InfoLab

IC612: Data Warehousing and Data Mining

Lecture 4:
Association Pattern Mining

Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Introduction

- **Definition of classical problem**
  - transactions: supermarket data containing sets of items bought by customers
  - goal: find associations between groups of items bought by customers
- **Examples**
  - Supermarket data
    - itemset: a frequent pattern in the context of supermarket items bought by a customer
    - frequent itemsets: provide useful insights about target marketing and shelf placement of the items
  - Text mining
  - Generalization to dependency-oriented data types
  - Clustering, classification, and outlier analysis, …

Association rules

- **Association rules of the form X ⇒ Y**
  - X and Y : sets of items (especially, frequent itemsets)
  - e.g., {Beer} ⇒ {Diapers}: buying beer makes it more likely that diapers will also be bought
  - e.g., {Eggs,Milk} ⇒ {Yogurt}: supermarket owner can promote yogurt to customers who often buy eggs and milk

- **Frequency-based model is very popular**
  - for association pattern mining because of its simplicity

- **Numerous models for frequent pattern mining are based on statistical significance**

Data for frequent pattern mining

- **Database T : a set of n transactions (T_1 . . . T_n)**
  - each transaction T_i is drawn on the universe of items U
    - unordered set-wise data
    - a single transaction will typically contain less than 50 items
  - T_i also can be represented as a multidimensional record
    - d = |U|, containing only binary attributes
    - typically, tens of thousands of items

- **Itemset : a set of items**
  - k-itemset : an itemset that contains exactly k items
    - in other words, set of items of cardinality k

Support

Definition 4.2.1 (Support) *The support of an itemset I is defined as the fraction of the transactions in the database $\mathcal{T} = \{T_1 \ldots T_n\}$ that contain I as a subset.*

- **sup(I): support of an itemset I**
- **Example**
  - sup({Bread,Milk}) = 2/5 = 0.4
  - sup({Cheese, Yogurt}) = 0.2

tid	Set of Items	Binary Representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

Example of a snapshot of a market basket data set

Frequent pattern mining

Definition 4.2.2 (Frequent Itemset Mining) *Given a set of transactions* $\mathcal{T} = \{T_1 \ldots T_n\}$*, where each transaction* T_i *is a subset of items from* U*, determine all itemsets* I *that occur as a subset of at least a pre-defined fraction minsup of the transactions in* $\mathcal{T}$*.*

- **minsup: minimum support**
  - items that are correlated will frequently occur together in transactions
  - such itemsets will have high support
  - e.g., minsup = 0.3, then only {Bread,Milk} will be reported

tid	Set of Items	Binary Representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

Example of a snapshot of a market basket data set

- **Frequent itemsets**
  - 1-itemsets: {Bread}, {Milk}, {Eggs}, {Cheese}, {Yogurt}
  - 2-itemsets: {Bread,Milk}, {Eggs,Milk}, {Cheese,Milk}, {Eggs,Yogurt}, {Milk,Yogurt}
  - 3-itemsets: {Eggs,Milk,Yogurt}
- **Lower minsup yields a larger number of frequent patterns**
  - if minsup is too high, then no frequent patterns will be found
  - appropriate choice of minsup is crucial

tid	Set of Items	Binary Representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

Example of a snapshot of a market basket data set

Monotonicity and Downward closure

Property 4.2.1 (Support Monotonicity Property) *The support of every subset J of I is at least equal to that of the support of itemset I.*

$$sup(J) \geq sup(I) \quad \forall J \subseteq I \tag{4.1}$$

Property 4.2.2 (Downward Closure Property) *Every subset of a frequent itemset is also frequent.*

- **Downward closure property is algorithmically very convenient**
  - it provides an important constraint on the inherent structure of frequent patterns
    - this constraint is often leveraged to prune the search process and achieve greater efficiency
  - it can also create concise representations of frequent patterns

Maximal frequent itemsets

Definition 4.2.4 (Maximal Frequent Itemsets) *A frequent itemset is maximal at a given minimum support level minsup, if it is frequent, and no superset of it is frequent.*

- **Example (minsup = 0.3)**
  - {Eggs,Milk,Yogurt} is maximal
  - {Eggs,Milk} is not maximal
    - because it has a superset that is also frequent
  - only 3 maximal frequent itemsets (# of frequent itemsets is 11)

tid	Set of Items	Binary Representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

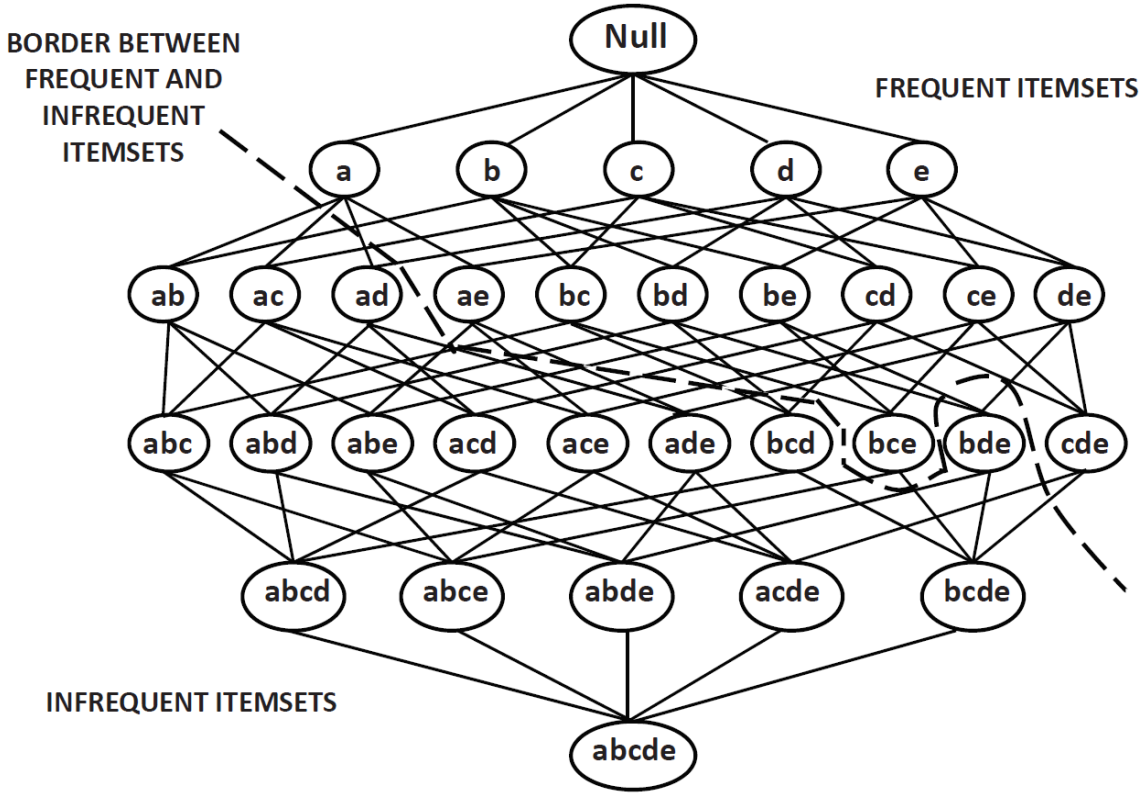
Example of a snapshot of a market basket data set

Closed frequent itemsets

- **All frequent itemsets can be derived from the maximal patterns**
  - by enumerating the subsets of the maximal frequent patterns
- **Condensed representations of the frequent patterns**
  - does not retain information about the support values of the subsets
  - closed frequent itemsets: maximal patterns that retain information about the support values
- **Example**
  - support of {Eggs,Milk,Yogurt} is 0.4
  - but it does not provide any information about the support of {Eggs,Milk}, which is 0.6, and so not closed frequent itemsets

Lattice of itemsets

- **Properties of itemsets can be conceptually arranged in the form of a lattice of itemsets**
  - # of nodes: 2^|U|
- **All frequent pattern mining algorithms, implicitly or explicitly, traverse lattice search space**
  - border separates the lattice into frequent and infrequent itemsets



Association rule generation framework

- **Frequent itemsets can be used to generate association rules with the use of confidence**
  - itemset *X* : *antecedent* of the rule
  - itemset *Y* : *consequent* of the rule

