UNIVERSITÄT MANNHEIM

Data Mining I
Association Analysis

Heiko Paulheim

Outline

1. What is Association Analysis?
2. Frequent Itemset Generation
3. Rule Generation
4. Interestingness Measures
5. Handling Continuous and Categorical Attributes

Association Analysis

- First algorithms developed in the early 90s at IBM by Agrawal & Srikant
- Motivation
  - Availability of barcode cash registers

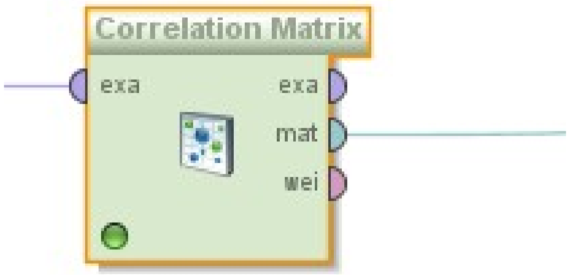
Association Analysis

- initially used for Market Basket Analysis
  - to find how items purchased by customers are related
- later extended to more complex data structures
  - sequential patterns (see Data Mining II)
  - subgraph patterns
- and other application domains
  - life science
  - social science
  - web usage mining

Simple Approaches

- To find out if two items x and y are bought together, we can compute their correlation
- E.g., Pearson's correlation coefficient:

$$\frac{\sum(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum(x_i-\overline{x})^2}\sqrt{\sum(y_i-\overline{y})^2}}$$



- Numerical coding:
  - 1: item was bought
  - 0: item was not bought
- $\overline{x}$: average of x (i.e., how often x was bought)

Correlation Analysis in RapidMiner

Correlation Matrix (Correlation Matrix)

Table View | Pairwise Table | Plot View | Annotations

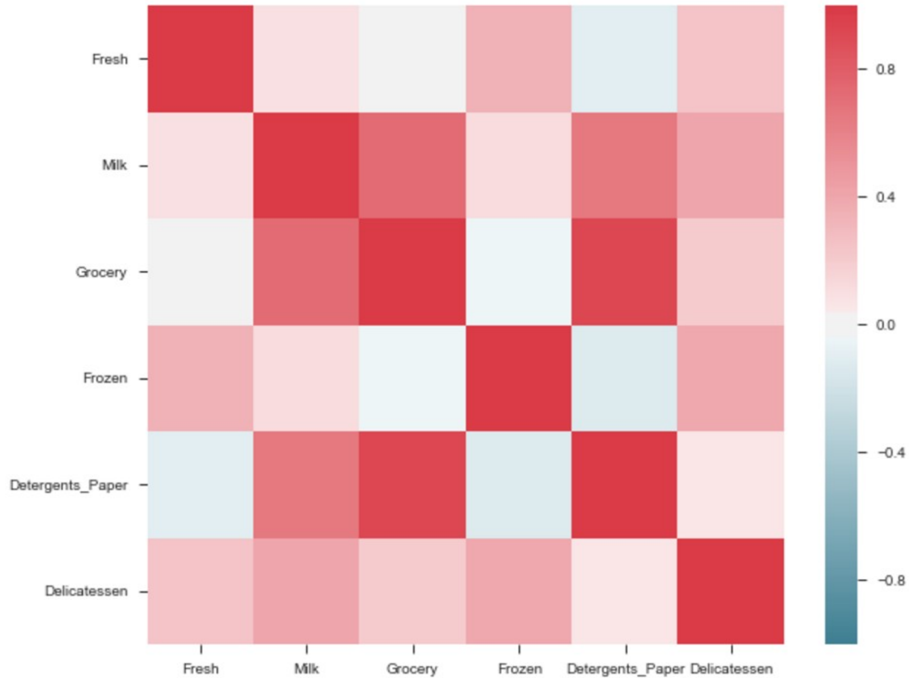
Attributes	ThinkPad X...	Asus EeePC	HP Laserjet...	2 GB DDR3...	8 GB DDR3...	Lenovo Tab...	Netbook-Sc...	HP CE50 T...	LT Laser M...	LT Minimaus
ThinkPad X2	1	-1	0.356	-0.816	0.612	0.583	-0.667	0.356	0.167	-0.408
Asus EeePC	-1	1	-0.356	0.816	-0.612	-0.583	0.667	-0.356	-0.167	0.408
HP Laserjet	0.356	-0.356	1	-0.218	-0.327	0.356	-0.535	1	-0.089	-0.655
2 GB DDR3	-0.816	0.816	-0.218	1	-0.500	-0.816	0.816	-0.218	0	0.200
8 GB DDR3	0.612	-0.612	-0.327	-0.500	1	0.102	-0.408	-0.327	0.102	0
Lenovo Tabl	0.583	-0.583	0.356	-0.816	0.102	1	-0.667	0.356	-0.250	0
Netbook-Sch	-0.667	0.667	-0.535	0.816	-0.408	-0.667	1	-0.535	0.167	0.408
HP CE50 To	0.356	-0.356	1	-0.218	-0.327	0.356	-0.535	1	-0.089	-0.655
LT Laser Ma	0.167	-0.167	-0.089	0	0.102	-0.250	0.167	-0.089	1	-0.408
LT Minimaus	-0.408	0.408	-0.655	0.200	0	0	0.408	-0.655	-0.408	1

Correlation Analysis in Python

- e.g., using Pandas:

```
import seaborn as sns

corr = dataframe.corr()
sns.heatmap(corr)
```



Association Analysis

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

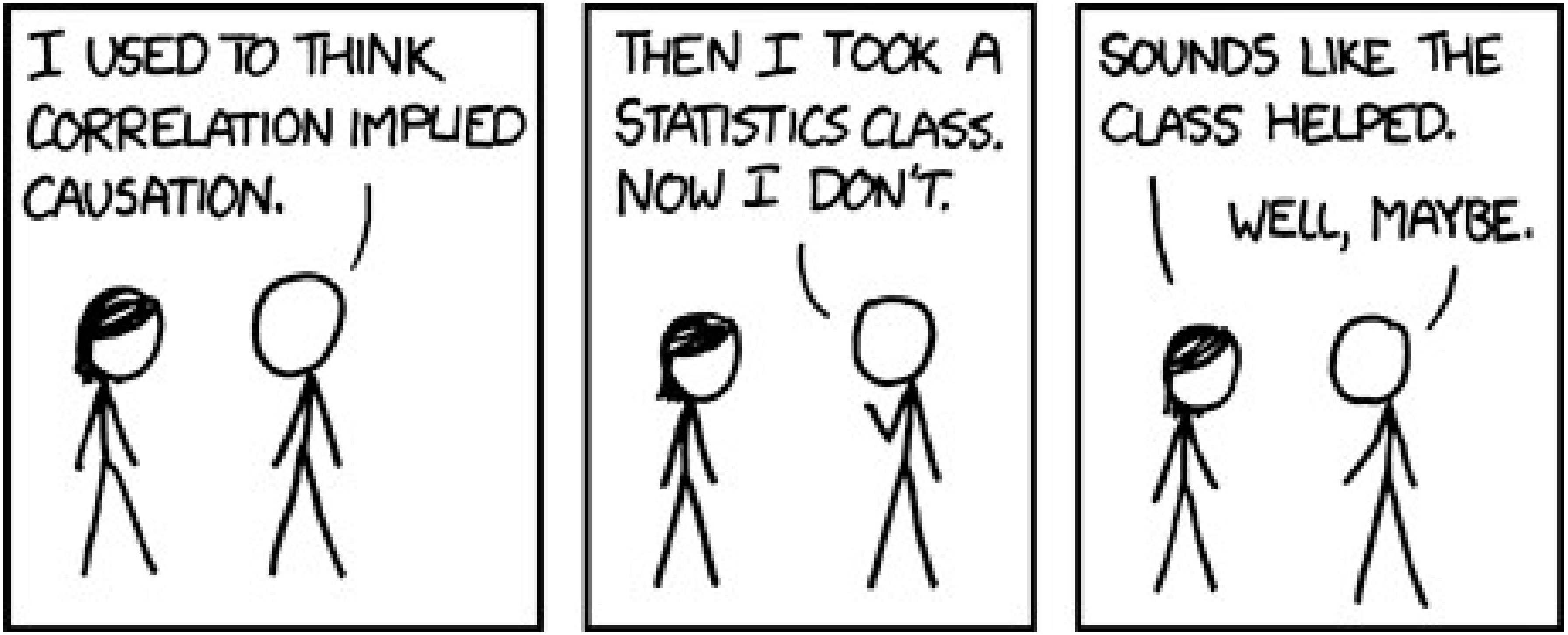
TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Examples of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

→ denotes co-occurence, not causality!

