Data Intensive Systems (DIS) KBH-SW7 E25

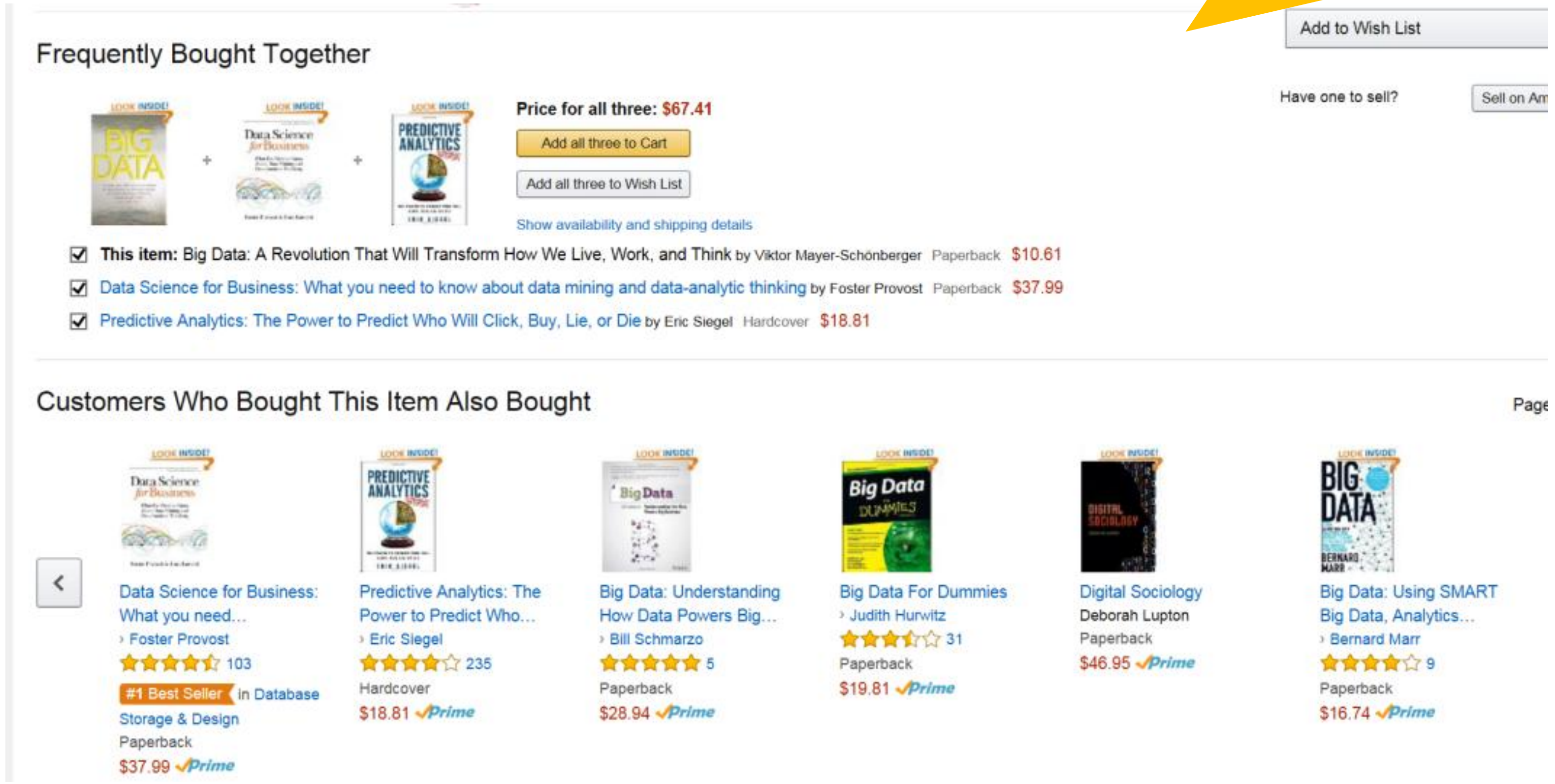
9. Association Rules

AALBORG UNIVERSITY

A Real Application of Association Rules

- Amazon's recommendation
  - 90% buyers who bought A also bought B.
  - Since you've bought A, you may also want B.

Make recommendations based on rules of high support, confidence and lift.



Frequently Bought Together

Price for all three: $67.41

Add all three to Cart

Add all three to Wish List

Show availability and shipping details

☑ **This item:** Big Data: A Revolution That Will Transform How We Live, Work, and Think by Viktor Mayer-Schönberger Paperback $10.61

☑ Data Science for Business: What you need to know about data mining and data-analytic thinking by Foster Provost Paperback $37.99

☑ Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die by Eric Siegel Hardcover $18.81

Add to Wish List

Have one to sell? Sell on Am

Customers Who Bought This Item Also Bought

Page

Data Science for Business: What you need...	Predictive Analytics: The Power to Predict Who...	Big Data: Understanding How Data Powers Big...	Big Data For Dummies	Digital Sociology	Big Data: Using SMART Big Data, Analytics...
› Foster Provost	› Eric Siegel	› Bill Schmarzo	› Judith Hurwitz	Deborah Lupton	› Bernard Marr
103	235	5	31	Paperback	9
#1 Best Seller in Database Storage & Design	Hardcover	Paperback	Paperback	$46.95 Prime	Paperback
Paperback	$18.81 Prime	$28.94 Prime	$19.81 Prime		$16.74 Prime
$37.99 Prime					

Agenda

- Problem definition
  - Support, confidence, lift, and association rule
  - Frequent itemsets
  - Steps for association rule mining
- Apriori principle, Apriori algorithm
- Deriving association rules from frequent itemsets
- Reflection on Apriori algorithm

Market Basket Data

- Large set of *items*, i.e., things sold in a supermarket
- Large set of *baskets*, each a small subset of items, i.e., things that one customer buys in one **transaction**
- Transaction table **T**: market-basket data
  - Each record is a transaction, containing a set of items
  - Many-to-many mapping (association) between items and baskets
- What can we do with this type of data?
  - E.g., counting whether the combination {Milk, Bread} is *frequent* or not

TID	Items
1	{Milk, Bread, Beer, Diapers}
2	{Bread, Eggs}
3	{Bread, Diapers}
4	{Milk, Bread, Cola}
5	{Milk, Bread, Diapers}

Transaction table

What Is Association Rule Mining?

- Finding frequent patterns and associations (rules) among sets of items in a transaction table
- Motivation (market basket analysis):
  - How likely is that the customers buying *milk* are also buying *bread*?
  - Such rules help retailers making decisions
    - Plan the shelf space: placing milk close to bread, more convenient for the customers
    - Offer promotions/discounts for those products together

What Is an Association Rule?

- An **association rule** correlates (associates) the presence of one set of items with that of another set of items
- Examples
  - Rule form: Body $\Rightarrow$ Head [support, confidence]
  - milk $\Rightarrow$ bread [5%, 70%]
    - 5% of transactions buy both milk and bread
    - transactions that buy milk have 70% chance of buying also bread
- Applications: basket data analysis, catalog design
  - * $\Rightarrow$ *chocolate* (How to boost the sales of chocolate)
  - *Home Electronics* $\Rightarrow$ * (What other products should the store stock up?)

Rule Components

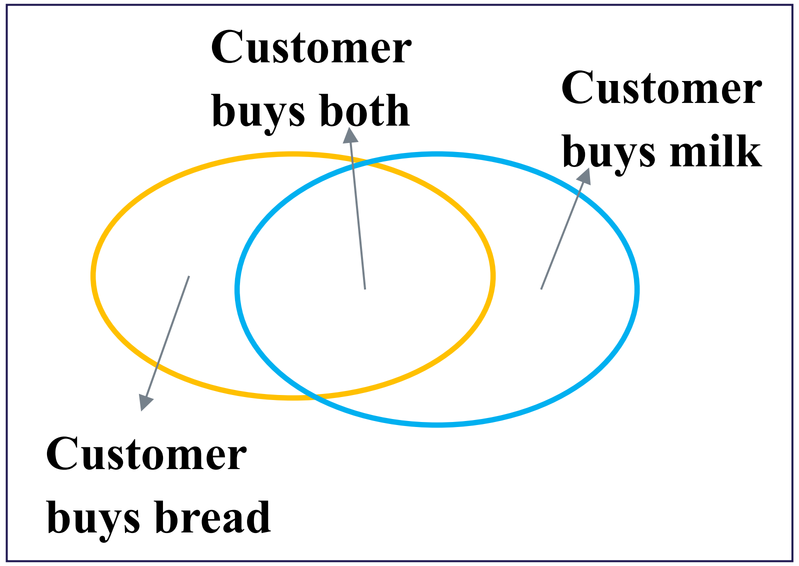
- An itemset means a set of items
  - E.g., {a, b, c}
- Let T be a collection of transactions
  - E.g., T = {TID 1, TID 2, TID 3, TID 4}
- Let I be the set of items that appear in the database, e.g., I = {a, b, c, g, e, f}
- A rule is defined by $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$
  - E.g., {b,c} $\Rightarrow$ {e} is a rule. We may simplify it as bc $\Rightarrow$ e if the context is clear.
  - E.g., {b,c} $\Rightarrow$ {c, e} is not a rule

TID	Items
1	a,b,c
2	a,c
3	a,g
4	b,e,f

Interesting Rules

- A rule is said to be interesting (or valid) when:
  - Its items appear frequently in the transaction table (support)
  - It holds with a high probability (confidence)

Example: *milk* $\Rightarrow$ *bread*



NB: X and Y are itemsets.

Find all the rules $X \Rightarrow Y$ with confidence and support above given thresholds

- **support** *s*, probability that a transaction contains $X \cup Y$
- **confidence** *c*, conditional probability that a transaction having X also contains Y

Example (1)

- Find the support and confidence of the rule: {B,D} $\Rightarrow$ {A}
- Support value of *sup*(ABD):
  - percentage of tuples with {A,B,D}

    = (3/4)*100% = 75%
- Confidence value of *conf*(BD $\Rightarrow$ A)

$$\frac{\text{number of transactions that contain } \{A,B,D\}}{\text{number of transactions that contain } \{B,D\}} = \frac{3}{3} = 100\%$$

TID	items_bought
100	{F,A,D,B}
200	{D,A,C,E,B}
300	{C,A,B,E}
400	{B,A,D}

$$\text{prob}(Y \mid X) = \frac{\text{prob}(X \cup Y)}{\text{prob}(X)}$$

$$\text{conf}(X \Rightarrow Y) = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)} = \frac{\text{frequency}(X \cup Y)}{\text{frequency}(X)}$$

Example (1)

- Find interesting rules

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Thresholds:

Min. support 50%
Min. confidence 50%

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

"$A \Rightarrow C$" is a valid rule because:

support = support($\{A \cup C\}$) = 2/4 = 50%

confidence = support($\{A \cup C\}$)/support($\{A\}$) = 50%/75% = 66.6%

Lift of A Rule

- **Lift**($X \Rightarrow Y$) = confidence($X \Rightarrow Y$)/support(Y)

  = support($X \cup Y$)/ (support(X) * support(Y))

  = (frequency($X \cup Y$)*$|T|$) / (frequency(X)*frequency(Y))

- Lift($X \Rightarrow Y$) refers to the increase in the ratio of sale of Y when X is sold
  - Lift = 1: No association between products X and Y.
  - Lift > 1: Products X and Y are more likely to be bought together.
  - Lift < 1: The two products are unlikely to be bought together.