Definition 4.3.1 (Confidence) *Let X and Y be two sets of items. The confidence $conf(X \cup Y)$ of the rule $X \cup Y$ is the conditional probability of $X \cup Y$ occurring in a transaction, given that the transaction contains X. Therefore, the confidence $conf(X \Rightarrow Y)$ is defined as follows:*

$$conf(X \Rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)} \tag{4.2}$$

- **Example**
  - sup({Eggs,Milk}) = 0.6, sup({Eggs,Milk,Yogurt}) = 0.4
  - confidence of the rule {Eggs,Milk} ⇒ {Yogurt} is (0.4/0.6) = 2/3

- **Association rules are defined using both support and confidence criteria**
  - minsup
  - minconf: minimum confidence threshold
    - used to generate the most relevant association rules

Definition 4.3.2 (Association Rules) *Let X and Y be two sets of items. Then, the rule $X \Rightarrow Y$ is said to be an association rule at a minimum support of minsup and minimum confidence of minconf, if it satisfies both the following criteria:*

1. *The support of the itemset $X \cup Y$ is at least minsup.*
2. *The confidence of the rule $X \Rightarrow Y$ is at least minconf.*

- **Two phases of association rule generation**
  1. all the frequent itemsets are generated
     - more computationally intensive (more interesting part of the process)
  2. the association rules are generated from the frequent itemsets

Generating association rules

- **Assume that a set of frequent itemsets F is provided**
- **Generating the rules for each itemset I ∈ F**
  - partition the set I into all possible combinations of sets X and Y (Y = I −X, such that I = X ∪ Y)
  - rule X ⇒ Y is determined if it satisfies the minconf requirement

Property 4.3.1 (Confidence Monotonicity) *Let X_1, X_2, and I be itemsets such that $X_1 \subset X_2 \subset I$. Then the confidence of $X_2 \Rightarrow I - X_2$ is at least that of $X_1 \Rightarrow I - X_1$.*

$$conf(X_2 \Rightarrow I - X_2) \geq conf(X_1 \Rightarrow I - X_1) \quad (4.3)$$

- **Example**
  - conf({Bread} ⇒ {Butter,Milk}) ≤ conf({Bread,Butter} ⇒ {Milk})
  - both have the same numerator
  - but, the second rule has a smaller denominator than the first rule

Frequent Itemset Mining Algorithms

- Brute Force Algorithms
- The Apriori Algorithm
- Enumeration-Tree Algorithms
- Recursive Suffix-based Pattern Growth Methods

Brute Force Algorithms

- **Terms**
  - **candidate itemsets** : itemsets that might possibly be frequent
  - **support counting** : verify candidates against T
    - check whether a given itemset *I* is a subset of each transaction $T_i \in T$

- **Generate all these candidate itemsets, and count their support against the transaction database T**
  - for a universe of items U, there are 2^|U| − 1 distinct subsets
  - this exhaustive approach is likely to be impractical
  - e.g., d = |U| = 1000, a total of 2^1000 > 10^300 candidates

Making the brute-force approach faster

- no (k + 1)-patterns are frequent if no k-patterns are frequent
  - downward closure property
- one can enumerate and count the support of all the patterns with increasing length
  - enumerate and count the support of all patterns containing one item, two items, and so on, until for a certain length l
  - for sparse transaction databases, the value of l is typically very small
- this approach is orders of magnitude faster, but still not satisfactory for large values of U
  - $|U|$ = 1000 and l = 10, $\sum_{i=1}^{10} \binom{|U|}{i}$ is of the order of 10^23

Approaches for better efficiencies

① **Reducing the size of the explored search space by pruning candidate itemsets (i.e., lattice nodes)**
- e.g., the **Apriori** algorithm

② **Pruning transactions that are known to be irrelevant for counting a candidate itemset**
- for more efficient support counting
- e.g., enumeration-tree algorithms (**Eclat**)

③ **Using compact data structures to represent either candidates or transaction databases**
- for more efficient support counting
- counting of candidate supports is the most expensive part
- e.g., recursive suffix-based pattern growth methods (**FP-growth**)

The Apriori Algorithm

- **Use the downward closure property in order to prune the candidate search space**
  - frequent k-itemsets are used to restrict the number of (k + 1)-candidates
  - it is extremely important to keep the number of candidates low

- **Assumption**
  - the items in U have a lexicographic ordering
  - an itemset {a, b, c, d} can be treated as a string “abcd” of items

Example of Apriori

minsup= 2

Database TDB

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

1st scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

F_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

F_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

F_3

Itemset	sup
{B, C, E}	2

Algorithm of Apriori

Algorithm *Apriori*(Transactions: $\mathcal{T}$, Minimum Support: *minsup*)
begin
  $k = 1$;
  $\mathcal{F}_1 = \{$ All Frequent 1-itemsets $\}$;
  while $\mathcal{F}_k$ is not empty **do begin**
    Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
    Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
    Determine $\mathcal{F}_{k+1}$ by support counting on $(\mathcal{C}_{k+1}, \mathcal{T})$ and retaining
      itemsets from $\mathcal{C}_{k+1}$ with support at least *minsup*;
    $k = k + 1$;
  end;
  return$(\cup_{i=1}^{k} \mathcal{F}_i)$;
end

Step 1: Generating candidates

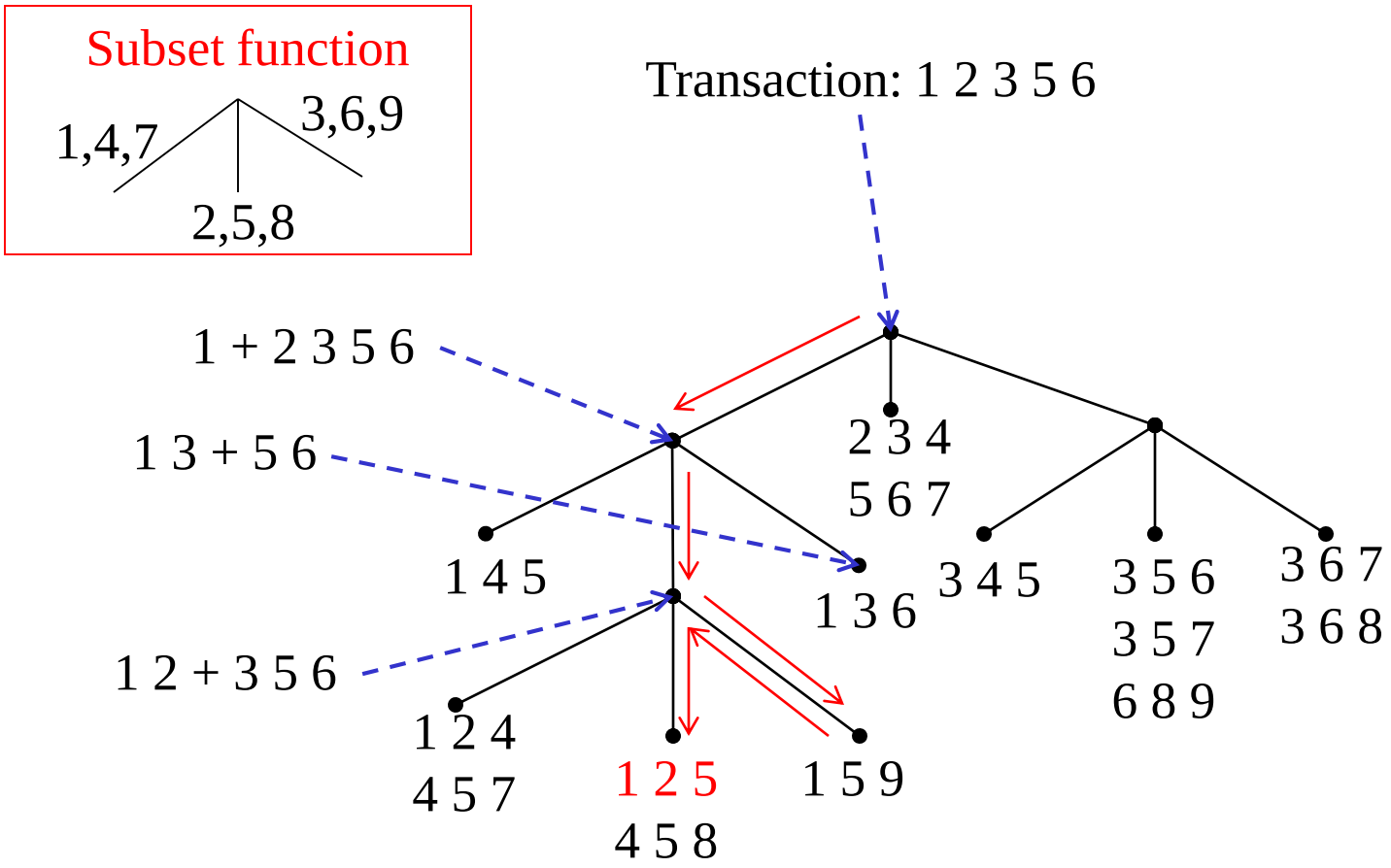
- **How to generate candidates?**
  - Step 1: self-joining C_k
  - Step 2: pruning
- **Example of Candidate-generation**
  - C_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: C_3*C_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in C_3
  - C_4 = {*abcd*}

