Lecture 2

Association rules

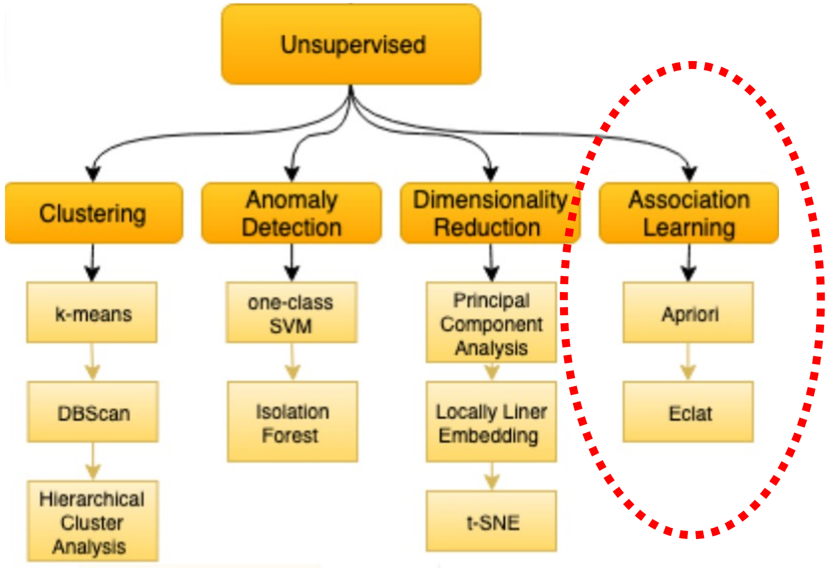
Ioanna Miliou, PhD

Senior Lecturer, Stockholm University

Stockholms universitet

Unsupervised learning

<u>Experience:</u> objects for which **no class labels** have been given
<u>Performance:</u> typically concerns the ability to output useful **characterizations** (or groupings) of objects



Problem 1:
Mining
Frequent Itemsets

Problem 2:
Mining
Association Rules

Stockholms universitet

Problem 1: Mining frequent itemsets

- Given a set of transactions ***D***, find combinations of items that occur *frequently*

Market-Basket transactions

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

Examples of *frequent* itemsets

{Diaper, Beer},
{Milk, Bread},
{Beer, Bread, Milk}

Stockholms universitet

Problem 2: Mining Association Rules

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

Market-Basket transactions

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

Examples of *association rules*

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Diaper, Coke},
{Beer, Bread} $\rightarrow$ {Milk}

Stockholms universitet

Why do we want to find frequent itemsets?

- Find all combinations of items that occur together

- They might be interesting
  - e.g., in placement of items in a store ☺

- Frequent itemsets are *only positive combinations* (we do not report combinations that do not occur frequently together)

- Frequent itemsets aim at providing a summary of the data

Stockholms universitet

Definitions

- Itemset
  - A set of one or more items
    - e.g.: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- Absolute Support
  - Number of transactions in which an itemset appears
  - e.g., $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$
- Relative Support
  - Fraction of the transactions in which an itemset appears
  - e.g., $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$
- Frequent Itemset
  - An itemset whose relative support is greater than or equal to a ***minsup*** threshold

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**

Stockholms universitet

Mining frequent itemsets

- Task:

  Given a transaction database ***D*** and a ***minsup*** threshold, find all frequent itemsets and the frequency of each set in this collection

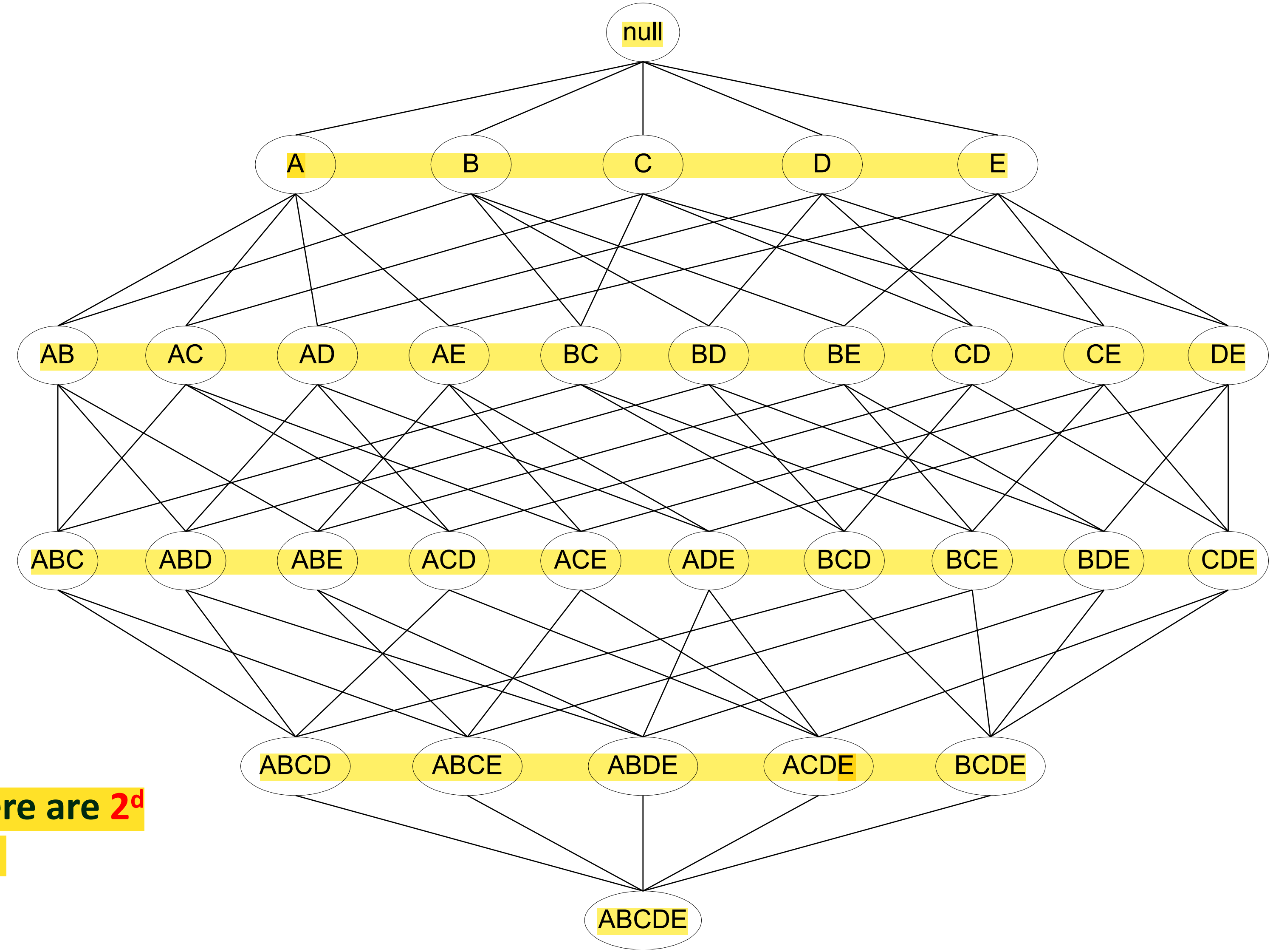
- Stated differently:

  Count the number of times combinations of attributes occur in the data. If the count of a combination is above ***minsup***, report it

- **Recall**: The input is a transaction database ***D*** where every transaction consists of a subset of items from some universe ***I***

Stockholms universitet

How many itemsets are there?



