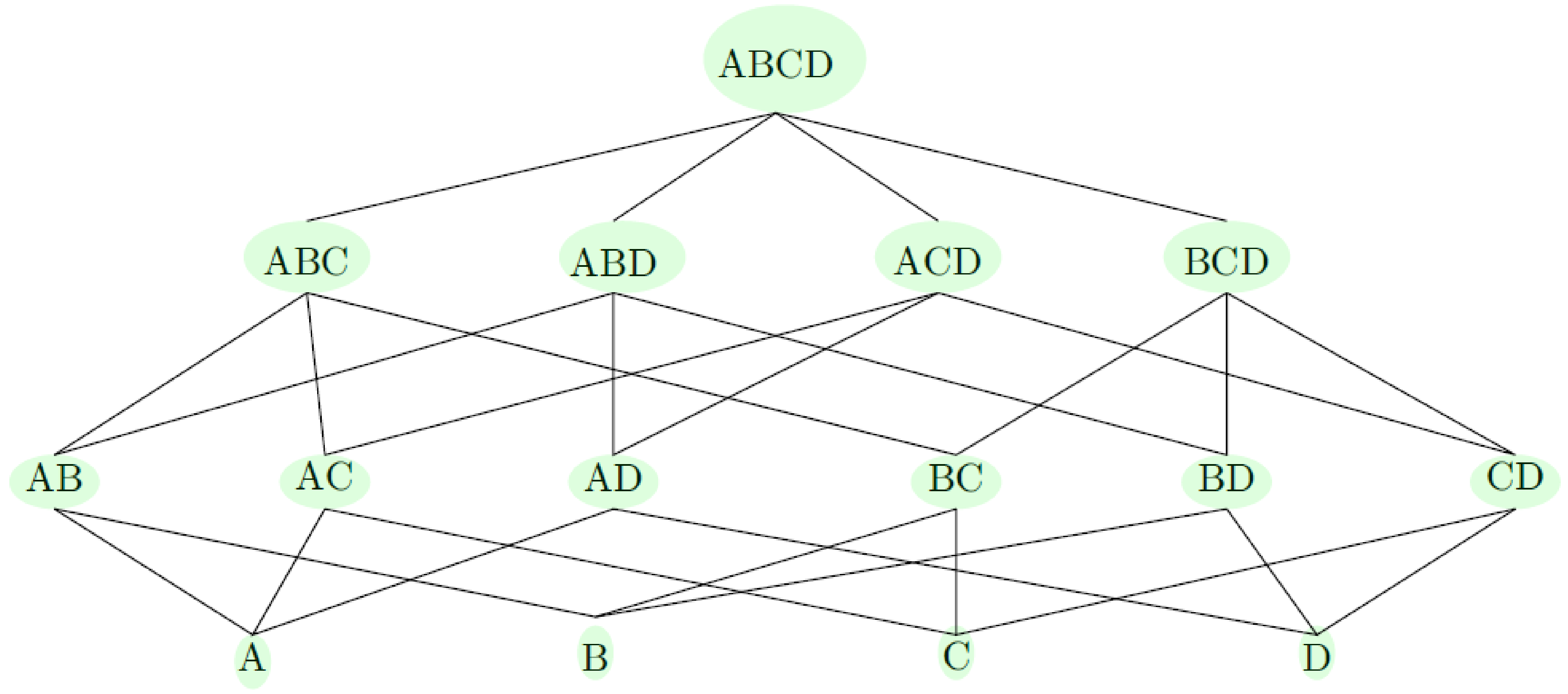
Apriori algorithm

- $\mathcal{F}\_I$ $(r$,min_$sup)$ is the set of frequent sets from R of I size
- Given a set of patterns of length I, the only frequent set candidates of length I+1 will be those which all subsets are in the frequent sets of length I

$$C(\mathcal{F}\_(I+1)\ (r))=\{X\subseteq R/|X|\ =I+1\wedge\forall\_Y\ (Y\subseteq X\wedge|Y|=I\Longrightarrow Y\in\mathcal{F}\_I\ (r))\}$$