Step 2: Support counting

- **Why counting supports of candidates a problem?**
  - The total number of candidates can be very huge
  - One transaction may contain many candidates

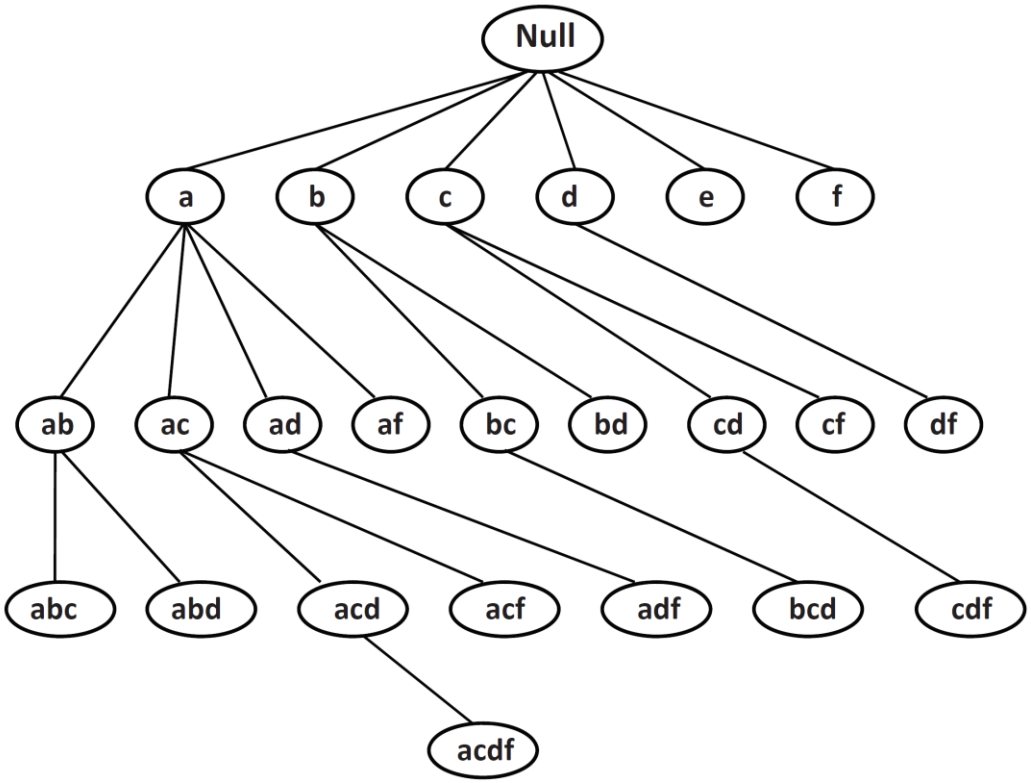
- **Method:**
  - Candidate itemsets are stored in a **hash-tree**
  - Node has a fixed degree
  - Leaf node : contains a list of itemsets and counts
    - every itemset in C_{k+1} is contained in exactly one leaf node
  - Interior node : contains a hash table
    - all interior nodes use the same hash function $f(\cdot)$
  - Subset function: finds all the candidates contained in a transaction

Support counting using hash tree



Enumeration-Tree Algorithms

- **Enumeration tree: systematic and non-redundant exploration of the itemset search space**
- **(frequent) tree extension**
  - item that is used to extend a given node to its (frequent) child
  - e.g., frequent tree extensions of node a : b, c, d, and f



lexicographic or enumeration tree of frequent itemsets

- **F(P) ⊆ C(P) ⊂ F(Q) is always true**
  - F(Q) : frequent lexicographic tree extensions of node Q
  - C(P) : candidate extension items of node P
  - P = Q ∪ {i}: the item i used to extend node Q to node P
  - e.g., when Q = a, F(Q) = {b, c, d, f}
  - e.g, when P = ab, F(P) = {c, d}, C(P) = {c, d, f}

Algorithm *GenericEnumerationTree*(Transactions: $\mathcal{T}$,
Minimum Support: *minsup*)
begin
Initialize enumeration tree $\mathcal{ET}$ to single *Null* node;
while any node in $\mathcal{ET}$ has not been examined **do begin**
Select one of more unexamined nodes $\mathcal{P}$ from $\mathcal{ET}$ for examination;
Generate candidates extensions $C(P)$ of each node $P \in \mathcal{P}$;
Determine frequent extensions $F(P) \subseteq C(P)$ for each $P \in \mathcal{P}$ with support counting;
Extend each node $P \in \mathcal{P}$ in $\mathcal{ET}$ with its frequent extensions in $F(P)$;
end
return enumeration tree $\mathcal{ET}$;
end

TreeProjection

- **Family of methods that uses recursive projections of the transactions down the enumeration tree structure**
  - c.f., Apriori algorithm implicitly extends the enumeration tree in a level-wise fashion
  - a variety of different strategies for construction of the enumeration tree: breadth-first, depth-first, or a combination of both
- **Projected databases at node P, T(P)**
  - transactions that do not contain the itemset P are not included at node P and its descendants
  - projected database at node P can be expressed only in terms of the items in C(P)
    - e.g, P=ab, C(P) = {c, d, f}, transaction *abcfg* maps to the projected transaction *cf* in T(P)
    - e.g., transaction *acfg* is not present in T(P) because P=*ab* is not a subset of *acfg*

DepthProject

- **TreeProjection with the depth-first strategy**

Algorithm *ProjectedEnumerationTree*(Transactions: $\mathcal{T}$,
Minimum Support: *minsup*)
begin
  Initialize enumeration tree $\mathcal{ET}$ to a single $(Null, \mathcal{T})$ root node;
  while any node in $\mathcal{ET}$ has not been examined **do begin**
    Select an unexamined node $(P, \mathcal{T}(P))$ from $\mathcal{ET}$ for examination;
    Generate candidates item extensions $C(P)$ of node $(P, \mathcal{T}(P))$;
    Determine frequent item extensions $F(P) \subseteq C(P)$ by support counting
      of individual items in smaller projected database $\mathcal{T}(P)$;
    Remove infrequent items in $\mathcal{T}(P)$;
    for each frequent item extension $i \in F(P)$ **do begin**
      Generate $\mathcal{T}(P \cup \{i\})$ from $\mathcal{T}(P)$;
      Add $(P \cup \{i\}, \mathcal{T}(P \cup \{i\}))$ as child of P in $\mathcal{ET}$;
    end
  end
  return enumeration tree $\mathcal{ET}$;
end

Vertical database representations

- **Each item is associated with a list of its tids**
  - transpose of the binary transaction data matrix representing
- **Intersection of k item tid lists for k-itemset**
  - e.g., intersection of the *tid* lists of *Milk* and *Yogurt*
  - e.g., intersection of the *tid* lists of *{Milk,Yogurt}* and *Eggs*
  - intersecting *tid* lists of *k*-itemsets is preferable to intersecting *tid* lists of 1-itemsets