Correlation vs. Causality



http://xkcd.com/552/

Definition: Frequent Itemset

- Itemset
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- Support (s)
  - Frequency of occurrence of an itemset
    - e.g. s({Milk, Bread, Diaper}) = 2/5

- Frequent Itemset
  - An itemset w/ support ≥ a minimum support threshold (minsup)

Definition: Association Rule

- Association Rule
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
- Interpretation: when X occurs,
  Y occurs with a certain probability

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- More formally, it's a *conditional probability*
  - P(Y|X) – given X, what is the probability of Y?
- Known as *confidence* (c)
  - e.g., for {Bread, Milk} → {Diaper}, the probability is 2/3

Definition: Evaluation Metrics

- Given the rule {Milk, Diaper} → {Beer}

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

- Support:
  - Fraction of total transactions which contain both X and Y

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

- Confidence:
  - Fraction of transactions containing X which also contain Y

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ minsup threshold
  - confidence ≥ minconf threshold
- minsup and minconf are provided by the user

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Remove rules that fail the minsup and minconf thresholds

  → *Computationally prohibitive due to large number of candidates!*

Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Examples of Rules:

- {Milk, Diaper} → {Beer} (s=0.4, c=0.67
- {Milk, Beer} → {Diaper} (s=0.4, c=1.0)
- {Diaper, Beer} → {Milk} (s=0.4, c=0.67)
- {Beer} → {Milk, Diaper} (s=0.4, c=0.67)
- {Diaper}→r {Milk, Beer} (s=0.4, c=0.5)
- {Milk} → {Diaper, Beer} (s=0.4, c=0.5)

Observations

- All the above rules are partitions of the same itemset, i.e. {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support
  - but can have different confidence

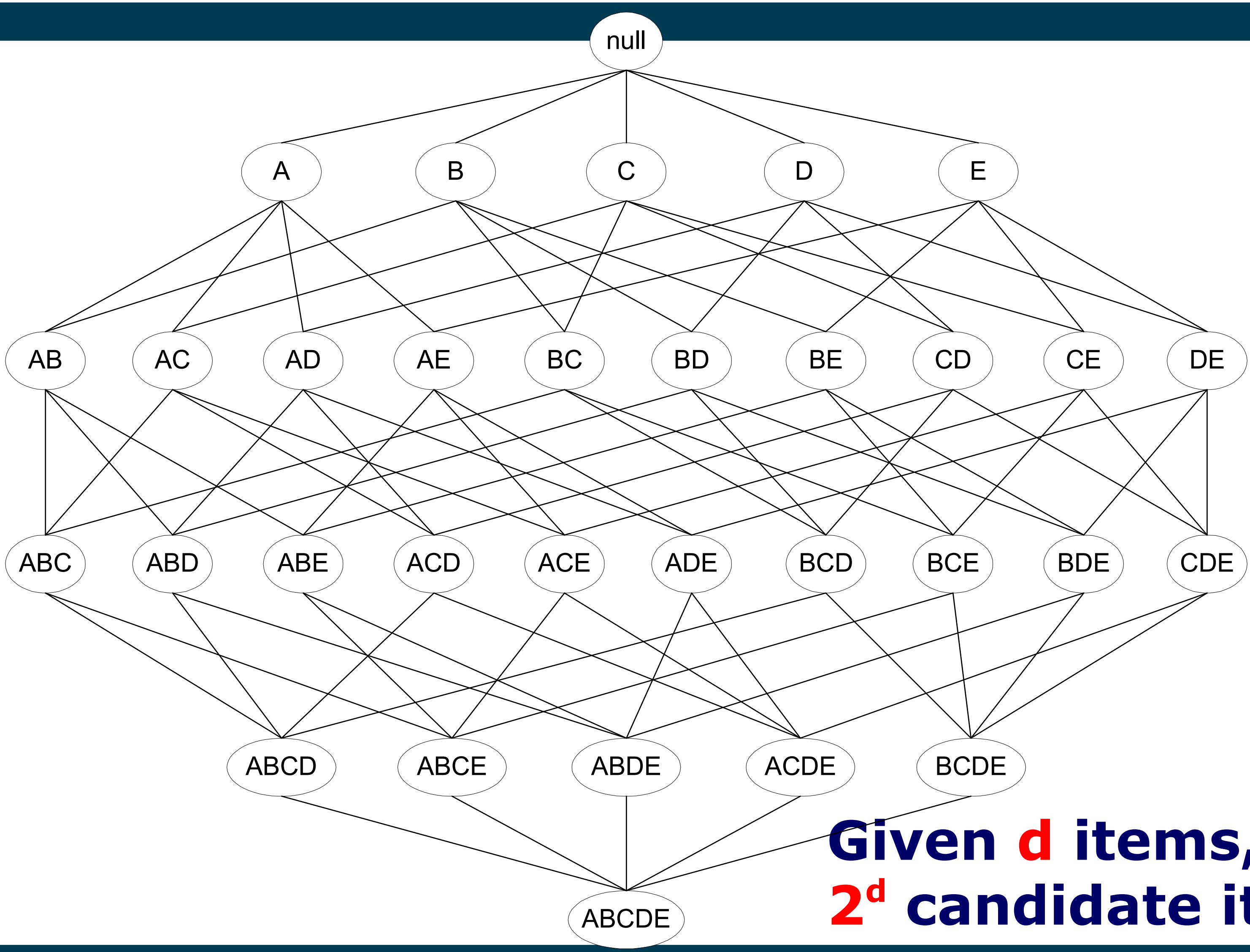
$$s(X \rightarrow Y) := \frac{|X \cup Y|}{|T|}$$

→ we may decouple the support and confidence requirements

Apriori Algorithm: Basic Idea

- Two-step approach
- First: Frequent Itemset Generation
  - Generate all itemsets whose support ≥ minsup
- Second: Rule Generation
  - Generate high confidence rules from each frequent itemset
  - where each rule is a binary partitioning of a frequent itemset
- However: Frequent itemset generation is still computationally expensive....

Frequent Itemset Generation



Given d items, there are 2^d candidate itemsets!

Brute-force Approach

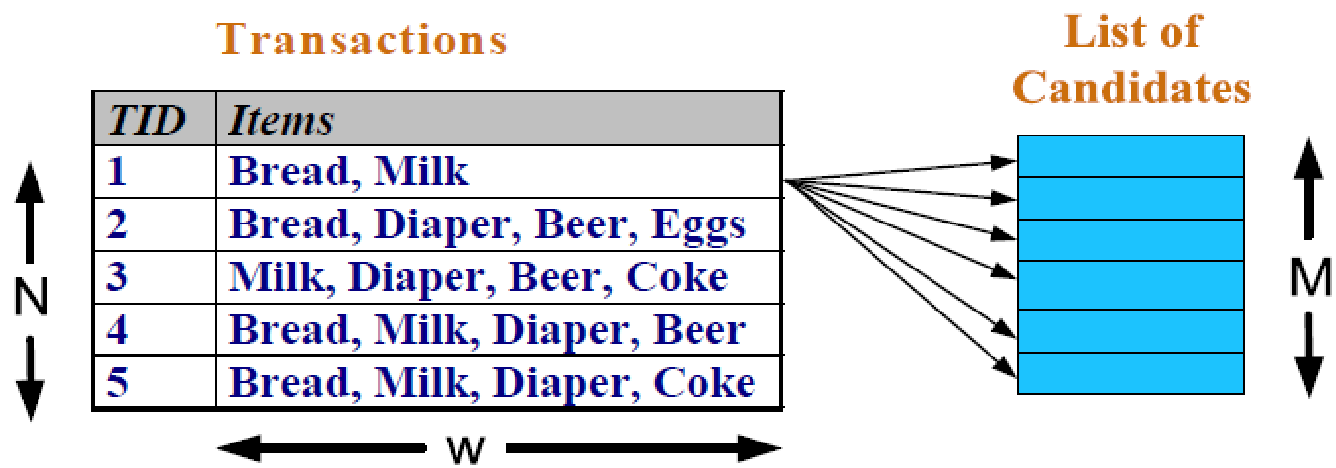
- Example:
  - Amazon sells 120 million products (Amazon.com, as of April 2019)
- That is $2^{120000000}$ possible itemsets
- As a number:
  - $3.017... \times 10^{36,123,599}$
  - That is: a number with 36 million digits!
  - Comparison: the largest supercomputer has a capacity of 40 Petabytes (=$3.2x10^{17}$ bits)

- However:
  - most itemsets will not be important at all
  - e.g., a book on Chinese calligraphy and an iPhone cover bought together
  - thus, smarter algorithms should be possible

https://www.scrapehero.com/number-of-products-on-amazon-april-2019/

Brute-force Approach

- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database
- Match each transaction against every candidate



TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- Complexity ~ O(NMw) → Expensive since $M = 2^d$
- A smarter algorithm is required

Anti-Monotonicity of Support

- What happens when an itemset gets larger?

- s({Bread}) = 0.8
  - s({Bread,Milk}) = 0.6
  - s({Bread,Milk,Diaper}) = 0.4

- s({Milk}) = 0.8
  - s({Milk,Diaper}) = 0.6
  - s({Milk,Diaper,Beer}) = 0.4

- There is a pattern here!