Example of Lift

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

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

Rule $A \Rightarrow C$:

support = support($\{A \cup C\}$) = 2/4 = 50%

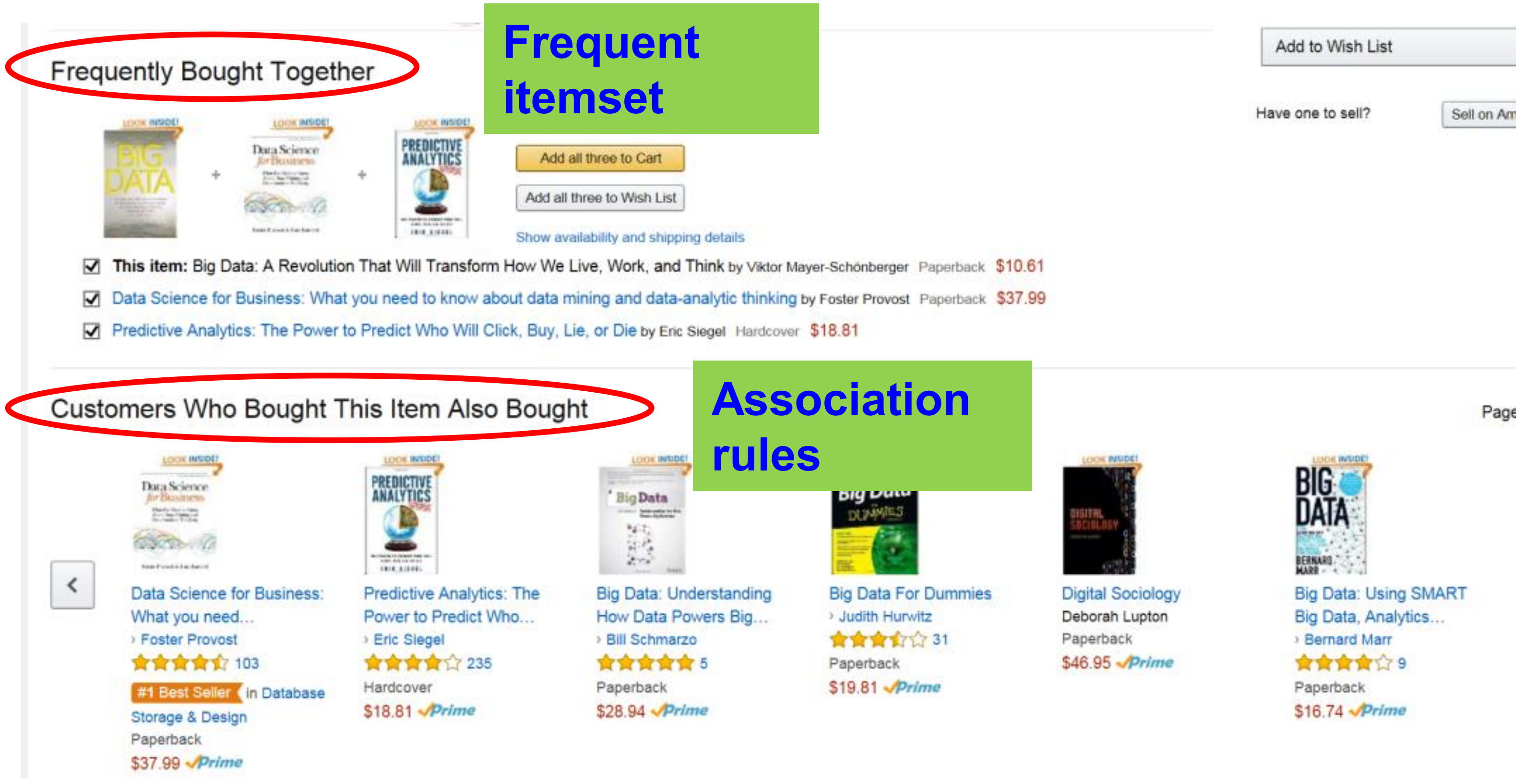
confidence = support($\{A \cup C\}$)/support($\{A\}$) = 50%/75% = 66.6%

lift = confidence($A \Rightarrow C$)/support(C) = 66.6%/50% = 1.333

Lift's meaning: the likelihood of buying a A and C *together* is 1.33 times more than the likelihood of just buying the C.

Recommendation in Amazon

- Two types of recommendation
  - 90% buyers who bought A also bought B.
  - Since you've bought A, you may also want B.



Frequently Bought Together

Frequent itemset

Add to Wish List

Have one to sell? Sell on Am

Add all three to Cart

Add all three to Wish List

Show availability and shipping details

- ☑ **This item:** Big Data: A Revolution That Will Transform How We Live, Work, and Think by Viktor Mayer-Schönberger Paperback $10.61
- ☑ Data Science for Business: What you need to know about data mining and data-analytic thinking by Foster Provost Paperback $37.99
- ☑ Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die by Eric Siegel Hardcover $18.81

Customers Who Bought This Item Also Bought

Association rules

Page

Data Science for Business: What you need…	Predictive Analytics: The Power to Predict Who…	Big Data: Understanding How Data Powers Big…	Big Data For Dummies	Digital Sociology	Big Data: Using SMART Big Data, Analytics…
Foster Provost	Eric Siegel	Bill Schmarzo	Judith Hurwitz	Deborah Lupton	Bernard Marr
★★★★☆ 103	★★★★☆ 235	★★★★★ 5	★★★½☆ 31	Paperback	★★★★☆ 9
#1 Best Seller in Database Storage & Design	Hardcover	Paperback	Paperback	$46.95 Prime	Paperback
Paperback	$18.81 Prime	$28.94 Prime	$19.81 Prime		$16.74 Prime
$37.99 Prime					

Causality *vs.* Correlation

- Causality
  - From the very first day, humans are curious about *why*.
  - With big data, it may be very hard to see the exact reasons.
- Correlation
  - Instead, we can find interesting patterns or associations of different things from big data.
  - Probability instead of certainty (not totally random).
    - Association rule mining.
- **NB**: Association rules are empiricism! What they tell may not be the true cause and effect.

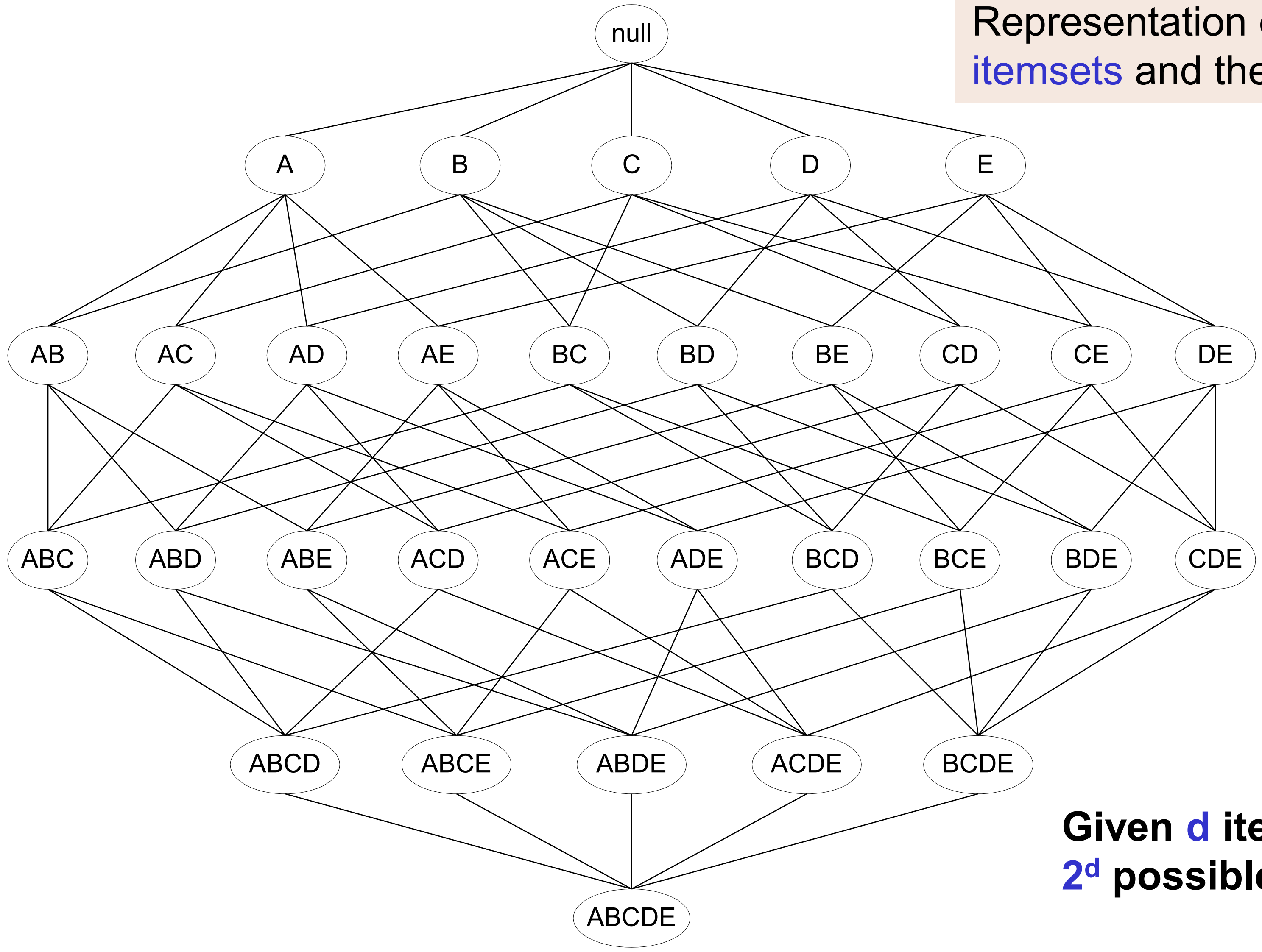
Steps of Association Rule Mining

1. Find the ***frequent itemsets***
   - The sets of items that have minimum support
   - How to do this efficiently?
2. Use the frequent itemsets to generate association rules

Mining Frequent Itemsets

- **Input**: A set of transactions **T**, over a set of items **I**
- **Output**: All itemsets with items in **I** having
  - support ≥ minsup (support threshold)
- Problem parameters:
  - N = |**T**|: number of transactions
  - d = |**I**|: number of (distinct) items
  - w: max width of a transaction
  - Number of possible itemsets: $M = 2^d$
- Scale of the problem:
  - WalMart sells 100,000 items and can store billions of baskets.
  - The Web has billions of words and many billions of pages.

The Itemset Lattice



Representation of all possible itemsets and their relationships

Given d items, there are 2^d possible itemsets

Agenda

- Problem definition
- Apriori principle, Apriori algorithm
- Deriving association rules from frequent itemsets
- Reflection on Apriori algorithm

The Apriori Principle

- Main observations: $\forall \boldsymbol{X}, \boldsymbol{Y}: \boldsymbol{X} \subseteq \boldsymbol{Y} \Rightarrow \boldsymbol{s}(\boldsymbol{X}) \geq \boldsymbol{s}(\boldsymbol{Y})$
  - If an itemset is frequent, so are its subsets
  - If an itemset is infrequent, so are its supersets
- The Apriori principle: A *subset* of a frequent itemset must also be a frequent itemset
    - E.g., if {*AB*} is a frequent itemset, both {*A*} and {*B*} must be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to *m* (*m*-itemset): Use frequent k-itemsets to explore (k+1)-itemsets

Illustration of Apriori Principle

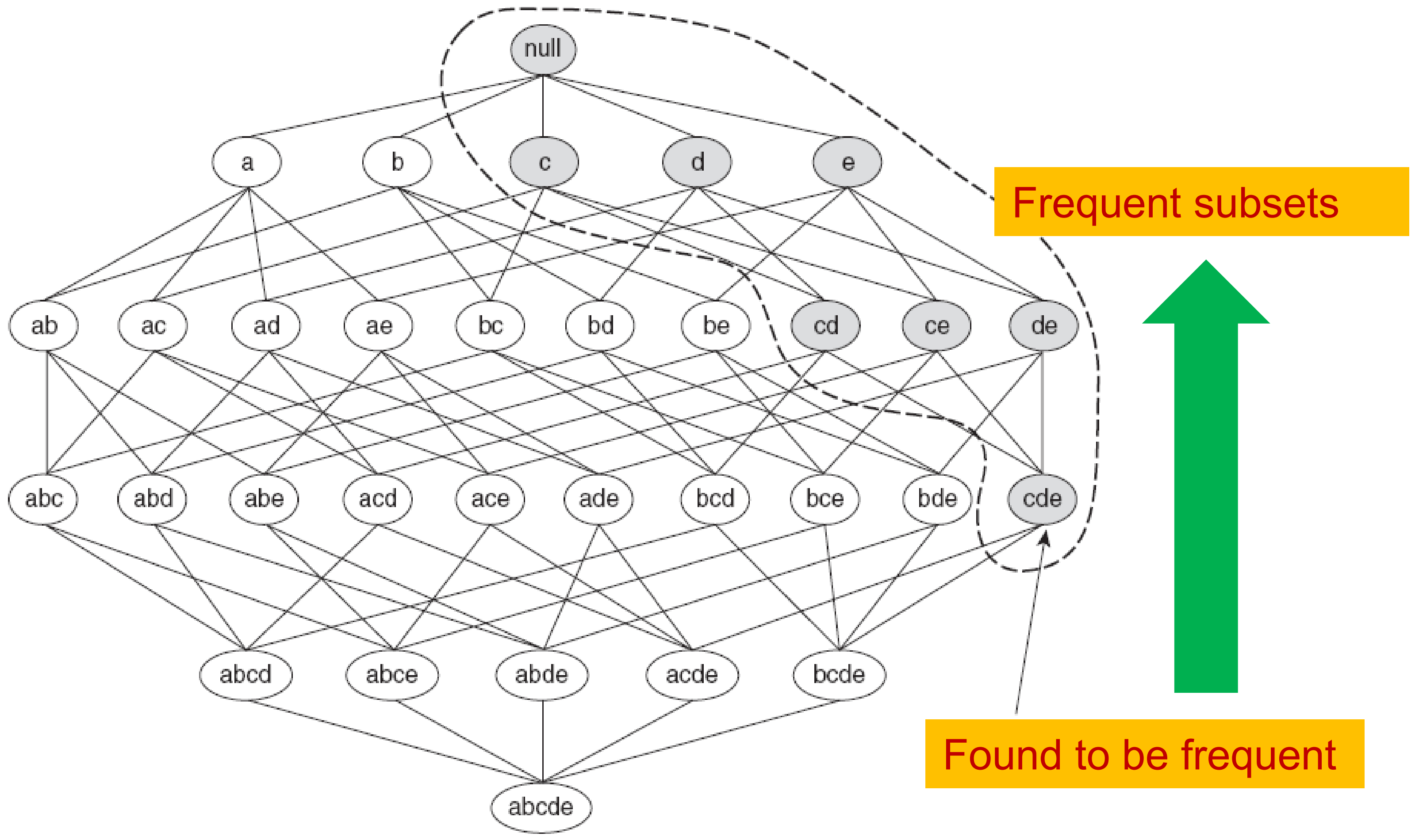
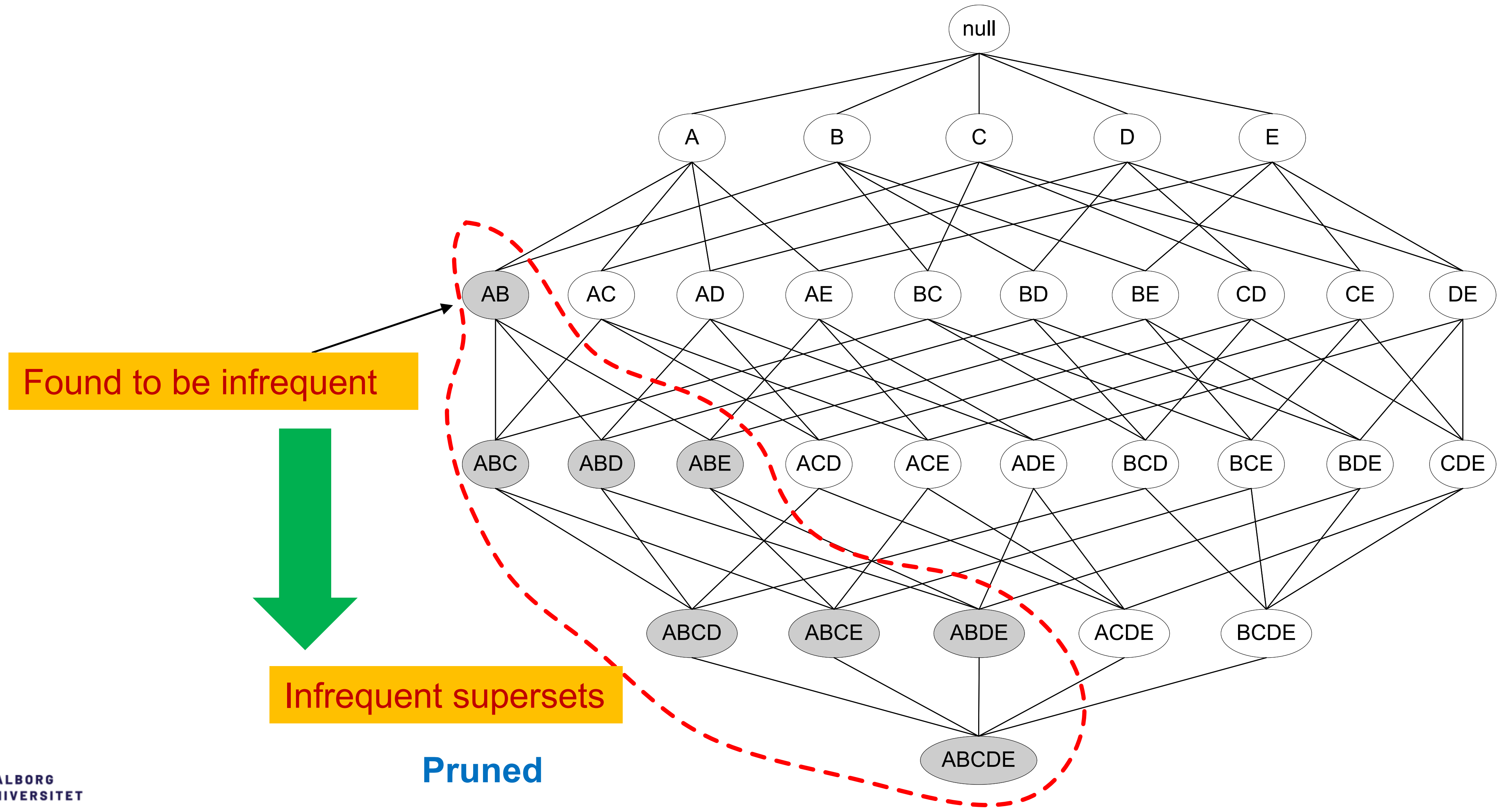


Figure 6.3. An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

AALBORG UNIVERSITET

Illustration of Apriori Principle (cont.)



Level-wise Process of Apriori Principle

Level 4 (frequent quadruples): {….}

Level 3 (frequent triplets): {ABD}, {BDF}

Level 2 (frequent pairs): {AB}, {AD}, {BD}, {BF}, {DF}

Level 1 (frequent items): {A}, {B}, {D}, {F}

Remember:

All subsets of a frequent itemset must be frequent as well

Question: Can ADF be frequent?

NO: because AF is not frequent

The Apriori Algorithm

- Notations
  - C_k: Candidate itemset of size k
  - L_k: Frequent itemset of size k
- Important steps in candidate generation
  - Prune Step: Any k-itemset that is not frequent cannot be a subset of a frequent (k+1)-itemset
  - Join Step: C_{k+1} is generated by joining L_k with itself

```
$C_1 = \{\{item_1\}, \ldots, \{item_N\}\}$;
for ($k = 1$; $L_k \neq \emptyset$; $k$++)
    for each transaction $t$ in transaction table $T$
        increment the count of all candidates in $C_k$ that are contained in $t$
    $L_k$ = candidates in $C_k$ with min_support (frequent)
    $C_{k+1}$ = candidates generated from $L_k$;
return $\cup_k L_k$;
```

Special self-join!

The Apriori Algorithm Example (1)

Trans. Table T

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

min_sup=2 (or 50%)

$\xrightarrow{\text{Scan T}}$ C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

$\longrightarrow$ L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

The Apriori Algorithm Example (2)

Trans. Table T

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

min_sup=2 (or 50%)

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan T

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

The Apriori Algorithm Example (3)

Trans. Table T

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

min_sup=2 (or 50%)

C_3

itemset
{2 3 5}

$\xrightarrow{\text{Scan T}}$

L_3

itemset	sup
{2 3 5}	2

The Apriori Algorithm Example (4)

Trans. Table T

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

min_sup=2 (or 50%)

The result of frequent itemsets

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

itemset	sup
{2 3 5}	2

$L_1 \cup L_2 \cup L_3$

Candidates Generation

- Suppose the items in L_k are listed in an order
- Step 1: self-joining L_k to get C_{k+1} (In SQL)

```
INSERT INTO C_{k+1}
SELECT p.item_1, p.item_2, …, p.item_k, q.item_k
FROM L_k p, L_k q
WHERE p.item_1=q.item_1, …, p.item_{k-1}=q.item_{k-1}, p.item_k < q.item_k
```

- Step 2: pruning frequent itemsets in C_{k+1}

```
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 $C_{k+1}$
```

AALBORG UNIVERSITET

The Previous Example

Trans. Table T

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

min_sup=2 (or 50%)

We only need to match {2 3} with {2 5}

L_2

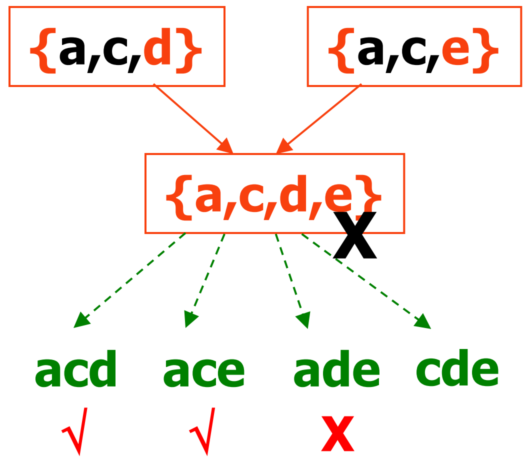
itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Example of Candidates Generation

- L_3={*abc, abd, acd, ace, bcd*}
- Self-joining: $L_3 \bowtie L_3$
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
  - No need to match other pairs
- Pruning:
  - *acde* is removed because *ade* is not in L_3
- C_4={*abcd*}
  - Scanning transaction table *T* is still needed to get the frequencies for items in C_4 *(to decide the correct* L_4*)*



Agenda

- Problem definition
- Apriori principle, Apriori algorithm
- Deriving association rules from frequent itemsets
- Reflection on Apriori algorithm

Generating Association Rules from Frequent Itemsets

- Assume that we have discovered the frequent itemsets and their support
- How do we generate association rules?
- Frequent itemsets:

{1}	2
{2}	3
{3}	3
{5}	3
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2
{2,3,5}	2

$l \rightarrow$ {2,3,5}

Not a transaction table!

?

- For each frequent itemset *l*, find all nonempty subsets *s*.
- For each *s*, generate rule ***s*** $\Rightarrow$ ***l-s***, if sup(l)/sup(s) $\geq$ min_conf

Example: $l = \{2,3,5\}$, min_conf = 75%

$\{2,3\} \Rightarrow \{5\}$	2/2=100%	√
$\{2,5\} \Rightarrow \{3\}$	2/3=66.6%	X
$\{3,5\} \Rightarrow \{2\}$	2/2=100%	√

do the rest as an exercise

AALBORG UNIVERSITET

Association Rules in Jupyter Notebook

- Library **mlxtend**
  - To install the library: pip install mlxtend in Anaconda Prompt
  - from mlxtend.frequent_patterns import apriori: frequent itemsets
  - from mlxtend.frequent_patterns import association_rules: rules

- Exercises on real data
  - store_data.csv (in Moodle)
  - (7501, 20)
    - 7501 transactions, each having at most 20 items

Agenda

- Problem definition
- Apriori principle, Apriori algorithm
- Deriving association rules from frequent itemsets
- Reflection on Apriori algorithm

Performance Bottlenecks of Apriori

- Is Apriori fast enough?
- The core of the Apriori algorithm:
  - Use frequent *k*-itemsets to generate **candidate** frequent (*k+1*)-itemsets
  - Use full table scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of Apriori: **candidate generation**
  - Huge candidate sets:
    - A 10^4-sized frequent 1-itemset will generate 10^7 candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.
  - Multiple scans of database table:
    - Needs (*n +1*) scans, *n* is the length of the longest pattern

Methods to Improve Apriori's Efficiency

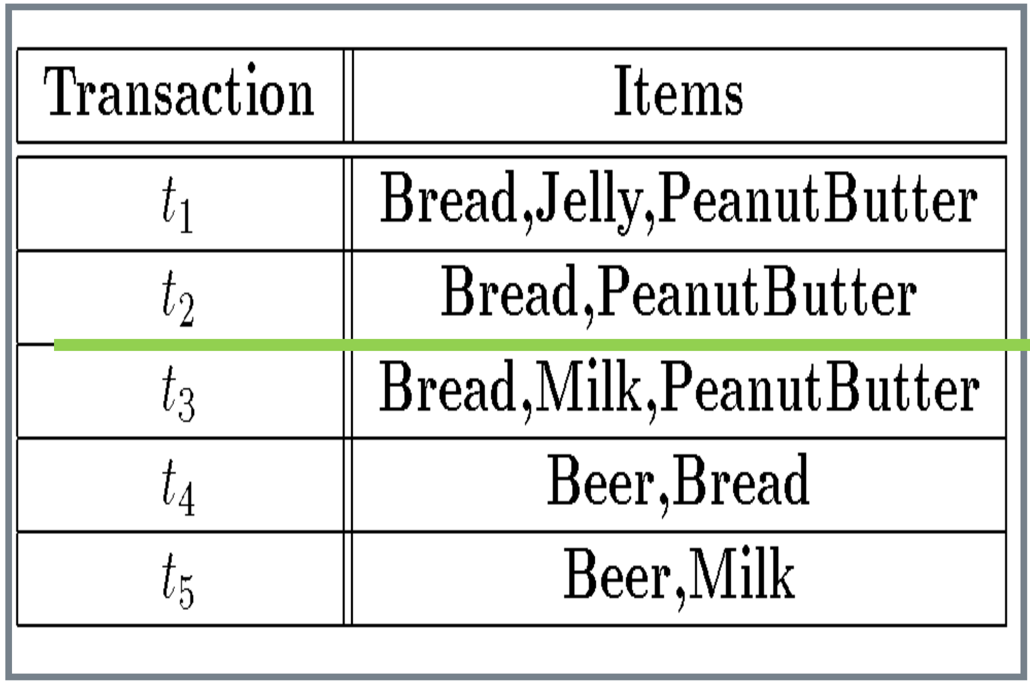
- Transaction reduction
  - A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning
  - Any itemset that is potentially frequent in transaction table T must be frequent in at least one of the partitions of T.

Partitioning

- Divide transaction table T into partitions T1, T2, …, Tp such that each Ti fits in the main memory
- Apply Apriori to each partition
- Any frequent itemset must be frequent in at least one partition

```
1. Divide $T$ into partitions $T^1, T^2, \ldots, T^p$;
2. For $i = 1$ to $p$ do
3.     $L^i$ = Apriori($T^i$);
4. $C = L^1 \cup \ldots \cup L^p$;
5. Count $C$ on $T$ to generate $L$;
```

Partitioning Example (1)



T_1

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

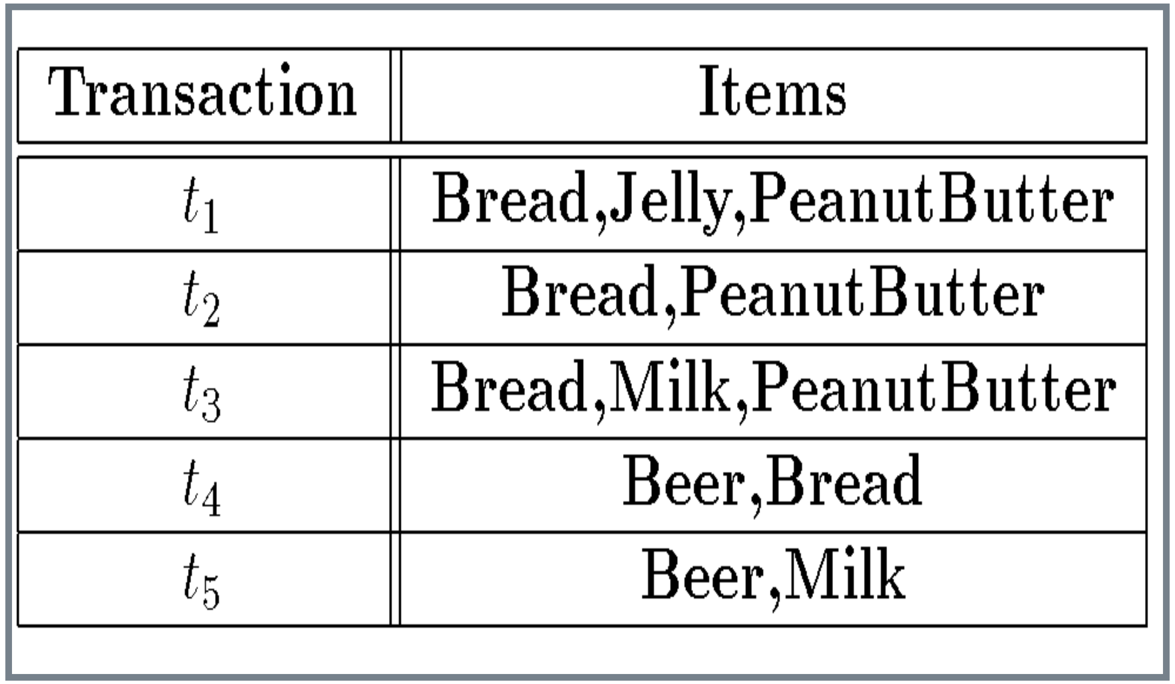
T_2

min_support = 10%

$L(T_1)$ = {{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}

$L(T_2)$ = {{Bread}, {Milk}, {PeanutButter}, {Bread,Milk}, {Bread,PeanutButter}, {Milk, PeanutButter}, {Bread,Milk,PeanutButter}, {Beer}, {Beer,Bread}, {Beer,Milk}}

Partitioning Example (2)



Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

min_support = 10%

$L(T_1)$ = {{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}

$L(T_2)$ = {{Bread}, {Milk}, {PeanutButter}, {Bread,Milk}, {Bread,PeanutButter}, {Milk, PeanutButter}, {Bread,Milk,PeanutButter}, {Beer}, {Beer,Bread}, {Beer,Milk}}

$C = L(T_1) \cup L(T_2)$

Count itemsets in *C* with respect to *T*, and prune infrequent ones.

Partitioning's Pros and Cons

- Advantages:
  - It adapts to available main memory
  - It can be easily parallelized
    - Maximum number of database table scans is two (why?)
      - One for partitioning the transaction table, and one for the final counting

- Disadvantages:
  - May have many candidates for the second scan
  - A countermeasure: associate the frequency to each itemset in each partition, and the final global counting can be a simple aggregation.

More Efficient Approach: FP-tree

- Using FP-tree for finding frequent items
- Compress a large database table into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database table scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!
- FP-growth: mining frequent patterns without candidate generation

Summary

- Association rule definition
  - Support, confidence, lift and association rule
  - Frequent itemsets
  - Steps for association rule mining
- Apriori algorithm
- Deriving association rules from frequent itemsets
- Criticism on Apriori

Exercises

1. Refer to the transaction table to the right. Say sup(ab)=100
   - Determine the possible values of sup(a)
     - Conclusion: sup(a)___100
     - *Hint*: Is it possible that sup(a)=70? Why?
   - Determine the possible values of sup(abc)
     - Conclusion: sup(abc)___100
     - *Hint*: Is it possible that sup(abc)=120? Why?
2. Slides 33 (Hands-on, optional)

Choose either "≤" or "≥"

Transaction table (1000 rows)

TID	Items
1	a,b,c
2	a,c
3	b,e,f
…	……

AALBORG UNIVERSITET

Readings

- Mandatory reading
  - Jiawei Han, Micheline Kamber and Jian Pei. Data Mining: Concepts and Techniques (3rd edition), Elsevier Science Ltd, 2011.
    - Chapter 6

- Further readings
  - Rakesh Agrawal, Ramakrishnan Srikant: Fast Algorithms for Mining Association Rules in Large Databases. VLDB 1994: 487-499
  - Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. SIGMOD 2000: 1-12

- Acknowledgment: Slides are from
  - Margaret H. Dunham (Data Mining: Introductory and Advanced Topics, Prentice Hall, 2002)
  - The HKP textbook
  - Man Lung Yiu and Panagiotis Karras

Readings for Coding

- Mandatory readings
  - Association Rule: https://www.geeksforgeeks.org/association-rule/?ref=lbp
  - Frequent Itemsets: https://www.geeksforgeeks.org/frequent-item-set-in-data-set-association-rule-mining/?ref=lbp
  - Apriori Algorithm: https://www.geeksforgeeks.org/apriori-algorithm/?ref=lbp
- Further readings
  - Documentation of mlxtend's frequent
    - http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/
    - http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/
  - Tutorials
    - https://www.geeksforgeeks.org/implementing-apriori-algorithm-in-python/
    - https://www.kaggle.com/code/annettecatherinepaul/apriori-algorithm-association-rule-mining
    - https://towardsdatascience.com/understand-and-build-fp-growth-algorithm-in-python-d8b989bab342 (FP-Growth, advanced)