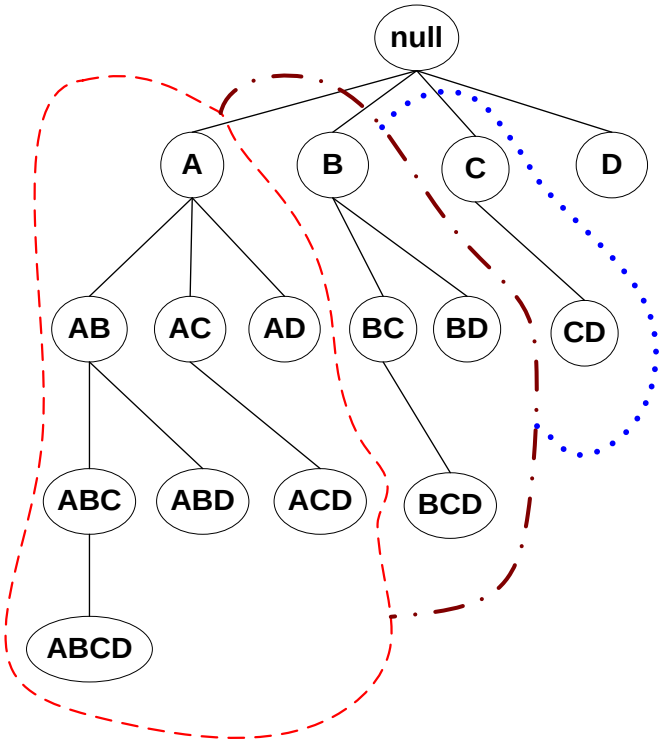
Item	Set of tids	Binary Representation
$Bread$	$\{1,3\}$	10100
$Butter$	$\{1\}$	10000
$Cheese$	$\{3,5\}$	00101
$Eggs$	$\{2,3,4\}$	01110
$Milk$	$\{1,2,3,4,5\}$	11111
$Yogurt$	$\{2,4,5\}$	01011

Vertical version of Apriori (Partition)

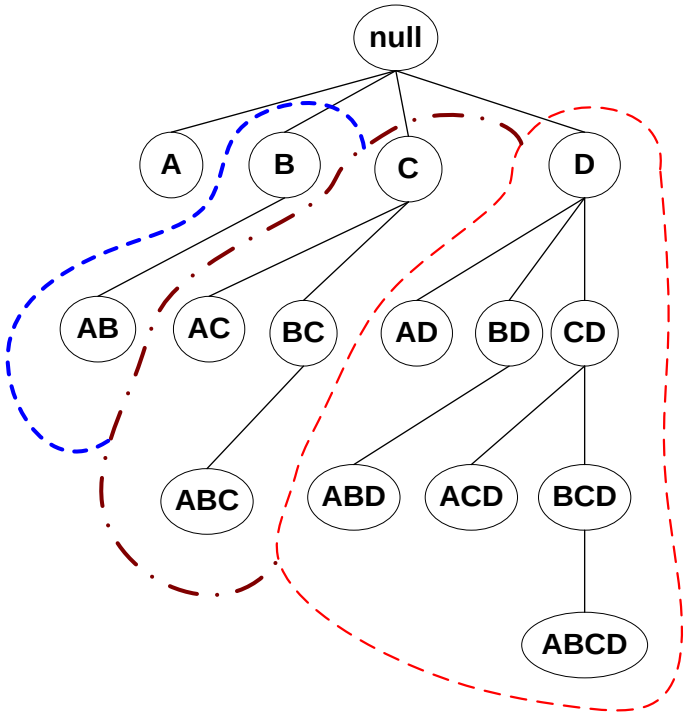
Algorithm *VerticalApriori*(Transactions: $\mathcal{T}$, Minimum Support: *minsup*)
begin
  $k = 1$;
  $\mathcal{F}_1 = \{$ All Frequent 1-itemsets $\}$;
  Construct vertical *tid* lists of each frequent item;
  while $\mathcal{F}_k$ is not empty **do begin**
    Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
    Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
    Generate *tid* list of each candidate itemset in $\mathcal{C}_{k+1}$ by intersecting
      tid lists of the itemset-pair in $\mathcal{F}_k$ that was used to create it;
    Determine supports of itemsets in $\mathcal{C}_{k+1}$ using lengths of their *tid* lists;
    $\mathcal{F}_{k+1}$= Frequent itemsets of $\mathcal{C}_{k+1}$ together with their *tid* lists;
    $k = k + 1$;
  end;
  return$(\cup_{i=1}^{k} \mathcal{F}_i)$;
end

Eclat

- **Independent Apriori-like breadth-first strategy is used on each of equivalence classes**
  - equivalence classe: the sublattice of itemsets with a common prefix
  - this can reduce the memory requirements by partitioning the candidate space into groups that are processed independently



(a) Prefix tree



(b) Suffix tree

Recursive Suffix-based Pattern Growth Methods

- **FP-Tree**
  - a space- and time- efficient way to implement the recursive pattern exploration
  - implemented with the use of arrays and pointers
- **Assumption**
  - items in the database are ordered with decreasing support
  - ordering of items is maintained within itemsets and transactions
- **Goal of a recursive call to the algorithm**
  - determine all the frequent patterns that have the suffix P
  - suffix P is empty at the top-level recursive call of the algorithm

Construct FP-tree from a Transaction Database

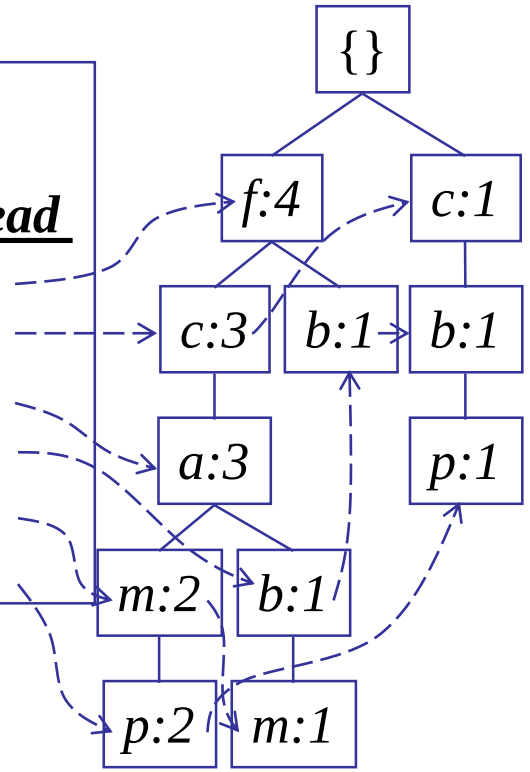
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



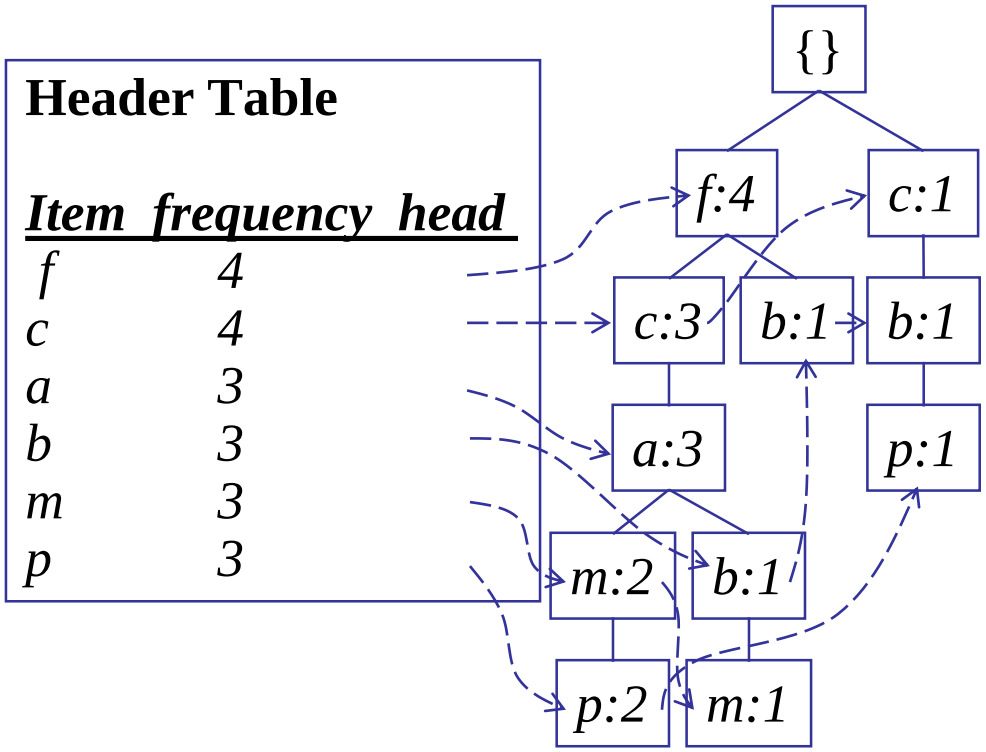
F-list = f-c-a-b-m-p

Partition Patterns and Databases

- **Frequent patterns can be partitioned into subsets according to f-list**
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- **Completeness and non-redundancy**

Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item p
- Accumulate all of transformed prefix paths of item p to form p's conditional pattern base

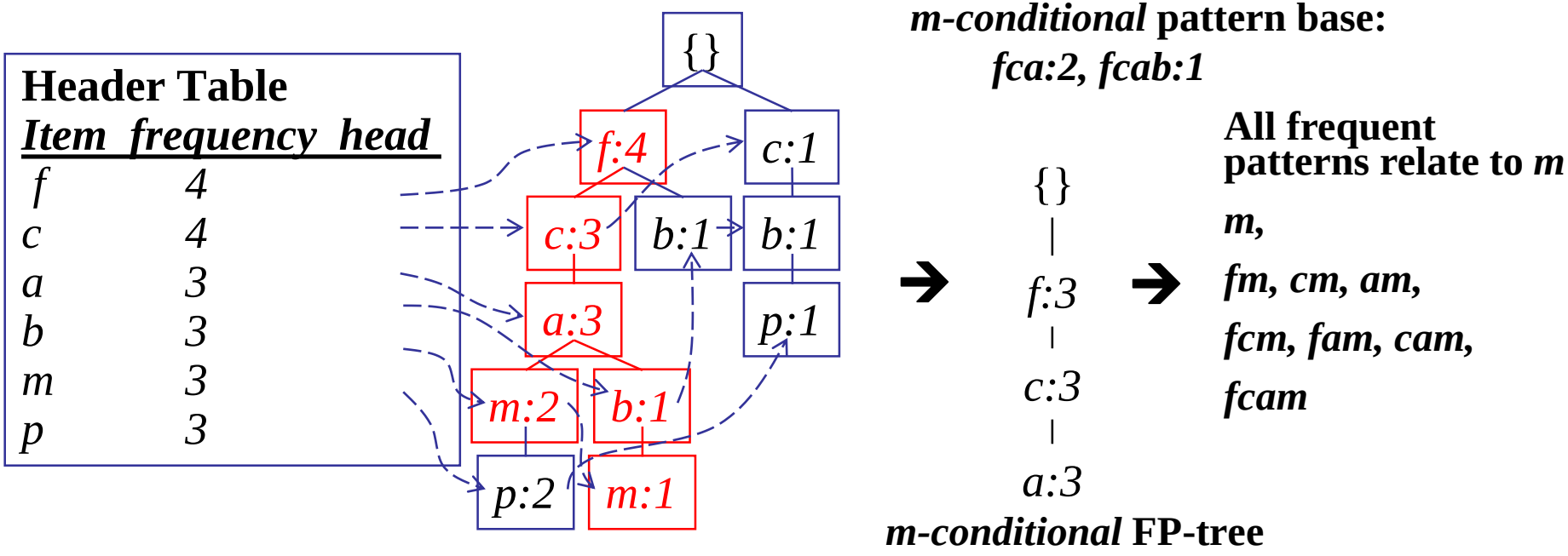


***Conditional* pattern bases**

item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

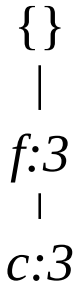
From conditional pattern-bases to conditional FP-trees

- **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

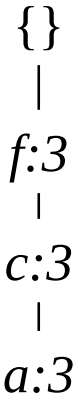


Recursion: Mining Each Conditional FP-tree

Cond. pattern base of "am": (fc:3)



***am-conditional* FP-tree**

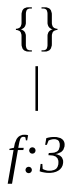


***m-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)

***cm-conditional* FP-tree**

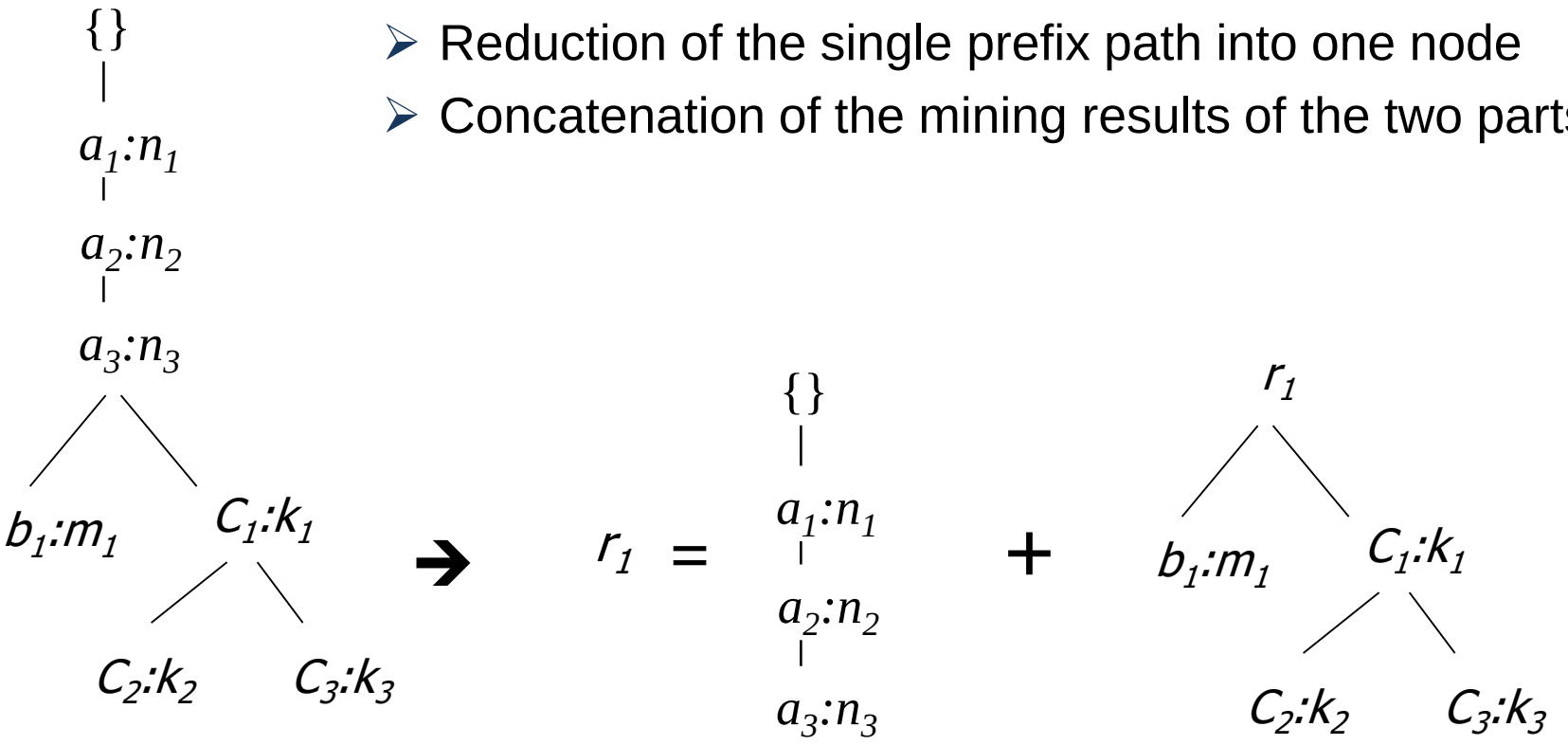
Cond. pattern base of "cam": (f:3)