Given d items, there are 2^d possible itemsets, including *null*

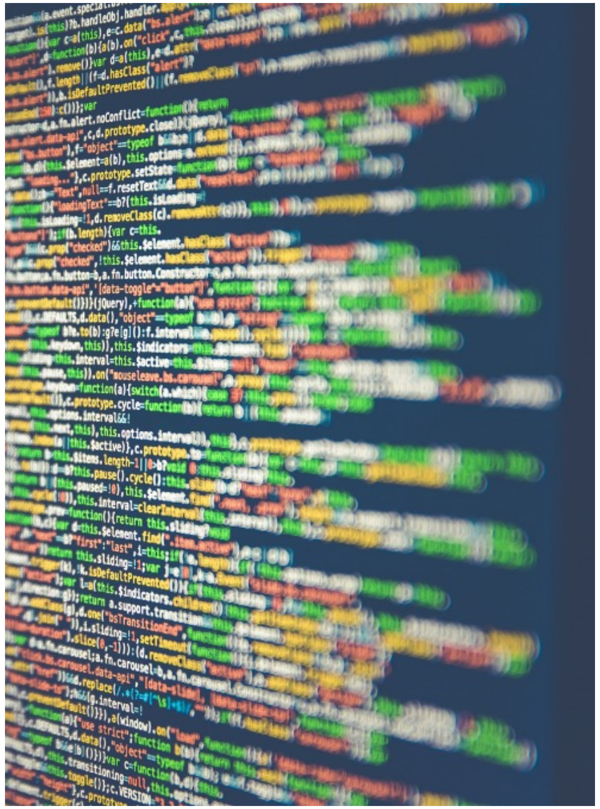
Stockholms universitet

Brute-force algorithm for finding all frequent itemsets?

- Generate all possible itemsets (lattice of itemsets)
  - Start with 1-itemsets, 2-itemsets, ... , d-itemsets
- Compute the frequency of each itemset from the data
  - Count in how many transactions each itemset occurs
- If the support of an itemset is above ***minsup*** report it as a frequent itemset

Brute-force algorithm for finding all frequent itemsets?

- Complexity?
  - Match every candidate against each transaction
  - For M candidates and N transactions, the complexity is approximately **O(NM)**
    - Expensive since $M = 2^d$!!!



Speeding-up the brute-force algorithm

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- 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
- Reduce the number of transactions (N)
  - Reduce the size of N as the size of itemsets increases
  - Use vertical-partitioning of the data to apply the mining algorithms

Stockholms universitet

Reduce the number of candidates

- **Apriori principle** (Main observation):
  - 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:

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

– The support of an itemset ***never exceeds*** the support of its subsets
– This is known as the ***anti-monotone*** property of support

Example

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

s(Bread) ≥ s(Bread, Beer)
s(Milk) ≥ s(Bread, Milk)
s(Diaper, Beer) ≥ s(Diaper, Beer, Coke)

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

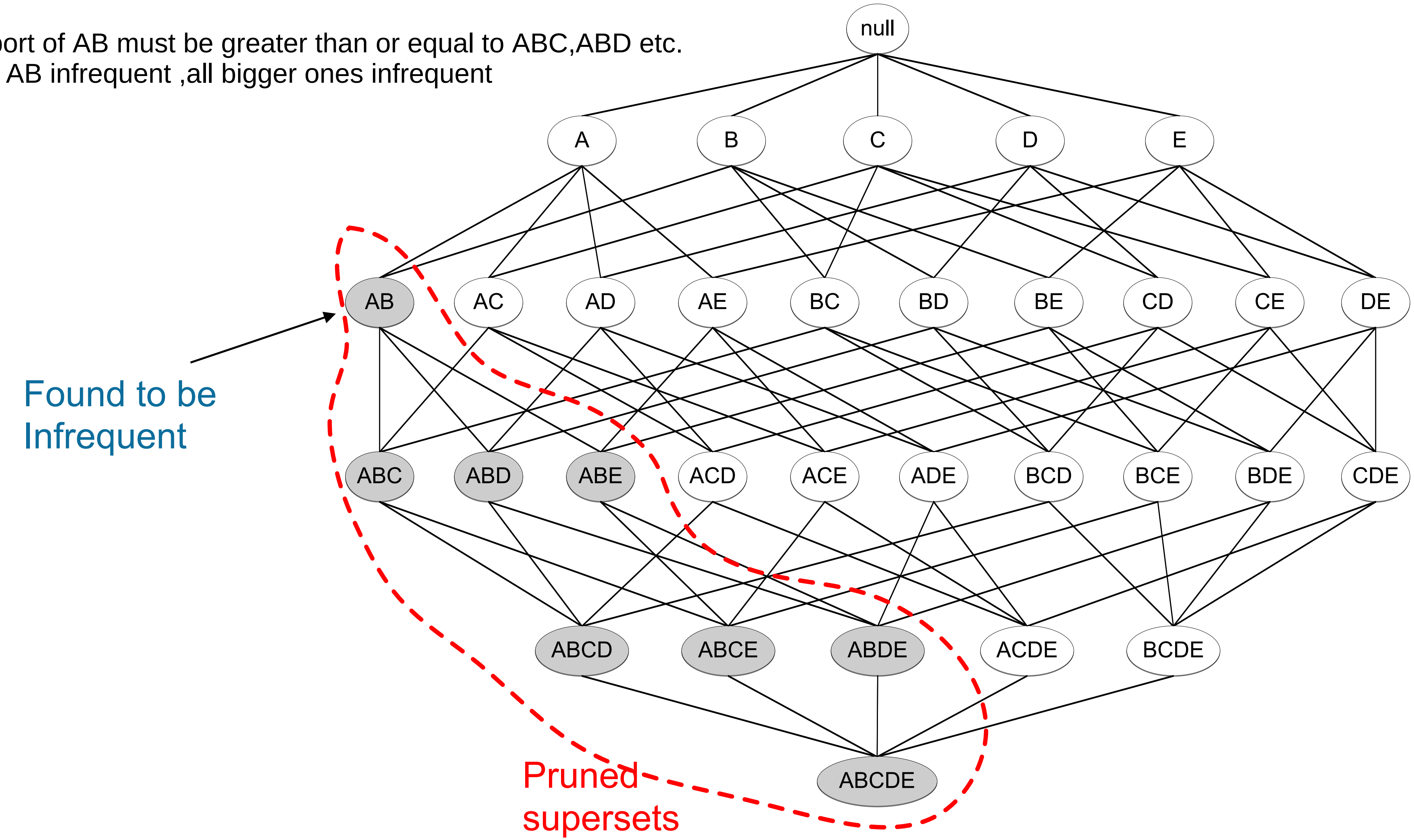
itemset infrequent all supersets also infrequent

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

Stockholms universitet

Illustrating the Apriori principle

support of AB must be greater than or equal to ABC,ABD etc.
So if AB infrequent ,all bigger ones infrequent



The Apriori algorithm [Agrawal&Srikant 1994]

C_k: Candidate itemsets of size k
L_k: Frequent itemsets of size k

L_1 = {frequent 1-itemsets};
for (k = 1; L_k != ∅; k++)
  C_{k+1} = GenerateCandidates(L_k)
  for each transaction t in the database do
    increment count of candidates in C_{k+1} that are contained in **t**
  endfor
  L_{k+1} = candidates in C_{k+1} with support ≥***minsup***
endfor
return $\cup_k L_k$;

Stockholms universitet

C_{k+1} = GenerateCandidates()

- Assume the items in L_k are listed in an order (e.g., alphabetical)
- **Step 1**: ***self-joining L_k*** *(IN SQL)*

  insert into C_{k+1}

  select $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$

  from L_k p, L_k q

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

$$p = \{\boxed{\mathbf{item_1, item_2, \ldots, item_{k-1}}}, \mathit{item_k}\}$$

$$= \qquad \neq$$

$$q = \{\boxed{\mathbf{item_1, item_2, \ldots, item_{k-1}}}, \mathit{item_k}\}$$

$$\text{candidate} = \{\mathbf{item_1, item_2, \ldots, item_{k-1}}, \mathit{item_k}, \mathit{item_k}\}$$

Stockholms universitet

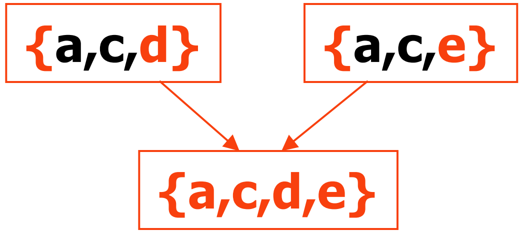
Example of Candidate Generation

- $k=3$
- $L_3=\{abc, abd, acd, ace, bcd\}$
- $C_4 = ?$

Stockholms universitet

Example of Candidate Generation

- *k=3*
- L_3={**abc**, **abd**, **acd**, **ace**, *bcd*}
- ***Self-joining***: $L_3 * L_3$
  - ***abcd*** from ***ab****c* and ***ab****d*
  - ***acde*** from ***ac****d* and ***ac****e*



C_{k+1} = GenerateCandidates()

- Assume the items in L_k are listed in an order (e.g., alphabetical)
- **Step 2**: ***pruning***

forall ***itemsets c in*** C_{k+1} do

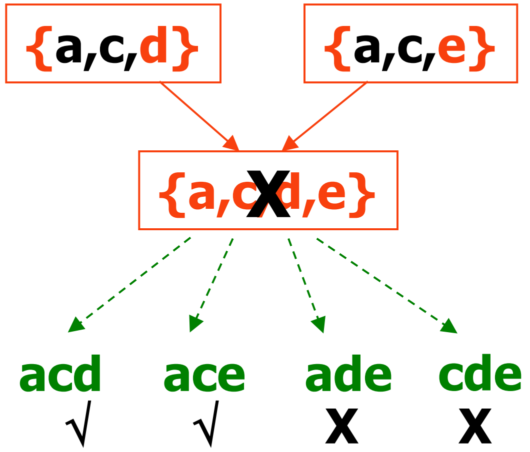
forall ***k-subsets*** s ***of*** c do

if (s *is* ***not*** *in* L_k**) then delete** c **from** C_{k+1}

Stockholms universitet

Example of Candidate Pruning

- L_3=*{abc, abd, acd, ace, bcd}*
- ***Self-joining***: $L_3 * L_3$
  - ***abcd*** from ***abc*** and ***abd***
  - ***acde*** from ***acd*** and ***ace***
- ***Pruning***
  - ***acde*** is removed because ***ade*** is not in L_3
- C_4=***{abcd}***



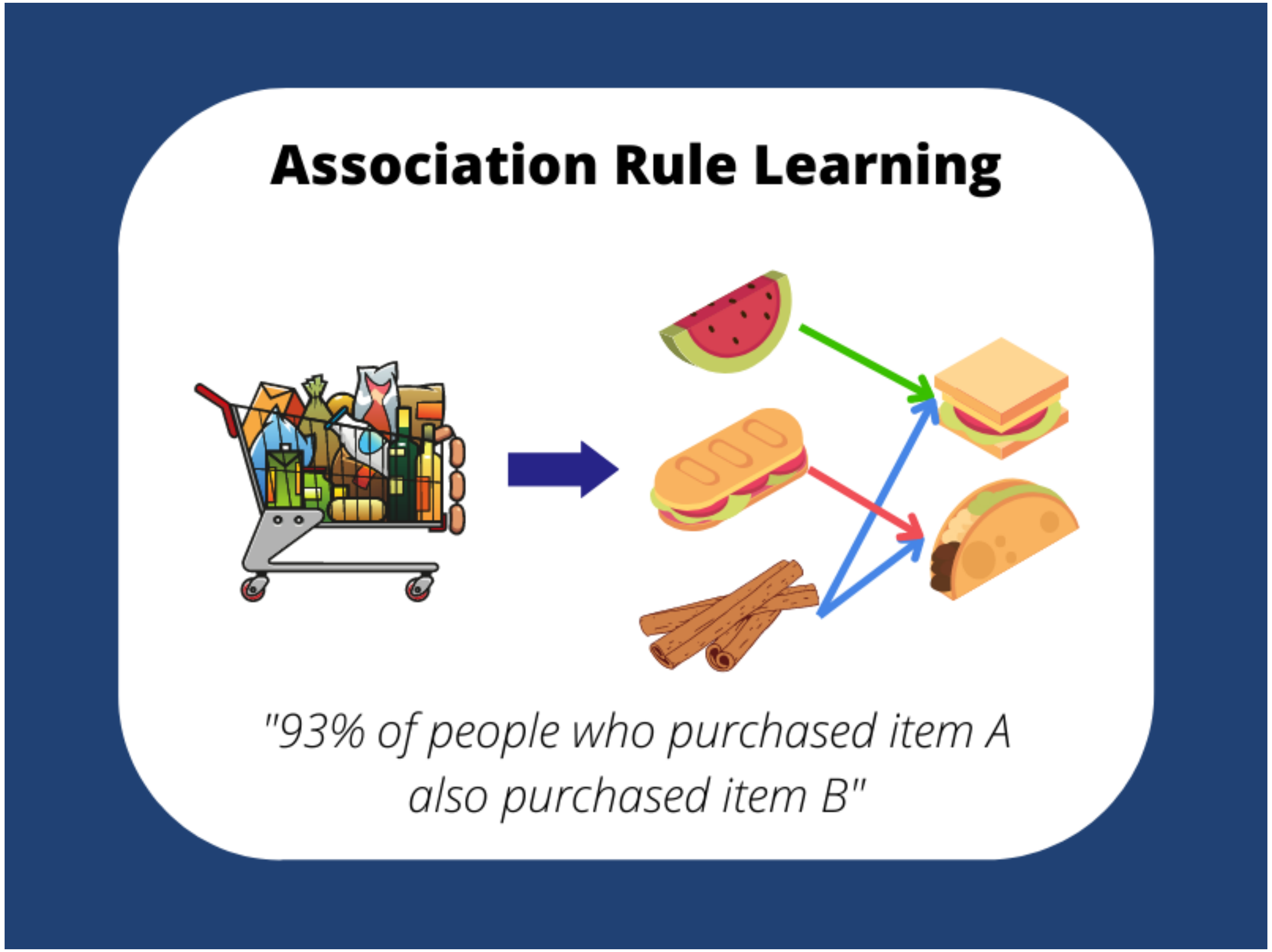
Discussion of the Apriori algorithm

- Much faster than the Brute-force algorithm
  - It avoids checking all elements in the lattice
- The running time is in the worst case **$O(2^d)$**
  - Pruning really prunes in practice
- It makes multiple passes over the dataset
  - One pass for every level **k**
- Multiple passes over the dataset are inefficient when we have thousands of candidates and millions of transactions

Stockholms universitet

Implementations

- Lots of them around
- See, for example, the web page of Bart Goethals: http://www.adrem.ua.ac.be/~goethals/software/
- Typical input format: each row lists the items (using item id's) that appear in every row



Stockholms universitet

Association Rules

- Let **D** be a database of transactions, e.g.,

TID	*Items*
1	**A, B, C**
2	**A, C**
3	**A, D**
4	**B, E, F**

- Let ***I*** be the set of items that appear in the database, e.g., $I=\{A,B,C,D,E,F\}$
- A rule is defined by $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$
  - e.g., $\{B,C\} \rightarrow \{A\}$ is a rule

Stockholms universitet

Definitions

Association Rule

- An implication expression of the form **X** $\rightarrow$ **Y**, where **X** and **Y** are non-overlapping itemsets
  - Example:
    {Milk, Diaper} $\rightarrow$ *{Beer}*

Rule Evaluation Metrics

- **Support (s)**
  - Fraction of transactions that contain both **X** and **Y**
- **Confidence (c)**
  - Measures how often items in **Y** appear in transactions that also contain **X**

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

Example:

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**

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

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

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

Stockholms universitet

Mining association rules

- **Task**:

  Given a set of transactions ***D***, the goal of association rule mining is to find all rules having:

  - support ≥ ***minsup*** threshold
  - confidence ≥ ***minconf*** threshold

Stockholms universitet

Brute-force algorithm for association-rule mining

- 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!

