Association rules

Reguły asocjacyjne i zbiory częste

Lecturer: JERZY STEFANOWSKI
Institute of Computing Sciences
Poznan University of Technology
Poznan, Poland

Master Course ITI/ISWD

Update for 2016 / 2020

Acknowledgments:

This lecture is based on the following resources - slides:
G.Piatetsky-Shapiro: Association Rules and Frequent Item Analysis.
and partly on two lectures
J.Han: Mining Association Rules in Large Databases;
Tan, Steinbach, Kumar: Introduction to Data Mining
and my other notes.

Wykład będzie używał slajdów w języku angielskim

Outline

- Transactions
- Frequent itemsets
- Subset Property
- Association rules
- Applications

Association rules

- Transaction data
- Market basket analysis

TID	Produce
1	MILK, BREAD, EGGS
2	BREAD, SUGAR
3	BREAD, CEREAL
4	MILK, BREAD, SUGAR
5	MILK, CEREAL
6	BREAD, CEREAL
7	MILK, CEREAL
8	MILK, BREAD, CEREAL, EGGS
9	MILK, BREAD, CEREAL

- {Cereal, Milk} → Bread [sup=5%, conf=80%]
- Association rule:
  „80% of customers who buy *cereal* and *milk* also buy *bread* and 5% of customers buy all these products together"

Association means co-occurrence, not causality!

Frequent Pattern Analysis and Associations

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together? — Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Transactions Example

TID	Produce
1	MILK, BREAD, EGGS
2	BREAD, SUGAR
3	BREAD, CEREAL
4	MILK, BREAD, SUGAR
5	MILK, CEREAL
6	BREAD, CEREAL
7	MILK, CEREAL
8	MILK, BREAD, CEREAL, EGGS
9	MILK, BREAD, CEREAL

Transaction database: Example, 1

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

ITEMS:

Instances = Transactions

A = milk
B= bread
C= cereal
D= sugar
E= eggs

Transaction database: Example, 2

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

Attributes converted to binary flags

TID	A	B	C	D	E
1	1	1	0	0	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	1	0	1	0
5	1	0	1	0	0
6	0	1	1	0	0
7	1	0	1	0	0
8	1	1	1	0	1
9	1	1	1	0	0

Definitions

- Item: *attribute*=*value* pair or simply *value*
  - usually attributes are converted to binary *flags* for each value, e.g. **product="A"** is written as **"A"**
- Itemset I : a subset of possible items
  - Example: I = {A,B,E} (order unimportant!)
- *k*-itemset
    - An itemset that contains *k* items
- Transaction: (TID, itemset)
  - TID is transaction ID

Support and Frequent Itemsets

- Support of an itemset
  - sup(I) = no. of transactions t that support (i.e. contain) I
  - e.g. I={A,B,E} and TID8 {A,B,C,E}
- In the exemplary database:
  - sup ({A,B,E}) = 2, sup ({B,C}) = 4
- Support could be also expressed as a fraction
  - s ({B,C}) = 4/9
- Frequent itemset I is one with at least the minimum support count
  - sup(I) >= *minsup*

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

SUBSET PROPERTY (Agrawal et al..)

- **Every subset of a frequent set is frequent!**
- A: Example: Suppose {A,B} is frequent. Since each occurrence of A,B includes both A and B, then both A and B must also be frequent
- Similar argument for larger itemsets
- Almost all association rule algorithms are based on this subset property

Association Rules

- Association rule *R* : *Itemset1* => *Itemset2*
  - *Itemset1, 2* are disjoint and *Itemset2* is non-empty
  - meaning: if transaction includes *Itemset1* then it also has *Itemset2*
- Examples
  - A,B => E,C
  - A => B,C

From Frequent Itemsets to Association Rules

- *Q: Given frequent set {A,B,E}, what are possible association rules?*
  - A => B, E
  - A, B => E
  - A, E => B
  - B => A, E
  - B, E => A
  - E => A, B
  - __ => A,B,E (empty rule), or true => A,B,E

Rule Support and Confidence

- Suppose $R: I => J$ is an association rule
  - sup (R) = sup (I ∪ J) is the *support count*
    - support of itemset I ∪ J (should be both I and J)
  - conf (R) = sup(R) / sup(I) is the *confidence* of R
    - fraction of transactions with I ∪ J that have J
  - Supports could be also expressed in a relative form as fractions
- Association rules with minimum support and count are sometimes called "***strong***" rules

Mining Association Rules—an Example

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 2 / 50%
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule $A \Rightarrow C$:

support = support($\{A\} \cup \{C\}$) = 2 trans =50%

confidence = support($\{A\} \cup \{C\}$)/support($\{A\}$) =

= 2/3 =66.6%

Classification vs Association Rules

Classification Rules

- Focus on one target field
- Specify class in all cases
- Measures: Accuracy

Association Rules

- Many target fields
- Applicable in some cases
- Measures: Support, Confidence, Lift