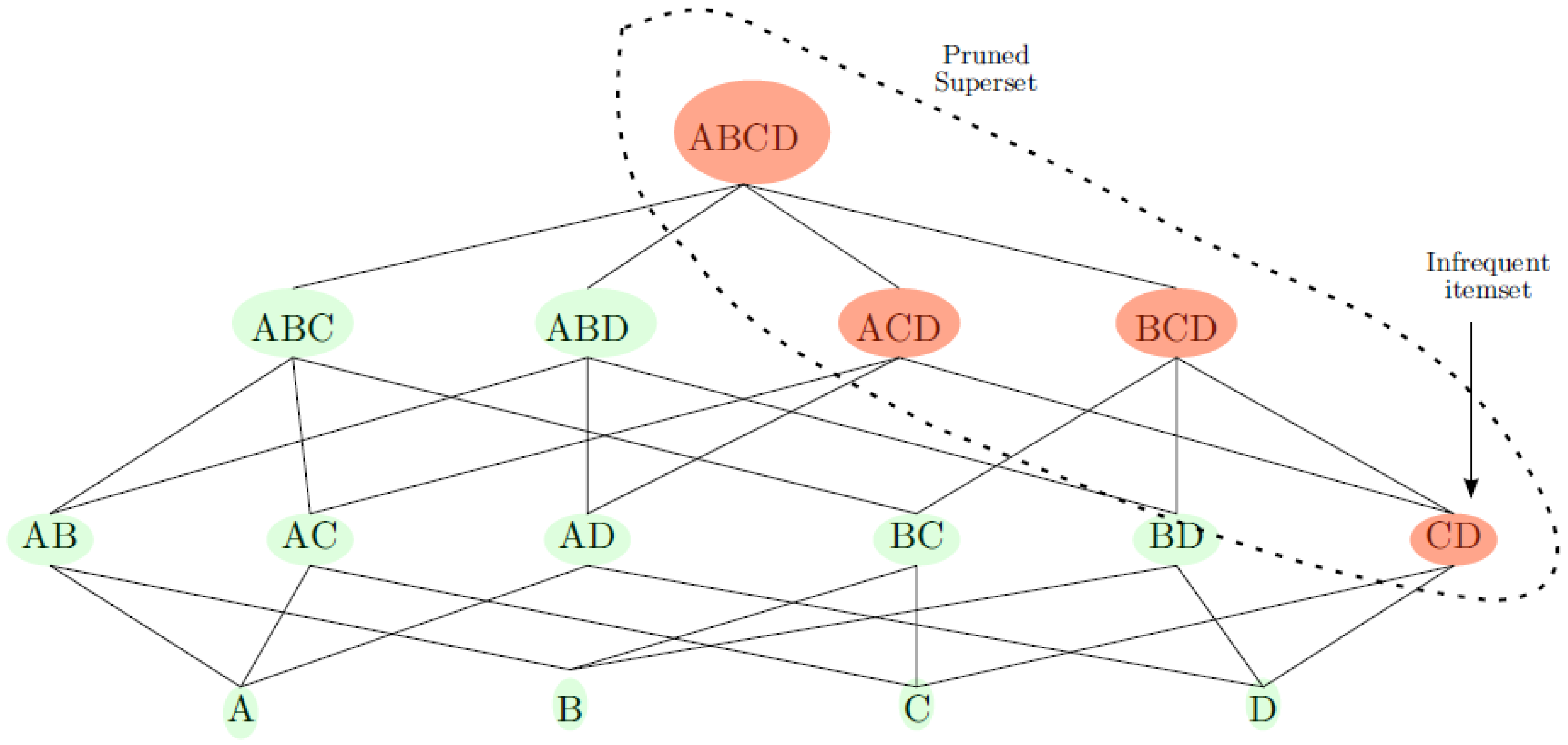
- Computation of association rules may be done iteratively
- Start with the smallest frequent subsets until no more candidates are obtained

