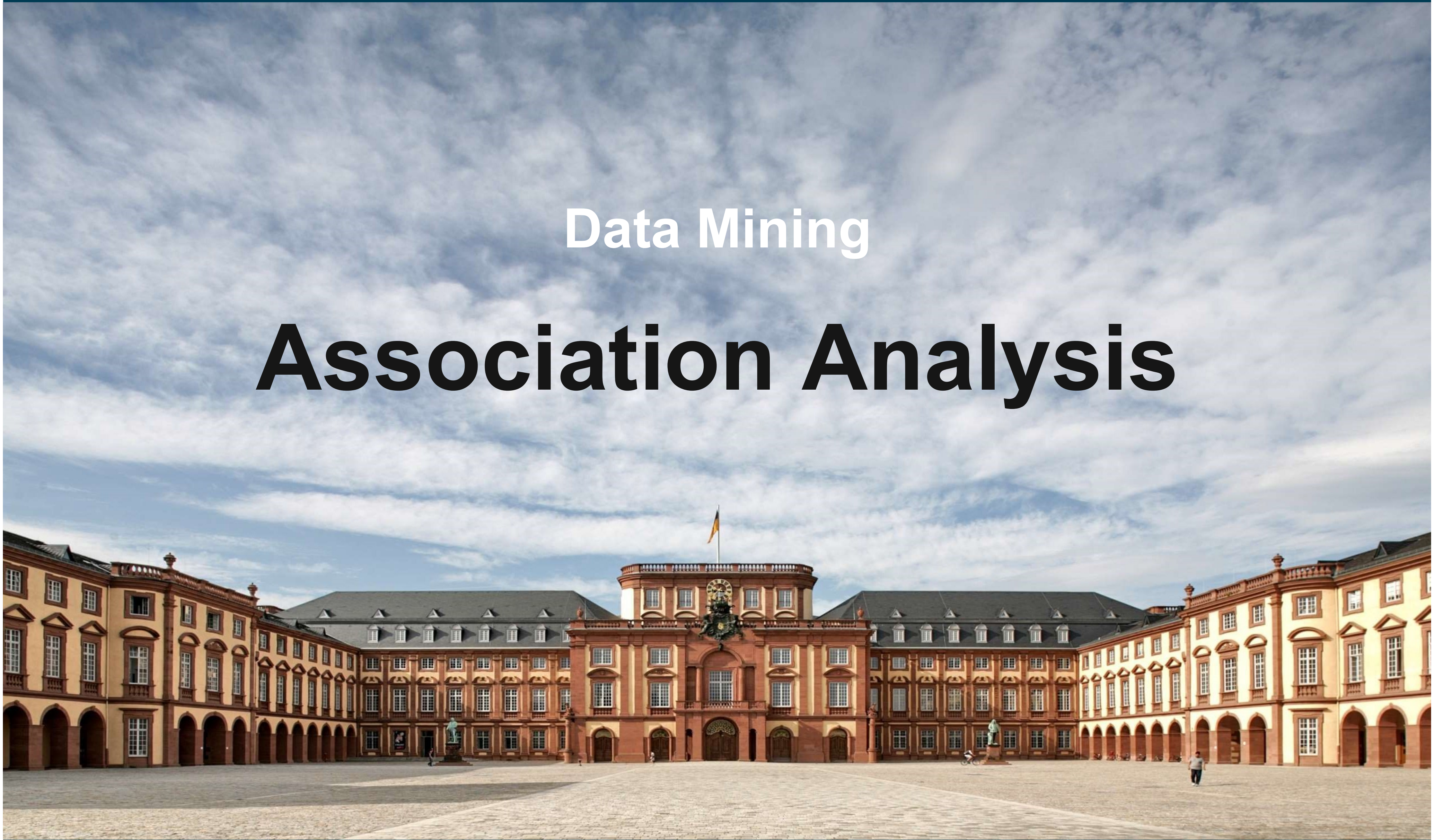
UNIVERSITY OF MANNHEIM



Data Mining

Association Analysis

Example Applications in which Co-Occurrence Matters

- We are often interested in co-occurrence relationships
- Marketing
  1. identify items that are bought together by sufficiently many customers
  2. use this information for marketing or supermarket shelf management purposes

- Inventory Management
  1. identify parts that are often needed together for repairs
  2. use this information to equip your repair vehicles with the right parts

- Usage Mining
  1. identify words that frequently appear together in search queries
  2. use this information to offer auto-completion features to the user

Outline

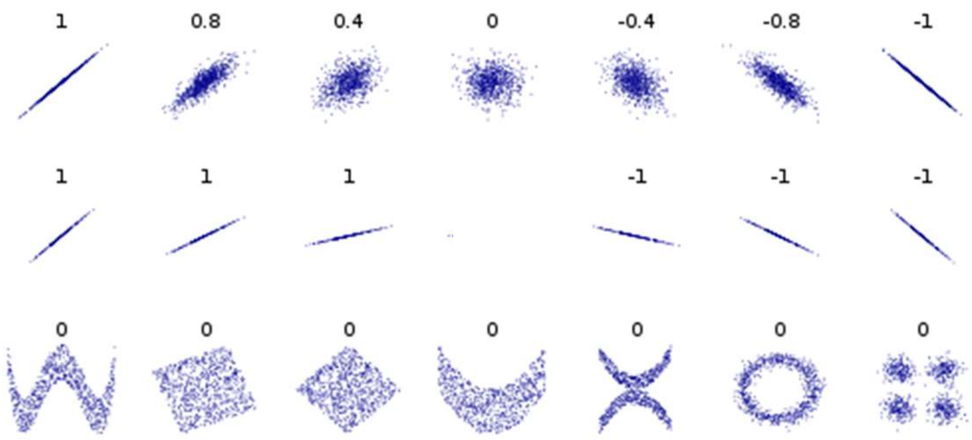
1. Correlation Analysis
2. Association Analysis
   1. Frequent Itemset Generation
   2. Rule Generation
   3. Handling Continuous and Categorical Attributes
   4. Interestingness Measures

1. Correlation Analysis

– Correlation analysis measures the degree of dependency between two variables
  - Continuous variables: Pearson’s correlation coefficient (PCC)
  - Binary variables: Phi coefficient

$$PCC(x,y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}} \qquad Phi(x,y) = \frac{f_{11} f_{00} - f_{01} f_{10}}{\sqrt{f_{1+} f_{+1} f_{0+} f_{+0}}}$$

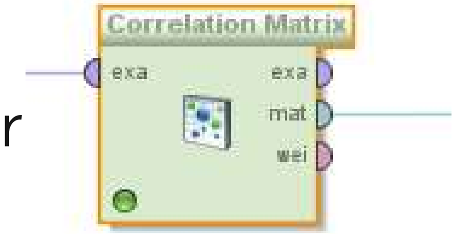
– Value range [-1,1]
  - 1 : positive correlation
  - 0 : variables independent
  - -1 : negative correlation



Correlations between Products in Shopping Baskets

	P1	P2	P3	P4	P5
Basket 1	1	1	0	1	1
Basket 2	1	0	0	1	1
Basket 3	1	0	0	0	1

1 : always bought together
0 : sometimes bought together
-1 : never bought together



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

Shortcoming: Measures correlation only between two items but not between multiple items, e.g. {ThinkPad, Cover} $\rightarrow$ {Minimaus}

2. Association Analysis

- Association analysis can find multiple item co-occurrence relationships (descriptive method)
- focuses on occurring items, not absent items
- first algorithms developed in the early 90s at IBM by Agrawal & Srikant
- initially used for shopping basket analysis to find how items purchased by customers are related
- later extended to more complex data structures
  - sequential patterns
  - subgraph patterns
- and other application domains
  - web usage mining, social science, life science

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.

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

Implication means co-occurrence, not causality!

Definition: Support and Frequent Itemset

- Itemset
  - collection of one or more items
  - example: {Milk, Bread, Diaper}
  - k-itemset: An itemset that contains k items
- Support count (σ)
  - frequency of occurrence of an itemset
  - e.g. σ({Milk, Bread, Diaper}) = 2
- Support (s)
  - fraction of transactions that contain an itemset
  - e.g. s({Milk, Bread, Diaper}) = 2/5 = 0.4
- Frequent Itemset
  - an itemset whose support is greater than or equal to a minimal support (*minsup)* threshold specified by the user

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

Definition: Association Rule

- Association Rule
  - an implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - an association rule states that when X occurs, Y occurs with certain probability.
  - Example:
    {Milk, Diaper} $\rightarrow$ {Beer}
    Condition Consequent

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

- Rule Evaluation Metrics
  - Support (s)
    fraction of transactions that contain both X and Y

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

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

  - Confidence (c)
    measures how often items in Y appear in transactions that contain X

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

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

Main Challenges concerning Association Analysis

1. Mining associations from large amounts of data can be computationally expensive
   - algorithms need to apply smart pruning strategies
2. Algorithms often discover a large number of associations
   - many of them are uninteresting or redundant
   - the user needs to select the subset of the associations that is relevant given her task at hand

The Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  1. support ≥ *minsup* threshold
  2. confidence ≥ *minconf* threshold
- *minsup* and *minconf* are provided by the user.
- Brute Force Approach:
  1. list all possible association rules
  2. compute the support and confidence for each rule
  3. 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

Example 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} → {Milk, Beer} (s=0.4, c=0.5)
{Milk} → {Diaper, Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence.
- Thus, we may decouple the support and confidence requirements.

Mining Association Rules

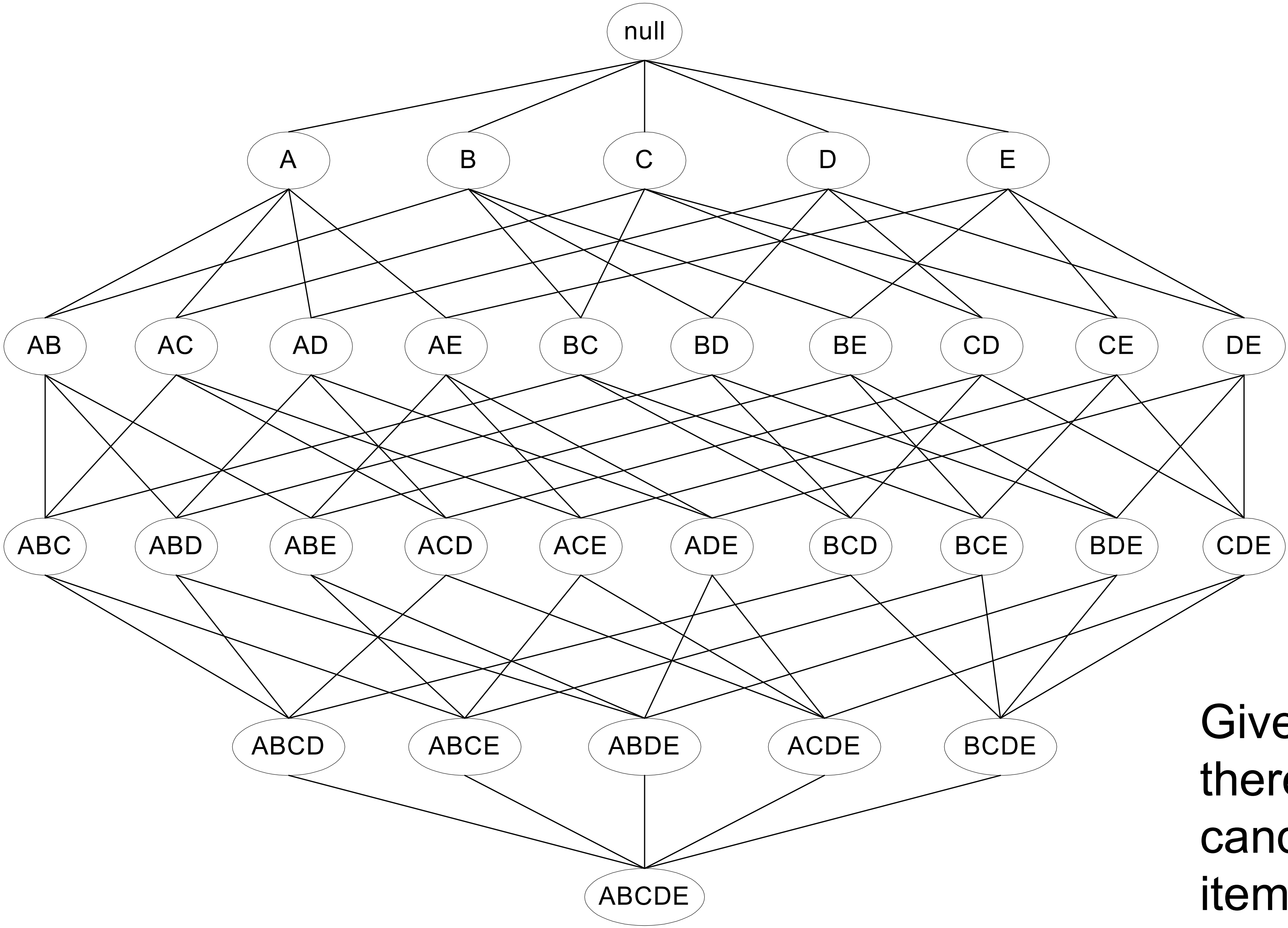
– Two-step approach:

1. Frequent Itemset Generation
   – generate all itemsets whose support ≥ minsup

2. Rule Generation
   – generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

– Frequent itemset generation is still computationally expensive

2.1 Frequent Itemset Generation



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

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

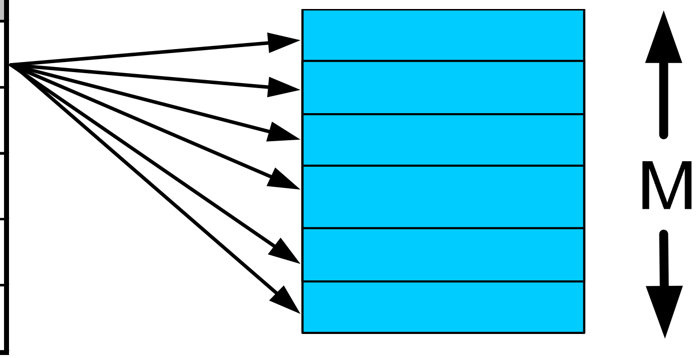
Transactions

N

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

w

List of Candidates



– Complexity ~ O(NMw) ➔ Expensive since $M = 2^d$!!!
– A smarter algorithm is required

Example: Brute Force Approach

– Example:
  - Amazon has 10 million books (i.e., Amazon Germany, as of 2011)
– That is $2^{10.000.000}$ possible itemsets
– As a number:
  - $9.04981... \times 10^{3.010.299}$
  - that is: a number with 3 million digits!
– However:
  - most itemsets will not be important at all, e.g., books on Chinese calligraphy, Inuit cooking, and data mining bought together
  - thus, smarter algorithms should be possible
  - intuition for the algorithm: All itemsets containing Inuit cooking are likely infrequent

Reducing the Number of Candidates

– Apriori Principle

> **If an itemset is frequent, then all of its subsets must also be frequent.**

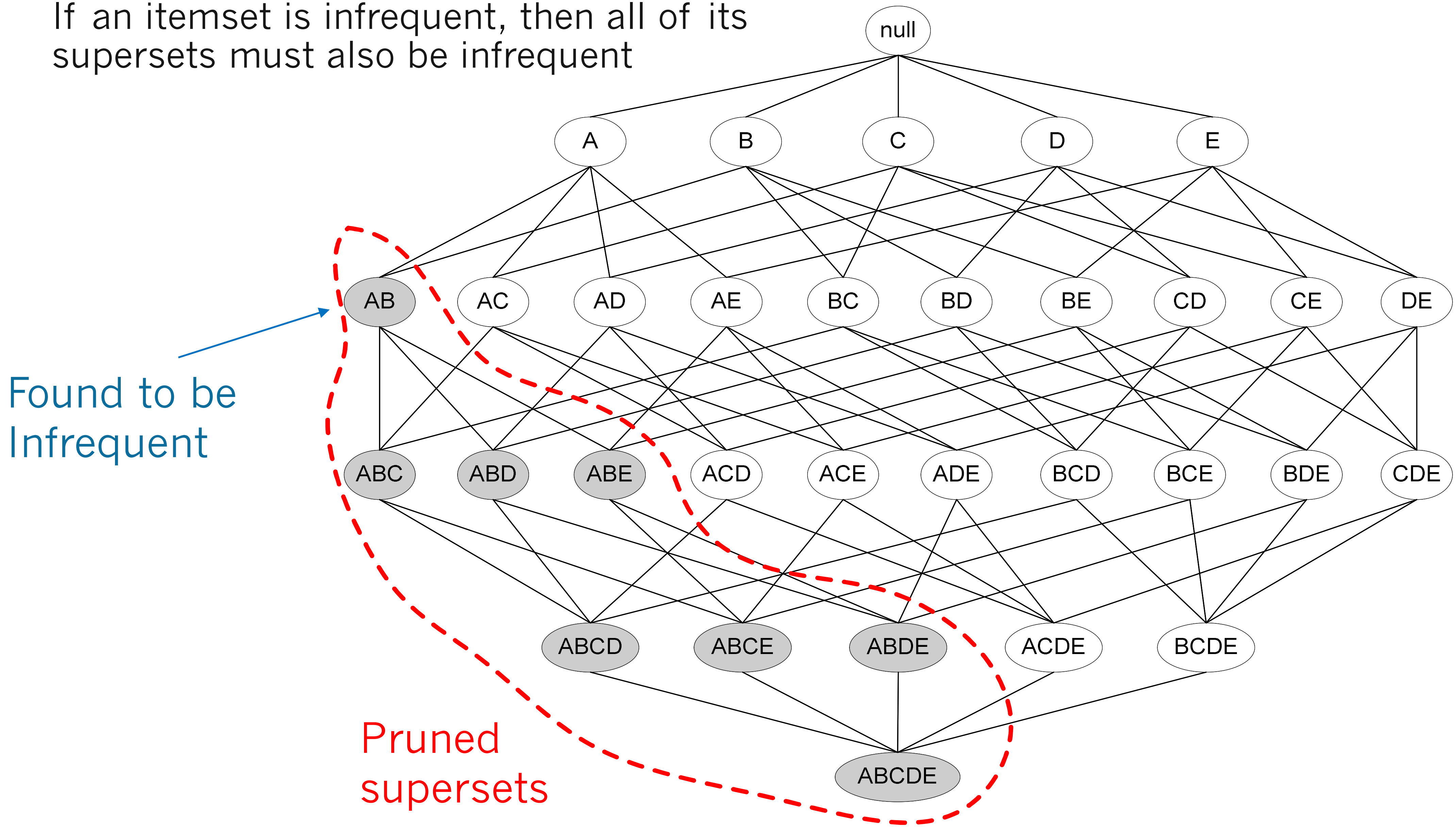
– Apriori principle holds due to the following property of the support measure:

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

- support of an itemset never exceeds the support of its subsets
- this is known as the anti-monotone property of support

Using the Apriori Principle for Pruning

If an itemset is infrequent, then all of its supersets must also be infrequent



Example: Using the Apriori Principle for Pruning

Minimum Support Count = 3

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

Items (1-itemsets)

No need to generate candidates involving Coke or Eggs

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

Pairs (2-itemsets)

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

No need to generate candidate {Milk, Diaper, Beer} as count {Milk, Beer} = 2

The Apriori Algorithm

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

Example: Apriori Algorithm

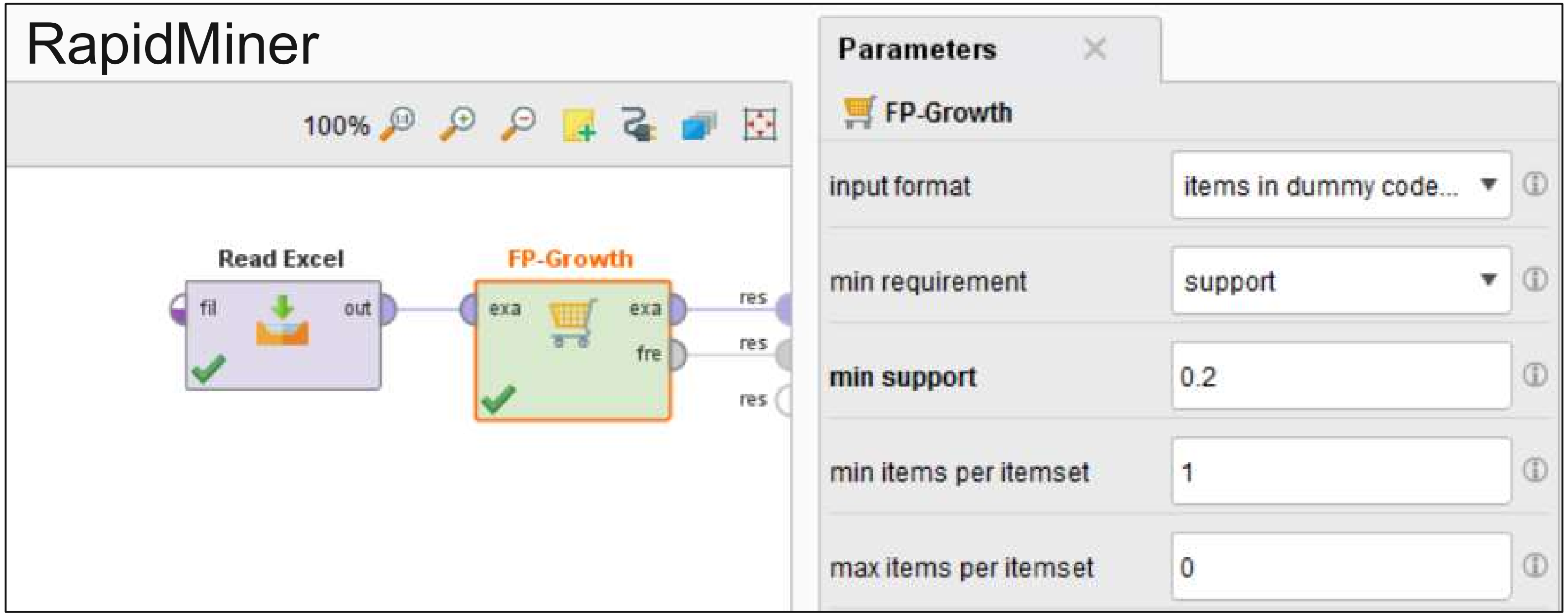
itemset:count minsup=0.5

Dataset T

TID	Items
T100	1, 3, 4
T200	2, 3, 5
T300	1, 2, 3, 5
T400	2, 5

1. scan T
   - ➔ $Cand_1$: {1}:2, {2}:3, {3}:3, {4}:1, {5}:3
   - ➔ $Fequ_1$: {1}:2, {2}:3, {3}:3, {5}:3
   - ➔ $Cand_2$: {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}
2. scan T
   - ➔ $Cand_2$: {1,2}:1, {1,3}:2, {1,5}:1, {2,3}:2, {2,5}:3, {3,5}:2
   - ➔ $Fequ_2$: {1,3}:2, {2,3}:2, {2,5}:3, {3,5}:2
   - ➔ $Cand_3$: {2, 3, 5}
3. scan T
   - ➔ C_3: {2, 3, 5}:2
   - ➔ $F_{3:}$ {2, 3, 5}

Frequent Itemset Generation in Rapidminer and Python



Python

```
from orangecontrib.associate.fpgrowth import frequent_itemsets

# Calculate frequent itemsets
itemsets = dict(frequent_itemsets(dataset.values, 0.20))
```

FP-Growth

Alternative frequent itemset generation algorithm which compresses data into tree structure in memory. Details Tan/Steinbach/Kumar: Chapter 4.6

Frequent Itemsets in Rapidminer

Result History | FrequentItemSets (FP-Growth)

Data

Annotations

No. of Sets: 83

Total Max. Size: 4

Min. Size: 1

Max. Size: 4

Contains Item:

Update View

Size	Support	Item 1	Item 2	Item 3
1	0.600	Asus EeePC		
1	0.500	LT Minimaus		
1	0.500	2 GB DDR3 RAM		
2	0.500	Asus EeePC	2 GB DDR3 RAM	
1	0.400	ThinkPad X220		
1	0.400	Netbook-Schutzhülle		
1	0.400	Lenovo Tablet Sleeve		
1	0.400	LT Laser Maus		
2	0.400	Asus EeePC	LT Minimaus	
2	0.400	Asus EeePC	Netbook-Schutzhülle	
2	0.400	2 GB DDR3 RAM	Netbook-Schutzhülle	
3	0.400	Asus EeePC	2 GB DDR3 RAM	Netbook-Schutzhülle
1	0.300	HP Laserjet P2055		
1	0.300	HP CE50 Toner		
2	0.300	LT Minimaus	2 GB DDR3 RAM	
2	0.300	LT Minimaus	Netbook-Schutzhülle	
2	0.300	ThinkPad X220	Lenovo Tablet Sleeve	
2	0.300	HP Laserjet P2055	HP CE50 Toner	
3	0.300	Asus EeePC	LT Minimaus	2 GB DDR3 RAM

Example Application of Frequent Itemsets

1. Take top-k frequent itemsets of size 2 containing item A
2. Rank second item according to
   - profit made by selling item
   - whether you want to reduce number of items B in stock
   - knowledge about customer preferences
3. Offer special price for combination with top-ranked second item

2.2 Rule Generation

– Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement.

Example Frequent Itemset:

$$\{\text{Milk}, \text{Diaper}, \text{Beer}\}$$

Example Rule:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

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: Large Number of Candidate Rules

- If {A,B,C,D} is a frequent itemset, then the candidate rules are:

ABC $\rightarrow$D,	ABD $\rightarrow$C,	ACD $\rightarrow$B,	BCD $\rightarrow$A,
A $\rightarrow$BCD,	B $\rightarrow$ACD,	C $\rightarrow$ABD,	D $\rightarrow$ABC
AB $\rightarrow$CD,	AC $\rightarrow$ BD,	AD $\rightarrow$ BC,	BC $\rightarrow$AD,
BD $\rightarrow$AC,	CD $\rightarrow$AB		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Rule Generation

– How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone with respect to the number of items on the right hand side of the rule

Explanation

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

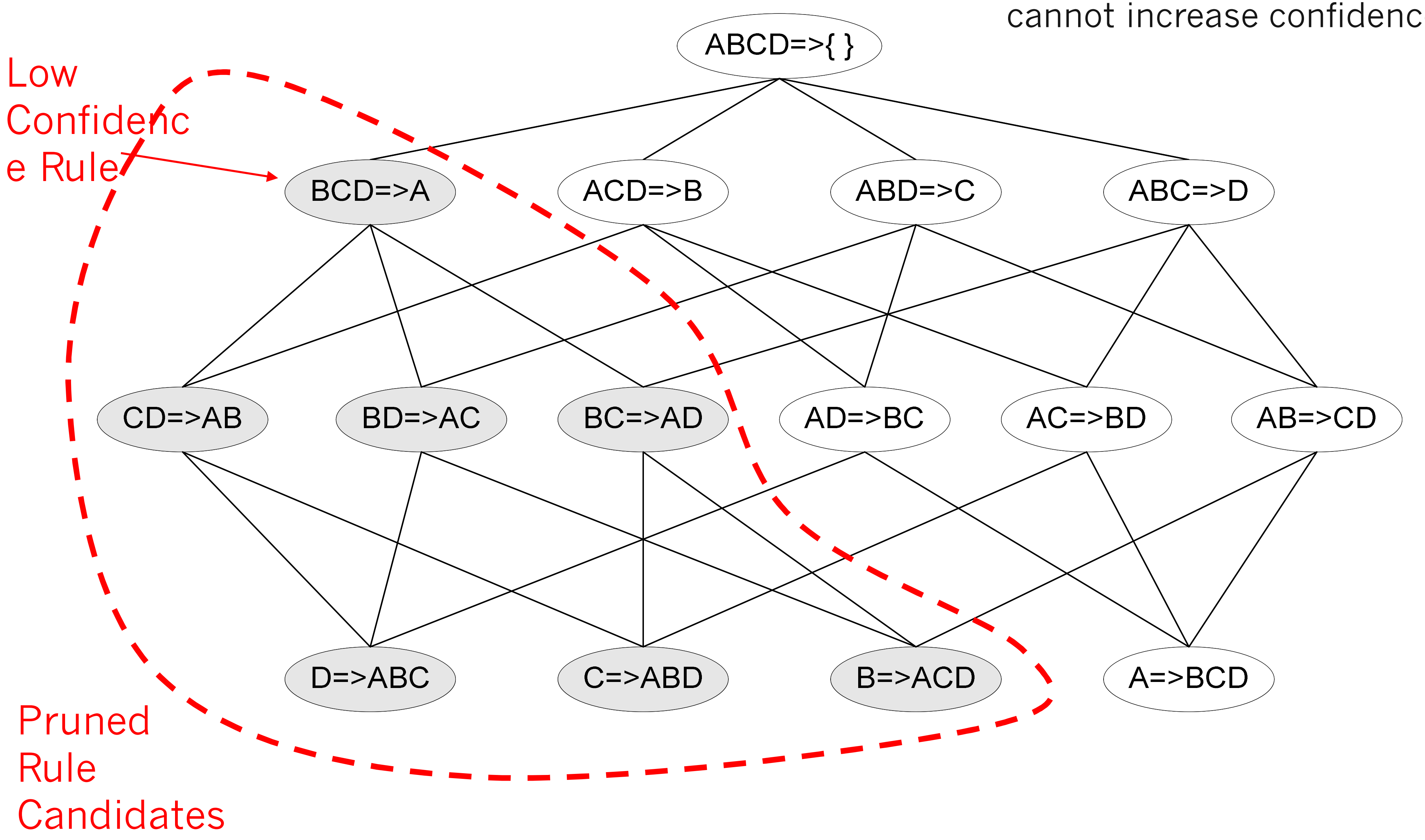
  s(AB) ≤ s(A)

- Hence

  c(AB → C) ≥ C(A → BC)

Candidate Rule Pruning

Moving elements from left to right cannot increase confidence

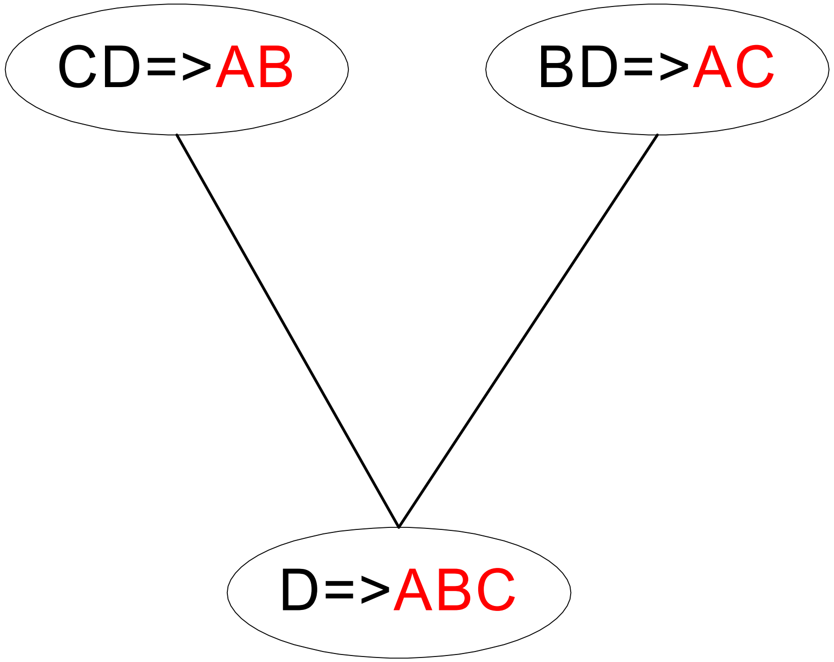


Candidate Rule Generation within Apriori Algorithm

– Candidate rule is generated by merging two rules that share the same prefix in the rule consequent (right hand side of rule)

1. join(CD $\rightarrow$ AB, BD $\rightarrow$ AC) would produce the candidate rule D $\rightarrow$ ABC

2. Prune rule D $\rightarrow$ ABC if one of its parent rules does not have high confidence (e.g. AD $\rightarrow$ BC)

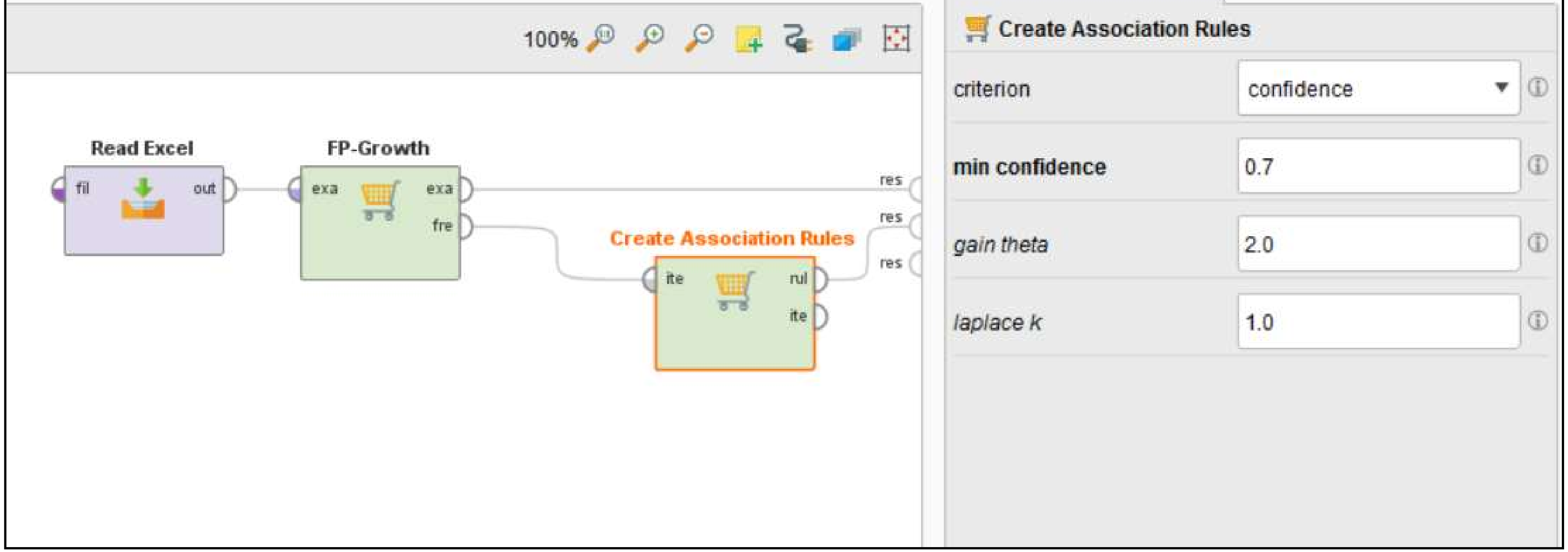


– All the required information for confidence computation has already been recorded in itemset generation.

– Thus, there is no need to scan the transaction data *T* any more

Creating Association Rules in Rapidminer and Python

RapidMiner



Python

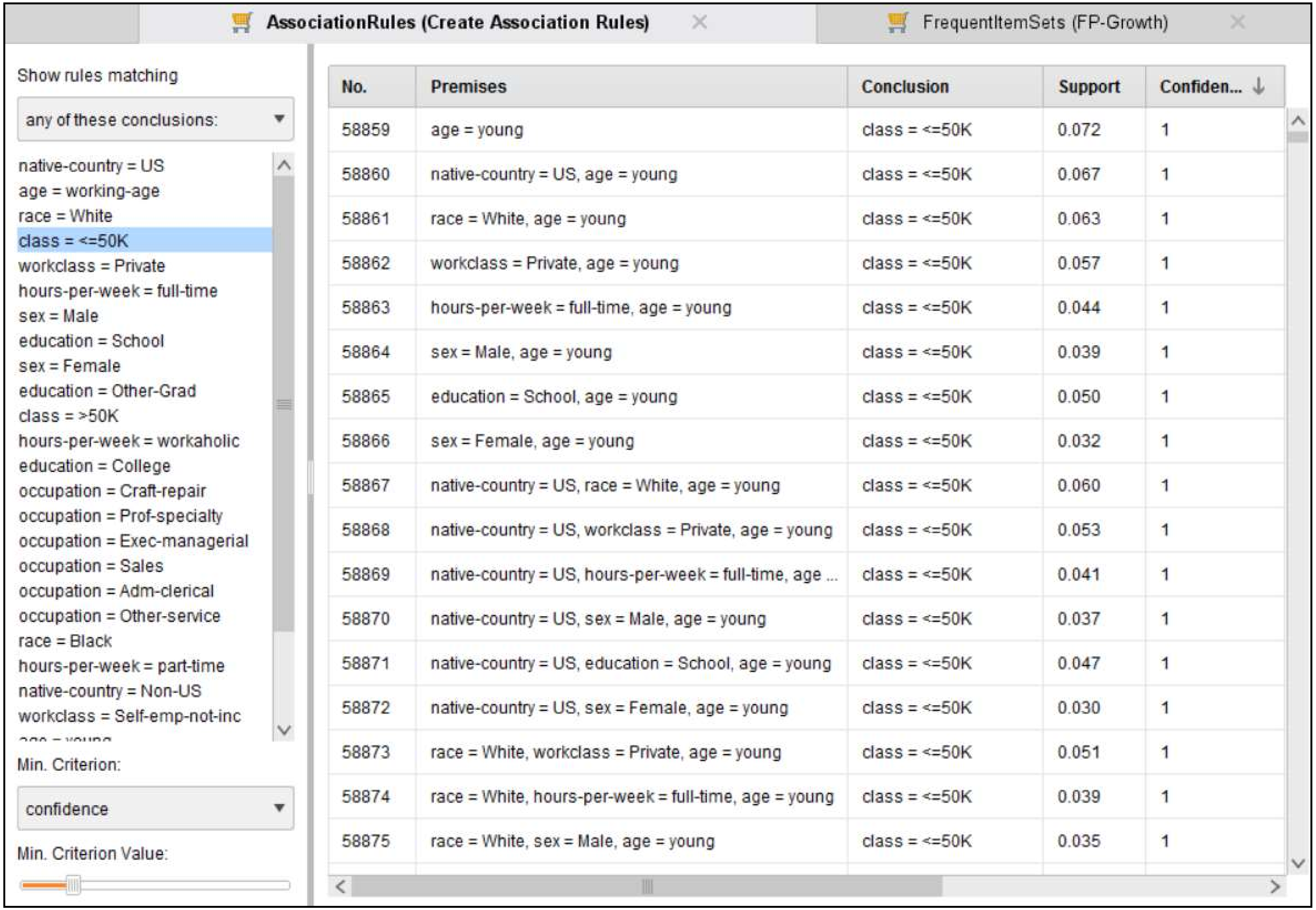
```python
from orangecontrib.associate.fpgrowth import association_rules

# Calculate association rules from itemsets
rules = association_rules(itemsets, 0.70)
```

Exploring Association Rules in Rapidminer

Filter by conclusion

Filter by confidence



No.	Premises	Conclusion	Support	Confiden...
58859	age = young	class = <=50K	0.072	1
58860	native-country = US, age = young	class = <=50K	0.067	1
58861	race = White, age = young	class = <=50K	0.063	1
58862	workclass = Private, age = young	class = <=50K	0.057	1
58863	hours-per-week = full-time, age = young	class = <=50K	0.044	1
58864	sex = Male, age = young	class = <=50K	0.039	1
58865	education = School, age = young	class = <=50K	0.050	1
58866	sex = Female, age = young	class = <=50K	0.032	1
58867	native-country = US, race = White, age = young	class = <=50K	0.060	1
58868	native-country = US, workclass = Private, age = young	class = <=50K	0.053	1
58869	native-country = US, hours-per-week = full-time, age ...	class = <=50K	0.041	1
58870	native-country = US, sex = Male, age = young	class = <=50K	0.037	1
58871	native-country = US, education = School, age = young	class = <=50K	0.047	1
58872	native-country = US, sex = Female, age = young	class = <=50K	0.030	1
58873	race = White, workclass = Private, age = young	class = <=50K	0.051	1
58874	race = White, hours-per-week = full-time, age = young	class = <=50K	0.039	1
58875	race = White, sex = Male, age = young	class = <=50K	0.035	1

2.3 Handling Continuous and Categorical Attributes

– How to apply association analysis to attributes that are not asymmetric binary variables?

