Managing Big Data

Association Analysis: Basic Concepts and Algorithms

Manolis Tzagarakis
Assistant Professor
Department of Economics
University of Patras

tzagara@upatras.gr
2610 969845
google:tzagara
Facebook: tzagara
SkypeID: tzagara
QuakeLive: DeusEx

Association analysis

- What is association analysis?
  - The task of **analyzing** so called **"transactions"** that **indicate the likely occurrence of an item** based **on the occurrences of other items** in the transactions of large datasets
  - The discovered **relationships** are represented in the **form of association rules**

Association analysis

- Where is it used?
  - **Biology and bioinformatics**
    - E.g. Co-occurrence of genes
  - **Medicine**
    - Occurrence of symptoms
  - **Geology**
    - Relationships between oceans and land masses
  - **Retail**
    - Market basket analysis

Association analysis – Main idea

- Main idea exemplified

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

Market-Basket transactions i.e. what customers have bought

Transactions i.e. what a customer bought in the supermarket

The <u>main task</u> is now to analyze transactions and come up with association rules of the form

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

These mean that e.g. <u>customers that bought Diapers also bought Beer</u>, customer that bought Beer and Bread also bought Milk.

Association analysis – Main idea

- **Association rules**
  - Rule suggest that a **strong relationship** exist between items of transaction
    - Form of rules: **Antecedent → Consequent**
    - Note: rules implies **relationship/co-occurrence** **not causality!**
  - **Practical issues**: Helps in devising sale strategies and discounts
    - Used heavily by retailers to identify opportunities of cross-selling to customers

Association analysis – Main idea

- Practical issues
  - Let a discovered rule be as follows:
    - **{Bagels,…} → {Potato Chips}**
  - Potato Chips as **<u>consequent</u>**: what should be done to boost its sales
  - Bagels in **<u>antecedent</u>**: can be used to see which products will be affected if the store discontinues selling bagels
  - Bagels in antecedent and Potato chips in consequent: can be used to see what products should be sold with bagels to promote sell of potato chips

Association analysis

- What **problems** exist when trying **to find associations and rules** in transactions?
  - When **number of transactions is huge** finding such rules is **computational expensive**
    - True even for small/midsized supermarkets
  - Some **rules** may be **accidental** or **no rules at all (i.e. simply false)**

Basic Concepts – problem definition

Basic concepts

One item

Transactions database

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

One transaction

- **Items**
  - A finite set of atomic elements $I = \{ i_1, i_2, i_3, \ldots, i_d \}$ e.g. {milk, beer, diapers, bagel}
- **Transaction t**
  - is a subset of I, i.e. $\subseteq$ I which is observed
  - Transaction usually have IDs (see column TID in the table)
- **Transaction Database**
  - A set of transactions $T=\{t_1, t_2, t_3, \ldots, t_n\}$
- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - **k-itemset**
    - An itemset that contains k items e.g. 3-itemset: {Milk, Beer, Bagel}, 2-itemset: {Diaper, Milk}
  - Important: itemsets different from transactions
- We say that a **transaction t contains itemset X when X $\subseteq$ t.**

Basic concepts

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 σ and s:

σ({Milk, Bread, Diaper}) = 2
s({Milk, Bread, Diaper}) = 2/5

Note: order of items in itemsets does not matter. **E.g. σ({Bread, Milk, Diaper}) = 2**

- Metrics for itemsets
  - **Support count of itemset, σ**
    - Frequency of occurrence of an itemset
  - **Support of itemset, s**
    - Fraction (pct) of transactions that contain an itemset

Basic concepts

- What are **association rules?**
  - An **association rule** is an implication of the form:

    $X \rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \emptyset$
  - Examples of valid rules
    - **{Milk, Beer} → {Diapers}**
    - **{Beer, Bagel} → {Milk, Diapers, Potato chips}**
  - Examples **of invalid rules**
    - **{Beer, Bagel} → {Beer}** (violates $X \cap Y = \emptyset$)

Basic concepts

- Metrics for association rules
  - **<u>Support</u> of association rule X → Y**
    - Fraction of transactions that contain both X and Y:

$$support, s(X \rightarrow Y) = \frac{\sigma(X \cap Y)}{N}$$

  - **<u>Confidence</u> of association rule X → Y**
    - Fraction of transactions in which every time there is X, there also is Y:

$$confidence, c(X \rightarrow Y) = \frac{\sigma(X \cap Y)}{\sigma(X)}$$

Basic concepts

- Example of rule metrics

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**

- Assume rule
  - **{Milk, Diaper} → Beer**

Support of rule {Milk, Diaper} → Beer:

$$s(\{milk, diaper\} \rightarrow Beer\}) = \frac{\sigma(\{milk, diaper, beer\})}{|T|} = \frac{2}{5} = 0.4$$

Confidence of rule {Milk, Diaper} → Beer:

$$c(\{milk, diaper\} \rightarrow Beer\}) = \frac{\sigma(\{milk, diaper, beer\})}{\sigma(\{milk, diaper\})} = \frac{2}{3} = 0.67$$

Basic concepts

- Problem statement
  - Given a set of transactions T, the goal of association rule mining is to **find all rules** having
    - **support ≥ *minsup* threshold**
    - **confidence ≥ *minconf* threshold**
  - **Note: *minsup, minconf* user specified**. E.g. minsup = 0.6, minconf = 0.9 given as input

Basic concepts

- How to find such rules?
- One solution: **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
- Is brute force a good solution?
  - No! Computationally prohibitive!
    - Exponential complextity!

Basic concepts

- Observations helping in improving the situation

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} $\rightarrow$ {Beer} (s=0.4, c=0.67)
{Milk,Beer} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Beer} $\rightarrow$ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk,Beer} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper,Beer} (s=0.4, c=0.5)

Some 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 !

Basic concepts

- Use this to derive a two-step approach for finding rules:
  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. Such rules are called **strong rules**.
- **Step 1 i.e. Frequent itemset generation** is still computationally expensive

Basic concepts

- The problem now becomes:
  - **How to solve step 1 i.e. How to find all frequent itemsets?**
    - How easy is it given a set of transactions to find all frequent itemsets?

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

The problem?

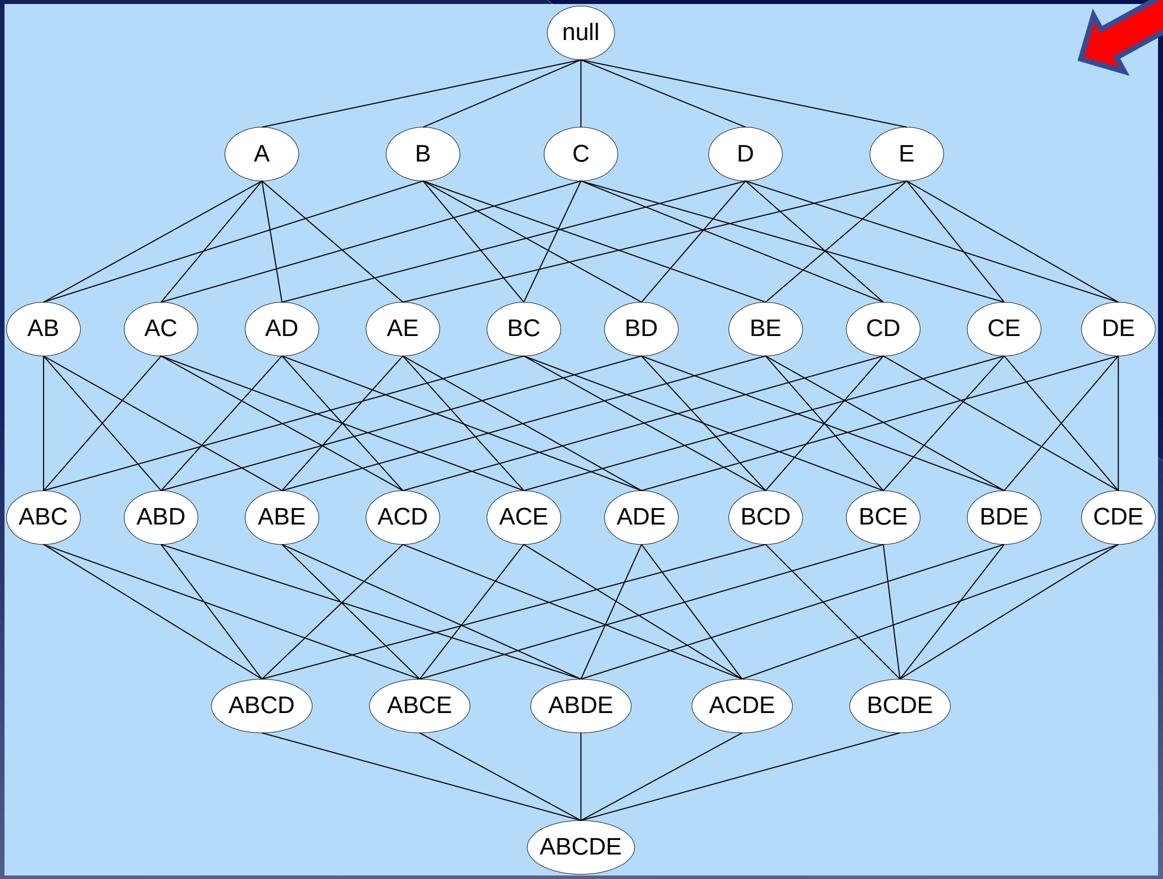
How to find that e.g. {Bread, Beer} is a frequent itemset i.e. above a threshold (minsup)

Look at all the combinations that you have to check!

Basic concepts

One way of dealing with finding the frequent itemsets is the <u>Brute force approach</u>: List all possible itemsets, called candidate itemsets

Candidate itemset lattice: All itemsets generated from 5 items

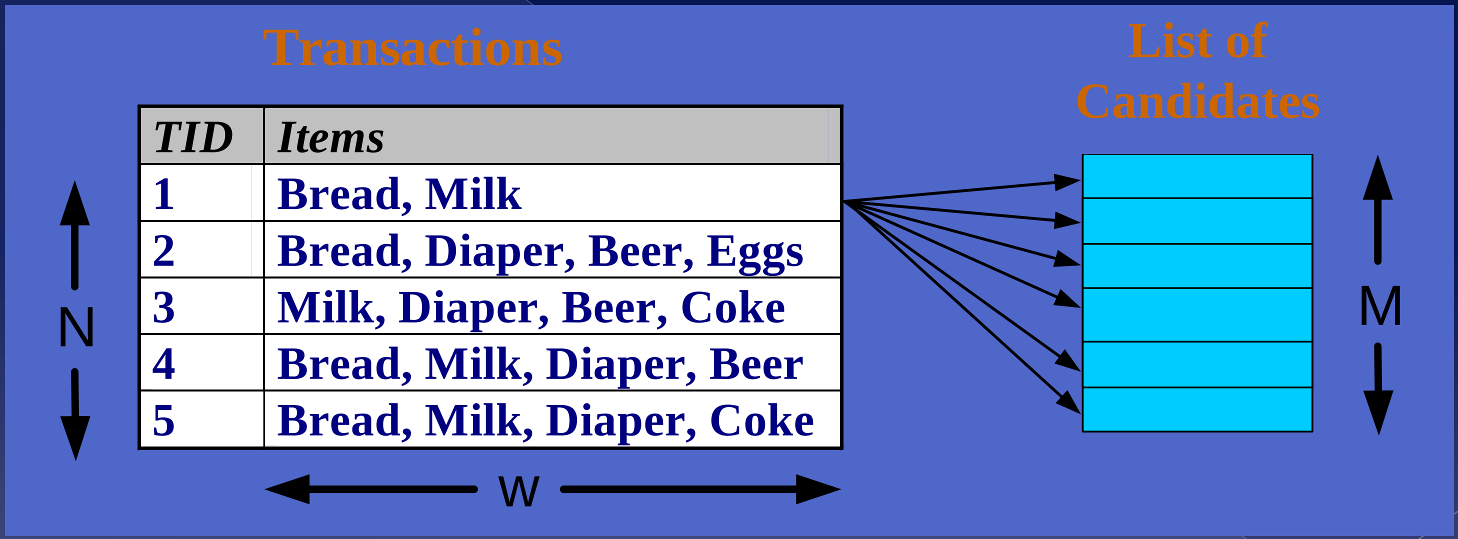


Finding frequent itemsets not easy. Still computationally expensive:

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

Basic concepts

- **Brute-force approach <u>for finding frequent itemsets:</u>**
  - Each itemset in the lattice is a **candidate** frequent itemset
  - Count the support of each candidate by scanning the database

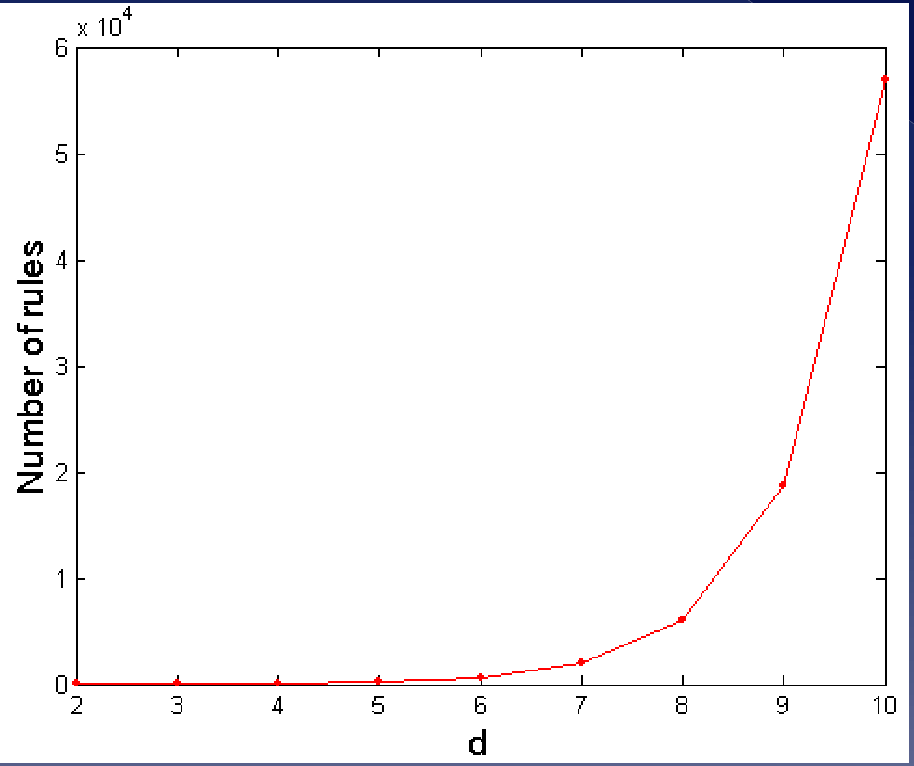


TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate
  - Complexity ~ **O(NMw)**, where **<u>M = 2^d -1 and w the maximum with of transaction => expensive!!!</u>**

Basic concepts

- Given **d unique items**:
  - **Total number of itemsets = 2^d**
  - **Total number of possible association rules, R:**



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$$

This means with d=6 items you can generate R=602 different rules !

Basic concepts

- How to conquer this complexity **in finding the frequent itemsets** ?
  - Reduce the **number of candidate itemsets (M)**
    - Complete search: $M=2^d$
    - Use pruning techniques to reduce M
  - Reduce the **number of transactions (N)**
    - Reduce size of N as the size of itemset increases
    - Used by DHP and vertical-based mining algorithms
  - Reduce the **number of comparisons (NM)**
    - Use efficient data structures to store the candidates or transactions
    - No need to match every candidate against every transaction

Apriori principle

- **Reduce** number of candidates **based on itemset support**
  - Prune/ignore itemsets with support lower than a threshold
  - To do this, use the **apriori principle** which allows to **“automatically” prune/ignore some itemsets**

Apriori principle

- Apriori principle
  - **"If an itemset is frequent, then all of its subsets must also be frequent"**
  - Or equivalently "**if itemset not frequent, it's supersets won't be frequent either**"

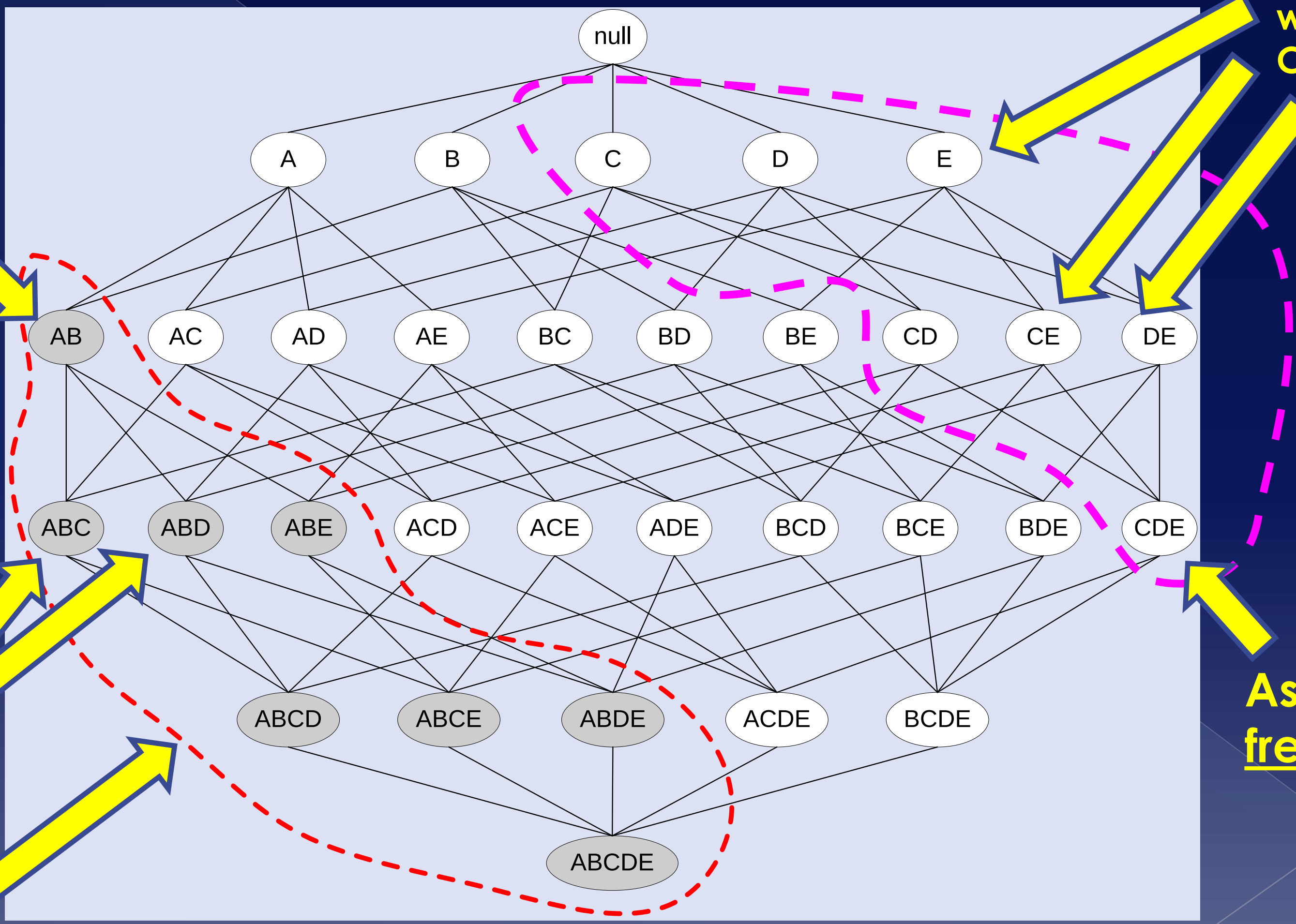
Apriori principle illustrated

Assume AB not frequent

If AB not frequent then also not frequent: ABC, ABD, ABE, ABCD, …

(all supersets)

Can prune/ignore these itemsets.



If CDE frequent, then also frequent will be subsets: DE, CE, CD, C, D, E.

Candidate frequent itemsets

Assume CDE frequent

Apriori principle

- **Apriori principle allows the pruning** of an exponential search space (itemset lattice) based on support
  - Hence **called support-based pruning**
- **Support-based pruning possible** due to an important property of the **support measure**: **the anti-monotone property**
  - The Anti-monotone property: **support of an itemset never exceeds the support of its subsets**

Apriori principle

- **Monotone/anti-monotone** property more formally defined
  - Assume I a set of items and J = 2^I its powerset. A **measure f is said to be monotone or upward closed if**:

$$\forall X, Y \in J: (X \subseteq Y) \rightarrow f(X) \leq f(Y)$$

  - Measure **f is said to be anti-monotone or downward-closed if**

$$\forall X, Y \in J: (X \subseteq Y) \rightarrow f(X) \geq f(Y)$$

Apriori principle

- In general, **every measure that has the anti-monotone property** can be integrated into algorithms and **used to prune** the exponential search space of candidate itemsets

Apriori algorithm

Apriori algorithm

- **Apriori algorithm** uses the **apriori principle (support-based pruning)** to **find frequent itemsets**
- The Apriori algorithm
  - Best known algorithms of this category
  - Very good results
  - Used today in many application domains

Apriori algorithm

- Apriori psudocode

Assume
C_k : Candidate itemsets of size k (i.e. k-itemsets)
L_k : Frequent itemsets of size k (k-itemsets)
minsup: minimum support count, given

```
1. L_1 = {frequent 1-itemsets} /* 1-itemsets with support >= minsup */
2. for (k=1; L_k != Ø; k++) do begin
3.       C_{k+1} = generate candidates from L_k /* gen. k+1-itemsets */
4.       for each transaction t in Database do
5.           increment support count for all candidate itemset in C_{k+1}
             found in t
6.       L_{k+1} = all candidates in C_{k+1} with at least minsup support
              (i.e. prune/ignore all candidates in C_{k+1} with support <
             minsup)
   end
7.  return ∪_k L_k /* List of all frequent itemsets */
```

Apriori algorithm: example

TID	Transactions
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2 ,I3, I5
9	I1, I2, I3

- Database with **9 transactions**
- Assume **minimum support required - minsup = 2 (i.e. 2/9 = 22%)**
- Applying the **Apriori** algorithm to find **frequent itemsets**
- **List of items = {I1, I2, I3, I4, I5}**

Apriori algorithm: example

- **Step 1:** find frequent 1-itemsets

Itemset	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1 : Candidate frequent 1-itemsets

Scan candidate 1-itemsets C_1 and prune/remove all itemsets having support count < minsup (=2). Will generate L_1

Itemset	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1 : Frequent 1-itemsets

L_1 generated by removing all itemsets in C_1 having support count < minsup (=2)

Apriori algorithm: example

- **Step 2:** find frequent 2-itemsets generated from L_1

Generate C_2 from L_1

Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2 : Candidate frequent 2-itemsets

Scan candidate 2-itemsets C_2 and remove all itemsets having support count < minsup (=2). This generates L_2

Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
~~{I1, I4}~~	~~1~~
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
~~{I3, I4}~~	~~0~~
~~{I3, I5}~~	~~1~~
~~{I4, I5}~~	~~0~~

L_2 : frequent 2-itemsets after pruning C_2

C_2 is produced by **joining/concatenating itemsets of size 2 from L_1 that generate 3-itemsets. Note: Apriori principle still not used!**

Apriori algorithm: example

- **Notes on step 2**
  - How to join 1-itemsets to produce C_2 ?
    - Joining means simply concatenating 1-itemsets

Itemset	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Join/concatenate 1-itemsets

{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

2-itemsets

Notes on joining itemsets:

Order does not matter. I.e. {I1, I2} = {I2, I1}

When an item appears 2 times in itemset, it is shows up once. I.e. {I1, **I2, I2**, I3} = {I1, I2, I3}

Apriori algorithm: example

- **Step 3:** find frequent 3-itemsets generated from L_2

Generate C_3 from L_2

Itemsets	Support count
{I1, I2, I3}	
{I1, I2, I5}	
{I1, I3, I5}	
{I2, I3, I4}	
{I2, I3, I5}	
{I2, I4, I5}	

C_3 : Candidate frequent 3-itemsets BEFORE apriori principle

Apply apriori principle and prune!
Then calculate support count

Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2
~~{I1, I3, I5}~~	
~~{I2, I3, I4}~~	
~~{I2, I3, I5}~~	
~~{I2, I4, I5}~~	

C_3 : Candidate frequent 3-itemsets AFTER apriori principle

Remove itemsets with support count < minsup (=2)

Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3 : frequent 3-itemsets

Apriori algorithm: example

- **Notes on step 3**
  - From L_2 generate all **3-itemsets by joining 2-itemsets in set L_2**. But **keep** only those that **result in 3-itemsets**.
    - Example joining **{I1, I2}** and **{I1, I3}** results in **{I1, I1, I2, I3} => {I1, I2, I3}** , 3-itemset so keep it. **Will be in C_3**.
    - Example joining **{I1, I5}** and **{I2, I3}** results in **{I1, I2, I3, I5}** which is **not a 3-itemset** (it's a 4-itemset). **So won't be in C_3.**
  - Apply apriori principle on the C_3 candidate 3-itemsets.

Apriori algorithm: example

- **Notes on step 3**
  - How is the apriori principle applied on C_3?
    - "**If an itemset is frequent then all its subsets must be frequent also**" OR "**if a itemset is not frequent, then all its supersets won't be frequent either**".
    - Lets examine **one 2-itemset in C_3 e.g. {I1, I2, I3}** and lets check all its 2-itemset subsets i.e. {I1, I2}, {I1, I3}, {I2, I3}. If **all these subsets are not frequent**, then neither {I1, I2, I3} will be frequent (apriori principle)
      - However all subsets appear in L2, hence are frequent, so {I1, I2, I3} will also be frequent. So {I1, I2, I3} will be not pruned and should stay in C_3 .
    - However, **examine now 3-itemset {I2, I3, I5} in C_3** and its 2-itemset subsets **{I2, I3}, {I2, I5}, {I3, I5}**. 2-itemsets **{I2, I3}** and **{I2, I5}** are in L_2 and hence frequent. **But {I3, I5} is not in L_2 meaning its not frequent. Hence {I2, I3, I5} won't be frequent either! So prune/remove this from C_3**

Apriori algorithm: example

- **Step 4:** find frequent 4-itemsets generated from L_3

Generate C_4 from L_3

Itemsets	Support count
{I1, I2, I3, I5}	?

C_4 : Candidate frequent 4-itemsets BEFORE apriori principle

Apply apriori principle to see if all 3-itemset subsets frequent

Itemsets	Support count
~~{I1, I2, I3, I5}~~	?

C_4 : Candidate frequent 4-itemsets. **Empty set! Apriori algorithm terminates**

How to apply apriori principle here: for 4-itemset **{I1,I2,I3,I5} in C_4** list all **3-itemset subsets: {I1, I2, I3}, {I1, I2, I5}, {I1,I3,I5}, {I2, I3,I5}**. See if **all these subsets are frequent i.e. are in the L_3 list.**

Not all are in L_3 list. For example **subset {I1, I3, I5}** is not in L_3 meaning it is not frequent. Hence **{I1,I2,I3,I5}** will **not be frequent either and must be pruned/removed.** Since C_4 becomes **empty**, **apriori algorithm terminates**

Apriori algorithm: example

- **Step 5:** List the frequent itemsets found

Frequent itemsets found by apriori alg. = $\cup_k L_k$, $k=1,2,3$

Result =

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Frequent itemsets found by the apriori algorithm that have the required minimum support of 2 .

Apriori algorithm

- Methods to join itemsets to produce candidate itemsets (C_2, C_3, C_4 in example)?
  - **Brute force method**
    - Generate all k-itemsets choose k items from the set of items, d. There are $\binom{d}{k}$ number of k-itemsets. **Complexity $O(d\ 2^{d-1})$.** Expensive!
  - **F_{k-1} x F_1 method**
    - Increase k-1-itemsets with 1 item each time. **Complexity $O(\sum_k k|F_{k-1}|\ |F_1|)$** . Still expensive
  - **F_{k-1} x F_{k-1} method**
    - Join 2 itemsets only if they have k-2 itemsets in common

Apriori algorithm

- **Time complexity** of apriori algorithm?
  - Assume input transactions is N, the threshold is M, number of unique elements is R. Then time complexity of Apriori algorithm (finding frequent itemsets) is:

$$O\left(MN + \sum_{i=1}^{M} R^i\right) = O\left(MN + \frac{1 - R^M}{1 - R}\right)$$

Apriori algorithm

- Until now **we have completed step 1** i.e. finding frequent itemsets
- Need to **complete step 2, finding association rules** that satisfy a **minimum confidence threshold, minconf**
  - How to find such rules?

Apriori algorithm

- How to generate rules
  - **Generate rules from frequent itemsets**
  - Two approaches
    - **Brute force approach**
    - **Confidence-based pruning approach**

Apriori algorithm

- Brute force approach procedure
  - Assume you have already all frequent itemsets, *S*
  - For each itemset I in *S* calculate all nonempty subsets of I
  - For each non-empty subset s of I output the rule :

$$\mathbf{s \rightarrow (I - s)}$$

If confidence of rule is at least minconf i.e.
c(s → (I-s)) >= minconf

Brute force rule generation-Example

- Frequent itemsets from previous example
  - **Assume minimum confidence = 70% (0.7)**

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Frequent itemsets found by the apriori algorithm that have the required minimum support of 2 .

Take one frequent itemset e.g. I = {I1, I2,I5}

1) Calculate all nonempty subsets of I, {I1, I2, I5} => {I1}, {I2}, {I5}, {I1, I2}, {I1,I5}, {I2, I5}
2) For each subset s of I, devise **rule s → (I-s)**:
   1) **{I1} → {I2, I5}**
   2) **{I2} → {I1, I5}**
   3) **{I5} → {I1, I2}**
   4) **{I1, I2} → {I5}**
   5) **{I1, I5} → {I2}**
   6) **{I2, I5} → {I1}**

Brute force rule generation-Example

- Frequent itemsets from previous example
  - **Assume minimum confidence = 70% (0.7)**

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Frequent itemsets found by the apriori algorithm that have the required minimum support of 2 .

3) Calculate confidence for each rule in step 2). Keep those that have confidence >= minconf
   1) **{I1} → {I2, I5}, c= 2/6 = 0.333 (REJECT!)**
   2) **{I2} → {I1, I5}, c = 2/7 = 0.28 (REJECT!)**
   3) **{I5} → {I1, I2}, c=2/2 = 1 (KEEP!)**
   4) **{I1, I2} → {I5}, c=2/4 = 0.5 (REJECT!)**
   5) **{I1, I5} → {I2}, c=2/2 = 1 (KEEP!)**
   6) **{I2, I5} → {I1}, c=2/2 = 1 (KEEP!)**

Brute force rule generation- Example

- Found 3 strong association rules satisfying threshold of **minconf = 0.7**
  - **{I5} → {I1, I2}, confidence = 1 (>= minconf)**
  - **{I1, I5} → {I2}, confidence =1 (>= minconf)**
  - **{I2, I5} → {I1}, confidence =1 (>= minconf)**
- Do this process for **all frequent itemsets**
  - I.e. {I1, I2}, {I1, I3}, {I1, I5}, …{I1, I2, I3}
- Output all strong rules (confidence >= minconf)

Brute force rule generation

- **Brute force looks nice** and easy, but has an important problem!
  - **For large databases (usually the case)** it's very, very slow
  - Complexity of brute force approach?
    - If for an itemset I, | I | = k, the number of candidate association rules derived from I is:

$$2^k - 2 = O(2^k)$$

(ignoring **X → ø** and **ø → X**)

Brute force rule generation

- For example
  - 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

- Brute force in **general prohibitive**
  - **Can we do better?**
    - Yes! Using **Confidence-based pruning**

Confidence-based pruning

- **<u>In general</u>**, confidence has **not the anti-monotone property**
  - E.g. **$c(ABC \rightarrow D)$** can be larger or smaller than **$c(AB \rightarrow D)$** although AB subset of ABC
- **HOWEVER**, **<u>rules generated from the same itemset</u> HAVE the anti-monotone property!**
  - Example: X = {A,B,C,D}
    - **$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$ (WHY?)**

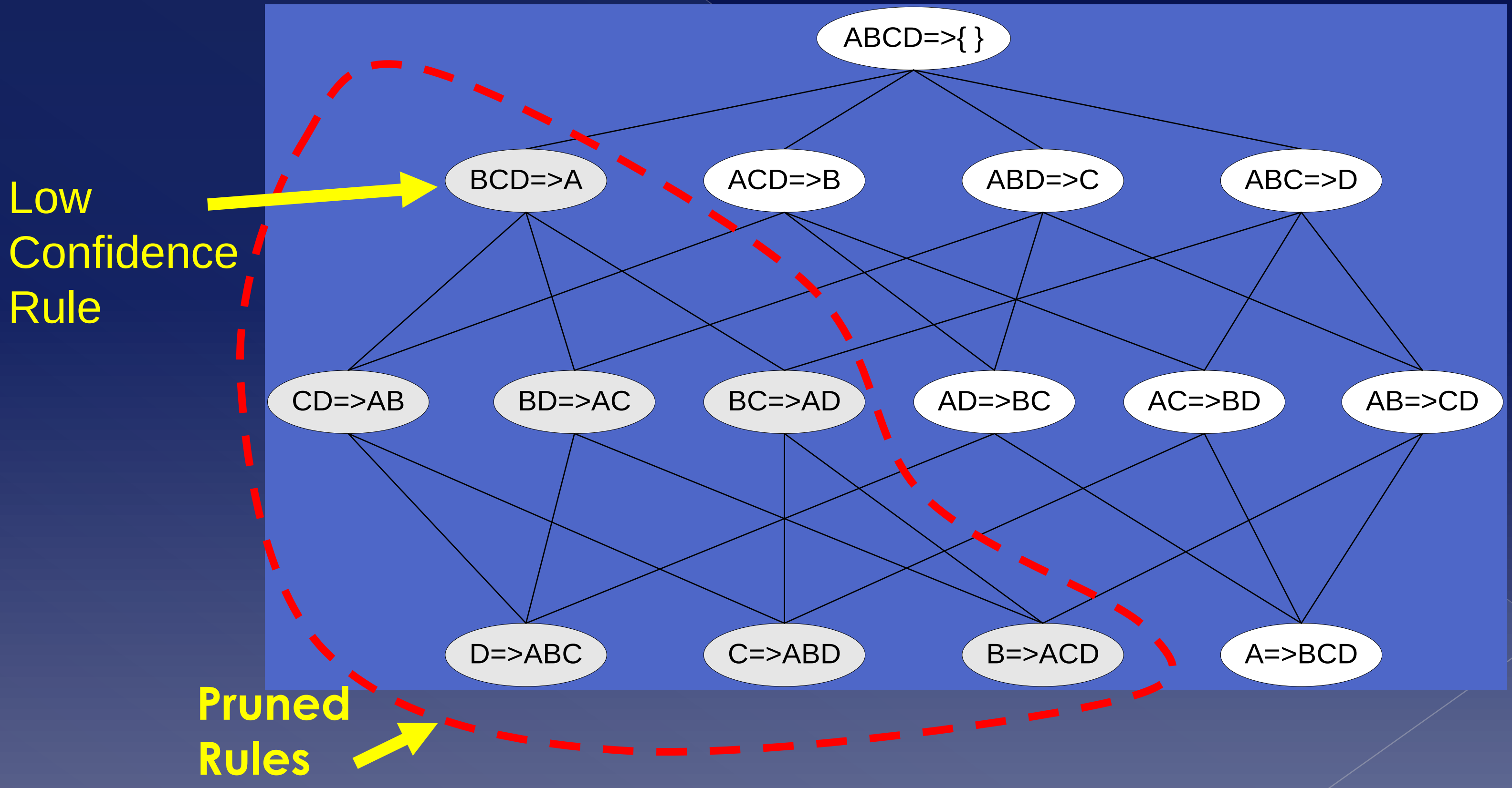
Confidence-based pruning

- **Confidence is anti-monotone** w.r.t. number of items on the RHS of the rule
- **Anti-monotone property of confidence**
  - If an association rule $\mathbf{X \rightarrow S - X}$ has less than the minimum confidence threshold, then all rules $\mathbf{X' \rightarrow S - X'}$, where $\mathbf{X' \subseteq X}$ will have also less than the confidence threshold
    - Hence, you **can "automatically" prune/ignore them**

Confidence-based pruning

- Put this idea into play to prune association rules
  - We don’t need to check them all as in the brute force approach

Lattice of rules



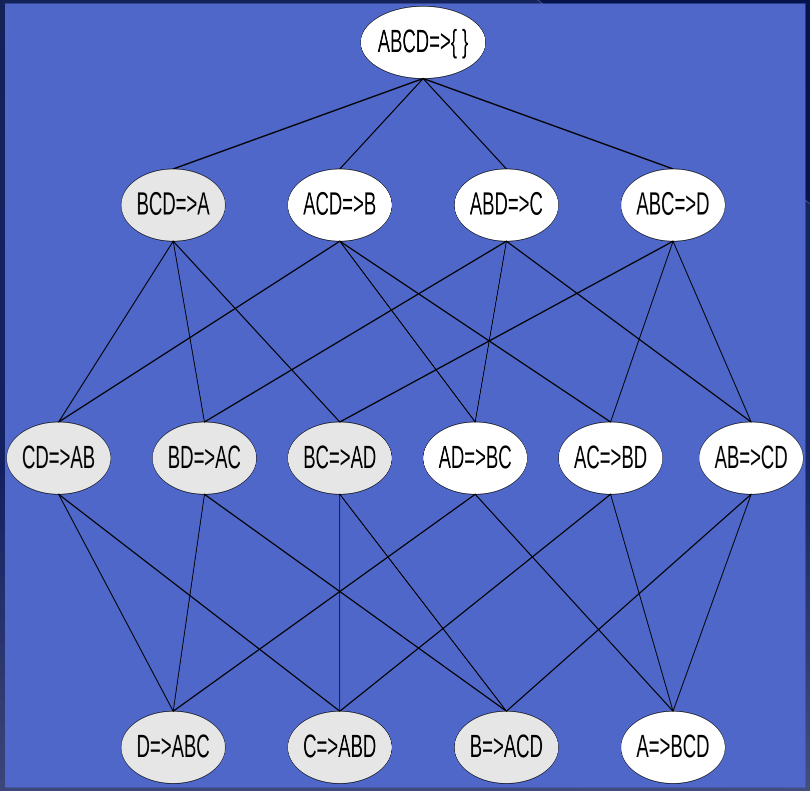
Confidence-based pruning

- The idea explained
  - Suppose **frequent itemset S={1,2,3,4}** and some **minconf** value
    - If rule **{1,2,3} → {4} does <u>not have minimum confidence</u>** <u>(i.e. < minconf)</u> then **all these rules won't have minconf either** (i.e. < minconf):

{1,2} → {3,4}
{1,3} → {2,4}
{1,4} → {2,3}
{1} → {2,3,4}
{2} → {1,3,4}
{3} → {1,2,4}

Confidence of all these rules will be less than minconf also as LHS subset of {1,2,3} → {4}.

Confidence-based pruning

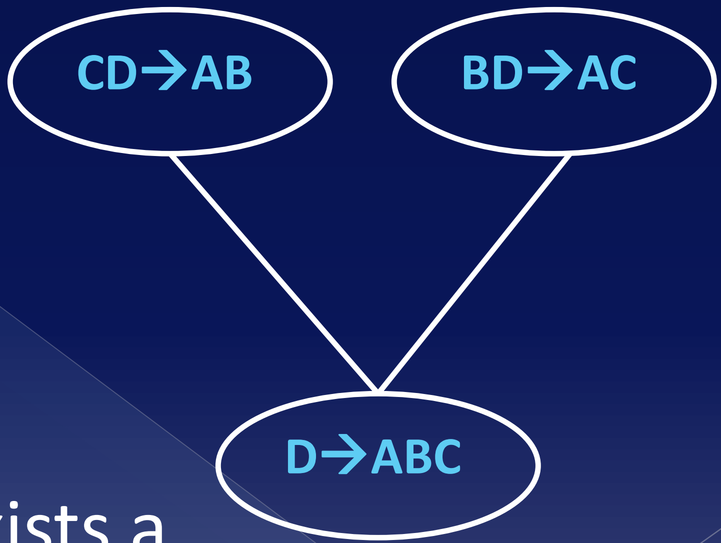


- Algorithm for building rules on confidence-based pruning:
  - Generate rules in a level-wise approach of the lattice:

1. First find rules of the form **{…} → {x}** i.e. **only one item in the consequent**
2. Prune rules of the form **{…} → {x}** that **do not have minconf**
3. Generate/join rules of the form **{…} → {x, y} i.e. two items in the consequent**, only from rules in step 2 **(note: here confidence-based pruning is applied)**
4. Prune rules **{…} →{x,y}** that **do not have minconf**
5. Generate/join rules of the form **{…} → {x,y,z}**
6. Continue incrementally that way….

Confidence-based pruning

- Candidate rule is generated by **joining/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 there exists a subset (e.g., **AD→BC**) that **does not have high confidence (minconf)**



Confidence-based pruning – Example

TID	Transactions
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2 ,I3, I5
9	I1, I2, I3

- Take one frequent itemset found: e.g. **{I1, I2, I5}, minconf=0.7**
- Generate rules **with one item in the consequent**
  - **{I1, I2} -> {I5}**, conf=2/4 = 0.5 **< 0.7** => Prune this (confidence-based pruning) and don't generate rules out of this.
  - **{I1, I5} -> {I2}**, conf=2/2 = 1
  - **{I2, I5} -> {I1}**, conf=2/2 = 1

Confidence-based pruning – Example

TID	Transactions
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2 ,I3, I5
9	I1, I2, I3

- Join/merge rules with **confidence >= minconf**
  - Join **{I1,I5} →I2 and {I2, I5} →I1** resulting in **I5 → {I1, I2}**
  - Confidence for **I5 →{I1,I2}**
    - Confidence = 2/2 = 1, ok!
  - Rules from {I1, I2, I5}:
    - **{I1, I5} -> {I2}**
    - **{I2, I5} -> {I1}**
    - **{I5} → {I1, I2}**
- Do this for **all frequent itemsets** found!

Apriori algorithm in R

```
#Includes functions for apriori algorithm
library(arules)

#We will be using the Congressional Voting Records Data Set
#From: http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

#First read the data. Note the dataset HAS NO headers, hence set header to FALSE.
#We well add headers later. NOTE: Change your path to data appropriately!
voteData = read.csv("house-votes-84.data", header=FALSE)
attach(voteData)

#Add headers to data. Makes working with dataset easier
colnames(voteData) <- c("party", "infants", "water-cost", "budgetRes", "PhysicianFr",
"ElSalvador", "ReligSch", "AntiSat", "NicarAid", "Missile", "Immigration",
"CorpCutbacks", "EduSpend", "RightToSue", "Crime", "DFExports", "SAExport")

#Take a quick look at the data. Is everything ok?
head(voteData)

#Now we are ready to execute the apriori algorithm for finding association rules
#See next slide…
```

Apriori algorithm in R

```
#Execute now the apriori algorithm without any parameter.
#This means that no minsup and minconf is provided and
#that all possible rules will be generated
rules <- apriori(voteData)

#Variable rules has all the rules. Can we see the rules now?
#Yes, but this may take a huge amount of time due to the number
#of rules
#CAVEAT LECTOR: DO THIS ONLY IF YOU HAVE NOTHING BETTER TO DO
#YOU HAVE BEEN WARNED.
inspect(rules)

#Lets execute apriori with the following parameters: minimum support 20%,
#minimum confidence=100%, on the LHS we need at least 2 items and on the
#RHS only the party should appear i.e. rules of the form {X,Y}->{republican} or
#{X,Y}->{democrat}
rules <- apriori(voteData, parameter = list(minlen=2, supp=0.2, conf=1), appearance =
list(rhs=c("party=democrat", "party=republican"), default="lhs"))

#Lets see the rules. Should not be that much. You can also see support and
#confidence of each rule.
inspect(rules)
```

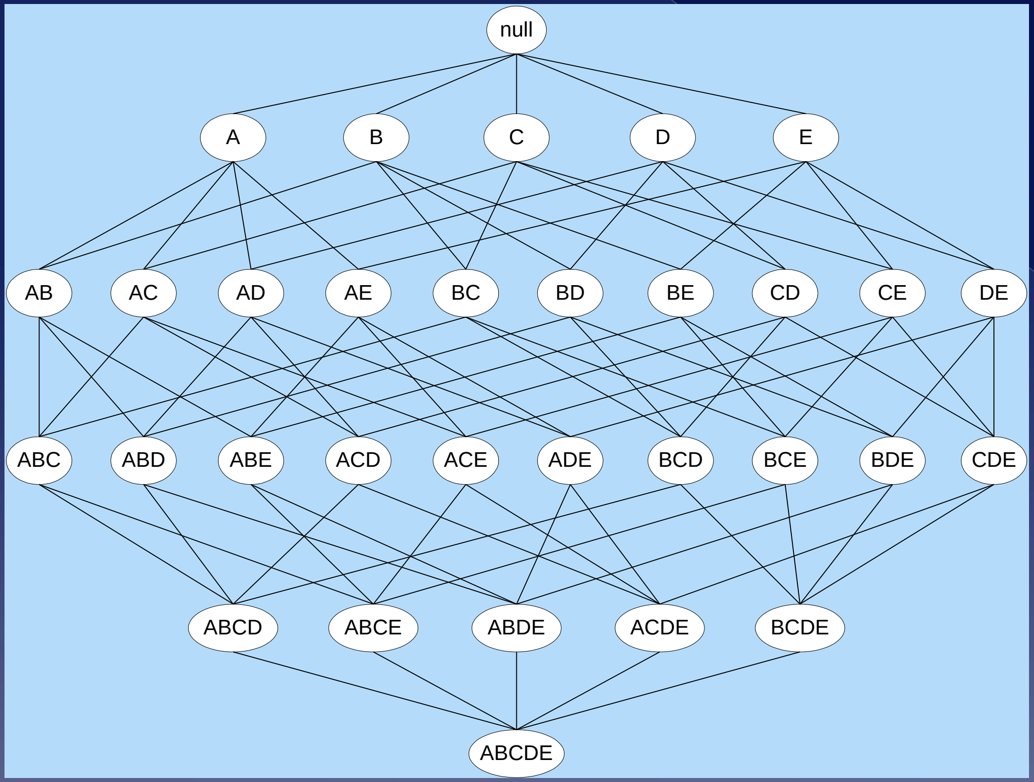
Alternative representations of frequent itemsets

Alternative representations

- For **very large datasets**, there may be **large number of frequent itemsets**
  - **Enumerating, storing** them may be very **costly**
  - Some frequent itemsets are redundant because they have identical support as their (frequent) supersets
  - **Question:** Is there a better way to represent frequent itemsets?

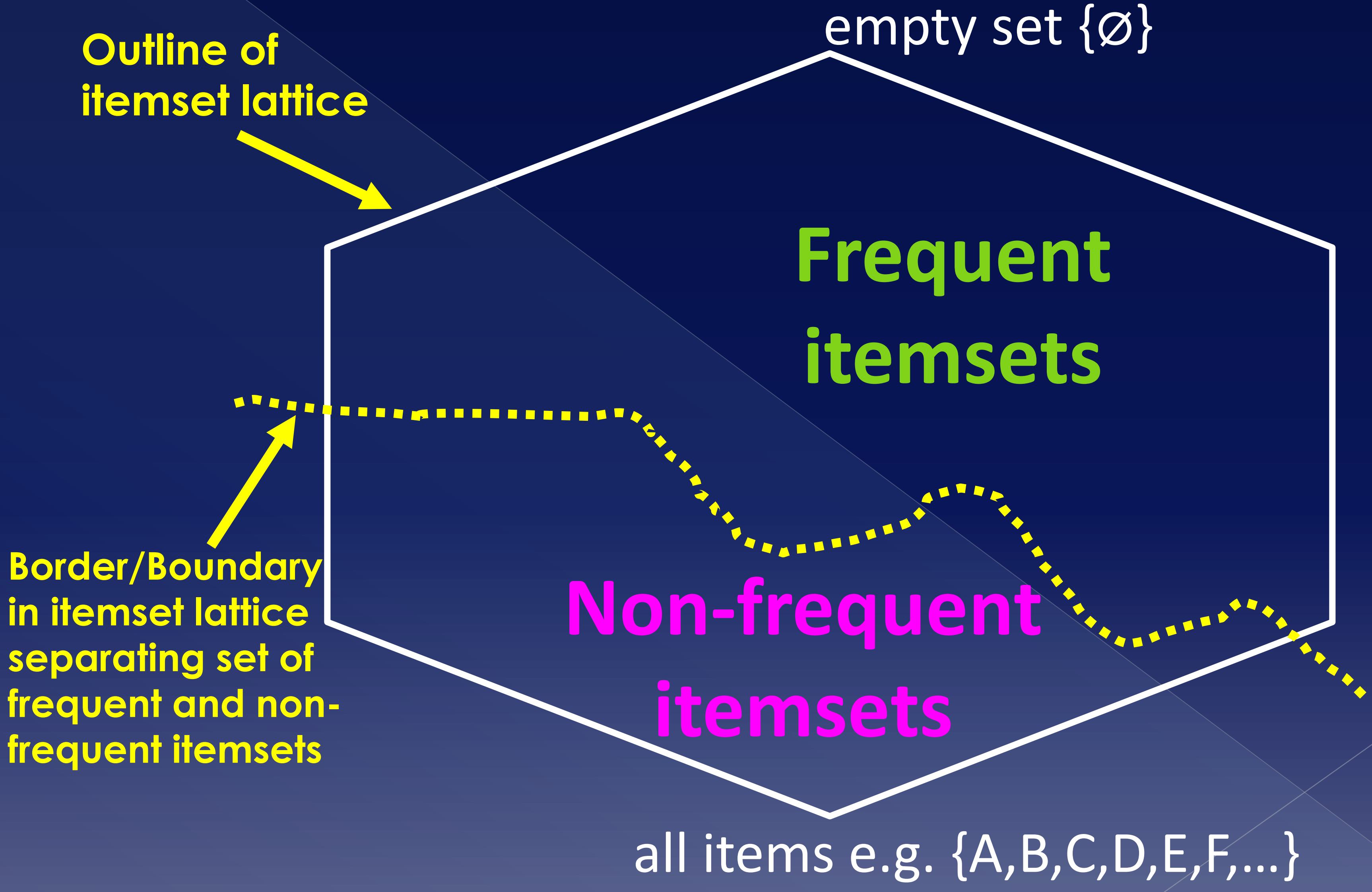
Alternative representations

- Yes. Exploit the notion of **border** in the itemset lattice and **find the boundary frequent itemsets**



Itemset lattice: lists all combinations by proceeding level-wise.
We say that e.g. <u>immediate subset</u> of {BE} is {B, E}

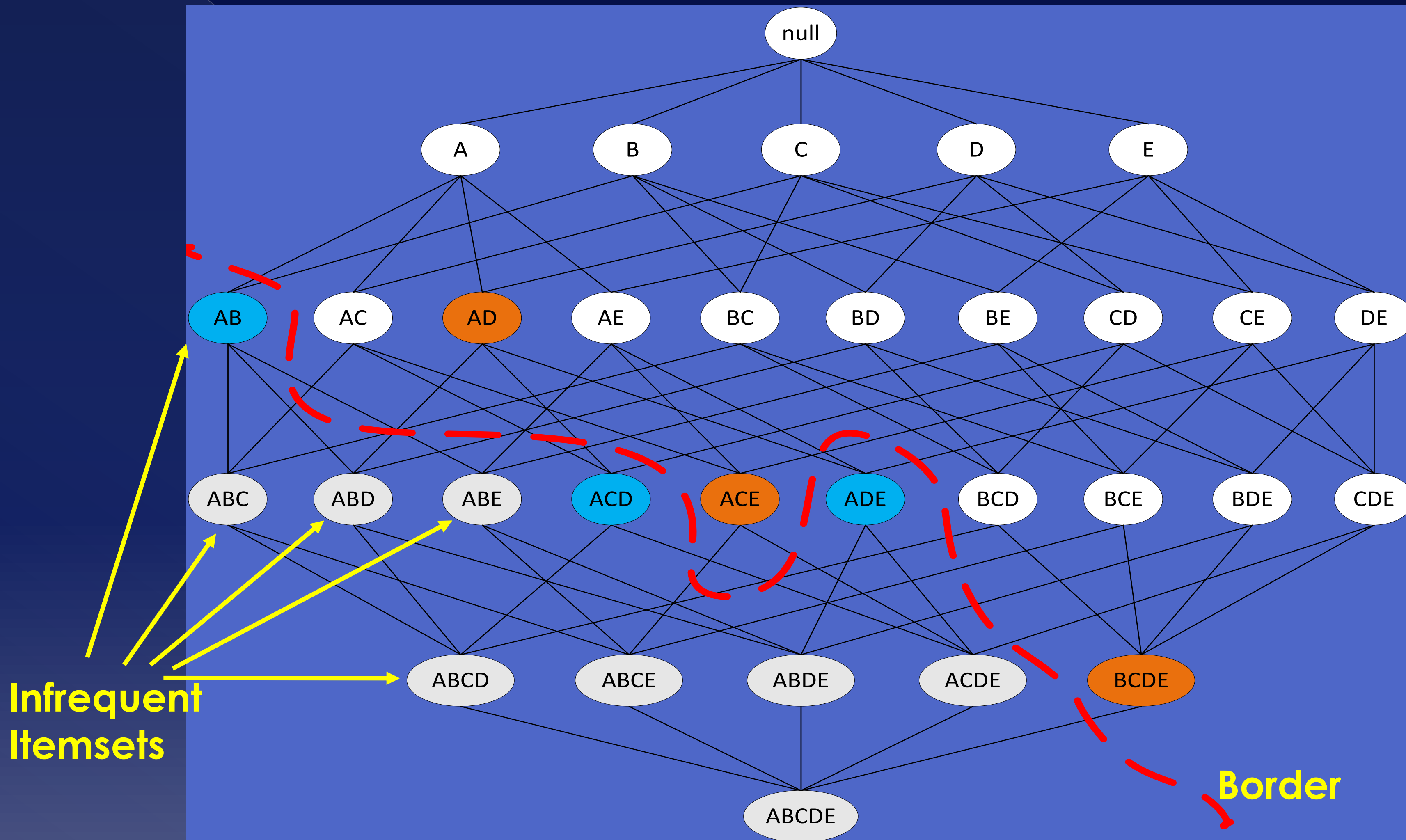
Alternative representations



Alternative representations

- Defining the Border in an itemset lattice
  - **Border** = set of itemsets whose **all their <u>immediate subsets</u> are frequent** AND **all their <u>immediate supersets</u> are infrequent (not frequent)**.
  - **Positive Border, B+(S)**= Frequent itemsets whose all their immediate supersets are not frequent
  - **Negative Border, B-(S)**= Non-frequent Itemsets (in border) whose all their immediate subsets are frequent

Alternative representations

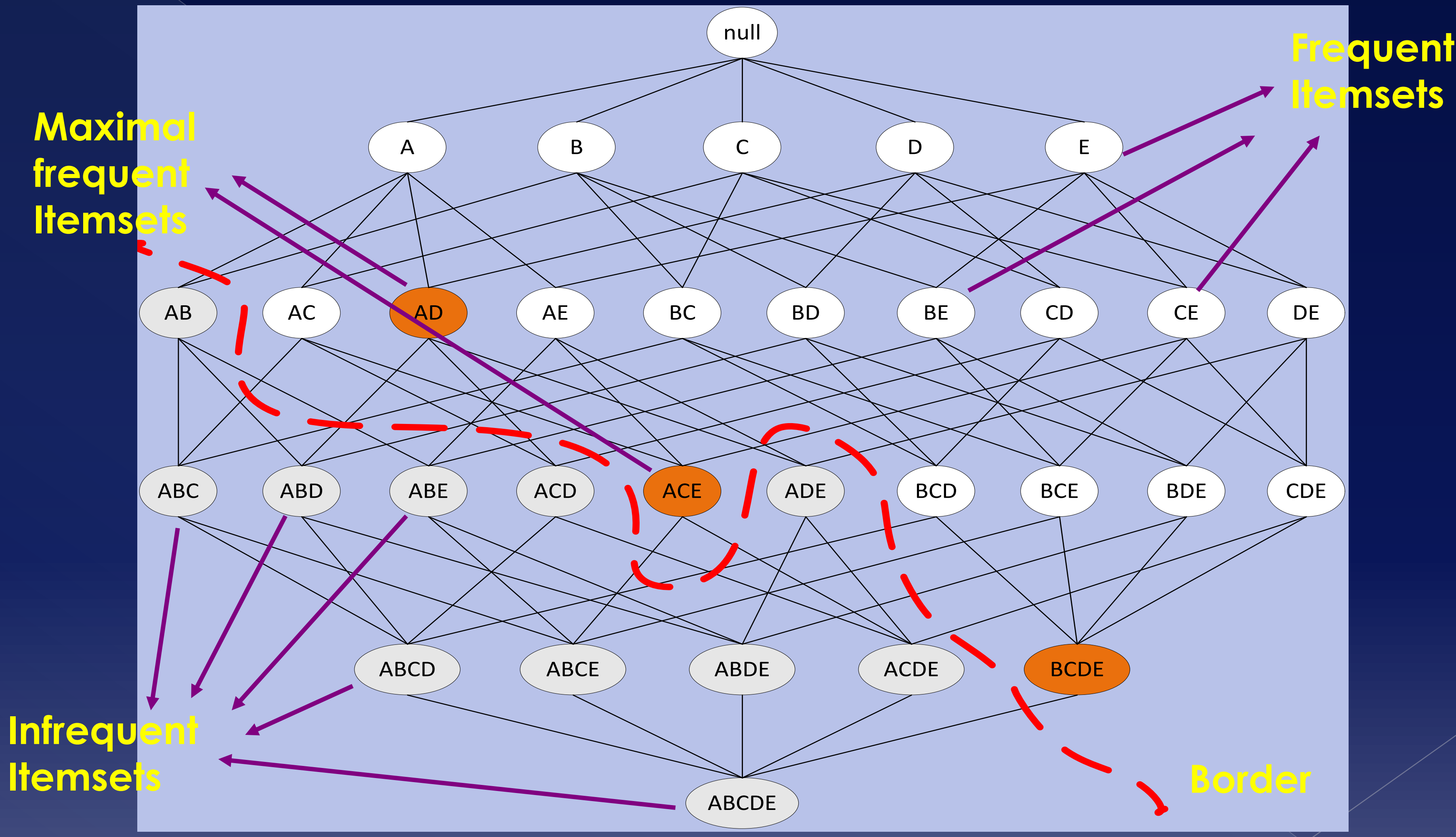


<u>Above</u>: Negative Border = {AB, ACD, ADE} . E.g. ABD not in negative border since not all its immediate subsets frequent.

Alternative representations

- **Maximal frequent itemsets**
  - An **itemset is maximal frequent** if **none of its immediate supersets is frequent**
    - *Maximal*: no superset has this property (i.e. is frequent)

Alternative representations



<u>Above</u>: **Maximal Itemsets = {AD, ACE, BCDE} . E.g. AE not in maximal itemsets since not all its immediate supersets infrequent.**

Finding maximal frequent itemsets-Example

Assume these frequent itemsets found

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Finding Maximal frequent temsets:

Recap: For each **frequent** itemset check all its **immediate supersets to see if they are frequent (=if at least one immediate superset frequent, the itemset is NOT MAXIMAL frequent)**

{I1} => Immediate supersets = {I1, I2}, {I1,I3}, {I1,I4}, {I1,I5} => {I1, I2} frequent hence **{I1} not maximal**

{I2} => Immediate supersets = {I1, I2}, {I2, I3}, {I2, I4}, {I2, I5} => all frequent hence **{I2} not maximal**

Finding maximal frequent itemsets-Example

Assume these frequent itemsets found

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Finding Maximal itemsets (continued):

{I3} => Immediate supersets = {I1, I3}, {I2, I3}, {I3,I4}, {I3,I5} => some frequent hence **{I3} not maximal**

{I4} => Immediate supersets = {I1, I4}, {I2, I4}, {I3, I4}, {I4, I5} => {I2, I4} frequent hence **{I4} not maximal**

{I5} => Immediate supersets = {I1, I5}, {I2, I5}, {I3, I5}, {I4, I5} => {I1, I5}, {I2, I5} frequent hence **{I5} not maximal**

Finding maximal frequent itemsets-Example

Assume these frequent itemsets found

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Finding Maximal itemsets (continued):
{I1, I2} => Immediate supersets = {I1, I2, I3}, {I1, I2, I4}, {I1, I2, I5} => some frequent (e.g. {I1,I2,I3}) hence **{I1, I2} not maximal**
{I1, I3} => Immediate supersets = {I1, I2, I3}, {I1, I3, I4}, I1, I3, I5} => some frequent hence **{I1, I3} not maximal**
{I1, I5} => Immediate supersets = {I1,I2, I5}, {I1, I3, I5}, {I1, I4, I5} => {I1, I2, I5} frequent hence **{I1, I5} not maximal**

Finding maximal frequent itemsets-Example

Assume these frequent itemsets found

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Finding Maximal itemsets (continued):
{I1, I2} => Immediate supersets = {I1, I2, I3}, {I1, I2, I4}, {I1, I2, I5} => some frequent (e.g. {I1,I2,I3}) hence **{I1, I2} not maximal**
{I1, I3} => Immediate supersets = {I1, I2, I3}, {I1, I3, I4}, {I1, I3, I5} => some frequent hence **{I1, I3} not maximal**
{I1, I5} => Immediate supersets = {I1,I2, I5}, {I1, I3, I5}, {I1, I4, I5} => {I1, I2, I5} frequent hence **{I1, I5} not maximal**

Finding maximal frequent itemsets-Example

Assume these frequent itemsets found

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

Finding Maximal itemsets (continued):

{I2, I3} => **not maximal**

{I2, I4} => Immediate supersets = {I1, I2, I4}, {I2, I3, I4}, {I2, I3, I5} => all supersets not frequent hence **<u>{I2, I4} MAXIMAL!</u>**

{I2, I5} => **not maximal**

{I1, I2, I3} => **<u>MAXIMAL!</u>**

{I1, I2, I5} => **<u>MAXIMAL!</u>**

Finding maximal frequent itemsets-Example

Assume these frequent itemsets found

Frequent 1-Itemsets	Support count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Frequent 2-Itemsets	Support count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Frequent 3-Itemsets	Support count
{I1, I2, I3}	2
{I1, I2, I5}	2

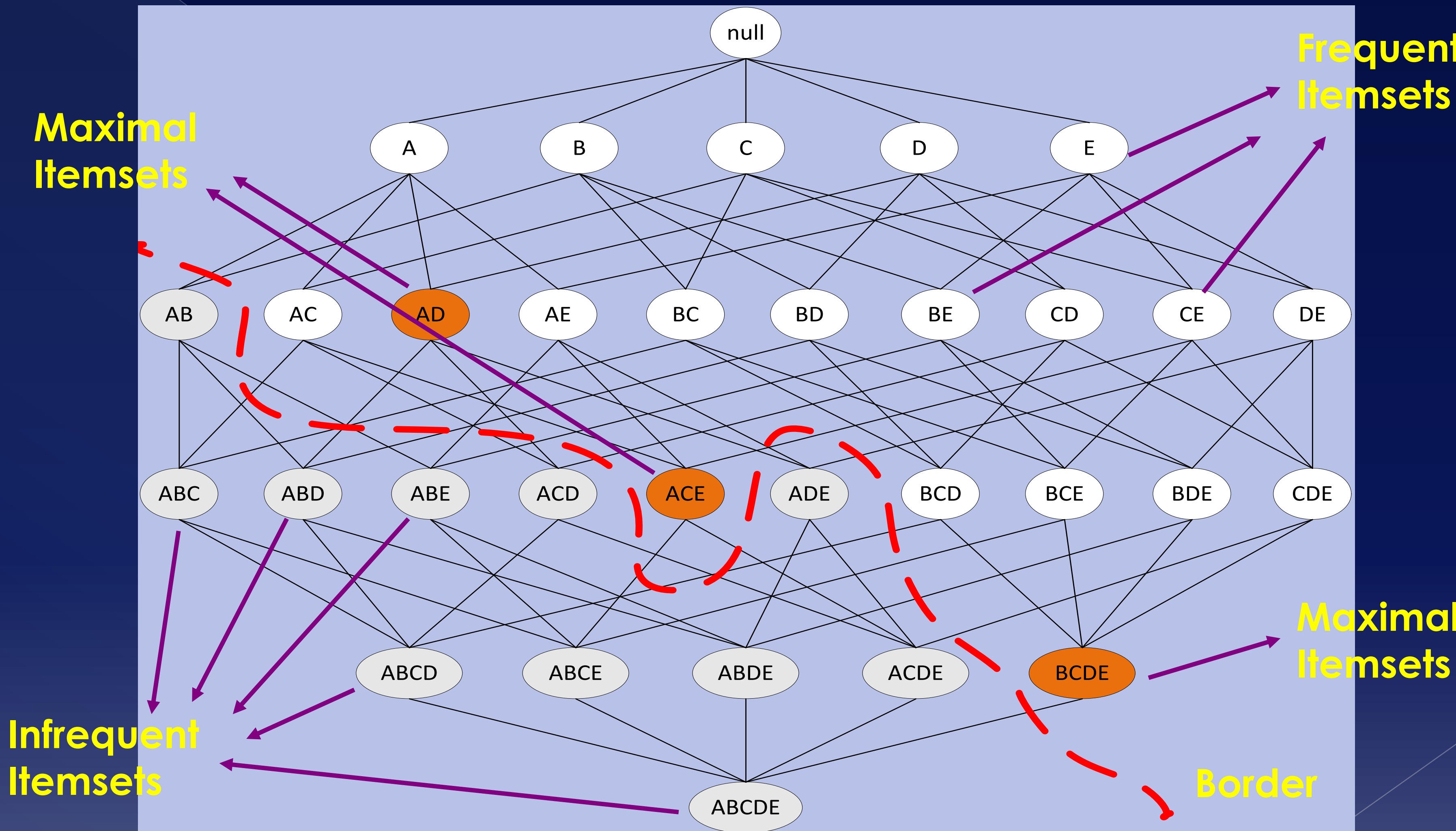
Finding Maximal itemsets (continued):

Maximal itemsets = {I2, I4}, {I1, I2, I3}, {I1, I2, I5} **Q.E.D**

Alternative representations

- **Important note**
  - **Maximal frequent itemsets** = the **Positive border** of the lattice tree

Alternative representations



Above: Positive border = maximal frequent itemsets (orange nodes).

Alternative representations

- Why define **Border, Negative and Positive border (B-(S), B+(S))** ? Are they useful?
  - Yes! The **Positive or the Negative border** is **sufficient to fully describe** all frequent itemsets !
    - Hence, don't need to store all frequent itemsets. Just B-(S) or B+(S)

Alternative representations

- Maximal frequent itemsets look very nice!
  - They can **summarize nicely frequent itemsets**.
  - But, maximal frequent itemsets don’t tell us anything about the support measure - σ
    - This might be needed
- Define **closed itemsets**

Alternative representations

- Closed itemsets
  - An **itemset X is closed** if **<u>none of its immediate supersets</u> has exactly the same support as the itemset X**
  - Example

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Closed itemsets:
{A}
{B}
{A,B}
{B, C}
{B,D}
{C,D}
{A, B, D}
{A, B, C, D}

Alternative representations

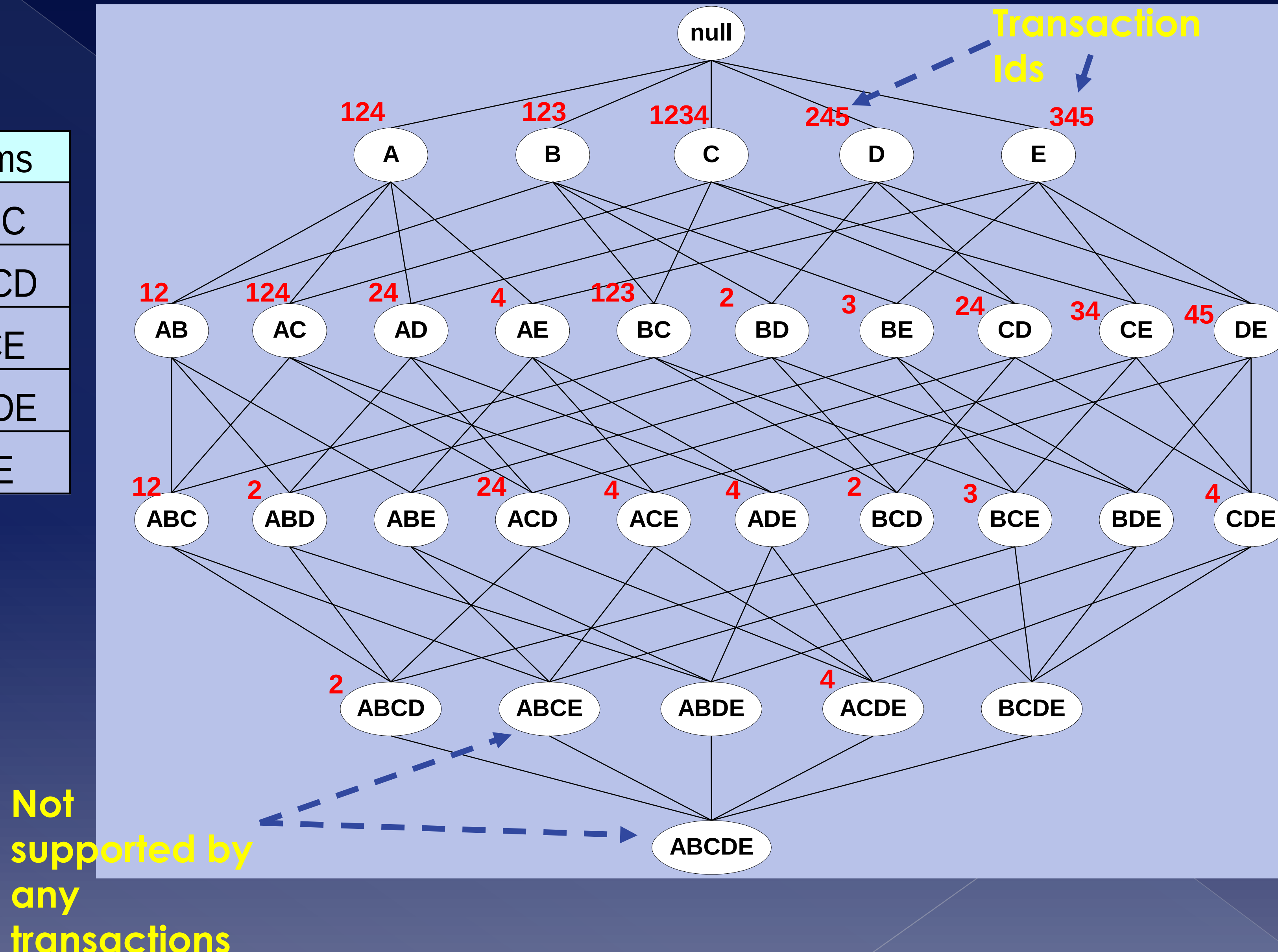
- Why are closed itemsets interesting?
  - Assume rule **{A}→{B}** and **{A,B} closed itemset**. Moreover, assume s({A,B}) = s(A).
    - Then confidence of rule is: **conf({A}→{B}) = 1**
    - In addition, for every itemset X it will hold that
      - **s(A ∩ {X}) = s({A,B} ∩ X)**
      - **No need to count the frequencies** of sets X ∩ {A,B} from the database!
  - If there are lots of **rules with confidence 1**, then a **significant amount of work can be saved**

Alternative representations

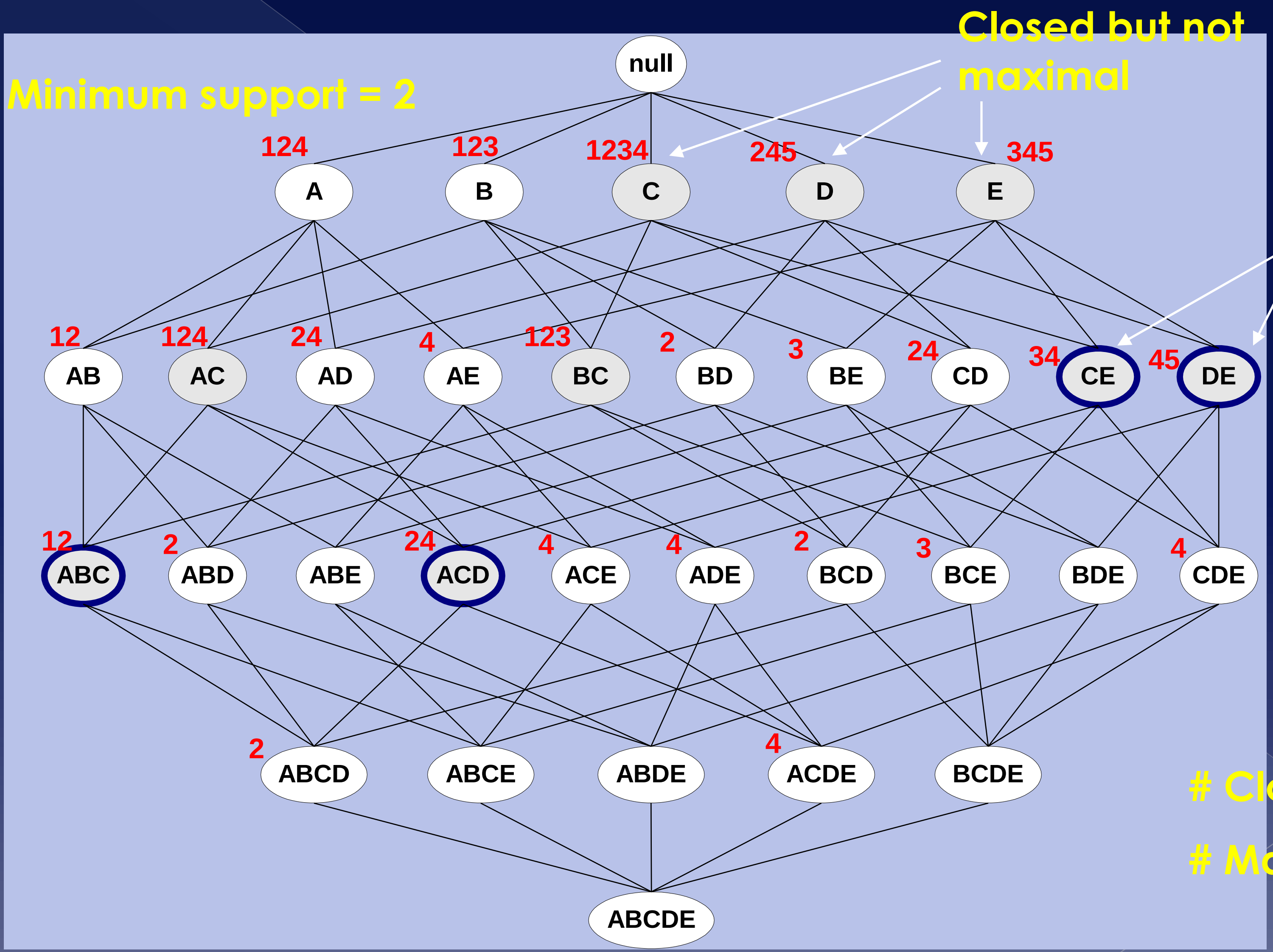
- **Closed patterns and their frequencies** alone are **sufficient representation for all the frequencies of all frequent patterns**

Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs Closed Frequent Itemsets

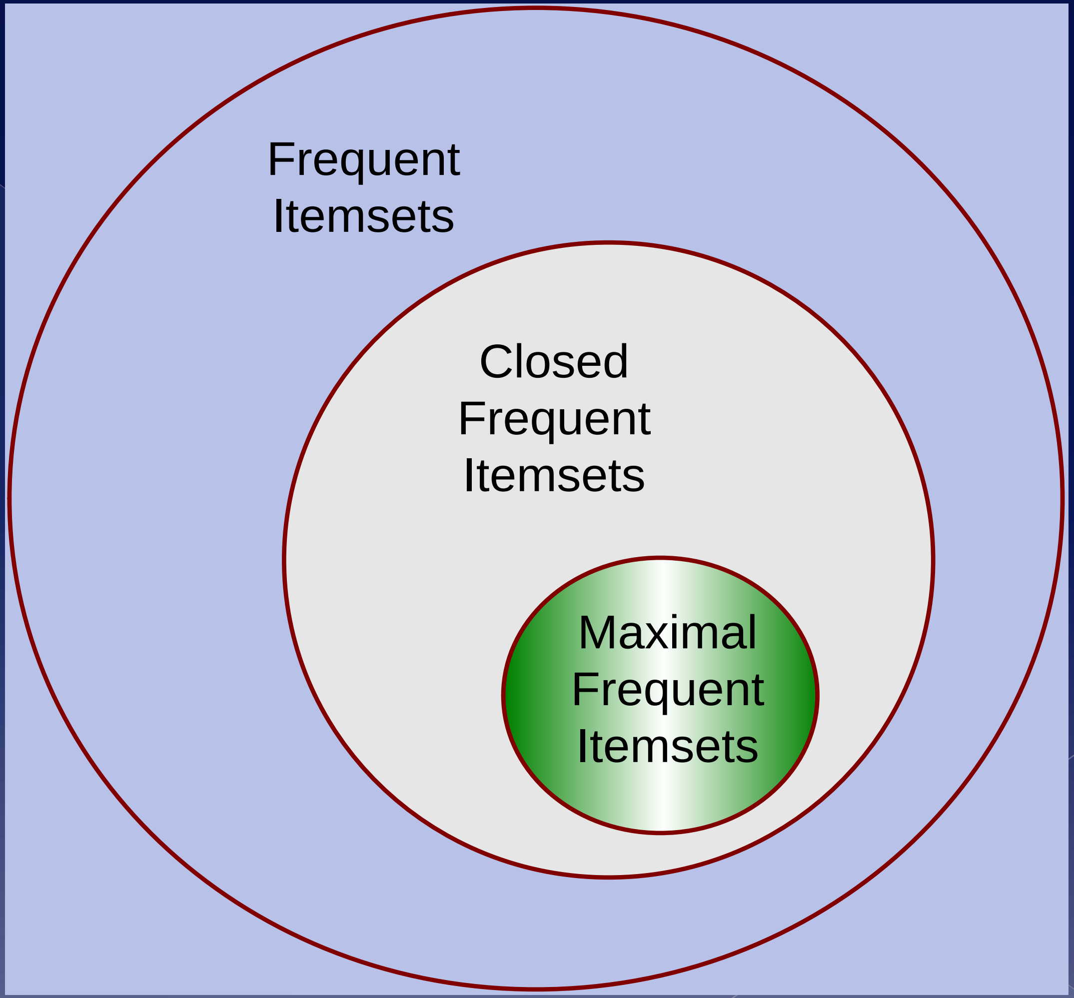


Closed = 9

Maximal = 4

Maximal vs Closed Frequent Itemsets

- Knowing **all maximal itemsets** (and their frequencies) allows us to **reconstruct the set of frequent itemsets**
- Knowing all **closed itemsets and their frequencies** allows us to **reconstruct the set of all frequent itemsets and their frequencies**



Interestingness measures of association rules

Rule interestingness measures

- Are all the rules discovered **interesting to the user?**
  - How to measure “interestingness” of a rule?
- When is a discovered association rule interesting (subjective measure)?
  - It is **unexpected** (surprising to the user)
    - E.g. {Cigarettes} → {Lighter} not unexpected. But {Cigarettes} → {Barbie Doll} <u>unexpected</u>
  - It is **actionable** (i.e. user can do something with it, lead to profitable actions)
  - **Only the user** can **judge the interestingness** of a rule (subjective)

Rule interestingness measures

- In general, algorithms (like Apriori) tend to produce many rules
  - Many of them **not interesting** or **redundant**
  - Example of **redundant rule**:
    - Redundant if discovered rules **{A,B,C} → {D}** and **{A,B} → {D}** have same support & confidence
- The **original formulation** of the problem of finding association rules is **only based on <u>support</u> and <u>confidence</u> of rules**

Rule interestingness measures

- Idea
  - Use some form of correlation measure for rules i.e. given rule **$A \rightarrow B$ measure the correlation between itemsets A and B**
  - In essence, find a way of **comparing co-occurrence of itemsets A and B** with the **<u>probability of itemsets A and B appearing together by chance (at random)</u>**
    - Hence see if a rule is discovered randomly
    - Or check if two itemset A, B are statistically independent

Recall: Statistical independence

- Assume some students, where **some can swim (S)**, **some can Bike (B), some can Swim and Bike ($S \cap B$) and some can neither**
  - **Q: Are events "know how to swim (S)" and "know how to bike (B)" independent or not?**
    - I.e. Does occurrence of event S influence the occurrence of event B (and vice versa) or not?
  - To check for **statistical independence** between S and B, **check if $P(S \cap B) = P(S)\,P(B)$** . <u>If this holds then event S, B independent</u>. If not, not independent and hence somehow correlated.

Recall: Statistical independence

- Assume population of 1000 students
  - 600 students know how to **swim (S)**
  - 700 students know how to **bike (B)**
  - 420 students know how to **swim** and **bike ($S \cap B$)**
    - **$P(S \cap B) = 420/1000 = 0.42$**
    - **$P(S) = 600/1000 = 0.6$**
    - **$P(B) = 700/1000 = 0.7$**
    - **$P(S)P(B) = 0.6 \times 0.7 = 0.42$**
    - **Since $P(S \cap B) = P(S)\,P(B)$ => S, B <u>Statistical independence</u>**

Recall: Statistical independence

- Population of 1000 students
  - 600 students know how to **swim (S)**
  - 700 students know how to **bike (B)**
  - **500** students know how to **swim** and **bike (S ∩ B)**
    - **P(S ∩ B) = 500/1000 = 0.5**
    - **P(S) P(B) = 0.6 x 0.7 = 0.42**
    - **Since P(S ∩ B) > P(S) P(B) => <u>S,B positively correlated</u>**
      - **This means that if S increases, so will B. If S decreases, so will B.**

Recall: Statistical independence

- Population of 1000 students
  - 600 students know how to **swim (S)**
  - 700 students know how to **bike (B)**
  - **300** students know how to **swim** and **bike ($S \cap B$)**
    - **$P(S \cap B) = 300/1000 = 0.3$**
    - **$P(S)\,P(B) = 0.6 \times 0.7 = 0.42$**
    - **Since $P(S \cap B) < P(S)\,P(B)$ => <u>S,B negatively correlated</u>**
      - **This means that if S increases, B will decrease. If S decreases, B will increase.**

Rule interestingness measures

- Build "interestingness"/correlation measures of rules around statistical independence
  - Many available like χ^2, Φ-coefficient etc
  - However in Association rule mining, **<u>Lift/Interest</u> is used**

Rule interestingness measures

- Idea of Lift based on Contingency table
  - Given a rule **X → Y**, information needed to compute rule interestingness can be obtained from a **contingency table**

Contingency table for X → Y

	Y	$\bar{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\bar{X}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

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

X: itemset X appears in tuple
Y: itemset Y appears in tuple
$\bar{X}$: itemset X does not appear in tuple
$\bar{Y}$: itemset Y does not appear in tuple

Used to define various measures support, confidence, lift, Gini, J-measure etc.

Drawback of Confidence

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

Number of people that drink tea

Number of people that drink coffee and tea

Number of people that drink coffee but not tea

Number of people that drink coffee

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = $\frac{15}{20} = \mathbf{0.75}$

but P(Coffee) = $\frac{90}{100} = \mathbf{0.9}$

Important note:
P(Tea) = support {Tea}
P(Coffee|Tea) = conf {Tea $\rightarrow$ Coffee}

- Although confidence is high, rule is misleading
- Because: P(Coffee|$\overline{\text{Tea}}$) = **0.9375**

Lift/Interest

- Definition of **Lift/Interest measure**

$$\mathbf{Lift}(\mathbf{X} \rightarrow \mathbf{Y}) = \frac{\mathbf{P}(\mathbf{Y}|\mathbf{X})}{\mathbf{P}(\mathbf{Y})} = \frac{\mathbf{P}(\mathbf{X} \cap \mathbf{Y})}{\mathbf{P}(\mathbf{X})\mathbf{P}(\mathbf{Y})}$$

If Lift = 1, this means $P(X \cap Y) = P(X)P(Y)$ i.e. statistical independence

If Lift < 1, this means $P(X \cap Y) < P(X)P(Y)$ i.e. negative correlation

If Lift > 1, this means $P(X \cap Y) > P(X)P(Y)$ i.e. positive correlation

How to use Lift? Use Lift to find interesting rules. In particular, <u>rules for which Lift > 1</u> .

Lift/Interest

- Interpretation of Lift in a different way
  - **P(X)P(Y)** = probability of appearing X, Y together **by chance/at random** (expected co-occurrence)
    - **If $P(X \cap Y) = P(X)P(Y)$** this means that **X,Y appear together as expected (not interesting).** Not interesting.
    - **If $P(X \cap Y) < P(X)P(Y)$** this means that X,Y **appear less times together than expected** (negative correlation). Not interesting
    - **If $P(X \cap Y) > P(X)P(Y)$** this means that **X,Y appear more often together than expected (positive correlation)**
      - This is **interesting!**

Lift examples

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

Assume rule:

Tea → Coffee

- **Interesting rule?** Calculate Lift to see:
  - **Lift = P(Coffee |Tea) / P(Coffee) = 0.75/0.9 = 0.8333.**
  - Since **Lift < 1**, **Tea, Coffee negatively correlated** hence **not interesting rule**!

Lift examples: more complex rules?

Assume more complex rule:

Gun,Milk → Diapers, Flowers

Contingency table would be e.g.:

	Diapers,Flowers	$\overline{\text{Diapers,Flowers}}$	
Gun,Milk	22	23	45
$\overline{\text{Gun,Milk}}$	61	8	69
	83	31	114

Calculate Lift of above rule as:

Lift = P(Diaper,Flowers | Gun,Milk) / P(Diaper,Flowers)

Other metrics?

- Instead of lift/Interest?
- Sure! **Can use χ^2**
  - Use again contingency table

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

Assume rule:

Tea → Coffee

Appendices

Appendix A: Bibliography

- R. Agrawal, T. Imielinski and A. Swami. Mining association rules between sets of items in large databases, Proceedings of the 1993 ACM SIGMOD international conference on Management of data, SIGMOD '93. pg: 207-216)
- J. Han and M. Kamber. Data Mining – Concepts and Techniques. 2001. Morgan Kaufmann.
- M. Kantardzic. Data Mining – Concepts, Models, Methods, and Algorithms. 2003. IEEE.
- M. H. Dunham. Data Mining – Introductory and Advanced Topics.
- I.H. Witten and E. Frank. Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations. 2000. Morgan Kaufmann.

Appendix A: Bibliography

- M.J. Zaki. Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering, Volume 12, Issue 3 (2000), Page 372-390)
- Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, AAAI Workshop on Knowledge Discovery in Databases (KDD-94), pages 181--192, Seattle, Washington, 1994. AAAI Press.
- Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining -- A general survey and comparison. SIGKDD Explorations, 2(2):1--58, 2000.
- J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future directions. Data Mining and Knowledge Discovery, 14(1), 2007

Appendix A: Bibliography

- Julien Blanchard, Fabrice Guillet, Henri Briand, and Regis Gras. Assessing rule interestingness with a probabilistic measure of deviation from equilibrium. In Proceedings of the 11th international symposium on Applied Stochastic Models and Data Analysis ASMDA-2005, pages 191--200. ENST, 2005.
- Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning. IEEE Transactions on Knowledge and Data Engineering, 13(1):64--78, 2001.