Apriori algorithm



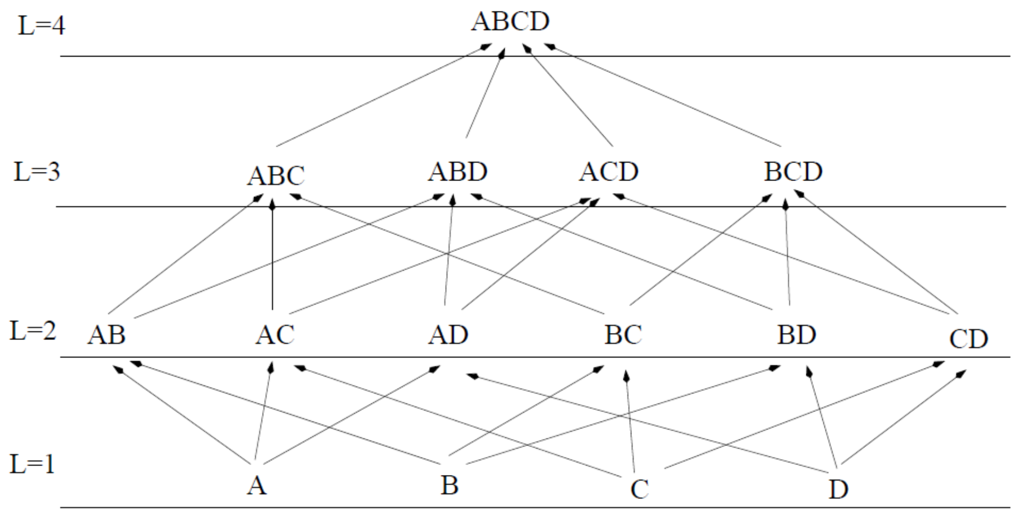
Source:
Bejar 2015

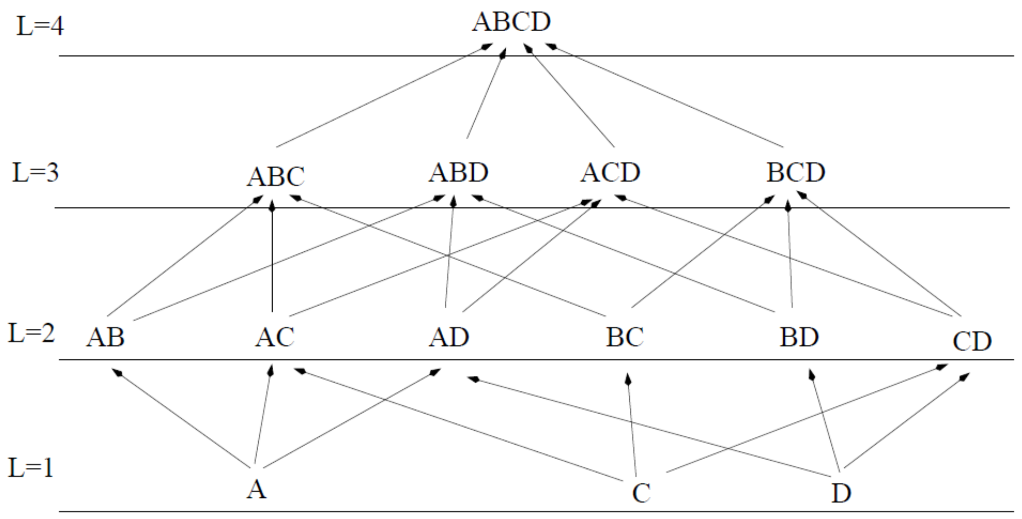
Apriori algorithm



Source:
Bejar 2015

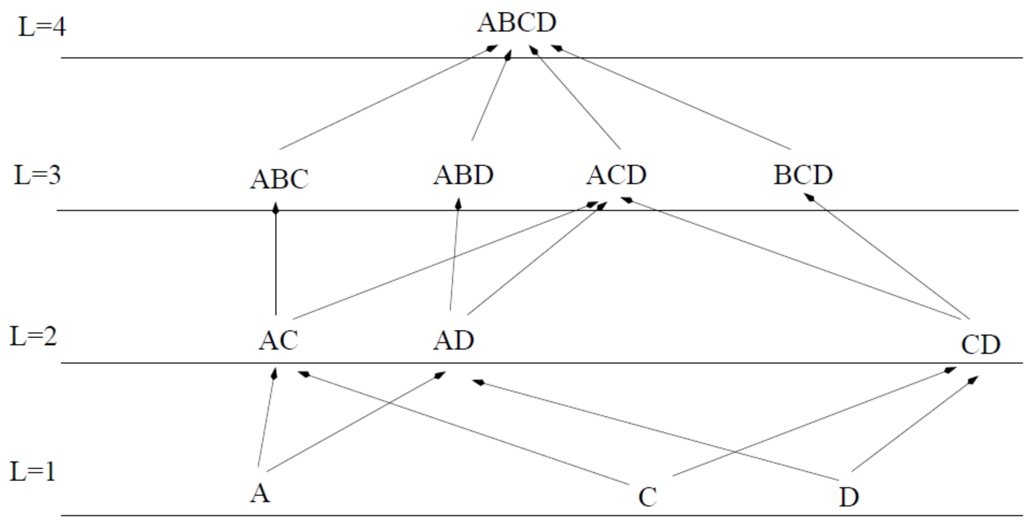
Apriori algorithm

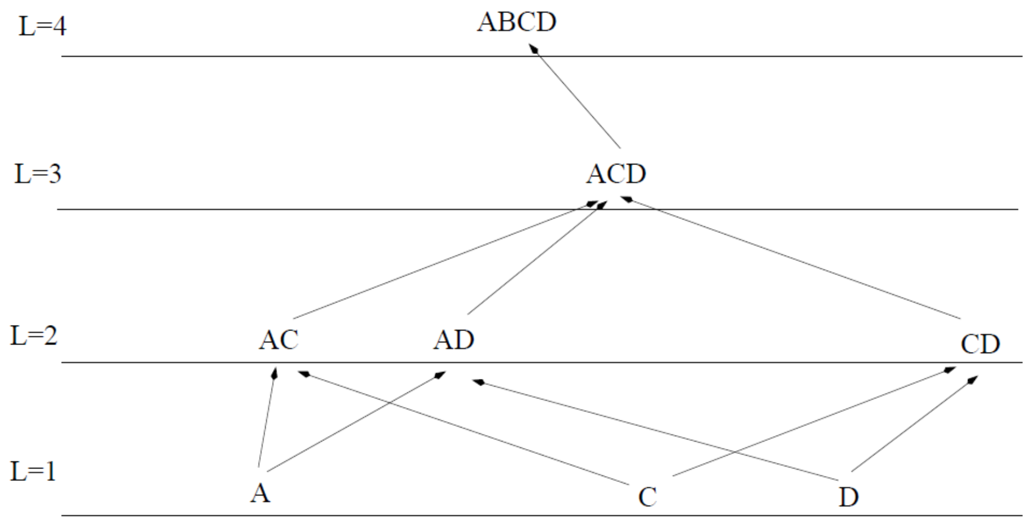




Source:
Bejar 2015

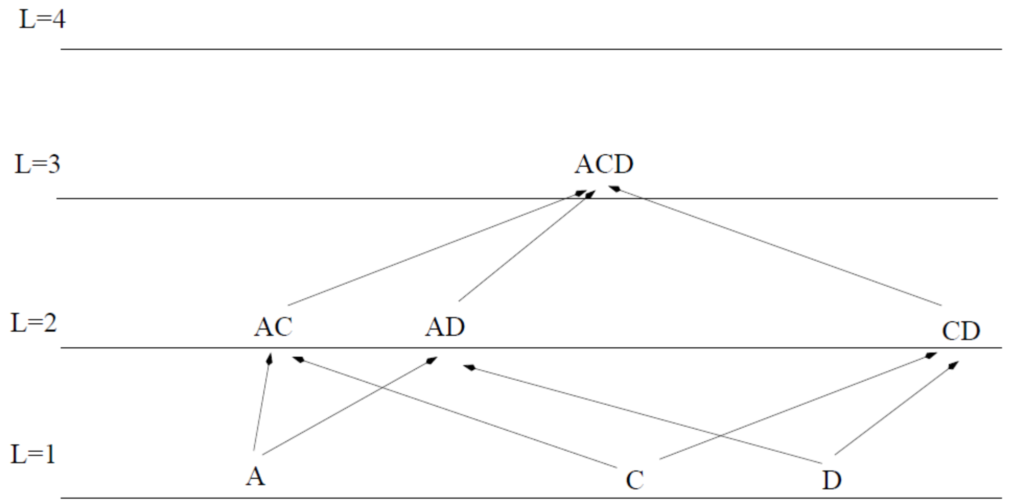
Apriori algorithm





Source:
Bejar 2015

Apriori algorithm



Source:
Bejar 2015

Apriori algorithm

- Outcome of support
  - Computational cost of the search depends on the value of min_sup
    - Is too high, just a few patterns will be developed [interesting, but rare occurrences will be dropped]
    - If too low, the computational cost will be very high [too many associations will be present]
- The threshold value is not knowne before the computation
- Multiple minimum supports are often necessary to be considered
- The more items present, the more space needed to count the support
- Size of the database matters – the algorithm has to perform multiple passes in order to count the support
- Average transaction width has an influence of the maximum length of frequent itemsets

Apriori algorithm

- Pruning strategies
  - Itemsets may be redundant as they have identical support as the supersets
  - Thus, we may focus on those itemsets and reduce the number of candidates
    - Maximal frequent itemsets [none of the immediate supersets of the frequent itemset are frequent]
    - Closed frequent itemsets [none of the immediate supersets of the frequent itemset have the same support and the support is larger that the $min\_support$]

Apriori algorithm

STRENGTHS

- Ideally suited for working with very large amounts of transactional data
- Results in rules that are easy to understand
- Useful for data mining and discovering unexpected knowledge in databases

WEAKNESSES

- Not very helpful for small datasets
- Takes effort to separate the insight from the common sense
- Easy to draw spurious conclusions from random pattern
- Execution time is more as wasted in producing candidates for rules every time - its computational cost is very high

Frequent pattern growth (FP-growth)

FP-growth

- We would like to extract association rules by using some specialized data structures
- Apriori approach: the numer of candidates to explore in order to find long patterns may be very large (computation problems)
- FP-growth tries to obtain patterns without candidate generation and is based on a specialized data structure, FP-tree
- FP-tree summarizes the frequency of the patterns in the database
- The patterns are explored incrementally by adding prefixes without candidate generation

FP-growth

- Our goal is to avoid querying the database to compute the frequency of patterns
- We assume that there is an order among the elements of a transaction and it allows to obtain some common prefixes from the transaction
- Then, the transactions with common prefixes are merged in the structure maintaining the frequency of each subprefix

FP-growth

- The logic
  - Compute the frequency of each indivitual item in the database
  - Create the tree root
  - For every transaction
    - Pick the frequent items
    - Order the items (by their original frequency)
    - Iterate for each item
    - Insert it in the tree
      - If the node has a descendant equal to the actual item, increase its frequency
      - Otherwise, new node is created

FP-growth

- Select all the paths that contain the item (prefix paths)
- Convert all those paths, which contain the item onto a conditional FP-tree
  - Update the counts along the path by using the frequency of the item
  - Truncate the paths: remove the nodes for the item
  - Eliminate from the paths the items, which are no longer frequent
- For all the items, which are previous in order that are frequent in the conditional FP-tree
  - Treat the prefix as frequent
  - Find the frequent items for the prefix recursively

FP-growth

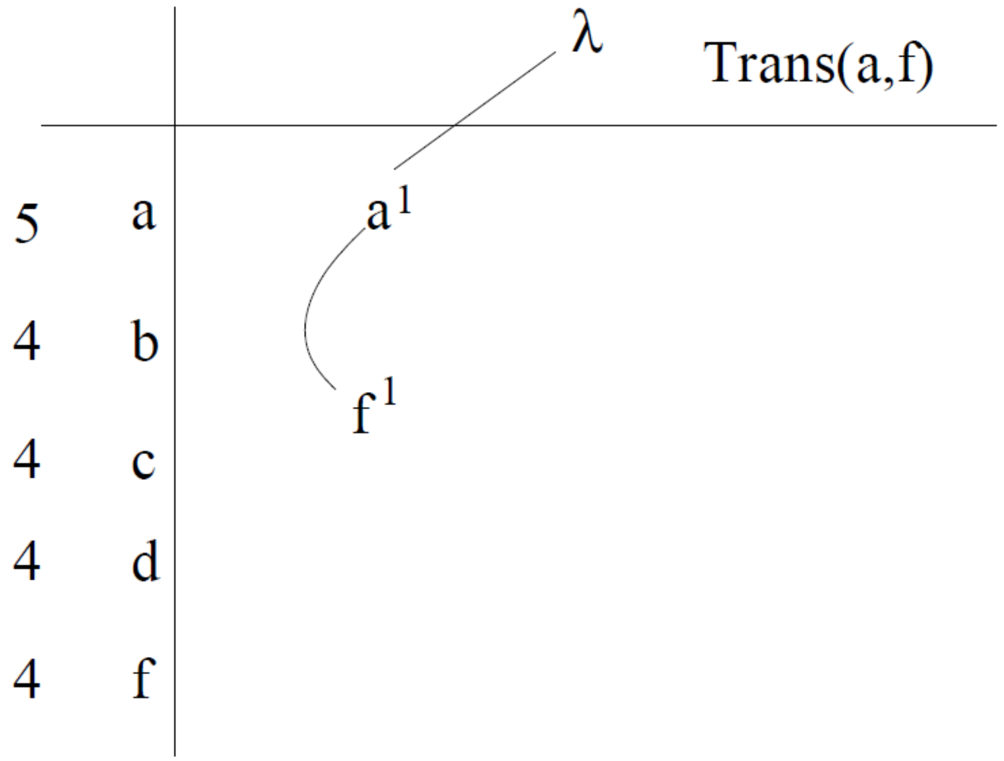
BD	Transactions
1	a, e, f
2	a, c, b, f, g
3	b, a, e, d
4	d, e, a, b, c
5	c, d, f, b
6	c, f, a, d

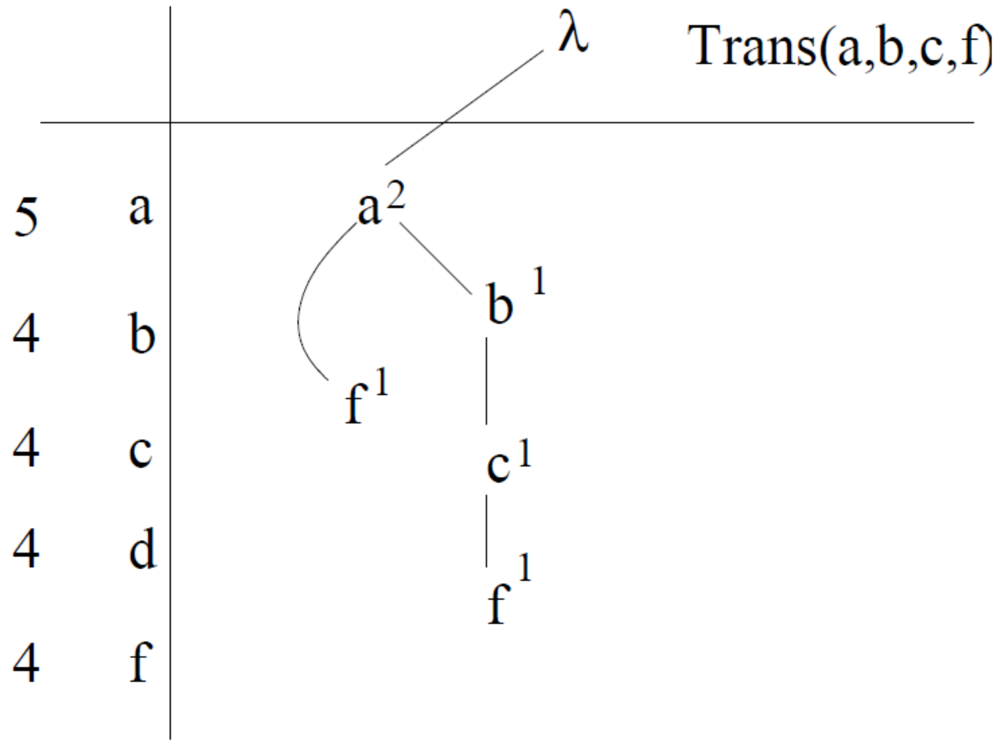
Item	frequency
a	5
b	4
c	4
d	4
e	3
f	4
g	1

BD	Transactions (fr=4)
1	a, f
2	a, c, b, f
3	a, b, d
4	a, b, c, d
5	b, c, d
6	a, c, d, f