A faster solution…

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)

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

Stockholms universitet

A faster solution…

- Two-step approach:
  - **Frequent Itemset Generation**

    Generate all itemsets whose support ≥ ***minsup***

    solved

  - **Rule Generation**

    Generate high-confidence rules from each frequent itemset, where each rule is a binary partition of a frequent itemset

Stockholms universitet

Rule Generation – Naive algorithm

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

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

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

Stockholms universitet

Efficient rule generation

- How to *efficiently* generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property
    - **c(ABC → D)** can be larger or smaller than **c(AB → D)**
  - ***But the confidence of rules generated from the same itemset has an anti-monotone property***
  - Example: X = {A,B,C,D}:

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

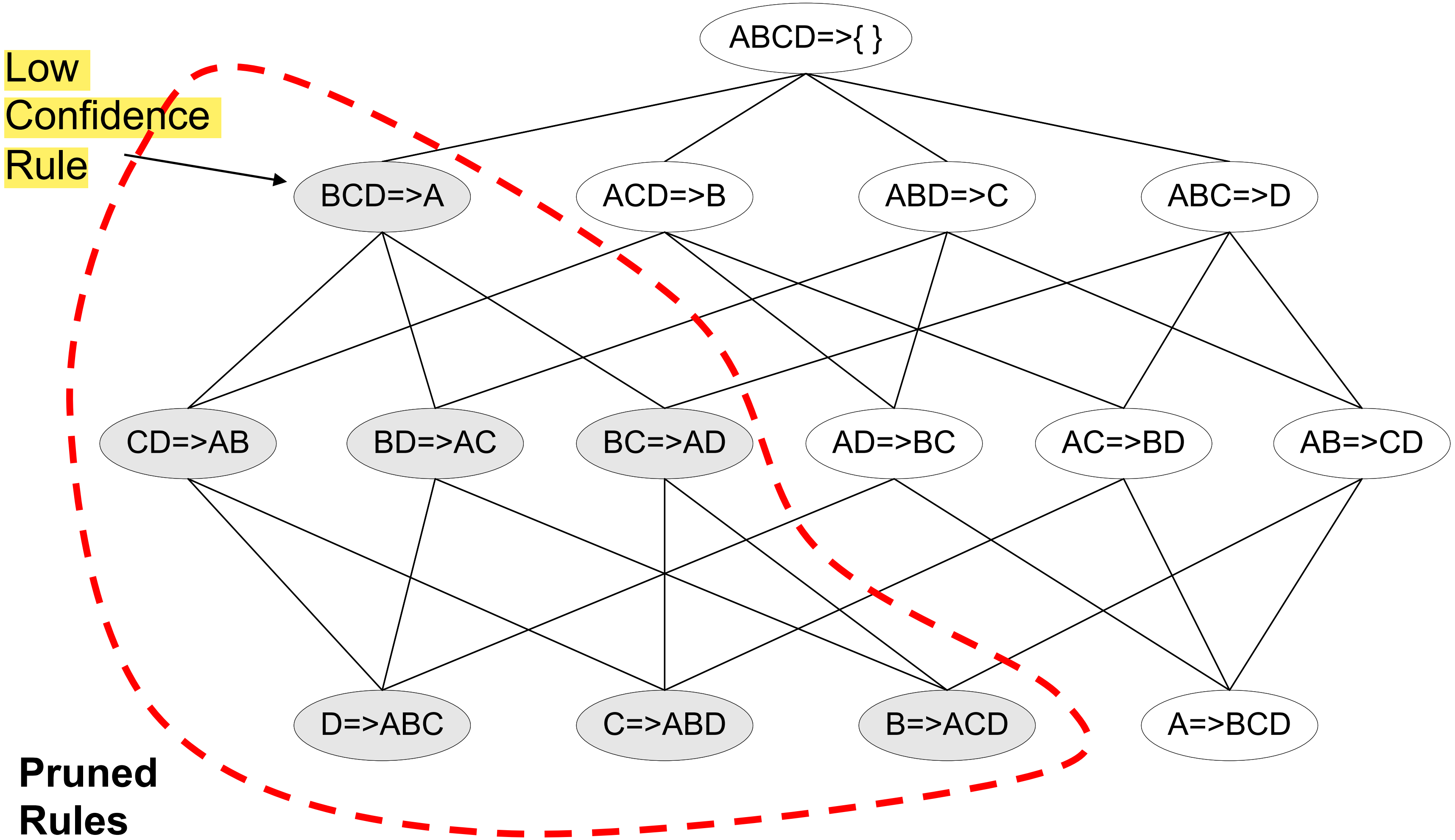
  - **Why?**

$$c(ABC->D) = \frac{\sigma(\{A,B,C,D\})}{\sigma(\{A,B,C\})}$$

Confidence is anti-monotone w.r.t. number of items on the Right-Hand-Side of the rule

Stockholms universitet

Rule Generation for Apriori Algorithm



Stockholms universitet

Other interestingness measures

- Lift
- Cosine
- All-confidence
- Leverage
- …

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

Number of people that drink coffee but not tea

Stockholms universitet

Drawback of Confidence

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

Association Rule: Tea → Coffee

- **Confidence:** $P(Coffee|Tea) = \frac{15}{20} = 0.75$, but $P(Coffee) = \frac{90}{100} = 0.9$
  - Although confidence is high, the rule is misleading
    - someone buying tea is less likely to buy coffee than someone for whom we have no information
    - $P(Coffee|\overline{Tea}) = 0.9375$

Stockholms universitet

Lift of a Rule

Given association rule X → Y: $lift(X \rightarrow Y) = \dfrac{P(X,Y)}{P(X)P(Y)} = \dfrac{conf(X \rightarrow Y)}{P(Y)} = \dfrac{|X,Y|n}{|X||Y|}$

- If P(X,Y) = P(X) P(Y), then lift(X→Y) = 1
  - means that the occurrences of X and Y in the same transaction are independent events, i.e., X and Y are not correlated, and the rule is meaningless!
- lift(X→Y) = k > 1
  - means that X and Y are dependent so that if X occurs, then Y is k times more likely to occur than expected
- lift(X→Y) = k < 1
  - means that the occurrence of X prevents the occurrence of Y, i.e., Y is k times less likely to occur than expected

Stockholms universitet

Example: Lift/Interest

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

Association Rule: Tea $\rightarrow$ Coffee

Confidence: $P(Coffee|Tea) = \frac{15}{20} = 0.75$, but $P(Coffee) = \frac{90}{100} = 0.9$

$\Rightarrow$ Lift = 0.75 / 0.9= 0.8333 (< 1, thus it is negatively associated)

$\Rightarrow$ therefore, *coffee occurs less than expected when tea also occurs*

Stockholms universitet

Too many frequent itemsets

- If $\{a_1, \ldots, a_{100}\}$ is a frequent itemset, then there are

$$\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1$$

$1.27 \cdot 10^{30}$ frequent sub-patterns!

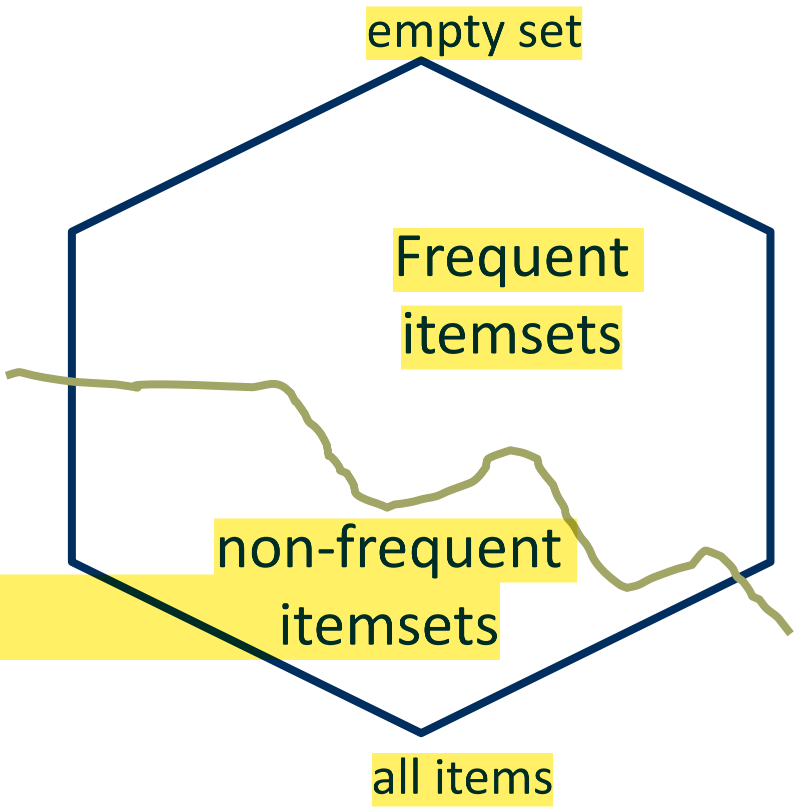
- There should be some more **condensed** way to describe the data

Stockholms universitet

Frequent itemsets maybe too many to be helpful

- If there are many & large frequent itemsets, enumerating all of them is costly!
- We may be interested in finding the ***boundary*** frequent patterns

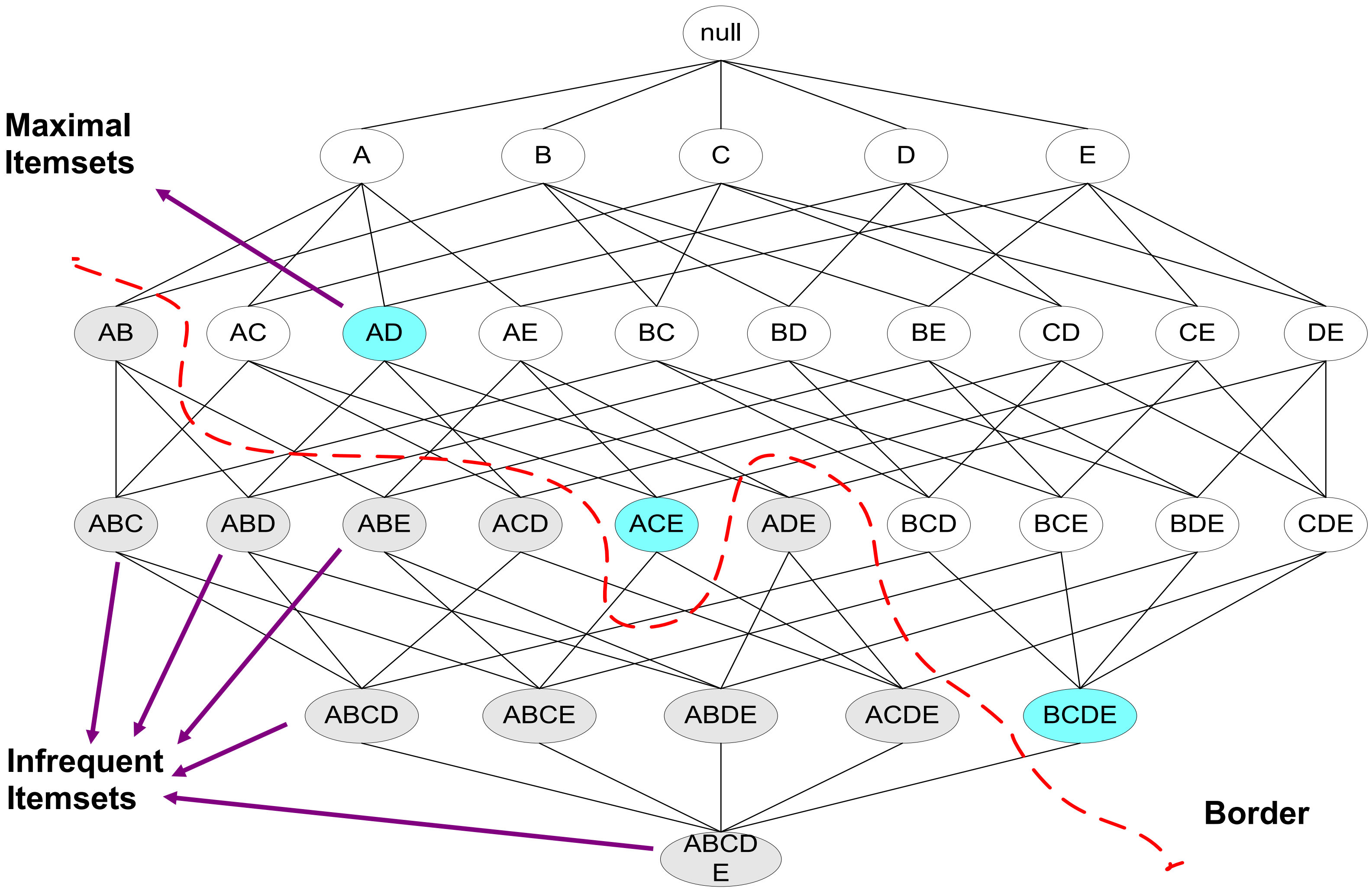
Question: Is there a good definition of such boundary?



Stockholms universitet

Maximal frequent itemsets

An itemset is **maximal frequent** if none of its immediate supersets is frequent



Stockholms universitet

Descriptive power of maximal patterns

- Knowing the set of all maximal patterns allows us to reconstruct the set of all frequent itemsets!!

- We can only reconstruct the set, not the actual frequencies

Stockholms universitet

Closed itemsets

An itemset is **closed** if **none of its immediate supersets** has the same support as the itemset

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

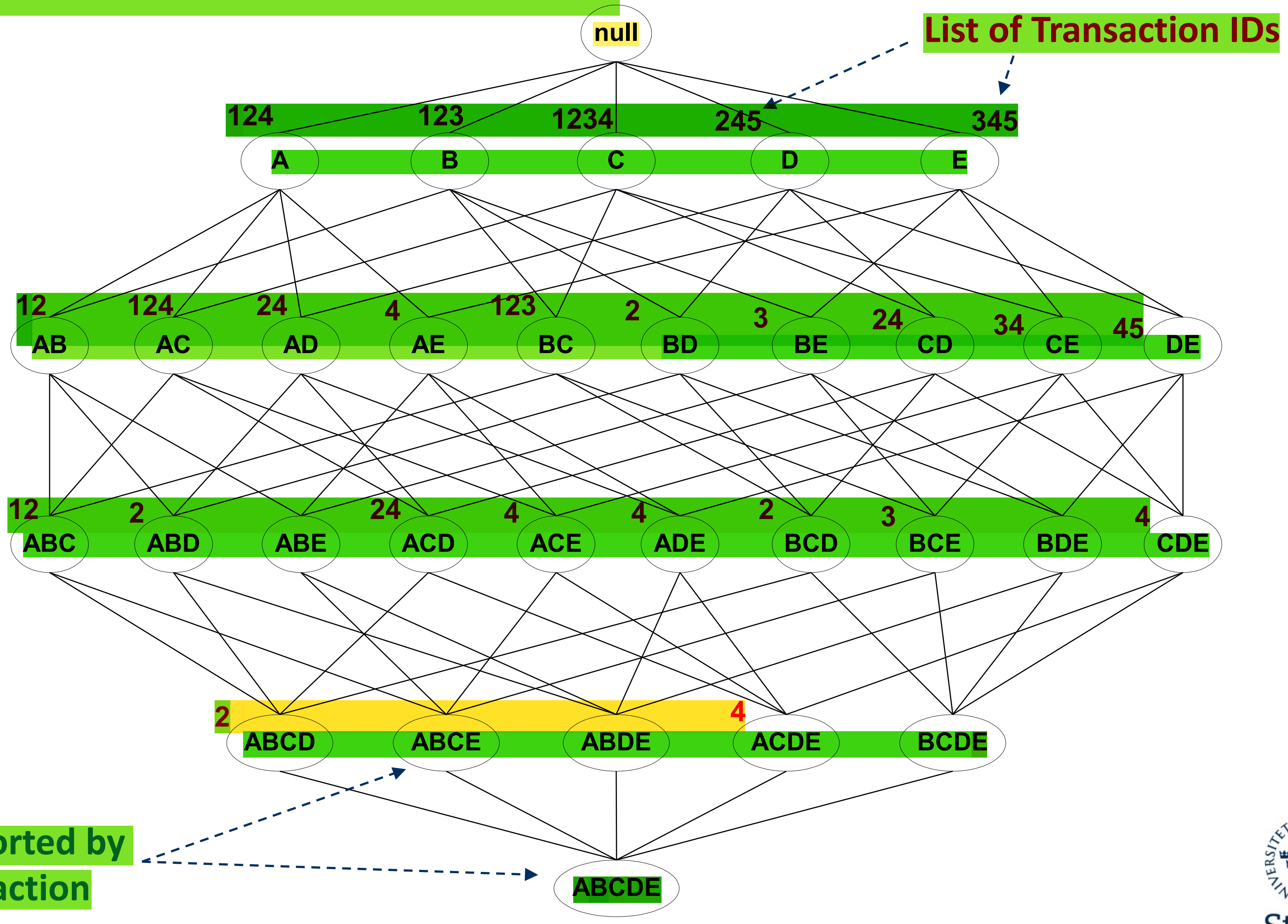
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Closed itemsets: green color

Stockholms universitet

Maximal vs Closed itemsets

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



Stockholms universitet

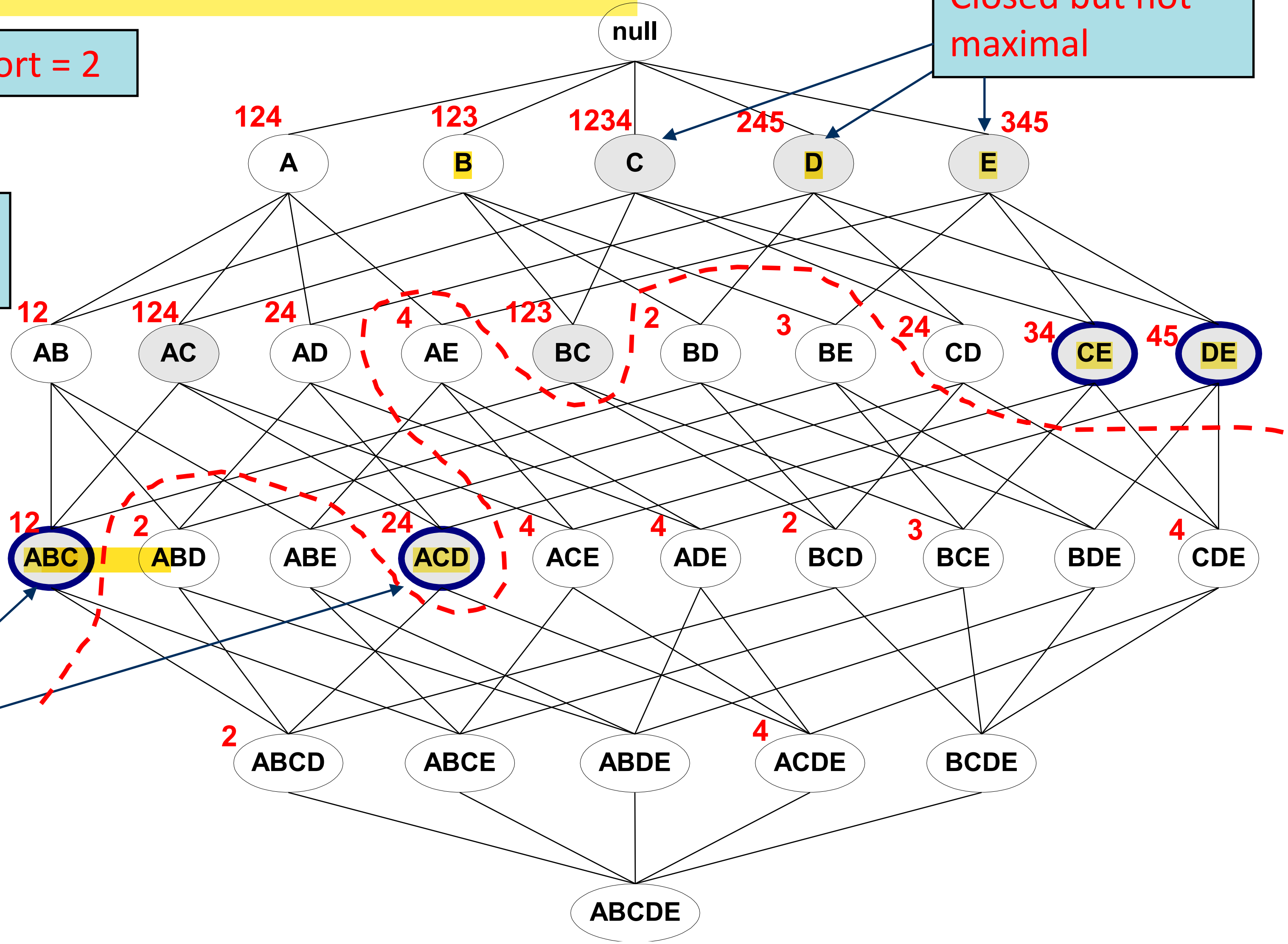
Maximal vs Closed itemsets

Minimum absolute support = 2

Closed = 9
Maximal = 4

Closed but not maximal

Closed and maximal



Stockholms universitet

Why are closed itemsets interesting?

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

- **Proof:** Assume a frequent itemset ***X***:
    - ***X*** is closed → ***s(X)*** is known
    - ***X*** is not closed →

      s(X) = max {s(Y) | Y is closed and X subset of Y}

Stockholms universitet

Closed itemsets

An itemset is **closed** if **none of its immediate supersets** has the same support as the itemset

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}	?
{B}	5
{C}	?
{D}	?
{A,B}	4
{A,C}	?
{A,D}	?
{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

s(X) = max {s(Y) | Y is closed and X subset of Y}

Closed itemsets

An itemset is **closed** if **none of its immediate supersets** has the same support as the itemset

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}	?
{B}	5
{C}	?
{D}	?
{A,B}	4
{A,C}	?
{A,D}	?
{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

s(X) = max {s(Y) | Y is closed and X subset of Y}

Stockholms universitet

Closed itemsets

An itemset is **closed** if **none of its immediate supersets** has the same support as the itemset

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}	?
{B}	5
{C}	3
{D}	?
{A,B}	4
{A,C}	?
{A,D}	?
{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

s(X) = max {s(Y) | Y is closed and X subset of Y}

Closed itemsets

An itemset is **closed** if **none of its immediate supersets** has the same support as the itemset

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}	?
{B}	5
{C}	3
{D}	?
{A,B}	4
{A,C}	?
{A,D}	?
{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

s(X) = max {s(Y) | Y is closed and X subset of Y}

Stockholms universitet

Closed itemsets

An itemset is **closed** if **none of its immediate supersets** has the same support as the itemset

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}	?
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	?
{A,D}	?
{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

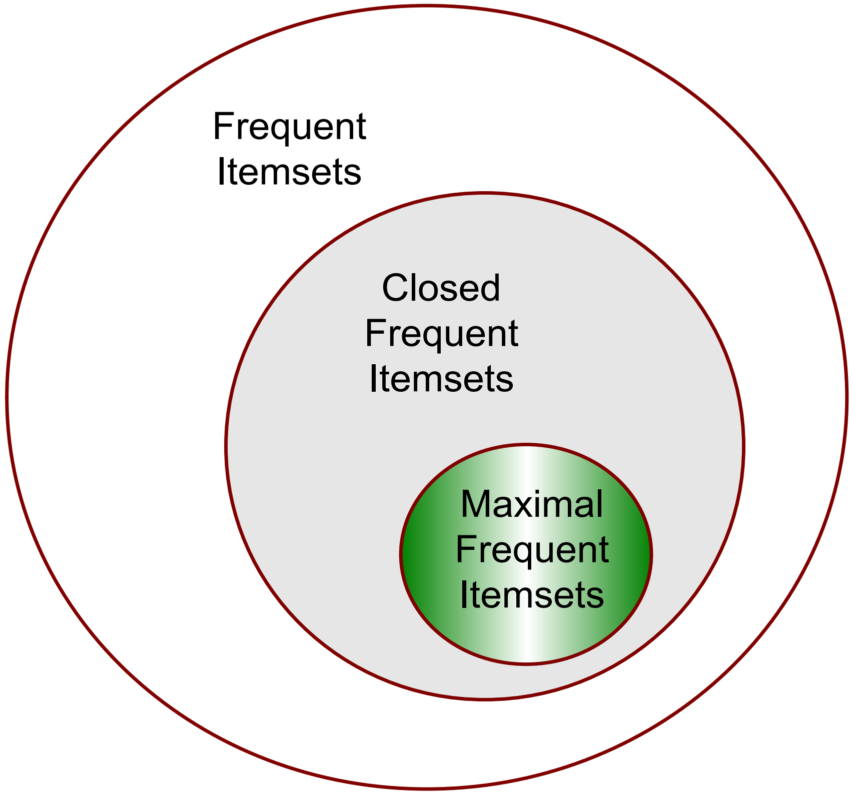
s(X) = max {s(Y) | Y is closed and X subset of Y}

Stockholms universitet

Maximal vs Closed itemsets

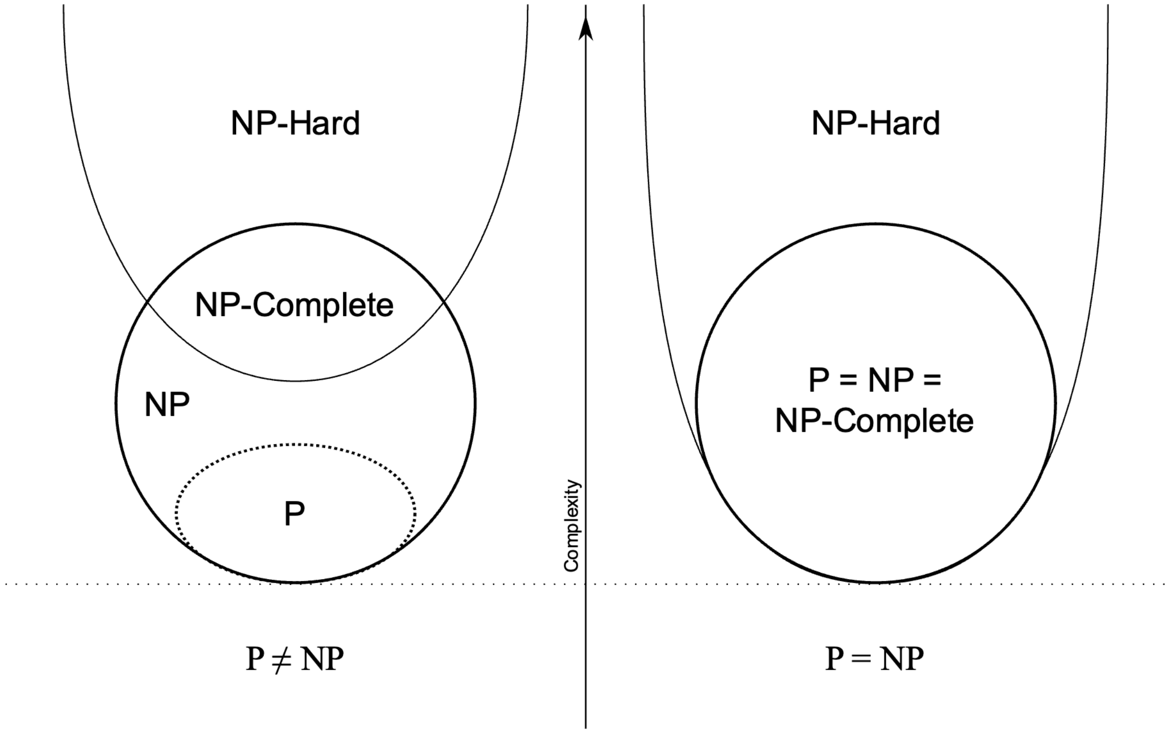
- Knowing all maximal patterns (and their frequencies) allows us to reconstruct the set of frequent itemsets
- Knowing all closed patterns and their frequencies allows us to reconstruct the set of all frequent itemsets and their frequencies
- The problems of finding the set of frequent itemsets, maximal itemsets, and closed itemsets are all **NP-hard**

Proof?



Stockholms universitet

Back to the theory of complexity



NP: a solution can be verified in polynomial time

NP-hard: every NP problem can be reduced to it in polynomial time (the problem can be used as a subroutine to solve any NP problem)

NP-complete: the problem is in NP and it is NP-hard

Stockholms universitet

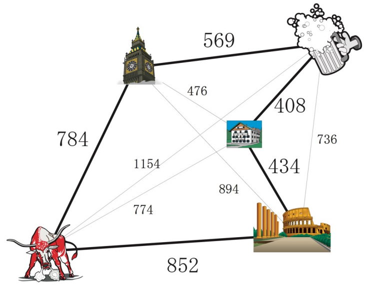
Examples of NP-hard problems

- Subset sum: given a set of integers is there a non-empty set of those integers that adds up to zero?

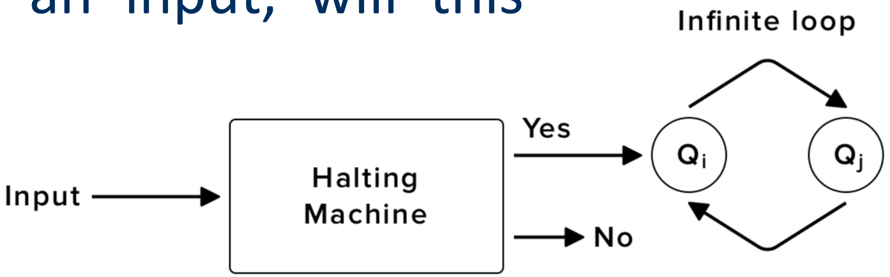
{-7, -3, -2, 5, 8}

{ -3, -2, 5 }

- Traveling salesman: given a set of cities and the distances between the cities, what is the possible shortest route visiting each and every city (once) and returning to the starting city?



- Halting problem: given a program and an input, will this program run for ever?



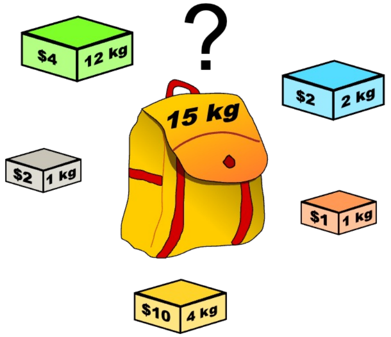
Examples of NP-complete problems

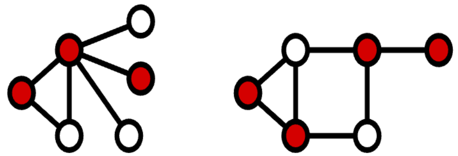
- Boolean satisfiability (SAT): given a boolean formula, is there a configuration of the variables that satisfies it?
- Knapsack problem: given a set of items, each having a value and a weight, find the subset of items so that the weight is **at most W** (minimized) and the total value is **at least V** (maximized)
- Vertex cover: given a graph, find a set of **at most k vertices** so that each and every edge in the graph is incident to at least one of these vertices
- Maximum Clique: given a graph, find the largest complete subgraph of **size at most k**

$(x \text{ OR } y \text{ OR } z) \text{ AND } (x \text{ OR } \bar{y} \text{ OR } z) \text{ AND}$

$(x \text{ OR } y \text{ OR } \bar{z}) \text{ AND } (x \text{ OR } \bar{y} \text{ OR } \bar{z}) \text{ AND}$

$(\bar{x} \text{ OR } y \text{ OR } z) \text{ AND } (\bar{x} \text{ OR } \bar{y} \text{ OR } \bar{z})$





Stockholms universitet

Proving NP-hardness

- Not always easy
- But sometimes easier than you think
- To prove that problem **T** is NP-hard:
  - **find** a known NP-hard problem **L**
  - **reduce** **L** to **T** in polynomial time: show that all instances of **L** can be solved by solving **T**, which is done by mapping **L** to **T** in polynomial time
  - since **L** is NP-hard, **T** cannot be solved faster than that…hence it is also NP-hard
- If we can also **verify** any solution to **T** in polynomial time, then **T** is in NP; hence **T** is also NP-complete

Stockholms universitet

Maximal frequent itemset mining

- *Use the set-cover problem formulation!*
  - universe of **m** elements **$U = \{U_1,\ldots,U_m\}$:** each **U_i** is a frequent itemset
  - set of **n** sets **$S = \{S_1,\ldots,S_n\}$:** each **S_i** contains all itemsets covered by a given frequent itemset X, i.e., all subsets of X
- **Formulation:**
  - **Problem A:** find the **smallest collection** of sets in **S** such that **all elements in U** are **covered** (set cover)
  - **equivalent to Problem B:** find the **smallest collection** of itemsets each covering a set of **frequent itemsets**, so that ***all frequent itemsets*** are **covered** (maximal itemsets)
  - so…if we can solve **Problem B** we can also solve all instances of **Problem A**
  - since A is known to be NP-hard, then B is also NP-hard

Stockholms universitet

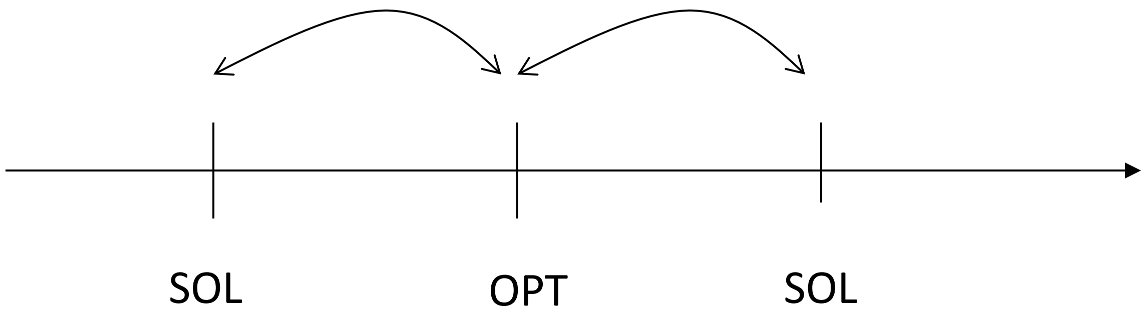
Approximation algorithms

For an NP-hard problem, we cannot compute an optimal solution in polynomial time

The key of designing a polynomial time approximation algorithm is to obtain a good (lower or upper) bound to the optimal solution

- **OPT:** value of an optimal solution
- **SOL:** value of the solution that our algorithm returns

The general strategy (for an optimization problem) is:



$f \cdot \text{OPT} \leq \text{SOL} \leq \text{OPT}, \quad \text{if } f < 1$

Maximization Problem:
maximize some cost function

$\text{OPT} \leq \text{SOL} \leq f \cdot \text{OPT}, \quad \text{if } f > 1$

Minimization Problem:
minimize some cost function

Approximation for set cover

The greedy algorithm for set cover has an approximation factor of:

$f = |s_{max}|$

s_{max}: the size of the largest set

Proof: From CLR "Introduction to Algorithms"

The set cover cannot be approximated with a factor better than **$O(\log |s_{max}|)$**

Stockholms universitet

Best-collection problem

- Universe of **m** elements **U = {U_1,…,U_m}**
- A set of **n** sets **S = {S_1,…,S_n}** such that they cover the universe
- **Question:** Find a collection **C** consisting of **k** sets from **S** such that

$f(C) = |\bigcup_{c \in C} c|$ is ***maximized***

f: the number of elements in **U** that are covered by **C**

- The best-collection problem is NP-hard

Stockholms universitet

Greedy approximation algorithm for the best-collection problem

- **C = {}**
- **for every** set **s** in **S** and ***not*** in **C** compute the gain of **s**:

  g(s) = f(C U {s}) – f(C)

- Select the set **s** with the ***maximum*** gain
- **C = C U {s}**
- **Repeat until C** has **k** elements

The ***greedy*** algorithm for the best-collection problem has the following approximation factor:

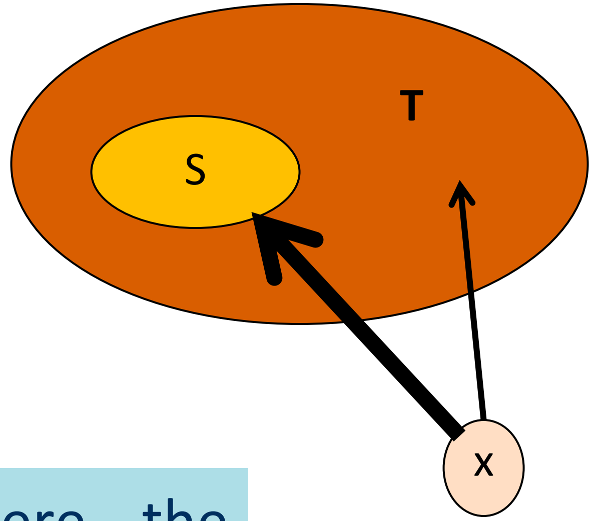
F = (e-1)/e = 0.6321

Stockholms universitet

Submodular functions

- A function **f** (defined on sets of some universe) is **submodular** if
  - ***for all*** sets **S**, **T,** such that **S** is ***subset*** of **T** and **x** ***any*** element in the universe
  - **f(S U {x}) – f(S) ≥ f(T U {x}) – f(T)**

adding an element to a smaller subset will have a higher gain than adding it to a superset

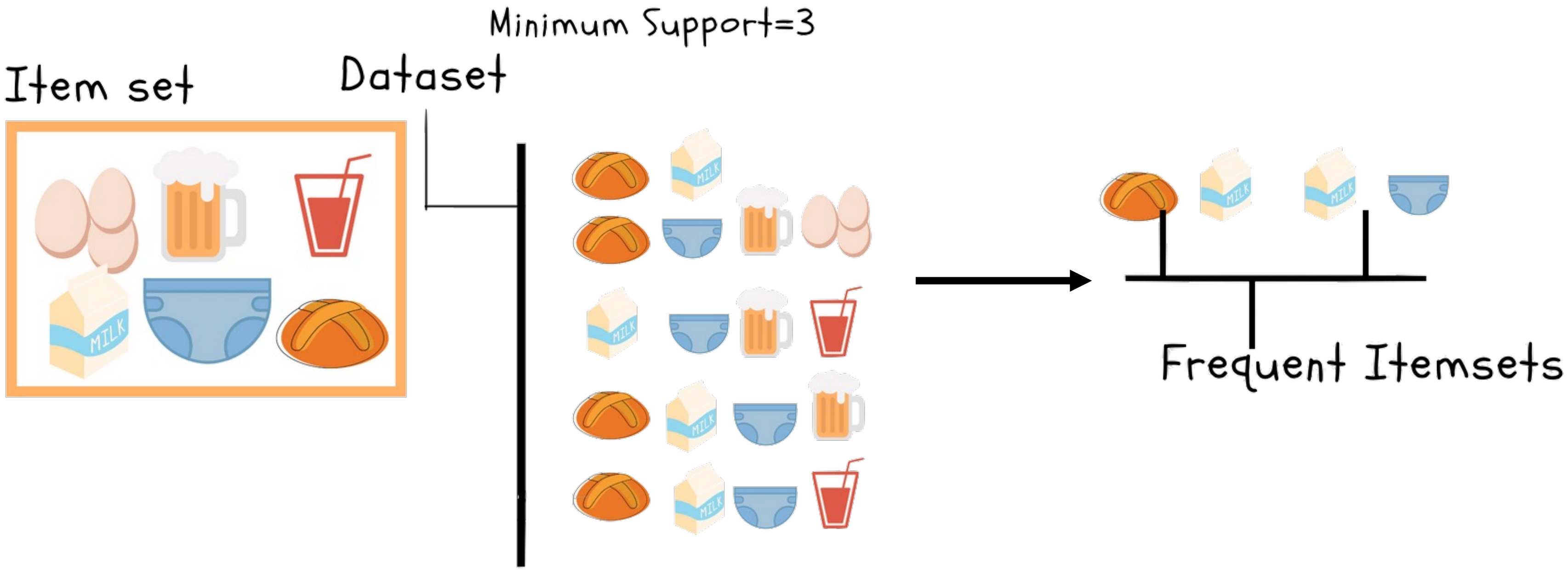


Theorem: For **all maximization** problems where the optimization function is **submodular**, the ***greedy*** algorithm has the following approximation factor

F = (e-1)/e = 0.6321

Today…

- What is **Itemset Mining**?
- What is the **Apriori Principle**?
- How do we use the **Apriori Algorithm**?
- The importance of **Maximal** and **Closed** Itemsets
- The **set cover problem** and **Approximation Algorithms**



Stockholms universitet

TODOs

Reading:

Main course book: Chapter 3.5

Extra Material

Lab 0

Recommended to complete the lab before the end of the week

Quiz 1

Stockholms universitet

Coming up next week

Monday

Lecture 3 – Dimensionality Reduction

Wednesday

Lab 1 – Data Exploration

Thursday

Lecture 4 – Clustering I