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Anti-Monotonicity of Support

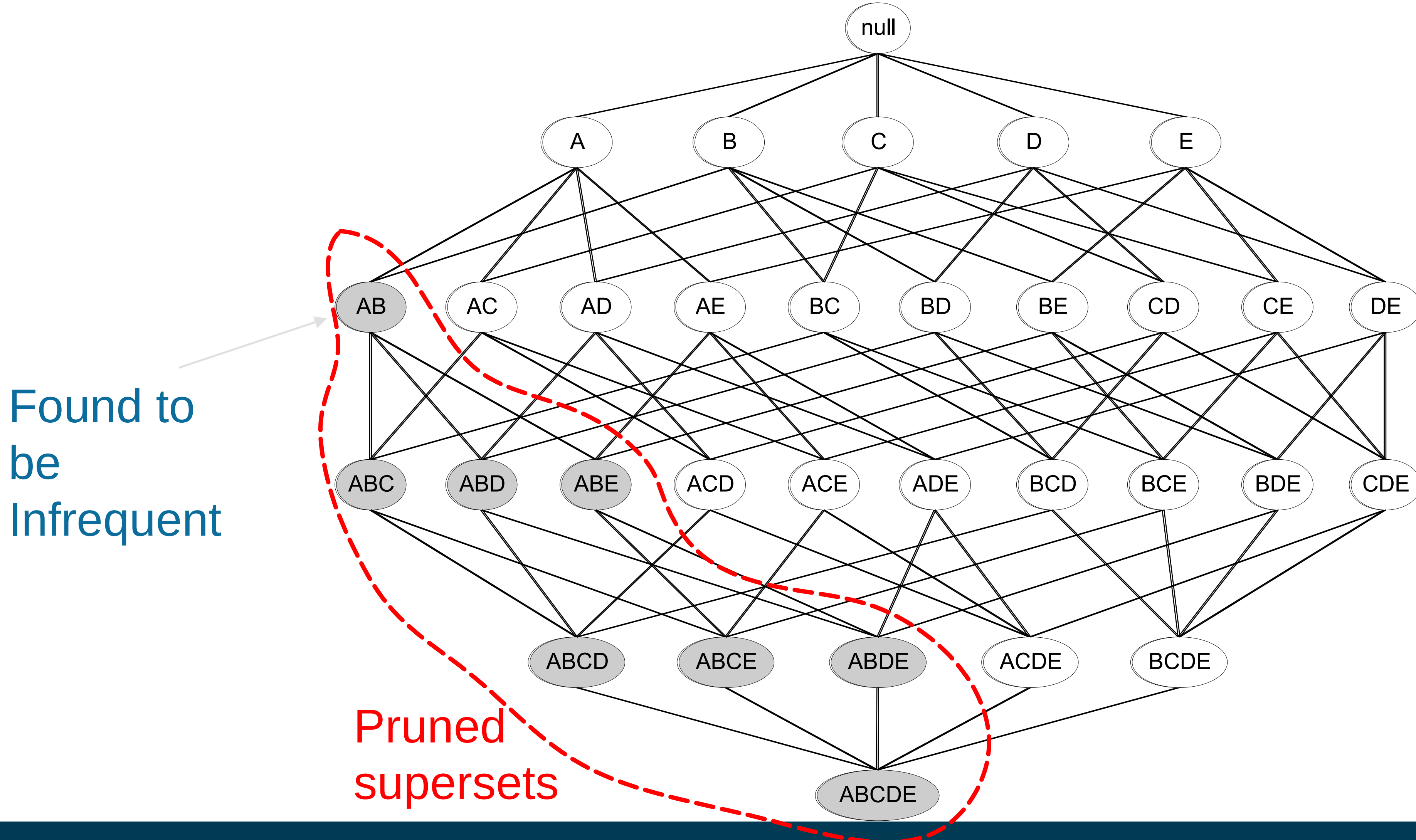
- There is a pattern here!
  - It is called *anti-monitonicity* of support
- If X is a subset of Y
  - s(Y) is at most as large as s(X)

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Consequence for frequent itemset search (aka Apriori principle):
  - If Y is frequent, X also has to be frequent
  - i.e.: all subsets of frequent itemsets are frequent

Illustrating the Apriori Principle



The Apriori Algorithm

1. Start at k=1
2. Generate frequent itemsets of length k=1
3. Repeat until no new frequent itemsets are identified
    1. Generate length (k+1) candidate itemsets from length k frequent itemsets; increase k
    2. Prune candidate itemsets that cannot be frequent because they contain subsets of length k that are infrequent (Apriori Principle)
    3. Count the support of each remaining candidate by scanning the DB
    4. Eliminate candidates that are infrequent, leaving only those that are frequent

Illustrating the Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Minimum Support = 3

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

No need to generate candidates involving Coke or Eggs.

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	**3**

No need to generate candidate {Milk, Diaper, Beer}

From Frequent Itemsets to Rules

- Given a frequent itemset F, find all non-empty subsets f $\subseteq$ F such that f $\rightarrow$ F \ f satisfies the minimum confidence requirement

- Example Frequent Itemset:
  - F= {Milk,Diaper,Beer}

- Example Rule:
  - f = {Milk,Diaper}
  - {Milk,Diaper} $\rightarrow$ {Beer}

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Challenge: Combinatorial Explosion

- Given a 4-itemset {A,B,C,D}, we can generate

  {A} → {B,C,D}, {B} → {A,C,D}, {C} → {A,B,D}, {D} → {A,B,C},

  {A,B} → {C,D}, {A,C} → {B,D}, {A,D} → {B,C},

  {B,C} → {A,D}, {B,D} → {A,C}, {C,D} → {A,B},

  {A,B,C} → {D}, {A,B,D} → {C}, {A,C,D} → {B}, {B,C,D} → {A}
- i.e., a total of 14 rules for just one itemset!

- General number for a k-itemset: 2^k-2
  - it's not 2^k since we ignore Ø → {…} and {…} → Ø

Challenge: Combinatorial Explosion

- Wanted: another pruning trick like Apriori principle

- However

  {Milk,Diaper} → {Beer} c=0.67

  {Milk} → {Beer} c=0.5

  {Diaper} → {Beer} c=0.8

- It's obviously not as straight forward

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Challenge: Combinatorial Explosion

- Wanted: another pruning trick like Apriori principle
- Let's look at it another way
  - {Milk,Diaper,Beer} → Ø c=1.0
    - {Milk,Diaper} → {Beer} c=0.67
      - {Milk} → {Diaper,Beer} c=0.5
      - {Diaper} → {Milk,Beer} c=0.5
    - {Milk,Beer} → {Diaper} c=1.0
      - {Milk} → {Diaper,Beer} c=0.5
      - {Beer} → {Milk,Diaper} c=0.67

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Observation:** moving elements in the rule from left to right never increases confidence!