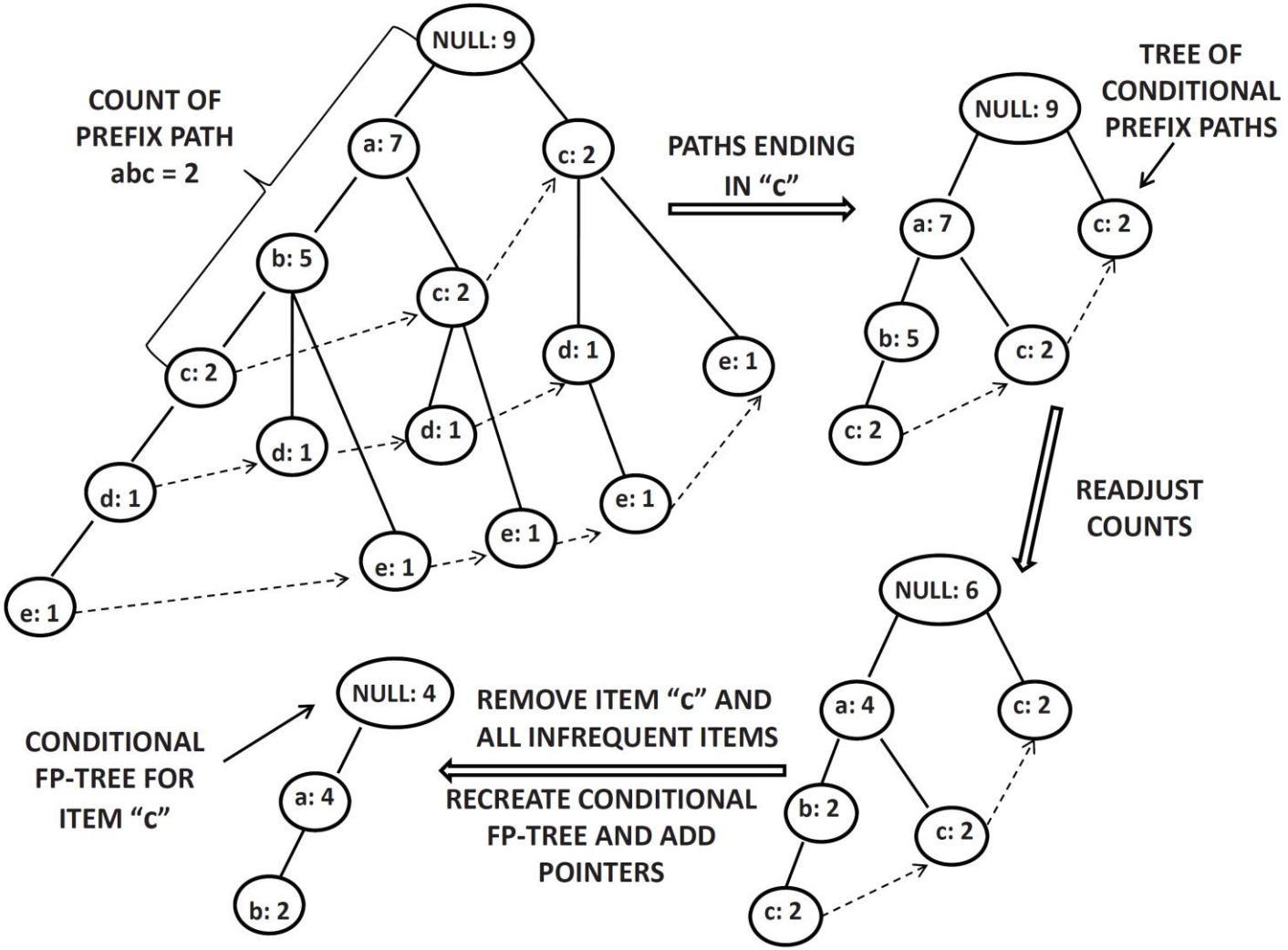
***cam-conditional* FP-tree**

A Special Case: Single Prefix Path in FP-tree

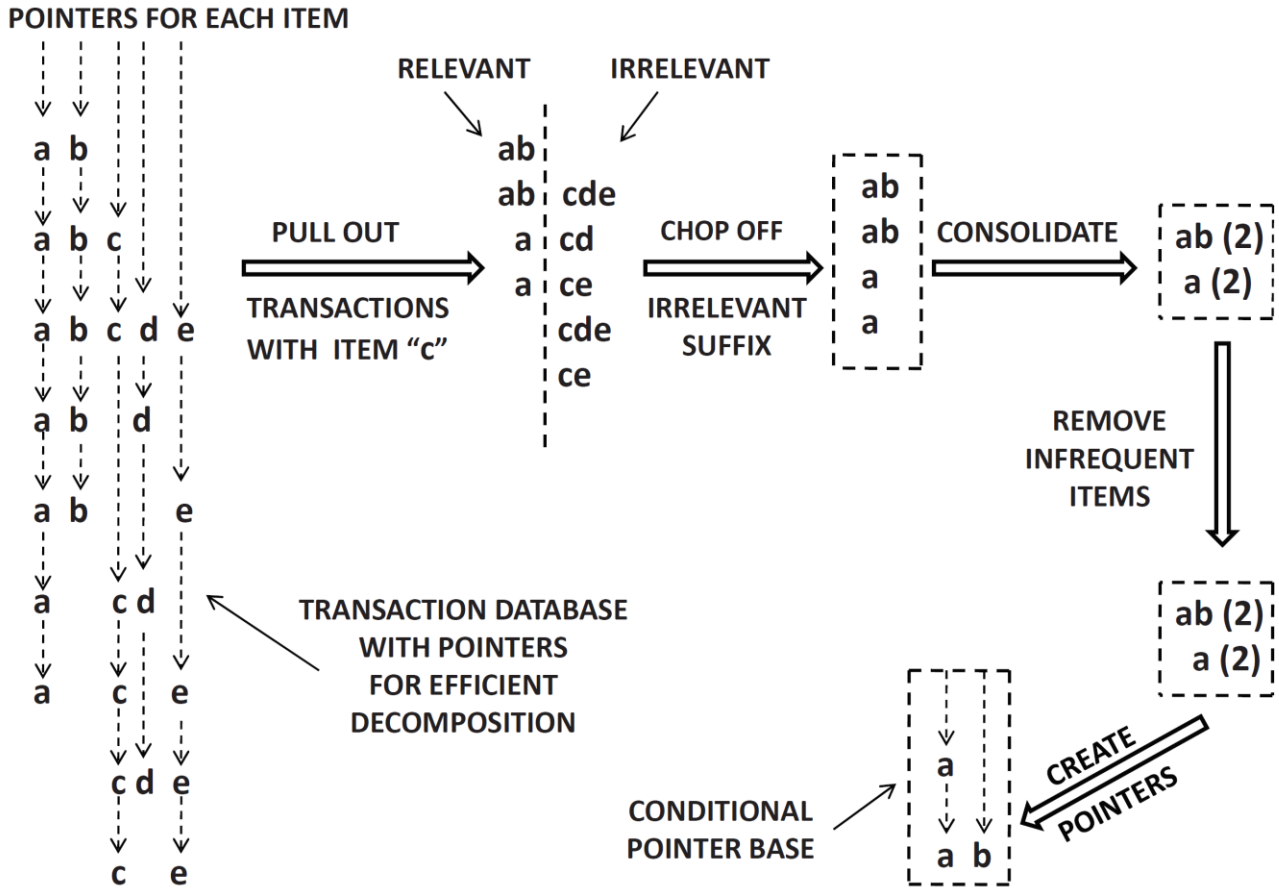
- **Suppose a (conditional) FP-tree T has a shared single prefix-path P**
- **Mining can be decomposed into two parts**
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