Source:
Bejar 2015

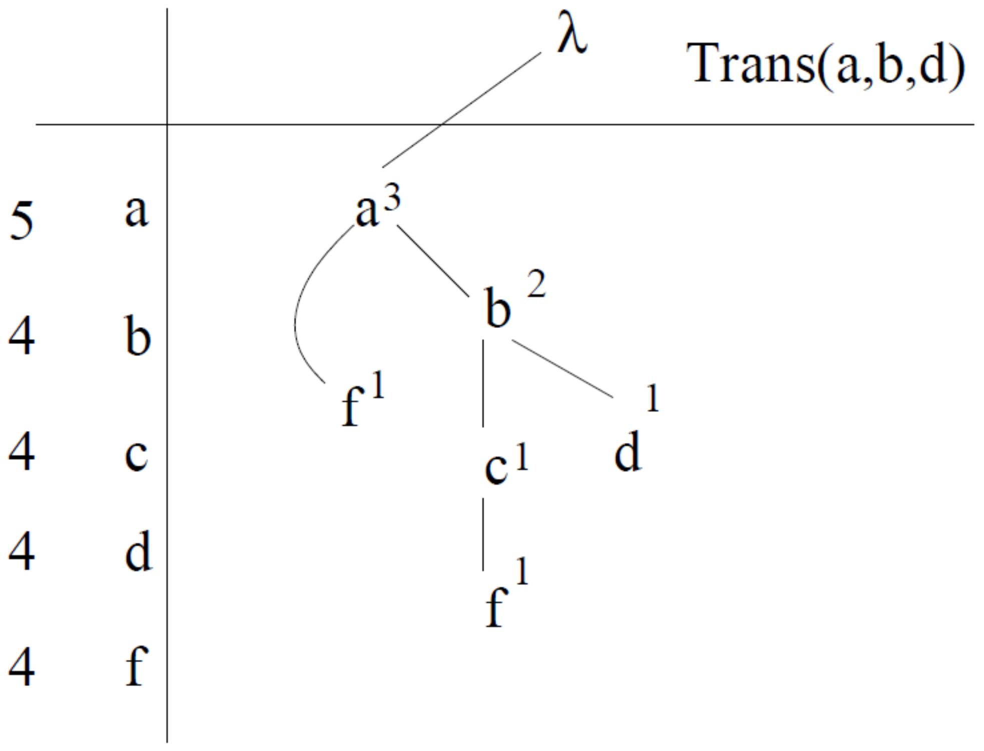
FP-growth

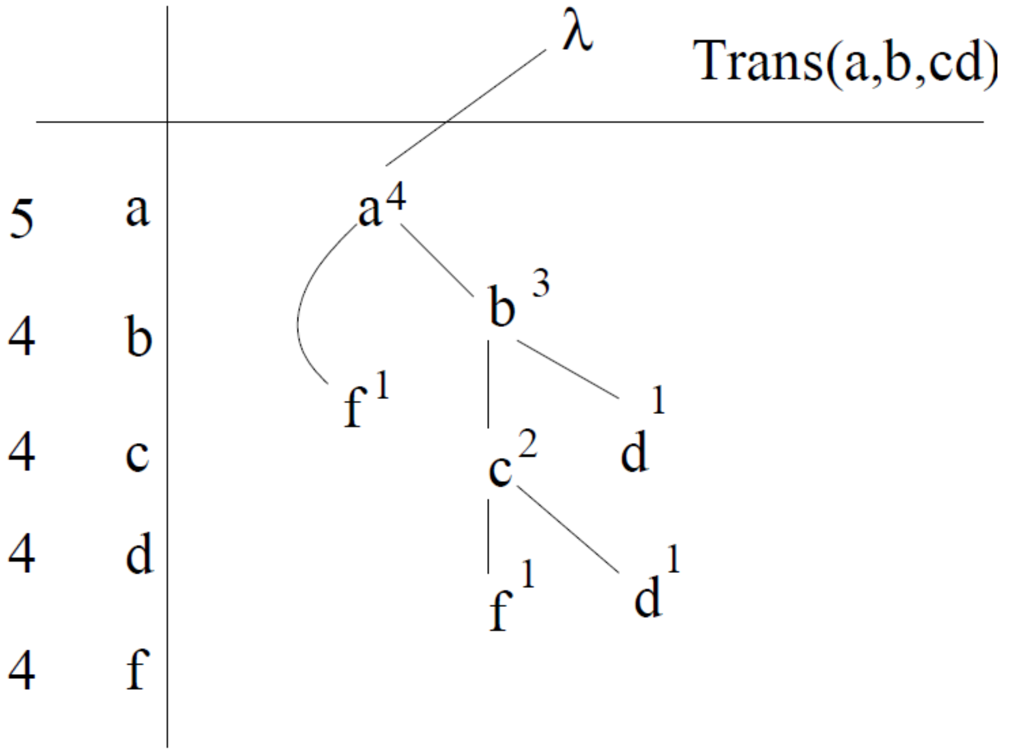




Source:
Bejar 2015

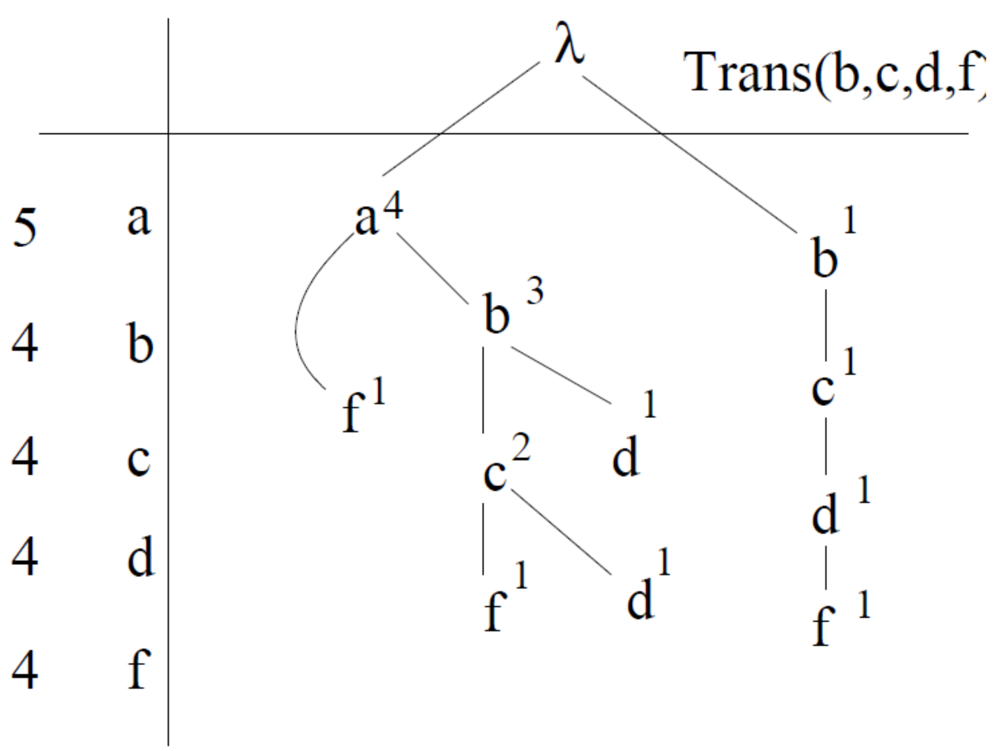
FP-growth

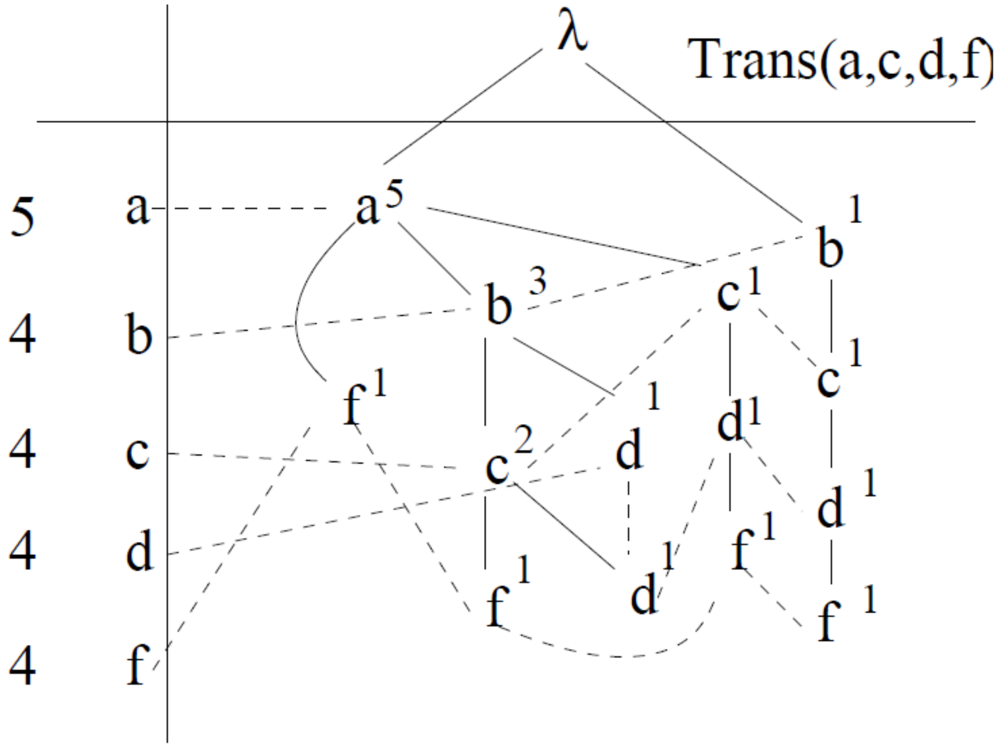




Source:
Bejar 2015