Association Rules Example, 1

- ***Q: Given frequent set {A,B,E}, what association rules have minsup = 2 and minconf= 50% ?***

  A, B => E : conf=2/4 = 50%

TID	List of items
1	**A, B, E**
2	B, D
3	B, C
4	**A, B,** D
5	A, C
6	B, C
7	A, C
8	**A, B, C, E**
9	**A, B**, C

Association Rules Example, 2

- ***Q: Given frequent set {A,B,E}, what association rules have minsup = 2 and minconf= 50% ?***

  A, B => E : conf=2/4 = 50%

  A, E => B : conf=2/2 = 100%

  B, E => A : conf=2/2 = 100%

  E => A, B : conf=2/2 = 100%

Don't qualify

A =>B, E : conf=2/6 =33%< 50%

B => A, E : conf=2/7 = 28% < 50%

__ => A,B,E : conf: 2/9 = 22% < 50%

TID	List of items
1	**A, B, E**
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	**A, B, C, E**
9	A, B, C

Find Strong Association Rules

- A rule has the parameters *minsup* and *minconf*:
  - sup(R) >= *minsup* and conf (R) >= *minconf*
- **Problem**:
  - Find all association rules with given *minsup* and *minconf*
- How could we discover such rules?

Brute force approach?

- List all possible association rules
- Compute the support and confidence for each rule
- Prune rules that fail the *minsup* and *minconf* thresholds

$\Rightarrow$ Computationally prohibitive!

However, how to list all possible association rules?

Mining Association Rules → decomposing the problem

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

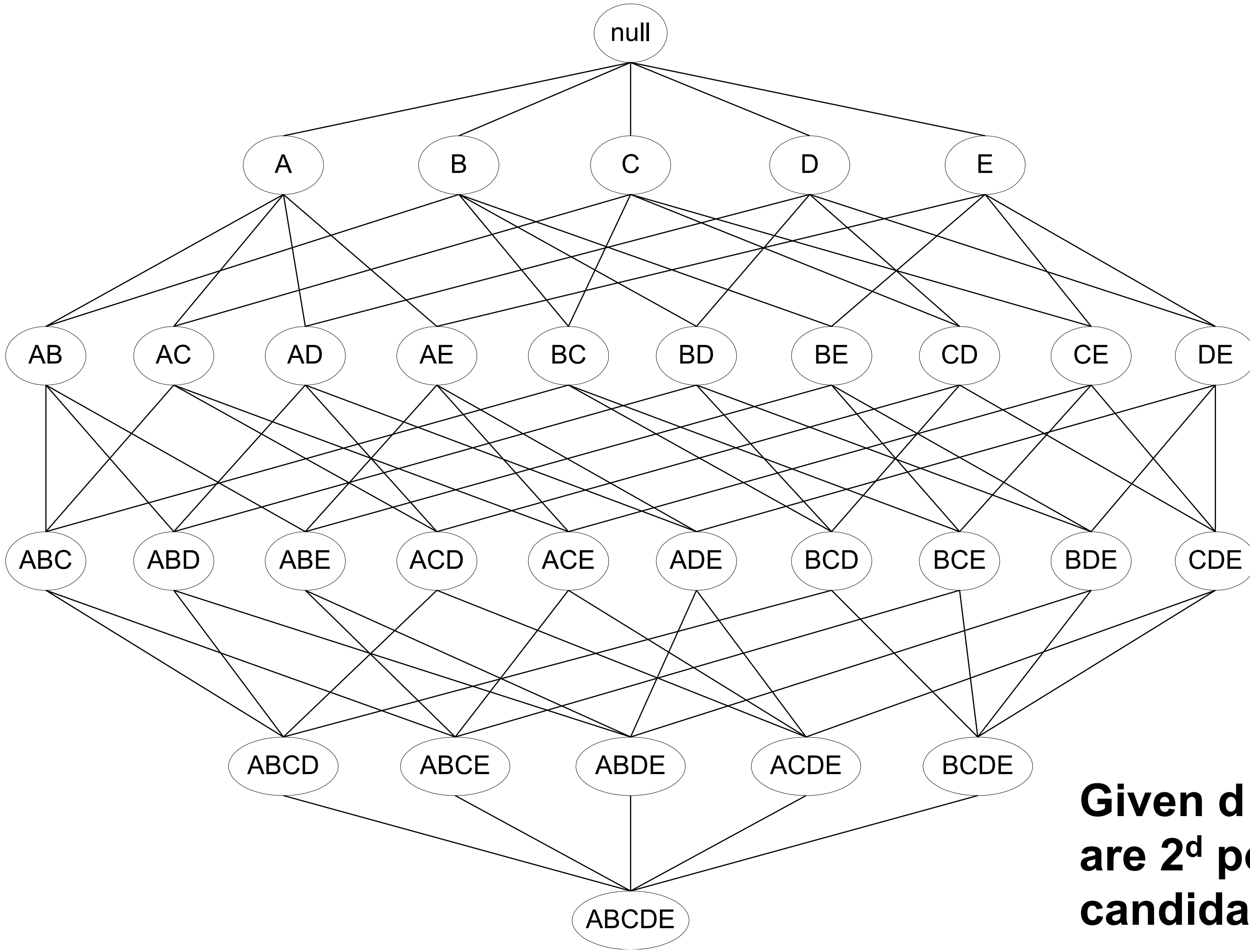
Observations:

- 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!

Decomposing 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

Frequent Itemset Generation



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

Frequent Itemset Generation?

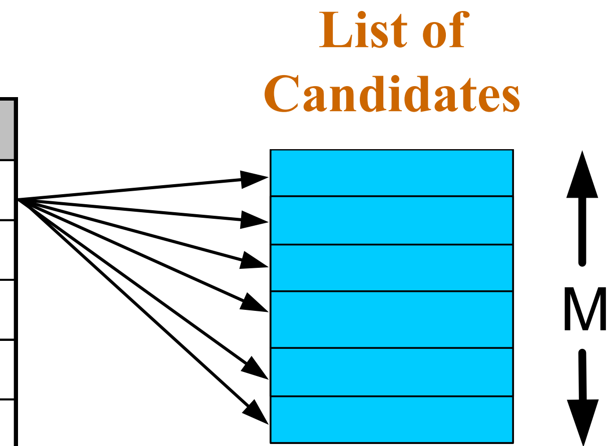
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

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



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Finding Frequent Itemsets → Apriori

- Before generating rules → first, find all frequent itemsets – however in appropriate way using extra pruning properties.
- Let us consider ideas proposed by Agrawal et al..
- Start by finding one-item sets (easy)
- *Q: How?*
- A: Simply count the frequencies of all items

Apriori finding itemsets: next level

- Apriori algorithm (Agrawal & Srikant 94)
- Idea: use one-item sets to generate two-item sets, two-item sets to generate three-item sets, …
  - If (A B) is a frequent item set, then (A) and (B) have to be frequent item sets as well!
  - In general: if X is frequent *k*-item set, then all (*k*-1)-item subsets of X are also frequent

  $\Rightarrow$ Compute *k*-item set by merging (*k*-1)-item sets

Reducing Number of Candidates

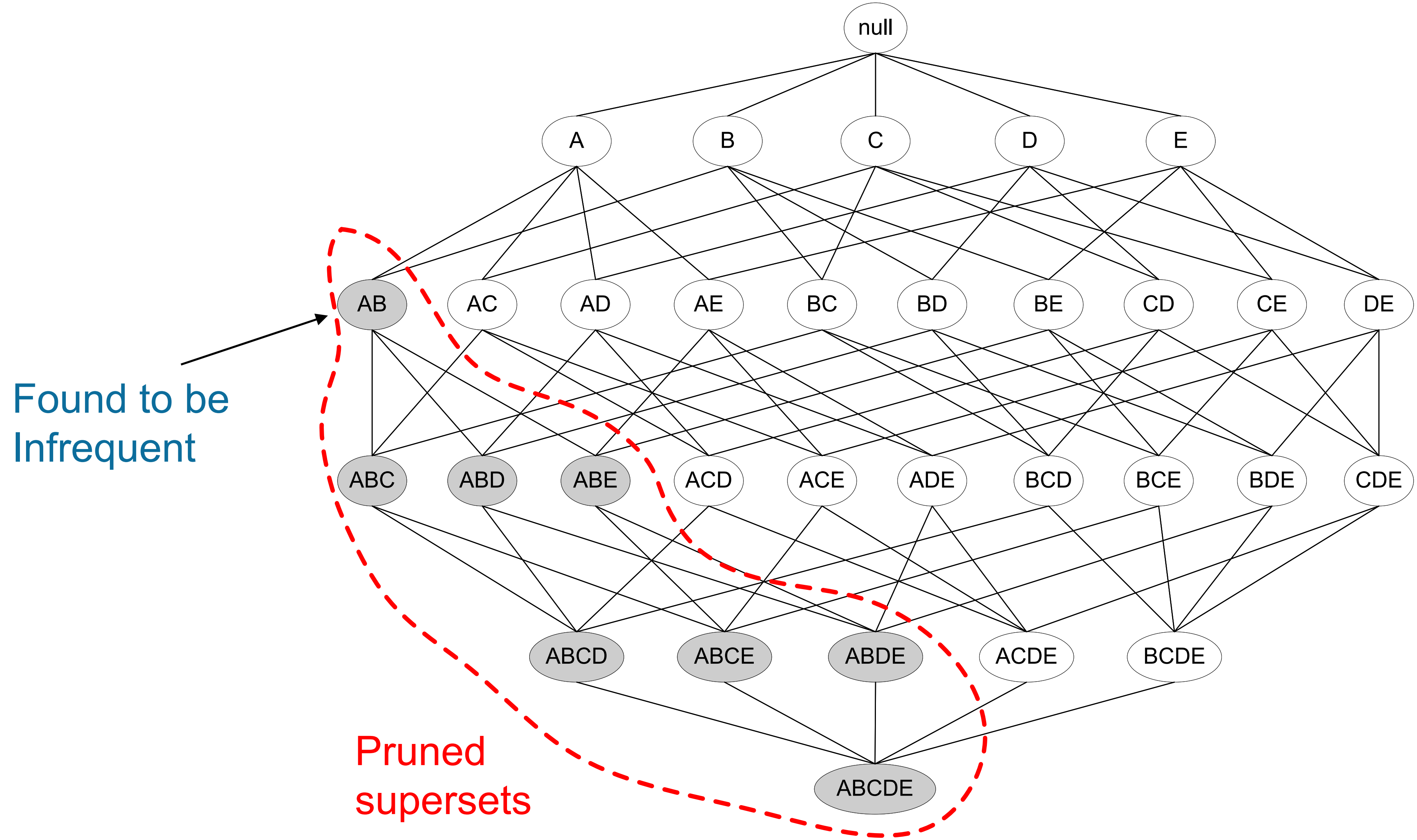
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent!