Recursive pattern growth with pointers and FP-tree



Recursive pattern growth with pointers and no FP-Tree



FP-growth algorithm

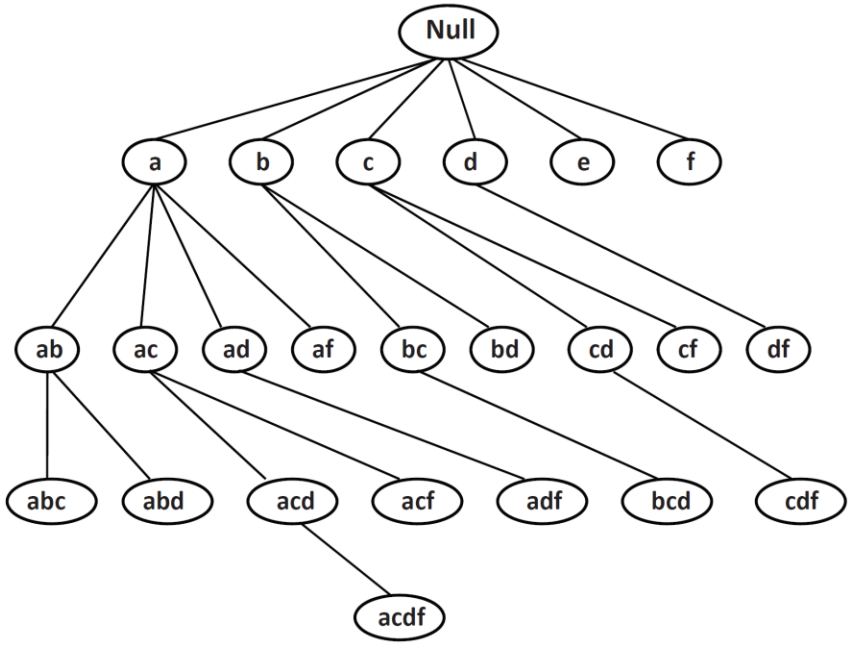
Algorithm *FP-growth*(FP-Tree of frequent items: $\mathcal{FPT}$, Minimum Support: *minsup*, Current Suffix: P)

begin
  if $\mathcal{FPT}$ is a single path
    then determine all combinations C of nodes on the path, and report $C \cup P$ as frequent;
  else (Case when $\mathcal{FPT}$ is not a single path)
  for each item i in $\mathcal{FPT}$ **do begin**
    report itemset $P_i = \{i\} \cup P$ as frequent;
    Use pointers to extract conditional prefix paths from $\mathcal{FPT}$ containing item i;
    Readjust counts of prefix paths and remove i;
    Remove infrequent items from prefix paths and reconstruct conditional FP-Tree $\mathcal{FPT}_i$;
    if $(\mathcal{FPT}_i \neq \phi)$ **then** *FP-growth*$(\mathcal{FPT}_i, minsup, P_i)$;
  end
end

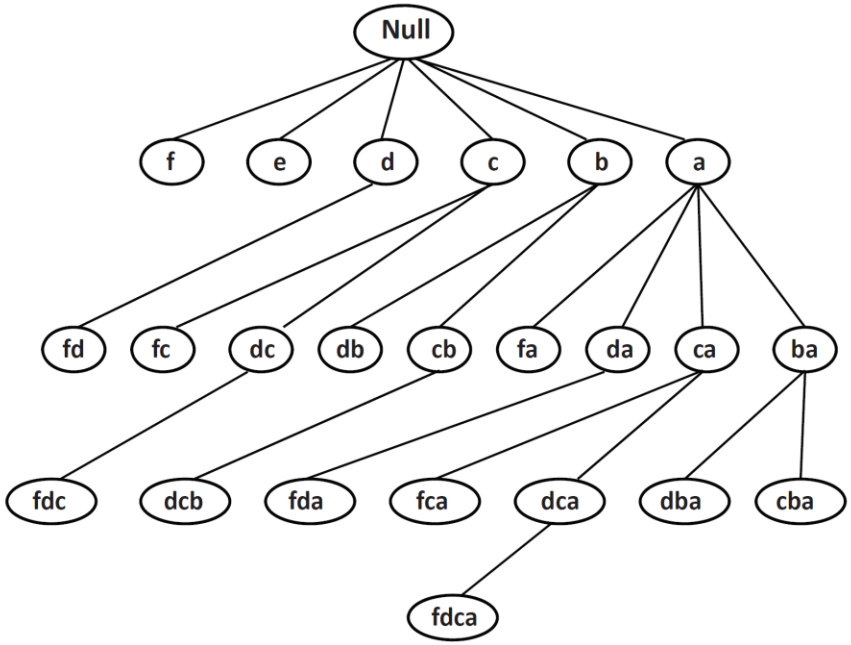
Trade-offs with Different Data Structures

- **The main advantage of an FP-Tree over the pointer-based implementation is one of space compression**
  - FP-Tree requires less space than the pointer-based implementation
- **It is crucial to maintain FP-tree in main memory**
  - because projected databases are repeatedly constructed and scanned during recursive calls
  - otherwise, drastic disk-access costs will be incurred by the potentially exponential number of recursive calls
- **Sizes of the projected databases increase with the original database size**

- **Each call of FP-growth discovers the set of frequent patterns extending a particular suffix of items**
  - c.f., each branch of an enumeration tree explores the itemsets for a particular prefix
  - most enumeration-tree methods order items from the least frequent to the most frequent, whereas FP-growth does the reverse



(a) Prefix extensions with ordering of a, b, c, d, e, f (Enumeration Tree Prefixes shown)



(b) *FP-growth* with ordering of f, e, d, c, b, a (Recursion Tree Suffixes shown)

Interesting Patterns

- **Raw frequencies of itemsets do not always correspond to the most interesting patterns**
- **Limitation of the traditional support-confidence model**
  - all the transactions contain the item Milk (Milk can be appended to any set of items, without changing its frequency)
  - it does not mean that Milk is truly associated with any set of items
  - association rule X ⇒ {Milk} has 100% confidence
  - it does not mean the items X is discriminatively indicative of Milk

tid	Set of Items	Binary Representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

Contingency table

- **Given a rule X → Y, information needed to compute rule interestingness can be obtained from a contingency table**

Contingency table for X → Y

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	|T|

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, lift, Gini, J-measure, etc.

Drawback of Confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Although confidence is high, rule is misleading

$\Rightarrow$ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

Statistical Independence

- **Population of 1000 students**
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

  - $P(S \wedge B) = 420/1000 = 0.42$
  - $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

  - $P(S \wedge B) = P(S) \times P(B)$ => Statistical independence
  - $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
  - $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

Statistical-based Measures

- **Measures that take into account statistical dependence**

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

$$Interest = \frac{P(X,Y)}{P(X)P(Y)}$$

$$PS = P(X,Y) - P(X)P(Y)$$

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

Example: Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

Drawback of Lift & Interest

	Y	$\bar{Y}$	
X	10	0	10
$\bar{X}$	0	90	90
	10	90	100

$$Lift = \frac{0.1}{(0.1)(0.1)} = 10$$

	Y	$\bar{Y}$	
X	90	0	90
$\bar{X}$	0	10	10
	90	10	100

$$Lift = \frac{0.9}{(0.9)(0.9)} = 1.11$$

Statistical independence:

If P(X,Y)=P(X)P(Y) => Lift = 1

There are lots of measures proposed in the literature

Some measures are good for certain applications, but not for others

What criteria should we use to determine whether a measure is good or bad?

What about Apriori-style support based pruning? How does it affect these measures?