FP-growth

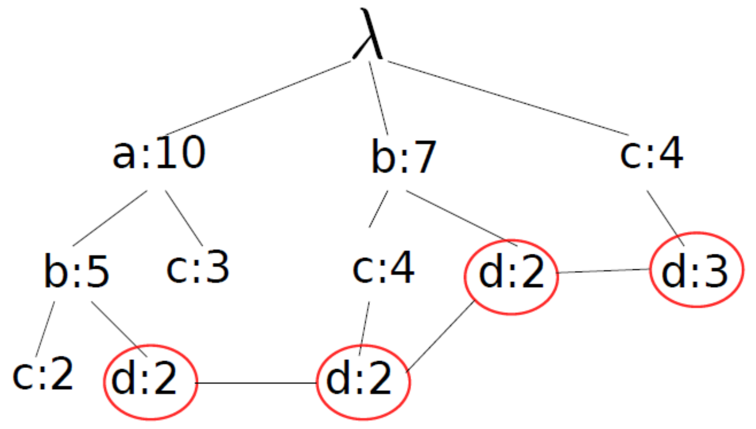




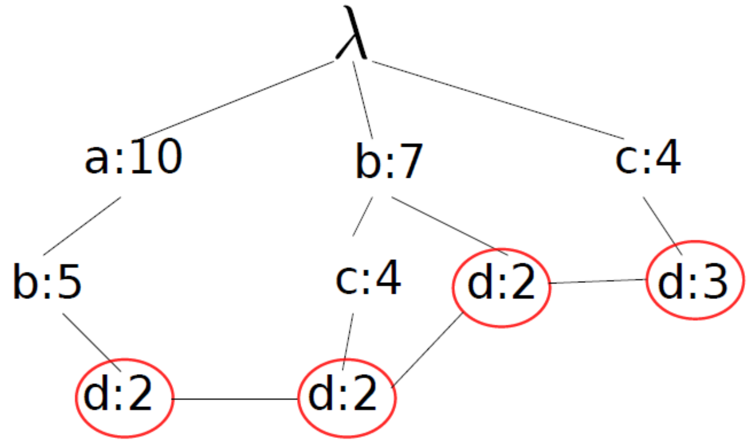
Source:
Bejar 2015

FP-growth

Extract patterns with suffix *d*



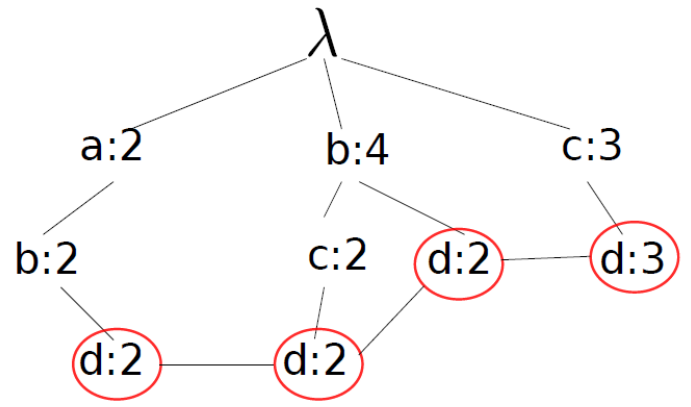
Keep only the paths that contain *d* (minsupport = 2)