- 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

Illustrating Apriori Principle



Apriori: A Candidate Generation-and-test Approach - Summary

- Any subset of a frequent itemset must be frequent
  - if **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - generate length (k+1) candidate itemsets from length k frequent itemsets, and
  - test the candidates against DB
- The performance studies show its efficiency and scalability
- Agrawal & Srikant 1994, Mannila, et al. 1994

Apriori – trick of using lexicographic order in generating (K+1) itemsets

- Given: five three-item sets

  `(A B C), (A B D), (A C D), (A C E), (B C D)`

- Lexicographic order improves efficiency!

  Create C_k from L_{k-1} p, L_{k-1} q

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

- Candidate four-item sets:

  (A B C D) Q: OK?

A: yes, because all 3-item subsets are frequent

  (A C D E) Q: OK?

A: No, because (C D E) is not frequent

The Apriori Algorithm—An Example

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

L_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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm

- <u>Pseudo-code</u>:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k

  L_1 = {frequent items};
  for ($k = 1$; $L_k \mathrel{!}= \emptyset$; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
  increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
  return $\cup_k L_k$;

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into $\boldsymbol{C_k}$

  select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

  from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

  where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
- Step 2: pruning

  forall ***itemsets c in*** $\boldsymbol{C_k}$ do

  forall ***(k-1)-subsets s of c*** do

  if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

Generating Association Rules

- Two stage process:
  - Determine frequent itemsets e.g. with the Apriori algorithm.
  - For each frequent item set I
    - for each subset J of I
      - determine all association rules of the form: $I - J => J$
- Is it efficient?
  - Main idea used in both stages : subset property

Rule Generation

- Given a frequent itemset I, find all non-empty subsets $J \subset I$ such that $I-J \rightarrow J$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

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

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

Rule Generation

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

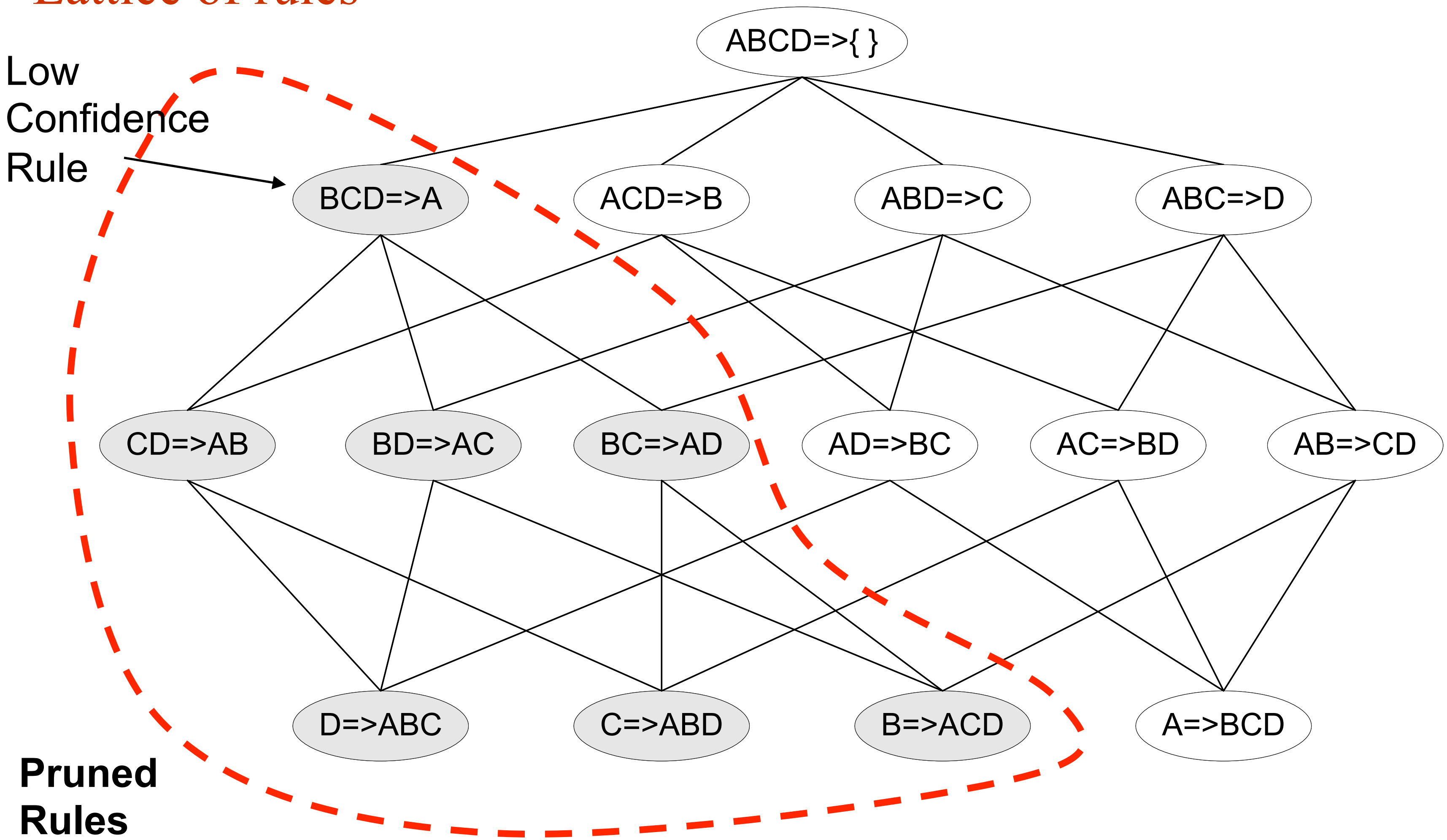
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., $L = \{A,B,C,D\}$:

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

    - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

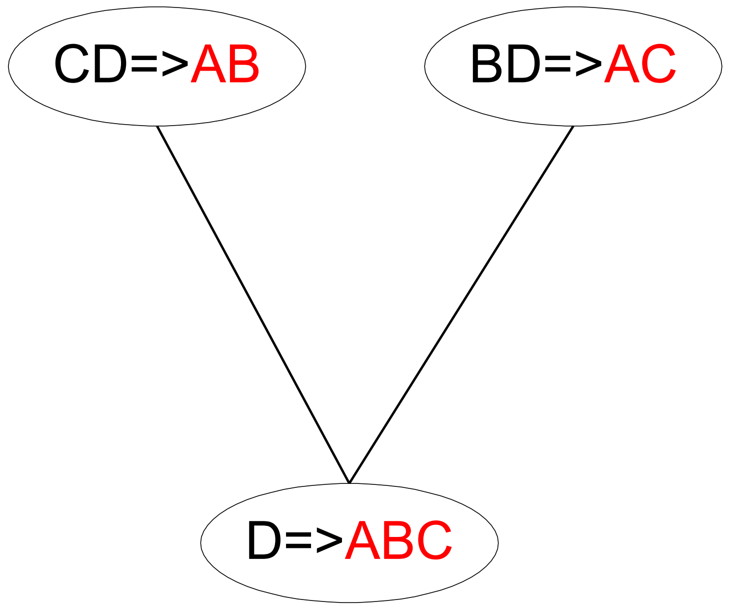
Rule Generation for Apriori Algorithm

Lattice of rules



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

Example: Generating Rules from an Itemset

- Frequent itemset from golf data:

```
Humidity = Normal, Windy = False, Play = Yes (4)
```

- Seven potential rules:

```
If Humidity = Normal and Windy = False then Play = Yes          4/4
If Humidity = Normal and Play = Yes then Windy = False          4/6
If Windy = False and Play = Yes then Humidity = Normal          4/6
If Humidity = Normal then Windy = False and Play = Yes          4/7
If Windy = False then Humidity = Normal and Play = Yes          4/8
If Play = Yes then Humidity = Normal and Windy = False          4/9
If True then Humidity = Normal and Windy = False and Play = Yes 4/12
```

Rules for the weather data

- Rules with support > 1 and confidence = 100%:

	Association rule		Sup.	Conf.
1	Humidity=Normal Windy=False	⇒Play=Yes	4	100%
2	Temperature=Cool	⇒Humidity=Normal	4	100%
3	Outlook=Overcast	⇒Play=Yes	4	100%
4	Temperature=Cold Play=Yes	⇒Humidity=Normal	3	100%
...	...	...	...	...
58	Outlook=Sunny Temperature=Hot	⇒Humidity=High	2	100%

- In total: 3 rules with support four, 5 with support three, and 50 with support two

How to Count Supports of Candidates?

- 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*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Factors Affecting Complexity

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Association Rule Mining

mining association rules
(Agrawal et. al SIGMOD93)

Better algorithms

Fast algorithm
(Agrawal et. al VLDB94)

Hash-based
(Park et. al SIGMOD95)

(Navathe et. al VLDB95)

Direct Itemset Counting
(Brin et. al SIGMOD97)

Problem extension

Generalized A.R.
(Srikant et. al; Han et. al. VLDB95)

Quantitative A.R.
(Srikant et. al SIGMOD96)

N-dimensional A.R.
(Lu et. al DMKD'98)

Meta-ruleguided mining

Parallel mining
(Agrawal et. al TKDE96)

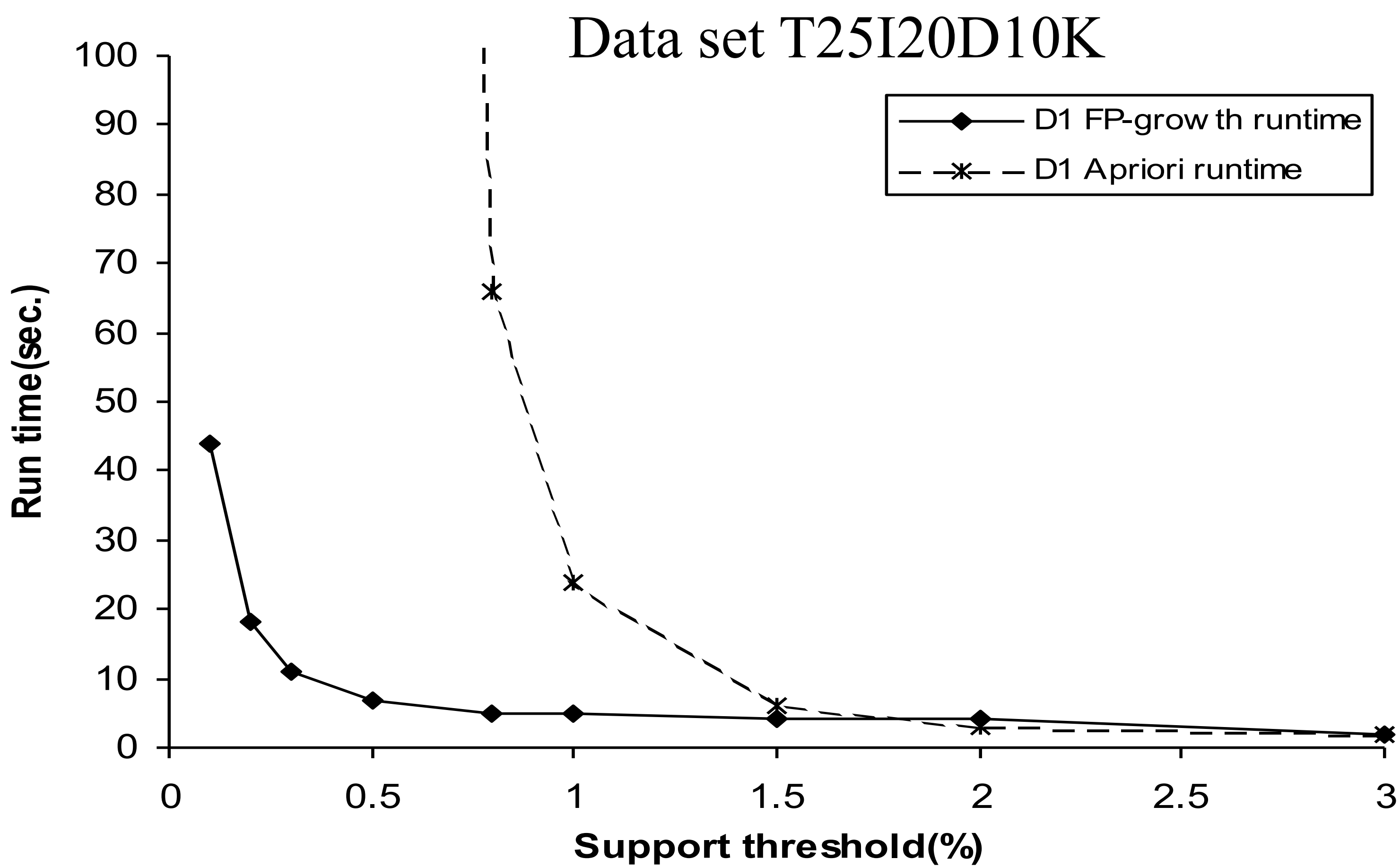
Distributed mining
(Cheung et. al PDIS96)

Incremental mining
(Cheung et. al ICDE96)

Bottleneck of Frequent-pattern Mining with Apriori

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1 = 1.27*10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?
- Another algorithms → FP Tree and GROWTH

FP-Growth vs. Apriori: Scalability With the Support Threshold



Beyond Binary Data

- Hierarchies
  - drink → milk → low-fat milk → Stop&Shop low-fat milk …
  - find associations on any level
- Sequences over time
- …

Multiple-level Association Rules

Figure 1: A taxonomy for the relevant data items

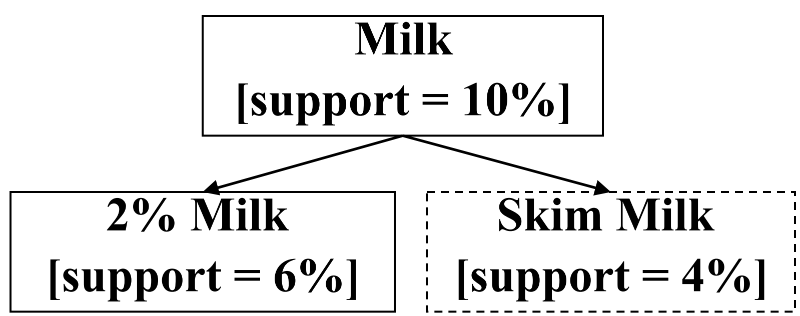
- Items often form hierarchy
- Flexible support settings:
  Items at the lower level are expected to have lower support.
- Transaction database can be encoded based on dimensions and levels
- explore shared multi-level mining

uniform support

reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 5%



Level 1
min_sup = 5%

Level 2
min_sup = 3%

Multi-level Association Rules

- Why should we incorporate concept hierarchy?
  - Rules at lower levels may not have enough support to appear in any frequent itemsets

  - Rules at lower levels of the hierarchy are overly specific
    - e.g., skim milk $\rightarrow$ white bread, 2% milk $\rightarrow$ wheat bread, skim milk $\rightarrow$ wheat bread, etc.

    are indicative of association between milk and bread

Quantitative Association Rules

ID	Age	Salary	Maritial Status	NumCars
100	44	30 000	married	2
200	55	45 000	married	3
300	45	50 000	divorced	1
400	34	44 000	single	0
500	45	38 000	married	2
600	33	44 000	single	2

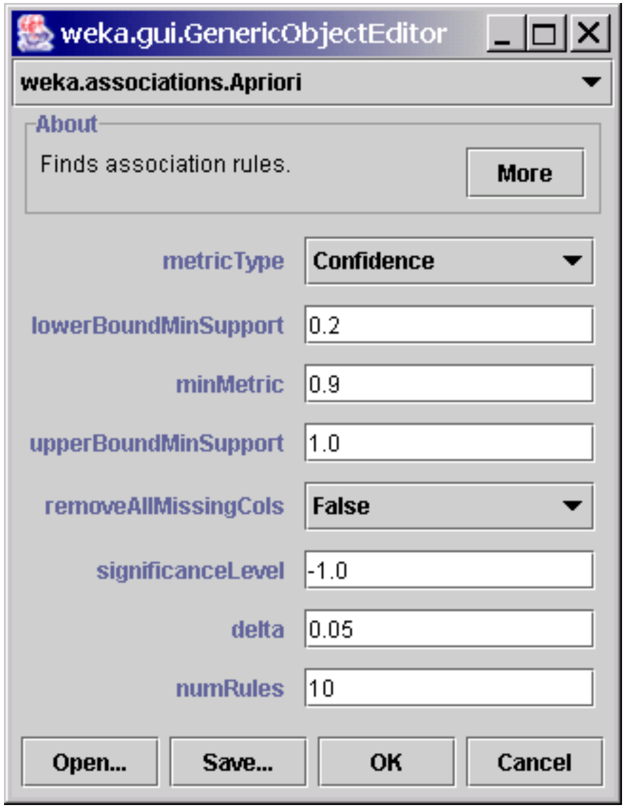
Sample Rules	Support	Confidence
<age:44..55> and < status: married> ==> <numCars:2>	50%	100%
<NumCars: 0..1> ==> <Married: No>	33%	66,70%

Handling Continuous Attributes

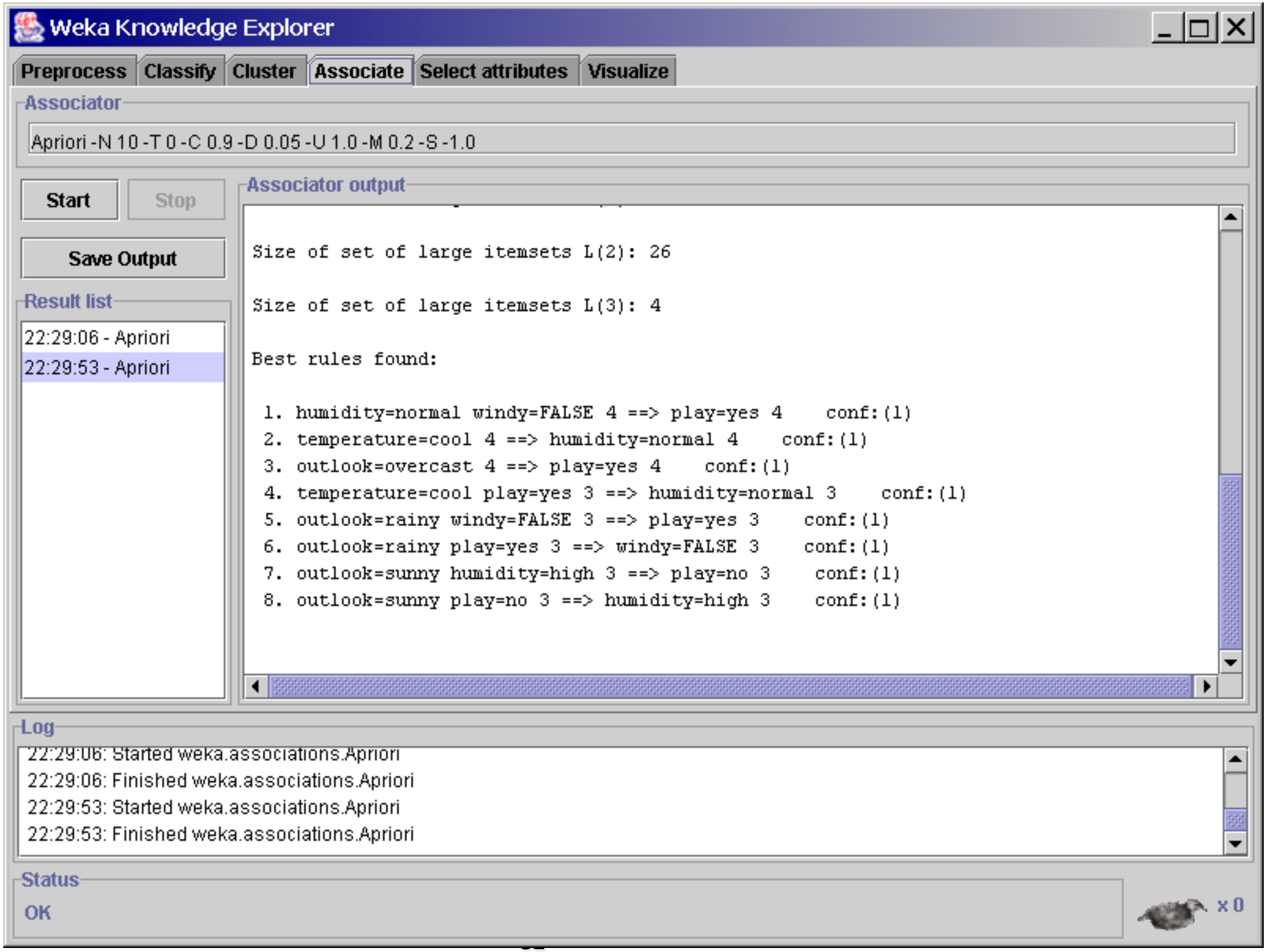
- Different kinds of rules:
  - $Age \in [21,35) \wedge Salary \in [70k,120k) \rightarrow Buy$
  - $Salary \in [70k,120k) \wedge Buy \rightarrow Age: \mu=28, \sigma=4$
- Different methods:
  - Discretization-based
  - Statistics-based
  - Non-discretization based
    - minApriori

Weka associations

File: weather.nominal.arff
MinSupport: 0.2



Weka associations: output



Case study of using association rules

- See D.Larose: Discovering Knowledge in Data.
  - Analyse the description of discovering interesting association rules from legal databases (Australia analysis of problems with immigrates) – chapter 1
  - You can also study smaller cases in chapter 10

References: Apriori and related …

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- See others …

Any questions, remarks?

?