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

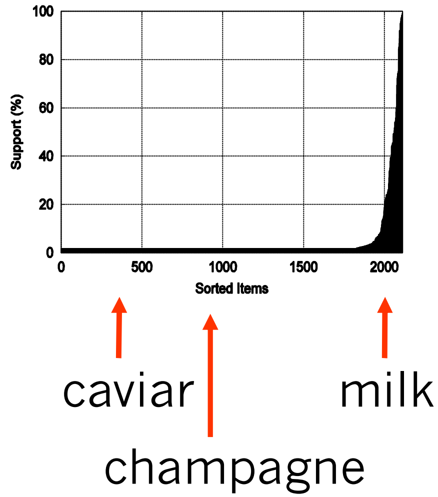
– Example Rule:

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

Handling Categorical Attributes

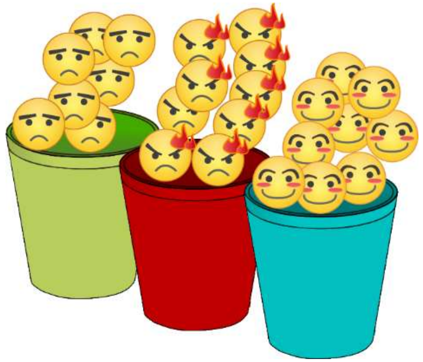
– Transform categorical attribute into asymmetric binary variables
– Introduce a new “item” for each distinct attribute-value pair
  - e.g. replace “Browser Type” attribute with
    – attribute: “Browser Type = Chrome”
    – attribute: “Browser Type = Firefox”
    – …..

– Issues
1. What if attribute has many possible values?
   – many of the attribute values may have very low support
   – potential solution: aggregate low-support attribute values
2. What if distribution of attribute values is highly skewed?
   - 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 item



Handling Continuous Attributes

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



– Issue: Size of the discretization intervals affects support & confidence

$$\{Refund = No, (Income = \$51{,}251)\} \rightarrow \{Cheat = No\}$$

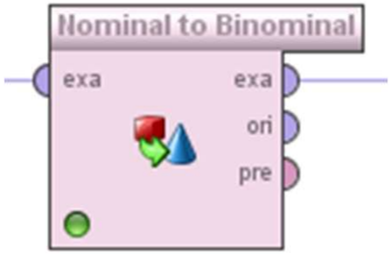
$$\{Refund = No, (60K \leq Income \leq 80K)\} \rightarrow \{Cheat = No\}$$

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

- If intervals are too small
  - itemsets may not have enough support
- If intervals too large
  - rules may not have enough confidence
  - e.g. combination of different age groups compared to a specific age group

Attribute Transformation in RapidMiner and Python

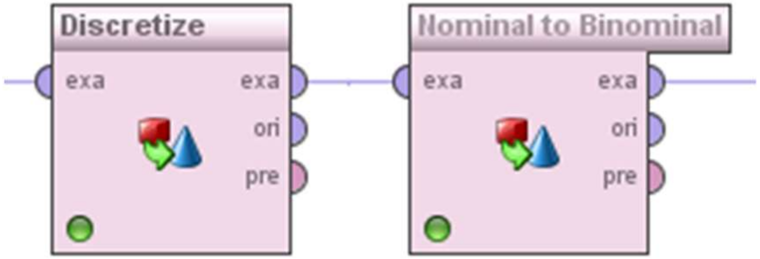
– Categorical attribute values to binary attributes




```
from sklearn.preprocessing import OneHotEncoder

# Apply one-hot encoding
encoder = OneHotEncoder(sparse=False)
onehot_data = encoder.fit_transform(dataset)
```

– Continuous attribute values to binary attributes




```
from sklearn.preprocessing import KBinsDiscretizer

# Discretize and one-hot encode dataset
discretizer = KBinsDiscretizer(n_bins=5, encode='onehot', strategy='quantile')
discretized_onehot_data = discretizer.fit_transform(dataset)
```

2.4 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 of patterns depends on application
  - one man's rubbish may be another's treasure
- Interestingness measures can be used to prune or rank the derived rules.
- In the original formulation of association rules, support & confidence were the only interestingness measures used.
- Later, various other measures have been proposed
  - See Tan/Steinbach/Kumar, Chapter 6.7
  - We will have a look at one: Lift

Drawback of Confidence

Contingency table

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

Association Rule: Tea $\rightarrow$ Coffee

- confidence(Tea $\rightarrow$ Coffee) = 0.75
- but support(Coffee) = 0.9
- although confidence is high, rule is misleading as the fraction of coffee drinkers is higher than the confidence of the rule
- we want confidence(X $\rightarrow$ Y) > support(Y)
- otherwise rule is misleading as X reduces probability of Y

Lift

– The lift of an association rule $X \rightarrow Y$ is defined as:

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

– Confidence normalized by support of consequent

– Interpretation
  - if lift > 1, then X and Y are positively correlated
  - if lift = 1, then X and Y are independent
  - if lift < 1, then X and Y are negatively correlated

Example: Lift

Contingency table

	Coffee	Coffee	
Tea	15	5	20
Tea	75	5	80
	90	10	100

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

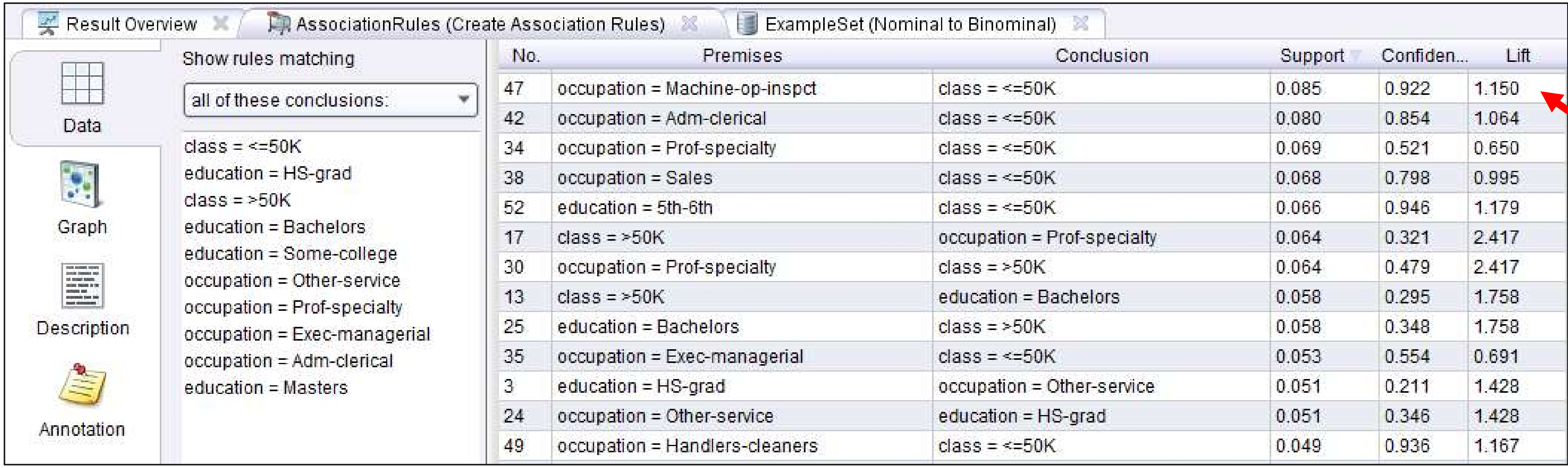
Association Rule: Tea $\rightarrow$ Coffee

- confidence(Tea $\rightarrow$ Coffee) = 0.75
- but support(Coffee) = 0.9

Lift(Tea $\rightarrow$ Coffee) = 0.75/0.9= 0.8333

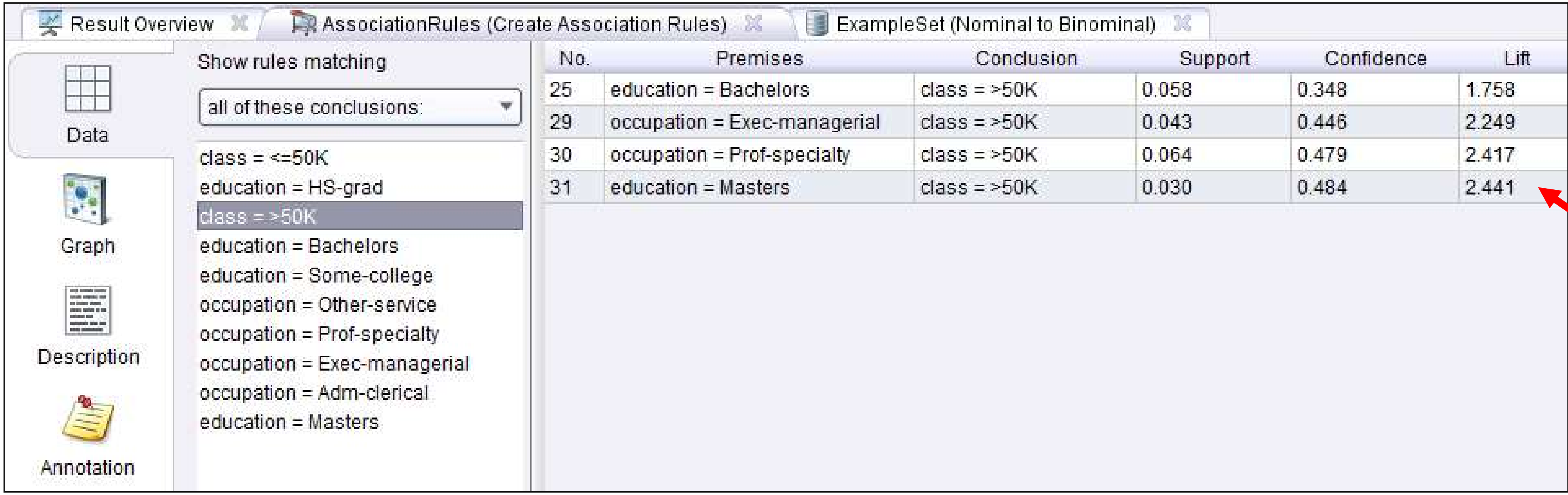
- lift < 1, therefore is negatively correlated

Exploring Association Rules in RapidMiner



No.	Premises	Conclusion	Support	Confiden...	Lift
47	occupation = Machine-op-inspct	class = <=50K	0.085	0.922	1.150
42	occupation = Adm-clerical	class = <=50K	0.080	0.854	1.064
34	occupation = Prof-specialty	class = <=50K	0.069	0.521	0.650
38	occupation = Sales	class = <=50K	0.068	0.798	0.995
52	education = 5th-6th	class = <=50K	0.066	0.946	1.179
17	class = >50K	occupation = Prof-specialty	0.064	0.321	2.417
30	occupation = Prof-specialty	class = >50K	0.064	0.479	2.417
13	class = >50K	education = Bachelors	0.058	0.295	1.758
25	education = Bachelors	class = >50K	0.058	0.348	1.758
35	occupation = Exec-managerial	class = <=50K	0.053	0.554	0.691
3	education = HS-grad	occupation = Other-service	0.051	0.211	1.428
24	occupation = Other-service	education = HS-grad	0.051	0.346	1.428
49	occupation = Handlers-cleaners	class = <=50K	0.049	0.936	1.167

Lift close to 1



No.	Premises	Conclusion	Support	Confidence	Lift
25	education = Bachelors	class = >50K	0.058	0.348	1.758
29	occupation = Exec-managerial	class = >50K	0.043	0.446	2.249
30	occupation = Prof-specialty	class = >50K	0.064	0.479	2.417
31	education = Masters	class = >50K	0.030	0.484	2.441

Solid lift

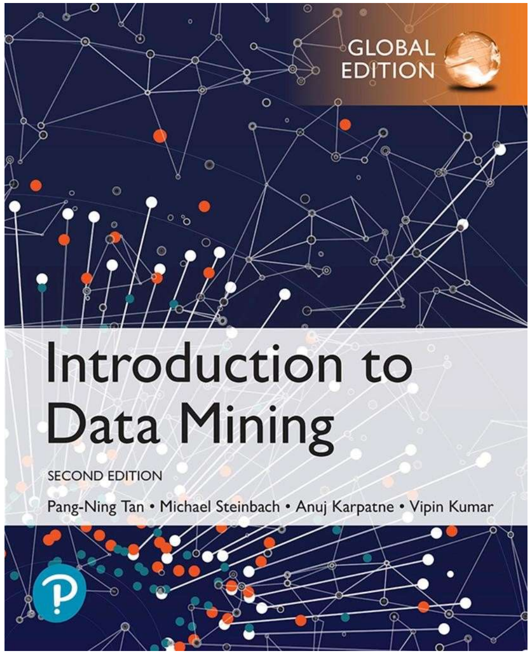
Conclusion

– The algorithm does the counting for you and finds patterns in the data

– You need to do the interpretation based on your knowledge about the application domain.
  - Which patterns are meaningful?
  - Which patterns are surprising?

Literature for this Slideset

Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar: **Introduction to Data Mining.** 2nd Edition. Pearson.



Chapter 4: Association Analysis: Basic Concepts and Algorithms

Chapter 7: Association Analysis: Advanced Concepts