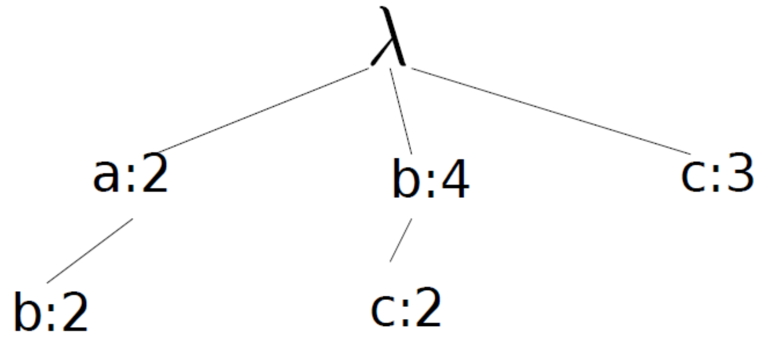
Source:
Bejar 2015

FP-growth

Generate the **conditional FP-tree** for *d* (updating the counts)



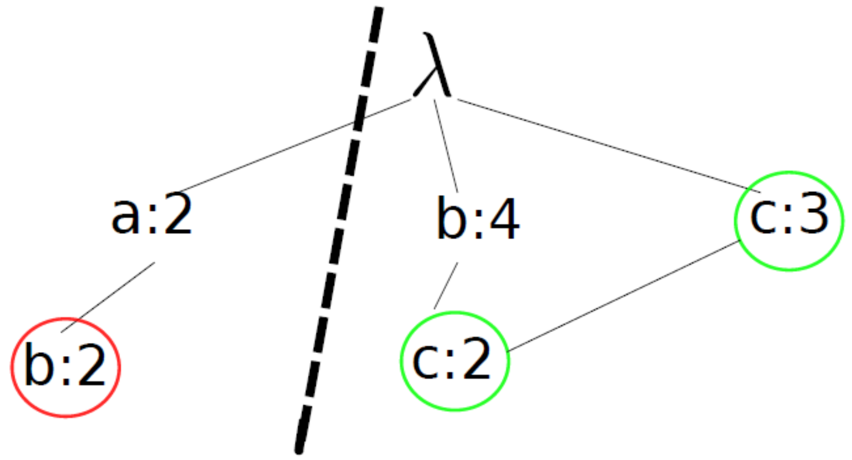
Prune the path eliminating *d* (is no longer needed)



Source:
Bejar 2015

FP-growth

Solve the problem for the predecesors of *d*, in this case *b* and *c* (we are looking if *bd* and *cd* are frequent)



If we continue the algorithm, the patterns extracted would be {[d], [bd], [cd], [abd], [bcd]}

Source:
Bejar 2015

FP-growth

STRENGTHS

- Relatively fast
- Does not require candidate selection
- Use compact data structure
- Tree structure is informative

WEAKNESSES

- FP tree is expensive to build and may not fit in memory

Equivalence Class Clustering and bottom-up Lattice Traversal (ECLAT)

ECLAT

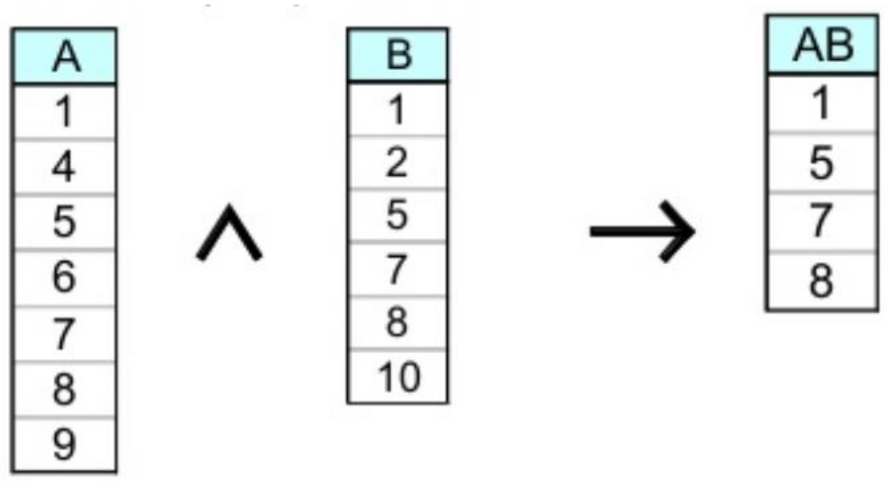
- Used for frequent itemset generation
- While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph
- For every item, store a list of transaction tids in a vertical data layout
- Determine suport of any $k\text{−}itemset$ by intersecting tid-list of two of its subsets $(k\text{−}1)$
- We may take top-down, bottom-up or hybrid approach
- Very fast computing

ECLAT

- In the first call of the function, all single items are used along with their tidsets
- Then the function is called recursively and in each recursive call, each item-tidset pair is verified and combined with other item-tidset pairs
- This process is continued until no candidate item-tidset pairs can be combined
- In the output there are the itemsets which are frequent
- In fact, it calculates the support for the frequent items

ECLAT

- This algorithm uses simple intersection operations for equivalence class clustering along with bottom-up lattice traversal



ECLAT

STRENGTHS

- Fast computing
- Does not involve the repeated scanning of the data to compute the individual support values

WEAKNESSES

- Intermediate tid-lists may become too large for memory

Thank you!