#	Measure	Formula
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\left(P(A,B)\log(\frac{P(B\|A)}{P(B)})+P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),\right.$ $\left.P(A,B)\log(\frac{P(A\|B)}{P(A)})+P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\right)$
9	Gini index (G)	$\max\left(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A})[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]\right.$ $-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B})[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]$ $\left.-P(A)^2-P(\overline{A})^2\right)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A),P(A\|B))$
12	Laplace (L)	$\max\left(\frac{NP(A,B)+1}{NP(A)+2},\frac{NP(A,B)+1}{NP(B)+2}\right)$
13	Conviction (V)	$\max\left(\frac{P(A)P(\overline{B})}{P(A\overline{B})},\frac{P(B)P(\overline{A})}{P(B\overline{A})}\right)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B)-P(A)P(B)$
17	Certainty factor (F)	$\max\left(\frac{P(B\|A)-P(B)}{1-P(B)},\frac{P(A\|B)-P(A)}{1-P(A)}\right)$
18	Added Value (AV)	$\max(P(B\|A)-P(B),P(A\|B)-P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})}\times\frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A)-P(B),P(A\|B)-P(A))$

Properties of A Good Measure

- **Piatetsky-Shapiro:**
  3 properties a good measure M must satisfy:
  - M(A,B) = 0 if A and B are statistically independent
  - M(A,B) increase monotonically with P(A,B) when P(A) and P(B) remain unchanged
  - M(A,B) decreases monotonically with P(A) [or P(B)] when P(A,B) and P(B) [or P(A)] remain unchanged

Comparing Different Measures

10 examples of contingency tables:

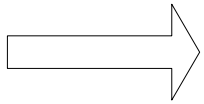
Example	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

Rankings of contingency tables using various measures:

#	ϕ	λ	α	Q	Y	κ	M	J	G	s	c	L	V	I	IS	PS	F	AV	S	ζ	K
E1	1	1	3	3	3	1	2	2	1	3	5	5	4	6	2	2	4	6	1	2	5
E2	2	2	1	1	1	2	1	3	2	2	1	1	1	8	3	5	1	8	2	3	6
E3	3	3	4	4	4	3	3	8	7	1	4	4	6	10	1	8	6	10	3	1	10
E4	4	7	2	2	2	5	4	1	3	6	2	2	2	4	4	1	2	3	4	5	1
E5	5	4	8	8	8	4	7	5	4	7	9	9	9	3	6	3	9	4	5	6	3
E6	6	6	7	7	7	7	6	4	6	9	8	8	7	2	8	6	7	2	7	8	2
E7	7	5	9	9	9	6	8	6	5	4	7	7	8	5	5	4	8	5	6	4	4
E8	8	9	10	10	10	8	10	10	8	4	10	10	10	9	7	7	10	9	8	7	9
E9	9	9	5	5	5	9	9	7	9	8	3	3	3	7	9	9	3	7	9	9	8
E10	10	8	6	6	6	10	5	9	10	10	6	6	5	1	10	10	5	1	10	10	7

Property under Variable Permutation

	B	$\bar{B}$
A	p	q
$\bar{A}$	r	s



	A	$\bar{A}$
B	p	r
$\bar{B}$	q	s

Does M(A,B) = M(B,A)?

Symmetric measures:

- support, lift, collective strength, cosine, Jaccard, etc

Asymmetric measures:

- confidence, conviction, Laplace, J-measure, etc

Example: ϕ-Coefficient

- **ϕ-coefficient is analogous to correlation coefficient for continuous variables**

	Y	$\overline{Y}$	
X	60	10	70
$\overline{X}$	10	20	30
	70	30	100

	Y	$\overline{Y}$	
X	20	10	30
$\overline{X}$	10	60	70
	30	70	100

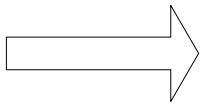
$$\phi = \frac{0.6 - 0.7 \times 0.7}{\sqrt{0.7 \times 0.3 \times 0.7 \times 0.3}} = 0.5238$$

$$\phi = \frac{0.2 - 0.3 \times 0.3}{\sqrt{0.7 \times 0.3 \times 0.7 \times 0.3}} = 0.5238$$

ϕ Coefficient is the same for both tables

Property under Null Addition

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s



	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s + k

Invariant measures:

- support, cosine, Jaccard, etc

Non-invariant measures:

- correlation, Gini, mutual information, odds ratio, etc

Null-Invariant Measures

- **Null-(transaction) invariance is crucial for correlation analysis**
- **Lift and χ^2 are not null-invariant**
- **5 null-invariant measures**

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

Measure	Definition	Range	Null-Invariant
$\chi^2(a,b)$	$\sum_{i,j=0,1} \frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0,\infty]$	No
$Lift(a,b)$	$\frac{P(ab)}{P(a)P(b)}$	$[0,\infty]$	No
$AllConf(a,b)$	$\frac{sup(ab)}{max\{sup(a),sup(b)\}}$	$[0,1]$	Yes
$Coherence(a,b)$	$\frac{sup(ab)}{sup(a)+sup(b)-sup(ab)}$	$[0,1]$	Yes
$Cosine(a,b)$	$\frac{sup(ab)}{\sqrt{sup(a)sup(b)}}$	$[0,1]$	Yes
$Kulc(a,b)$	$\frac{sup(ab)}{2}(\frac{1}{sup(a)}+\frac{1}{sup(b)})$	$[0,1]$	Yes
$MaxConf$(a,b)	$max\{\frac{sup(ab)}{sup(a)},\frac{sup(ab)}{sup(b)}\}$	$[0,1]$	Yes

Table 3. Interestingness measure definitions.

Null-transactions w.r.t. m and c

Kulczynski measure (1927)

Null-invariant

Data set	mc	$\overline{mc}$	$m\overline{c}$	$\overline{mc}$	χ^2	$Lift$	$AllConf$	$Coherence$	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	90557	9.26	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0	1	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	670	8.44	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	24740	25.75	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	8173	9.18	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	965	1.97	0.01	0.01	0.10	0.5	0.99

Table 2. Example data sets.

Subtle: They disagree

Comparison of Interestingness Measures

Table 6: Properties of interestingness measures. Note that none of the measures satisfies all the properties.

Symbol	Measure	Range	P1	P2	P3	O1	O2	O3	O3'	O4
ϕ	ϕ-coefficient	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
λ	Goodman-Kruskal's	$0 \cdots 1$	Yes	No	No	Yes	No	No*	Yes	No
α	odds ratio	$0 \cdots 1 \cdots \infty$	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes	No
Q	Yule's Q	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Y	Yule's Y	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
κ	Cohen's	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	No	No	Yes	No
M	Mutual Information	$0 \cdots 1$	Yes	Yes	Yes	No**	No	No*	Yes	No
J	J-Measure	$0 \cdots 1$	Yes	No	No	No**	No	No	No	No
G	Gini index	$0 \cdots 1$	Yes	No	No	No**	No	No*	Yes	No
s	Support	$0 \cdots 1$	No	Yes	No	Yes	No	No	No	No
c	Confidence	$0 \cdots 1$	No	Yes	No	No**	No	No	No	Yes
L	Laplace	$0 \cdots 1$	No	Yes	No	No**	No	No	No	No
V	Conviction	$0.5 \cdots 1 \cdots \infty$	No	Yes	No	No**	No	No	Yes	No
I	Interest	$0 \cdots 1 \cdots \infty$	Yes*	Yes	Yes	Yes	No	No	No	No
IS	Cosine	$0 \cdots \sqrt{P(A,B)} \cdots 1$	No	Yes	Yes	Yes	No	No	No	Yes
PS	Piatetsky-Shapiro's	$-0.25 \cdots 0 \cdots 0.25$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
F	Certainty factor	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	No**	No	No	Yes	No
AV	Added value	$-0.5 \cdots 0 \cdots 1$	Yes	Yes	Yes	No**	No	No	No	No
S	Collective strength	$0 \cdots 1 \cdots \infty$	No	Yes	Yes	Yes	No	Yes*	Yes	No
ζ	Jaccard	$0 \cdots 1$	No	Yes	Yes	Yes	No	No	No	Yes
K	Klosgen's	$(\frac{2}{\sqrt{3}} - 1)^{1/2}[2 - \sqrt{3} - \frac{1}{\sqrt{3}}] \cdots 0 \cdots \frac{2}{3\sqrt{3}}$	Yes	Yes	Yes	No**	No	No	No	No

where: P1: $O(\mathbf{M}) = 0$ if $det(\mathbf{M}) = 0$, *i.e.*, whenever A and B are statistically independent.
P2: $O(\mathbf{M_2}) > O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [k \ \ -k;\ -k \ \ k]$.
P3: $O(\mathbf{M_2}) < O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [0 \ k;\ 0 \ \ -k]$ or $\mathbf{M_2} = \mathbf{M_1} + [0 \ 0;\ k \ \ -k]$.
O1: Property 1: Symmetry under variable permutation.
O2: Property 2: Row and Column scaling invariance.
O3: Property 3: Antisymmetry under row or column permutation.
O3': Property 4: Inversion invariance.
O4: Property 5: Null invariance.
Yes*: Yes if measure is normalized.
No*: Symmetry under row or column permutation.
No**: No unless the measure is symmetrized by taking $\max(M(A,B), M(B,A))$.

InfoLab

Thank you!

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology