Frequent Patterns

Nicolas PASQUIER

Laboratoire I3S (UMR-7271 UNS/CNRS)

Université Nice Sophia-Antipolis

http://www.i3s.unice.fr/~pasquier

mailto:nicolas.pasquier@unice.fr

Association Rules

- Directed links of association (co-occurrence) between two sets of variable values expressing causality
- Example: Market basket data
  - Buy:Cereal ∧ Buy:Sugar → Buy:Milk (support=10%, confidence=50%)
- Support: Weight (scope) of the rule
  - Proportion of objects (instances/tuples) containing all items
  - 10% of all customers have bought both three items
- Confidence: Precision (reliability) of the rule
  - Proportion of objects containing the consequent among those containing the antecedent
  - 50% of customers having bought cereal and sugar also have bought milk

Data Mining Context

- Dataset: Finite binary relation **R** ⊆ **O** x **I**
  - **O**: Finite set of objects
  - **I**: Finite set of attribute values (items)
- Example dataset **D**

Transactions

OID	Items
1	ABCE
2	BCE
3	ACD
4	ABCE
5	BCE

Equivalent ⟷ representations

Binary Matrix

OID	A	B	C	D	E
1	1	1	1	0	1
2	0	1	1	0	1
3	1	0	1	1	0
4	1	1	1	0	1
5	0	1	1	0	1

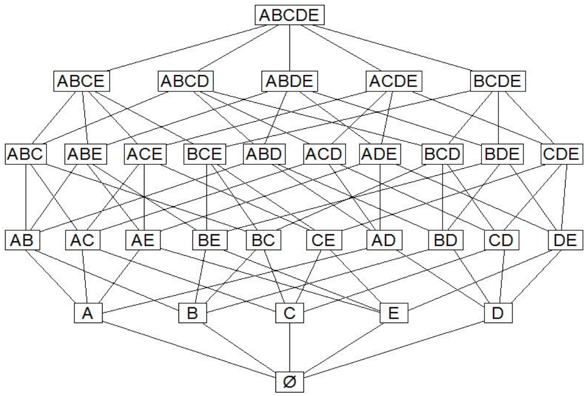
- Items are ordered (e.g., lexicographic order)

Itemsets and Itemset Support

- Itemset
  - A set of items
  - Ex: {A}, {B}, {AC}, {ABC} in **D**
- K-itemset
  - A set of k-items
  - Ex: {A} and {B} are 1-itemsets, {ABC} is a 3-itemset
- Support of an itemset
  - Proportion of objects containing the itemset
  - support(L) = COUNT(L) / COUNT()
  - Ex: support({AC}) = COUNT({AC}) / COUNT() = |{1,3,4}| / 5 = 3/5

Itemset Lattice

- The search space is the *itemset lattice* or *subset lattice*
- Its size is exponential in the number of items: $2^{|I|}$



Association Rule Support and Confidence

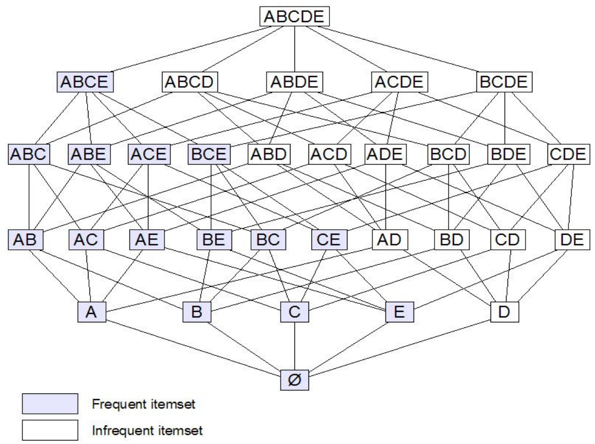
- <u>Association rule</u>
  - Implication rule between two distinct itemsets
    R = LHS → RHS, support(R), confidence(R)
- support(R) = COUNT(LHS ∪ RHS) / COUNT()
  - Proportion of objects "containing" the rule
  - Ex: support(A → B) = COUNT({AB}) / COUNT()
    = |{1,4}| / 5 = 2/5
  - Interpretation: Scope (coverage) of the rule in the dataset
- confidence(R) = COUNT(LHS ∪ RHS) / COUNT(LHS)
  - Proportion of objects "verifying" the rule
  - Ex: confidence(A → B) = COUNT({AB}) / COUNT({A})
    = |{1,4}| / |{1,3,4}| = 2/3
  - Interpretation: Precision (accuracy) of the rule in the dataset

Minimum Support and Confidence

- An association rule R is considered useful and significant iff:
  - Sufficiently frequent in the dataset
    support(R) ≥ *minsupport*
  - Sufficiently precise in the dataset
    confidence(R) ≥ *minconfidence*
- *minsupport* and *minconfidence* are user defined thresholds
  - Values depend on the application context and objectives
- An itemset L is <u>frequent</u> if support(L) ≥ *minsupport*
- Association rules are generated from frequent itemsets

Frequent Itemsets

- *minsupport* = 2/5



Association Rule Mining

- First approach (Apriori algorithm [1])
  1. Extract all frequent itemsets and their support (support ≥ *minsupport*)
  2. Generate all valid association rules (confidence ≥ *minconfidence*)
- Based on the following properties
  i. All supersets of an infrequent itemset are infrequent
  - Ex: ABD infrequent ⇒ ABCD, ABDE, ABCDE infrequent

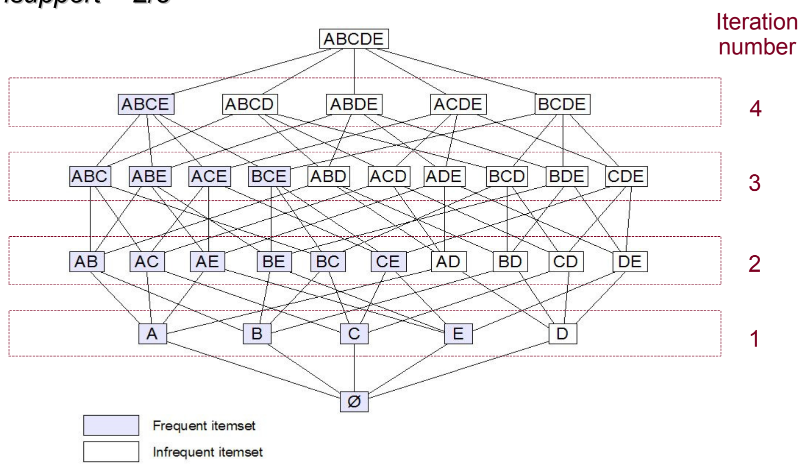
  ii. All subsets of a frequent itemset are frequent
  - Ex: ABC frequent ⇒ A, B, C, AB, AC, BC frequent

Association Rule Mining: Apriori

- Iterative pruning the itemset lattice
  - Bottom-up level-wise traversal of the itemset lattice
- First iteration:
  - Frequent 1-itemsets (i.e., items) are extracted
- Subsequent iteration k:
  - Candidate k-itemsets are generated from frequent (k-1)-itemsets
  - Supports of candidate k-itemsets are extracted from the dataset
  - Infrequent candidate k-itemsets are discarded

Itemset Lattice Pruning

- *minsupport* = 2/5



Apriori Algorithm

- C_k: Candidate k-itemsets
- L_k: frequent k-itemsets

```
L_1 = {frequent 1-itemsets};
for (k ← 1; L_k != Ø; k++)
    C_{k+1} ← AprioriGen(L_k)
    for each object o in dataset D do
        for each candidate c in C_{k+1} do
            if (c ⊆ o) then c.support++
        endfor
    endfor
    L_{k+1} ← candidates in C_{k+1} with support ≥ minsupport
endfor
return ∪_k L_k
```

AprioriGen Function

- **Step 1**: Self-joining frequent k-itemsets in L_k
  to generate k+1-candidates (SQL syntax)

1. insert into C_{k+1}
2. select $P_1, P_2, \ldots, P_k, Q_k$
3. from L_k P, L_k Q
4. where $P_1 = Q_1$ and … and $P_{k-1} = Q_{k-1}$ and $P_k < Q_k$

- **Step 2**: Pruning invalid candidates

1. <u>for each</u> itemset c <u>in</u> C_{k+1} <u>do</u>
2. <u>for each</u> k-subsets s <u>of</u> c <u>do</u>
3. <u>if</u> ($s \notin L_k$) <u>then</u> <u>delete</u> c <u>from</u> C_{k+1}
4. <u>endfor</u>
5. <u>endfor</u>

AprioriGen Function Example

- Generating candidate 4-itemsets from set L_3 of frequent 3-itemsets
  L_3 = {ABC, ABD, ACD, ACE, BCD}
- Step 1: Join frequent 3-itemsets with the same prefix
  - <u>AB</u>C and <u>AB</u>D : ABCD
  - <u>AC</u>D and <u>AC</u>E : ACDE
  - C_4 = {ABCD, ACDE}
- Step 2: Pruning invalid 4-candidates with infrequent 3-subsets
  - ABCD: Valid candidate since ABC, ABD, ACD, BCD in L_3
  - ACDE: Invalid candidate since ADE is infrequent
- L_4 = {ABCD}

Apriori Algorithm Example

- Dataset **D**
- *minsupport* = 2/5

C_1

itemset
A
B
C
D
E

scan **D** →

C_1

itemset	support
A	3/5
B	4/5
C	5/5
D	1/5
E	4/5

pruning infrequent →

L_1

itemset	support
A	3/5
B	4/5
C	5/5
E	4/5

C_2

itemset
AB
AC
AE
BC
BE
CE

scan **D** →

C_2

itemset	support
AB	2/5
AC	3/5
AE	2/5
BC	4/5
BE	4/5
CE	4/5

pruning infrequent →

L_2

itemset	support
AB	2/5
AC	3/5
AE	2/5
BC	4/5
BE	4/5
CE	4/5

C_3

itemset
ABC
ABE
ACE
BCE

scan **D** →

C_3

itemset	support
ABC	2/5
ABE	2/5
ACE	2/5
BCE	4/5

pruning infrequent →

L_3

itemset	support
ABC	2/5
ABE	2/5
ACE	2/5
BCE	4/5

C_4

itemset
ABCE

scan **D** →

C_4

itemset	support
ABCE	2/5

pruning infrequent →

L_4

itemset	support
ABCE	2/5

Generating Association Rules

- *minconfidence* = 2/3

Itemsets are processed in decreasing order of their size

L_4

itemset	support
ABCE	2/5

1 →

L_3 — Valid rules

itemset	support	Rule	Confidence
ABC	2/5	ABC → E	2/2
ABE	2/5	ABE → C	2/2
ACE	2/5	ACE → B	2/2
BCE	3/5	BCE → A	2/3

2 →

L_2 — Valid rules

itemset	support	Rule	Confidence
AB	2/5	AB → CE	2/2
AC	3/5	AC → BE	2/3
AE	2/5	AE → BC	2/2
BC	4/5	~~BC → AE~~	2/4
BE	4/5	~~BE → AC~~	2/4
CE	4/5	~~CE → AB~~	2/4

3 →

L_1 — Valid rules

itemset	support	Rule	Confidence
A	3/5	A → BCE	2/3
~~B~~	~~4/5~~		
~~C~~	~~5/5~~		
~~E~~	~~4/5~~		

L_3

…

Subsets are processed in decreasing order of their size

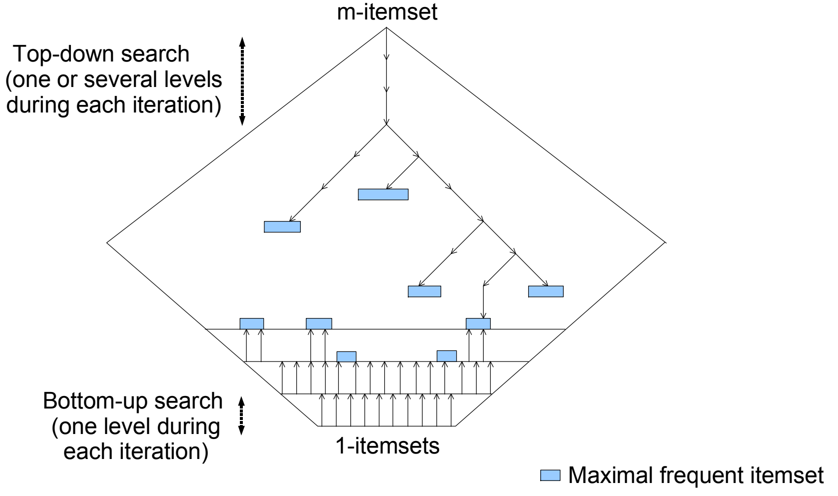
Maximal Frequent Itemset Approach

- A maximal frequent itemset is a frequent itemset with no frequent superset (maximality vs. inclusion)
- Approach based on the following property:
  - Maximal frequent itemsets define a border under which all itemsets are frequent and above which all itemsets are infrequent
- All frequent itemsets can be deduced from the maximal frequent itemsets, but their support must be extracted from the dataset
- Second approach (Max-Miner [2] and Pincer-Search [3] algorithms)
  1. Extract all maximal frequent itemsets (dataset scans)
  2. Derive frequent itemsets and extract their support (dataset scan)
  3. Generate all valid association rules

Maximal Frequent Itemsets

- Iterative approach
  - Simultaneous top-down and bottom-up level-wise traversals of the itemset lattice
  - Two sets of candidates:
    a) Candidate frequent itemsets (bottom-up)
    b) Candidate maximal frequent itemsets (top-down)
- During each iteration
  - One scan of the dataset
  - Reduce the search space of one level bottom-up
  - Reduce the search space of one or several levels top-down
  - Subsets of discovered maximal frequent itemsets are deleted from bottom-up traversal candidates

Maximal Frequent Itemset Extraction



Association Rule Mining Problems

- Dense and correlated data constitute a challenge for extracting association rules
- Problem of execution times
  - Execution times of several hours most often (sometimes several days!)
  - Datasets are large (cannot fit in main memory)
  - The dataset must be scanned (sequential read of all instances) several times during the process
- Problem of relevance of extracted association rules
  - Several tens of thousands of rules extracted (sometimes millions)
  - Among these rules many are redundant (i.e., represent the same information)

Galois Connection

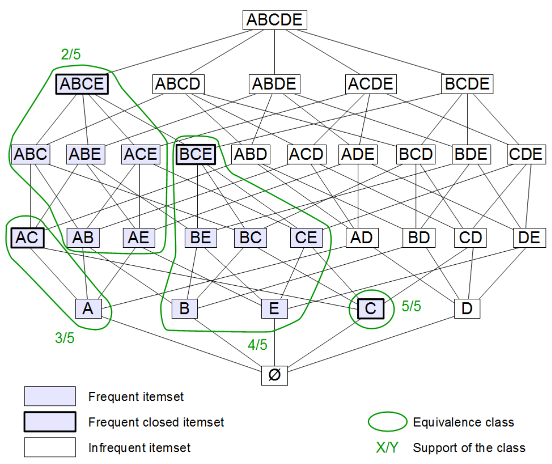
- Let $I \subseteq \mathbf{I}$ and $O \subseteq \mathbf{O}$
- Galois connection of a finite binary relation
  - f(O): Items common to all objects $o \in O$
    - $f(O): 2^{\mathbf{O}} \rightarrow 2^{\mathbf{I}}$
    - $f(O) = \{i \in I \mid \forall o \in O, (o,i) \in \mathbf{R}\}$
  - g(I): Objects in relation with all items $i \in I$
    - $g(I): 2^{\mathbf{I}} \rightarrow 2^{\mathbf{O}}$
    - $g(I) = \{o \in O \mid \forall i \in I, (o,i) \in \mathbf{R}\}$
- Closure operator of the Galois connection: $h = f \circ g$
  - Extension: $I \subseteq h(I)$
  - Idempotency: $h(h(I)) = h(I)$
  - Monotonicity: $I_1 \subseteq I_2 \Rightarrow h(I_1) \subseteq h(I_2)$

Frequent Closed Itemsets

- The Galois closure of an itemset I is computed by intersecting all objects containing I
  - Ex : h(BC) = *intersection*(*objects*(BC)) = BCE
- A <u>closed itemset</u> is an itemset that is equal to its closure: I = h(I)
  - Closed itemset: Maximal set of items common to a set of objects
  - Examples:
    - BC is not closed since *objects*(BC) = {1,2,4,5} but *intersection*({1,2,4,5}) = BCE
    - BCE is a closed itemset since *intersection*(*objects*(BCE)) = BCE
    - C is a closed itemset since *intersection*(*objects*(C)) = C
- A <u>frequent closed itemset</u> is a closed itemset that is frequent

Frequent Closed Itemsets

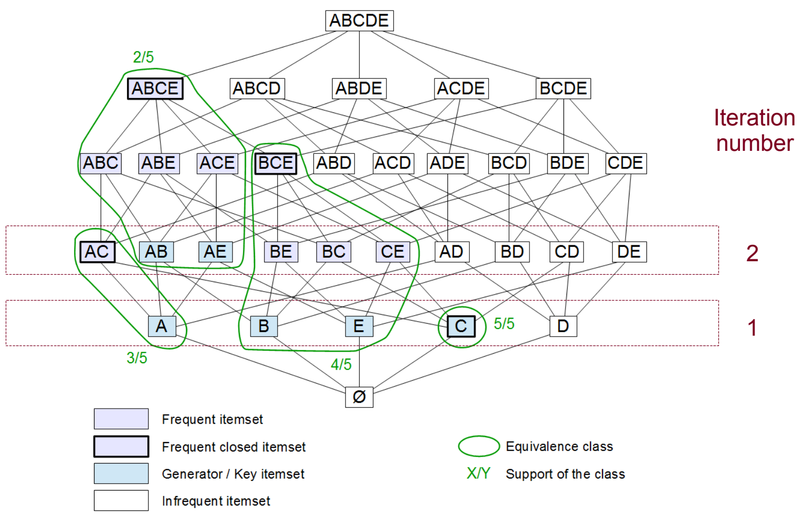
- *minsupport* = 2/5



Galois Closure based Approach

- Approach based on the following properties:
  - i. Maximal frequent itemsets are maximal frequent closed itemsets
  - ii. The support of an itemset is equal to the support of its closure
- Third approach (Close algorithm [4])
  1. Extract all frequent closed itemsets (dataset scans)
  2. Derive frequent itemsets and their support
  3. Generate all valid association rules
- Pruning the closed itemset lattice
  - Bottom-up level-wise traversal of the itemset lattice
  - Iteration k: The support and the closure of all candidate k-itemsets are extracted simultaneously

Frequent Closed Itemset Mining



Close Algorithm

- CC_k: Candidate k-itemsets (fields: generator, support, closure)
- E_k: Frequent equivalence classes (fields: generator, support, closure)

```
CC1.generators ← {1-itemsets}
for (k ← 2; CCk != Ø; k++)
    for each object o in dataset D do
        for each candidate c in CCk do
            if (c.generator ⊆ o) then
                c.support++
                if (c.closure = Ø) then c.closure ← o
                else c.closure ← c.closure ∩ o
            endif
        endfor
    endfor
    Ek ← candidates in CCk with support ≥ minsupport
    CCk ← GenGenerators(Ek)
endfor
return ∪k Ek
```


GenGenerators Function

- **Step 1**: Self-joining frequent itemsets in E_k (SQL syntax)

1. insert into CC_{k+1}.generator
2. select $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$
3. from E_k p, E_k q
4. where $p.item_1 = q.item_1, \ldots, p.item_{k-1} = q.item_{k-1}, p.item_k < q.item_k$

- **Step 2**: Pruning invalid candidates

1. for each candidate generator c in CC_{k+1}.generator do
2.   for each k-subsets s of c do
3.   if ($s \notin E_k$.generators) then delete c from CC_{k+1}
4.   elsif ($c \subseteq$ s.closure) then delete c from CC_{k+1}
5.   endfor
6. endfor

Close Algorithm Example

- Dataset **D**
- *minsupport* = 2/5

CC_1

generator	closure	support
A	Ø	0
B	Ø	0
C	Ø	0
D	Ø	0
E	Ø	0

scan **D** →

CC_1

generator	closure	support
A	AC	3/5
B	BCE	4/5
C	C	5/5
D	ACD	1/6
E	BCE	4/5

pruning infrequent →

E_1

generator	closure	support
A	AC	3/5
B	BCE	4/5
C	C	5/5
E	BCE	4/5

CC_2

generator	closure	support
AB	Ø	0
AE	Ø	0

scan **D** →

CC_2

generator	closure	support
AB	ABCE	2/5
AE	ABCE	2/5

pruning infrequent →

E_2

generator	closure	support
AB	ABCE	2/5
AE	ABCE	2/5

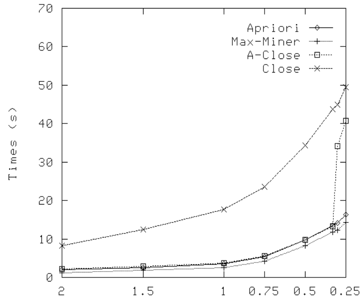
Hybrid Approaches

- A-Close
  - All frequent generators are found using the Apriori approach
  - Closures of generators that are not closed are computed in the last phase
  - More efficient than Close for weakly correlated data, less efficient for dense and highly correlated data
  - Less efficient than Apriori for weakly correlated data, more efficient for dense and highly correlated data
- Pascal
  - *Itemset Counting Inference* in Apriori
  - Generators and closed itemsets are identified among frequent itemsets by comparing supports of subsets and super-sets

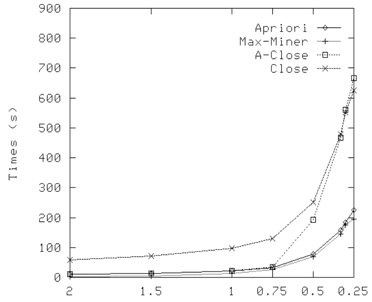
Experimental Results: Execution Times

- Synthetic market basket data (Almaden IBM Research Center)
- Sparse and weakly correlated data:
  - Low execution times
  - Closure computation is inefficient (increase number of operations)

T10I4D100K



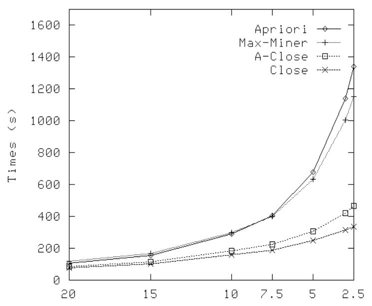
T20I6D100K



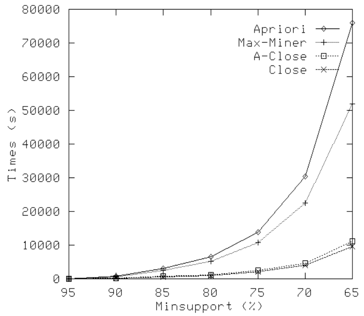
Experimental Results: Execution Times

- Census dataset (UCI Machine Learning Repository) benchmark
- Dense and correlated data:
  - Important execution times
  - Galois Closure usage reduces the number of candidates

C20D10K



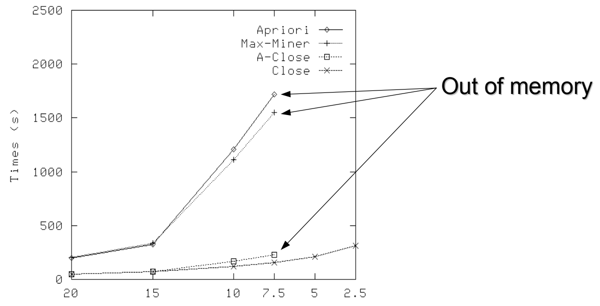
C73D10K



Experimental Results: Execution Times

- Mushrooms dataset (UCI Machine Learning Repository) benchmark: Physiological characters of 8046 mushrooms
- Galois Closure use reduces the number of candidates
- Lower memory usage than itemset based approaches

Mushrooms

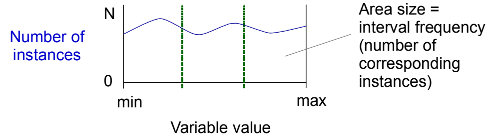


Numerical Continuous Variables

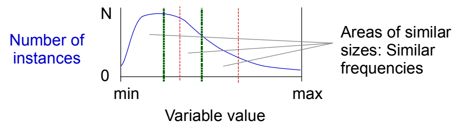
- Discretization: Transformation in ordinal values by intervals
  - Ex: Discretize variable Age ∈ [16,60] in 3 intervals of same width

    Variable values are replaced in the data matrix by the corresponding interval: Age=[16,30], Age=[31,45], Age=[46-60]

- Equal-width discretization: Intervals are defined such that they all have an identical width

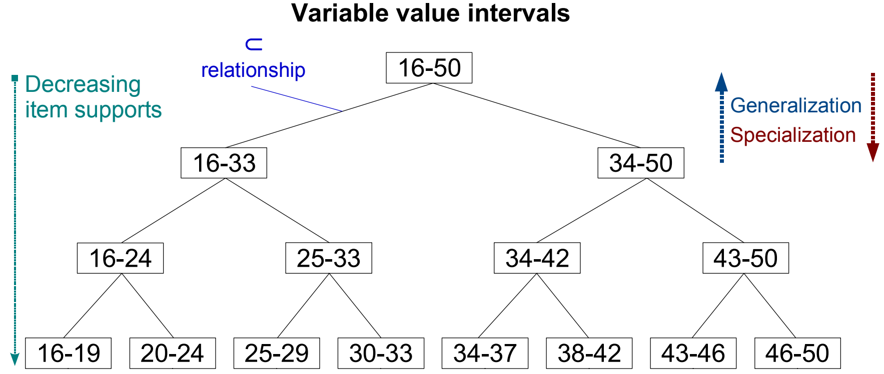


- Equal-frequency (quantiles): Intervals are defined such that they all correspond to the same number of instances (as far as possible)



Hierarchical Discretization

- Hierarchical decomposition of intervals (taxonomy)



- Note: The smallest support value of an item in the dataset defines a constraint (maximal value) on the *minsupport* threshold value

Statistical Measures

- A correlation statistical measure (software dependent) is often computed for each association rule to assess its statistical validity
- Lift: Statistical correlation between LHS and RHS
  - Lift(R: LHS → RHS) = P(LHS ∪ RHS) / P(LHS) . P(RHS) ∈ [0,+∞]
  - Lift < 1: negative correlation
  - Lift = 1: independence
  - Lift > 1: Positive correlation
- Conviction: Takes into account the absence of RHS
  - Lift(R: LHS → RHS) = P(LHS) . P(¬RHS) / P(LHS ∪ ¬RHS) ∈ [0,+∞]
  - Conviction < 1: negative correlation
  - Conviction = 1: independence
  - Conviction > 1: Positive correlation
- Both can be used to filter useless rules (i.e., rules with measure ≤ 1)

References

[1] R. Agrawal, R. Srikant. *Fast Algorithms for Mining Association Rules in Large Databases*. Proceedings VLDB international conference, 1994.

[2] R. J. Bayardo. *Efficiently Mining Long Patterns from Databases*. Proceedings SIGMOD international conference, 1998.

[3] D.I. Lin, Z.M. Kedem. *Pincer Search: A New Algorithm for Discovering the Maximum Frequent Set*. Proceedings EDBT international conference, 1998.

[4] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal. *Efficient Mining of Association Rules using Closed Itemset Lattices*. Information Systems, 24(1):25-46, Elsevier, 1999.

[5] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal. *Computing Iceberg Concept Lattices with Titanic*. Data and Knowledge Engineering, 42(2):189-222, Elsevier, 2002

[6] J. Han, H. Cheng, D. Xin, X. Yan. *Frequent Pattern Mining: Current Status and Future Directions*. Data Mining and Knowledge Discovery, 15:55-86, 2007.