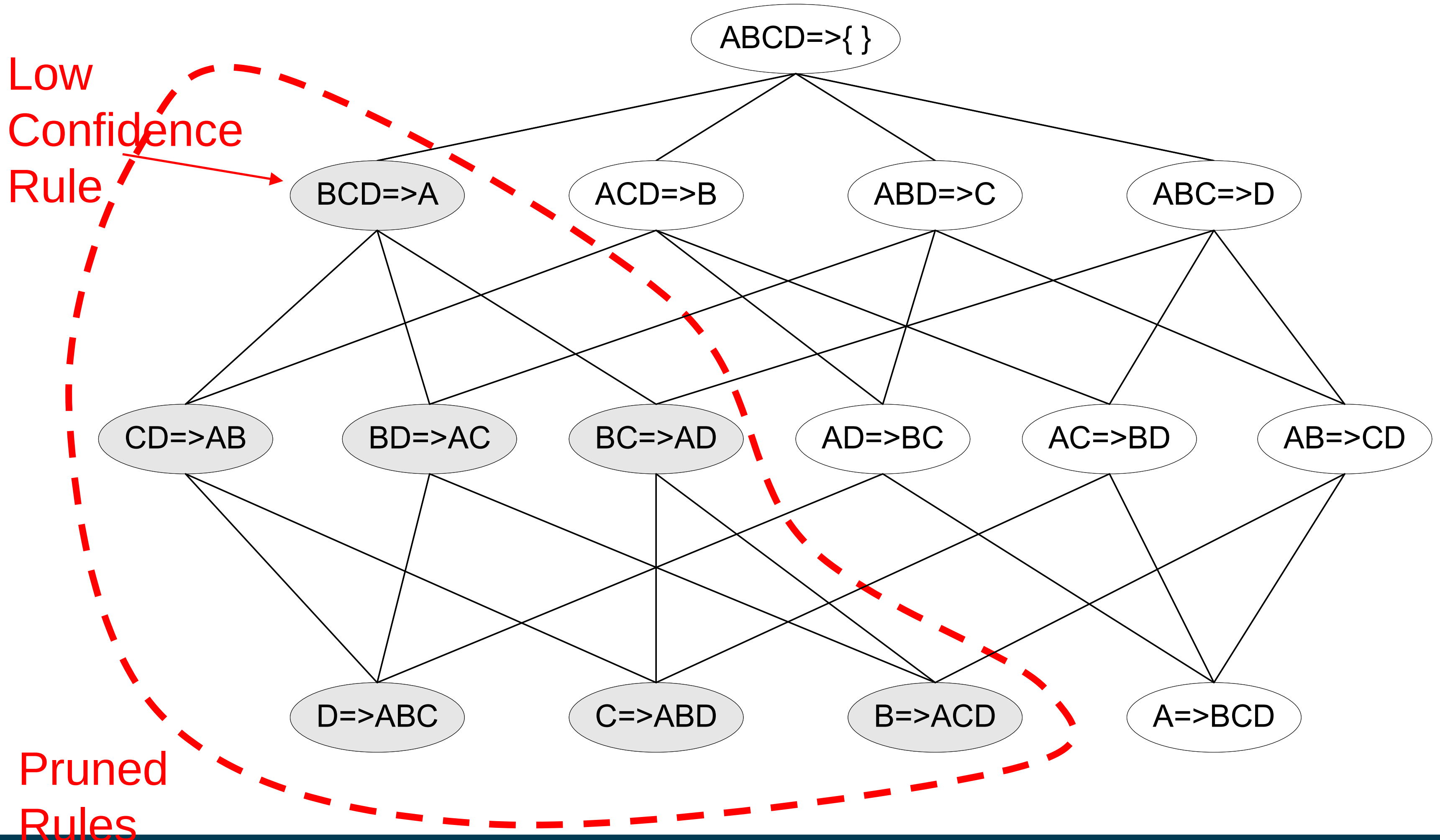
Rule Generation

- Confidence is anti-monotone w.r.t. number of items on the RHS of the rule
  - i.e., “moving elements from left to right” cannot increase confidence
  - reason:

$$c(AB \rightarrow C) := \frac{s(ABC)}{s(AB)} \qquad c(A \rightarrow BC) := \frac{s(ABC)}{s(A)}$$

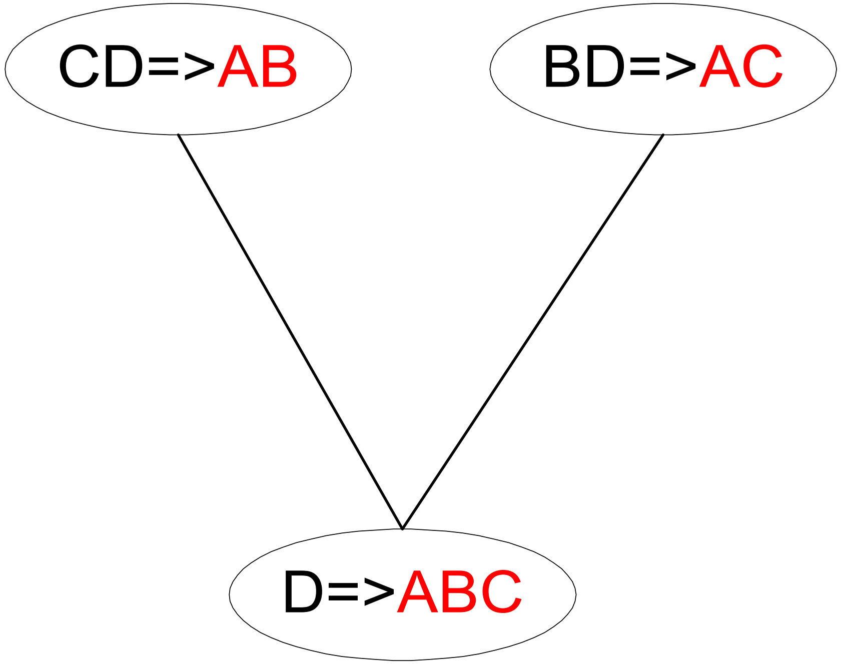
  - Due to anti-monotone property of support, we know
    - S(AB) ≤ S(A)
  - Hence
    - c(AB → C) ≥ C(A → BC)

Rule Generation for Apriori Algorithm



Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
- join(CD=>AB, BD=>AC)
  - would produce the candidate rule D => ABC
- Prune rule D=>ABC
  - if its subset AD=>BC does not have high confidence



- All the required information for confidence computation has already been recorded during itemset generation.
  → No need to see the data anymore!

$$c(X \rightarrow Y) := \frac{s(X \cup Y)}{s(X)}$$

Complexity of Apriori Algorithm

- Expensive part is scanning the database
  - i.e., when counting the support of frequent itemsets
- The database is scanned once per pass of frequent itemset generation
  - one pass to count frequencies of 1-itemsets
  - one pass to count frequencies of 2-itemsets
  - etc.
- i.e., for frequent itemsets of size $\leq k$,
  - k passes over the database are required

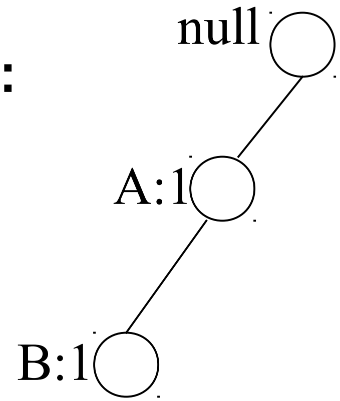
FP-growth Algorithm

- An alternative method for finding frequent itemsets
  - usually faster than Apriori
  - requires at most two passes over the database
- Use a compressed representation of the database using an FP-tree
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

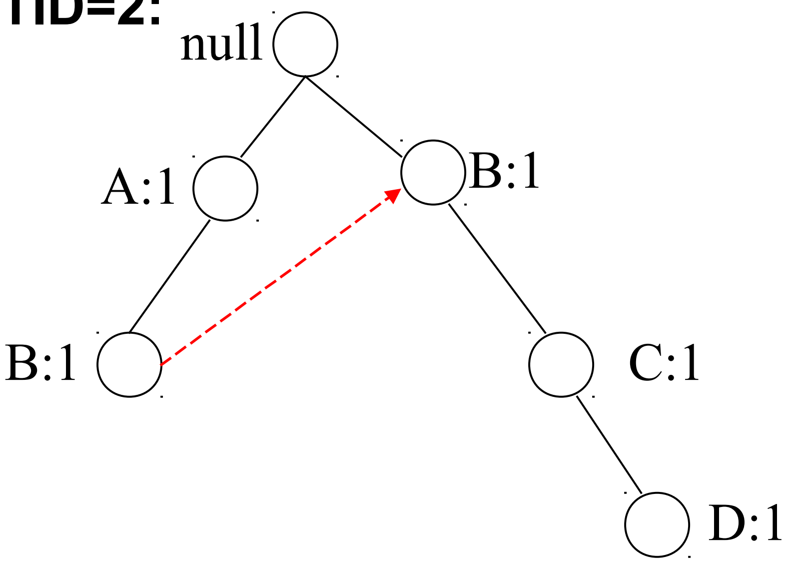
FP-Tree Construction

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



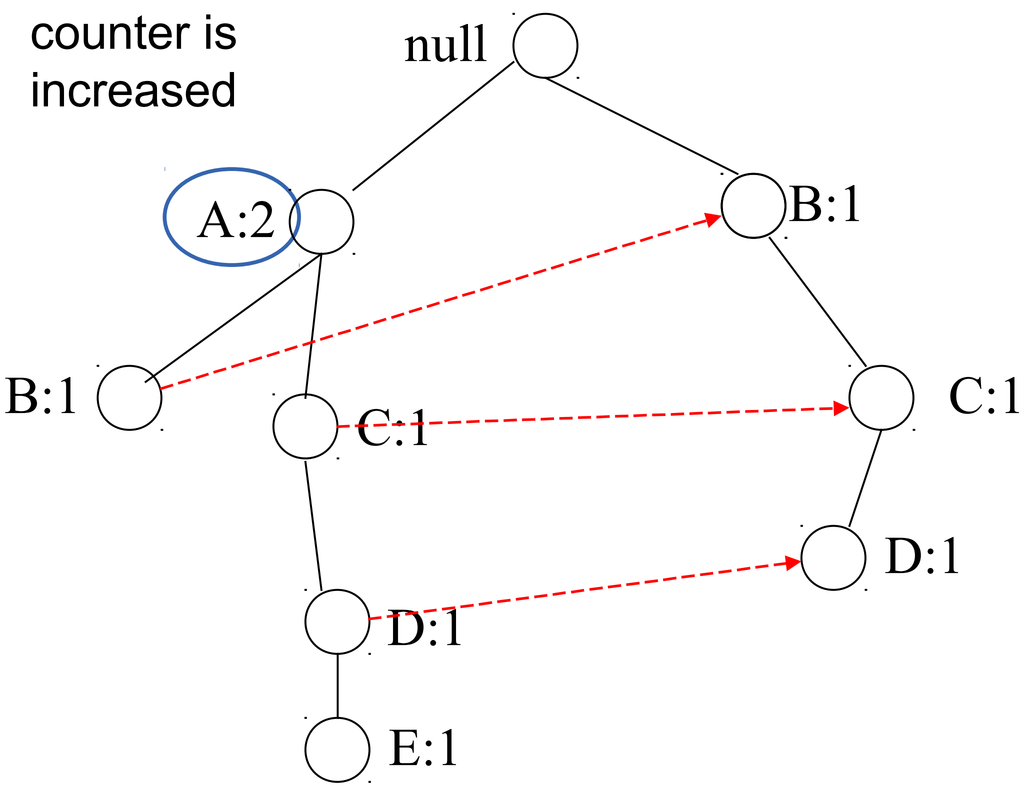
After reading TID=2:



FP-Tree Construction

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

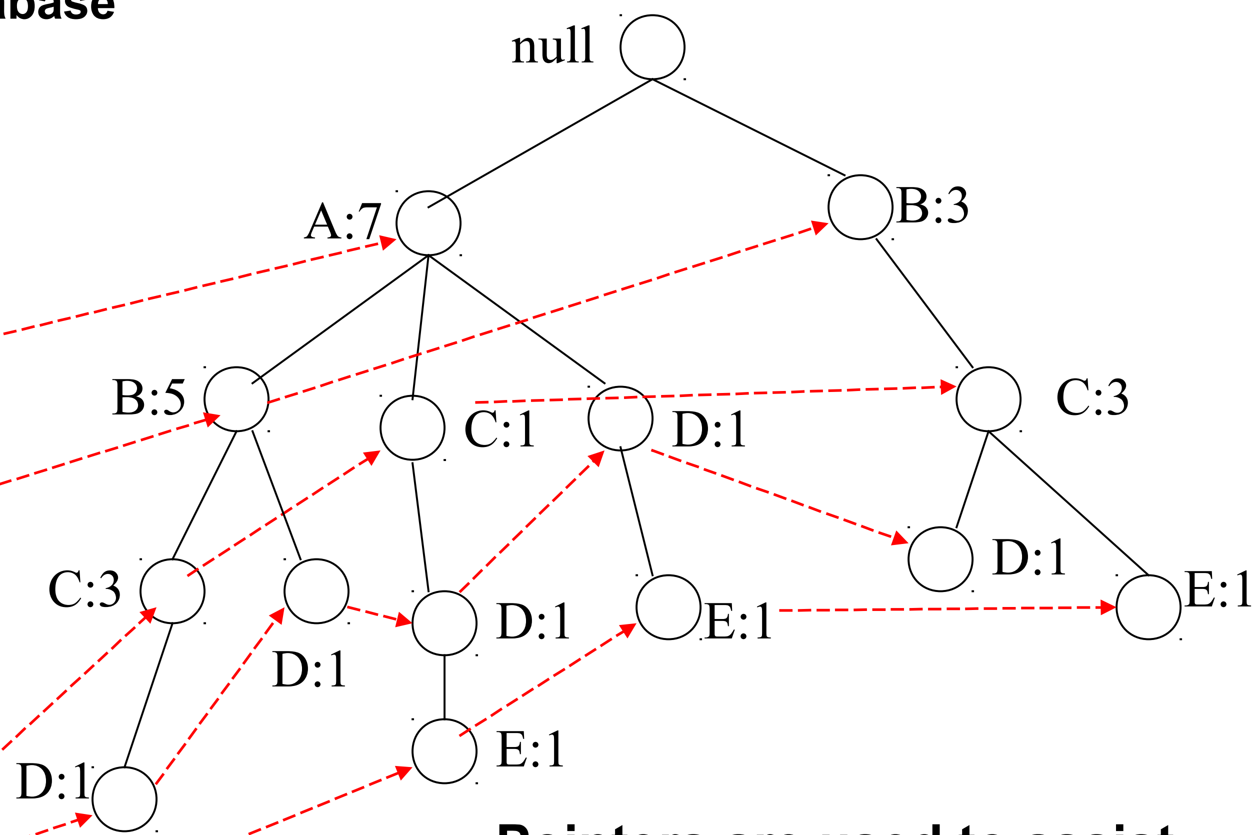
After reading TID=3:



FP-Tree Construction

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database



Header table

Item	Pointer
A	
B	
C	
D	
E	

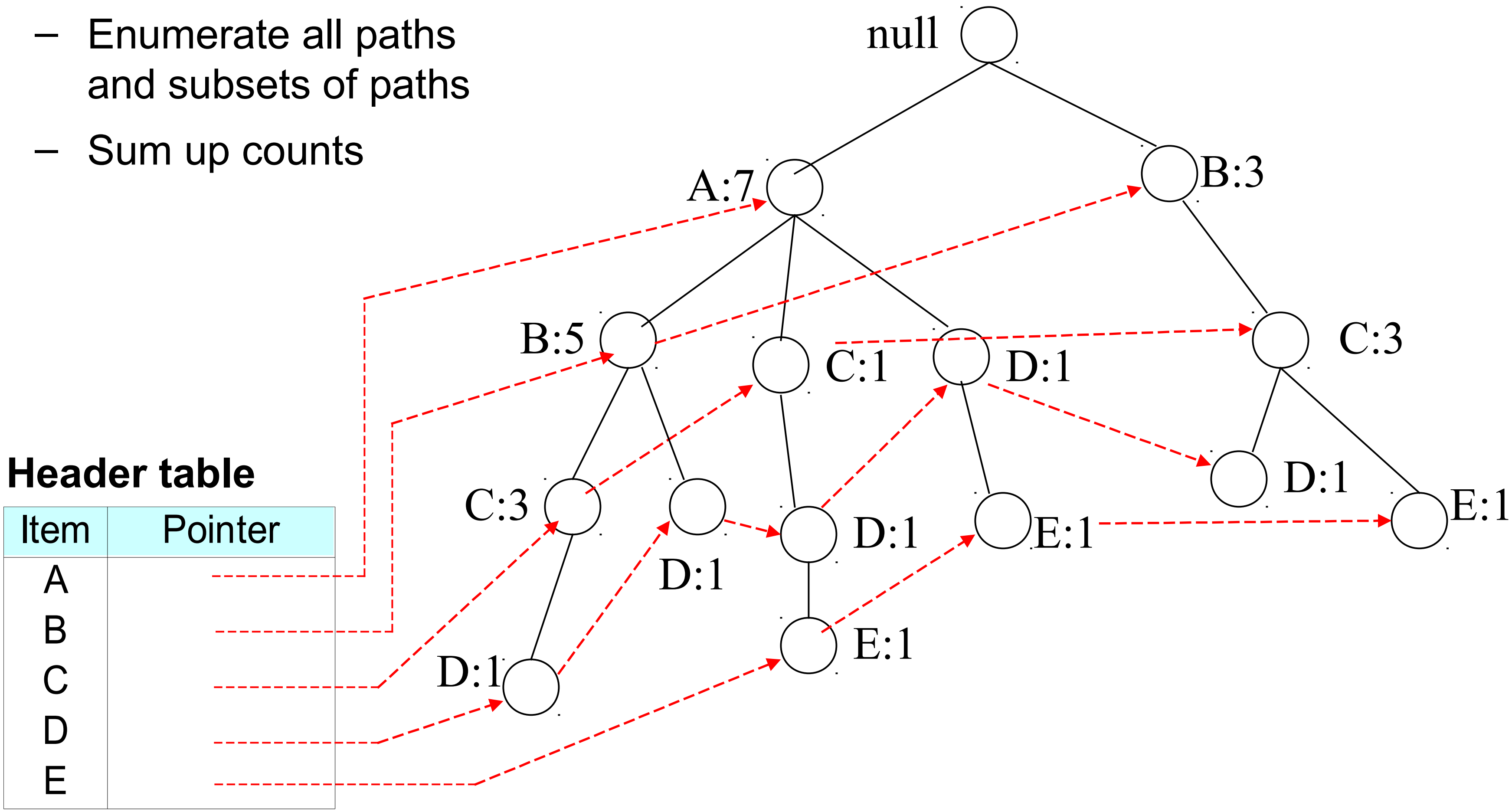
Pointers are used to assist frequent itemset generation

FP-Tree Construction

- Properties of the FP-Tree
  - a very compact representation
  - fits in memory
    - even for larger transaction databases
    - more transactions of the same kind do not increase the tree size
  - can be optimized
    - sorting most frequent items first
    - good compression for many similar transactions
    - up-front pruning of infrequent itemsets

From the FP Tree to Patterns

- Naively:
  - Enumerate all paths and subsets of paths
  - Sum up counts



From the FP Tree to Paths

- Enumeration:
  - A:7, AB:5, AC:3, AD:1, BC:3, CD:1, ABC:3, ABCD:1, …

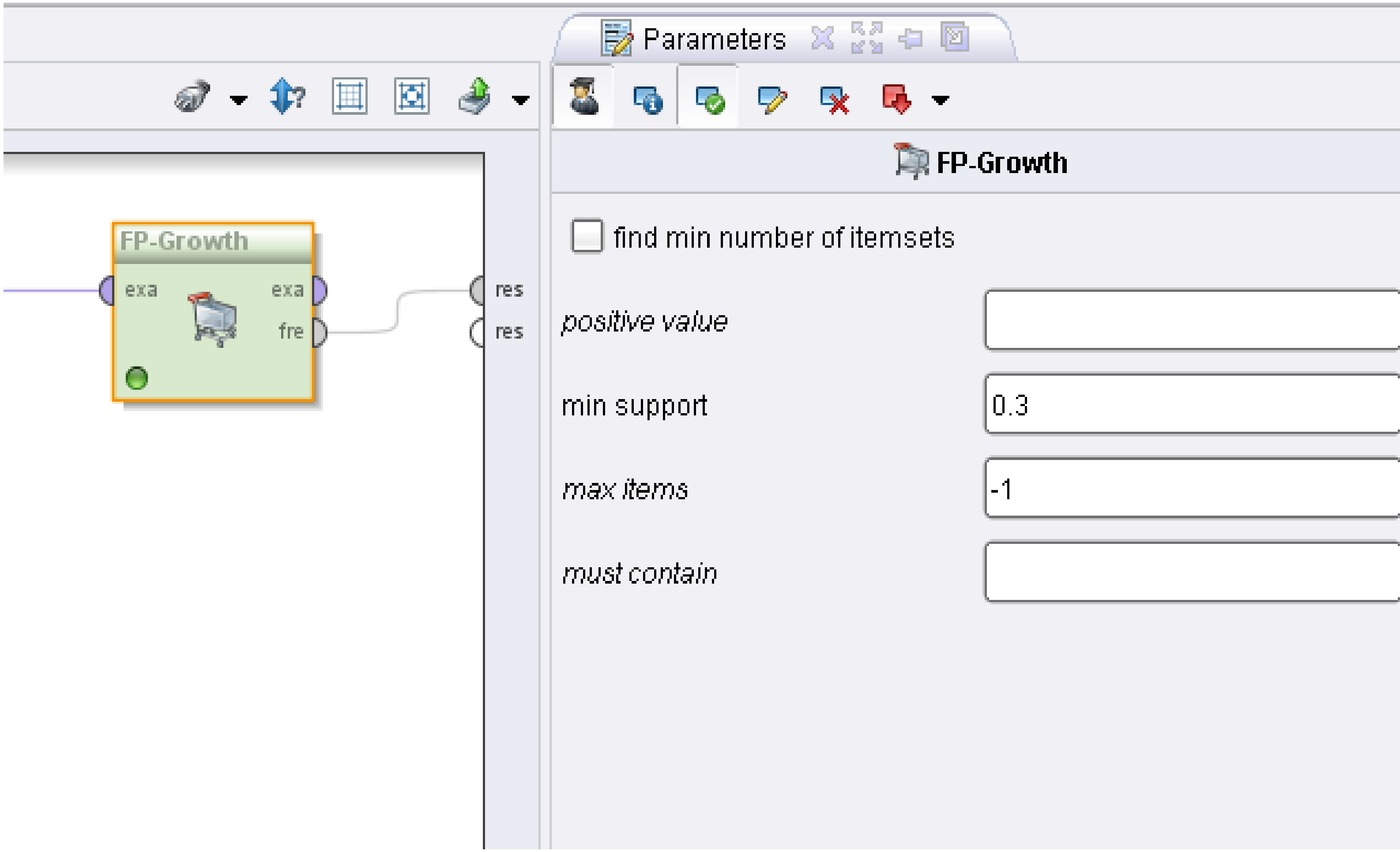
- However, we can do better
  - Single path tree: enumerate all subsets
  - Multi path tree: Build FP-Tree of subtrees recursively
    - For that recursion, we use the links
    - e.g., build FP-Tree for all itemsets ending in E

- Details
  - See literature

FP-Growth (Summary)

- Scans the database only twice:
  - first scan counts all 1-itemsets
    - for ordering by most frequent (more compact tree)
    - and for removing itemsets below minsup
  - second scan for constructing the FP-tree
    - recursive constructions only work on compact representation, not the actual database
- Finding patterns from the tree
  - algorithm recursively decomposes the tree into smaller subtrees
  - details: see books

Frequent Itemset Generation in Rapidminer



Frequent Itemset Generation in Rapidminer

No. of Sets: 22

Total Max. Size: 4

Min. Size: 1

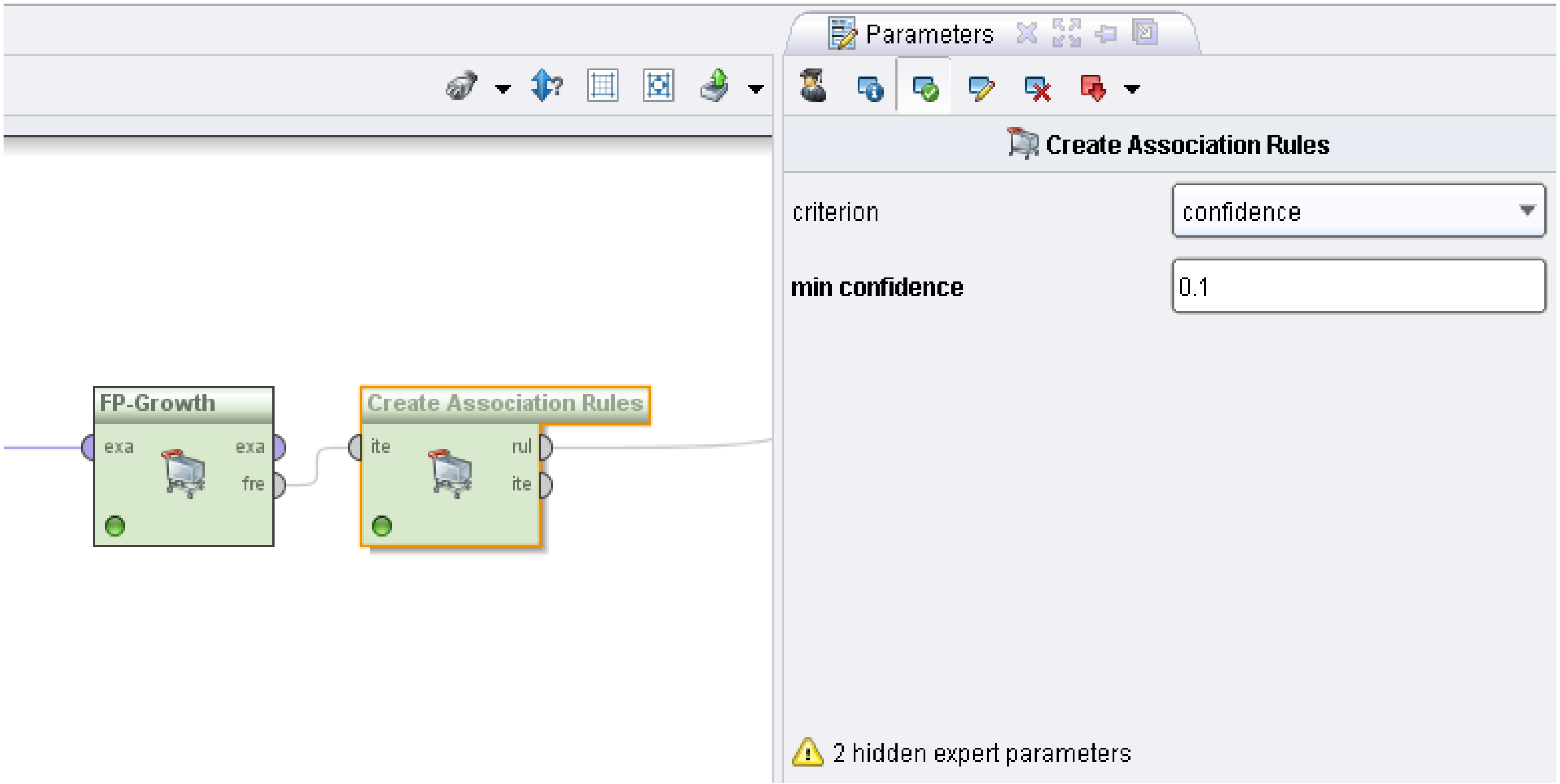
Max. Size: 4

Contains Item:

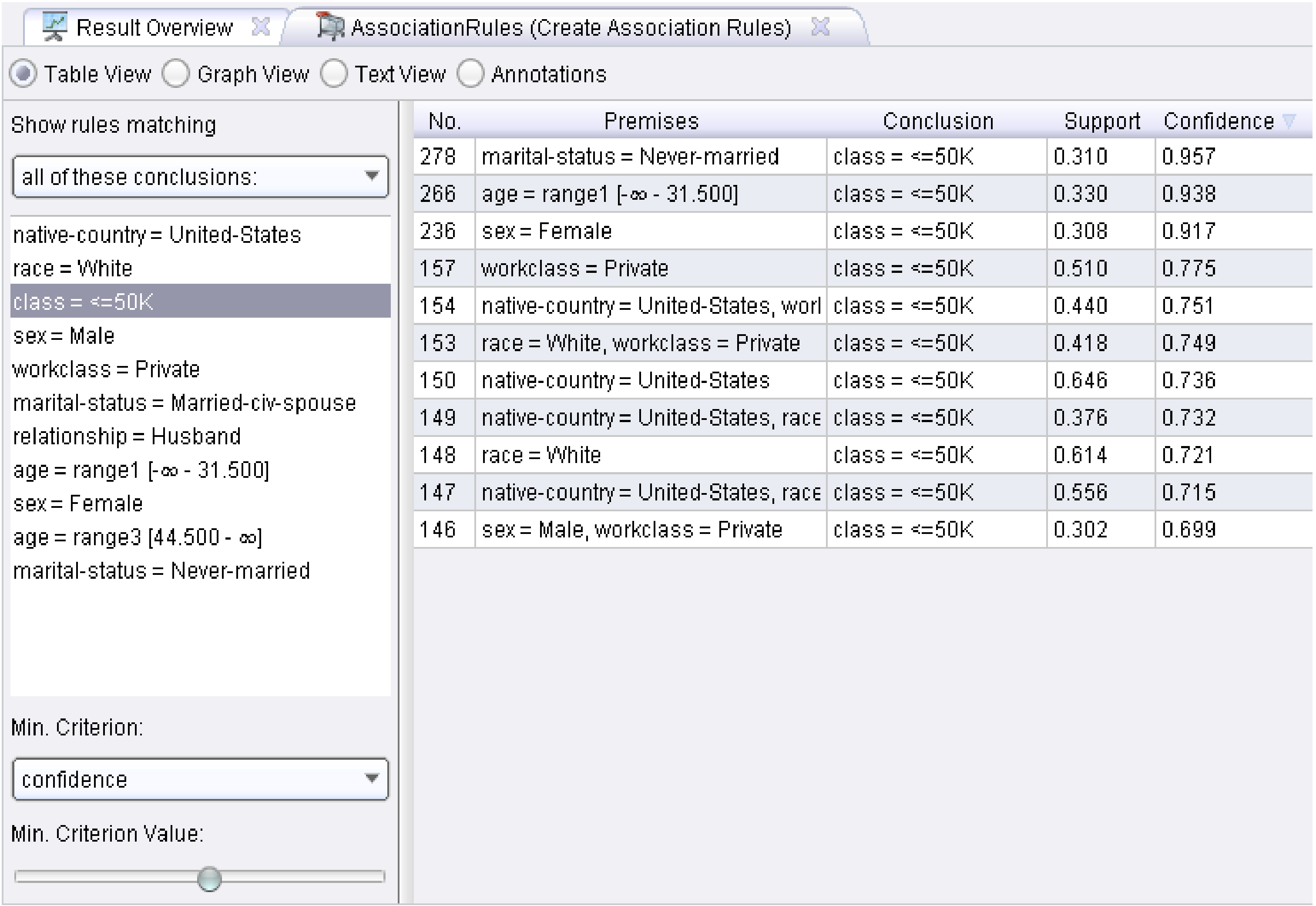
Update View

Size	Support	Item 1	Item 2	Item 3	Item 4
1	0.600	Asus EeePC			
1	0.500	LT Minimaus			
1	0.500	2 GB DDR3			
1	0.400	ThinkPad X2			
1	0.400	Netbook-Sch			
1	0.400	Lenovo Tabl			
1	0.400	LT Laser Ma			
1	0.300	HP Laserjet			
1	0.300	HP CE50 To			
2	0.400	Asus EeePC	LT Minimaus		
2	0.500	Asus EeePC	2 GB DDR3		
2	0.400	Asus EeePC	Netbook-Sch		
2	0.300	LT Minimaus	2 GB DDR3		
2	0.300	LT Minimaus	Netbook-Sch		
2	0.400	2 GB DDR3	Netbook-Sch		
2	0.300	ThinkPad X2	Lenovo Tabl		
2	0.300	HP Laserjet	HP CE50 To		
3	0.300	Asus EeePC	LT Minimaus	2 GB DDR3	
3	0.300	Asus EeePC	LT Minimaus	Netbook-Sch	
3	0.400	Asus EeePC	2 GB DDR3	Netbook-Sch	
3	0.300	LT Minimaus	2 GB DDR3	Netbook-Sch	
4	0.300	Asus EeePC	LT Minimaus	2 GB DDR3	Netbook-Sch

Creating Association Rules in Rapidminer



Exploring Association Rules in Rapidminer



No.	Premises	Conclusion	Support	Confidence
278	marital-status = Never-married	class = <=50K	0.310	0.957
266	age = range1 [-∞ - 31.500]	class = <=50K	0.330	0.938
236	sex = Female	class = <=50K	0.308	0.917
157	workclass = Private	class = <=50K	0.510	0.775
154	native-country = United-States, worl	class = <=50K	0.440	0.751
153	race = White, workclass = Private	class = <=50K	0.418	0.749
150	native-country = United-States	class = <=50K	0.646	0.736
149	native-country = United-States, race	class = <=50K	0.376	0.732
148	race = White	class = <=50K	0.614	0.721
147	native-country = United-States, race	class = <=50K	0.556	0.715
146	sex = Male, workclass = Private	class = <=50K	0.302	0.699

Frequent Itemset Mining in Python

- Various packages exist
  - In the exercise, we'll use the Orange3 package

```
itemsets = dict(fp_growth.frequent_itemsets(X, .2))
rules = association_rules(itemsets, .8)
```

Interestingness Measures

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if {A,B,C} → {D} and {A,B} → {D} have same support & confidence
- Interestingness measures can be used to prune or rank the derived rules
- In the original formulation of association rules, support & confidence are the only interest measures used
- Later, various other measures have been proposed
  - See Tan/Steinbach/Kumar, Chapter 6.7
  - We will have a look at two: Correlation & Lift

Drawback of Confidence

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

Association Rule: Tea → Coffee

- Confidence= s(Tea ∩ Coffee)/s(Tea) = 15/20 = 0.75

Correlation

- We discover a high confidence rule for tea → coffee
  - 75% of all people who drink tea also drink coffee
  - Hypothesis: people who drink tea are likely to drink coffee
    - Implicitly: *more* likely than people not drinking tea
- Cross check:
  - What is the confidence of not(tea) → coffee?
  - Even higher: ~94% of people who **don't** drink tea do drink coffee

- We have two rules here
  - One is learned on all people who drink tea
  - The other is learned on all people who don't trink tea
  - Only together, they cover the whole dataset

Correlation

- Correlation takes into account all data at once
- In our scenario: corr(tea,coffee) = -0.25
  - i.e., the correlation is negative
  - Interpretation: people who drink tea are **less** likely to drink coffee

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

Lift

- We discover a high confidence rule for tea → coffee
  - 75% of all people who drink tea also drink coffee
  - Hypothesis: people who drink tea are likely to drink coffee
    - Implicitly: more likely than **all** people

- Test: Compare the confidence of the two rules
  - Rule: Tea → coffee
  - *Default rule: all* → coffee

- c(tea → coffee) = s(tea ∩ coffee)/s(tea)
- c(*all* → coffee) = s(*all* ∩ coffee)/s(*all*) = s(coffee) / 1

Lift

- Test: Compare the confidence of the two rules
  - Rule: tea → coffee
  - *Default rule: all* → coffee

- We accept a rule iff its confidence is higher than the default rule
  - c(tea → coffee) = s(tea ∩ coffee)/s(tea)
  - c(*all* → coffee) = s(*all* ∩ coffee)/s(*all*) = s(coffee) / 1

c(tea → coffee) > c(all → coffee)

↔ c(tea → coffee) / c(all → coffee) > 1

↔ s(tea ∩ coffee)/ (s(tea) * s(coffee)) > 1

$$Lift(X \rightarrow Y) = \frac{s(X \cap Y)}{s(X) \times S(Y)}$$

Lift

- The *lift* of an association rule X → Y is defined as:

$$Lift(X \rightarrow Y) = \frac{s(X \cap Y)}{s(X) \times S(Y)}$$

- Interpretation:
  - if lift > 1, then X and Y are positively associated
  - if lift < 1, then X are Y are negatively associated
  - if lift = 1, then X and Y are independent.

Example: Lift

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

Association Rule: Tea → Coffee

s(Tea ∩ Coffee) = 0.15

s(Tea) = 0.2, s(Coffee) = 0.9

⇒ Lift = 0.15/(0.2*0.9)= 0.8333 (< 1, therefore is negatively associated)

Combination of Confidence and Lift/Correlation

- So why not try to find rules with high lift/correlation directly?

- By design, lift and correlation are *symmetric*
  - i.e., lift(tea → coffee) = lift(coffee → tea)
- Confidence is *asymmetric*
  - c(coffee → tea) is only 15/90 = 0.167

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

Interestingness Measures

- There are lots of measures proposed in the literature
- Some measures are good for certain applications, but not for others
- Details: see literature (e.g., Tan et al.)

#	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)	$\max\Big(P(A,B)\log(\frac{P(B\mid A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\mid A)}{P(\overline{B})}),$ $P(A,B)\log(\frac{P(A\mid B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\mid B)}{P(\overline{A})})\Big)$
9	Gini index (G)	$\max\Big(P(A)[P(B\mid A)^2+P(\overline{B}\mid A)^2]+P(\overline{A})[P(B\mid\overline{A})^2+P(\overline{B}\mid\overline{A})^2]$ $-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\mid B)^2+P(\overline{A}\mid B)^2]+P(\overline{B})[P(A\mid\overline{B})^2+P(\overline{A}\mid\overline{B})^2]$ $-P(A)^2-P(\overline{A})^2\Big)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\mid A), P(A\mid B))$
12	Laplace (L)	$\max\Big(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\Big)$
13	Conviction (V)	$\max\Big(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\Big)$
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)	$\max\Big(\frac{P(B\mid A)-P(B)}{1-P(B)}, \frac{P(A\mid B)-P(A)}{1-P(A)}\Big)$
18	Added Value (AV)	$\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
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)	$\sqrt{P(A,B)}\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$

Handling Continuous and Categorical Attributes

- How to apply association analysis formulation to other types of variables?

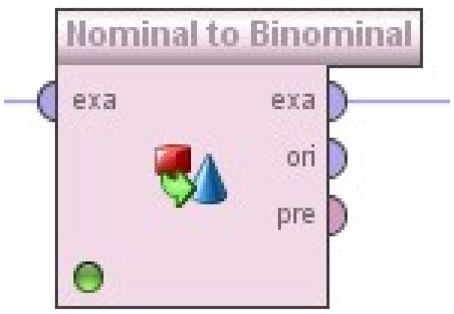
Session Id	Country	Session Length (sec)	Number of Web Pages viewed	Gender	Browser Type	Buy
1	USA	982	8	Male	IE	No
2	China	811	10	Female	Netscape	No
3	USA	2125	45	Female	Mozilla	Yes
4	Germany	596	4	Male	IE	Yes
5	Australia	123	9	Male	Mozilla	No
…	…	…	…	…	…	…

- Example of Association Rule:

  {Number of Pages $\in$ [5,10) $\wedge$ (Browser=Mozilla)} $\rightarrow$ {Buy = No}

Handling Categorical Attributes

- Transform categorical attribute into asymmetric binary variables
- Introduce a new “item” for each distinct attribute-value pair
  - Example: replace Browser Type attribute with
    - Browser Type = Internet Explorer
    - Browser Type = Mozilla



Handling Categorical Attributes

- Introduce a new "item" for each distinct attribute-value pair
  - Example: replace Browser Type attribute with
    - Browser Type = Internet Explorer
    - Browser Type = Mozilla

- This method is also known as *one-hot-encoding*
  - We create n new variables, only one of which is 1 ("hot") at a time

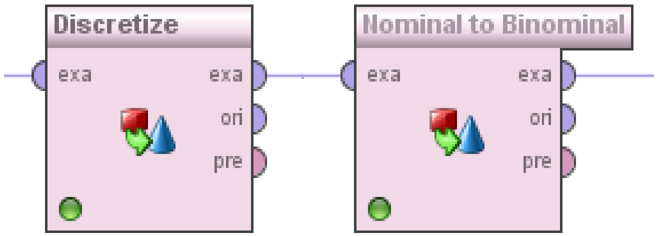
```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit_transform(data)
```

Handling Categorical Attributes

- Potential Issues
  - Many attribute values
    - Many of the attribute values may have very low support
    - Potential solution: Aggregate the low-support attribute values
      - bin for “other”
  - Highly skewed attribute values
    - Example: 95% of the visitors have Buy = No
    - Most of the items will be associated with (Buy=No) item
    - Potential solution: drop the highly frequent items

Handling Continuous Attributes

- Transform continuous attribute into binary variables using discretization
  - Equal-width binning
  - Equal-frequency binning



- Issue: Size of the intervals affects support & confidence

{Refund = No, (51,253 $\leq$ Income $\leq$ 51,254)} $\rightarrow$ {Cheat = No}

{Refund = No, (60K $\leq$ Income $\leq$ 80K)} $\rightarrow$ {Cheat = No}

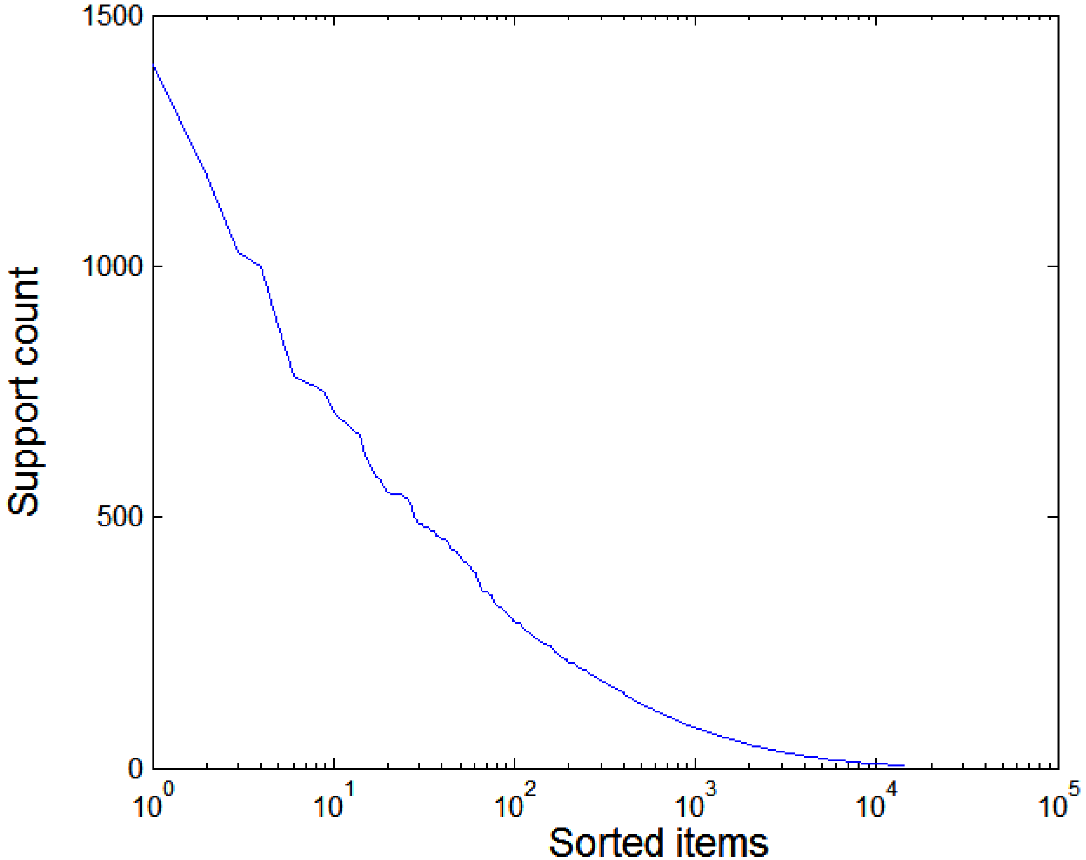
{Refund = No, (0K $\leq$ Income $\leq$ 1B)} $\rightarrow$ {Cheat = No}

  - Too small intervals: not enough support
  - Too large intervals: not enough confidence

Effect of Support Distribution

- Many real data sets have a skewed support distribution

Support distribution of a retail data set



Effect of Support Distribution

- How to set the appropriate *minsup* threshold?
  - If *minsup* is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
  - If *minsup* is set too low, it is computationally expensive and the number of itemsets is very large
- Using a single minimum support threshold may not be effective

Multiple Minimum Support

- How to apply multiple minimum supports?
  - MS(i): minimum support for item i
  - e.g.: MS(Milk)=5%, MS(Coke) = 3%,
    MS(Broccoli)=0.1%, MS(Salmon)=0.5%
  - MS({Milk, Broccoli}) = min (MS(Milk), MS(Broccoli))
    = 0.1%

- Challenge: Support is no longer anti-monotone
  - Suppose: Support(Milk, Coke) = 1.5% and
    Support(Milk, Coke, Broccoli) = 0.5

    → {Milk,Coke} is infrequent but {Milk,Coke,Broccoli} is frequent
  - Requires variations of Apriori algorithm
  - Details: see literature

Association Rules with Temporal Components

- Good example:
  - (Twilight) (New Moon) → (Eclipse)
- Bad example:
  - *mobile phone* → *charger* vs. *charger* → *mobile phone*
  - are indistinguishable by frequent pattern mining
    - both will be used for recommendation
  - customers will select a charger after a mobile phone
    - but not the other way around!
    - however, Amazon does not respect sequences…
- See: Data Mining 2 for *sequential* pattern mining

Wrap-up

- Association Analysis:
  - discovering patterns in data
  - patterns are described by rules
- Apriori & FP-Growth algorithm:
  - Finds rules with minimum support (i.e., number of transactions)
  - and minimum confidence (i.e., strength of the implication)
- You'll play around with it in the upcoming exercise...

What's Next?

- Data Mining 2 (next FSS)
- Machine Learning / Hot Topics in Machine Learning (HWS / FSS), Prof. Gemulla
- Relational Learning (HWS), Dr. Meilicke

- Information Retrieval and Web Search (next FSS), Prof. Glavaš
- Text Analytics (HWS), Prof. Ponzetto & Prof. Glavaš
- Web Mining (FSS), Prof. Ponzetto
- Image Processing (HWS) and
  Higher-Level Computer Vision (FSS), Prof. Keuper
- Network Analysis (HWS), Dr. Karnstedt-Hulpus

Questions?